References

Benyahia, S. Arastoopour, H., Knowlton, T.M., and Massah, H. (2000). Simulation of particles and gas flow behavior in the riser section of a circulating fluidized bed using the kinetic theory approach for the particulate phase. Powder Technology, 112, 24-33.

Bird, R.B., Stewart, W.E., and Lightfoot, E.N. (1960). Transport Phenomena, John Wiley and Sons.

Booncharoen, K. (2001). Bubbles in A Two-dimensional Fluidized Bed, Master Thesis, The petroleum and Petrochemical College, Chulalongkorn University.

Chapman, S., and Cowling, T.G. (1970). The Mathematical Theory of Non-uniform Gases, 3rd Edition, Cambridge.

Darton, R.C., LaNauze, R.D., Davidson, J.F., and Harrison, D. (1977). Bubble Growth Due to Coalescence in Fluidised Beds. Trans. Inst. Chem. Eng., 55, 274-280.

Gidaspow, D. (1994). Multiphase Flow and Fluidization. Continuum and Kinetic Theory Descriptions, Boston, Academic Press, Inc.

Gidaspow, D., and Huilin, L. (1996). Collisional Viscosity of FCC Particles in a CFB. AIChE Journal, 42(9), 2503-2510.

Gidaspow, D., and Huilin, L. (1997). Liquid Solid Fluidization Using Kinetic Theory. AIChE SYMPOSIUM SERIES, 93(317), 12-17.

Gidaspow, D., and Huilin, L. (1998). Equation of State and Radial Distribution Functions of FCC Particles in a CFB. AIChE Journal, 44(2), 279-293.

Gidaspow, D. (2002). Hydrodynamics of Fluidization Using Kinetic Theory An Emerging Paradigm, 2002 Flour-Daniel Lecture. AIChE Annual Indianapolis Meeting, 138a.

Grace, J.R. (1982) Fluidized Bed Hydrodynamics. Handbook of Multiphase Systems, Washington, D.C., McGraw-Hill Hemisphere, 8.5-8.83.

Hatziavramidis, D., Sun, B., and Gidaspow, D. (1997). Gas-Liquid Flow through Horizontal Tees of Branching and Impacting Type. AIChE Journal, 43(7), 1675-1683.

Huilin, L., and Gidaspow, D. (1995). Dimension Measurements of Hydrodynamic Attractors in Circulating Fluidized Beds. AIChE SYMPOSIUM SERIES, 91(308), 103-111.

Huilin, L., and Gidaspow, D., Manger, E. (2001). Kinetic theory of fluidized binary granular mixtures. PHYSICAL REVIEW E, 64, 061301-1 – 061301-8.

Jung, J., Ph.D. Thesis, Illinois Institute of Technology, 2003 Progress

Manger, E. (1996). Modelling and Simulation of Gas/Solids Flow in Curvilinear Coordinates, Telemark College Department of Technology, Norway.

Matonis, D. (2000). Hydrodynamic Simulation of a Gas-liquid–solid Fluidization, Ph.D. Thesis, Illinois Institute of Technology, Chicago, Illinois.

Matonis, D., Gidaspow, D. and Bahary, M. (2002). CFD Simulation of Flow and Turbulence in a Slurry Bubble Column. AIChE Journal, 48, 1413-1429.

McCabe, W.L., Smith, J.C., and Herriott, P. (1985). Unit Operations of Chemical Engineering, 4th Edition, McGRAW-HILL, 195-199.

Mostofi, R. (2002). CFD Simulations of Particulate Two-phase and Three-phase Flows, Ph.D. Thesis, Illinois Institute of Technology, Chicago, Illinois.

Neri, A. (1998). Multiphase Flow Modeling and Simulation of Explosive Volcanic eruptions, Ph.D. Thesis, Illinois Institute of Technology, Chicago, Illinois.

Neri, A., and Gidaspow, D. (2000). Riser Hydrodynamics: Simulation Using Kinetic Theory. AIChE Journal, 46(1), 52-67.

Pape, R., and Gidaspow, D. (1998). Numerical Simulation of Intense Reaction Propagation in Multiphase Systems. AIChE Journal, 44(2), 294-309.

Sun, B. (1996). Simulation of Gas-liquid and Gas-solid Two Phase Flows, Ph.D. Thesis, Illinois Institute of Technology, Chicago, Illinois.

Sun, B., and Gidaspow, D. (1999). Computation of Circulating Fluidzed-Bed Riser Flow for the Fluidization VIII Benchmark Test. Industrial & Engineering Chemistry Research, 38, 787-792.

Tartan, M., and Gidaspow, D. (2002). Measurement of Radial Granular Temperature Distribution in Risers: Comparison to an Analytical Solution. AIChE Annual Indianapolis meeting, 140e.

Wang, W., and Li, Y. (2001). Hydrodynamic Simulation of Fluidization by Using Modified Kinetic Theory. Industrial & Engineering Chemistry Research, 40, 5066-5073.

Wu, Y. (1996). Simulation of Methanol Synthesis in Slurry Bubble Column Reactors, Ph.D. Thesis, Illinois Institute of Technology, Chicago, Illinois.

Wu, Y., Gidaspow, D. (2000). Hydrodynamic simulation of methanol synthesis in gas-liquid slurry bubble column reactors. Chemical Engineering Science, 55, 573-587.

APPENDICES

APPENDIX A Finite difference equations.

The computations are carried out with a two-dimensional Eulerian mesh of non-uniform size finite difference computational cells by using ICE method. The cells (x,y) are rectangles with dimensions δx_i and δy_j. A typical computational cell (i,j) is shown in Figure A1.

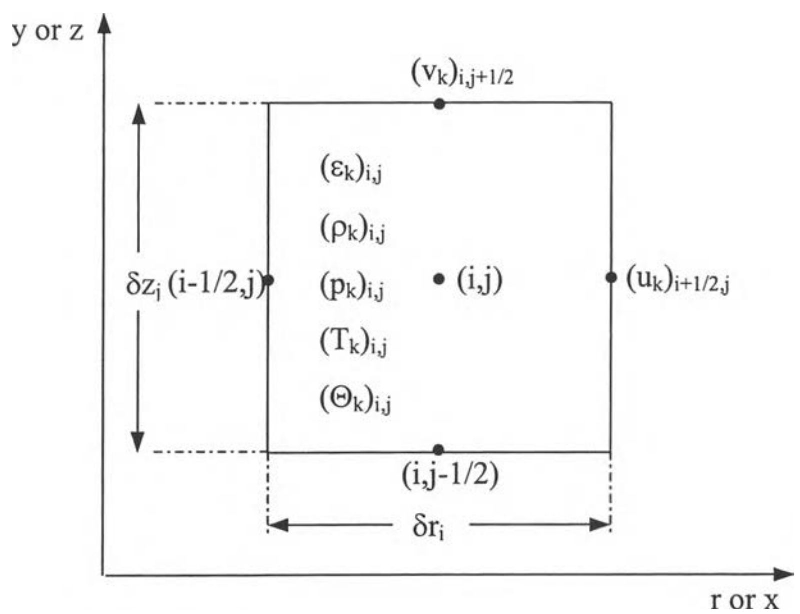

Figure A1 Computational mesh cell centered in a fluid cell.

The label cell (i,j) counts cell centers in the x direction and the y direction, respectively, and assumes only positive integer values. The half-integer indexes denote cell edge positions. The scalar variables (ε_k, r_k, P_k, T_k) are located at the cell center and the vector variables (v_k, $[\tau_k]$) at the cell boundaries.

The finite difference is approximated to the hydrodynamic equations from a system of nonlinear algebraic equations relating quantities at time $t = (n+1)\,\delta t$, where n is zero or a positive integer and δt is the time increment by which these quantities advance each computational cycle.

Averaging Process

Quantities in the finite difference equations required at spatial locations other than where they are defined are obtained by weighted averaging.

1) Cell Centered Quantities

The cell centered properties Ψ are defined at the cell center at (i,j). At other locations averaging is used as follows,

$$\Psi_{i+1/2,j} = \frac{1}{2\delta r_{i+1/2}}\left(\delta r_{i+1}\Psi_{i,j} + \delta r_i \Psi_{i+1,j}\right) \quad \text{(A1)}$$

$$\Psi_{i+1/2,j+1/2} = \frac{1}{2\delta z_{j+1/2}}\left(\delta z_{j+1}\Psi_{i,j} + \delta rz\ {}_j\Psi_{i,j+1}\right) \quad \text{(A2)}$$

$$\Psi_{i,j+1/2} = \frac{1}{2\delta z_{j+1/2}}\left(\delta r_{i+1}\delta z_{j+1}\Psi_{i,j} + \delta r_i \delta z_{j+1}\Psi_{i+1,j} + \delta r_{i+1}\delta z_j \Psi_{i,j+1} + \delta r_i z_j \Psi_{i+1,j+1}\right) \quad \text{(A3)}$$

Boundary Centered Quantities

The boundary centered quantity in r(x) direction is u which is defined at i+1/2,j. The average is as follows,

$$u_{i,j} = \frac{1}{2}\left(u_{i-1/2,j} + u_{i+1/2,j}\right) \quad \text{(A4)}$$

$$u_{i+1/2,j+1/2} = \frac{1}{2\delta z_{j+1/2}}\left[\delta z_{j+1}u_{i+1/2,j} + \delta z_j u_{i+1/2,j+1}\right] \quad \text{(A5)}$$

$$u_{i,j+1/2} = \frac{1}{4\delta z_{j+1/2}}\left[\delta z_{j+1}\left(u_{i-1/2,j} + u_{i+1/2,j}\right) + \delta z_j\left(u_{i-1/2,j+1} + u_{i+1/2,j+1}\right)\right] \quad \text{(A6)}$$

The boundary centered quantity in z direction is v which is defined at i+1/2,j. The average is as follow,

$$v_{i,j} = \frac{1}{2}\left(v_{i,j-1/2} + v_{i,j+1/2}\right) \tag{A7}$$

$$v_{i+1/2,j+1/2} = \frac{1}{2\delta r_{i+1/2}}\left[\delta r_{i+1} v_{i,j+1/2} + \delta r_i v_{i+1,j+1/2}\right] \tag{A8}$$

$$v_{i+1/2,j} = \frac{1}{4\delta r_{i+1/2}}\left[\delta r_{i+1}\left(v_{i,j-1/2} + v_{i,j+1/2}\right) + \delta r_i\left(v_{i+1,j-1/2} + v_{i+1,j+1/2}\right)\right] \tag{A9}$$

Continuity Equations

The continuity equations are difference fully implicitly as follows,

$$\left(\varepsilon_k \rho_k\right)_{i,j}^{n+1} = \left(\varepsilon_k \rho_k\right)_{i,j}^{n} - \frac{\delta t}{\delta r_i}\left(\varepsilon_k \rho_k u_k\right)_{i,j}^{n+1} - \frac{\delta t}{\delta z_j}\left(\varepsilon_k \rho_k v_k\right)_{i,j}^{n+1} \tag{A10}$$

