NPHASE-PSU, V3.1

Theory and User's Manual

Robert F. Kunz

Rev. 1.0, 12/28/05
Rev 1.1, 1/27/06
Rev 1.2, 8/11/06
Rev 1.3, 10/7/06
Rev. 1.4, 10/18/06
Rev. 1.5, 10/29/06
Rev. 1.6, 12/8/06
Rev 1.7, 12/15/06

Table of Contents

Revision History

V1.6: Updates subsequent to NRC delivery of NPHASE-PSU V3.1 in December 2006:
i) all equations converted to MS MathType, numbered and cross-referenced globally

V1.7: Updates subsequent to NRC delivery of NPHASE-PSU V3.1 in December 2006:
i) control commands included for every keyword available in V3.1
ii) fully functional version of multiphase 5415 tutorial

Executive Summary

This report summarizes technology developed under USNRC Contract NRC-04-03-048, and DARPA Contract H0011-04-C-0011 This document represents the first formal documentation of the NPHASE-PSU computer code. It is being delivered along with the software to the USNRC and DARPA in 2006.

Significant upgrades to the NPHASE-PSU have been made since the first delivery of draft documentation to USNRC in April, 2006. These include a much lighter, faster and memory efficient face based front end, support for arbitrary polyhedra in front end, flow-solver and back-end, a generalized homogeneous multiphase capability, and several two-fluid modelling and algorithmic elements.

Overview of NPHASE-PSU

NPHASE is a CFD code developed by Robert Kunz at the Penn State University Applied Research Laboratory (PSU-ARL) and Steve Antal at Rensselaer Polytechnic Institute (RPI). The code has been under development since 1998. Since NPHASE Version 2.0 was established in 2000, two separate versions of the code have been developed independently by Kunz and Antal. This document applies to the version developed by Kunz at ARL Penn State, NPHASE-PSU is distributed for free to two sponsoring US government agencies: the USNRC and DARPA. NPHASE-PSU V3.1 (document Rev 1.6) includes recent major updates related to homogeneous and two-fluid multiphase modelling and algorithmics, pre-and post-processing, and support for arbitrary polyhedra.

NPHASE-PSU is not a commercial CFD code, nor is it used for commercial consulting. The mission of the software developer is to support government and industrial sponsors of programs related to PSU-ARL's core research activities.

NPHASE-PSU is written in standard ANSI-C, and compiles under (at least) the open-source GNU C compiler, gcc. NPHASE-PSU refers to the CFD code itself, but employs several front-end and back-end processing tools for domain decomposition and reassembly, grid readers for standard COTS formats, pointer topology construction, and writers to standard postprocessing software file formats. These processing codes are written in FORTRAN 77/90 and ANSI-C. Accordingly, if the user wishes to modify these front/back-end programs they must also have access to a FORTRAN 90 compiler. NPHASE-PSU also requires several open source software libraries including MPI, PETSC, METIS and SUGGAR/DirtLib each of which must be installed with NPHASE-PSU on the system. Although the code has in the past been installed on Windows and SGI systems, the present delivered version, V3.1, is verified to install and run only on desktops and clusters running LINUX.

NPHASE-PSU has the following characteristics, features, and capabilities:

- Arbitrary number of fields and/or species, where different species are assumed to be in dynamic and thermodynamic equilibrium, and different fields are not (i.e. have different velocities and enthalpies). Mass fraction and volume fraction transport options are avilable for species/field transport.
- Numerous interfacial mass, momentum, energy and turbulence exchange models associated with multiphase flow simulations.

- 3D unstructured: Arbitrary polyhedral formulation with front-back ends supporting 4 standard element types: tetrahedral, hexahedra, pyramids, prisms, as well as completely arbirtrary element types (n-faced polyhera). This features is new as of V3.0.
- Overset mesh capability, utilizing open source Suggar and DirtLib software.
- Moving and deforming mesh capability (Geometric Conservation Law satisfying).
- Fully matrix level parallelized using MPI and domain decomposition.
- METIS used for domain decomposition embedded in front end.
- PETSC and simple point linear equation solvers.
- All-Mach number formulation: incompressible, weakly compressible, strongly compressible flows (partial capability). Isothermal, Boussinesq and perfect gas single-phase compressible state relations are available.
- Segregated pressure based algorithm and CPE algorithm for multiphase flow, partially capable fully coupled formulation.
- Face based finite volume scheme: 1^{st} through 3^{rd} order accurate convection discretization schemes, 2^{nd} order accurate viscous term discretization.
- 1^{st} and 2^{nd} order, dual time based temporally accurate formulation.
- Several low and high Reynolds number form 2-equation, and v2f turbulence models.
- Structural mechanics coupling to NASTRAN.
- Radiation heat transfer coupling to RADTHERM.
- Numerous "specialty" face and volume elements (conducting solid regions, porous regions, various quasi-1D conjugate heat transfer boundaries).
- Full turbomachinery capability (rotating and non-rotating reference frames) including rotor-stator interaction and body force modeling.
- "Light" face based file formats supported in front end.
- ENSIGHT file format supported in back end.
- Coded purely in ANSI-C, with some front and back end utilities coded in F77, F90.

Partial development (features that are not fully implemented but are in source code in various stages of completion):

- Non-isotropic mesh adaption
- Full Reynolds Stress modeling
- Conformation tensor transport
- VOF for discrete interfaces
- 6DOF dynamics
- Fully coupled parallel algorithm

NPHASE-PSU has been applied to and validated against a broad range of complex single-phase and multiphase configurations including:

- Gas-particle flows through a branching pipe junctions and human lung geometries
- Bubble column reactors
- Full-annulus rotor-stator pump and turbine stage analyses, including rotor-stator interactions

- High Reynolds number submarine configurations at a range of angles of attack
- Power plant cooling ponds
- Microbubble drag reduction applications
- Geometrically complex UUV (MRUUV) and SEAL deliverivery vehicles (ASDS)
- Several surface ship configurations (5415, Athena)
- High speed maritime lifting pod
- Micro-flows of biological cell systems
- Numerous multiphase flows of relevance to the NRC (thermally driven counter-current reactor flows, 2-phase duct and pipe flows)
- DES simulations of urban/atmospheric dispersion
- Bubbly surface ship wakes
- Thermal management of tank engine compartment
- Thermal management of eco-friendly structures.

Documentation of many of these cases appears in Kunz et al. (2001, 2003, 2007) or can be obtained from the author.

NPHASE-PSU Theory Manual

Governing Equations

The single-pressure ensemble averaged continuity and momentum equations are cast in conservation law form as:

$$\frac{\partial \alpha^k \rho^k}{\partial t} + \frac{\partial \alpha^k \rho^k u_j^k}{\partial x_j} = \sum_{k \neq l} \left(\Gamma^{lk} - \Gamma^{kl} \right) \tag{1}$$

$$\frac{\partial \alpha^k \rho^k u_i^k}{\partial t} + \frac{\partial \alpha^k \rho^k u_i^k u_j^k}{\partial x_j} = -\alpha^k \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\alpha^k \mu_t^k \left(\frac{\partial u_i^k}{\partial x_j} + \frac{\partial u_j^k}{\partial x_i} \right) \right] + \rho^k \alpha^k g_i + M_i^{kl} + \sum_{k \neq l} \left(D^{kl}\left[u_i^l - u_i^k \right] + \Gamma^{lk} u_i^l - \Gamma^{kl} u_i^k \right) \tag{2}$$

Superscripts k and l designate donor and receptor fields for mass transfer (Γ^{kl}), and drag (D^{kl}) and non-drag (M_i^{kl}) interfacial forces. In general each field, k, will have a different density, volume fraction, velocity and viscosity. For single phase flow, equations (1)-(2) reduce to:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_j}{\partial x_j} = 0 \tag{3}$$

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right] + \rho g_i \tag{4}$$

For homogeneous multiphase flow, it is assumed that the fields are in dynamic and thermodynamic equilibrium, and equations (1)-(2) reduce to:

$$\frac{\partial \alpha^k \rho^k}{\partial t} + \frac{\partial \alpha^k \rho^k u_j^k}{\partial x_j} = \sum_{k \neq l} \left(\Gamma^{lk} - \Gamma^{kl} \right) \tag{5}$$

$$\frac{\partial \rho^m u_i^m}{\partial t} + \frac{\partial \rho^m u_i^m u_j^m}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu_t^k \left(\frac{\partial u_i^m}{\partial x_j} + \frac{\partial u_j^m}{\partial x_i} \right) \right] + \rho^m g_i \tag{6}$$

where the set of momentum equations is reduced to a single equation for the mixture. Superscript m represents mixture quantities. In equations (1)-(6), a high Reynolds number form viscous term is assumed with dilatation and turbulence energy terms neglected (although these terms are available in NPHASE-PSU). Energy and turbulence equations are considered below.

Physical Modeling

Generalize Field Transport

The generalized n-field formulation in equations (1)-(2) can be applied to non-equilibrium multiphase flows in two ways. The more fundamental approach involves solving mass and momentum equations for each field that is present. For example, in the context of disperse bubbly flows, one could solve a single continuous liquid field and a

number of disperse fields, "binned" by size. In this approach each bubble field exchanges momentum with the continuous field through drag and non-drag interfacial forces which depend in magnitude on the local interfacial area density of that field, $A_{int}=6\alpha^{gas}/D_b$ (for spherical bubbles). This approach was used in our earlier work [Kunz et al., (2003, 2007)], where up to 11 bubble fields were solved.

Interfacial Area Density Transport

An alternative is to solve a single mass and momentum equation for each *phase* that is present and to accommodate the variation in dynamics due to phase interface evolution by modelling and solving for interfacial area transport. For example, in the context of disperse bubbly flows, a single gas field continuity and momentum equation would be solved, and an interfacial area density transport (IADT) equation would also be solved to determine a local mean characteristic diameter for the bubbles. This approach significantly reduces the model's CPU requirements compared to solving an (N+1)-field system (N bubble fields). The numerical complexity associated with interfield transfer terms is also reduced considerably.

Since mass transfer can be fully accommodated in the context of IADT (details presented below), the physical appropriateness of employing IADT rests on whether the interface dynamics can be sufficiently captured using a single local mean inter-phase interfacial area, with an assumed/modeled distribution of characteristic size/shape about that mean. This is demonstrated to be the case for an example calculation below. Currently in NPHASE-PSU, a generalized IADT formulation is available, with physical models in place to accommodate disperse bubbly flows related to Microbubble Drag Reduction (hereafter MBDR). In this context mass transfer corresponds to coalescence and breakup (between bubbles of different sizes). The IADT formulation in NPHASE-PSU is presented here, in that context, although any interface evolution (e.g., annular flow, droplet laden gas flows) can be modeled through addition of subroutines corresponding to those available for the disperse bubbly flow models currently available in V3.1.

Following Hibiki et al. (2001), the IADT equation with source terms for breakup and coalescence can be written:

$$\frac{\partial a_i}{\partial t}+\frac{\partial\left(a_i u_{g,j}\right)}{\partial x_j}=\Phi_B+\Phi_C \tag{7}$$

where a_i is the interfacial area density, $u_{g,j}$ are the gas phase velocity components, and Φ_b and Φ_c are source terms for breakup and coalescence, respectively. The interfacial area density is defined as:

$$a_i=\frac{6\alpha_g}{\bar{D}} \tag{8}$$

where α_g is the volume fraction of the gas phase and $\bar{D}$ is the mean bubble diameter. The source terms are rates of change of interfacial area concentration, written as:

$$\Phi_B=\frac{1}{3\psi}\left(\frac{\alpha_g}{a_i}\right)^2\phi_b,\Phi_C=\frac{-1}{3\psi}\left(\frac{\alpha_g}{a_i}\right)^2\phi_c \tag{9}$$

where ϕ_b and ϕ_c are the rates of change of bubble number density ($1/m^3s$) due to breakup and coalescence, respectively. The factor ψ depends on the bubble shape, here taken as spherical, so $\psi=1/(36\pi)$. The particular models used for breakup and coalescence for MBDR are presented below.

Figure 1 illustrates that the dynamics of MBDR can be sufficiently captured using a single local mean gas-liquid interfacial area, with an assumed/modeled distribution of bubble size about that mean. Three MBDR cases are considered, corresponding to three gas injection rates at injector plates near the leading edge of a very high Reynolds number flat plate flow (see "HIPLATE" tutorial below). First, each case was run with three bubble fields using an approximation to the experimentally measured bubble size distribution. Then each case was run using a single gas field and interfacial area density as described above. For these comparisons no coalescence or breakup was incorporated so as to isolate the effect of the different interfacial dynamics modeling approaches. Details of the HIPLATE simulations are provided below, but Figure 1 serves to illustrate that incorporating interfacial area density has only a small impact on accuracy of drag reduction and bubble velocity predictions for MBDR.

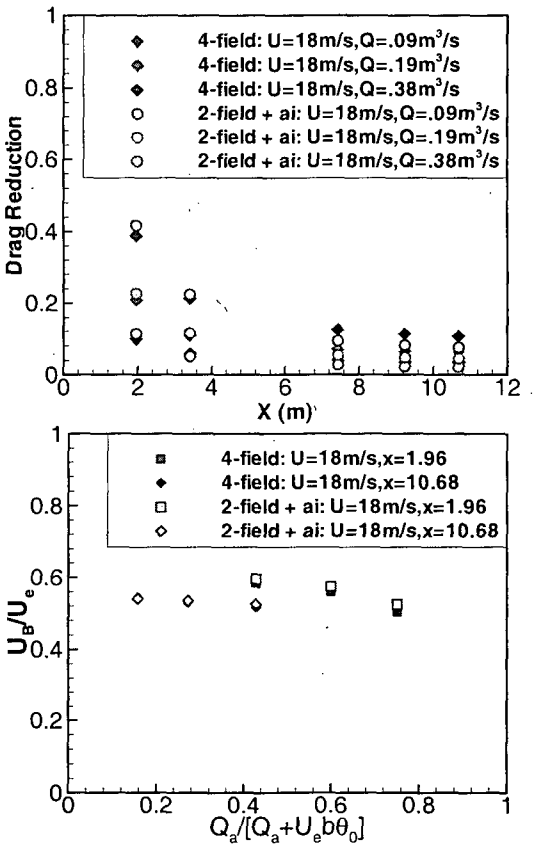

Figure 1. Comparison of 2-field and 4-field simulations for U_∞=18 m/s HIPLATE cases. (top) Drag reduction vs. x, (bottom) Normalized bubble velocity vs. normalized flow rate.

Interface Dynamics

Overview

The structure of NPHASE-PSU supports arbitrary forms for drag (D^{kl}) and non-drag (M_i^{kl}) interfacial dynamics models that appear in equation (2). The focus of two-fluid NPHASE-PSU research performed to date has been in the context of disperse bubbly flows (where the single continuous field is liquid) and disperse particle flows (where the single continuous field is gaseous). Accordingly the physical model set that currently resides within V3.1 are appropriate for these interface dynamics.

A suite of bubble dynamics models have been developed, adapted from the open literature, and calibrated over the course of the development of NPHASE-PSU. These interfacial force models are summarized here. These models have been used in the context of a full-up n-bubble-field formulation where the bubble diameters that appear and hence the implied interfacial area between that field and the liquid are unique to and representative of that field. As indicated in the previous section, these models are also implemented in the context of a single gas field represented by a mean bubble diameter and attendant interfacial area density, which is transported and evolved with the flow.

Drag

In the context of particles, a conventional corrected Stokes drag law is incorporate

$$D^{kl} = \frac{1}{8}\rho^{gas} C_D \left|u_i^l - u_i^k\right| a_i, a_i = \frac{6\alpha^{gas}}{D_p}, C_D = \frac{24}{Re_p} f_D(Re_p) \tag{10}$$

where the local particle Reynolds number is $Re_p = \rho^{gas}\left|\underline{V}^{rel}\right| D_p / \mu^{gas}$. The solid particle drag-law correction used [Loth (2000), for example] is:

$$\begin{aligned} f_D &= 1. + 0.1875\,Re_p && \text{for } Re_p \le 1 \\ f_D &= 1. + 0.1935\,Re_p^{.6305} && \text{for } 1 < Re_p \le 285 \\ f_D &= 1. + 0.015\,Re_p + 0.2283\,Re_p^{.427} && \text{for } 285 < Re_p \le 2000 \\ f_D &= 0.44\,Re_p / 24. && \text{for } 2000 < Re_p \le 3.5\text{x}10^5 \end{aligned} \tag{11}$$

In the context of bubbles, drag models have been implemented for spherical bubbles in seawater, clean fresh water and contaminated (tap) water. Again, a corrected Stokes drag law is employed:

$$D^{kl} = \frac{1}{8}\rho^{liq} C_D \left|u_i^l - u_i^k\right| a_i, a_i = \frac{6\alpha^{liq}}{D_b}, C_D = \frac{24}{Re_b} f_D(Re_b) \tag{12}$$

where the local bubble Reynolds number is $Re_b = \rho^{liq}\left|\underline{V}^{rel}\right| D_b / \mu^m$.

For fresh water without impurities, the drag-law correction [Loth (2000), for example] is:

$$\begin{aligned} f_D &= 1. + 0.1875\,Re_b && \text{for } Re_b \le 0.1 \\ f_D &= 1. + 0.0565\,Re_b^{.525} && \text{for } 0.1 < Re_b \le 500 \end{aligned} \tag{13}$$

For contaminated (tap) water, the drag-law correction for solid spheres equation (11), is used. For seawater, a drag-law correction due to Detch (1991) is available.

In addition to water purity, locally high gas volume fraction and bubble deformation can influence the drag, so corrections to the spherical bubble, disperse flow models in equations (11) and (13) may be appropriate. For uniformly disperse flows, an increased drag coefficient is appropriate [Richardson-Zaki (1954), for example], and for flows where gas structures are streamlined (bubble columns, sheets) a reduced drag coefficient is appropriate. This latter effect likely is important in the near injector region of MBDR flows, where application of the standard disperse flow model gives rise to too much local drag, thereby inhibiting the penetration of the injected gas into the boundary layer. This observation became clear in the course of the HIPLATE validation studies, where a significant defect in measured bubble velocity could not be obtained unless a "cluster" drag model was incorporated. Specifically, a model proposed by Johansen and Boysan (1988) has been adapted to an Eulerian framework:

$$C_D = C_{D0}\left(1 - 1.54\left[\mathrm{MIN}\left(.5157, \alpha^{gas}\right)\right]^{2/3}\right) \tag{14}$$

where C_{D0} is the original drag coefficient in equations (11) or (13), α^{gas} is the total gas volume fraction and the MIN function is provided to ensure that the corrected drag coefficient does not drop to below 1% of the uncorrected value. The importance of incorporating such a cluster drag form is demonstrated in Kunz et al. (2007).

Virtual Mass

Virtual mass is modeled following Lahey and Drew (2000):

$$\underline{M}_{VM}^{liq-gas} = \alpha^{gas}\rho^{liq}C_{VM}\left[\frac{D\underline{V}^{gas}}{Dt} - \frac{D\underline{V}^{liq}}{Dt}\right] \tag{15}$$

Lift

The lift model employed in the NPHASE-PSU also follows Lahey and Drew (2000):

$$\underline{M}_{LIFT}^{liq-gas} = \alpha^{gas}\rho^{liq}C_l\underline{V}^{rel} \times \nabla \times \underline{V}^{liq} \tag{16}$$

Wall Lift

An empirical turbulent near-wall bubble lift force has been implemented based on the formulation of Kawamura and Yoshiba (2004). This force can be thought of as a repulsive force due to wall collisions. The form of the wall-lift force used is:

$$\begin{aligned} F_{WL} &= C_{WL}\left(\pi D_b^3/6\right)\rho^{liq}\left(k/D_b\right)F_{damp} \\ F_{damp} &= 0.5\left[1 - \tanh\left(y_{wall}/D_b - 1.5\right)\right] \end{aligned} \tag{17}$$

where F_{damp} decays the force to zero away from the wall and the model constant used here, $C_{WL} = 0.012/\sqrt{1 + St_k}$, is significantly smaller than that proposed by Kawamura and Yoshiba. The Stokes number is defined as $St_k = D_B^2\rho^{liq}\varepsilon/(18k\mu_m)$.

Turbulence Dispersion

The homogeneous turbulence dispersion model is implemented in the framework of the Carrica, et al. (1999) gradient diffusion force model. The general expression for the dispersive force per unit volume (N/m^3) may be written as:

$$\underline{M}_i^{k,TD} = -\rho^{liq} \frac{\nu_t}{Sc} C_{TD} \frac{\partial \alpha^k}{\partial x_i} \tag{18}$$

where C_{TD} is the turbulent dispersion coefficient (units s^{-1}). For the Carrica, et al. (1999) model, C_{TD} is defined as:

$$C_{TD} = \frac{3}{8} \frac{C_D}{R_b} \left| \underline{u}_{rel}^k \right| \tag{19}$$

where $\underline{u}_{rel}^k$ is the relative velocity between the continuous phase and disperse phase "k".

At the high gas volume fractions, dispersion is enhanced by collisions among bubbles. A new dispersion model has been developed, based on the collision frequency from the Prince-Blanch (1990) coalescence model. This dispersion mechanism is used in addition to one of the homogeneous turbulence dispersion models discussed above. Since DNS computations [Maxey, et al. (2005)] show a significant effect of collision on dispersion for high gas volume fractions, a heuristic dispersion model based on the bubble collision rate has been formulated. The collision-induced dispersion model is implemented in the framework of the [Carrica, et al. (1999)] gradient diffusion force model, equation (18).

We assume the dispersion model coefficient is an unknown function of the "dispersive collision rate", which excludes bubbles that coalesce. To properly formulate the coefficient relationship, the collision rate must be normalized. For that purpose we choose a turbulent characteristic bubble response time (τ_{BC}^j) defined as:

$$\tau_{BC}^j = \frac{4}{3} \frac{D_j}{C_D \left| \underline{u}_{rel}^j \right|} \tag{20}$$

where *k* is the turbulent kinetic energy. Note that this is the bubble response time normally used to define the Stokes number,

$$St = \frac{\tau_{BC}^j}{\tau_c} \tag{21}$$

Alternative characteristic times were evaluated with some success; however, the above relation is a reasonable choice with physical basis.

The normalized dispersive collision rate (θ_{ij}^{TD}) for bubbles "i" and "j" with an equivalent volume V_{ij} is written as:

$$\bar{\theta}_{ij}^{TD} = \theta_{ij}^T V_{ij} \left(1 - \lambda_{ij}\right) \tau_{BC}$$
$$V_{ij} = \left(V_i + V_j\right)/2 \tag{22}$$

The turbulent dispersion coefficient (for a bubble "j") $C_{TD,j}^{coll}$ is chosen to be proportional to the square root of the dispersive collision rate (normalized by a representative time scale, τ_{BC}^j).

$$C_{TD}^{coll} = \frac{\hat{C}_{TD}}{\tau_{BC}} \sum_i \left[\overline{\theta}_{ij}^{TD}\right]^{1/2} \quad (23)$$

where $\hat{C}_{TD}$ is a constant to be determined. Note that the square root is chosen to obtain a consistent relation with the collisional pressure identified by Maxey, et al. (2005). The Brown DNS calculations confirmed the functional dependence of collisional pressure on volume fraction.

Further modifications to the dispersion model are required to treat other conditions, especially limiting cases with high gas volume fraction. A heuristic model as been implemented for the dispersion models and bubble lift model.

The dispersion in NPHASE-PSU is modeled by summing the two contributions discussed above (equations (19) and (23)), i.e.,

$$\underline{M}_i^{k,TD} = -\rho^{liq} \frac{\nu_t}{Sc} \left(C_{TD} + C_{TD}^{coll}\right) \frac{\partial \alpha^k}{\partial x_i} \quad (24)$$

In general this relation applies to each bubble field "k".

Breakup and Coalescence - Multi-Bubble-Field Formulation

Overview

A general formulation for discrete bubble size distributions based on the approach of Kumar and Ramkrishna (1996) has been implemented in NPHASE-PSU. The approach allows one to rigorously conserve two functions of the bubble distribution function (or kernel) regardless of the discrete bubble sizes (bins) selected. There is a unique formulation for coalescence and breakup of bubbles. In both cases we have chosen to conserve volume moments of the bubble number density distribution function, *n(v,t)*, i.e.,

$$M_\mu = \int_0^\infty v^\mu n(v,t)\, dv \quad (25)$$

where *v* denotes the bubble volume and *t* is time. Of course, the distribution function is a function of spatial location as well. We have chosen the zero-th (μ=0) and first (μ=1) moments at present, though the coding permits arbitrary moments to be conserved. The rational for this choice is the conservation of the number of bubbles and the volume of bubbles during the coalescence and breakup process. Though the interfacial area of the bubbles is an important quantity in two-phase bubbly flows, it is not conserved in general. The correct interfacial area will be preserved by conserving the number of bubbles and the volume of the bubbles.

It should be noted that other investigators [e.g., Carrica, et al (1999)] have used formulations based on bubble mass, since in cases with significant gas compressibility the bubble mass is conserved while the volume changes (in the absence of either coalescence or breakup). The present implementation of the Kumar-Ramkrishna scheme in NPHASE-PSU easily permits the use of bubble mass as the bubble size metric rather than bubble volume, if necessary.

Prince-Blanch Coalescence Model

For coalescence, the rate kernel employed is due to Prince and Blanch (1990) and Williams and Loyalka (1991). The latter text offers a fairly complete description of the physics of coalescence and various mathematical approaches for modeling the various coalescence mechanisms. Three primary mechanisms may be included in the coalescence kernel – (1) turbulent diffusion, (2) "laminar" shear, and (3) buoyancy. The so-called "laminar" shear contribution is modeled as a function of the local velocity gradient, and is relevant only for laminar flow and therefore, is not considered. The formulation of Prince and Blanch models the effect of turbulent diffusion due to "small" eddies, while the formulation of Williams and Loyalka also purports to model the effect of small eddies, though with an approach that relies on a bubble scale that is small compared to the scale of the turbulence. Hence the Williams and Loyalka formulation may not apply to the bubble sizes expected to be present in microbubble drag reduction applications.

The turbulent diffusion contribution is due to a statistical average of the fluid velocity fluctuations. However, a general turbulent flow also has a mean velocity gradient which has an effect on collisions. Williams and Loyalka discuss the impact of a laminar flow velocity gradient on coalescence. For the present application their formulation was adapted to treat the mean velocity gradient effect in turbulent flow.

The coalescence model considering turbulent diffusion due to small-scale turbulence and mean-shear is operational in NPHASE-PSU. The effect of buoyancy on coalescence has been neglected.

The turbulent collision rate is a dominant factor in bubble coalescence according to both Prince and Blanch (1990), and Williams and Loyalka (1991) . For small eddies the turbulence is assumed to be isotropic (at least on the scale of the bubble diameter) and the bubble size is assumed to lie in the inertial subrange. The same assumption is made in the breakup model formulation discussed below. Following Prince and Blanch, the collision frequency [θ_{ij}^{T} ,1/(m^3 s)] between bubbles i and j due to turbulent motion may be written:

$$\theta_{ij}^{T} = n_i n_j S_{ij} \left(\overline{u_i^2} + \overline{u_j^2} \right)^{1/2} \quad (26)$$

where n_i and n_j are the number densities (m^{-3}) of bubbles with diameters D_i and D_j, respectively. Also, $\overline{u_i^2}$ is the root mean square of the fluctuating velocity of bubble i and S_{ij} is the collision cross-sectional area defined by Prince and Blanch:

$$S_{ij} = \frac{\pi}{16} \left(D_i + D_j \right)^2 \quad (27)$$

The required fluctuating velocity in the inertial subrange for isotropic turbulence is given by Prince and Blanch:

$$u_i = \sqrt{2} \left(\varepsilon D_i \right)^{1/3} \; u_i = \sqrt{2} \left(\varepsilon D_i \right)^{1/3} \quad (28)$$

where the relevant turbulence length scale is assumed to be the bubble diameter. The leading constant in equation (28) is not universally agreed upon in the literature and the turbulence length scale also appears in several different forms, although always as a function of bubble diameter.

Combining the above expressions yields the desired relation for the collision frequency,

$$\theta_{ij}^{T} = n_i n_j \frac{\left(\sqrt{2}\pi\right)}{16} \varepsilon^{1/3} \left(D_i + D_j\right)^2 \left(D_i^{2/3} + D_j^{2/3}\right)^{1/2} \quad (29)$$

The probability that a collision results in coalescence is required to complete the rate kernel formulation. Again, following Prince and Blanch, this probability is termed the collision efficiency and is a function of the contact time between bubbles and the time required for bubbles to coalesce. For a pair of bubbles, this efficiency (λ_{ij}) is written as [following Coulaloglou and Tavlarides (1977)]:

$$\lambda_{ij} = \exp\left(-t_{ij} / \tau_{ij}\right) \quad (30)$$

where t_{ij} is the time required for bubbles of diameters D_i and D_j to coalesce and τ_{ij} is the contact time for the two bubbles. From other literature, Prince-Blanch presented the following expression for the coalescence time (t_{ij}):

$$t_{ij} = \left(\frac{\left(0.5D_{ij}\right)^3 \rho^{liq}}{16\sigma}\right) \ln\left(\frac{h_0}{h_f}\right) \quad (31)$$

where h_o is an initial film thickness between two bubbles as they just come into contact and h_f is a final critical film thickness where rupture occurs and the bubbles coalesce. The quantity D_{ij} is an equivalent diameter for bubbles of unequal size and is given by:

$$D_{ij} = \left(\frac{2}{D_i} + \frac{2}{D_j}\right)^{-1} \quad (32)$$

For air-water systems, the film thickness values quoted by Prince-Blanch (from other sources) are

$$h_0 = 10^{-4}\,\text{m}, h_f = 10^{-8}\,\text{m} \quad (33)$$

Finally, an estimate of the contact time (τ_{ij}) for bubbles in turbulent flow was made by Levich (1962) from dimensional analysis. A modification due to the relative velocity between the bubbles is noted by Carrica, et al. (1999), resulting in the following expression:

$$\tau_{ij} = \frac{D_{ch}}{u_{rel,ij} + 2\left(0.5D_{ch}\varepsilon\right)^{1/3}} \quad (34)$$

where D_{ch} is a characteristic length related to the bubble sizes and $u_{rel,ij}$ is the mean relative velocity between the colliding bubbles. The characteristic length (D_{ch}) in equation (34) may be taken as an adjustable parameter in this model. In the absence of better

information, D_{ch} will be taken as the average of the inverse of the bubble diameters, $2D_{ch} = \left(D_i^{-1} + D_j^{-1}\right)^{-1}$, as suggested by Carrica, et al. (1999), and Prince and Blanch (1990).

Furthermore, all quantities in the model are assumed to be statistical averages for a turbulent flow, thus further uncertainties in the model may result. There appears to be very little data or analysis in the literature addressing these complex issues.

Lehr-Mewes Breakup Model

For breakup, one rate kernel investigated is due to Lehr and Mewes (2001). This kernel has some important properties that allow the formation of a small bubble and a large bubble when a large bubble breaks up. The breakup mechanism considered is due to small-scale turbulent eddies. The Kumar-Ramkrishna (1996) formulation for breakup requires evaluation of volume integrals of the rate kernel, and the form of the kernel has some characteristics that can lead to numerical problems if not addressed carefully.

The Lehr-Mewes rate kernel for the (binary) breakup of a "mother" bubble with non-dimensional volume $\hat{x}_k$ into daughter bubbles with non-dimensional volumes $\hat{v}$ and $(\hat{x}_k - \hat{v})$ is given by:

$$r_1(v, x_k) = C_{LM} \frac{\hat{x}_k^{1/3}}{\hat{v}^{4/3}} \left[F_{min}(\hat{v}) - \frac{1}{\hat{x}_k^{7/9}} \right] \quad \text{for } \hat{v} \le \frac{\hat{x}_k}{2} \tag{35}$$

The non-dimensional daughter bubble volume $\hat{v}$ is a defined as:

$$\hat{v} = \frac{v}{v_{st}} \tag{36}$$

and v_{st} is related to the maximum stable bubble, v_{stable}, size by:

$$v_{stable} = 2^{3/5} \frac{\pi}{6} \frac{\sigma^{9/5}}{\rho_l^{9/5} \varepsilon^{6/5}} = 2^{3/5} v_{st} \tag{37}$$

where σ is the surface tension (N/m) between the gas and liquid phases, ρ_l is the liquid density (kg/m^3) and ε is the turbulence energy dissipation rate (m^2/s^3). The function F_{min} is given by:

$$F_{min}(\hat{v}) = \hat{v}^{7/6} \quad \text{for } \hat{v} \le 1$$
$$F_{min}(\hat{v}) = \frac{1}{\hat{v}^{7/9}} \quad \text{for } \hat{v} > 1 \tag{38}$$

Note that the rate kernel is symmetric about $\hat{v} = \hat{x}_k/2$ which allows its evaluation for $\hat{v} > \hat{x}_k/2$ using equation (35). Since the rate kernel must be non-negative, equation (35) must be restricted. This leads to a minimum value for a minimum daughter bubble size given by:

$$\hat{v}_{min} = \hat{x}_k^{-2/3} \tag{39}$$

where the rate kernel becomes zero. Also, the rate kernel has a slope discontinuity at $\hat{v} = \hat{v}_{min}$.

The form of the function F_{min} also gives rise to a slope discontinuity in the rate kernel at $\hat{v} = 1$, and possibly at $\hat{v} = \hat{x}_k/2$. Furthermore, the kernel has very large gradients for large non-dimensional bubble sizes. This is shown in Figure 2, where the normalized daughter size distribution for the L-M rate kernel is shown for several values of a volume ratio parameter, *VR*, defined by:

$$VR = \frac{\hat{x}_k}{\hat{v}_{stable}} = \frac{x_k}{v_{stable}} \qquad (40)$$

where v_{stable} is the maximum stable bubble volume. As a result of equation (37), the non-dimensional bubble volume is a function of the local flow properties, thus it will vary throughout the flow field.

In general the required moment integrals of the rate kernel required for the K-R formulation cannot be evaluated analytically. The zero-th moment is an exception.

All moments can be evaluated numerically; however for large values of *VR*, accuracy has been shown to be poor unless caution is exercised in selecting the integration step size. This is due to the large gradients shown in Figure 3. An approximate analytical evaluation of the kernel integrals for large *VR* was explored, but proved to be impractical and did not reduce CPU time. Thus an adaptive procedure was implemented for selecting the integration step size based on a prescribed accuracy in resolving the L-M rate kernel. For a very wide range of mother bubble sizes, this approach requires only a moderate number of integration steps (< 1000) to determine the necessary moments very accurately (< 0.01% error).

Martinez-Bazan Breakup Model

Another rate kernel investigated is due to Martinez-Bazan, et al. (1999a, b). This kernel is much different than the Lehr-Mewes kernel in that the formation of a small bubble and a large bubble from a bubble breakup has very low probability. The breakup mechanism considered is due to turbulent eddies and a phenomenological model for the breakup kernel (frequency) was developed using experimental data from a high-Reynolds number water jet flow with bubble injection. The experiments were conducted very carefully to insure that the turbulence in the jet was locally homogeneous, isotropic and in near-equilibrium. The model assumes that the initial bubble size, D_0 is in the inertial subrange, i.e., $\eta << D_0 << L_x$, where η is the Kolmogorov microscale and L_x is the integral scale of the turbulence.

$$\eta = \left(\frac{\upsilon^3}{\varepsilon}\right)^{1/4} \qquad (41)$$

$$L_x = \frac{\pi E_{11}(k_1 = 0)}{2\overline{u'^2}} \qquad (42)$$

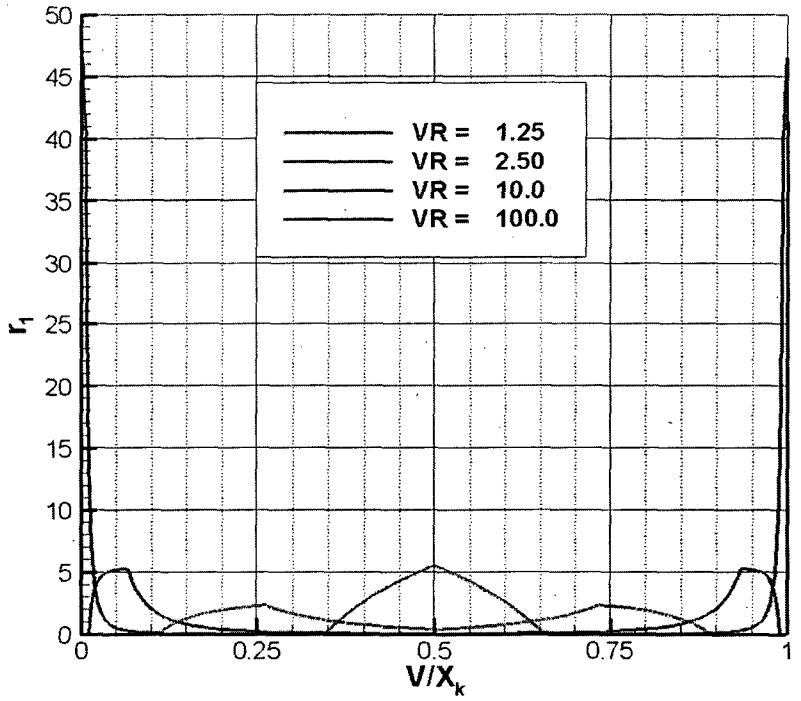

Figure 2. Normalized daughter size distribution for Lehr-Mewes rate kernel.

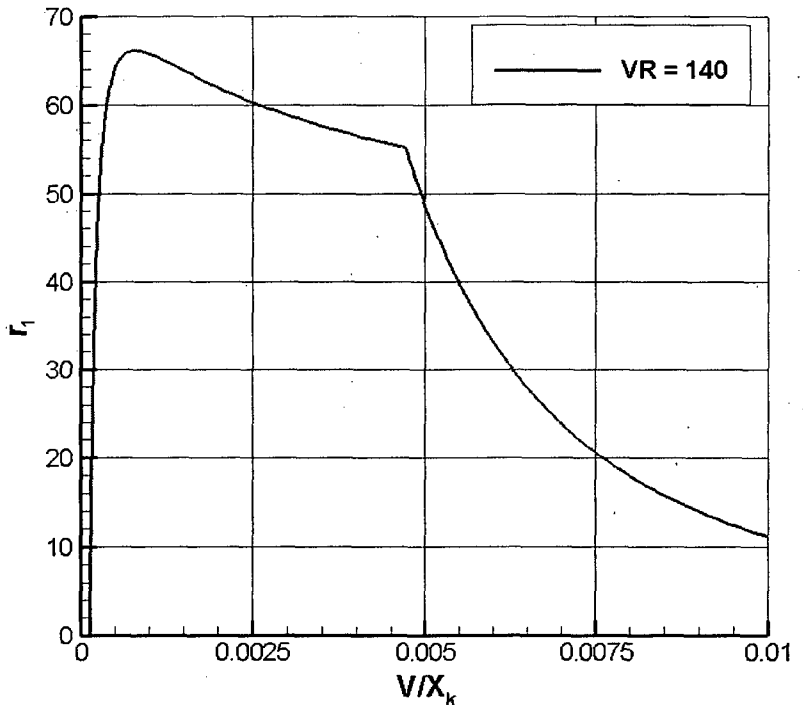

Figure 3. Normalized daughter size distribution for Lehr-Mewes rate kernel near V/Xk=0, for VR=140.

where u' is the fluctuating component of axial velocity, k_1 is the turbulence wave number in the axial direction and the tensor E is the turbulence energy-spectrum function [Hinze (1975)].

It should be noted that the experimental technique of Martinez-Bazan, et al. (1999a, b) had a minimum measurable bubble size of 83 μm, which Martinez-Bazan states did not affect their breakup frequency results.

A critical bubble diameter D_c exists and if $D \le D_c$ the bubbles will never breakup:

$$D_c = \left(\frac{12\sigma}{\beta\rho}\right)^{3/5} \varepsilon^{-2/5} \quad (43)$$

A minimum diameter exists below which there is insufficient turbulence induced stress to result in bubble breakup.

$$D_{min} = \left(\frac{12\sigma}{\beta\rho D}\right)^{3/2} \varepsilon^{-1} \quad (44)$$

The breakup frequency (1/s) is given by:

$$g(\varepsilon, D) = K_g \frac{\sqrt{\beta(\varepsilon D)^{2/3} - 12\sigma/(\rho D)}}{D} \quad (45)$$

where $\beta = 8.2$ [Batchelor (1956)] and $K_g = 0.25$ was determined experimentally by Martinez-Bazan, et al. (1999a).

The Martinez-Bazan, et al. (1999a) breakup model was developed for conditions more representative of MBDR flows than the Lehr-Mewes model, thus the former has been utilized in the present work.

Kumar-Ramkrishna Partitioning-Breakup

The Kumar-Ramkrishna particle bin size representation is shown schematically in Figure 4. Here x_i is the representative bubble bin volume (e.g. average or mid-point volume) due to breakup of mother with volume x_k

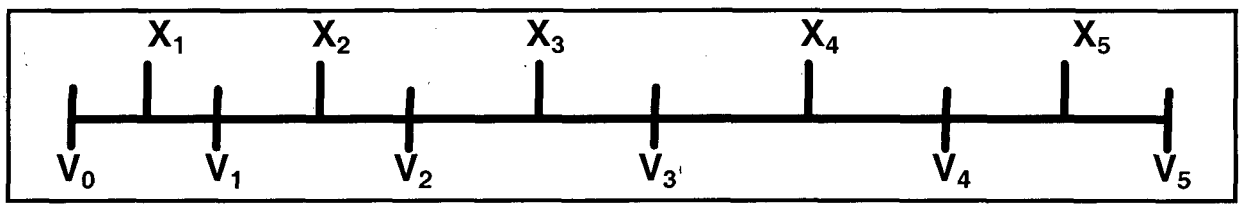

Figure 4. Kumar-Ramkrishna particle bin size representation.

Conservation leads to the following relation with the breakup rate is written as:

$$R_{Bb} = \sum_{k=1}^{N_B} n_{i,k} \Gamma_B(x_k) N_k(t) \quad (46)$$

where N_k is the total number of particles in bin "k" and $\Gamma_B(x_k)$ is the breakup frequency (kernel) for mother particle x_k and $n_{i,k}$ is the contribution to the population of the "i-th" bin size (x_i) due to breakup of particle x_k.

$$n_{i,k} = \frac{B^{\mu}_{i,k} x^{\nu}_{i+1} - B^{\nu}_{i,k} x^{\mu}_{i+1}}{x^{\mu}_{i} x^{\nu}_{i+1} - x^{\nu}_{i} x^{\mu}_{i+1}} + \frac{B^{\mu}_{i-1,k} x^{\nu}_{i-1} - B^{\nu}_{i-1,k} x^{\mu}_{i-1}}{x^{\mu}_{i} x^{\nu}_{i-1} - x^{\nu}_{i} x^{\mu}_{i-1}} \quad (47)$$

$$B^{\xi}_{i,k} \equiv \int_{x_i}^{x_{i+1}} v^{\xi} \beta(v, x_k) dv \quad (48)$$

The death rate due to breakup of a particle of size x_k is:

$$R_{Db} = \Gamma(x_k) N_k(t) \quad (49)$$

Kumar-Ramkrishna Partitioning-Coalescence

The coalescence formulation is simpler than that for breakup. The birth rate of particles due to the coalescence of particles in bins j and k is given by:

$$R_{Cb} = \sum_{\substack{j,k \\ x_{i-1} \le (x_j + x_k) \le x_{i+1}}}^{j \ge k} \left(1 - 0.5\delta_{j,k}\right) \eta_{j,k} \Gamma^{i,k} N_j N_k \quad (50)$$

where the distribution function due to the coalescence ($\eta_{j,k}$) is given by:

$$\eta_{j,k} = \frac{v^{\mu} x^{\nu}_{i+1} - v^{\nu} x^{\mu}_{i+1}}{x^{\mu}_{i} x^{\nu}_{i+1} - x^{\nu}_{i} x^{\mu}_{i+1}}, \quad x_i \le v \le x_{i+1}$$

$$\eta_{j,k} = \frac{v^{\mu} x^{\nu}_{i-1} - v^{\nu} x^{\mu}_{i-1}}{x^{\mu}_{i} x^{\nu}_{i-1} - x^{\nu}_{i} x^{\mu}_{i-1}}, \quad x_{i-1} \le v \le x_i$$

$$\text{where } v = x_j + x_k \quad (51)$$

where $v=x_j+x_k$, and $\Gamma^{i,k}$ is the coalescence rate (kernel) due to the coalescence of particles in bins j and k.

The death rate of particles due to the coalescence of particles in bins j and k is given by:

$$R_{Cd} = N_i \sum_{k=1}^{N_B} \Gamma^{i,k} N_k \quad (52)$$

Breakup and Coalescence – Interfacial Area Density Transport Formulation

An approximate formulation including bubble breakup and coalescence within the interfacial area framework was proposed by Lehr and Mewes (2001). Lehr and Mewes solved the population balance equation "to describe the evolution of bubble sizes in two-phase flow." To reduce the numerical complexity due to a large number of equations and

strong coupling, they formulated an equation for average bubble volume (equivalent to the interfacial area transport equation) using an approximate analytical approach. A summary of the Lehr-Mewes approach follows. Source terms in the population balance equation involve breakup and coalescence kernel functions that are a function of the bubble volume, v. By assuming that an arithmetically averaged bubble volume ($\bar{v}$) may be used in the kernel functions, a simplified solution for the bubble number-density distribution function, $f(v)$, results:

$$f(v) = \frac{\alpha_g}{\bar{v}^2}\exp\left(-\frac{v}{\bar{v}}\right) \quad (53)$$

$$n_B = \int_0^\infty f(v')dv' = \frac{\alpha_g}{\bar{v}} \quad (54)$$

Lehr and Mewes obtained a transport equation for average bubble volume with simplified source terms due to breakup and coalescence (equivalent to the source terms Φ_B and Φ_C and in equation (7). The bubble number-density PDF implies a bubble size distribution consistent with the above noted assumptions. We use this PDF to evaluate bubble number densities for discrete "bins." The bins are defined as shown in Figure 4.

Here x_i is the representative bubble bin volume (e.g. average or mid-point volume) of bin "i" and v_{i-1} and v_i are the lower and upper bin volumes of bin "i", respectively. The number density PDF of bubbles in bin "i" is then

$$N_{B(i)} = \frac{\alpha_g}{\bar{v}}\left(e^{-v_{i-1}/\bar{v}} - e^{-v_i/\bar{v}}\right) \quad (55)$$

This result approaches the number density PDF for a sufficiently large number of bins and a sufficiently large maximum bin volume. Also the first bin is assumed to contain all bubbles from zero bin volume to the uppermost volume of this bin (i.e. $v_1 = 0$). Further to prevent errors due to an insufficiently "large" maximum volume, the distribution must normalized such that $\sum_{\text{all bins}} N_{B(i)} = 1$.

As in the N-bin formulation, we use the Prince and Blanch (1990) rate kernel for coalescence and the Martinez, et al (1999a, b) rate kernel and daughter size distribution for breakup. A complete description of these models is included above; only the essentials are summarized here under the assumption that the rates may be evaluated using the mean bubble diameter.

$$\phi_C = n_B^2\left(\frac{\pi}{2}\right)\varepsilon^{1/3}\bar{D}^{7/3}\exp\left(-t_B/\tau_B\right) \quad (56)$$

where n_B is the bubble number density, ε is the turbulence energy dissipation rate, t_B is the time required for two bubbles of diameter $\bar{D}$ to coalesce and τ_B is the contact time for the two bubbles. In the interfacial area density formulation, the bubble number density is given by

$$n_B = \frac{a_i}{\pi \overline{D}^2} \qquad (57)$$

As in the prior section the time required for two bubbles to coalesce is given by:

$$t_B = \left(\frac{\left(0.5\overline{D}\right)^3 \rho^{liq}}{16\sigma} \right)^{1/2} \ln\left(\frac{h_0}{h_f}\right) \qquad (58)$$

where h_o is an initial film thickness between two bubbles as they just come into contact and h_f is a final critical film thickness where rupture occurs and the bubbles coalesce. For air-water systems, the film thickness values quoted by Prince and Blanch (from other sources) are

The contact time for bubbles in turbulent flow [Levich (1962)] with a modification due to the relative velocity between the bubbles [Carrica, et al. (1999)], is given by:

$$\tau_B = \frac{D_{ch}}{U_{B,rel} + 2\left(0.5 D_{ch} \varepsilon\right)^{1/3}} \qquad (59)$$

where, as before, D_{ch} is a characteristic length related to the bubble sizes and $u_{B,rel}$ is the mean relative velocity between the colliding bubbles. The characteristic length (D_{ch}) is taken as $D_{ch} = \overline{D}$.

Enthalpy Transport

For compressible flows and flows with heat transfer, it is necessary to solve for an energy equation. NPHASE-PSU incorporates an enthalpy transport equation for each field:

$$\frac{\partial}{\partial t}\left(\alpha^k \rho^k h^k\right) + \frac{\partial}{\partial x_j}\left(\alpha^k \rho^k u_j^k h^k\right) = \frac{\partial}{\partial x_j}\left[\alpha^k \left(\mu^k + \frac{\mu_t^k}{\sigma_h^k}\right)\frac{\partial h^k}{\partial x_j}\right] + S_h^k \qquad (60)$$

Turbulence Model

NPHASE-PSU has a number of low and high Reynolds number form 2-equation turbulence models (k-ε, q-ω, k-ω, k-R) and a low Reynolds number 4-equation v2f model [Durbin, (1991)]. In the context of multifield flows, separate turbulence transport scalars are solved for each field. For example, the high Reynolds number k-ε model is written:

$$\frac{\partial}{\partial t}\left(\alpha^k \rho^k k^k\right) + \frac{\partial}{\partial x_j}\left(\alpha^k \rho^k u_j^k k^k\right) = \frac{\partial}{\partial x_j}\left[\alpha^k \left(\mu^k + \frac{\mu_t^k}{\sigma_k^k}\right)\frac{\partial k^k}{\partial x_j}\right] + P^k - \alpha^k \rho^k \varepsilon^k + S_k^k \qquad (61)$$

$$\frac{\partial}{\partial t}\left(\alpha^k \rho^k \varepsilon^k\right) + \frac{\partial}{\partial x_j}\left(\alpha^k \rho^k u_j^k \varepsilon^k\right) = \frac{\partial}{\partial x_j}\left[\alpha^k \left(\mu^k + \frac{\mu_t^k}{\sigma_\varepsilon^k}\right)\frac{\partial \varepsilon^k}{\partial x_j}\right] + C_1 \frac{\varepsilon^k}{k^k} P^k - C_2 \frac{\varepsilon^k}{k^k} \alpha^k \rho^k \varepsilon^k + S_\varepsilon^k$$

In equation (61), all field indicator superscripts are eliminated if only the liquid field is solved. S_k and S_ε are available source/sink terms to: extract turbulence energy

associated with breakup [Meng and Uhlman (1998), Kunz et al. (2003)], and modify production due to interface dynamics and mass transfer mechanisms proposed by various authors [Ferrante and Elghobashi (2004, 2005), Tryggvason and Lu (2005)].

Numerics/Code

For single phase flow, the present algorithm follows established segregated pressure based methodology. A colocated variable arrangement is used and a lagged coefficient linearization is applied [Clift and Forsyth (1994), for example]. One of several diagonal dominance preserving, finite volume spatial discretization schemes is selected for the momentum and turbulence transport equations. Continuity is introduced through a pressure correction equation, based on the SIMPLE-C algorithm [Van Doormal and Raithby (1984)]. In constructing cell face fluxes, a momentum interpolation scheme [Rhie and Chow (1983)] is employed which introduces damping in the continuity equation. At each iteration, the discrete momentum equations are solved approximately, followed by a more exact solution of the pressure correction equation. Turbulence scalar and volume fraction equations are then solved in succession. As discussed above, several important numerical issues arise in two-fluid CFD, foremost among these, that sufficient implicit coupling between the constituents be established. In the present work this is accomplished using the Coupled Phasic Exchange (CPE) algorithm [Kunz et al. (1998)]. In NPHASE-PSU, CPE has been extended to a fully unstructured, parallel, time accurate scheme employing higher-order discretization practices. Details of the data structure, discretization, and CPE elements of the scheme are summarized in this section.

Data Structure

The hierarchal data structure employed is illustrated in Figure 5. The cell-centered finite volume flow solver accepts arbitrary polyhedral elements. The data structure is face based, that is, subsequent to the assembly of geometric parameters in the front end, all inter-element connectivity is retained in face pointers to the two adjacent cells. The fundamental data structure member is the "fedge" (face edge) which points to its two vertices and faces. Each face points to its bounding elements. This data structure provides a convenient framework for assembly of all required geometric parameters.

fedges and faces are identified as either internal or boundary. A "boundary_patch" structure in the C flow solver includes as members a number of attributes for boundary faces including areas and other geometric information, scalar values at the face center, fluxes, and inter-partition boundary data storage and transfer buffers.