Momentum Equations

The momentum equations are difference over a staggered mesh (Figure 3) using a scheme in which the convective terms are treated explicitly and all other terms are treated implicitly. The difference equations are,

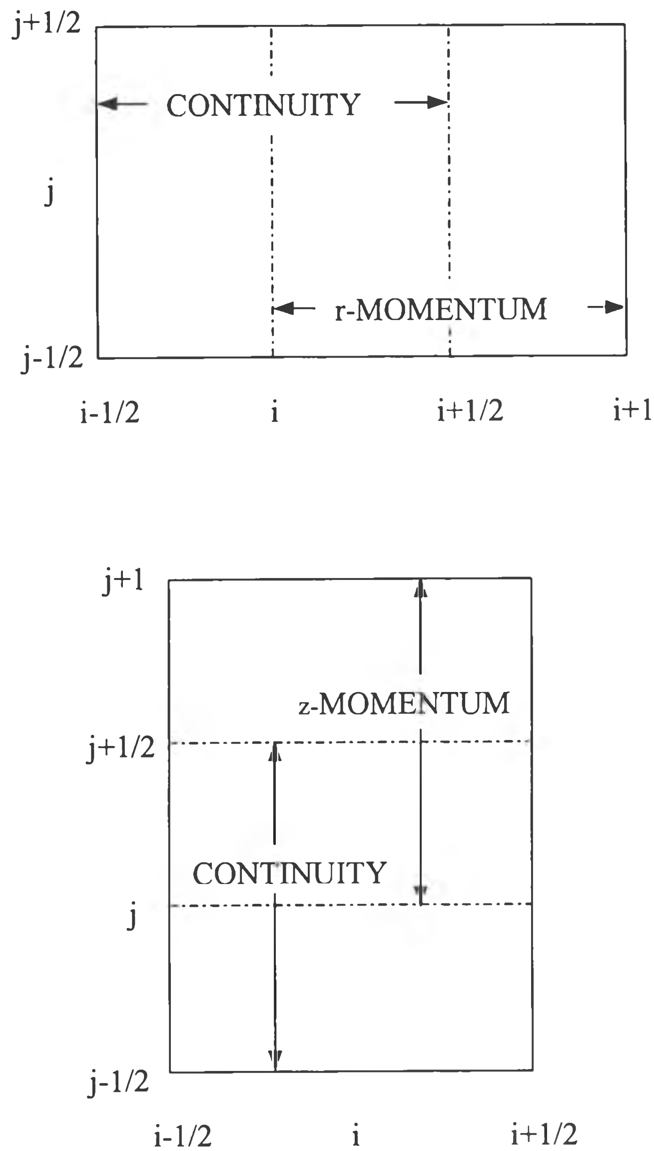

Figure A2 The staggered finite difference computational mesh for momentum equations.

$$(\varepsilon_k\rho_k u_k)^{n+1}_{i+1/2,j} = \left(\overline{\varepsilon_k\rho_k u_k}\right)_{i+1/2,j} - \frac{\delta t}{\delta r_{i+1/2}}\left((p_k)^{n+1}_{i+1,j} - (p_k)^{n+1}_{i,j}\right) - (w_k)^{n+1}_{i+1/2,j} g_r \delta t$$

$$+ \delta t \sum_{\substack{l=f,1\\ l\neq k}}^{N} (\beta_{lk})^{n+1}_{i+1/2,j}\left((u_l)^{n+1}_{i+1/2,j} - (u_k)^{n+1}_{i+1/2,j}\right) + \frac{\delta t}{\delta r_{i+1/2}}\left((\tau_{ck})^{n+1}_{i+1,j} - (\tau_{ck})^{n+1}_{i,j}\right) \qquad \text{(A11)}$$

$$\left(\varepsilon_k \rho_k v_k\right)_{i,j+1/2}^{n+1} = \left(\widetilde{\varepsilon_k \rho_k v_k}\right)_{i,j+1/2} - \frac{\delta t}{\delta z_{j+1/2}}\left(\left(p_k\right)_{i,j+1}^{n+1} - \left(p_k\right)_{i,j}^{n+1}\right) - \left(w_k\right)_{i,j+1/2}^{n+1} g_z \delta t$$

$$+ \delta t \sum_{\substack{l=f,1 \\ l \neq k}}^{N} \left(\beta_{lk}\right)_{i,j+1/2}^{n+1}\left(\left(v_l\right)_{i,j+1/2}^{n+1} - \left(v_k\right)_{i,j+1/2}^{n+1}\right) + \frac{\delta t}{\delta z_{j+1/2}}\left(\left(\tau_{ck}\right)_{i,j+1}^{n+1} - \left(\tau_{ck}\right)_{i,j}^{n+1}\right) \quad \text{(A12)}$$

Where for fluid phase $w_f = \rho_f$ and $\tau_{cf} = 0$, and for particulate phases (k=1,2,3,...,N),

$$w_f = \frac{\varepsilon_k}{\varepsilon_f}\left(\rho_k - \sum_{i=f,1}^{N} \varepsilon_l \rho_l\right) \quad \text{(A13)}$$

All the explicit terms are lumped into the "tilde" quantities as shown below,

$$\left(\widetilde{\varepsilon_k \rho_k u_k}\right)_{i+1/2,j} = \left(\varepsilon_k \rho_k u_k\right)_{i+1/2,j}^{n} - \frac{\delta t}{\delta r_{i+1/2}}\left\langle\left(\varepsilon_k \rho_k u_k\right) u_k\right\rangle_{i+1/2,j}^{n} - \frac{\delta t}{\delta z_j}\left\langle\left(\varepsilon_k \rho_k u_k\right) v_k\right\rangle_{i+1/2,j}^{n}$$

$$+ \frac{\delta t}{\delta r_{i+1/2}}\left[\left(\tau_{krr}\right)_{i+1,j}^{n} - \left(\tau_{krr}\right)_{i,j}^{n}\right] + \frac{\delta t}{\delta z_j}\left[\left(\tau_{krz}\right)_{i+1/2,j+1/2}^{n} - \left(\tau_{krz}\right)_{i+1/2,j-1/2}^{n}\right] \quad \text{(A14)}$$

$$\left(\widetilde{\varepsilon_k \rho_k v_k}\right)_{i,j+1/2} = \left(\varepsilon_k \rho_k v_k\right)_{i,j+1/2}^{n} - \frac{\delta t}{\delta r_i}\left\langle\left(\varepsilon_k \rho_k u_k\right) u_k\right\rangle_{i,j+1/2}^{n} - \frac{\delta t}{\delta z_{j+1/2}}\left\langle\left(\varepsilon_k \rho_k v_k\right) v_k\right\rangle_{i,j+1/2}^{n}$$

$$+ \frac{\delta t}{\delta z_{j+1/2}}\left[\left(\tau_{kzz}\right)_{i,j+1}^{n} - \left(\tau_{kzz}\right)_{i,j}^{n}\right] + \frac{\delta t}{\delta r_j}\left[\left(\tau_{krz}\right)_{i+1/2,j+1/2}^{n} - \left(\tau_{krz}\right)_{i-1/2,j+1/2}^{n}\right] \quad \text{(A15)}$$

The flux quantities denoted by $\langle \Psi u_k \rangle$ and $\langle \Psi v_k \rangle$ are calculated using donor-cell difference, where Ψ refers to ($\varepsilon_k\rho_k$), ($\varepsilon_k\rho_k u_k$), or ($\varepsilon_k\rho_k v_k$) quantities. The angular brackets represent donor cell different quantities as shown below,

$$\langle \Psi u_k \rangle_{m,p} = \begin{cases} (u_k)_{m+1/2,p} * \begin{cases} (\Psi)_{m,p} & \text{if } (u_k)_{m+1/2,p} \geq 0. \\ (\Psi)_{m+1,p} & \text{if } (u_k)_{m+1/2,p} < 0. \end{cases} \\ -(u_k)_{m-1/2,p} * \begin{cases} (\Psi)_{m-1,p} & \text{if } (u_k)_{m-1/2,p} \geq 0. \\ (\Psi)_{m,p} & \text{if } (u_k)_{m-1/2,p} < 0. \end{cases} \end{cases} \tag{A16}$$

$$\langle \Psi u_k \rangle_{m,p} = \begin{cases} (u_k)_{m+1/2,p} * \begin{cases} (\Psi)_{m,p} & \text{if } (u_k)_{m+1/2,p} \geq 0. \\ (\Psi)_{m+1,p} & \text{if } (u_k)_{m+1/2,p} < 0. \end{cases} \\ -(u_k)_{m-1/2,p} * \begin{cases} (\Psi)_{m-1,p} & \text{if } (u_k)_{m-1/2,p} \geq 0. \\ (\Psi)_{m,p} & \text{if } (u_k)_{m-1/2,p} < 0. \end{cases} \end{cases} \tag{A17}$$

The viscous stress components are calculated with standard centered difference, i.e.,

$$(\nabla \cdot v_k)_{i,j} = \frac{(u_k)_{i+1/2,j} - (u_k)_{i-1/2,j}}{\delta r_i} + \frac{(v_k)_{i,j+1/2} - (u_k)_{i,j-1/2}}{\delta z_j} \tag{A18}$$

For the fluid phase, μ_k is replaced by $\varepsilon_f \mu_f$.

$$(\tau_{krr})_{i,j} = 2(\mu_k)_{i,j}\left(\frac{(u_k)_{i+1/2,j} - (u_k)_{i-1/2,j}}{\delta r_i}\right) + \frac{2}{3}(\mu_k)_{i,j}(\nabla \cdot v_k)_{i,j} \tag{A19}$$

$$\left(\tau_{kzz}\right)_{i,j} = 2\left(\mu_k\right)_{i,j}\left(\frac{\left(v_k\right)_{i,j+1/2} - \left(v_k\right)_{i,j-1/2}}{\delta z_j}\right) + \frac{2}{3}\left(\mu_k\right)_{i,j}\left(\nabla \cdot v_k\right)_{i,j} \tag{A20}$$

$$\left(\tau_{krz}\right)_{i,j} = 2\left(\mu_k\right)_{i,j}\left(\frac{\left(u_k\right)_{i,j+1} - \left(u_k\right)_{i,j-1}}{\delta z_{j-1/2} + \delta z_{j+1/2}} + \frac{\left(v_k\right)_{i+1,j} - \left(v_k\right)_{i-1,j}}{\delta r_{i-1/2} + \delta r_{i+1/2}}\right) \tag{A21}$$

$$\left(\tau_{krz}\right)_{i+1/2,j+1/2} = 2\left(\mu_k\right)_{i+1/2,j+1/2}\left(\frac{\left(u_k\right)_{i+1/2,j+1} - \left(u_k\right)_{i+1/2,j}}{\delta z_{j+1/2}} + \frac{\left(v_k\right)_{i+1,j+1/2} - \left(v_k\right)_{i,j+1/2}}{\delta r_{i+1/2}}\right) \tag{A22}$$

Solution of the Momentum Equations

To facilitate the particular method of solution the equations are recast in the following form. The momentum equations in r- direction could be collected together in a matrix form.

$$(A)_{i+\frac{1}{2},j}(U)^n_{i+\frac{1}{2},j} = \left(B_U\right)_{i+\frac{1}{2},j} \tag{A23}$$

$$A = \begin{pmatrix} A_{ff} & A_{f1} & A_{f2} & \cdots & A_{fN} \\ A_{1f} & A_{11} & A_{12} & \cdots & A_{1N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{Nf} & A_{N1} & A_{N2} & \cdots & A_{NN} \end{pmatrix} \tag{A24}$$

Where

$$A_{kk} = \left(\varepsilon_k \rho_k\right)^{n+1} + \delta t \sum_{\substack{l=f,1 \\ l \neq k}}^{N} \left(\beta_{lk}\right)^n \qquad \text{for } k\ (= f,1,2,3,\ldots,N) \tag{A25}$$

$$A_{kl} = A_{lk} = -\delta t \left(\beta_{lk}\right)^n \qquad \text{for } k,l\ 1\ (= f,1,2,3,\ldots,N) \tag{A26}$$

$$A_{i+\frac{1}{2},j} \bullet \begin{pmatrix} u_f \\ u_1 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} \left(\overline{\varepsilon_f \rho_f u_f}\right) - \frac{\delta t}{\delta r_{i+\frac{1}{2}}}\left((p_f)_{i+1,j}^{n+1} - (p_f)_{i,j}^{n+1}\right) - (\varepsilon_f \rho_f) g_r \delta t \\ \left(\overline{\varepsilon_1 \rho_1 u_1}\right) - \frac{\delta t}{\delta r_{i+\frac{1}{2}}}\left((p_1)_{i+1,j}^{n+1} - (p_1)_{i,j}^{n+1}\right) - (\varepsilon_1 \rho_1) g_r \delta t \\ \vdots \\ \left(\overline{\varepsilon_N \rho_N u_N}\right) - \frac{\delta t}{\delta r_{i+\frac{1}{2}}}\left((p_N)_{i+1,j}^{n+1} - (p_N)_{i,j}^{n+1}\right) - (\varepsilon_N \rho_N) g_r \delta t \end{pmatrix} \tag{A27}$$

and similarly, momentum equation in z- direction can be written as

$$(A)_{i,j+\frac{1}{2}} (V)^n_{i,j+\frac{1}{2}} = (B_V)_{i,j+\frac{1}{2}} \tag{A28}$$

$$A_{i,j+\frac{1}{2}} \bullet \begin{pmatrix} v_f \\ v_1 \\ \vdots \\ v_N \end{pmatrix} = \begin{pmatrix} \left(\overline{\varepsilon_f \rho_f v_f}\right) - \frac{\delta t}{\delta z_{j+\frac{1}{2}}}\left((p_f)_{i,j+1}^{n+1} - (p_f)_{i,j}^{n+1}\right) - (\varepsilon_f \rho_f) g_z \delta t \\ \left(\overline{\varepsilon_1 \rho_1 v_1}\right) - \frac{\delta t}{\delta z_{j+\frac{1}{2}}}\left((p_1)_{i,j+1}^{n+1} - (p_1)_{i,j}^{n+1}\right) - (\varepsilon_1 \rho_1) g_z \delta t \\ \vdots \\ \left(\overline{\varepsilon_N \rho_N v_N}\right) - \frac{\delta t}{\delta z_{j+\frac{1}{2}}}\left((p_N)_{i,j+1}^{n+1} - (p_N)_{i,j}^{n+1}\right) - (\varepsilon_N \rho_N) g_z \delta t \end{pmatrix} \tag{A29}$$

Convergence on Fluid Continuity Equation

The solution process proceeds as follows. The calculations are started with a guessed pressure field that is either the specified initial condition or the pressure field computed in the previous time step. Using this guessed pressure field, the velocities are calculated from the matrices above. The particulate phase continuity equations are solved using the updated velocities to calculate the particulate phase volume fractions. The gas volume fraction, ε_f, is them calculated from equation.

$$\varepsilon_f = 1 - \sum_{k=1}^{N} \varepsilon_k \tag{A30}$$

Using ε_f and the updated fluid velocities, the residue of the fluid continuity equations, $D_{i,j}$ is calculated,

$$D_{i,j} = -(\varepsilon_f \rho_f)_{i,j}^{n+1} + (\varepsilon_f \rho_f)_{i,j}^{n} - \frac{\delta t}{\delta r_i}\langle(\varepsilon_f \rho_f)u_f\rangle_{i,j}^{n+1} - \frac{\delta t}{\delta z_j}\langle(\varepsilon_f \rho_f)v_k\rangle_{i,j}^{n+1} \tag{A31}$$

Ideally, for a converged solution, $D_{i,j}$ should be zero. In the code, $D_{i,j}$ is compared with a very small number. The value of the convergence criterion is,

$$D_{i,j} \leq \mathrm{CONV}_{i,j}^{n+1} = \mathrm{EPSG}\ (\varepsilon_f \rho_f)_{i,j}^{n} \tag{A32}$$

where EPSG is read in. The default and recommended value of EPSG is 10^{-5}. The computations begin at the left bottom-corner fluid cell. The pressure is corrected in one cell at a time until convergence is obtained or the number of iterations exceeds and inner iterations limit. The computations proceed from left to right and from bottom to top until the entire computational regime is covered. At the end of such a computational sweep, if a pressure adjustment was necessary in any of the cells, the procedure is repeated until simultaneous convergence in all the cells is obtained. The number of iterations, however, is restricted by an outer iteration limit. Figure A3 illustrates the procedure of computational sweep.

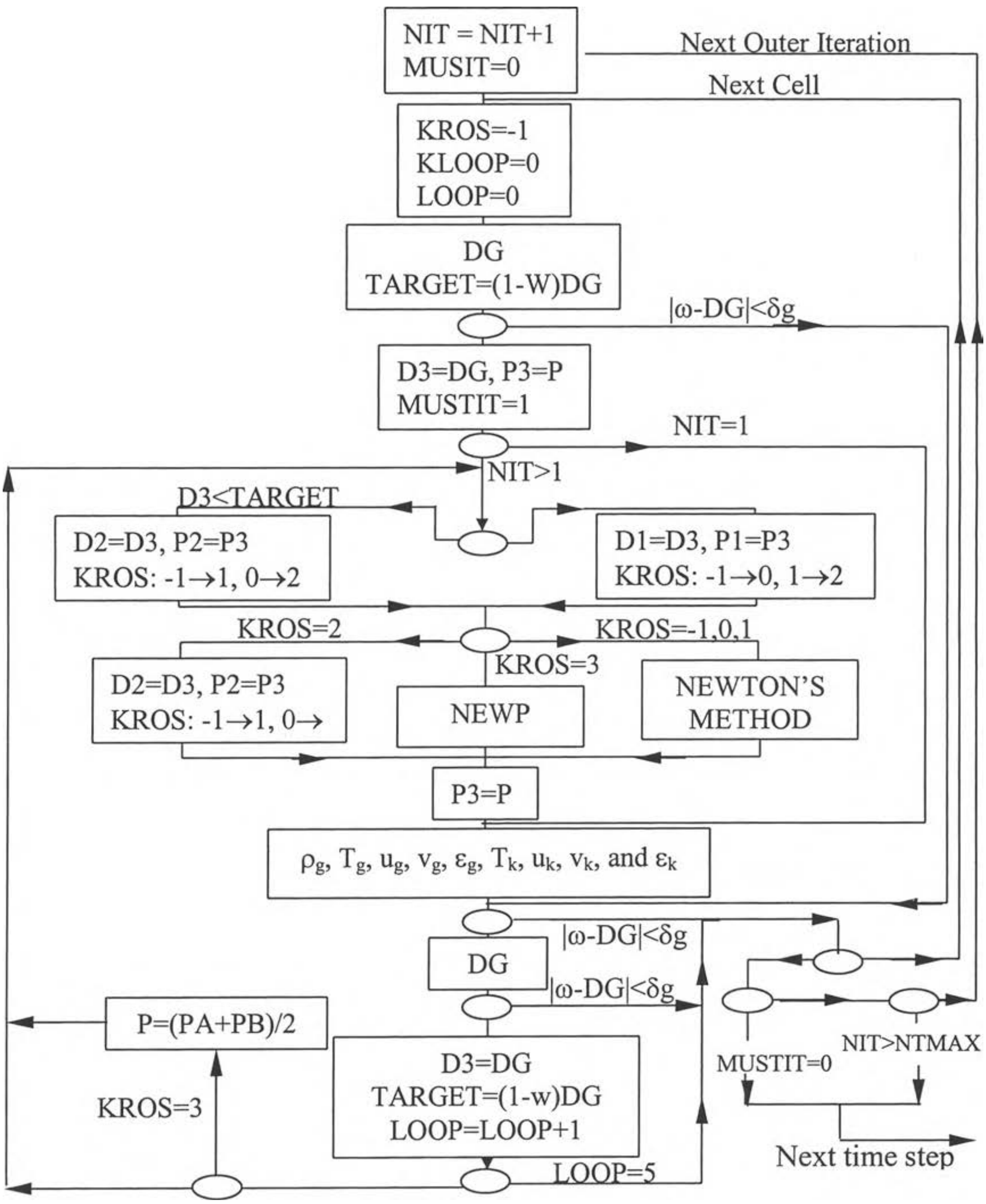

Figure A3 The iterative procedure of the computational sweep.

Pressure Iteration

When $D_{i,j}$ fails to meet the convergence criterion in any cell, the pressure is adjusted using a combination of Newton's method and secant method. The initial adjustment of pressure uses Newton's method,

$$(p_f)^{m+1} = (p_f)^m - \omega \frac{D^m}{\partial D / \partial (p_f)^m} \tag{A33}$$

where the indices i,j, and n have been omitted. The index, m, indicates the iteration level. This is equivalent to using Newton's method for each cell, where ω is a relaxation parameter near unity, and $\overline{\beta}$ is computed as,

$$\frac{1}{\overline{\beta}_{i,j}} = \frac{\partial D_{i,j}}{\partial (p_f)_{i,j}} = \frac{\varepsilon_f}{C_{i,j}^2} + \frac{1}{R_i}\left(\frac{\delta t}{\delta r_i}\right)^2 \left(R_{i+\frac{1}{2}}(\varepsilon_f)_{i+\frac{1}{2},j} + R_{i-\frac{1}{2}}(\varepsilon_f)_{i-\frac{1}{2},j}\right)$$
$$+\left(\frac{\delta t}{\delta z_j}\right)^2 \left((\varepsilon_f)_{i,j+\frac{1}{2}} + (\varepsilon_f)_{i,j-\frac{1}{2}}\right) \tag{A34}$$

once every time step. The sound speed $C_{i,j}$ is given by,

$$C_{i,j}^2 = \left(\frac{\delta p_f}{\delta \rho_f}\right)_{i,j} \tag{A35}$$

where $(\delta p_f / \delta \rho_f)$ can be determined from the equation of state.