Discretization

The governing equations are discretized using a cell centered finite volume method applied to arbitrary polyhedral cell types. Inviscid and viscous fluxes are accumulated by sweeping through internal and boundary faces. For inviscid flux evaluation:

$$\int_{\underline{A}} \alpha^k \rho^k \phi^k \underline{V}^k \cdot d\underline{A} = \sum_f C_f^k \phi_f^k \qquad (62)$$

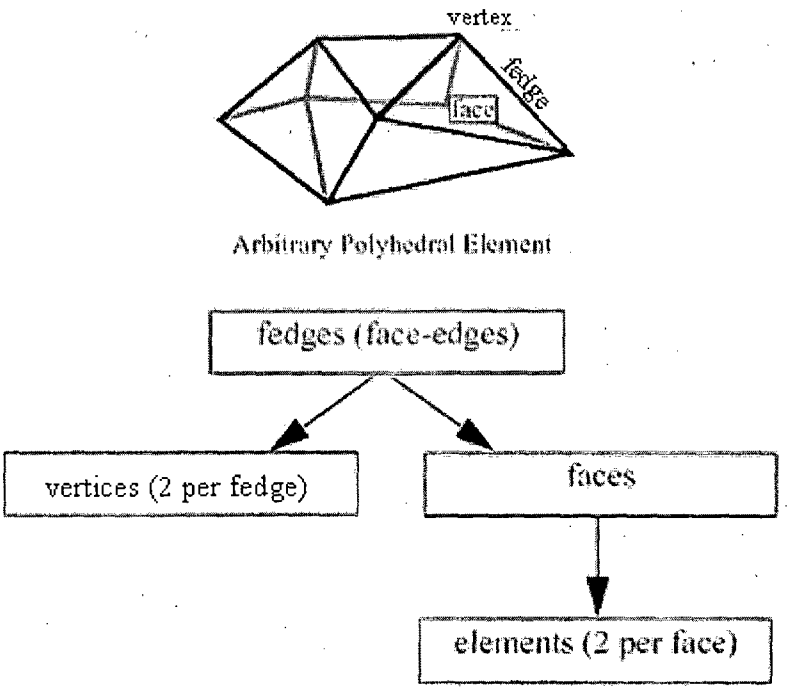

Figure 5. Heirarchal data structure in NPHASE-PSU

where C_f^k is face mass flux for field k, and ϕ_f^k is the value of general transport scalar ϕ^k evaluated at face, f. The summation is taken over all faces bounding the element. C_f^k is evaluated based on field variables available prior to the solution of the transport equation for ϕ^k (lagged coefficient linearization). Second order accuracy is obtained by evaluating C_f^k using a central plus 4th difference pressure artificial dissipation term due to Rhie and Chow (1983) : C

$$C_f^k = \rho_f^k \alpha_f^k \overline{\underline{V}^k} \cdot \underline{A}^f + \rho_f^k \alpha_f^k \left[\overline{B}^k \left(\overline{\nabla p} \cdot \underline{A}^f - \Delta p \left| \underline{A}^f \right|^2 \right) \right] + \rho_f^k \alpha_f^k \left[\overline{F}^k \left(\overline{\nabla \alpha} \cdot \underline{A}^f - \Delta \alpha \left| \underline{A}^f \right|^2 \right) \right] \quad (63)$$

and by evaluating ϕ_f^k from Lien (2000):

$$\phi_f^k = \phi_U^k + \left(\nabla \phi^k \cdot d\underline{r} \right)_U \quad (64)$$

In equation (63), the overbar denotes a geometrically weighted mean at the face, i.e., referring to Figure 6. :

$$\overline{\nabla p} = (1-s)(\nabla p_1) + s(\nabla p_2)$$
$$s \equiv \delta s_1 / (\delta s_1 + \delta s_2) \quad (65)$$

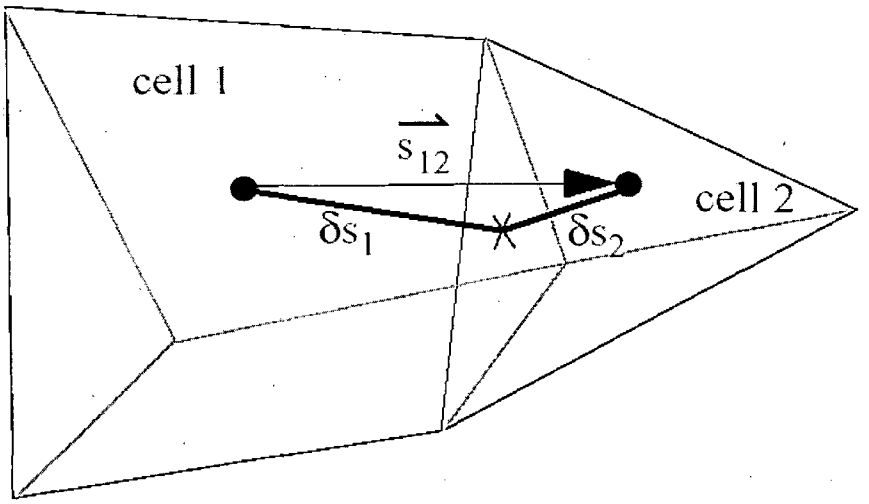

Figure 6. Geometry nomenclature for cell face evaluations.

and Δ designates a difference across the face (i.e., $\Delta p \equiv p_2 - p_1$). In equation (64) subscript U designates the quantity associated with the element upwind of face, f (which can vary with field), and $d\vec{r}$ is the vector from the upwind cell center to the face center. In Figure 7, results of a two-dimensional inviscid parallel stream test case are presented using a square mesh on a square domain aligned 45° skew to the flow direction and also using a triangular mesh. Inflow axial velocities are specified as =2 along the upper half inlet and =1 on the lower half. On both meshes, the significant interface smearing associated with first order upwinding is significantly reduced using the second order expression in equation (64). As detailed in Kunz et al. (1998) , dissipation parameters, $\overline{B^k}$ and $\overline{F^k}$ in equation (63) are scaled in a fashion that accommodates interfacial drag, mass transfer and dispersion forces:

$$B^k \equiv \sum_{l=1,nfield} \alpha^l \left(\underline{\underline{P}}^{-1}\right)^{kl}, F^k \equiv K \sum_{l=1,nfield} \left(\underline{\underline{P}}^{-1}\right)^{kl} \qquad (66)$$

where **P** is the NPx NP point coefficient matrix for the momentum equations defined below, which incorporates drag and mass transfer. F^k is consistent with a widely used class of dispersive interfacial forces, $M_i^{kl} = K\nabla\alpha^l$ [e.g., Lopez DeBertodano (1998)].

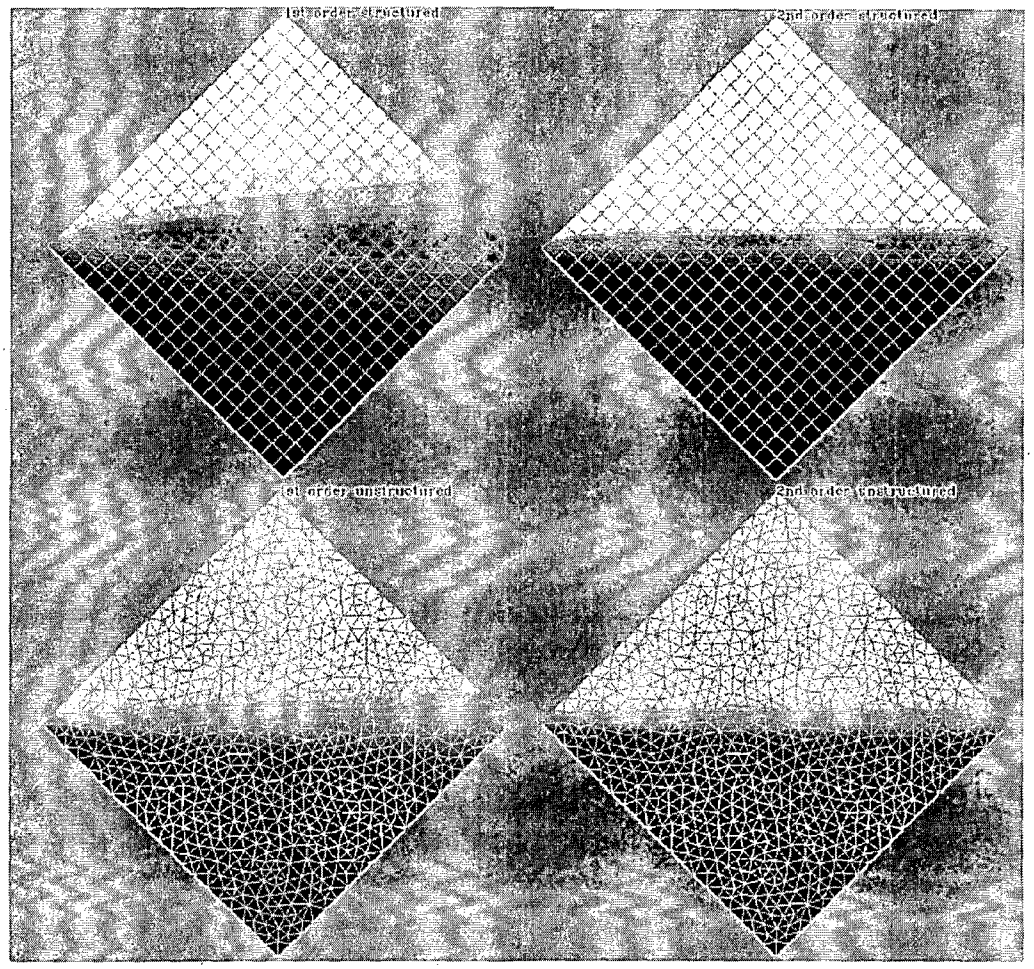

Figure 7. Comparison of first and second order convection discretization for an inviscid "mixing" layer. Flow is left to right. Axial velocity contours, white: $\bar{V} = 2$, black: $\bar{V} = 1$.

The evaluation of B^k and F^k in equation (66), requires the inversion of a rank NP matrix **P** at each grid point, at each iteration. This potentially CPU intensive procedure is circumvented by applying a simple Jacobi fixed point iterative procedure to approximately invert **P**. This procedure is rapidly convergent (2 sweeps are employed) since the **P** matrices are very well conditioned as discussed below.

Neglecting cross-diffusion and dilatation, the viscous flux in the momentum equations can be written for an element face as:

$$\int \left(\underline{\underline{\tau}} \cdot d\underline{A}\right), \underline{\underline{\tau}} = \alpha\mu\left(\nabla \underline{V}\right) \qquad (67)$$

Referring to Figure 6, the gradient of a scalar, ϕ, on the face can be written as:

$$\nabla\phi = \underbrace{\nabla\phi - \left(\nabla\phi \cdot \frac{\underline{S}_{12}}{|\underline{S}_{12}|}\right)\hat{\varepsilon}_{s12}}_{\mathbf{A}} + \underbrace{\left(\nabla\phi \cdot \frac{\underline{S}_{12}}{|\underline{S}_{12}|}\right)\hat{\varepsilon}_{s12}}_{\mathbf{B}} \qquad (68)$$

The terms labelled A represent components of the gradient that are orthogonal to $\bar{s}_{12}$. These terms are generally small (for hexahedral or prismatic elements extruded from geometric surfaces, neglecting them is nearly equivalent to the thin-layer assumption). Their discrete form is treated explicitly in the solution of the momentum equations (term S^k in equation (75) below). The terms labelled B represent components of the gradient that are parallel to $\bar{s}_{12}$. These are discretized as:

$$\int_f (\alpha\mu \nabla \underline{V} \cdot d\underline{A}) = \overline{(\alpha\mu)}_f \left(\nabla \underline{V} \cdot \frac{d\underline{s}}{|d\underline{s}|} \right) (\hat{\varepsilon}_s \cdot d\underline{A}) = \overline{(\alpha\mu)}_f \frac{(\underline{V}_2 - \underline{V}_1)(d\underline{s} \cdot d\underline{A})}{|d\underline{s}|^2} \quad (69)$$

and are treated implicitly (terms A_P^k and A_{nb}^k in equation (75) below).

Gradients that appear in the flux calculations, and elsewhere, are computed using Gauss' Law:

$$\nabla\phi = \frac{1}{\forall} \sum_f \underline{A}_f \bar{\phi}_f^k \quad (70)$$

with internal face values of $\bar{\phi}_f^k$ computed from equation (65), and the summation take over all faces bounding an element. Equation **Error! Reference source not found.** is computed by sweeping all internal and boundary faces, accumulating adjacent element contributions to V and $\bar{A}_f \phi_f^k$ from the face.

Interfacial Force Evaluation

In order to discuss interfacial force discretization issues, we consider three classes of these terms. First, when cast as in equation (10), drag can be viewed as a scalar sink term. That is to say, drag term, D^{kl}, which appears in the momentum equations as:

$$\sum_{k \neq l} D^{kl} \left(u_i^l - u_i^k \right) \quad (71)$$

is generally evaluated for each element, and by virtue of its relative velocity factor, incorporated implicitly in the NPx NP block diagonal, **P**, of the momentum equation coefficient matrix, as seen in equation (79) below. As discussed above, the appearance of drag in this term is accommodated consistently in the Rhie-Chow scale factors in equation (66). It is noted that numerically, mass transfer plays a very similar role to drag (though $D^{kl} = D^{lk}$ and in general $\Gamma^{kl} \neq \Gamma^{lk}$), and accordingly its treatment is consistent with that discussed for drag.

The second class of interfacial force terms are those that are linear in the gradient of volume fraction. These are generally dispersive in nature. These terms are evaluated straightforwardly using model equations such as (18), however, as is demonstrated in Kunz et al. (1998), including the Rhie-Chow-like term, $\rho_f^k \alpha_f^k \left[\bar{F}^k \left(\overline{\nabla\alpha} \cdot \underline{A}^f - \Delta\alpha \left| \underline{A}^f \right|^2 \right) \right]$, in equation (66) is critical for obtaining convergent oscillation free solutions when such forces are present.

The third class of forces are simply those that do not conform to drag-like or dispersive-like forms. An example is lift, a particular form of which is taken here from Lahey and Drew (2000):

$$\underline{M}_{LIFT}^{c-d} = \alpha^d \rho^c C_L \underline{V}^{rel} \times \nabla \times \underline{V}^c \qquad (72)$$

where superscripts c and d refer to continuous and disperse fields respectively.

A straightforward discretization of equation (72) in an element centered (or variable colocated) scheme such as presented here, would involve evaluating gradients using equation (70) and multiplying by appropriate velocity, volume fraction and density factors using element values. In Kunz and Venkateswaran (2000) it was demonstrated that such an approach can also lead to solution oscillations and attendant convergence degradation. There it was observed that staggered grid methods (i.e., those where the momentum equations are evaluated at locations staggered to the element centers) do not exhibit this behavior for this class of force. Accordingly, a staggered force discretization was proposed wherein the force in equation (72) is evaluated at each cell-face. Face values are then averaged to obtain element values. This force distribution renders staggered and colocated forms identical for linear forces and thereby removes solution oscillations. In the present unstructured framework this "distribution" of force across several nodes can be written:

$$\overline{M} = \frac{\sum_f M_f \forall_f}{\forall} \qquad (73)$$

where M_f represents the force averaged to the face per equation (72), $\forall$ is the element volume and $\forall_f$ is the volume formed by face f and the segments connecting the face vertices to the volume centroid. It was observed in Kunz and Venkateswaran (2000) that this approach is equivalent to the addition of a second difference artificial dissipation to the standard colocated discretization, i.e., in the present unstructured context:

$$\overline{M} = M + \nabla \square K \nabla M \qquad (74)$$

where scaling factor, K, has dimensions of length^2.

Boundary Conditions

A palette of boundary conditions are available in the code including walls, symmetry boundaries, inlets (transport scalars specified, pressure extrapolated from the domain interior), pressure boundaries (transport scalars extrapolated from the domain interior, pressure specified), and cyclic boundaries (for turbomachinery analysis). All boundary conditions are treated implicitly in the formation of influence coefficients for the transport scalars. For scintered metal plate injection, porous wall boundary conditions are used, where an area permeability, λ, is specified. Shear force on porous boundary faces is apportioned as $F = \tau_w A_f (1 - \lambda)$, where A_f is the face area and $A_f \lambda$ is the area available for injection flux.

Implicit Solution Procedure

Invoking a dual-time formulation, the discretized governing equations for transport scalar, ϕ^k, can be written in Δ-form as:

$$\left[A_P^k + \sum_{k\neq l} b^{kl} + \frac{\rho^k \alpha^k \forall}{\Delta\tau} + \frac{3\rho^k \alpha^k \forall}{2\Delta\tau}\right]\Delta\phi_P^k$$
$$-\sum_{k\neq l} b^{lk}\Delta\phi_P^l - \sum_{nb} A_{nb}^k \Delta\phi_{nb}^k = \left[\sum_{nb} A_{nb}^k \left(\phi_{nb}^k\right)^{n+1,m} - \right.$$
$$\left.\left(A_P^k + \sum_{k\neq l} b^{kl}\right)\left(\phi_P^k\right)^{n+1,m} - \sum_{k\neq l} b^{lk}\left(\phi_P^l\right)^{n+1,m}\right]$$
$$+S^k - \left[\frac{3\rho^k \alpha^k \forall}{2\Delta t}\left(\phi_P^k\right)^{n+1} - \frac{2\rho^k \alpha^k \forall}{\Delta t}\left(\phi_P^k\right)^{n} + \frac{\rho^k \alpha^k \forall}{2\Delta t}\left(\phi_P^k\right)^{n-1}\right] \quad (75)$$

where $\Delta\phi^k \equiv \left(\phi^k\right)^{n+1,m+1} - \left(\phi^k\right)^{n+1,m}$, and b^{kl} represents the accumulated drag and mass transfer terms (i.e., for the momentum equations, $b^{kl} = D^{kl} + \Gamma^{kl}$)

In equation (75), second order backward differencing has been used for the physical time derivative (Δt) and Euler implicit differencing is employed for the pseudo-time derivative ($\Delta\tau$). A standard under-relaxation procedure is employed where an appropriate underrelaxation factor, ω is selected ($0.3 \leq \omega \leq 0.7$) and the pseudo-timestep is evaluated from:

$$\Delta\tau \equiv \frac{\omega}{1-\omega}\left(\frac{\rho^k \alpha^k \forall}{A_P^k + \sum_{k\neq l} b^{kl}}\right) \quad (76)$$

It has been observed in Venkateswaran et al. (1997) that such a specification is equivalent to a local timestepping procedure that accommodates CFL and VonNeuman stability. For physical transients, pseudo-timesteps correspond to sub-iterations of the SIMPLE-C algorithm.

Phase Coupled Scalar Linear Solution Strategy

Equation (75) represents a coupled system of NP equations for the NP unknowns ϕ^k:

$$\underline{\underline{A}}\,\underline{\phi} = \underline{S} \quad (77)$$

where coefficient matrix **A** has the form:

$$\underline{\underline{A}} = \begin{bmatrix} \underline{\underline{P}} & & & & \\ & \underline{\underline{P}} & & \underline{\underline{U}} & \\ & & \underline{\underline{P}} & & \\ & & & \ddots & \\ & \underline{\underline{L}} & & & \underline{\underline{P}} \\ & & & & \underline{\underline{P}} \\ & & & & & \underline{\underline{P}} \end{bmatrix} \tag{78}$$

with

$$\underline{\underline{P}} = \begin{bmatrix} \frac{A_P^1 + \sum_{k\neq 1}^{NP} b^{1k}}{\omega} & -b^{21} & \cdots & -b^{NP1} \\ -b^{12} & \frac{A_P^2 + \sum_{k\neq 2}^{NP} b^{2k}}{\omega} & \cdots & -b^{NP2} \\ \vdots & \vdots & \cdots & \vdots \\ -b^{1NP} & -b^{2NP} & \cdots & \frac{A_P^{NP} + \sum_{k\neq NP}^{NP} b^{NPk}}{\omega} \end{bmatrix} \tag{79}$$

and upper and lower block triangular matrices **U** and **L** containing neighbor cell influence coefficients.

For the diagonal dominance preserving discretizations employed, conventional iterative schemes will have diagonally dominant iteration matrices with spectral radii less than or equal to the underrelaxation factor, ω [Kunz et al. (1998)], a direct consequence of the well conditioned nature of the main diagonal block matrix **P**. Accordingly, we consistently employ a simple point Jacobi scheme for solving equation (75) for all scalars (u_i, α, k, ε), as this scheme is guaranteed to provide adequate convergence within several sweeps. For the momentum equations, all three velocity components are solved for all fields simultaneously using point Jacobi iteration.

As discussed above, the well conditioned nature of **P** renders determination of dissipation parameters B^k and F^k, in equation (66), (which scale with $\mathbf{P}^{-1}$) amenable to a simple point Jacobi iteration as well.

Continuity Equation Linear Solution Strategy

In the present work a mixture volume conservation equation is derived by summing individual field volume fraction equations, each normalized by field density. A SIMPLE-C [Van Doormal and Raithby, (1984)] based pressure-velocity corrector relation (which

accommodates the same level of interfield coupling as the artificial dissipation operators discussed above, Kunz et al. [1998]) is applied to develop an elliptic pressure correction equation. Transport equations for the field volume fraction equations are then solved. In this fairly standard method, under-relaxation is not employed for the pressure corrector equation in order to achieve a measure of mixture volume conservation at each pseudo-timestep. As as result, the discrete pressure corrector equation system is symmetric positive semi-definite ($A_P = \sum_{nb} A_{nb}$) and thereby its linear solution is a challenging and important factor in the nonlinear convergence rate of the overall scheme. In NPHASE-PSU, the PETSC suite of solvers are employed for the solution of this system. Depending on the degree to which mass conservation needs to be satisfied at a given non-linear iteration, a GMRES solver or a more CPU intensive Algebraic Multigrid procedure are invoked from the PETSC library of solvers. Details of these solvers are provided in (PETSC [2006]).

Parallelization

The code is parallelized based on domain decomposition using MPI. Partitioning is carried out in the pre-processor, fump, as described in the user's Manual below. Inter-partition boundaries are input to the flow code from fump as any other boundary condition with a single additional boundary patch attribute being the neighbor partition processor number. fump writes inter-partition face pointers to the NPHASE-PSU input files (unphase.gridxxx) in the same order that these faces are encountered in fump. Accordingly no reordering is required when loading and unloading 1-D structures associated with message passing. Data is passed after each scalar is computed in the segregated procedure. For the point iterative solvers used for the scalar equations, $\Delta\phi$ is passed at every sweep of the linear solver, so that there is no degradation in convergence due to domain decomposition. For the PETSC solvers used for the pressure corrector equation, the code is parallelized at the matrix level. Accordingly, a global matrix is assembled each non-linear iteration and global mass conservation is strictly enforced each timestep as the pressure solver is converged.

Further details on the physical models, numerics and code are available in Kunz et al. (1999, 2000, 2001).

Preprocessing

Overview

Figure 8 illustrates the front end for NPHASE-PSU V 3.0. This latest version of NPHASE-PSU has a significantly different front-end. Specifically, the code now accepts much 'lighter' faced-based grid specification. This achieves several goals. Firstly it enables the specification of arbitrary polyhedral meshes rather than being restricted to the four simplicial element types (tetrahedral, hexahedra, prisms and pyramids) like the previous "element-based" front end. Secondly, most commercial grid generators produce the face-based COBALT files now native to NPHASE-PSU (Gridgen, ICEM, HARPOON) or closely related face based formats (e.g., GAMBIT). Thirdly, the front end pre-processor, fump (**f**ace-based **u**nstructured **m**esh **p**re-processor) is much faster and easier to extend than its element-based predecessor, pump.

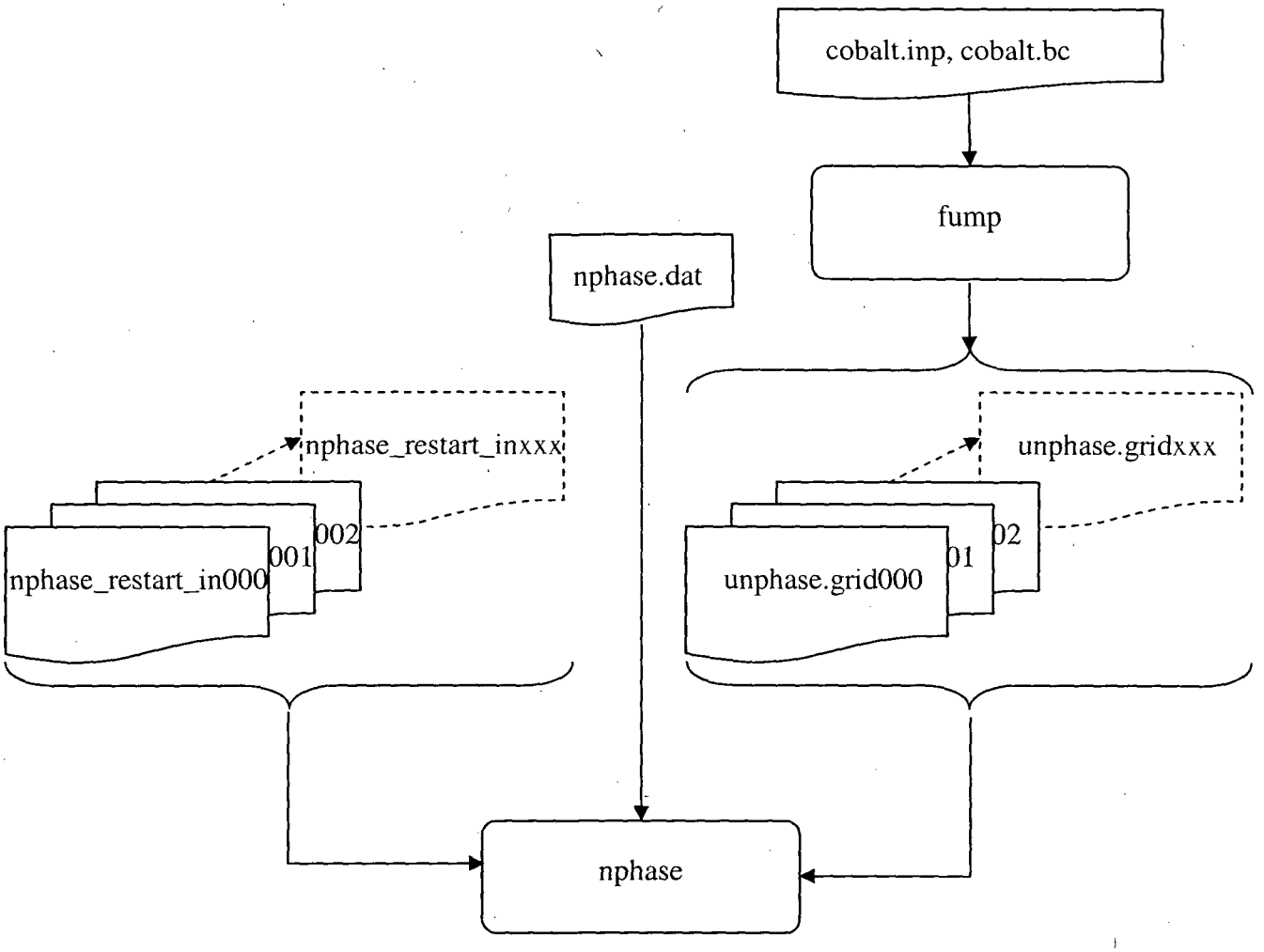

Figure 8. Sketch of frontend for NPHASE

Input Files

Referring to Figure 8, there are three sets of input files to NPHASE-PSU. The first is the author file, nphase.dat, which is a simple key-word based ascii file that specifies all of the real and integer data defining execution control, boundary condition flags and attributes, fluid properties and initial conditions. A simple example nphase.dat file is included in Figure 9, and the "Control Commands" section below is devoted to a description of all of the key words available. The keywords (character attributes such as "number of fields") are case sensitive but blanks are ignored. Integer and real attributes can be written in free format. Lines are commented out by placing # in column 1.

```
#case title:
#simple nphase.dat file

iterations to perform 100

number of fields 1

time accurate simulation
temporal discretization momentum 1
physical timestep in seconds .1
number of physical timesteps 1
transient file write frequency 2

#initialize run with restart file

produce ensight output
restart file write frequency 100

dont perform wall match logic

inlet patch 1 0
1.0 0. 0. 1.  1. 0.1 0.1 0. 0.

pressure patch 1 0
0. 1. 1. 0.1 0.1 0. 0.

turbulent flow high reynolds number k epsilon

constant fluid molecular viscosity 1.0e-2
constant fluid density 1.0

function entry/exit echo off

solversweepsforu 3
solversweepsforv 3
solversweepsforw 3
solversweepsfork 3
solversweepsfore 3

solver choice for velocity components jacobiuvw
solver choice for pressure petsc
parallel strategy for pressure corrector: matrixlevel

initialize u field   1.
initialize k field   .1
initialize e field   .1
```

Figure 9. Sample nphase.dat file

The second set of files are the grid files, written in COBALT unstructured format, cobalt.inp and cobalt.bc. The cobalt.inp file completely defines the input grid in faced based format. The file is written in COBALT format, as defined in Figure 10. The user need not concern himself with the content of this file although if a formatted cobalt.inp file is written by the grid generator, it will be readable.

ndim nzones nbc
nvert nface ncell maxppf maxfpc
x_1 y_1 z_1
x_2 y_2 z_2

.
.
.

x_{nvert} y_{nvert} z_{nvert}
nvf_1 ($f\_v_1(ivf)$,ivf=1,nvf_1) f_e1_1 f_e2_1
nvf_2 ($f\_v_2(ivf)$,ivf=1,nvf_2) f_e1_2 f_e2_2
.
.
.
nvf_{nface} ($f\_v_{nface}(ivf)$,ivf=1,nvf_{nface}) f_e1_{nface} f_e2_{nface}

Figure 10. cobalt.inp grid file format

In this file the parameters are defined as follows:

- ndim = # of dimensions → always 3 for NPHASE-PSU, even if a 2D case is set up (see "Running Two-Dimensional Problems,' below)
- nzones = 1 (always for NPHASE-PSU)
- nbc = total number of boundary conditions that are defined in cobalt.inp and cobalt.bc
- nvert = number of vertices in model
- nface = number of vertices in model
- ncell = number of vertices in model
- maxppf = maximum number of vertices per face in any one face in domain
- maxfpc = maximum number of faces per element in any one element in domain
- x_{ivert} y_{ivert} z_{ivert} = vertex coordinates for vertex ivert
- nvf_{iface} = number of vertices on face iface
- f_v_{iface} = list of vertex numbers for face iface, listed in oder around the periphery of the face such that the right-hand-rule applied to this ordering defines a direction from bounding element) f_e1 to f_e2

- f_e1_{iface} = element number on "low" side of face iface
- f_e2_{iface} = element number on "high" side of face iface. If $f\_e2_{iface} < 0$ then iface is a boundary face and f_e2_{iface} specifies the (negative of the) boundary identifier as defined in cobalt.bc.

The cobalt.bc file defines boundary condition names. This ASCII file is written in standard COBALT boundary condition file format, an example of which is shown in Figure 11. The user need not concern himself with the content of this file unless it is not written by the grid generator (e.g., HARPOON), in which case the user must build this file by hand to conform to the boundary condition numbering in the grid generator.

```
################################################
 Boundary Condition Specification File for:
 Gridgen grid exported : Thu May 18 10:51:30 2006
################################################
11
Wall_00
Gridgen bc region: 11
Methods: User Created BC
User data supplied here - see COBALT doc!
################################################
9
Inflow_00
Gridgen bc region: 9
Methods: User Created BC
User data supplied here - see COBALT doc!
################################################
10
Pressure_00
Gridgen bc region: 10
Methods: User Created BC
User data supplied here - see COBALT doc!
################################################
```

Figure 11. cobalt.bc file format

In this file three boundary conditions have been defined, Wall_00, Inflow_00 and Pressure_00. This file tells the pre-processor, fump, that Wall_00 faces in the cobalt.inp file will have identifier f_e2_{iface} = - 11. Also for Inflow_00, f_e2_{iface} = -9 and for Pressure_00, f_e2_{iface} = -10.

The third set of files are solution restart files. NPHASE-PSU always generates a restart file for each processor after execution completion (and optionally at intermediate iterations/timesteps as defined in "Control Commands" section below). These output files conform to the naming convention nphase_restart_out*xxx* where *xxx* is the 3-digit processor identifier (range from 000 to 999). Solution restarts are invoked by 1) copying each of these output restart files to a corresponding input restart file (i.e., mv nphase_restart_out000 nphase_restart_in000) and then 2) activating the keyword "initialize run with restart file" in nphase.dat.

Boundary Condition Specification

NPHASE-PSU supports no-slip wall, porous wall, pressure, inflow, symmetry, farfield and other specialized boundary conditions. The user needs to define boundary patch names that conform to what NPHASE-PSU is expecting. Specifically, NPHASE-PSU requires boundary patch names that conform to the following syntax: Boundarytype_boundarynumber, where Boundary type is either of these character strings: Wall, Porwall, Inflow, Symmetry, Farfield, Pressure. Boundarynumber is a 2-digit integer starting at 00. So for example there may be two inflows in a model with different attributes, these would be named by the user Inflow_00 and Inflow_01. These strings must appear in cobalt.bc. The grid generators GRIDGEN and ICEM automatically propagate these names into cobalt.bc upon grid output, provided the user names them in the grid generator. If HARPOON is used the user must define these in cobalt.bc.

Each boundary type has its own attributes, which are defined in nphase.dat as specified in the "Control Commands" section below.

fump

The pre-processor to NPHASE-PSU, fump, reads the cobalt.inp file (formatted or unformatted) and the cobalt.bc files as input and performs two tasks:

1) Executes domain decomposition by invoking METIS (2006). fump does this by first extracting graph information for METIS. Specifically, each element in the grid is designated as a "vertex" in the graph and each element with which it shares a face represents an "edge". The kmetis module is used to partition the graph into approximately equal sizes.
2) Builds the internal pointer connectivity (e.g., face→element, edge→ vertex, edge→ face), boundary pointer connectivity and interprocessor communication information.

fump is run interactively (or in a script) by simply typing the name of the exeucutable, fump, to a UNIX shell prompt. fump has two small user inputs: 1) # of processors to use for domain decomposition, 2) any scale factor the user may wish to apply to the grid.

The output of fump is a series of files, unphase.grid*xxx*, where where *xxx* is the 3-digit processor identifier (range from 000 to 999). Each of these files is read by the corresponding processor from the executing front end of NPHASE-PSU.

fump is written in ANSI-C and compiles (at least) under the gnu C compiler. The executable is generated by invoking make from the FUMP directory delivered with the software. The METIS libraries that fump requires are delivered with the software.

Code Execution

If a single processor job is required, one simply need type the name of the exeucutable, nphase, to a UNIX shell prompt. NPHASE-PSU is instrumented with mpi for inter-processor communication. The software is delivered with MPICH libraries. Accordingly, if more than one processor is required, mpirun is used. On most production

LINUX cluster systems interactive invokation of mpirun is not allowed, rather, the user must build a submit script, and submit the job to a queueing system such as PBS.

A typical run script for execution of NPHASE-PSU on the banyan cluster at Naval Surface Warfare Center Carderrock Division is included in the tutorial section below. The input files (nphase.dat, unphase.grid*xxx* and nphase_restart_in*xxx*) must be located in the directory from where the executable is invoked. Output files (discussed in next section) are written to this directory as well.

Postprocessing

Figure 12 illustrates the back end for NPHASE-PSU V3.0.

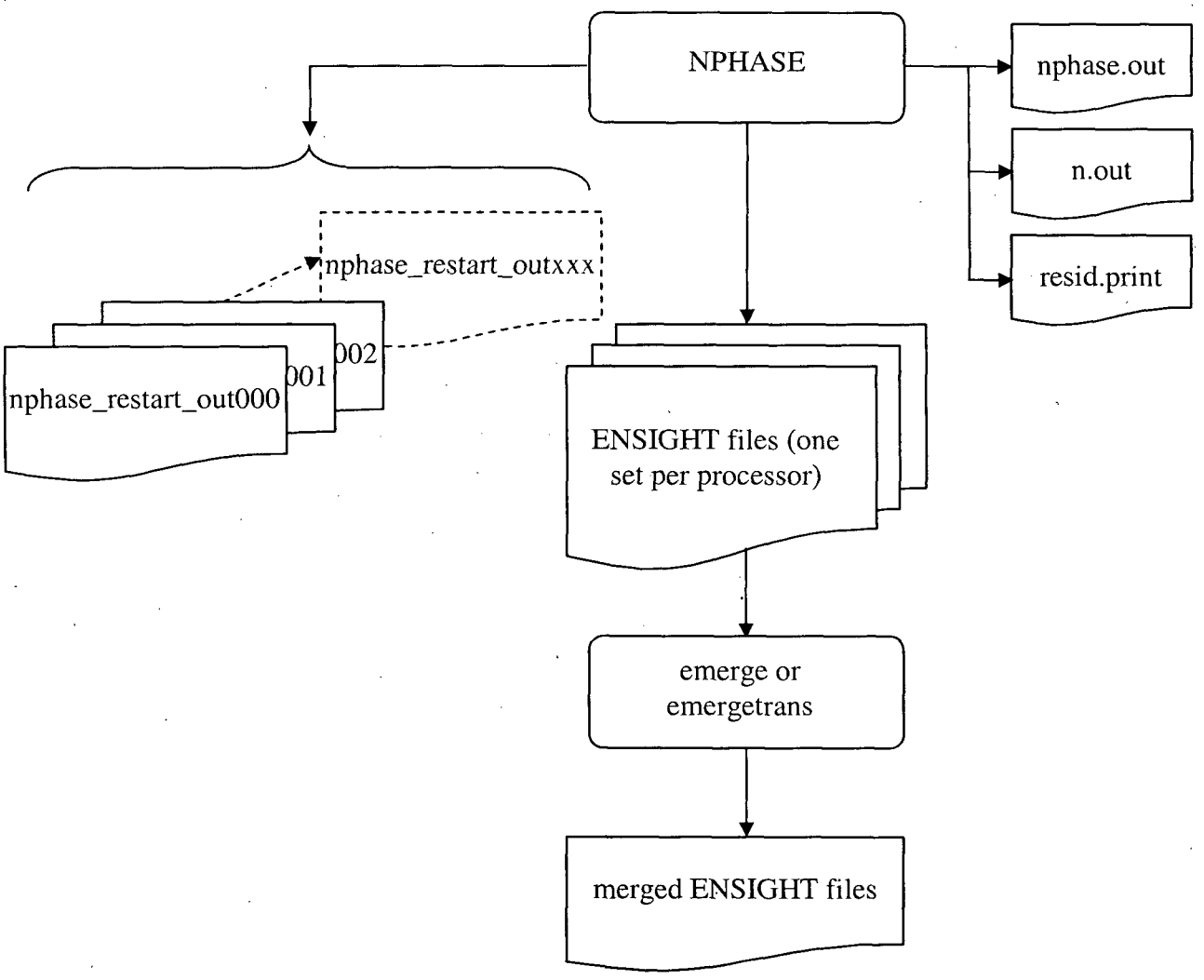

Figure 12. Sketch of back end for NPHASE

As with the front end the backend now supports arbitrary polyhedra. Specifically, ENSIGHT GOLD format is used to write the output files. ENSIGHT Versions 8.0 and later will read and display these files.

Output Files

There are five classes of output files to NPHASE-PSU. The first is the standard ASCII printed output file, nphase.out, an example of which is included in Figure 13. This contains an echo of the input, residual history, and, if enabled in backend, printed field data for single processor jobs. (these print_nodal commands are commented out in backend.c since it is unlikely that a user would ever wish to obtain a printed output for an unstructured domain).

```
  NN      NN  PPPPPPPP   HH     HH  AAAAAAAA  SSSSSSSS  EEEEEEEEE
  NNN     NN  PP    PPP  HH     HH  AAAAAAAA  SSSSSSSS  EEEEEEEEE
  NNNN    NN  PP     PP  HH     HH  AA     AA SS        EE
  NN NN   NN  PP    PPP  HH     HH  AA     AA SS        EE
  NN  NN  NN  PPPPPPPP   HHHHHHHHH  AAAAAAAAA SSSSSSSS  EEEE
  NN   NN NN  PP         HH     HH  AA     AA        SS EE
  NN    NNNN  PP         HH     HH  AA     AA        SS EE
  NN     NNN  PP         HH     HH  AA     AA SSSSSSSS  EEEEEEEEE
  NN      NN  PP         HH     HH  AA     AA SSSSSSSS  EEEEEEEEE


 NPHASE: A COMPUTER PROGRAM FOR THE PREDICTION OF MULTIFIELD
         FLOWS WITH MASS, MOMENTUM AND ENERGY TRANSFER

  Developed by: Rob Kunz
                Steve Antal
  Copyright 2006 Penn State University


 *** entered pre_author: reading nphase.dat input ***


 *** exiting pre_author: finished pass 1 on input file ***


 *** entered pre_author2: reading nphase.dat input ***


 *** exiting pre_author2: finished pass 1 on input file ***


 *** entered author: reading nphase.dat input file ***

iterations to perform 100
number of fields 1
produce ensight output
restart file write frequency 100
dont perform wall match logic
inlet patch 1 0
1.0 0. 0. 1.  1. 0.1 0.1 0. 0.
pressure patch 1 0
0. 1. 1. 0.1 0.1 0. 0.
turbulent flow high reynolds number k epsilon
constant fluid molecular viscosity 1.0e-2
constant fluid density 1.0
function entry/exit echo off
solversweepsforu 3
solversweepsforv 3
solversweepsforw 3
solversweepsfork 3
solversweepsfore 3
solversweepsforp 6
solver choice for velocity components jacobiuvw
solver choice for pressure petsc
parallel strategy for pressure corrector: matrixlevel
initialize u field   1.
initialize k field   .1
initialize e field   .1

 *** exiting author: finished 2nd pass on input file ***

  about to exit front end

 iter fld   ru        rv        rw        rp        ra        rh        rk        re        gmass     ws/cif
    1   1 1.996e-03 0.000e+00 0.000e+00 1.138e-02 0.000e+00 0.000e+00 6.205e-03 2.659e-01 -7.525e-04 5.381e-05
    2   1 1.968e-03 1.736e-04 1.733e-04 9.678e-03 0.000e+00 0.000e+00 5.294e-03 5.726e-02 -1.426e-03 5.415e-05
    3   1 1.416e-03 1.886e-04 1.883e-04 2.501e-02 0.000e+00 0.000e+00 3.857e-03 4.036e-02 -1.723e-03 5.366e-05
    4   1 1.095e-03 1.614e-04 1.610e-04 2.796e-02 0.000e+00 0.000e+00 3.151e-03 3.965e-02 -1.837e-03 5.381e-05
    5   1 8.819e-04 1.336e-04 1.332e-04 5.263e-02 0.000e+00 0.000e+00 2.715e-03 3.441e-02 -1.635e-03 6.922e-05
```

Figure 13. Sample nphase.out file

The second output file that is generated is standard output + standard error conventionally redirected to n.out, an example of which is included in Figure 13. This file contains a summary of the grid topology, front end progress, iteration history, user defined outputs, and summary flow rate and CPU performance information.

```
 *** begin execution of nphase ***

   1 processor run


    125058 Elements with Material ID (MID) =   1 exist in model
 nvert=40670 nnode=125058 nfedge=866444 nface=267148
    inlet.nbcfedge=   1600,     inlet.nbcface=    400,     inlet.nbcfaceid=     1
 farfield.nbcfedge=      0,  farfield.nbcface=      0,  farfield.nbcfaceid=     0
   outlet.nbcfedge=      0,    outlet.nbcface=      0,    outlet.nbcfaceid=     0
specified.nbcfedge=      0, specified.nbcface=      0, specified.nbcfaceid=     0
partition.nbcfedge=      0, partition.nbcface=      0, partition.nbcfaceid=     0
    cyclic.nbcfedge=      0,    cyclic.nbcface=      0,    cyclic.nbcfaceid=     0
 pressure.nbcfedge=   4260,  pressure.nbcface=   1420,  pressure.nbcfaceid=     1
     wall.nbcfedge=  29948,      wall.nbcface=   8916,      wall.nbcfaceid=     1
  porwall.nbcfedge=      0,   porwall.nbcface=      0,   porwall.nbcfaceid=     0
  intwall.nbcfedge=      0,   intwall.nbcface=      0,   intwall.nbcfaceid=     0
 symmetry.nbcfedge=      0,  symmetry.nbcface=      0,  symmetry.nbcfaceid=     0
     nobc.nbcfedge=      0,      nobc.nbcface=      0,      nobc.nbcfaceid=     0

 in read_grid_unstruct: renumbering MID range from 0 to   0


 finished reading grid in read_grid_unstruct

 *(xti+ifield+stride)  0.0000000000000e+00
 number of tets     in model                        =     88258
 number of hexs     in model                        =      8000
 number of prisms   in model                        =     28400
 number of pyramids in model                        =       400
 number of non-simplicial volume elements in model =         0
 Number of internal solid faces = 0
   about to exit front end

   iter fld   ru        rv        rw        rp        ra        rh        rk        re        gmass     ws/cif
      1   1 1.998e-03 0.000e+00 0.000e+00 1.138e-02 0.000e+00 0.000e+00 6.205e-03 2.659e-01 -7.525e-04 5.345e-05
      2   1 1.958e-03 1.736e-04 1.733e-04 9.678e-03 0.000e+00 0.000e+00 5.294e-03 5.726e-02 -1.426e-03 5.276e-05
      3   1 1.416e-03 1.886e-04 1.883e-04 2.501e-02 0.000e+00 0.000e+00 3.857e-03 4.036e-02 -1.723e-03 5.275e-05
      4   1 1.095e-03 1.614e-04 1.610e-04 2.796e-02 0.000e+00 0.000e+00 3.151e-03 3.965e-02 -1.837e-03 5.278e-05
      5   1 8.819e-04 1.336e-04 1.332e-04 5.263e-02 0.000e+00 0.000e+00 2.715e-03 3.441e-02 -1.835e-03 5.288e-05
      6   1 7.288e-04 1.156e-04 1.153e-04 1.691e-02 0.000e+00 0.000e+00 2.403e-03 2.881e-02 -1.195e-03 5.267e-05
 ------------------------------------------------------------------------------------------------------------------
     94   1 7.578e-05 2.106e-05 2.102e-05 1.367e-04 0.000e+00 0.000e+00 3.100e-04 5.024e-04  1.039e-05 6.372e-05
     95   1 7.483e-05 2.079e-05 2.074e-05 1.274e-04 0.000e+00 0.000e+00 3.064e-04 4.932e-04  9.867e-06 6.403e-05
     96   1 7.389e-05 2.052e-05 2.048e-05 1.202e-04 0.000e+00 0.000e+00 3.028e-04 4.842e-04  9.483e-06 6.150e-05
     97   1 7.296e-05 2.026e-05 2.021e-05 1.152e-04 0.000e+00 0.000e+00 2.994e-04 4.754e-04  9.212e-06 6.425e-05
     98   1 7.204e-05 2.000e-05 1.996e-05 1.123e-04 0.000e+00 0.000e+00 2.959e-04 4.669e-04  9.058e-06 6.248e-05
     99   1 7.114e-05 1.975e-05 1.970e-05 1.113e-04 0.000e+00 0.000e+00 2.926e-04 4.585e-04  9.011e-06 6.387e-05
    100   1 7.026e-05 1.950e-05 1.946e-05 1.119e-04 0.000e+00 0.000e+00 2.893e-04 4.504e-04  9.053e-06 6.437e-05

 writing output restart file(s) at iter = 100

 finished writing output restart file(s) at iter = 100


 total wall secs elapsed =  7.22058033e+02
 total cpu  secs used by all processors =  7.20080000e+02
 total elements on all processors =     125058
 cpu  secs/element/iteration =  5.75796830e-05
 wall secs/element/iteration =  5.77378523e-05
   about to enter back end

 writing ensight output file(s)
 Finished writing ensight output file(s)

 writing ensight output file(s)
 finished writing ensight output file(s)

 Number of Inlet Boundaries = 1
 Area and Average Pressure for Inlet Boundary ID 0 = 1 m**2, 0 Pa
 Mass Flow for Field 0 Through Inlet Boundary ID 0 = -1 Kg/s
 Mass Flow for Field 0 Through All Inlet Boundaries = -1 Kg/s

 Number of Pressure Boundaries = 1
 Area and Average Pressure for Pressure Boundary ID 0 = 1 m**2, 0 Pa
 Mass Flow for Field 0 Through Pressure Boundary ID 0 = 1.00001 Kg/s
 Mass Flow for Field 0 Through All Pressure Boundaries = 1.00001 Kg/s

 Done with boundary mass flow print

 ifield,(massin-massout)/massin:         0  9.0531670263739e-06
 Mass avg inlet velocity : -1.0000000000000e+00
 Porous wall flux sum (kg/s):  0.0000000000000e+00


 *** ending execution of nphase ***
```

Figure 14. Sample n.out file

The third output file that is generated is resid.print which is text file containing only the residual history for convenient plotting. An example is included in Figure 15. Residuals in the code are not "true" residuals, rather they are the RMS of $\Delta\phi$, the change in value of the variable solved for at a given iteration. (Note that this is the value solved for by NPHASE-PSU, see equation (75) in the theory manual.). The columns in resid.print (and the residual prints in nphase.out and n.out for that matter) are column 1: iteration number; column 2: field number (there will be a residual for each field [say liquid and gas]; columns 3-6: RMS(Δu), RMS(Δw), RMS(Δw), RMS(Δp); For multi-field simulations column 7 contains the volume fraction residual for each field, RMS($\Delta\alpha$); For diabatic simulations column 8 contains the enthalpy residual, RMS(Δh); For two-equation turbulence simulations columns 9 and 10 contains the k and ε residuals, RMS(Δk), RMS($\Delta\varepsilon$) (or q and ω, k and ω, for other turbulence models); column 10 contains the wall-clock seconds per (cell*iteration*field) for that iteration.

1	1	1.996e-03	0.000e+00	0.000e+00	1.138e-02	0.000e+00	0.000e+00	6.205e-03	2.659e-01	-7.525e-04
2	1	1.968e-03	1.736e-04	1.733e-04	9.678e-03	0.000e+00	0.000e+00	5.294e-03	5.726e-02	-1.426e-03
3	1	1.416e-03	1.886e-04	1.883e-04	2.501e-02	0.000e+00	0.000e+00	3.857e-03	4.036e-02	-1.723e-03
4	1	1.095e-03	1.614e-04	1.610e-04	2.796e-02	0.000e+00	0.000e+00	3.151e-03	3.965e-02	-1.837e-03
5	1	8.819e-04	1.336e-04	1.332e-04	5.263e-02	0.000e+00	0.000e+00	2.715e-03	3.441e-02	-1.635e-03
6	1	7.288e-04	1.156e-04	1.153e-04	1.691e-02	0.000e+00	0.000e+00	2.403e-03	2.881e-02	-1.195e-03
7	1	6.596e-04	1.026e-04	1.023e-04	1.255e-02	0.000e+00	0.000e+00	2.164e-03	2.407e-02	-7.996e-04
8	1	6.206e-04	9.439e-05	9.413e-05	3.820e-03	0.000e+00	0.000e+00	1.977e-03	2.025e-02	-2.927e-04
9	1	5.890e-04	8.817e-05	8.796e-05	1.047e-03	0.000e+00	0.000e+00	1.826e-03	1.719e-02	1.957e-04
10	1	5.605e-04	8.313e-05	8.297e-05	2.504e-03	0.000e+00	0.000e+00	1.703e-03	1.473e-02	6.738e-04
11	1	5.347e-04	7.913e-05	7.901e-05	1.099e-02	0.000e+00	0.000e+00	1.600e-03	1.275e-02	1.061e-03
12	1	4.976e-04	7.523e-05	7.513e-05	9.648e-02	0.000e+00	0.000e+00	1.512e-03	1.113e-02	7.315e-04
13	1	4.812e-04	7.881e-05	7.868e-05	6.256e-03	0.000e+00	0.000e+00	1.436e-03	9.803e-03	4.853e-04
14	1	4.669e-04	7.254e-05	7.225e-05	3.667e-03	0.000e+00	0.000e+00	1.370e-03	8.714e-03	2.734e-04
15	1	4.548e-04	6.911e-05	6.875e-05	1.714e-03	0.000e+00	0.000e+00	1.311e-03	7.812e-03	9.532e-05
16	1	4.404e-04	6.653e-05	6.614e-05	8.748e-04	0.000e+00	0.000e+00	1.259e-03	7.060e-03	-6.927e-05
17	1	4.198e-04	6.422e-05	6.381e-05	3.219e-03	0.000e+00	0.000e+00	1.213e-03	6.426e-03	-2.094e-04
18	1	3.951e-04	6.193e-05	6.151e-05	1.189e-02	0.000e+00	0.000e+00	1.170e-03	5.889e-03	-2.697e-04

Figure 15. Sample resid.print file

The fourth set of output files are solution restart files. NPHASE-PSU always generates a restart file for each processor after execution completion (and optionally at intermediate iterations/timesteps as defined in "Control Commands" section below). These output files conform to the naming convention nphase_restart_out*xxx* where *xxx* is the 3-digit processor identifier (range from 000 to 999). Solution restarts are invoked by 1) copying each of these output restart files to a corresponding input restart file (i.e., mv nphase_restart_out000 nphase_restart_in000) and then 2) activating the keyword "initialize run with restart file" in nphase.dat.

The fifth set of files are ENSIGHT output files. For steady state cases, each processor generates at least a geometry/grid file: engold.geo.*xxx*, pressure and velocity scalar files: engold.Esca.p00.*xxx*, engold.Esca.u*nn.xxx*, engold.Esca.v*nn.xxx*, and engold.Esca.w*nn.xxx*, where *xxx* is the processor number (0-999) and *nn* are the fields present (00-99). (Since NPHASE-PSU employs a single pressure formulation only engold.Esca.p00.*xxx* is generated). In addition to velocity and pressure files, other ENSIGHT files are generated depending on problem type and user specification (keyword – see Control Commands section below). Specifically, volume fraction files are generated for multi-field simulations: engold.Esca.a*nn.xxx*; enthalpy files are generated for diabatic simulations: engold.Esca.h*nn.xxx*; turbulence quantity files are generated for turbulence

simulations: engold.Esca.k*nn.xxx* (always turbulent kinetic energy [independent of turbulence model]), engold.Esca.e*nn.xxx*(always turbulent dissipate rate, ε [independent of turbulence model]), engold.Esca.m*nn.xxx* (eddy viscosity).

emerge and emergetrans

Once NPHASE-PSU has completed there exist many ENSIGHT files in the working directory from which the case is executed. One runs emerge to merge the ENSIGHT files generated on the different processors into single ENSIGHT files for each variable. One runs emergetrans to merge the ENSIGHT files generated on the different processors into single ENSIGHT files for each variable at each timestep that the user has selected for outputting these files. emerge and emergetrans are delivered in software directory EMERGE. They are written in FORTRAN and some C (I/O functions) and are compiled using makefiles (makeem, makeemt) that expect pgf90 and gcc, for which these codes are guaranteed to compile and run. Other compilers are untested.

To run emerge, simply types "emerge" to the UNIX shell. The user will be prompted for the number of processors and the number of fields. Once emerge has completed, the following files are resident in the user's working directory: engold.geo, engold.Esca.p00, engold.Esca.u*nn*, engold.Esca.v*nn,* and engold.Esca.w*nn* (where *nn* are the fields present (00-99)). Other case specific merged files are also created (e.g., engold.Esca.a*nn*, engold.Esca.h*nn*, engold.Esca.k*nn*, etc...). Finally emerge generates velocity vector files: engold.Evec.uvw*nn*. All files generated appear in the file engold.case, an example of which appears in Figure 16. Upon execution of emerge the user needs to transfer the case file, and all of the files it points to, to the file system where he will run ENSIGHT. To run ENSIGHT, the case file is read in and each of the other files are automatically brought in.

```
# BOF:  engold.case

FORMAT

type:   ensight gold

GEOMETRY

model:  engold.geo

VARIABLE

scalar per element:        Pressure      engold.p00.Esca
scalar per element:        X-velocity_00 engold.u00.Esca
scalar per element:        Y-velocity_00 engold.v00.Esca
scalar per element:        Z-velocity_00 engold.w00.Esca
scalar per element:              TKE_00  engold.k00.Esca
scalar per element:              TDS_00  engold.e00.Esca
scalar per element:   Eddy-viscosity_00  engold.m00.Esca

vector per element:  Velocity-vector_00 engold.uvw00.Evec
# EOF:  engold.case
```

Figure 16. Example engold.case file

To run emergetrans, simply type "emergetrans" to the UNIX shell. The user will be prompted for the number of processors, the number of fields, whether the grid is stationary or moving, the first integer timestep counter, the last integer timestep counter and the interger timestep increment. emergetrans writes all output files to a subdirectory, TRANS, once emergetrans has completed, the following files are resident in TRANS: engold.geo (or engold.geo.*ttttt* (if grid is moving), engold.Esca.p00.*ttttt,* engold.Esca.u*nn.ttttt*, engold.Esca.v*nn.ttttt,* and engold.Esca.w*nn.ttttt* (where *nn* are the fields present (00-99), and *ttttt* is the integer timestep). Other case specific merged files are also created (e.g., engold.Esca.a*nn.ttttt*, engold.Esca.h*nn.ttttt*, engold.Esca.k*nn.ttttt*, etc...). Finally emergetrans generates velocity vector files: engold.Evec.uvw*nn.ttttt*. All files generated appear in the file TRANS/engold_transient.case, an example of which appears in Figure 17. Upon execution of emerge the user needs to transfer the case file and all of the files it points to the file system where he will run ENSIGHT. To run ENSIGHT, the case file is read in and each of the other files are automatically brought in. Upon execution of emergetrans the user needs to transfer the case file, and all of the files it points to, to the file system where he will run ENSIGHT. To run ENSIGHT, the case file is read in and each of the other files are automatically brought in.

```
# BOF:  engold_transient.case

FORMAT

type:   ensight

GEOMETRY

model:  engold.geo

VARIABLE

scalar per element: Pressure      engold.p00.Esca.******
scalar per element: X-velocity_00 engold.u00.Esca.******
scalar per element: Y-velocity_00 engold.v00.Esca.******
scalar per element: Z-velocity_00 engold.w00.Esca.******
vector per element: Velocity_00  0 gold.uvw000 vec.******
scalar per element: TKE_00        engold0k00.Esca.******
scalar per element: TDS_00        engold0e00.Esca.******
scalar per element: Mut_00        engold0m00.Esca.******

TIME
time set: 1
number of steps:    5
filename numbers:
          0          1          2          3          4
time values:
  0.000E+00  0.100E+01  0.200E+01  0.300E+01  0.400E+01
# EOF:  engold_transient.case
```

Figure 17. Example engold_transient.case file

Tutorials

Three tutorials are provided here. The first is a general familiarization case, the second and third are specific to the DARPA FDR program which co-funded this documentation. Each is delivered with the software and they unpack to ./NPHASE-PSU/TUTORIAL_1, ./NPHASE-PSU/TUTORIAL_HIPLATE, ./NPHASE-PSU/TUTORIAL_5415. Specifically, the necessary input files are provided for each tutorial: nphase.dat, cobalt.inp, cobalt.bc, and a few other utility files. The user can follow the tutorials below successively applying the NPHASE-PSU pre-processing, execution and postprocessing steps, starting with these files.