This formulation is only approximate. Hence, subsequent correcions use the secant method:

$$(p_f)^{m+1} = (p_f)^m - \omega\left(\frac{(p_f)^{m-1} - (p_f)^m}{D^{m-1} - D^m}\right)D^m \tag{A36}$$

The use of secant method is continued until $D_{i,j}$ changes sign. Thereafter a combination of the secant method and a bisection method is used.

Given the three pressures p_1, p_2, and p_3 of which p_1 and p_2 bracket the desired pressure and p_3 lies between them and the respective mass residuals D_1, D_1, and D_3, do not satisfy the convergence criterion in cell (i,j), $D_1 > 0$, and $D_2 < 0$. With three pressures and their mass residuals obtained to describe, or otherwise a constrained two sided secant technique is used to obtain further pressure adjustments. From these pressures and their mass residuals, the pressure p_A and p_B are determined by straight line extrapolation and interpolation, respectively, as follows,

$$p_A = \begin{cases} (p_3D_1 - p_1D_3) / (D_1 - D_3) & \text{for } D_1 \neq D_3. \\ (p_2 - p_3) / 2 & \text{for } D_1 = D_3. \end{cases} \tag{A37}$$

and

$$p_B = \begin{cases} (p_3D_2 - p_2D_3) / (D_2 - D_3) & \text{for } D_2 \neq D_3. \\ (p_1 - p_3) / 2 & \text{for } D_2 = D_3. \end{cases} \tag{A38}$$

The new estimate of the advanced time pressure is then computed as,

$$(p_f)^{m+1} = \frac{1}{2}(p_A + p_B) \tag{A39}$$

If the pressure, p_A should lie outside the interval p_1 to p_3, it is given the value $\frac{1}{2}(p_A + p_B)$. After $(p_f)^{m+1}$ is estimated, point 2 is discarded and points 1 and 3 are retained as improved bounds for the next pressure estimate. When $D_{i,j}$ changes sign, the value of $\overline{\beta}_{i,j}$ is also updated for future iterations as,

$$\overline{\beta} = \frac{p_1 - p_2}{D_1 - D_2} \tag{A40}$$

Appendix B Means of subroutines and parameters in simulation.

This program is composed of the various subroutines and functions following below this here:

BDRY Sets the boundary conditions – reflects cell centered quantities

BETAS Calculates the reciprocal derivatives of the mass residuals with respect to pressure $\bar{\beta}_{i,j} = (\partial D / \partial p_f)_{i,j}$ for iteration procedure.

DENSG Calculates the steam density by using empirical correlation for specific volume of saturated steam

DENSL Calculates the steam density by using empirical correlation for specific volume of saturated liquid.

FACES Sets the boundary conditions in the continuous outflow and the rigid cells boundaries.

FLIC Sets cell flogs based on input data.

INDX Calculate indices for array quantities.

ITER Perform the iterative solution of the difference equations of mass momentum and energy equations.

KDRAGS Calculates fluid-liquid or solid drag for low or high particulate phase concentration.

MASFK Calculates mass fluxes for the phases.

MATS Calculates the matrix component for velocity of each component.

MULTI Calculates inter-phase momentum exchange coefficient and particle to particle interaction.

NEWP Calculates a new estimate of advanced time pressure from three (pressure, residual) points.

PROG Control the program flow and output.

QESOL Solves quadratic equation.

SETC Initialize constants, functions, static solids pressure and cohesive force, and calculate the volume fraction for each phase; SET =setup, and C = constant.

SETPRE Sets initial pressure profile: SET = setup, and PRE = pressure.

SETRZ	Initialize the radii for the variable computing mesh; SET = setup, and RZ = r- and z- direction.
SETUP	Defines the initial values of field variables in the fluid, inflow and outflow boundary cells, using the input data.
TAPERD	Read the restart file for initial conditions; TAPE = tape, and RD = read.
TAPEWR	Write to a restart file; TAPE = tape, and WR = write.
TILDE	Calculate momentum due to convection, gravity, viscous stress, particulate phase pressure and cohesive stress (tilde quantities).
ULMOMF	Calculates fluxes of radial momentum for the phases.
ULVS	Calculates stress tensor terms for the phases in radial direction.
VARDT	Adjusts time increment δt during execution of the program.
VELINV	Uses Gauss-Dolittle method for symmetric matrix inversion.
VELSK	Calculates velocities on the four boundaries of the cell.
VLMOMF	Calculate fluxes of axial momentum for the phases.
VLVS	Calculates stress tensor term for the phases in axial direction.
VRELS	Calculates square of relative velocity between fields.
PREFILE	This subroutine is for pressure profile if the pressure is constant at the top of the bed while the pressure at the bottom of the bed is being changed.

Description about variable in "misb.com".

NI	Number of x-axis's cells or cells in the radial direction.
NJ	Number of y-axis's cells or cells in the axial direction.
NS	Number of solid phases.
NO	Number of fluid, inflow and outflow blocks.
NT	Number of total blocks defined.
C1	First coefficient of saturated vapor temperature function.
C2	Second coefficient of saturated vapor temperature function.
C3	Third coefficient of saturated vapor temperature function (-6.064).
C4	Steam specific heat ($1.54*10^7$ erg/g*°C)

C5 Reference temperature T_o for fluid and particulate phases at which reference specific energy is zero (value = 300 K).

C6 steam specific internal energy at reference temperature 373 K ($2.509*10^{10}$ erg/g*°C).

C7 Water specific heat ($4.22*10^{7}$ erg/g*°C).

C8 Water specific internal energy at reference temperature 373 K ($4.19*10^{9}$ erg/g*°C).

But C1, C2, C3 , … ,C8 are not used in the program.

C9, C10, C11 and C12 are parameters in equation of state:

$$\rho_f = C9 + \frac{C10 * p_f}{C11 * p_f + C12 * T} \qquad g/cm^3$$

where C12 = R/M

R = Gas constant = $8.3143*10^{7}$ erg / gmole / K

M = Molecular weight (g / gmole)

But these value is defined in misb.com :

C9 = 0.0, C10 = 1., C11 = 0

Therefore,

$$\rho_f = \frac{p_f}{C12 * T}; \qquad C12 = R / M$$

And T = Temperature (K)

C13 and C14 are parameters in cohesive stress function:

$$\tau_{ck} = 10^{-C13*\varepsilon_k + C14} dynes / cm^2$$

C15 and C16 are parameters in solids stress function:

$$G(\varepsilon_k) = 10^{-C15*\varepsilon_k + C16} dynes / cm^2$$

Where C15 = 8.686 and C16 = 6.385

Table B1 A List of FORTRAN Symbols

Symbol	Description
ABETA(IJ)	$1/\beta_{i,j} = (\delta D / \delta p_f)^{-1}_{i,j}$ computed in **BETAS** and **ITER**.
AKL(K,IJ)	Thermal conductivity of fluid and particulate phases.
APP(M,IJ)	$\delta t \sum_{l=1}^{N} (\beta_{kl})_{i,j}$ as diagonal element.
AR(I)	$\delta r_{i+1}/2\delta r_{i+1/2}$
AU(N,N)	Component of matrix **A** for cell (I+1/2,j).
AU1(N,N)	Component of matrix **A** for cell (i-1/2,j).
AV(N,N)	Component of matrix **A** for cell (i,j+1/2).
AV1(N,N)	Component of matrix **A** for cell (i,j-1/2).
AZ(J)	$\delta z_{j+1}/2\delta z_{j+1/2}$
BR(I)	$\delta r_i/2\delta r_{i+1/2}$=1-AZ(I).
BU(N,N)	Component of matrix **B** for cell (I+1/2,j).
BU1(N,N)	Component of matrix **B** for cell (i-1/2,j).
BV(N,N)	Component of matrix **B** for cell (i,j+1/2).
BV1(N,N)	Component of matrix **B** for cell (i,j-1/2).
BZ(J)	$\delta z_j/2\delta z_{j+1/2}$=1-AZ(J).
CL(K)	Heat capacity(erg/(g*K)); K=0 for continuous phase and K > 0 for dispersed phase; K=0,NSOLID.
COHF(0:1000)	Cohesive stress values (τ_{ck}).
CONV(IJ)	Pressure iteration convergence criteria computed in **BETAS**.
CPHI(K)	ϕ_k = Packing fraction (ε_f) of each phases at maximum packing; K=0 for continuous phases, and K>0 for dispersed phases; K=0,NSOLID.
CRES	Coefficient of restitution.
D1,D2,D3	Values of $D_{i,j}$.
DENOM	=$d_k * \psi_k$ = DK(K)*PHI(K); K=0 for continuous phase and K>0 for dispersed phase.
DG	$D_{i,j}$ – Tolerance in fluid continuity equation.

Symbol	Description
DK(K)	Diameter of each phases; K=0 for continuous phases, and K>0 for dispersed phases; K=0,NSOLID.
DKF(K,K)	$\delta t \rho_k \rho_l (d_k + d_l)^3 / 2\varepsilon_f (\rho_k d_k^3 + \rho_l d_l^3)$
DR(I)	δr of each cell in x- or r- direction at i-th column; I=1,IB2.
DRCOE	C_D = drag coefficient = 0.44 at $Re_k \geq 1000$.
DRP(I)	$\delta r_{i+1/2}$ of each cell in x- or r- direction at the $(i+1/2)^{th}$ column; I=1,IB2..
DT	Δt for running simulation (second).
DTNXT	Time step changes supplies (second).
DTOBDR(IJ)	$\delta t / r_{i+1/2} \delta r_{i+1/2}$
DTODR(IJ)	$\delta t / \delta r_I$
DTODRP(IJ)	$\delta t / \delta r_{I+1/2}$
DTODZ(IJ)	$\delta t / \delta z_j$
DTODZP(IJ)	$\delta t / \delta z_{j+1/2}$
DZ(J)	δz_j of each cell in y- or z- direction at j^{th} column; J=1,JB2.
DZP(J)	$Dz_{j+1/2}$ of each cell y- or z- direction at (j+1/2) th column; J=1,JB2.
EPSG	Convergent limit.
EPSL	$[(\phi_k - \phi_l)+(1-\alpha)(1-\phi_k)\, \phi_l][\phi_k+(1-\phi_k)\, \phi_l]$
EPSU	$(1-\alpha)[\phi_k+(1-\phi_k)\, \phi_l]$
GRAVX	Horizontal gravity accelerate (cm/s^2).
GRAVY	Vertical gravity accelerate (cm/s^2). Positive = upward flow, and Negative = downward flow.
GTH(IK)	Solids stress values = $G(\varepsilon_k) = 10^{(-C15*\varepsilon k + C16)}$.
I	i –Computing mesh column index (x- or r- direction).
IB	Number of cell in the radial direction excluding the two fictitious columns along the boundaries; (IB = IB2-2).
IB1	IB2-1
IB1JB2	IB1*JB2 =(IB2-1)*JB2 = IB2*JB2 – JB2
IB2	Number of cells in x- or r- direction.

Symbol	Description
IB2JB1	IB2*JB1 = IB2*(JB2 - 1) = IB2*JB2 – IB2
IB2JB2	IB2*JB2
ICOH	Cohesive force model; 0 -No, and >0 –Yes.
IFL(IJ)	Cell flags set up in **FLIC**.
IFLZB	Fluidized bed; 0 -No, and 1 -Yes.
IJ	Index of quantities for cell(i,j); IJ = I+(J-1)IB2.
IJB	Index of cell centered quantities associated with cell (i,j-1).
IJBR	Index of cell centered quantities associated with cell (i+1,j-1).
IJL	Index of cell centered quantities associated with cell (i-1,j).
IJM	Index of cell (i,j-1).
IJP	Index of cell (i,j+1).
IJR	Index of cell centered quantities associated with cell (i+1,j).
IJRR	Index of cell centered quantities associated with cell (i+2,j).
IJT	Index of cell centered quantities associated with cell (i,j+1)
IJTL	Index of cell centered quantities associated with cell (i-1,j+1)
IJTR	Index of cell centered quantities associated with cell (i+1,j+1)
IJTT	Index of cell centered quantities associated with cell (I,j+2)
IKINT	Kinetic theory of granular solids model.
IMJ	Index of cell (i-1,j).
IMJM	Index of cell (i-1,j-1).
IMJP	Index of cell (i-1,j+1)
INDC( )	Storage of IPJ, IMJ, IJP, IJM, IPJP, IMJP, IPJM, and IMJM for each (i,j).
INDS( )	Storage of IJR, IJL, IJT, IJB, IJTR, IJTL, IJBR, IJRR, and IJTT for each (i,j).
INENT	Energy equations; 1 – Internal energy, 2 – Enthalpy.
IOB(M,N)	Coordinates; IOB(1,N)=Value of started x, IOB(2,N)=Value of final x, IOB(3,N)=Value of started y, and IOB(4,N)=Value of final y; M=1,4 and N=1,NTOT.
IPJ	Index of cell (i+1,j).

Symbol	Description
IPJM	Index of cell (i+1,j-1).
IPJP	Index of cell (i+1,j+1).
IPRES	Initial pressure profile; 0=Fluid weight, and 1=Bed weight.
IREVS	Cylindrical coordinate specification; IREVS=0 for center line at left side, IREVS=1 for center line at right side, and IREVS=2 for center line at center of the bed.
ISTW	System property (not used for fluidized bed); ISTW=0 for air/water system, ISTW=1 for fluidized bed; and else for steam/water system.
ISWIT	Switch continuous and dispersed phases; 0 –No, and 1 -Yes.
ITC	Set coordinates system; 0=Cartesian, and 1=Cylindrical.
ITD	ITD=0 (No restart), and ITD=2 (Restart)
ITX	Number of loops for iteration.
ITXMX	Number of DTNXT (Time step changes supplies).
J	j – Computing mesh index (y- or z- direction).
JB	Number of cell in the axial direction excluding the two fictitious rows along the boundaries; JB = JB2-2.
JB1	JB2-1.
JB2	Number of cells in y – direction.
K	Index for the phases.
KV	Index for continuous phase.
LCAT1	Locations of leg II.
LCAT2	Locations of leg III (not used for fluidized bed).
MAXIT	Maximum number of iterations.
MODAB	Model of professor Dimitri Gidaspow; 1=Model A and 2=Model B.
NAME	Problem identifier – input data line2.
NC	Total number of cells ($\geq$ IB2JB2).
NCAL	Number of total blocks, excepted obstacle = NFL(fluid) + NIN (inflow) + NOUT(outflow).
NCONT	Number of continuous phases; 1=Gas/Liquid, and 2= Gas & Liquid.
NF	NPHASE*(NPHASE+1)/2.
NFL	Number of fluid blocks.

Symbol	Description
NIMBR	0 – branching tee, =1 –fluidized bed, and else - impacting tee.
NIN	Number of inflow blocks.
NIT	Iteration counter used in **ITER**.
NO	Number of inflow, outflow and fluid blocks.
NOBS	Number of obstacle blocks.
NOUT	Number of outflow blocks.
NP	Total number of phases ($\geq$ NPHASE).
NPHASE	Number of total phases.
NS	Total number of solids phases ($\geq$ NSOLID).
NSL(1)	Boundary condition at bottom of column; 1-Active, 2-Free slip, 3-No slip for gas/liquid, 4-Outflow(dP|exit=0), 5-Inflow(Flow&P), 6-Inflow, 7-Outflow(P|exit=const), 8-Fluid only outflow(P|exit=const).
NSL(2)	Boundary condition at left of column.
NSL(3)	Boundary condition at top of column.
NSL(4)	Boundary condition at right of column.
NSO(N)	Flag of the N^{th} blocks on zone 6 in "sam.d"; 1-Active, 2-Free slip, 3-No slip for gas/liquid, 4-Outflow(dP|exit=0), 5-Inflow(Flow&P), 6-Inflow, 7-Outflow(P|exit=const), 8-Fluid only outflow(P|exit=const); N=1,NTOT.
NSOLID	Number of solids in the system.
NT	Number of total obstacles.
NTHS(K)	Continuous phase number (0 or 1).
NTOT	Number of total blocks = NFL(fluid) + NIN(inflow) + NOUT (outflow) + NOBS(obstacle).
P(IJ)	$(p_f)_{i,j}$ – Pressure in cell (i,j).
P1,P2,P3	Values of $(p_f)_{i,j}$.
PHI(K)	Sphericity shape factor of each phases; K=0 for continuous phases, and K>0 for dispersed phases; K=0,NSOLID.
PHILIM(K,KK)	X_k.
PIO(N)	Pressure of fluid or continuous phase; ;N=1,NCAL.

Symbol	Description
PLP(K)	D_k/ε_k^3.
PN(IJ)	$(p_f)_{i,j}^n$ – Pressure in cell (i,j) at time level n.
PS(K,IJ)	$(p_f)_{i,j}$ – Solids Pressure in cell (i,j).
PVISC(K)	Viscosity coefficient of each phases; K=0 for continuous phases, and K>0 for dispersed phases; K=0,NSOLID.
R(I)	r_I – Radial coordinate of the center of cell (i,j).
RADP	Pipe radius (cm).
RAGS	C_f^{-2} – Square of the reciprocal fluid adiabatic sound speed.
RB(I)	$r_{I+1/2}$ – Radial coordinate of the right boundary of cell (i,j).
RDR(I)	$1/\delta r_i$.
RDRP(I)	$1/\delta r_{i+1/2}$.
RDZ(J)	$1/\delta z_j$.
RDZP(J)	$1/\delta z_{j+1/2}$.
REYN	Reynolds Number = $(\varepsilon_f\rho_f)(\lvert vf\text{-}vk\rvert)(d_k\psi_k)/\mu_f$ = RLK(KV,IJ) * VREL (K) * DENOM(K) / VISCL(KV,IJ).
RGP(IJ)	$(\varepsilon_f\rho_f)_{i,j}$ – Macroscopic density of the fluid for cell (i,j).
RKPG(K,IJ)	$\beta_{fk} = \beta_{kf}$ =Fluid – particulate phase drag for cell (i,j).
RL(K)	Density of each phases; K=0 for continuous phases, and K>0 for dispersed phases; K=0,NSOLID.
RLFRK(K,IJ)	Flux of $(\varepsilon_k\rho_k)$ across the right boundary of cell (i,j).
RLFTK(K,IJ)	Flux of $(\varepsilon_k\rho_k)$ across the top boundary of cell (i,j).
RLIM	Number of the minimum particles per cubic centimeter.
RLK(K,IJ)	$(\varepsilon_k\rho_k)_{i,j}$ – Macroscopic phase density of the fluid for cell (i,j).
RLKN(K,IJ)	$(\varepsilon_k\rho_k)_{i,j}^n$.
RLX(K,IJ)	Mixture density for cell (i,j) = $\rho_{bulk,ij} = (\varepsilon_f\rho_f + \varepsilon_k\rho_k)_{i,j}$.
ROG(IJ)	$(\rho_f)_{i,j}$ – Microscopic density of the fluid for cell (i,j).
ROK(IJ)	$(\rho_k)_{i,j}$ – Microscopic density of the fluid for cell (i,j).
RRB(I)	$1/r_{i+1/2}$.
RRIDR(I)	$1/r_i\delta r_i$.

Symbol	Description
RRIDRP(I)	$1/r_{i+1/2}\delta r_{I+1/2}$.
RST	Distance of the first cell in x- or r- direction from origin.
RUK(K,IJ)	$\left(\overline{\varepsilon_k \rho_k u_k}\right)$.
RVK(K,IJ)	$\left(\overline{\varepsilon_k \rho_k v_k}\right)$.
SULB(K)	$(R\varepsilon_k\tau_{k,rz})_{I,j}$ = Product of radius, volume fraction, and shear stress.
SULC(K)	$(R\varepsilon_k\tau_{k,\phi\phi})_{I+1/2,j}$ = Product of volume fraction, and azimuthal stress.
SULL(K)	$(R\varepsilon_k\tau_{k,rr})_{i,j}$ = Product of radius, volume fraction, and radial stress.
SULR(K)	$(R\varepsilon_k\tau_{k,rr})_{I+1,j}$ = Same as SULL(K), but evaluated at cell location (i+1,j).
SULT(K)	$(R\varepsilon_k\tau_{k,rz})_{I+1/2,j+1/2}$ = Same as SULB(K), but evaluated at cell location (i+1,j+1).
SVLB(K)	$(R\varepsilon_k\tau_{k,zz})_{I,j}$ = Product of volume fraction, and axial stress.
SVLL(K)	$(R\varepsilon_k\tau_{k,rz})_{I-1/2,j+1/2}$ = Product of radius, volume fraction, and axial stress.
SVLR(K)	$(R\varepsilon_k\tau_{k,rz})_{I+1/2,j+1/2}$ = Same as SVLL(K), but evaluated at cell location (i+1/2,j+1/2).
SVLT(K)	$(R\varepsilon_k\tau_{k,zz})_{I,j+1}$ = Same as SVLB(K), but evaluated at cell location (i,j+1).
SVREL(K)	Square of relative velocity of particulate.
TARGET	Used in the pressure iteration to provide over or under relaxation.
TDUMP	Dump time (second).
TEMIO(K,N)	Temperature; K=0 (velocity of fluid) and K>0 (velocity of solid); K=0,NSOLID and N=1,NCAL.
TH(K,IJ)	$\varepsilon_{k,ij}$ = Volume fraction for cell (i,j).
THIO(K,N)	Volume fraction; K=0 (velocity of fluid) and K>0 (velocity of solid); K=0,NSOLID and N=1,NCAL.
THMIN	Minimum fluid volume fraction.
TIME	Set start time for running simulation (second).
TPR	Print time (second).
TPRR	Kept data time.