Tutorial Case 1: Turbulent, unsteady, arbitrary polyhedra two-body model

This purely notional case is for flow around a deformed sphere in the vicinity of a solid wall in a channel. The grid was generated by HARPOON and is hex dominant, but contains "hanging nodes" which gives rise to elements that NPHASE-PSU interprets as arbitrary n-sided-polyhera (i.e., not tetrahedral, hexehedra, prisms or pyramids). The run is turbulent and unsteady.

To start the tutorial the user needs to go to the TUTORIAL_1 directory. There are five files there when the software is unpacked: nphase.dat, run.nphase, cobalt.inp, cobalt.bc and mrf. The first step is to execute fump as illustrated in Figure 18. Here we have chosen 4 processors and no scaling of the geometry. This step generates the 8 files unphase.grid000, unphase.grid001, ..., unphase.grid007. Figure 19 shows the file nphase.dat. This is an unsteady run, with four timesteps. 5 inner iterations per timestep and an ENSIGHT output after every timestep.

```
fump
 ***.begin execution of fump ***.

enter number of processors for domain decomposition:
8
model will be decomposed into 4 partitions

enter scale factor:
1.0
grid will be scaled by 1

in problem_size

cobalt.inp is ascii

*file_type       0
ibc =  0, is a Wall_00 boundary
ibc =  1, is a Inlet_00 boundary
ibc =  2, is a Pressure_00 boundary
ndim,nzones,nbc       3       1       3
*nvert,*nface_cobalt,*ncell,maxppf,maxfpc =    40670   277884   125058       4       6
*nface =  267148, pressure.nbcface =     1420, pressure.nbcfedge =     4260
inlet.nbcface   =     400, inlet.nbcfedge   =     1600
farfield.nbcface =       0, farfield.nbcfedge =       0
```

Figure 18. fump run stream for TUTORIAL_1. Bold-black=user supplied, blue=code generated.

```
#case title:
#2-CELL HANGING NODES

iterations to perform 5

number of fields 1

time accurate simulation
temporal discretization momentum 1
physical timestep in seconds .1
number of physical timesteps 4
transient file write frequency 1

#initialize run with restart file

produce ensight output

dont perform wall match logic

inlet patch 1 0
1.0 0. 0. 1.  1. 0.1 0.1 0. 0.

pressure patch 1 0
0. 1. 1. 0.1 0.1 0. 0.

turbulent flow high reynolds number k epsilon

constant fluid molecular viscosity 1.e-3
constant fluid density 1

function entry/exit echo off

solver sweeps for u 3
solver sweeps for v 3
solver sweeps for w 3
solver sweeps for p 6
solver sweeps for k 3
solver sweeps for e 3

solver choice for velocity components jacobiuvw
solver choice for pressure petsc
parallel strategy for pressure corrector: matrixlevel

initialize u field  1.
initialize k field  .1
initialize e field  .1
```

Figure 19. nphase.dat file for TUTORIAL_1.

The file run.nphase is the job submit script that needs to be edited by the user to conform to the system he is running on. The delivered version appears in Figure 20. The user must edit the script to point to mpirun and the executable (highlighted in blue) on their system. Standard PBS attributes are included that presumably are machine independent.

The job is submitted to PBS using the qsub command: qsub run.nphase (unless there are other submit scripts available on the user's system). This 388,114 element 8 processor job runs in about 90 seconds on a modern LINUX cluster. The code generates many files, most of them ENSIGHT files, as can be verified by doing an "ls".

```
#! /bin/csh -f
#PBS -l nodes=8:ppn=1
#PBS -l walltime=96:00:00
#PBS -j oe

cd $PBS_O_WORKDIR

echo $PBS_O_WORKDIR
echo $PWD

set run = /home/rfk102/local/bin/empirun.pgi
$run /mnt/vault/rfk102/NPHASE/TO_NSWCCD/src/nphase >& n.out
```

Figure 20. run.nphase script for TUTORIAL_1.

The user can view nphase.out, n.out (which is the standard output and standard error redirect of the NPHASE-PSU run,) and resid.print. The next postprocessing step is to execute emerge which produces processor merged ENSIGHT output files of the final solution. The emerge run stream for this case is included in Figure 21. Once emerge is completed the user can migrate the merged data files and case files to the location he plans on postprocessing using ENSIGHT. These files are: engold.case, engold.geo, engold.w00.Esca, engold.v00.Esca, engold.uvw00.Evec, engold.u00.Esca, engold.p00.Esca, engold.m00.Esca, engold.k00.Esca, engold.e00.Esca. The user can also execute emergetrans for this case which produces processor merged ENSIGHT output files at every timestep. The emergetrans run stream for this case is included in Figure 22. Once emergetrans is completed the user can migrate the merged data files and case files to the location he plans on postprocessing using ENSIGHT. These files are the complete contents of the subdirectory TRANS.

```
emerge

      ********************
      egoldmerge assembly
      ********************

 enter number of partitions/processors
8
 enter number of fields
1
engold.geo.000
        1      49860
        1 numcells_tets(iproc)   =      2555
        1 numcells_hexs(iproc)   =     39823
        1 numcells_prisms(iproc) =       650
.
.
.
no WU file
 reading F-V2F files
no F_V2F file
 reading IBLANK files
no IBLANK file
 reading Temperature files
no Temperature file
 reading Aint files
no Aint
 re-reading velocity files
FORTRAN STOP
```

Figure 21. emerge run stream for TUTORIAL_1. Bold-black=user supplied, blue=code generated.

```
emergetrans

      ***********************************
      egoldmerge assembly for transients
      ***********************************


 enter number of partitions/processors
8
 enter number of fields
1
 enter:
 1) stationary mesh
 2) moving mesh
1
 enter the integer number of the first timestep
0
 enter the integer number of the  last timestep
4
 enter the integer timestep increment
1
 reading from nphase.out to determine which ensight
 transient files were written by run being postprocessed
 Ensight transient variable files to be merged
 Pressure
 X-velocity
.
.
.
no VW file
 reading WU files
no WU file
 reading F-V2F files
no F file
 reading IBLANK files
no I file
 reading Temperature files
no Temperature file
 reading Aint files
no Aint file
 re-reading velocity files
FORTRAN STOP
```

Figure 22. emergetrans run stream for TUTORIAL_1. Bold-black=user supplied, blue=code generated.

The second part of this tutorial involves modifying nphase.dat to run a steady state problem, and then running this problem to convergence, including doing a mid-run restart. To do this copy nphase.dat to nphase.dat.sav, so it can be recovered later. Next change the number of iterations to 250 and comment out the unsteady run keywords as shown (in blue) in Figure 23. Submit and run the job to completion of 250 iterations. To execute a restart comment in the "initialize run with restart file" keyword (as shown in green in Figure 23), and copy all of the nphase_restart_out*xxx* files to nphase_restart_in*xxx*. A convenient way of doing this is to build a small script, mrf, as included in the TUTORIAL_1 directory and as appears in Figure 24. Resubmit, running the code out to 500 iterations, and then run emerge again.

Figure 25 and 26 show the convergence history (TECPLOT used with resid.print as input) and a representative ENSIGHT visualization of the output of this steady state run.

```
case title:
2-CELL HANGING NODES

iterations to perform 250

number of fields 1

#time accurate simulation
#temporal discretization momentum 1
#physical timestep in seconds .1
#number of physical timesteps 4
#transient file write frequency 1

#initialize run with restart file

produce ensight output
```

Figure 23. modified nphase.dat file for TUTORIAL_1.

```
mv nphase_restart_out000 nphase_restart_in000
mv nphase_restart_out001 nphase_restart_in001
mv nphase_restart_out002 nphase_restart_in002
mv nphase_restart_out003 nphase_restart_in003
mv nphase_restart_out004 nphase_restart_in004
mv nphase_restart_out005 nphase_restart_in005
mv nphase_restart_out006 nphase_restart_in006
mv nphase_restart_out007 nphase_restart_in007
```

Figure 24. mrf script

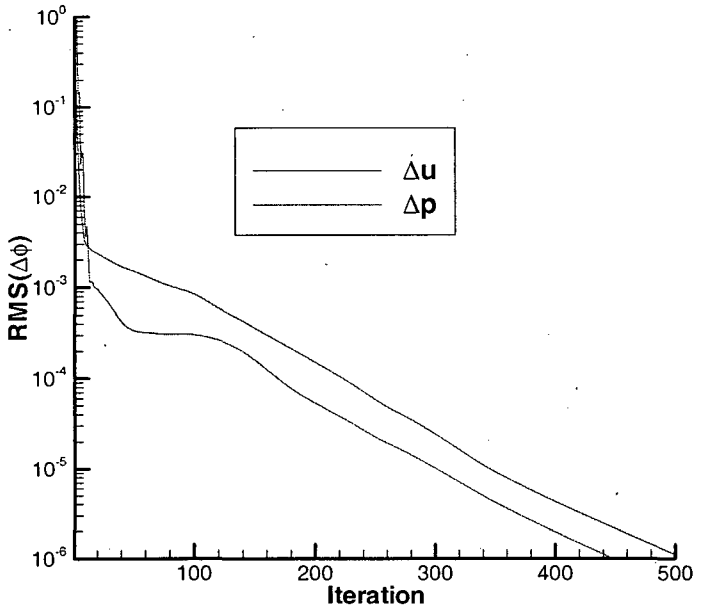

Figure 25. TECPLOT plot of residual history of steady TUTORIAL_1 simulation.

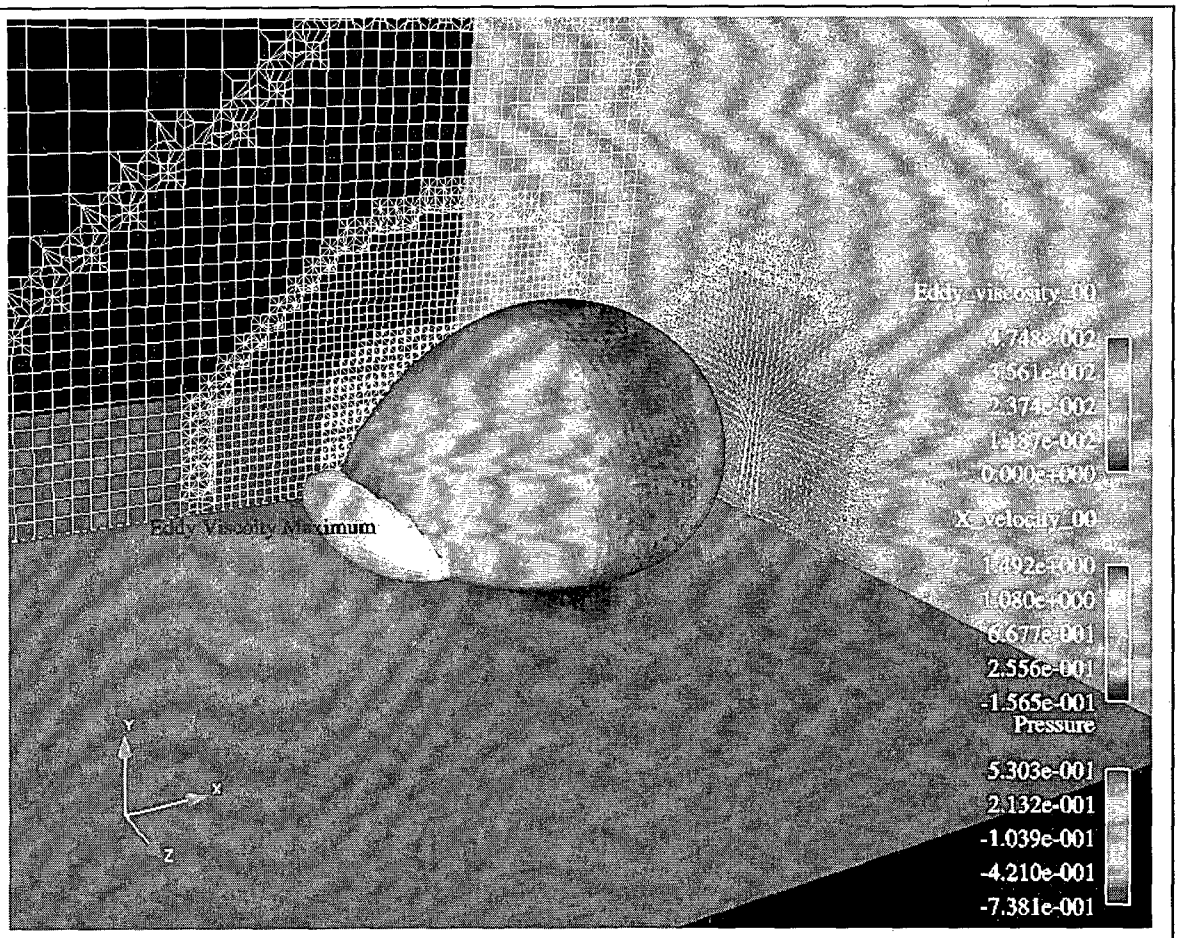

Figure 26. ENSIGHT visualization of steady TUTORIAL_1 simulation.

Tutorial Case 2: Multiphase HIPLATE Simulation

This case demonstrates the application of NPHASE-PSU to the HIPLATE configuration. The HIPLATE experimental programs were carried out by Ceccio and his colleagues at the Universty of Michigan (Sanders et al., 2006) under the DARPA Friction Drag Reduction program. Comparison of NPHASE-PSU to and calibration of NPHASE-PSU against these data sets were elements of the program (Kunz, et al., 2006, Kunz et al., 2007). HIPLATE is a very high Reynolds number plate configuration tested in the US Navy's Large Cavitation Channel, using microbubble and polymer drag reduction schemes.

Although NPHASE-PSU was validated against and/or applied to the entire spectrum of HIPLATE runs (five gas injection rates, three tunnel speeds, various injector port gas flow splits, with vs. without surfactant, two injector geometries, and two water tunnel test programs [HIPLATE I, II]), we here demonstrate a coarse grid (fast running) application corresponding to the January 2005 program deliverable (validation of code agains HIPLATE I data). Specifically, a 12 m/s, Q_{gas}=800 scfm case is set up and run. The physical model set used here is that "frozen" for the January 2006 deliverable to DARPA which was a validation study comparing NPHASE-PSU simulations to all HIPLATE I data.

To start the tutorial the user needs to go to the TUTORIAL_HIPLATE directory. There are four files there when the software is unpacked: nphase.dat, run.nphase, cobalt.inp, and cobalt.bc. The first step is to execute fump (as described in the *Preprocessing* section and *Tutorial_1* section above), using (here) 1 domain and a scaling factor of 1.0. This step generates a single grid+topology file, unphase.grid000. Figure 27 Figure 34 shows the file nphase.dat file used for this tutorial. The keywords "employ model hiplate modeling" are included to instruct nphase to execute the function hiplate_output.c each iteration which prints skin friction data (and, if commented in, other data) to standard output (redirected to n.out in run.nphase). (See the section entitled *Building User Specific/Case Specific Postprocessing* for details on how to modify or adapt this kind of output for different output or simulation case.)

First the code has to be run in single phase mode to generate the comparison flat plate skin friction values at the 6 axial locations on the HIPLATE where shear stress measurements were made. To do this, the user needs to modify the green highlighted sections as follows: 1) change number of iterations to 500, 2) change porous wall injection velocity from 9.0476 to 0.0000 and the porous wall permeability from 0.7 to 0.0., 3) comment out the relaxation factor lines (allow for defaults). So the code is being run in full 2-phase mode but there is no gas present or injected to it returns a single phase convergence and results.

This single phase job is run interactively (since its only one processor) or submitted to PBS using the qsub command: qsub run.nphase (unless there are other submit scripts available on the user's system). This single phase job processor job executes 500 iterations in about 55 wall seconds on a modern LINUX cluster. Figure 28Figure 36 shows a section of the n.out file for this single phase run that includes the computed C_f distribution at the six axial HIPLATE stations. To run the microbubble drag reduction case, the nphase.dat file is used as it appears in Figure 27. A total of 20,000 iterations are selected along with the conservative under-relaxation factors that appear.

```
iterations to perform 20000
number of fields 2

produce ensight output

restart file write frequency 1000
dont perform wall match logic

employ hiplate modeling

gravity vector 0. 9.81 0.
employ modified pressure
reference density for modified pressure 1

overwrite inlet patch boundary conditions on restart
overwrite porous wall patch boundary conditions on restart
overwrite pressure patch boundary conditions on restart
overwrite_gas_molecular_viscosities 200 1.5e+3

inlet patch 2 0
12.0 0. 0. 1000. 0.999999 1.33e-7 1.76e-6 0. 1.
12.0 0. 0. 1.    0.000001 0.      0.      0. 1.

porous wall patch 2 0
9.0476 0. 0. 1000. 0.000001 1.33e-7 1.76e-6 0. 0.7  1.
9.0476 0. 0. 1.    0.999999 0.      0.      0. 0.7  1.

pressure patch 2 0
0.0  1000.  0.999999 1.33e-7 1.76e-6  0.
0.0  1.     0.000001 0.      0.       0.

constant fluid density 1000. 1.
constant fluid molecular viscosity 1.e-3 1.5e-5
constant fluid surface tension .072 0.

turbulencemodelforeachfield 1 0

function entry/exit echo off
adiabatic flow

solver sweeps for u 3
solver sweeps for v 3
solver sweeps for a 3
solver sweeps for p 10
solver sweeps for tke 3
solver sweeps for tds 3

solver choice for velocity components jacobiuvw
solver choice for pressure petsc
parallel strategy for pressure corrector: matrixlevel
employ rhie chow for dispersion terms

relaxation factor for u .05 .05
relaxation factor for v .05 .05
relaxation factor for a .05 .05

initialize u field 12. 12.
initialize v field 0. 0.
initialize w field 0. 0.
initialize a field .999999 0.000001
initialize tke field 1.33e-7 0.
initialize tds field 1.76e-6 0.
```

```
interfacial area transport model for each field 0 1
interfacial area coalescence model for each field 0 3
initialize interfacial area using volume fraction and characteristic length
interfacial area feeds back into bubble diameter frequency 100
constant field characteristic diameter 99. .000400
interfield drag models 1
1 2 6 .8
bubble cluster drag modification
interfield nondrag models 5
1 2 2 5 10.
1 2 2 7 1. 2. -2.0 0.5 2.
1 2 3 3 .1
1 2 1 5 .075
1 2 5 3 1.
```

Figure 27. nphase.dat file for TUTORIAL_HIPLATE.

```
  iter fld   ru          rv          rw          rp          ra          raint       rh          rk …
     1  1 3.580e-03 4.631e-01 0.000e+00 5.767e+03 2.210e-08 0.000e+00 0.000e+00 1.104e-05 …
     1  2 1.005e+00 1.672e-14 0.000e+00 5.767e+03 2.139e-08 3.208e-04 0.000e+00 0.000e+00 …
cf1,cf2,cf3,cf4,cf5,cf6  2.57927225e-01  2.57927225e-01  2.57927225e-01  2.57927225e-01 …
clipcount0         64
     2  1 7.490e-02 9.948e-02 0.000e+00 1.157e+03 3.543e-08 0.000e+00 0.000e+00 1.324e-03 …
     2  2 5.775e-01 9.039e-02 0.000e+00 1.157e+03 3.467e-08 5.207e-04 0.000e+00 0.000e+00 …
cf1,cf2,cf3,cf4,cf5,cf6  4.48393267e+00  5.93114814e+00  6.51944986e+00  6.64137946e+00 …
```

```
   499  1 8.332e-06 1.609e-07 0.000e+00 3.750e-02 2.028e-06 0.000e+00 0.000e+00 2.511e-06 …
   499  2 2.803e-02 2.212e-03 0.000e+00 3.750e-02 9.705e-07 1.457e-02 0.000e+00 0.000e+00 …
cf1,cf2,cf3,cf4,cf5,cf6  1.59247143e+02  1.49084091e+02  1.39655412e+02  1.36212585e+02  …
   500  1 8.156e-06 1.578e-07 0.000e+00 3.754e-02 5.770e-07 0.000e+00 0.000e+00 2.460e-06 …
   500  2 2.599e-02 2.010e-03 0.000e+00 3.754e-02 7.361e-07 1.105e-02 0.000e+00 0.000e+00 …
cf1,cf2,cf3,cf4,cf5,cf6  1.59247144e+02  1.49084098e+02  1.39655468e+02  1.36212698e+02 …

final cf1,cf2,cf3,cf4,cf5,cf6  1.59247144e+02  1.49084098e+02  1.39655468e+02  1.36212698e+02 …
avgcf cf1,cf2,cf3,cf4,cf5,cf6  1.56144328e+02  1.41949562e+02  1.39703053e+02  1.38348503e+02 …
stdv  cf1,cf2,cf3,cf4,cf5,cf6  1.36066548e+01  1.79052552e+01  1.66656309e+01  1.97886396e+01 …
```

Figure 28. Snippets from standard output (n.out) for single phase TUTORIAL_HIPLATE simulation, illustrating case specific skin friction coefficient output.

```
  iter fld   ru          rv          rw          rp          ra          raint       rh          rk …
     1  1 2.007e-04 1.944e-02 0.000e+00 7.098e+05 2.068e-03 0.000e+00 0.000e+00 1.102e-05 …
     1  2 7.272e-02 1.021e-02 0.000e+00 7.098e+05 2.067e-03 9.201e+03 0.000e+00 0.000e+00 …
cf1,cf2,cf3,cf4,cf5,cf6  2.57927225e-01  2.57927225e-01  2.57927225e-01  2.57927225e-01 …
clipcount0         85
     2  1 4.648e-01 2.057e-01 0.000e+00 6.805e+05 1.970e-03 0.000e+00 0.000e+00 1.373e-03 …
     2  2 7.036e-02 3.113e-03 0.000e+00 6.805e+05 1.969e-03 7.497e+03 0.000e+00 0.000e+00 …
cf1,cf2,cf3,cf4,cf5,cf6  4.53209563e+00  5.96370783e+00  6.54783079e+00  6.66829586e+00 …
```

```
 19999  1 4.116e-04 1.186e-04 0.000e+00 1.694e+00 1.038e-03 0.000e+00 0.000e+00 7.449e-04 …
 19999  2 3.735e-03 1.083e-03 0.000e+00 1.694e+00 1.036e-03 1.053e+02 0.000e+00 0.000e+00 …
cf1,cf2,cf3,cf4,cf5,cf6  6.61059220e+01  1.09891416e+02  1.05604004e+02  1.06358347e+02 …
 20000  1 4.094e-04 1.195e-04 0.000e+00 1.696e+00 1.001e-03 0.000e+00 0.000e+00 7.493e-04 …
 20000  2 3.759e-03 1.068e-03 0.000e+00 1.696e+00 9.998e-04 1.022e+02 0.000e+00 0.000e+00 …
cf1,cf2,cf3,cf4,cf5,cf6  6.62048642e+01  1.09891117e+02  1.05605248e+02  1.06375858e+02 …

final cf1,cf2,cf3,cf4,cf5,cf6  6.62048642e+01  1.09891117e+02  1.05605248e+02  1.06375858e+02 …
avgcf cf1,cf2,cf3,cf4,cf5,cf6  6.66569680e+01  1.09960776e+02  1.05281584e+02  1.05289948e+02 …
stdv  cf1,cf2,cf3,cf4,cf5,cf6  3.74098782e-01  7.38330986e-02  6.06662852e-01  7.79479169e-01 …
```

Figure 29. Snippet from standard output (n.out) for two-phase TUTORIAL_HIPLATE simulation, illustrating case specific skin friction coefficient output.

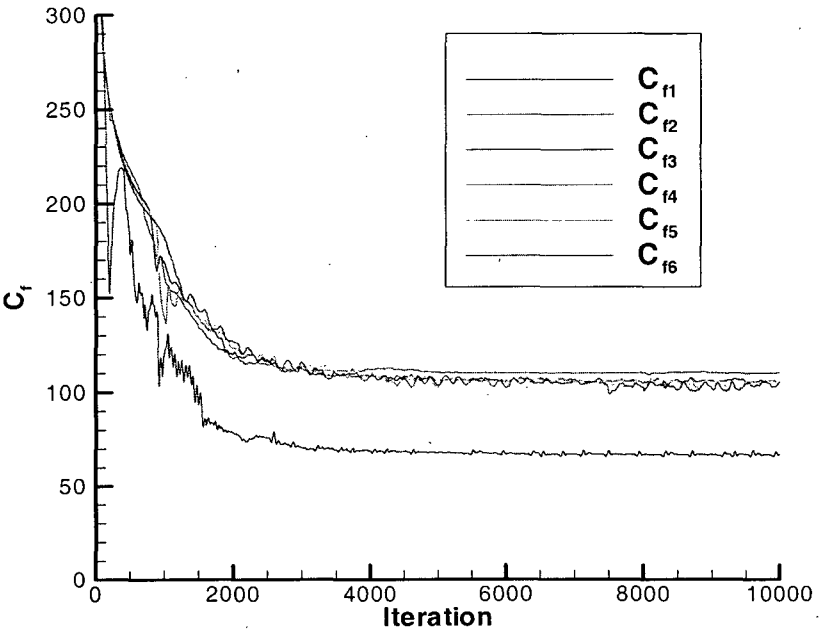

Figure 30. Predicted skin friction coefficient vs. iteration at the six HIPLATE stations for the two-phase TUTORIAL_HIPLATE simulation.

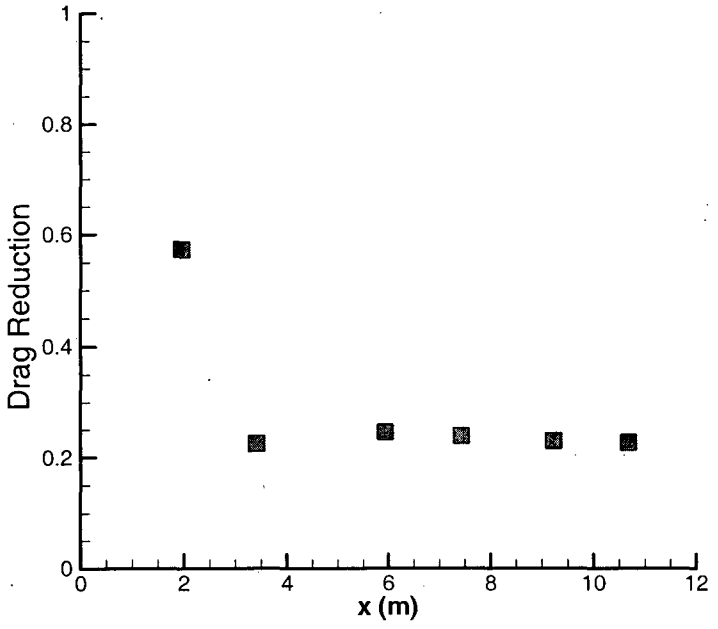

Figure 31. Predicted skin friction coefficient vs. x at the six HIPLATE stations for the two-phase TUTORIAL_HIPLATE simulation.

Figure 29 shows a section of the n.out file for the two-phase microbubble drag reduction run that includes the computed C_f distribution at the six axial HIPLATE stations.

Figure 30 shows the computed 2-phase skin friction iteration history. This solution converged to the final C_f values in about 5000 iterations. Figure 31 shows the predicted drag reduction vs. x for this case. These results are nearly identical to the January 2006 submission (grid is coarser here). Figure 32 shows contours of predicted gas volume fraction wit the vertical axis scaled by a factor of 100 in order that the thin microbubble layer can be viewed.