Symbol	Description
TSKIO(K,N)	Granular temperature; K=0 for continuous phase and K> 0 for dispersed phase; K=0,NSOLID and N=1,NCAL.
TSTOP	Set stop time for running simulation (second).
UIO(K,N)	Velocity in x- or r- direction; K=0 (velocity of fluid) and K>0 (velocity of solid); K=0,NSOLID and N=1,NCAL..
UK(K,IJ)	$(u_k)_{i+1/2,j}$ – Volume fractions for cell (i,j).
ULFB(K,IJ)	Radial momentum flux for the phases across the bottom boundary of the radial momentum control volume centered about the point (i+1/2,j).
ULFL(K,IJ)	Radial momentum flux for the phases across the left boundary of the radial momentum control volume centered about the point (i+1/2,j).
ULFR(K,IJ)	Radial momentum flux for the phases across the right boundary of the radial momentum control volume centered about the point (i+1/2,j).
ULFT(K,IJ)	Radial momentum flux for the phases across the top boundary of the radial momentum control volume centered about the point (i+1/2,j).
VIO(K,N)	Velocity in y- or z- direction; K=0 (velocity of fluid) and K>0 (velocity of solid); K=0,NSOLID and N=1,NCAL.
VISCL(K,IJ)	$(\mu_k)_{i,j}$ – particulate shear viscosity for cell (i,j).
VK(K,IJ)	$(v_k)_{i,j+1/2}$ – Axial velocity for cell (I,j).
VLFB(K,IJ)	Axial momentum flux for the phases across the bottom boundary of the radial momentum control volume centered about the point (I,j+1/2).
VLFL(K,IJ)	Axial momentum flux for the phases across the left boundary of the radial momentum control volume centered about the point (I,j+1/2).
VLFR(K,IJ)	Axial momentum flux for the phases across the right boundary of the radial momentum control volume centered about the point (I,j+1/2).
VLFT(K,IJ)	Axial momentum flux for the phases across the top boundary of the radial momentum control volume centered about the point (I,j+1/2).
VREL(K)	$\|v_f - v_k\|$ = Velocity of solid phase relative to fluid phase = (DU*DU+DV*DV)**0.5 .

APPENDIX C IIT code.

```
C--------------------------------------------------------+
C              TRANSIENT TWO-DIMENSIONAL              |
C               MULTIPHASE FLOW PROGRAM               |
C        WITH MODEL-A/-B & STEAM PROPERTY CORRECTIONS    |
C              Version 3.3 (October '94)           |
C                                       |
C                  MODIFIED BY                    |
C           BING SUN & DR. DIMITRI GIDASPOW            |
C                                       |
C   ILLINOIS INSTITUTE OF TECHNOLOGY, CHICAGO IL 60616    |
C                                       |
C-------------------MAINPROGRAM----MICEFLOW----------------+
      PROGRAM MICEFLOW
      INCLUDE 'misb.com'
      CHARACTER*10 RESFIL,GAS,WATER,RATIO
      CHARACTER*80 NAME
C
C   READ AND ECHO INPUT DATA
        OPEN(5,FILE='sam.d1')
        OPEn(6,FILE='sam.out')
      READ(5,'(A)')RESFIL
      OPEN(9,FILE=RESFIL,FORM='UNFORMATTED',STATUS='UNKNOWN')
      READ(5,'(A)')NAME
      WRITE(6,'(A)')' MULTIFLOW PROBLEM IDENTIFIER -  ',NAME
      READ(5,'(A)')GAS
      OPEN(12,FILE=GAS)
      READ(5,'(A)')WATER
      OPEN(13,FILE=WATER)
      READ(5,'(A)')RATIO
      OPEN(16,FILE=RATIO)
```

```
      READ(5,*)ITC,IB2,JB2,NSOLID
      WRITE(6,210)ITC,IB2,JB2
C
      NPHASE=NSOLID+1
      IB=IB2-2
      IB1=IB2-1
      JB=JB2-2
      JB1=JB2-1
      IB2JB2=IB2*JB2
      IB2JB1=IB2JB2-IB2
      IB1JB2=IB2JB2-JB2
C
      READ(5,*)RST,(DR(I),I=1,IB2)
      WRITE(6,215)RST,(DR(I),I=1,IB2)
      READ(5,*)(DZ(J),J=1,JB2)
      WRITE(6,216)(DZ(J),J=1,JB2)
      READ(5,*)NCONT,MODAB,IPRES,ICOH,ISWIT,IFLZB
      WRITE(6,320)NCONT,MODAB,IPRES,ICOH,ISWIT,IFLZB
      WRITE(6,250)NPHASE
      DO 6 K=0,NSOLID
      READ(5,*)DK(K),RL(K),PHI(K),CPHI(K),PVISC(K)
      WRITE(6,255)DK(K),RL(K),PHI(K),CPHI(K),PVISC(K)
6     CONTINUE
      READ(5,*)(NSL(M),M=1,4)
      WRITE(6,220)(NSL(M),M=1,4)
      READ(5,*)NFL,NIN,NOUT,NOBS
      WRITE(6,230)NFL,NIN,NOUT,NOBS
C
      NCAL=NIN+NOUT+NFL
      NTOT=NCAL+NOBS
C
      WRITE(6,240)
```

```
      DO 5 N=1,NTOT
      READ(5,*)NSO(N),(IOB(M,N),M=1,4)
5     WRITE(6,245)NSO(N),(IOB(M,N),M=1,4)
      WRITE(6,280)
      DO 10 N=1,NCAL
      READ(5,*)UIO(0,N),VIO(0,N),PIO(N),THIO(0,N),TEMIO(0,N)
      READ(5,*)(UIO(K,N),VIO(K,N),THIO(K,N),TEMIO(K,N)
     1 ,K=1,NSOLID)
10    WRITE(6,285)N-1,UIO(0,N),VIO(0,N),PIO(N),THIO(0,N)
     1,TEMIO(0,N)
      WRITE(6,290)
      DO 15 N=1,NCAL
15    WRITE(6,295)(N-1,K,UIO(K,N),VIO(K,N),THIO(K,N)
     1,TEMIO(K,N),K=1,NSOLID)
      READ(5,*)ITD
      READ(5,*)TIME,TSTOP,DT,ITX
      READ(5,*)TPR,TDUMP,TPRR
      WRITE(6,300)ITD,TIME,TSTOP,ITX,DT,TPR,TDUMP
      READ(5,*)ITXMX,(DTNXT(IX),IX=1,ITXMX)
      WRITE(6,305)ITXMX,(DTNXT(IX),IX=1,ITXMX)
      READ(5,*)GRAVX,GRAVY
      WRITE(6,310)GRAVX,GRAVY
      READ(5,*)EPSG,THMIN,RLIM,ISTW
      WRITE(6,330)EPSG,THMIN,RLIM
      WRITE(6,350)
      WRITE(6,340)ISTW
      READ(5,*)NIMBR,LCAT1,LCAT2
      WRITE(6,370)
      WRITE(6,380)NIMBR
      WRITE(6,390)LCAT1,LCAT2
C  READ RESTART FILE IF NECESSARY
      REWIND(9)
```

```
      IF(ITD.EQ.2)CALL TAPERD
C
C  INITIALIZE CELL FLAGS, CONSTANTS, AND DEPENDENT VARIABLES
      CALL FLIC
      CALL SETUP
C  MARCH IN TIME>>>>>
      CALL PROG
      STOP
210   FORMAT(' 1. GEOMETRY'/'   A. COORDINATES (0- CARTESIAN,'
     1,' 1- CYLINDRICAL, 2- SPHERICAL)=',I2/'   B. MESH SIZE',
     1'   R (or X)-dir, IB2=',I3,',     ',
     1'Z (or Y)-dir, JB2=',I3/'   C. CELL SIZES')
215   FORMAT(7X,'Distance of FIRST Cell from Center = ',F10.5,
     1' cm'/7X,'In R (or X)-direction, DR (cm) ='/6(2X,F9.5))
216   FORMAT(7X,'In Z (or Y)-direction, DZ (cm) ='/6(2X,F9.5))
320   FORMAT(/' 2. MODELS: (=0- NO, >0- YES)'/10X,'# OF ',
     1'CONTINUOUS PHASES (1- GAS/LIQUID, 2- GAS&LIQUID) = ',I2
     1/10X,'MODEL (1- A, 2-MOD. B)= ',I2/10X,'INITIAL ',
     1'PRESSURE PROFILE (0- FLUID WT., 1- BED WT.) = ',I2
     1/10X,'COHESIVE FORCE MODEL=',I2/10X,'SWITCH CONTINUOUS'
     1,' & DISPERSED PHASES (0- NO, 1- YES) = ',I2,
     1/10X,'FLUIDIZED BED (0- NO, 1- YES) = ',I2)
250   FORMAT(/' 3. DATA FOR ',I2,' PHASES :'/4X,'DIAMETER   ',
     1' DENSITY    SPHERICITY   PACKING      ',
     1'VISCOSITY'/40X,'FRACTION   COEFFICIENT'/
     1 6X,'(cm)',5X,'(g/cm^3)  ',27X,'(g/(cm.s))')
255   FORMAT(2X,2(2X,G9.3),2X,G10.3,4X,F6.4,5X,G9.4,3X,G10.3)
220   FORMAT(/' 3. CELL FLAGS'/6X,'(1- FLUID, 2- FREE SLIP,',
     1' 3- NO SLIP, 4- OUTFLOW (dP|exit = 0)'/7X,'5-',
     1' INFLOW (FLOW & P), 6- INFLOW, 7- OUTFLOW (P|exit',
     1' = const)'/7X,'8- FLUID OUTFLOW (P|exit = const))'/
     1'   A. BOUNDARIES'/7X,'BOTTOM=',I3,
```

```
     1'   LEFT=',I3,'   TOP=',I3,'  RIGHT=',I3)
230  FORMAT(4X,'B. NUMBERS OF:  FLUID BLOCKS=',I2,
     1',  INFLOW BLOCKS=',I2/20X,'OUTFLOW',
     1' BLOCKS =',I2,', OBSTACLE BLOCKS=',I2)
240  FORMAT(7X,'FLAG',5X,'---------COORDINATES---------')
245  FORMAT(5X,I3,4(4X,I5))
280  FORMAT('   C. INFLOW - OUTFLOW DATA  (FLUID)'/' BLOCK',
     1 '  UIO',8X,'VIO',8X,'PIO',8X,'THIO',6X,'TEMIO'
     1/7X,'(cm/s)',5X,'(cm/s)  (dynes/cm^2)',12X,'(Kelvin)')
285  FORMAT(1X,I2,1X,5(1PE11.4),2(1PE10.3))
290  FORMAT('   D. INFLOW - OUTFLOW DATA  (SOLIDS)'/
     1 ' BLOCK PHASE',6X,'UPIO',9X,'VPIO',8X,'THPIO',8X,
     1 'TEMPIO',7X,/17X,'(cm/s)',7X,'(cm/s)'
     1,6X,'(Kelvin)',4X,'((cm/s)^2)')
295  FORMAT((2X,I2,3X,I3,2X,5(2X,1PE11.4)))
300  FORMAT(/' 6. CONTROL'/ 3X,'A. DUMP AND RESTART:',
     1'   ITD=',I2,' (0- NO RESTART, 2- RESTART)'/'   B. ',
     1'TIME (secs.):  TSTART=',1PE11.4,',  TSTOP=',1PE11.4,
     1',  DT(',I2,')=',1PE11.4/'   C. PRINTING AND ',
     1'PLOTTING (secs.):   TPR=',1PE11.4,',  TDUMP=',1PE11.4)
305  FORMAT(3X,'D. TIME INCREMENTS:   TOTAL=',I2,3X,
     1 'DELTA T (secs.) = '/(8X,6(1X,1PE10.4)))
310  FORMAT(/' 7. GRAVITY (cm/s^2)'/'   A. GRAVX ',
     1 '- R (or X) component =',1PE15.7/,
     1 6X,'GRAVY - Z (or Y) component =',1PE15.7)
330  FORMAT(/' 8. OTHER:'/4X,
     1 'CONVERGENCE LIMIT = ',G10.4/
     1 4X,'MINIMUM FLUID VOLUME FRACTION = ',F9.4/
     1 4X,'FACTOR FOR MIN. SOLIDS VOL. FRACTION = ',G10.4/)
350  format(/' 9. AIR/WATER AND STEAM/WATER SWITCH:')
340  format(4x,'ISTW=',i3,/2x,'  ISTW=0 FOR AIR/WATER SYSTEM'
     1,1x,';',2x,'ELSE FOR STEAM/WATER SYSTEM')
```

```
370  FORMAT(/'10. BRANCHING TEE / IMPACTING TEE SWITCH:')
380  FORMAT(4X,'NIMBR=',I3,/2X,'  NIMBR=0 FOR BRANCHING TEE'
   1,1X,';',2X,'NIMBR=1 FOR FIUIDIZED BED',1X,';',2X,
   1'ELSE FOR IMPACTING TEE')
390  FORMAT(4X,'LOCAT1(LOCATION FOR LEG II)=',I2,4X,'LOCAT2
   1(LOCATION FOR LEG III)=',I2)
     END
C  ----------------------------------------BDRY
     SUBROUTINE BDRY
     INCLUDE 'misb.com'
C
C  SETS BOUNDARY CONDITIONS - REFLECTS CELL CENTER
QUANTITIES
C
     DO 200 J=2,JB1
     DO 200 I=2,IB1
     IJ=I+(J-1)*IB2
C  SKIP IF NOT A FLUID CELL
     IF(IFL(IJ).EQ.1)THEN
C
     IPJ=INDC(IJ,1)
     IMJ=INDC(IJ,2)
     IJP=INDC(IJ,3)
     IJM=INDC(IJ,4)
     IPJP=INDC(IJ,5)
     IMJP=INDC(IJ,6)
     IPJM=INDC(IJ,7)
     IJR=INDS(IJ,1)
     IJL=INDS(IJ,2)
     IJT=INDS(IJ,3)
     IJB=INDS(IJ,4)
C
```

```
      DO 10 K=NCONT+1,NSOLID
      IF(RLK(K,IJ).GT.0.0)THEN
      PLP(K)=DK(K)/TH(K,IJ)**(1./3.)
      ELSE
      PLP(K)=0.0
      ENDIF
10    CONTINUE
C
C  CHECKS OUTFLOW CELLS ON RIGHT AND TOP
      NFLX=IFL(IPJ)
      NFLXY=IFL(IPJP)
      NFLY=IFL(IJP)
      NFLYX=IFL(IPJP)
      IJX=IJR
      IJBX=IPJ
      IJY=IJT
      IJBY=IJP
      RBPM=RB(I+1)
C
      CALL FACES
C  CHECKS OUTFLOW CELLS ON LEFT AND BOTTOM
      NFLX=IFL(IMJ)
      NFLXY=IFL(IMJP)
      NFLY=IFL(IJM)
      NFLYX=IFL(IPJM)
      IJX=IJL
      IJBX=IMJ
      IJY=IJB
      IJBY=IJM
      RBPM=RB(I-1)
C
      CALL FACES
```

```
      ENDIF
 200  CONTINUE
      RETURN
      END
C  ----------------------------------------BETAS
      SUBROUTINE BETAS
      INCLUDE 'misb.com'
C
C  CALCULATES RECIPROCAL DERIVATIVES OF D WRT P, ABETA(IJ),
C  FOR ITERATION
C
      DO 10 J=2,JB1
      DO 10 I=2,IB1
      IJ=I+(J-1)*IB2
      IF(IFL(IJ).NE.1)GOTO 10
      IPJ=INDC(IJ,1)
      IMJ=INDC(IJ,2)
      IJP=INDC(IJ,3)
      IJM=INDC(IJ,4)
      IJR=INDS(IJ,1)
      IJL=INDS(IJ,2)
      IJT=INDS(IJ,3)
      IJB=INDS(IJ,4)
      KV=NTHS(IJ)
C
      IF(IFL(IPJ).EQ.1.OR.IFL(IPJ).EQ.4.OR.IFL(IPJ).GE.7)THEN
      RIG=RB(I)*(AR(I)*TH(KV,IJ)+BR(I)*TH(KV,IJR))*DTODRP(I)
      ELSE
      RIG=0.
      ENDIF
      IF(IFL(IMJ).NE.2.AND.IFL(IMJ).NE.3.AND.IFL(IMJ).NE.5)
     1 THEN
```

```
      EFL=RB(I-1)*(BR(I-1)*TH(KV,IJ)+AR(I-1)*TH(KV,IJL))
     1*DTODRP(I-1)
      ELSE
      EFL=0.
      ENDIF
      IF(IFL(IJP).EQ.1.OR.IFL(IJP).EQ.4.OR.IFL(IJP).GE.7)THEN
      TOP=(AZ(J)*TH(KV,IJ)+BZ(J)*TH(KV,IJT))*DTODZP(J)
      ELSE