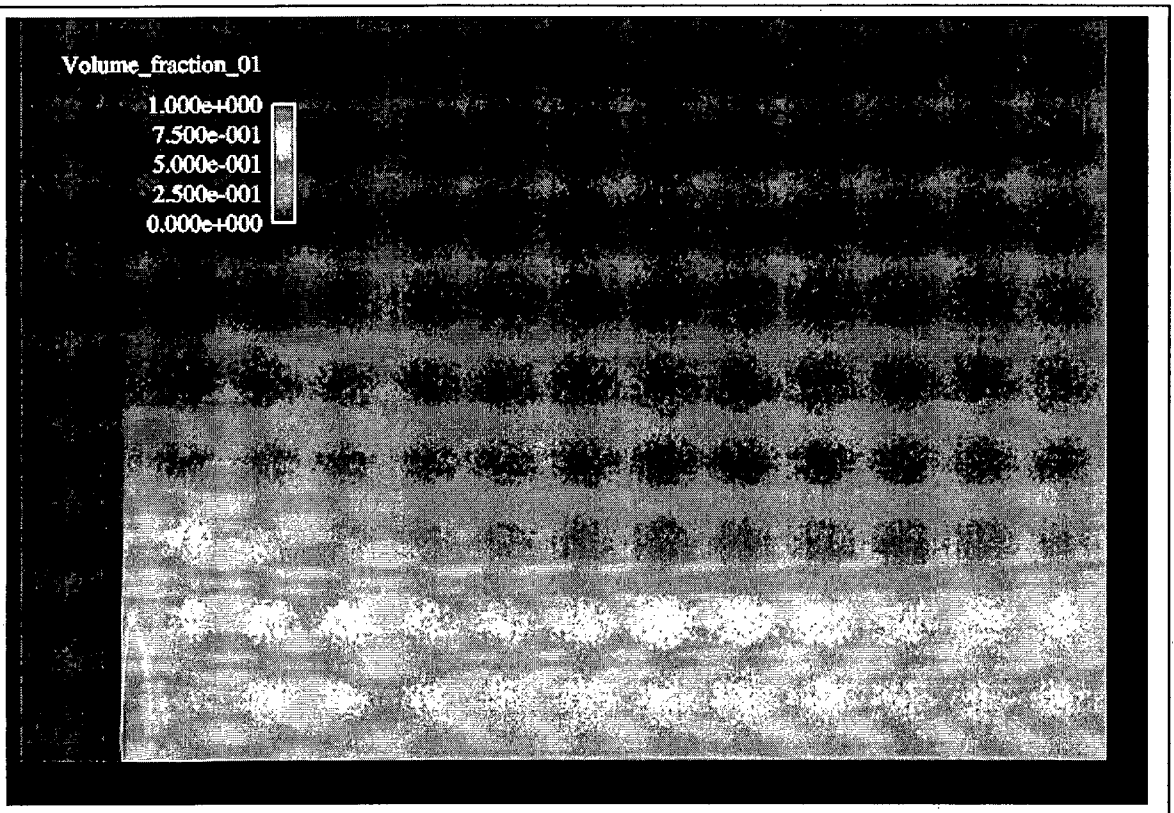

Figure 32. Predicted contours of gas volume fraction for the two-phase TUTORIAL_HIPLATE simulation. Vertical axis scaled by a factor of 100.

Tutorial Case 3: Multiphase 5415 Simulation

This case demonstrates the application of NPHASE-PSU to a Navy relevant configuration in 1-phase and 2-phase modes. The geometry is the 5415 model, a hull form representative of the Arleigh-Burke class destroyer. This model has been extensively tested at the Naval Surface Warfare Center Carderock Division (see http://www.dt.navy.mil/hyd/sur-shi-mod/index.html, for example.) The grid was generated using ICEM-CFD and is a standard tetra+prism mesh. Specifically, the mesh contains 1438852 elements of which 463837 are prisms extruded in (nominally) five layers from the triangulated hull surface. This comparatively coarse mesh is suitable for wall function turbulence modelling. The hull surface incorporates a standard solid wall patch and a porous wall patch, aft of the bow dome, from which gas is injected. Figure 33 shows an illustration of the mesh used.

To start the tutorial the user needs to go to the TUTORIAL_5415 directory. There are six files there when the software is unpacked: nphase.dat.1phase, nphase.dat.2phase, run.nphase, cobalt.inp, cobalt.bc and mrf. The first step is to execute fump (as described in the *Preprocessing* section and *Tutorial_1* section above), using (here) 32 domains and a scaling factor of .001 (to convert the grid from mm to m.) This step generates the 32 files unphase.grid000, unphase.grid001, ..., unphase.grid0031. Figure 34 shows the file nphase.dat.1phase. This file corresponds to a steady single phase run. The keywords "employ model 5415 modeling" are included to instruct nphase to execute the function model5415_output.c each iteration which generates a wetted area and a net vehicle drag coefficient printout to standard output (redirected to n.out in run.nphase). (See the section entitled *Building User Specific/Case Specific Postprocessing* for details on how to modify or adapt this kind of output for different output or simulation case.) A freestream velocity of 2.2134 m/s is specified (model scale) and the k-ε model is selected.

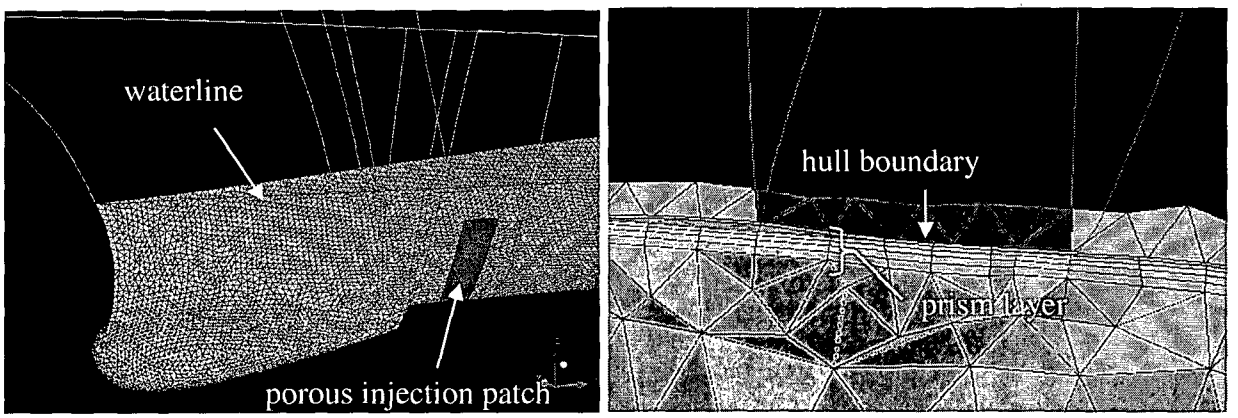

Figure 33. Views of mesh employed for 5415 tutorial

First the user should copy nphase.dat.1phase to nphase.dat. The job is submitted to PBS using the qsub command: qsub run.nphase (unless there are other submit scripts available on the user's system). This 32 processor job executes 1000 iterations in about 2800 wall seconds on a modern LINUX cluster (banyan.dt.navy.mil). As before, the user can view nphase.out, n.out, and resid.print. Then emerge is run and the user can migrate the

merged data files and case files to the location he plans on postprocessing using ENSIGHT. These files are: engold.case, engold.geo, engold.w00.Esca, engold.v00.Esca, engold.uvw00.Evec, engold.u00.Esca, engold.p00.Esca, engold.m00.Esca, engold.k00.Esca, engold.e00.Esca.

Figure 35 shows the convergence history for this case. Figure 36 shows a section of the n.out file which illustrates the computed frontal wetted area and C_D computation.

```
case title:
5415

iterations to perform 1000

employ model 5415 modeling

number of fields 1

#initialize run with restart file

produce ensight output
restart file write frequency 100

dont perform wall match logic
#read wall proximity from file
gravity vector 0. 0. -9.81
employ modified pressure
reference density for modified pressure 1.
simple hydrostatic pressure treatment

overwrite inlet patch boundary conditions on restart

inlet patch 1 0
2.2134 0. 0. 1000. 1. .002939 .02618 0. 0.

pressure patch 1 0
0.0  1000.  1. .002939 .02618  0. 0.

porous wall patch 1 0
0.0000 0. 0. 1000. 1. .002939 .02618 0. 0.7  0.

turbulent flow high reynolds number k epsilon

constant fluid molecular viscosity 1.e-3
constant fluid density 1000.

spatial discretization momentum 2

function entry/exit echo off

solversweepsforu 3
solversweepsforv 3
solversweepsforw 3
solversweepsfork 3
solversweepsfore 3

solver choice for velocity components jacobiuvw
solver choice for pressure petsc
parallel strategy for pressure corrector: matrixlevel
petscprintnorm

initialize u field   2.2134
initialize v field   0.
initialize w field   0.
initialize k field   .002939
initialize e field   .02618
```

Figure 34. nphase.dat file for TUTORIAL_5415 – single phase.

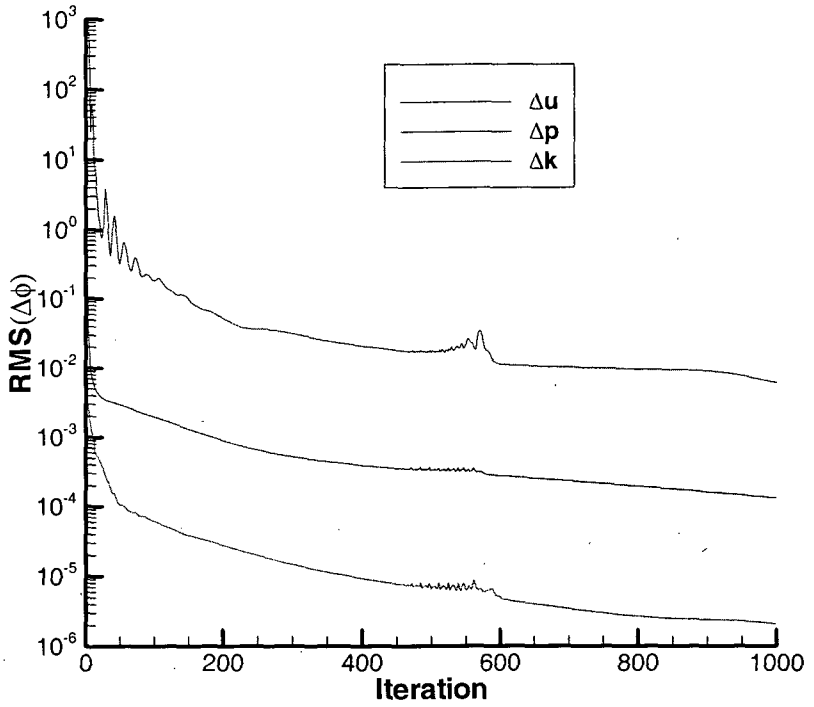

Figure 35. TECPLOT plot of residual history of single phase TUTORIAL_5145 simulation.

```
  iter fld   ru        rv        rw        rp        ra        rh        rk        re      ......
.
.
.
.

  1000  1 1.313e-04 2.219e-05 2.321e-05 6.033e-03 0.000e+00 0.000e+00 2.083e-06 5.176e-04 ......
Awetted dot i:  2.2368721880023e-05 m^2
Viscous  drag force:  3.6140082390256e+01
Pressure drag force:  5.8035216335649e+00
CD=:  7.6548160599064e+02
```

Figure 36. Snippets from standard output (n.out) for single phase TUTORIAL_5145 simulation, illustrating case specific drag coefficient output.

The second part of this tutorial involves running a notional microbubble drag reduction case, for comparison to the single phase drag just computed. To execute this part of the tutorial, the user should copy nphase.dat.2phase, shown in Figure 37, to nphase.dat. Numerous multifield extensions to the single phase input are apparent. Specifically:

1) a wall proximity computation (for all cells not just wall adjacent cells) is automatically carried out to support some of the interfacial force models (wall-lift). This wall proximity computation can be quite time consuming the first run of this case (several minutes). So the user should not be concerned. After a few minutes this process will be complete and the files wall_proximity.dat*xxx* (*xxx* = processor number, 000-031 here) will appear in the working directory and the iterations will begin. For all subsequent runs using this grid the user can forego the time consuming wall proximity calculation by reading these files on input, by commenting in the keyword: "read wall proximity from file".

2) Gravity of course needs to be specified. Here a conventional modified pressure approach is used with a reference density set equal to the gas density. This zeroes the magnitude of the buoyancy term in the gas momentum equation which has been found to improve convergence. A simple hydrostatic head initialization and exit pressure boundary specification is invoked.

3) Inlet, pressure and porous boundary conditions now have attributes specified for both fields. Note here that we specify liquid and gas volume fractions of $1x10^{-6}$ within 0 and 1, again arising for robustness (matrix singularity) reasons. A porous wall normal velocity of 0.3 m/s is set. This yields a nondimensional gas injection rate, $Q_a/(Q_a+Q_w) \cong 0.13$ which in turn corresponds to the HIPLATE 12m/s, 800 scfm case analyzed in TUTOTIAL_HIPLATE.

4) Turbulence scalars are solved only for the continuous field here.

5) Density and viscosity are set for both fields

6) Solver sweeps are set for both fields for all scalars (except pressure). Relaxation factors are set for both fields for all scalars (except pressure) and are significantly lower (0.2 vs. 0.6-0.7) for robustness reasons.

7) Initial values are set for both fields for all scalars (except pressure which is initialized using a hydrostatic assumption [see # 2 above]). All water is set initially ($\alpha^1=1$, $\alpha^2=0$.

8) The remainder of the keywords specify various interfacial dynamics, mass transfer and interfacial area density transport models.

The grid supplied in this tutorial is fairly coarse for a high Reynolds number boundary layer flow. Only five prism layers are employed and wall function resolution is specified. Accordingly the first order numerics associated with the volume fraction in this case gives rise to significant numerical smearing of the gas layer.

The model set chosen for this simulation includes all relevant interfacial dynamics (drag and non-drag) forces (see HIPLATE tutorial above and Kunz et al., (2006)). The model set is identical to that presented for the HIPLATE tutorial except that the bubble lift subid model was set to 6 not 5 since model 5 was "hardcoded' for a flat plate whereas lift model 6 is generalized for three dimensions.

In 2-phase mode, the code needs to always be run for quite a few more iterations at significantly reduced relaxation factors to obtain a reasonably converged drag prediction. Specifically, the two phase physics and model complexity of microbubble drag simulations give rise to non-linearities which conspire to reduce maximum allowable relaxation factor and introduce some modest quasi-unsteadiness into the flow field. Typically, for microbubble drag reduction applications one runs the code until it "flatlines". At that point one can look at the C_D history with iteration and its standard deviation. If the standard deviation is small compared to the magnitude of the predicted drag, then one accepts the mean C_D value as the prediction.

In this simulation, the code was run for 10000 iterations.

```
case title:
5415

iterations to perform 10000

employ model 5415 modeling

number of fields 2

#initialize run with restart file

produce ensight output
restart file write frequency 100

dont perform wall match logic
read wall proximity from file

gravity vector 0. 0. -9.81
employ modified pressure
reference density for modified pressure 1.
simple hydrostatic pressure treatment

overwrite inlet patch boundary conditions on restart
overwrite porous wall patch boundary conditions on restart
overwrite pressure patch boundary conditions on restart

inlet patch 2 0
2.2134 0. 0. 1000. .999999 .002939 .02618 0. 0.
2.2134 0. 0. 1.     .000001 .002939 .02618 0. 0.

pressure patch 2 0
0.0  1000.  .999999 .002939 .02618  0. 0.
0.0  1.     .000001 .002939 .02618  0. 0.

porous wall patch 2 0
0.3000 0. 0. 1000. .000001 .002939 .02618 0. 0.7  1.
0.3000 0. 0. 1.     .999999 .002939 .02618 0. 0.7  1.

turbulence model for each field 1 0

constant fluid molecular viscosity 1.e-3 1.5e-5
constant fluid density 1000. 1.
constant fluid surface tension .072 0.

spatial discretization momentum 2

function entry/exit echo off
adiabatic flow

solversweepsforu 3 3
solversweepsforv 3 3
solversweepsforw 3 3
solversweepsfora 3 3
solversweepsfork 3 3
solversweepsfore 3 3

solver choice for velocity components jacobiuvw
solver choice for pressure petsc
parallel strategy for pressure corrector: matrixlevel
petscprintnorm

employ rhie chow for dispersion terms
```

2) Gravity of course needs to be specified. Here a conventional modified pressure approach is used with a reference density set equal to the gas density. This zeroes the magnitude of the buoyancy term in the gas momentum equation which has been found to improve convergence. A simple hydrostatic head initialization and exit pressure boundary specification is invoked.

3) Inlet, pressure and porous boundary conditions now have attributes specified for both fields. Note here that we specify liquid and gas volume fractions of $1x10^{-6}$ within 0 and 1, again arising for robustness (matrix singularity) reasons. A porous wall normal velocity of 0.3 m/s is set. This yields a nondimensional gas injection rate, $Q_a/(Q_a+Q_w) \cong 0.13$ which in turn corresponds to the HIPLATE 12m/s, 800 scfm case analyzed in TUTOTIAL_HIPLATE.

4) Turbulence scalars are solved only for the continuous field here.

5) Density and viscosity are set for both fields

6) Solver sweeps are set for both fields for all scalars (except pressure). Relaxation factors are set for both fields for all scalars (except pressure) and are significantly lower (0.2 vs. 0.6-0.7) for robustness reasons.

7) Initial values are set for both fields for all scalars (except pressure which is initialized using a hydrostatic assumption [see # 2 above]). All water is set initially ($\alpha^1=1$, $\alpha^2=0$.

8) The remainder of the keywords specify various interfacial dynamics, mass transfer and interfacial area density transport models.

The grid supplied in this tutorial is fairly coarse for a high Reynolds number boundary layer flow. Only five prism layers are employed and wall function resolution is specified. Accordingly the first order numerics associated with the volume fraction in this case gives rise to significant numerical smearing of the gas layer.

The model set chosen for this simulation includes all relevant interfacial dynamics (drag and non-drag) forces (see HIPLATE tutorial above and Kunz et al., (2006)). The model set is identical to that presented for the HIPLATE tutorial except that the bubble lift subid model was set to 6 not 5 since model 5 was "hardcoded' for a flat plate whereas lift model 6 is generalized for three dimensions.

In 2-phase mode, the code needs to always be run for quite a few more iterations at significantly reduced relaxation factors to obtain a reasonably converged drag prediction. Specifically, the two phase physics and model complexity of microbubble drag simulations give rise to non-linearities which conspire to reduce maximum allowable relaxation factor and introduce some modest quasi-unsteadiness into the flow field. Typically, for microbubble drag reduction applications one runs the code until it "flatlines". At that point one can look at the C_D history with iteration and its standard deviation. If the standard deviation is small compared to the magnitude of the predicted drag, then one accepts the mean C_D value as the prediction.

In this simulation, the code was run for 10000 iterations.

```
case title:
5415

iterations to perform 10000

employ model 5415 modeling

number of fields 2

#initialize run with restart file

produce ensight output
restart file write frequency 100

dont perform wall match logic
read wall proximity from file

gravity vector 0. 0. -9.81
employ modified pressure
reference density for modified pressure 1.
simple hydrostatic pressure treatment

overwrite inlet patch boundary conditions on restart
overwrite porous wall patch boundary conditions on restart
overwrite pressure patch boundary conditions on restart

inlet patch 2 0
2.2134 0. 0. 1000. .999999 .002939 .02618 0. 0.
2.2134 0. 0. 1.     .000001 .002939 .02618 0. 0.

pressure patch 2 0
0.0  1000.  .999999 .002939 .02618  0. 0.
0.0  1.     .000001 .002939 .02618  0. 0.

porous wall patch 2 0
0.3000 0. 0. 1000. .000001 .002939 .02618 0. 0.7  1.
0.3000 0. 0. 1.     .999999 .002939 .02618 0. 0.7  1.

turbulence model for each field 1 0

constant fluid molecular viscosity 1.e-3 1.5e-5
constant fluid density 1000. 1.
constant fluid surface tension .072 0.

spatial discretization momentum 2

function entry/exit echo off
adiabatic flow

solversweepsforu 3 3
solversweepsforv 3 3
solversweepsforw 3 3
solversweepsfora 3 3
solversweepsfork 3 3
solversweepsfore 3 3

solver choice for velocity components jacobiuvw
solver choice for pressure petsc
parallel strategy for pressure corrector: matrixlevel
petscprintnorm

employ rhie chow for dispersion terms
```

```
case title:
5415

iterations to perform 10000

employ model 5415 modeling

number of fields 2

#initialize run with restart file

produce ensight output
restart file write frequency 100

dont perform wall match logic
read wall proximity from file

gravity vector 0. 0. -9.81
employ modified pressure
reference density for modified pressure 1.
simple hydrostatic pressure treatment

overwrite inlet patch boundary conditions on restart
overwrite porous wall patch boundary conditions on restart
overwrite pressure patch boundary conditions on restart

inlet patch 2 0
2.2134 0. 0. 1000. .999999 .002939 .02618 0. 0.
2.2134 0. 0. 1.     .000001 .002939 .02618 0. 0.

pressure patch 2 0
0.0  1000.  .999999 .002939 .02618  0. 0.
0.0  1.     .000001 .002939 .02618  0. 0.

porous wall patch 2 0
0.3000 0. 0. 1000. .000001 .002939 .02618 0. 0.7  1.
0.3000 0. 0. 1.    .999999 .002939 .02618 0. 0.7  1.

turbulence model for each field 1 0

constant fluid molecular viscosity 1.e-3 1.5e-5
constant fluid density 1000. 1.
constant fluid surface tension .072 0.

spatial discretization momentum 2

function entry/exit echo off
adiabatic flow

solversweepsforu 3 3
solversweepsforv 3 3
solversweepsforw 3 3
solversweepsfora 3 3
solversweepsfork 3 3
solversweepsfore 3 3

solver choice for velocity components jacobiuvw
solver choice for pressure petsc
parallel strategy for pressure corrector: matrixlevel
petscprintnorm

employ rhie chow for dispersion terms
```

```
relaxation factor for u .2 .2
relaxation factor for v .2 .2
relaxation factor for w .2 .2
relaxation factor for a .2 .2
relaxation factor for k .2 .2
relaxation factor for e .2 .2

initialize u field   2.2134 2.2134
initialize v field   0. 0.
initialize w field   0. 0.
initialize a field .999999 .000001
initialize k field   .002939 .002939
initialize e field   .02618 .02618
#
interfacial area transport model for each field 0 1
interfacial area coalescence model for each field 0 3
initialize interfacial area using volume fraction and characteristic length
interfacial area feeds back into bubble diameter frequency 100
constant field characteristic diameter 99. .000400
interfield drag models 1
1 2 6 0.8
bubble cluster drag modification
interfield nondrag models 5
1 2 2 5 10.
1 2 2 7 100. 2. -2.0 0.5 2.
1 2 3 3 .1
1 2 1 6 .075
1 2 5 3 1.
```

Figure 37. nphase.dat file for TUTORIAL_5415 – two phase.

```
  iter fld    ru          rv          rw          rp          ra          rh          rk          re      .....
.
.
.
  1000   1 1.313e-04 2.219e-05 2.321e-05 6.033e-03 0.000e+00 0.000e+00 2.083e-06 5.176e-04 .....
Awetted dot i:   2.2368721880023e-05 m^2
Viscous drag force:   3.6140082390256e+01
Pressure drag force:   5.8035216335649e+00
CD=:   7.6548160599064e+02
```

Figure 38. Snippets from standard output (n.out) for two phase TUTORIAL_5145 simulation, illustrating case specific drag coefficient output.

As of the writing of this version of the User's Manual (V1.5) a reasonably converged 5415 simulation has not been obtained with the model set that appears in Figure 37. The author is working to resolve this issue and this case will be updated accordingly in version 1.6 of the documentation.

Other Items of Interest

Running Two-Dimensional Problems

NPHASE-PSU can be run in two-dimensional mode quite easily by specifying a one-element thick grid and defining symmetry boundaries on the front and back faces. It does not matter how thick the element is. So, for example, a pure 2D triangular mesh will extrude to a one-element thick prism mesh.

Building User Specific/Case Specific Postprocessing

Often an NPHASE-PSU user is interested in case specific output, or even case specific coding that modifies some element of execution (say grid motion, boundary condition, hard coded initialization, etc...). For these situations, this section documents the procedures to incorporate such coding in a fashion that is accessed from the front end, i.e., does not affect the execution of the code for cases where user supplied keywords are not supplied. The process is illustrated with an example – defining a user specific output for the 5415 simulation carried out in the preceding tutorial.

The first step is to define a new keyword and attendant integer pointer flag. In this example we add the following lines to the main NPHASE-PSU data C-structure in NPHASE-PSU, *struct data*, which is defined in nphase_struct.h:

```
//  5415 specific stuff:
int *imodel5415_modeling ;
```

Next, storage is allocated for this pointer in constmemory.c:

```
// 5415 specific:
var.imodel5415_modeling= (int *) malloc(sizeof(int)) ;
```

The value of this integer is initialized to 0 (i.e., not set) in set_parameters.c:

```
int *imodel5415_modeling=var.imodel5415_modeling ;  //type definition

.

.

.

// 5415 specific
*imodel5415_modeling=0;
```

The value of this integer pointer is broadcast to all processors in broadcast.c:

```
int *imodel5415_modeling=var.imodel5415_modeling ;  //type definition

.

.

.

MPI_Bcast(imodel5415_modeling,1,MPI_INT,0,MPI_COMM_WORLD);
```

User access to setting this parameter is accomplished by defining a keyword in author_****.c, where *** is either "abcd", "efgh", "ijkl", "mnop", "qrstu", or "vwxyz" depending on the fist letter of the keyword. Here we define the keyword "employ model 5145 modeling" in author_efgh.c:

```
int *imodel5415_modeling=var.imodel5415_modeling ;  //type definition

.

.

.

/*-------------------------------------------------------*/
 else if(strncmp(keyword,"employmodel5415modeling"
                 ,strlen(keyword)) == 0) {
  *imodel5415_modeling = 1;
  *idid = 1;
  fprintf(fo,"%s",line) ;
 }
```

Note that keyword is defined with no spaces or special characters.

Now that all of the data structure and front end hooks are in place, the user can build or modify a function to execute what they wish. In this case, a new function is defined, model5415_output.c, the source of which appears in Figure 39. This function is called, in this case, at the end of every iteration, from nphase.c:

```
int *imodel5415_modeling=var.imodel5415_modeling ;  //type definition

.

.

.

if(*imodel5415_modeling==1)model5415_output();
```

Once these coding modifications have been made, the object model5415_output.o is add to the file Obj_nphase, and the entire code is recompiled using make –f makefile nphase *(options)*. The 5415 tutorial presented above includes the use of this coding as shown in the nphase.dat files included in Figure 34 and Figure 37. The output generated by this coding is shown in Figure 36 and Figure 38.

```
#include "nphase_struct.h"
#include "mpi.h"
#include <math.h>
#include <stdio.h>
extern struct boundary_patch wall;
extern struct boundary_patch porwall;
extern struct data var;
extern int myid;
        int  model5415_output()
{
/*-----------------------------------------------------
        called from:
        nphase
        V2.0 baseline code
        initials        comment
        rfk
-----------------------------------------------------*/
        int *nfield=var.nfield;
        int *nnode=var.nnode;
        int ibf,nstride,inode,fstride,ifield;
         real sumforce,sumforce0,sumflux,sumflux0;
         real awetted,awetted0;
         real rinf,uinf,cd;
         real *a=var.a;
         real *u=var.u;

        entered("model5415_output") ;

    uinf=2.2134;
    rinf=1000;

// wetted area

    awetted=0.;
    for(ibf=0;ibf<=wall.nbcface-1;++ibf){
    inode=wall.bcface_n[ibf];
    awetted+=wall.bamag[ibf];
    }
    for(ibf=0;ibf<=porwall.nbcface-1;++ibf){
    inode=porwall.bcface_n[ibf];
    awetted+=porwall.bamag[ibf];
    }
    MPI_Reduce(&awetted,&awetted0,1,mpireal,MPI_SUM,0,MPI_COMM_WORLD);
    if(myid==0)printf("Awetted: %20.13e m^2\n",awetted0);

// drag force on boat

    sumforce=0.;
    for(ifield=0;ifield<=*nfield-1;++ifield){
    fstride=ifield*wall.nbcface ;
    nstride=ifield* *nnode;
      for(ibf=0;ibf<=wall.nbcface-1;++ibf){
      inode=wall.bcface_n[ibf] ;
      sumforce+=wall.tmlt[ibf+fstride]*/*(a+inode+nstride) * wall.bamag[ibf]* *(u+inode+nstride);
      }
      for(ibf=0;ibf<=porwall.nbcface-1;++ibf){
      inode=porwall.bcface_n[ibf] ;
      sumforce+=porwall.tmlt[ibf+fstride]* *(a+inode+nstride) * porwall.bamag[ibf]* *(u+inode+nstride);
      }
    }
    MPI_Reduce(&sumforce,&sumforce0,1,mpireal,MPI_SUM,0,MPI_COMM_WORLD);
    if(myid==0)printf("Drag force: %20.13e\n",sumforce0);
    if(myid==0)cd=sumforce0/(.5*rinf*uinf*uinf*awetted0);
    if(myid==0)printf("CD=: %20.13e\n",cd);

        exiting("model5415_output") ;

        return 0 ;
}
```

Figure 39. model5415_output.c

Turbomachinery

Documentation not yet written.

Homogeneous Multiphase Flow

Documentation not yet written.

Control Commands

This section includes a description of most of the keyword commands available in NPHASE. These appear in nphase.dat in free format. They need not appear in any particular sequence. It is not exhaustive and many commands are not likely to be used by most users.

1.1. Comment Card

Any line in the "nphase.dat" input file that starts with a "#" is considered a comment card. The remainder of the line is ignored.

1.2. Job Control

1.2.1. Case Title

Specifies a title for the output of the job. Optional. A maximum of 132 characters can be used in the title. The title itself must appear on following line

```
case title:
$Title
```

1.2.2. Iterations to Perform

Specifies the number of iterations to perform in job. For a restart job, the specified number of iterations will be performed after the restart file is read. For a time accurate simulation, this is the number of inner iterations per physical timestep.

```
iterations to perform  $NITER
```

1.2.3. Over Write Boundary Conditions on Restart

During a restart from a previous run (See "initialize run with restart file" command) the boundary conditions are specified by the information in the restart file. If the user wants the boundary conditions specified by the "nphase.dat" input file, the over write boundary condition on restart command must be used. The command can be used on various boundary conditions as specified below.

```
over write inlet patch boundary conditions on restart
over write far field patch boundary conditions on restart
over write pressure patch boundary conditions on restart
over write wall patch boundary conditions on restart
over write porous patch boundary conditions on restart
```

1.2.4. Number of Physical Time Steps

The number of time steps for the case can be specified with this command.

```
number of physical time steps  $NTSTEPS
```

1.2.5. Physical Time Step in Seconds

This command is used to specify a constant time step size for a transient analysis.

```
physical time step in seconds  $DT
```

1.2.6. Time Accurate Simulation

Specifies that a transient analysis is to be run. The number of time steps and time step size can be specified with the "number of physical time steps" and "physical time step size in seconds" command

```
time accurate simulation
```

1.2.7. Transient File Write Frequency

Specifies the number of time steps between saving a time step numbered restart file. This command can be

used to save restart files for postprocessing animation.

restart file write frequency $NRST_FREQ

1.2.8. Volume Fraction Normalization

Specifies normalization of volume fraction equation. Normalization will ensure the sum of the volume fraction of all fields equals one.

carver volume fraction normalization

Specifies that volume fraction normalization is not used.

do not employ carver volume fraction normalization

1.2.9. Continuity Error Treatment for Momentum

Adds terms to the LHS and RHS of the discrete momentum equations that ensures that if the linear solver is brought in then no spurious sources of momentum will arise due to mass imbalances

continuity error treatment for momentum

1.2.10. Continuity Error Treatment for Enthalpy

Adds terms to the LHS and RHS of the discrete enthalpy equations that ensures that if the linear solver is brought in then no spurious sources of enthalpy will arise due to mass imbalances

continuity error treatment for enthalpy

1.2.11. Momentum Cross Diffusion

This command specifies that the nonorthogonal components of the viscous shear term are included in the momentum equation. For a general grid, this command should be included in the input deck. See equation **Error! Reference source not found.** in the theory manual.

cross diffusion included in momentum equations

1.2.12. Number of PISO Correction Steps

The number of PISO corrections steps is required when the PISO algroithm is employed (see piso employed command). The PISO algorithm is a more recent varient of the SIMPLE algroithm.

number of piso corrections $NPISO_STEPS

1.2.13. Parallel Strategy for Pressure Correction

Specifies whether the pressure correction equation is solved implicitly across all inter-processor boundaries.

parallel strategy for pressure correction: matrix level

parallel strategy for pressure correction: explicit partition boundary update

1.2.14. PISO Algorithm Employed

The PISO algorithm is a more recent variant of the SIMPLE algorithm. The PISO algorithm can be specified using this command. The number of PISO correction steps can be specified using the "number of piso corrections" command.

piso employed

1.2.15. Relaxation Factor

The relaxation factor command is used to improve numerical convergence by the addition of numerical damping to eliminate oscillations and improve stability in the solution. The form of the relaxation factor command is:

relaxation factor for u $RFU
relaxation factor for v $RFV
relaxation factor for w $RFW

relaxation factor for a $RFA
relaxation factor for k $RFK
relaxation factor for e $RFE
relaxation factor for h $RFH

Values for the false time step command are specified as:
$RFU - User input value for false time step in "u" momentum equation
$RFV - User input value for false time step in "v" momentum equation
$RFW - User input value for false time step in "w" momentum equation
$RFA - User input value for false time step in phasic volume fraction equation
$RFK - User input value for false time step in turbulent kinetic energy equation
$RFE - User input value for false time step in turbulent dissipation rate equation

1.2.16. SIMPLEC Factor

The use of this command can modify the numerical method to use the original SIMPLE algorithm ($SIMPLEC=0.) or a varient call the SIMPLEC algorithm ($SIMPLEC=1.). The SIMPLEC algorithm is the default value.

simplec factor $SIMPLEC

1.2.17. Solver Choice for All Scalars Jacobi

This command specifies the use of the Jacobi algorithm for solving the volume fraction, k-e turbulence and energy equations. This is the default solver for the volume fraction, k-e turbulence and energy equations.

solver choice for all scalars jacobi

1.2.18. Solver Choice for Enthalpy Jacobi

This command specifies the use of the Jacobi algorithm for solving the energy equation. This is the default solver for the energy equation.

solver choice for all scalars jacobi

1.2.19. Solver Choice for Pressure Jacobi

This command specifies the use of the Jacobi algorithm for solving the pressure correction equations in the SIMPLE algorithm. This is the default solver for the pressure correction equations

solver choice for pressure jacobi

1.2.20. Solver Choice for Turbulence Scalars Jacobi

This command specifies the use of the Jacobi algorithm for solving the k-e turbulence equations. This is the default solver for the k-e turbulence equations

solver choice for turbulence scalars jacobi

1.2.21. Solver Choice for Velocity Components Jacobi

This command specifies the use of the Jacobi algorithm for solving the momentum equations in the SIMPLE algorithm. This is the default solver for the momentum equations

solver choice for velocity components jacobi

1.2.22. Solver Sweeps

This command is used to specify the number of linear solver sweeps for each linear equation solver. Command are available to specify the number of solver sweeps separately for each equation or in combination..

solver sweeps for u $NSWEEP_field1, $NSWEEP_field1,...,$NSWEEP_fieldN
solver sweeps for v $NSWEEP_field1, $NSWEEP_field1,...,$NSWEEP_fieldN
solver sweeps for w $NSWEEP_field1, $NSWEEP_field1,...,$NSWEEP_fieldN
solver sweeps for p $NSWEEP_field1, $NSWEEP_field1,...,$NSWEEP_fieldN
solver sweeps for a $NSWEEP_field1, $NSWEEP_field1,...,$NSWEEP_fieldN

```
solver sweeps for h  $NSWEEP_field1, $NSWEEP_field1,...,$NSWEEP_fieldN
solver sweeps for tke  $NSWEEP_field1, $NSWEEP_field1,...,$NSWEEP_fieldN
solver sweeps for tds  $NSWEEP_field1, $NSWEEP_field1,...,$NSWEEP_fieldN
```

1.2.23. Spatial Discretization

This command is used to specify the spatial discretization used for the convective term in the governing equations. The default discretization is the hybrid scheme.

```
spatial discretization momentum $ISPATIAL
spatial discretization turbulence $ISPATIAL
spatial discretization volume fraction $ISPATIAL
spatial discretization turbulence $ISPATIAL
```

Values for the spatial discretization command are specified as:
$ISPATIAL = 0 (Use 1st order hybrid scheme)
1(Use 1st order upwind scheme)
2(Use 2nd order upwind scheme)

1.2.24. Temporal Discretization

This command is used to specify the spatial discretization used to transfrom the convective term in the governing equations. The default discretization is the hybrid scheme.

```
temporal discretization momentum $ITEMPOR
temporal discretization turbulence $ITEMPOR
temporal discretization volume fraction $ITEMPOR
temporal discretization turbulence $ITEMPOR
```

Values for the temporal discretization command are specified as:
$ITEMPOR = 1 (Use 1st order [Euler Implicit] scheme - default)
$ITEMPOR = 2 (Use 2nd order backward difference in time)

1.2.25. Thin Layer Approximation Employed/ Not Employed

Specifies whether or not a thin layer approximation is employed in the construction of viscous terms.

```
thin layer approximation employed
thin layer approximation not employed
```

1.3. Geometry Control

1.3.1. Cylindrical Coordinates

Specifies use of cylindrical coordinates (R-Z). Where axial flow (Z) is in the (X) coordinate direction, and radial (R) flow is in the (Y) coordinate direction. Requires (Y=0) to align with (R=0).

```
Cylindrical Coordinates
```

1.4. Boundary Conditions

1.4.1. Cyclic Boundary

All cyclic boundaries are specified in the "unphase.grid" file.

1.4.2. Far Field Boundary

1.4.3. Far Field Patch

All far field boundaries are specified in the "unphase.grid" file.

```
farfield patch $NCARD, $FACEID
$U, $V, $W, $RHO, $VF, $TKE, $TDS, $H,$P // for field one, repeat NCARD times for other fields
```

Specified far field boundary conditions for all farfield boundary faces with FACEID. Far field values are specified as:
$NCARD - Number of input lines imediately following. Usually equals NFIELD.
$FACEID - Identification of particular far field patch. (e.g., FACEID = 3 for Farfield_03)
$U - Cartesian "u" velocity at far field boundary (m/s)
$V - Cartesian "v" velocity at far field boundary (m/s)
$W - Cartesian "w" velocity at far field boundary (m/s)
$RHO - Fluid Density at inlet (kg/m^3), used for backflow conditions only
$VF - volume fraction at far field boundary
$TKE - Turbulent Kinetic Energy at far field boundary (m^2/s^2)
$TDS - Turbulence Dissipation at far field boundary
$H - Enthalpy at far field boundary (deg-K)
$P - Pressure at far field boundary (Pa)

1.4.4. Inlet Boundary

1.4.4.1.Inlet Patch

The Inlet Patch command is used to specify uniform inlet boundary conditions. All inlet boundaries are specified in the "unphase.grid" file.

Inlet Patch $NCARD, $FACEID
$U, $V, $W, $RHO, $VF, $TKE, $TDS, $T // for field one, repeat NCARD times for other fields

Specified inlet flow boundary conditions for all inlet boundary faces with FACEID. Inflow values are specified as:
$NCARD - Number of input lines imediately following. Usually equals NFIELD.
$FACEID - Identification of Faces to apply inlet patch. Face ID is specifed in "unphase.grid" file.
$U - Cartesian "u" velocity at inlet (m/s)
$V - Cartesian "v" velocity at inlet (m/s)
$W - Cartesian "w" velocity at inlet (m/s)
$RHO - Fluid Density at inlet (kg/m^3), used for backflow conditions only
$VF - volume fraction at inlet
$TKE - Turbulent Kinetic Energy at inlet (m^2/s^2)
$TDS - Turbulence Dissipation at inlet
$T - Temperature at inlet (deg-K)

1.4.4.2.Inlet Velocities Specified in Absolute Frame.

inlet velocities specified in absolute frame

1.4.4.3.Inlet Velocity Specified in Cylindrical Coordinates.

inlet velocities specified in cylindrical coordinates

1.4.4.4.Inlet Velocities Specified Transiently.

inlet velocities specified transiently

1.4.4.5.Inlet Profile Specified in inflow.pro

inlet profile specified in inflow.pro

1.4.4.6.Inlet Profiles 2D specified in inflow.pro

inlet profile 2d specified in inflow.pro

1.4.5. Porous Boundary

1.4.5.1.Porous Wall Patch

The Porous Wall Patch command is used to specify uniform boundary conditions through a porous surface. All porous boundaries are specified in the "unphase.grid" file.

porous wall patch $NCARD, $FACEID
$U, $V, $W, $RHO, $VF, $TKE, $TDS, $T, $PERMA // for field one, repeat NCARD times for other fields

Specified inlet flow boundary conditions for all inlet boundary faces with FACEID. Inflow values are specified as:
$NCARD - Number of input lines imediately following. Usually equals NFIELD.
$FACEID - Identification of Faces to apply inlet patch. Face ID is specifed in "unphase.grid" file.
$U - Cartesian "u" velocity through porous wall (m/s)
$V - Cartesian "v" velocity through porous wall(m/s)
$W - Cartesian "w" velocity through porous wall (m/s)
$RHO - Fluid Density through porous wall (kg/m^3)
$VF - volume fraction at through porous wall
$TKE - Turbulent Kinetic Energy through porous wall(m^2/s^2)
$TDS - Turbulence Dissipation through porous wall
$T - Temperature through porous wall (deg-K)
$PERMA - permeability of porous wall

1.4.6. Pressure Boundary

1.4.6.1.Pressure Patch

The Pressure Patch command is used to specify uniform pressure boundary conditions. All pressure boundaries are specified in the "unphase.grid" file.

Pressure Patch $NCARD, $FACEID
$P, $RHO, $VF, $TKE, $TDS, $H, $P0 // for field one, repeat NCARD times for other fields

Specified inlet flow boundary conditions for a patch of faces. Inflow values are specified as:
$NCARD - Number of input lines imediately following. Usually equals NFIELD.
$FACEID - Identification of Faces to apply inlet patch. Face ID is specifed in "unphase.grid" file.
$P - Pressure at pressure boundary (Pa)
$RHO - Fluid Density at pressure boundary (kg/m^3), used for backflow conditions only
$VF - volume fraction at pressure boundary, used for backflow conditions only
$TKE - Turbulent Kinetic Energy at pressure boundary (m^2/s^2), used for backflow conditions only
$TDS - Turbulence Dissipation at pressure boundary, used for backflow conditions only
$H - Enthalpy at pressure boundary (J/kgK), used for backflow conditions only
$P0 – Stagnation pressure at pressure boundary (Pa), currently unused

1.4.7. Symmetry Boundary

All symmetry boundaries are specified in the "unphase.grid" file.

1.4.8. Wall Boundary

1.4.8.1.Wall Patch

The Wall Patch command is used to modify wall boundary conditions. All wall boundaries are specified in the "unphase.grid" file.

Wall Patch $NCARD, $FACEID
$XTW, $XMW, $XFW, $XFW0, $XFW1, $XFW2 //heat transfer card $XTW_STRUCT, $XFW_TENSION //wall structural parameters card $XTW_VEL, $XFW_XVEL, $XFW_VVEL, $XFW_ZVEL //wall motion card //for field one, repeat these 3 lines NCARD times for other fields

NOTE: one does not need to specify any wall attributes if the wall is adiabatic, rigid and statinoary. If any of these attributes is non-default, all must be specified
Specified wall boundary conditions for all wall boundary faces with FACEID. Wall attributes are specified as:
$NCARD - Number of input lines imediately following. Usually equals NFIELD.

$FACEID - Identification of Faces to apply wall patch. Face ID is specifed in "unphase.grid" file.
$XTW – Type of wall boundary:
iwt=0: adiabatic
iwt=1: specify temperature
iwt=2: specify heat flux (xfw= 0. = adiabatic)
iwt=3: specify heat transfer coeficient (not implemented yet)
iwt=4: conjugate heat transfer across thin matching walls (continuous flux on either side of wall)
iwt=5: conjugate 1D heat transfer across thickness of assumed material adjacent to wall face with temp specified on outside face of that material
iwt=6: conjugate 1D heat transfer across thickness of assumed material adjacent to wall face with htc and tfilm specified on outside face of that material
$XMW – If this is a matching wall this integer is the the negative of the faceid of the matching wall. If this is not a matching wall it is ignored.
$XFW - Real # parameter:
if iwt=1, xfw=specified temperature in ^{o}K
if iwt=2, xfw=specified heat flux in W/m^2
if iwt=3, xfw=specified heat transfer coefficient (not implemented)
if iwt=4, xfw is unused since heat flux is continuous across matching internal walls
if iwt=5, xfw=specified temperature on outside of material assumed adjacent to wall patch
if iwt=6, xfw=specified film temperature (K) on outside of material assumed adjacent to wall patch
$XFW0 - unused for iwt=0->4
if iwt=5, xfw0=shell thermal conductivity (J/m*s*K) of material assumed adjacent to wall patch
if iwt=6, xfw0= heat transfer coefficient (W/m^2*K) on outside of material assumed adjacent to wall patch
$XFW1 - unused for iwt=0->4
if iwt=5, xfw1=shell thermal conductivity (J/m*s*K) of material assumed adjacent to wall patch
if iwt=6, xfw1= shell thickness (m) of material assumed adjacent to wall patch
$XFW2 - unused for iwt=0->5
if iwt=6, xfw2=shell thermal conductivity (J/m*s*K) of material assumed adjacent to wall patch

1.5. Initial Conditions

1.5.1. Initialize with Restart

The restart file "nphase_restart_in" will be used to specify the boundary and interior node values.

```
initialize run with restart file
```

1.5.2. Initial u Velocity

This command initializes the u velocity to a constant value specified by the user:.

```
initialize u velocity  $u_field0, $u_field1,...,$u_fieldN
```

1.5.3. Initial v Velocity

This command initializes the v velocity to a constant value specified by the user:.

```
initialize u velocity  $v_field0, $v_field1,...,$v_fieldN
```

1.5.4. Initial w Velocity

This command initializes the w velocity to a constant value specified by the user:.

```
initialize u velocity  $w_field0, $w_field1,...,$w_fieldN
```

1.5.5. Initial Pressure Field

This command initializes the pressure field to a constant value specified by the user:.

```
initialize p field  $w_field0, $w_field1,...,$w_fieldN
```

1.5.6. Initial Volume Fraction Field

This command initializes the volume fraction field to a constant value specified by the user:.

```
initialize volume fraction field  $VF_field0, $VF_field1,...,$VF_fieldN
```

1.5.7. Initial Interfacial Area Density

This command initializes the volume fraction field to a constant value specified by the user:.

```
initialize ai field  $AI_field0, $AI_field1,...,$AI_fieldN
```

1.5.8. Initial Turbulent Kinetic Energy (tke) Field

This command initializes the tke field to a constant value specified by the user:.

```
initialize tke field  $TKE_field0, $TKE_field1,...,$TKE_fieldN
```

1.5.9. Initial Turbulent Dissipation Rate (tds) Field

This command initializes the turbulent dissipation rate field to a constant value specified by the user:.

```
initialize tds field  $TDS_field0, $TDS_field1,...,$TDS_fieldN
```

1.5.10. Initial Enthalpy Field

This command initializes the enthalpy field to a constant value specified by the user:.

```
initialize h field  $H_field0, $H_field1,...,$H_fieldN
```

1.5.11. Hard Coded Initialization

```
hard coded initialization
```

1.6. Physical Models

1.6.1. Environmental Properties

1.6.1.1.Laminar Flow

This command specifies that the flow is to be treated as laminar flow..

```
laminar flow
```

1.6.1.2.Gravity Vector

This command specify the direction and magnitude of gravity.

```
gravity vector  $gx, $gy, $gz
```

Where,
$gx == "x" direction gravity magnitude,
$gy == "y" direction gravity magnitude,
$gz == "z" direction gravity magnitude

1.6.1.3. Employ Modified Pressure

This command specifies that a conventional gravity modified pressure treatment will be used.

```
employ modified pressure
```

1.6.1.4. Reference Density for Modified Pressure

This command specifies the reference density for the modified pressure treatment.

```
reference density for modified pressure  $rhoref
```

1.6.1.5. Employ Simple Hydrostatic Pressure Treatment

This command specifies that a simple single phase hydrostatic initialization of the pressure will be used based on local coordinate ($\nabla p=\rho g x$) and modified pressure reference density.

```
employ simple hydrostatic presure treatment
```

1.6.1.6. System Rotation About X Axis in Radians Per Second

Specifies the system rotation about the x-axis in radians per second.

```
system rotation about x axias in radians per second  $XRADS
```

1.6.2. Fluid and Solid Properties

1.6.2.1.Constant Fliud Molecular Viscosity

The constant fluid molecular viscosity is used to specify the molecular viscosity for each field.

constant fluid molecular viscosity $MU_field1, $MU_field1,...,$MU_fieldN

1.6.2.2.Constant Fluid Density

The constant fluid density command is used to specify the fluid density for each field..

constant fluid density $RHO_field1, $RHO_field1,...,$RHO_fieldN

1.6.2.3.Constant Fluid Surface Tension

The constant fluid surface tension command is used to specify the fluid surface tension for each field..

constant fluid surface tension $SIGMA_field1, $SIGMA_field1,...,$SIGMA_fieldN

1.6.2.4.Constant Fluid Specific Heat at Constant Pressure

The constant fluid specific heat at constant pressure command is used to specify the fluid specific heat (Cp) for each field..

constant fluid specific heat at constant pressure $CP_field1, $CP_field1,...,$CP_fieldN

1.6.2.5.Isothermal Compressibility Parameters

This command is used to specify compressibility parameters.

isothermal compressibility parameters $NCARD
$RHO_ref, $PRESS_ref, $1/C^2 // for field one, repeat NCARD times for other fields

1.6.2.6.Solid Region Density

The density of solid regions can be specified by this command.

solid region density $RHO_solids

1.6.2.7.Solid Region Thermal Conductivity

This command specifies the thermal conductive for solid regions.

solid region thermal conductivity $K_solids

1.6.2.8.Solid Region Specific Heat

This command specifies the specific heat for solid region.

solid region specific heat $CP_solids

1.6.3. Turbulence Models

1.6.3.1.Enforce Production Equals Dissipation

enforce production equals dissipation

1.6.3.2.Turbulent Flow High Reynolds Number k-ε Turbulence model

This command specified the use of the industry standard k-ε turbulence model.

turbulent flow high reynolds number k epsilon

1.6.3.3.Turbulent Flow Low Reynolds Number k-ε Turbulence model

This command specified the use of a low Reynolds number version of the k-ε turbulence model (Chien).

turbulent flow low reynolds number k epsilon

1.6.3.4.Turbulent Flow High Reynolds Number q-ω Turbulence model

This command specified the use of the q- ω turbulence model.

turbulent flow high reynolds number q omega

1.6.3.5.Turbulent Flow Low Reynolds Number q- ω Turbulence model

This command specified the use of a low Reynolds number version of the q-ω turbulence model (Coakley).

turbulent flow low reynolds number q omega

1.6.4. Multiphase Models

1.6.4.1. Constant Field Characteristic Diameter

The constant field characteristic diameter command is used to specify a single effective diameter for each field. This diameter is used in multiphase models such as drag, wall force and other models. It is ignored except for initialization when coalescence or breakup are implemented or if interfacial area transport is employed

constant field characteristic diameter $D_field1, $D_field1,...,$D_fieldN

NOTE: some fields (i.e., continuous liquid field) may not use the characteristic diameter, but an input is required.

1.6.4.2.Number of Fields

This command is used to specify the total number of fields. In a two-fluid ensembled averaged model, the fields can either represent different phases, of various forms of the same phase (i.e., small bubbles, large bubbles, taylor bubbles or continuous vapor).

number of fields $NFIELDS

1.6.4.3.Interfacial Drag Force

The interfacial drag force command is used to specify either a user defined or built-in model between any number of fields. At least one interfacial drag model is typically used for each field, however, none are requird and more than one is allowed. The form of the interfacial drag command is:

interfacial drag model $NCARD
$FIELDA,$FIELDB,$MODELID,$USERMULTIPLIER // for first drag model; repeat NCARD times for other drag models

The input for the drag force command is defined as:
$NCARDS - Number of input lines immediately following command line. Each line will specify a drag relation between fields.
$FIELDA - First Field for drag model
$FIELDB -Second field for drag model
$MODELID - Specifies Drag Model
0 - User Defined Drag Model (Use "user_drag.c" routine to define drag modeling)
1 - standard bubbly flow drag, C_D=0.44
2 – standard bubbly flow drag, C_D=0.44 with virtual mass force intergrated
3 – particle drag (solid spehere)
4 – seawater bubble drag
5 – fresh water bubble drag
6 – contaminated fresh water bubble drag
$USERMULTIPLIER - User specified multiplier (To modify drag coefficient, default=1.0)

1.6.4.4. Interfacial Non Drag Force

The interfacial nondrag force command is used to specify either a user defined or built-in model between any number of fields. The form of the interfacial nondrag command is:

interfacial nondrag model $NCARD
$FIELDA,$FIELDB,$MODELID,$MODELSUBID,$USERMULTIPLIER // for first nondrag model, repeat NCARD times for other nondrag models

The input for the nondrag force command is defined as:
$NCARDS - Number of input lines immediately following command line. Each line will specify a drag relation between fields.
$FIELDA - First Field for drag model
$FIELDB -Second field for drag model

$MODELID - Specifies Non Drag Force Model
0 - user specified model
1 - Lift Force Non-drag Model
2 - Volume Fraction Dispersion Non-drag Model
3 - Wall Force Non-drag Model
4 - Turbulence Non-drag Model
5 - Virtual Mass Force Non-drag Model
$MODELSUBID - Specifies subid for Non Drag Force Model
$USERMULTIPLIER - User specified multiplier (To modify non-drag coefficient, default=1.0)

1.6.4.5. Wall Shear Apportionment Model

The wall shear apportionment command is used to specify the fraction of wall shear associated with each field. Since, in many multiphase applications, the wall adjacent nodes may contain multiple fields, the fields in contact with the wall should be assigned the wall shear. The default for the code is the wall shear is apportioned by the local volume fraction in the wall adjacent nodes. In many applications (i.e., bubble flows, annular flows, ...) the user may assign all or a fraction of the wall shear to a particular field (or fields) with this command.

```
wall shear apportionment model $FRACT_field1, $FRACT_field2,...,$FRACT_fieldN
```

Where, $FRACT_field is the fraction of wall shear associated with each field. The sum of all $FRACT_field must add to one inorder to account for the wall shear.

1.6.4.6. Bubble Cluster Drag Modification

Specifies that the drag coefficient is modified according to equation **Error! Reference source not found.**.

```
Bubble cluster drag modification
```

1.6.4.7.Interfacial Area Transport Model for Each Field

Specifies the interfacial area transport model for each field.

```
interfacial area transport model for each field $iaint_field0, $iaint_field1, ..., $iaint_fieldN
```

1.6.4.8. Interfacial Area Coalescence Model for Each Field

Specifies the interfacial area coalescence model for each field.

```
interfacial area coalescence model for each field $coalescence_aint_model_field0,
$coalescence_aint_model_field1,..., $coalescence_aint_model_fieldN
```

1.6.4.9. Interfacial Area Breakup Model for Each Field

Specifies the interfacial area breakup model for each field.

```
interfacial area breakup model for each field $breakup_aint_model_field0,
$breakup_aint_model_field1,..., $breakup_aint_model_fieldN
```

1.6.4.10. Initialize Interfacial Area Using Volume Fraction and Characteristic Length

Specifies that the initial interfacial area density is defined using the local field volume fraction and characteristic length, not by "initialize ai field"

```
initialize interfacial area using volume fraction and characteristic length
```

1.6.4.11. Interfacial Area Feeds Back Into Bubble Diameter Frequency

Specifies the iteration frequency with which field bubble diameters are recomputed based on interfacial area density.

```
interfacial area feeds back into bubble diameter frequency $aint_db_feedback
```

1.6.4.12. Employ Rhie Chow for Dispersion Terms

Specifies that the Rhie-Chow artificial dissipation parameter **Error! Reference source not found.** includes the turbulence dispersion model.

```
employ rhie chow for dispersion terms
```

1.6.4.13.

Heat Transfer Models

1.6.4.14. Single Phase Heating

The use of this command will include the solution of the energy equation in the run. This command will allow only single phase heating (i.e., no mass transfer) in the analysis.

single phase heating

1.6.4.15. Adiabatic Flow

The adiabatic flow command is used to specify mass and momentum conservation are only needed for the NPHASE code.

adiabatic flow

1.6.4.16. Constant Fluid Reference Enthalpy

The constant fluid reference enthalpy command is used to specify a reference enthalpy for use in heat transfer models..

constant fluid reference enthalpy $HREF_field1, $HREF_field1,...,$HREF_fieldN

1.6.4.17. Constant Fluid Reference Temperature

The constant fluid reference temperature command is used to specify a reference temperature for use in heat transfer models.

constant fluid reference temperature $TREF_field1, $TREF_field1,...,$TREF_fieldN

1.6.5. Mass Transfer Models

1.6.5.1.Breakup sink for carrier field turbulence

breakup sink for carrier field turbulence

1.6.5.2.InterField Mass Transfer Models

This command can be used to specify a mass transfer model. The form of the interfacial drag command is:

interfield mass transfer models $NCARD
$FIELDA,$FIELDB,$CARRIER,$MODELID,$USERMULTIPLIER // for first model, repeat NCARD times for other models

The input for the drag force command is defined as:
$NCARDS - Number of input lines immediately following command line. Each line will specify a drag relation between fields.
$FIELDA - First Field for mass transfer model
$FIELDB - Second field for mass transfer model
$CARRIER - Carrier field for mass transfer
$MODELID - Specifies Drag Model
1 - Standard Mass Transfer Model
$USERMULTIPLIER - User specified multiplier (To modify mass transfer coefficient, default=1.0)

1.6.5.3.Mass Diffusion Coefficient Laminar

mass diffusion coefficient laminar $MCOEF_field1, $MCOEF_field1,...,$MCOEF_fieldN

1.6.5.4.Mass Diffusion coefficient Turbulent.

mass diffusion coefficient turbulent $MCOEF_field1, $MCOEF_field1,...,$MCOEF_fieldN

1.7. Additional Commands

allow deforming walls

anchor pressure

ausm artificial dissipation

boussinesq heating beta rho0t0 $BETA_BOUSS, $RHO_REF_BOUSS, $T_REF_BOUSS

build patran neutral file

clip species fraction stolie between zero and one inclusive

do not clip species fraction stolie between zero and one inclusive

clip volume fraction stolie between zero and one inclusive

do no tclip volume fraction stolie between zero and one inclusive

compute wall proximities

dont compute wall proximities

bubbly drag model $IP1, $IP2

constant fluid shear modulus $SHRMOD_field1, $SHRMOD_field1,...,$SHRMOD_fieldN

constant fluid reference density $RHOREF_field1, $RHOREF_field1,...,$RHOREF_fieldN

dimensionality offield view output $FV_OUT_DIM

continuous field0 turbulence drives all other fields

des modifications

dont perform wall match logic

drag and mass transfer in pseudo time step specification

drag and mass transfer not in pseudo time step specification

do not employ rhie chow for dispersion terms

continuity error treatment for turbulence

continuity error treatment for volume fraction

continuity error treatment for species

deforming region specification mid density modulus poisson $IMID, $L_REG_DEFORM, $R_REG_DEFORM, $E_REG_DEFORM, $N_REG_DEFORM

deforming wall specification face id mid modulus poisson thickness $IMID, $L_WALL_DEFORM, $R_WALL_DEFORM, $E_WALL_DEFORM, $N_WALL_DEFORM

deforming grid iteration frequency $ITER_FREQ

des model control fsst flag $FSST_FLAG

des model control c des $CDES

des model control t scale des $TSCALE_DES

des model control sigma x des $SIGMAX_DES

des model control ch1 des $CH1_DES

des model control ch2 des $CH2_DES

des model control ch3 des $CH3_DES

breakup model mu $BREAKUP_MULT

coalescence model mu $C_MULT

coalescence model surfactant $C_SURFACTANT

coalescence model initial film thi $TFILM_0

coalescence model final film thi $TFILM_FINAL

cfl number mear $CFL

cfl number tutrb $CFLT

enforce zero radial and tangential velocities for post processing

execute front and back ends only

employ froude damping

employ dirt lib over set technology

employ grid adaption

exclude continuous field from carver volume fraction normalization

exclude continuous field from carver

farfield grid smoothing

farfield patch species mass fractions $VAR

farfield patch species volume fractions $VAR

false time step for u $FTU_field1, $FTU_field1,...,$FTU_fieldN

false time step for v $FTV_field1, $FTV_field1,...,$FTV_fieldN

false time step for w $FTW_field1, $FTW_field1,...,$FTW_fieldN

false time step for a $FTA_field1, $FTA_field1,...,$FTA_fieldN

false time step for h $FTH_field1, $FTH_field1,...,$FTH_fieldN

false time step for k $FTK_field1, $FTK_field1,...,$FTK_fieldN

false time step for uu $FTUU_field1, $FTUU_field1,...,$FTUU_fieldN

false time step for vv $FTVV_field1, $FTVV_field1,...,$FTVV_fieldN

false time step for ww $FTWW_field1, $FTWW_field1,...,$FTWW_fieldN

false time step for uv $FTUV_field1, $FTUV_field1,...,$FTUV_fieldN
false time step for vw $FTVW_field1, $FTVW_field1,...,$FTVW_fieldN
false time step for wu $FTWU_field1, $FTWU_field1,...,$FTWU_fieldN
false time step for f $FTF_field1, $FTF_field1,...,$FTF_fieldN
false time step for e $FTE_field1, $FTE_field1,...,$FTE_fieldN
fluid rheology $FRHE_field1, $FRHE_field1,...,$FRHE_fieldN
employ mixture mass for pressure corrector
employ mixture volume for pressure corrector
employ cpe continuity coupling
employ symmetric form of modified pressure
employ thermodynamic data fits $THFIT_field1, $THFIT_field1,...,$THFIT_fieldN
function entry/exit echo on
function entry/exit echo off
employ leonard exact diffusion term
employ gibson modeling
employ crusader modeli $CRU_ELECTRHX_EFF, $CRU_TRANSOILHX_EFF, $CRU_ENGINEHX_EFF, $CRU_ELECTRHX_LOSS, CRU_TRANSOILHX,LOSS, $CRU_ENGINEHX_LOSS
employ model 5415 modeling
employ merkle deutsch flate plate modeling
employ hiplate modeling

```
employ darpa 12 inch modeling

employ large flat plate modeling

employ sub off modeling

employ asds modeling

employ uwx cavitator modeling

employ mruuv modeling

employ nrc case modeling

employ elgho bashi modeling

employ bubble rise modeling

employ meghan modeling  $IMEGHAN

employ bistline coding  $UCUR, $VCUT, $WCUR, $OXYCUR, $OYZCUR,
                        $OZXCUR,
                         $XCENTCUR, $YCENTCUR, $ZCENTCUR

employ haworth lungmodeling

employ visitor center modeling

employ cfd ship nphase bubble transport procedure

gibson modeling starting timestep

helicity filter  $HELICITY_FILTER

helicity smoothing  $HELICITY_SMOOTH

ensight gid not written

flow velocity initialization using stringer
```

hard coded pressure distributions

hard coded inlet

hard coded inlet swirl for cyclic verification

field1 constant molecular viscosity $VISMC

field1 constant density $RHOC

field1 constant surface tension $SIGMAC

free surface model $NFREESURF_surface1,
NFREESURF_surface2,...,NFREESURF_surfaceN

homogeneous gas mixture fields

homogeneous gas mixture field parameters $JANNAFOPT

homogeneous incompressible mixture fields

homogeneous incompressible mixture field parameters $VAR1, $VAR2

initialize u
field $u_field0, $u_field1,...,$u_fieldN

initialize v field $v_field0, $v_field1,...,$v_fieldN

initialize w field $w_field0, $w_field1,...,$w_fieldN

initialize a
field $a_field0, $a_field1,...,$wa_fieldN

initialize ai field $ai_field0, $ai_field1,...,$ai_fieldN

initialize species mass fractions $VAR1, $VAR2

initialize species volume fractions $VAR1

initialize
k field $K_field0, $K_field1,...,$K_fieldN

initialize e field $E_field0, $E_field1,...,$E_fieldN

initialize uu field $UU_field0, $UU_field1,...,$UU_fieldN

initialize vv field $VV_field0, $VV_field1,...,$VV_fieldN

initialize ww field $WW_field0, $WW_field1,...,$WW_fieldN

initialize uv field $UV_field0, $UV_field1,...,$UV_fieldN

initialize vw field $VW_field0, $VW_field1,...,$VW_fieldN

initialize wu field $WU_field0, $WU_field1,...,$WU_fieldN

initialize f field $F_field0, $F_field1,...,$F_fieldN

initialize pressure using s req

initialize with user initialize field routine $VAR1

inlet grid smoothing

inlet patch species mass fractions $MFRAC_N

inlet patch species volume fractions $VFRAC_N

interior wall patch $WPATCH_field0, $WPATCH_field1,...,$WPATCH_fieldN

intrinsic swirl filter $ISWRL_FILTER

intrinsic swirl smoothing $ISWRL_SMOOTH

ignore z direction indes delta computation

initial velocities specified in cylindrical coordinates

inlet profiles two dimensional specified in inflow.pro

inlet species profiles specified in inflow.pro

inlet species profiles 2d specified in inflow.pro

interfacial area does not feedback into bubble diameter

interfacial area coalescence model for each field $coales_aint_model_field0,
$coales_aint_model_field1,...,
$coales_aint_model_fieldN

interfield drag models $drag_fielda, $drag_fieldb, $drag_modelid, $drag_usermultiplier

interfield non drag models $nondrag_fielda, $nondrag_fieldb, $nondrag_modelid,
$nondrag_usermultiplier, $nondrag_coeff_id2,

interpolation scheme for face values $FACEINTERP

incorporate interior wall interface forces $INTERFACE_FORCES

interpret stringer coordinates as cylindrical

impose wall bubble diameter kinematic constraint

immersed boundaries

kumar bin partitioning $KUMAR_MOM1, $KUMAR_MOM2

kumar bin characteristic diameter $KUMAR_DIA

kumar bin characteristic volume $KUMAR_VOL

kumar breakup model $KUMAR_BREAKUP_MODELID, $KUMAR_BMULT

kumar coalescence model $KUMAR_COAL_MODELID, $KUMAR_CMULT

lake evaporation model $EVAP_MODELID, $EVAP_FACEID

limit dissipation in production term $EPS_PROD_MAX

limit dissipation in kumar breakup model $EPS_BREAKUP_MAX,
$LIMIT_EPS_BREAKUP

limit er form ean flow $LIMIT

limit er for turbulence $LIMIT2EQ

inviscid flux scheme $IFLX

pet sc specify solver iterations $IPETSC_ITER

pet sc specify solver tolerance $IPETSC_TOL

pet sc specify solver $IPETSC_SOLVR

pet sc specify preconditioner $IPETSC_PRECON

pet sc print norm $IPETSC_PRINTN

print t min and t max

modified production kato launder

moving grid

moving grid read frequency $GMR_FREQ

moving grid read grids

moving grid compute grids

moving grid compute grids prescribed motion

multiple frames of reference

pad vertex coordinates

neglect off diagonal terms in pressure poisson equation

parallel strategy for pressure corrector: matrix level

parallel strategy for pressure corrector: explicit partition boundary update

pressures at pressure boundaries specified transiently

perform agglomeration operation

print linear solver residual severys weep

print linear solver residuals after final sweep

overwrite density by mid $RHOC_MID

overwrite molecular viscosity by mid $VISMC_MID

overwrite gas molecular viscosities $VISGAS_field1, $VISGAS_field2,...,
$VISGAS_fieldN

produce viewable perprocessor ensight output files

produce ensight files with partition boundaries

produce ensight files without partition boundaries

produce ensight files with i blanked cells

produce ensight files without j blanked cells

produce ensight output

produce ensight scalar files for ai

produce ensight scalar files for a

produce ensight scalar files for temperature

produce ensight scalar files for h

produce ensigh tscalar files for k

produce ensight scalar files for e

produce ensight scalar files for v2f parameters

produce ensight scalar files for reynolds stresses

produce ensight scalar files for eddy viscosity

produce ensight scalar files for density

produce ensight scalar files for helicity

produce ensight scalar files for intrinsic swirl

produce ensight scalar files for debug var0

produce ensight scalar files for debug var1

produce ensight scalar files for debug var2

produce ensight scalar files for debug var3

produce ensight scalar files for debug var4

produce ensight scalar files for debug var5

produce ensight scalar files for debug var6

produce ensight scalar files for debug var7

produce ensight scalar files for debug var8

produce ensight scalar files for debug var9

produce ensight scalar files for species mass fractions

produce ensight scalar files for i blank

produce fieldview output

produce data explorer output

produce wall boundary layer output

pressure gradients req stream sheet correction

production destruction ratio clip $PRODCLIP

perfect gas compressibility parameters $RGAS_field1, $RGAS_field2,...$RGAS_fieldN
$SPRATIO_field1, SPRATIO_field2,..., SPRATIO_fieldN

calorically perfect gas compressibility parameters $RGAS_field1, $RGAS_field2,
...$RGAS_fieldN
$SPRATIO_field1, SPRATIO_field2,..., SPRATIO_fieldN

print pressure boundary reverse flow information

print outlet boundary reverse flow information

print cylindrical coordinate patchfiles

print fieldview in cylindrical coordinates

print pressure patch profile for inflow profile

pressure boundary use ambient velocities for inflow

pressure boundary use extrapolated velocities for inflow

prandtl number laminar $PRDTLL_field1, $PRDTLL_field2,..., $PRDTLL_fieldN

prandtl number turbulent $PRDTLT_field1, $PRDTLT_field2,..., $PRDTLT_fieldN

mass diffusion coefficient species laminar $LMASSDIFF_field1,
$LMASSDIFF_field2,..., $LMASSDIFF_fieldN

mass diffusion coefficient species turbulent $TMASSDIFF_field1,
$TMASSDIFF_field2,..., $TMASSDIFF_fieldN

porous wall patch species mass fractions

porous wall patch species volume fractions

pressure grid smoothing

pressure patch species mass fractions

pressure patch species volume fractions

pressure profile patch

pid attributes

number of additional species $NPHI

order of accuracy of inviscid terms mean flow

order of accuracy of inviscid terms turbulence

mass transfer model cprod rprod c destr dest $CPROD, $RPROD, $CDEST, $RDEST

mass transfer model flag $MTMODEL

mass transfer model linearization flag $ILMTRANS

natural cavitatinonumber $CAVNUM

pre conditioner flag $IPRECON

pre conditioning parameter beta $BETA

minimum number of sgs sweeps $ISGSMIN

maximum number of sgs sweeps $ISGSMAX

number of sgs sweeps $ISWEEP

solver choice for velocity components jacobi

solver choice for velocity components jacobi uvw

solver choice for velocity components petsc

solver choice for pressure amggs

solver choice for pressure amgilu

use directional coarsening

use direct solve on coarsest grid

solver choice for pressure block correction

solver choice for pressure petsc

solve mixture momentum mass centered mixture velocity

solve mixture momentum volume centered mixture velocity

read wall temperature data from file

read wall match data from file

read wall proximity from file

restart with field0 frozen

restart with pressure frozen

solver monitor iteration

solver choice for turbulence scalars petsc

solver choice for all scalars ilu

solver choice for enthalpy jacobi

solver choice for enthalpy petsc

relaxation factor for ai

relaxation factor for s

relaxation factor for uu

relaxation factor for vv

relaxation factor for ww

relaxation factor for uv

relaxation factor for vw

relaxation factor for wu

relaxation factor for f

relaxation factor for p

rhie chow multiplier

surface roughness height

residual print file not written

restart files not written

transient file write frequency

solver sweeps for k

solver sweeps for e

solver sweeps for species

rectilinear grid

second order viscous bcs for hexs and prisms

solve energy equation only

solve energy equation with frozen flow field

solve species equation only

solve species equation with frozen flow field

relaxation factor applied to linear solver only

thin layer approximation employed for momentum

thin layer approximation not employed for momentum

thin layer approximation employed for scalars

thin layer approximation not employed for scalars

spatial discretization interfacial area density

spatial discretization species equation

spatial discretization density

spatial discretization enthalpy

temporal discretization interfacial area density

temporal discretization species equation

temporal discretization enthalpy

temporal discretization grid

temporal discretization mass

temporal discretization

use absolute velocities as dependent variables

set pressure correction to zero on partition boundaries

set pressure correction gradient to zero on partition boundaries

use block correction to update pressure correction on partitions

turbulent flow menter k omega

turbulent flow menter k omega sst

turbulent flow goldber gkr

turbulent flow v2f

turbulent flow frsm

turbulent flow wilcox komega

turbulence model for each field

turbulence model reynolds number regime for each field

strongly coupled compressibility

update enthalpy during pressure corrector

update species during pressure corrector

use ficks law form for mass diffusion

use scalar relaxation for field coupling

use block relaxation for field coupling

use scalar relaxation for pc and rc coefficients

use block relaxation for pc and rc coefficients

symmetry grid smoothing

symmetry patch

smooth volume fraction in backend

solid region specification mid density conductivity cp

sgs parameters is chkgs tolsor fact

z test, no input yet

weakly coupled compressibility

wall grid smoothing

Software Delivery Summary – NRC delivery V3.1

The gzipped tar file with which the NPHASE-PSU software is delivered unpacks into the following UNIX directory structure:

tar –xzvf nphaseV3.1_NRC_delivery.tar.gz

NPHASE-PSU/

srcbase

METIS

PETSC-2.3.1

FUMP

EMERGE

SUGGAR_DIRT

TUTORIAL_1

TUTORIAL_HIPLATE

TUTORIAL_5415

The delivered NPHASE-PSU source code resides in the directory srcbase. The code unpacks with all objects (*.o) and executable (nphase) intact for execution on banyan. The code can be recompiled can be recompiled using the delivered makefile as follows:

make –f makefile.linux nphase

Executables for fump and emerge/emergetrans are delivered in their respective subdirectories. There is no need to recompile these so the source for these pre and postprocessors are not included. METIS is included since fump execution relies on run time library libmetis.a. The code compiles linking to the overset utilility library, dirtlib, which is delivered compiled for parallel execution and linking with NPHASE-PSU. PETSC version 2.3.1 is required for NPHASE-PSU compilation.

References

1. Batchelor, The Theory of Homogeneous Turbulence, Cambridge University Press, 1956.
2. Carrica, P. M., D. Drew, F. Bonetto, R. T. Lahey, Jr., "A Polydisperse Model for Bubbly Two-phase Flow around a Surface Ship," *International Journal of Multiphase Flow*, Vol. 25, 1999, pp. 257-305.
3. Clift, S. S., Forsyth, P. A., "Linear and Non-linear and Non-linear Methods for the Incompressible Navier-Stokes Equations," International Journal for Numerical Methods in Fluids, Vol. 18, 1994, pp. 229-256.
4. Coulaloglou, C. A., Tavlarides, L.L, "Description of Interaction Processes in Agitated Liquid-Liquid Dispersions," *Chemical Engineering Science*, Vol. 32, 1977, p. 1289.
5. Detch, ?, "??", *Journal of Geophyscial Research*, Vol. 96, No. 5, 1991, p. 8901.
6. Durbin, P.A., "Near-Wall Turbulence Closure Modeling Without Damping Functions," *Theoretical and Computational Fluid Dynamics*, Vol. 3, 1991, pp. 1-13.
7. ENSIGHT online documentation, http://www.ensight.com/ensight.html, 2006.
8. Ferrante, A., Elghobashi, S., "On the Physical Mechanisms of Drag Reduction in a Spatially Developing Turbulent Boundary Layer Laden with Microbubbles," *Journal of Fluid Mechanics*, Vol. 503, 2004, pp. 345-355.
9. Ferrante, A., Elghobashi, S., "Reynolds Number Effects on Drag Reduction in a Bubble-Laden Spatially Developing Turbulent Boundary Layer Over a Flat Plate," Proceedings of the 2nd International Symposium on Seawater Drag Reduction, Busan, Korea, May, 2005.
10. HARPOON online documentation, http://www.ensight.com/harpoon.html, 2006
11. Hibiki, T., T. Takamasa, M. Ishii: "Interfacial Area Transport of Bubbly Flow in a Small Diameter Pipe," *Journal of Nuclear Science and Technology*, Vol. 38, No. 8, 2001, pp. 614-620.
12. Hinze, J. O., Turbulence, 2nd edition, McGraw-Hill, 1975.
13. ICEM-CFD online documentation, http://www.ansys.com/products/icemcfd.asp, 2006.
14. Johansen, S.T., Boysan, F., "Fluid Dynamics in Bubble Stirred Ladles: Part II. Mathematical Modeling," *Metallurgical Transactions B*, Vol. 19b, 1988, pp. 755-764.
15. Kawamura, T., Yoshiba, H., "Numerical Modelng of Bubble Distributions in Horizontal Bubbly Channel Flow," Proceedings of the 5th International Conference on Multiphase Flow, Yokohama, Japan, 2004, Paper No. 354.

16. Kumar, S., Ramkrishna, D. "On the Solution of Population Balance Equations by Discretization – I. A Fixed Pivot Technique," Chemical Engineering Science, Vol. 51, No. 2, 1996, pp. 1311-1332.

17. Kunz, R.F., Siebert, B.W., Cope, W.K., Foster, N.F., Antal, S.P., Ettorre, S.M. "A Coupled Phasic Exchange Algorithm for Multi-Dimensional Four-Field Analysis of Heated Flows with Mass Transfer," Computers and Fluids, Vol. 27, No. 7, 1998.

18. Kunz, R.F., Venkateswaran, S., "On the Roles of Implicitness, Realizability, Boundary Conditions and Artificial Dissipation in Multidimensional Two-Fluid Simulations with Interfacial Forces," AMIF-ESF Workshop on Computing Methods for Two-Phase Flow, Centre Paul Langevin, Aussois, France, 12-14 January, 2000.

19. Kunz, R.F., Yu, W.S., Antal, S.P., Ettorre, S.M., "An Unstructured Two-fluid Method Based on the Coupled Phasic Exchange Algorithm," AIAA Paper 2001-2672, 2001.

20. Kunz, R.F., Deutsch, S., Lindau, J.W., "Two Fluid Modeling Of Microbubble Turbulent Drag Reduction," ASME Paper FED2003-45640, Proceedings of FEDSM'03: 4TH ASME–JSME Joint Fluids Engineering Conference, Honolulu, Hawaii, USA, July 6–11, 2003.

21. Kunz, R.F., Gibeling, H.J., Maxey, M.R., Tryggvason, G., Fontaine, A.A., Petrie, H.L., Ceccio, S.L., "Validation of Two-Fluid Eulerian CFD Modeling for Microbubble Drag Reduction Across a Wide Range of Reynolds Numbers," Proceedings of the 2nd International Symposium on Seawater Drag Reduction, Busan, Korea, May, 2005, to appear *ASME Journal of Fluids Engineering*, January, 2007.

22. Kunz, R.F., Fontaine, A.A., Maxey, M.A., "Multiscale Physical Modeling for Microbubble Drag Reduction at High Reynolds Numbers," FINAL REPORT to Defense Advanced Research Projects Agency Friction Drag Reduction Program, September 2006.

23. Lahey, R.T., Drew, D.A., "An Analysis of Two-Phase Flow and Heat Transfer Using a Multidimensional, Multi-field, Two-Fluid Computational Fluid Dynamics (CFD) Model," Japan/US Seminar on Two-Phase Flow Dynamics, Santa Barbara, California, 2000.

24. Lehr, F., Mewes, D., "A Transport Equation for the Interfacial Area Density Applied to Bubble Columns," *Chemical Engineering Science*, Vol. 56, 2001, pp. 1159-1166.

25. Levich, V. G.: Physicochemical Hydrodynamics, Prentice-Hall, Englewood Cliffs, NJ, 1962.

26. Lien, F.S. "A Pressure-Based Unstructured Grid Method for All-Speed Flows," *International Journal for Numerical Methods in Fluids*, Vol. 33, 2000.

27. Lopez de Bertodano, M.: "Two-fluid Model for Two-phase Turbulent Jets," *Nuclear Engineering and Design*, Vol. 179, 1998, pp. 65-74.

28. Loth, E., "Numerical Approaches for Motion of Dispersed Particles, Droplets and Bubbles," *Progress in Energy and Combustion Science*, Vol. 26, 2000, pp. 161-223.

29. Martinez-Bazan, C., Montanes, J.L., Lasheras, J.C., "On the Breakup of an Air Bubble Injected into a Fully Developed Turbulent Flow. Part 1. Breakup Frequency," *Journal of Fluid Mechanics*, Vol. 401, 1999a, pp. 157-182.

30. Martinez-Bazan, C., Montanes, J.L., Lasheras, J.C., "On the Breakup of an Air Bubble Injected into a Fully Developed Turbulent Flow. Part 2. Size PDF of the Resulting Daughter Bubbles," *Journal of Fluid Mechanics*, Vol. 401, 1999b, pp. 183-207.

31. Maxey, M.R., Dong, S., Xu, J., Karniadakis, G.E., "Simulations for Microbubble Drag Reduction (MBDR) at High Reynolds Numbers", Proceedings of the HPCMP Challenge Project, Users Group Conference 2005, Nashville TN, June 27-30, 2005. (Published by IEEE Computer Society).

32. Meng, J.C.S., Uhlman, J.S., "Microbubble Formation and Splitting in a Turbulent Boundary Layer for Turbulence Reduction," Proceedings of the International Symposium on Seawater Drag Reduction, Newport, RI, July, 1998, pp. 341-355.

33. METIS on-line documentation, http://glaros.dtc.umn.edu/gkhome/views/metis, 2006.

34. MPICH on-line documentation, http://www-unix.mcs.anl.gov/mpi/mpich , 2006.

35. PETSC documentation: http://www-unix.mcs.anl.gov/petsc

36. Prince, M.J., Blanch, H.W., "Bubble Coalescence and Breakup in Air-Sparged Bubble Columns," *AIChE Journal*, Vol. 36, No. 10, October, 1990, pp. 1485-1499.

37. Rhie, C. M., Chow, W. L. (1983) "Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation," *AIAA Journal*, Vol. 21, 1983, p. 1527.

38. Richardson, J. F., Zaki, W. N., "Sedimentation and Fluidization: Part I," Transactions of the Institute of Chemical Engineering, Vol. 32, 1954, p. 35.

39. Sanders, W.C., Dowling, D.R., Perlin, M., Ceccio, S.L.,"Bubble Friction Drag Reduction in a High Reynolds Number Flat Plate Turbulent Boundary Layer," *Journal of Fluid Mechanics*, Vol. 552, 2006, pp. 353-380

40. TECPLOT on-line documentation, http://www.tecplot.com , 2006.

41. Tryggvason, G., Lu, J. "DNS of Drag Reduction due to Bubble Injection into Turbulent Flow," Proceedings of the 2nd International Symposium on Seawater Drag Reduction, Busan, Korea, May, 2005.

42. Van Doormal, J. P., Raithby, G. D., "Enhancements of the SIMPLE Method for Predicting Incompressible Fluid Flows," *Numerical Heat Transfer*, Vol. 7, 1984, pp. 147-163.

43. Venkateswaran, S., Tamamidis, P., Merkle, C.L., "Interpretation of Pressure-Based Methods as Time Marching Schemes," Proceedings 15th International Conference on Numerical Methods in Fluid Dynamics, Springer Lecture Notes in Physics, 1997.

44. Williams, M.M.R., Loyalka, S.K., Aerosol Science Theory and Practice, Pergamon Press, Oxford,1991.