      TOP=0.
      ENDIF
      IF(IFL(IJM).EQ.1.OR.IFL(IJM).EQ.4.OR.IFL(IJM).EQ.6)THEN
      BOT=(BZ(J-1)*TH(KV,IJ)+AZ(J-1)*TH(KV,IJB))*DTODZP(J-1)
      ELSE
      BOT=0.
      ENDIF
C
      CONV(IJ)=EPSG*RLK(KV,IJ)
      RBETA=RLK(KV,IJ)/P(IJ)+DTODZ(J)*(TOP+BOT)
     1+DTORDR(I)*(RIG+EFL)
      ABETA(IJ)=1./RBETA
10    CONTINUE
      RETURN
      END
C------------------------------------------DENSG
      SUBROUTINE DENSG(MM)
      INCLUDE 'misb.com'
C
C    CALCULATE STEAM DENSITY BY USING EMPIRICAL
CORRELATIONS
C    FOR SPECIFIC VOLUME OF SATURATED STEAM
C
      PPC=220.55E+06
```

```
      VVC=3.10559
C     set coefficiens for equations
      AO=0.85188
      A1=-0.26786667
      A2=-0.10696316
      A3=-0.015826285
      A4=-9.0169534E-4
      AAO=2.8570183
      AA1=1.1796202
      AA2=-6.0239948
      AA3=5.5896454
      AA4=2.8224224E-01
      AA5=-1.1378390
      AA6=-4.1258599
      AA7=-5.3664933
      AA8=1.7579898E+01
      AA9=-9.5465496
C
      KKK=0
      IF(P(MM).LE.21.03E+07)KKK=1
      IF(P(MM).LE.2.79E+07)KKK=2
      GOTO(10,11)KKK
11    CC1=ALOG(P(MM)/PPC)
      CC=EXP(AO+A1*CC1+A2*CC1**2.+A3*CC1**3.+
     1A4*CC1**4.)
      GOTO 100
10    CC1=P(MM)/PPC
      CC=AAO+AA1*CC1+AA2*CC1**2.+AA3*CC1**3.+
     1AA4*CC1**4.+AA5*CC1**5.+AA6*CC1**6.+AA7*
     1CC1**7.+AA8*CC1**8.+AA9*CC1**9.
100   RC=1./CC
      ROG(MM)=P(MM)*RC/PPC/VVC
```

```
      RETURN
      END
C-----------------------------------------DENSL
      SUBROUTINE DENSL(LL)
      INCLUDE 'misb.com'
C
C     CALCULATE THE LIQUID DENSITY BY USING THE EMPIRICAL
CORRECTIONS
C     FOR SPECIFIC VOLUME OF SATURATED LIQUID
C
      KKK=0
C     set coefficients for equations
      PPC=220.55E+06
      VVC=3.10559
      AO=3.2931007E-1
      A1=1.6699940
      A2=-4.4002253E+01
      A3=7.3518335E+02
      A4=-4.7497294E+03
      AAO=3.4444232E-01
      AA1=4.6716602E-01
      AA2=-1.1950608
      AA3=2.6415460
      AA4=-2.1301049
      AAAO=-1.4115188
      AAA1=3.2105379
      AAA2=-4.7880621
      AAA3=-2.6579266
      AAA4=1.3685529E+01
      AAA5=-4.9699783E-01
      AAA6=-2.6656921E+01
      AAA7=2.6411591E+01
```

```
      AAA8=-7.5368481
      CC1=P(LL)/PPC
C
      IF(P(LL).LE.20.78E+07)KKK=1
      IF(P(LL).LE.9.20E+07)KKK=2
      IF(P(LL).LE.1.25E+07)KKK=3
      GOTO(10,11,12)KKK
12    CC=AO+A1*CC1+A2*CC1**2.+A3*CC1**3.+
     1A4*CC1**4.
      GOTO 100
11    CC=AAO+AA1*CC1+AA2*CC1**2.+AA3*CC1**3.+
     1AA4*CC1**4.
      GOTO 100
10    RC=AAAO+AAA1*CC1+AAA2*CC1**2.+AAA3*CC1**3.
     1+AAA4*CC1**4.+AAA5*CC1**5.+AAA6*CC1**6.+
     2AAA7*CC1**7.+AAA8*CC1**8.
      CC=EXP(RC)
100   ROK(LL)=1./VVC/CC
      RETURN
      END

C ----------------------------------------FACES
      SUBROUTINE FACES
      INCLUDE 'misb.com'
C
C  CONTINUOUS OUTFLOW TO THE RIGHT/LEFT
      IF(NFLX.EQ.4.OR.NFLX.GE.7)THEN
      IF(IJX.EQ.IJL)THEN
      UKKG=-UK(0,IJ)
      ELSE
      UKKG=UK(0,IJ)
      ENDIF
```

```
IF(UKKG.GT.0.)THEN
TL(0,IJX)=TL(0,IJ)
IF(NFLX.EQ.8)THEN
TH(0,IJX)=1.0
NTHS(IJX)=1
AKL(0,IJX)=8.67E5*(TL(0,IJX)/1400.0)**1.786
VISCL(0,IJX)=PVISC(0)
ELSE
TH(0,IJX)=TH(0,IJ)
NTHS(IJX)=NTHS(IJ)
AKL(0,IJX)=AKL(0,IJ)
VISCL(0,IJX)=VISCL(0,IJ)
ENDIF
IF(NFLX.EQ.4)P(IJX)=P(IJ)
ROG(IJX)=C9+C10*P(IJX)/(C12*TL(0,IJX)+C11*P(IJX))
IF(ISTW.NE.0)CALL DENSG(IJX)
RLK(0,IJX)=ROG(IJX)*TH(0,IJX)
UK(0,IJBX)=RB(I)*RLK(0,IJ)*UK(0,IJ)/(RBPM*RLK(0,IJX))
IF(NFLXY.GE.4)VK(0,IJBX)=VK(0,IJ)
ENDIF
IF(NFLX.NE.8)THEN
DO 10 K=1,NSOLID
IF(IJX.EQ.IJL)THEN
UKKL=-UK(K,IJ)
ELSE
UKKL=UK(K,IJ)
ENDIF
IF(UKKL.GT.0.0)THEN
TH(K,IJX)=TH(K,IJ)
RLK(K,IJX)=RLK(K,IJ)
TL(K,IJX)=TL(K,IJ)
IF(RLK(K,IJX).GT.0.0)THEN
```

```
      UK(K,IJBX)=RB(I)*RLK(K,IJ)*UK(K,IJ)/(RBPM*RLK(K,IJX))
      ELSE
      UK(K,IJBX)=0.0
      ENDIF
      AKL(K,IJX)=AKL(K,IJ)
      VISCL(K,IJX)=VISCL(K,IJ)
      VISBL(K,IJX)=VISBL(K,IJ)
      PS(K,IJX)=PS(K,IJ)
      GCON(K,IJX)=GCON(K,IJ)
      IF(NFLXY.GE.4)VK(K,IJBX)=VK(K,IJ)
      ENDIF
10    CONTINUE
      ENDIF
      ENDIF
C
C  CONTINUOUS OUTFLOW ON THE TOP/BOTTOM
      IF(NFLY.EQ.4.OR.NFLY.GE.7)THEN
      IF(IJY.EQ.IJB)THEN
      VKKG=-VK(0,IJ)
      ELSE
      VKKG=VK(0,IJ)
      ENDIF
      IF(VKKG.GT.0.0)THEN
      TL(0,IJY)=TL(0,IJ)
      IF(NFLY.EQ.8)THEN
      TH(0,IJY)=1.0
      NTHS(IJY)=0.0
      AKL(0,IJY)=8.67E5*(TL(0,IJY)/1400.0)**1.786
      VISCL(0,IJY)=PVISC(0)
      ELSE
      TH(0,IJY)=TH(0,IJ)
      NTHS(IJY)=NTHS(IJ)
```

```
AKL(0,IJY)=AKL(0,IJ)
VISCL(0,IJY)=VISCL(0,IJ)
ENDIF
IF(NFLY.EQ.4)P(IJY)=P(IJ)
ROG(IJY)=C9+C10*P(IJY)/(C12*TL(0,IJY)+C11*P(IJY))
IF(ISTW.NE.0)CALL DENSG(IJY)
RLK(0,IJY)=ROG(IJY)*TH(0,IJY)
VK(0,IJBY)=RLK(0,IJ)*VK(0,IJ)/RLK(0,IJY)
IF(NFLYX.GE.4)UK(0,IJBY)=UK(0,IJ)
ENDIF
IF(NFLY.NE.8)THEN
DO 30 K=1,NSOLID
IF(IJY.EQ.IJB)THEN
VKKL=-VK(K,IJ)
ELSE
VKKL=VK(K,IJ)
ENDIF
IF(VKKL.GT.0.0)THEN
TH(K,IJY)=TH(K,IJ)
RLK(K,IJY)=RLK(K,IJ)
TL(K,IJY)=TL(K,IJ)
IF(RLK(K,IJY).GT.0.0)THEN
VK(K,IJBY)=RLK(K,IJ)*VK(K,IJ)/RLK(K,IJY)
ELSE
VK(K,IJBY)=0.0
ENDIF
AKL(K,IJY)=AKL(K,IJ)
VISCL(K,IJY)=VISCL(K,IJ)
VISBL(K,IJY)=VISBL(K,IJ)
PS(K,IJY)=PS(K,IJ)
GCON(K,IJY)=GCON(K,IJ)
IF(NFLYX.GE.4)UK(K,IJBY)=UK(K,IJ)
```

```
      ENDIF
30    CONTINUE
      ENDIF
      ENDIF
C
C  SET BOUNDARY CONDITIONS IN RIGID CELLS-
C   GAS & PARTICLE VELOCITIES; GRANULAR TEMPERATURES
C
C  FREE SLIP WALL ON THE RIGHT/LEFT
      IF(NFLX .EQ. 2)THEN
      IF(NFLXY.EQ.2.OR.NFLXY.EQ.3)THEN
      DO 55 K=0,NSOLID
55    VK(K,IJBX)=VK(K,IJ)
      ENDIF
C
C  NO SLIP WALL ON THE RIGHT/LEFT
      ELSEIF(NFLX.EQ.3)THEN
      IF(NFLXY.EQ.2.OR.NFLXY.EQ.3)THEN
       DO 60 K=0,NCONT
60    VK(K,IJBX)=-VK(K,IJ)
      DO 65 K=NCONT+1,NSOLID
65    VK(K,IJBX)=VK(K,IJ)*(PLP(K)-DR(I))/(PLP(K)+DR(I))
      ENDIF
      ENDIF
C
C  FREE SLIP WALL ON THE TOP/BOTTOM
      IF(NFLY.EQ.2)THEN
      IF(NFLYX.EQ.2.OR.NFLYX.EQ.3)THEN
      DO 75 K=0,NSOLID
75    UK(K,IJBY)=UK(K,IJ)
      ENDIF
C
```

```
C  NO SLIP WALL ON THE TOP/BOTTOM
      ELSEIF(NFLY.EQ.3)THEN
      IF(NFLYX.EQ.2.OR.NFLYX.EQ.3)THEN
        DO 80 K=0,NCONT
80      UK(K,IJBY)=-UK(K,IJ)
        DO 85 K=NCONT+1,NSOLID
85      UK(K,IJBY)=UK(K,IJ)*(PLP(K)-DZ(J))/(PLP(K)+DZ(J))
      ENDIF
      ENDIF
      RETURN
      END
C  ----------------------------------------FLIC
      SUBROUTINE FLIC
      INCLUDE 'misb.com'
C
C  SETS CELL FLAG BASED UPON INPUT DATA
C
      DO 150 J=1,JB2
      DO 150 I=1,IB2
      IJ=I+(J-1)*IB2
C
C  SETS EACH CELL FLAG, IFL(IJ)=1, CELL FLAG WILL BE CHANGED
C  FOR OTHER TYPES
      IFL(IJ)=1
C
C  SETS CELL FLAG FOR LEFT COLUMN (I=1)
      IF(I.EQ.1)THEN
      IFL(IJ)=NSL(2)
C  SETS CELL FLAG FOR RIGHT COLUMN (I=IB2)
      ELSEIF(I.EQ.IB2)THEN
      IFL(IJ)=NSL(4)
      ENDIF
```

```
C  SETS CELL FLAG FOR TOP ROW (J=JB2)
      IF(J.EQ.JB2)THEN
      IFL(IJ)=NSL(3)
C  SETS CELL FLAG FOR BOTTOM ROW (J=1)
      ELSEIF(J.EQ.1)THEN
      IFL(IJ)=NSL(1)
      ENDIF
150   CONTINUE
C
C  SETS FLAGS FOR OBSTACLE CELLS AND OTHER FLUID CELLS
      DO 300 N=1,NTOT
      DO 300 I=IOB(1,N),IOB(2,N)
      DO 300 J=IOB(3,N),IOB(4,N)
      IJ=I+(J-1)*IB2
      IFL(IJ)=NSO(N)
300   CONTINUE
      IF(IFL(IB2JB2-1).EQ.4.AND.IFL(IB2JB1).EQ.4)IFL(IB2JB2)=4
      IF(IFL(IB2JB2-1).EQ.7.AND.IFL(IB2JB1).EQ.7)IFL(IB2JB2)=7
      RETURN
      END
C  ----------------------------------------INDX
      SUBROUTINE INDX
      INCLUDE 'misb.com'
      DO 30 J=1,JB2
      DO 30 I=1,IB2
      IJ=I+(J-1)*IB2
C
C  CALCULATE INDICES FOR ARRAY QUANTITIES
C
      IPJ=IJ+1
      IF(I.EQ.IB2)IPJ=IJ
      IJP=IJ+IB2
```

```
      IF(J.EQ.JB2)IJP=IJ
      IMJ=IJ-1
      IF(I.EQ.1)IMJ=IJ
      IJM=IJ-IB2
      IF(J.EQ.1)IJM=IJ
      IPJP=IPJ+IB2
      IF(J.EQ.JB2)IPJP=IPJ
      IMJP=IMJ+IB2
      IF(J.EQ.JB2)IMJP=IMJ
      IPJM=IPJ-IB2
      IF(J.EQ.1)IPJM=IPJ
      IMJM=IMJ-IB2
      IF(J.EQ.1)IMJM=IMJ
C
      INDC(IJ,1)=IPJ
      INDC(IJ,2)=IMJ
      INDC(IJ,3)=IJP
      INDC(IJ,4)=IJM
      INDC(IJ,5)=IPJP
      INDC(IJ,6)=IMJP
      INDC(IJ,7)=IPJM
      INDC(IJ,8)=IMJM
C
C  INITIALIZE 'INDC'
      IJR=IPJ
      IF(IFL(IPJ).EQ.2.OR.IFL(IPJ).EQ.3)IJR=IJ
      IJL=IMJ
      IF(IFL(IMJ).EQ.2.OR.IFL(IMJ).EQ.3)IJL=IJ
      IJT=IJP
      IF(IFL(IJP).EQ.2.OR.IFL(IJP).EQ.3)IJT=IJ
      IJB=IJM
      IF(IFL(IJM).EQ.2.OR.IFL(IJM).EQ.3)IJB=IJ
```

```
IJTR=IPJP
IF(IJ.EQ.(IB2JB1-1).AND.IFL(IPJP).EQ.4)IJTR=IJ
IF(IFL(IPJP).EQ.2.OR.IFL(IPJP).EQ.3)THEN
IJTR=IJT
IF(IJ.EQ.IJT)THEN
IJTR=IJR
ELSE
IF(IJ.NE.IJR)IJTR=IJ
ENDIF
ENDIF
IJTL=IMJP
IF(IFL(IMJP).EQ.2.OR.IFL(IMJP).EQ.3)THEN
IJTL=IJT
IF(IJ.EQ.IJT)THEN
IJTL=IJL
ELSE
IF(IJ.NE.IJL)IJTL=IJ
ENDIF
ENDIF
IJBR=IPJM
IF(IFL(IJBR).EQ.2.OR.IFL(IJBR).EQ.3)THEN
IJBR=IJB
IF(IJ.EQ.IJB)THEN
IJBR=IJR
ELSE
IF(IJ.NE.IJR)IJBR=IJ
ENDIF
ENDIF
IJRR=IJR+1
IF(I.GE.IB1)IJRR=IJR
IF(IFL(IJRR).EQ.2.OR.IFL(IJRR).EQ.3)IJRR=IJR
IJTT=IJT+IB2
```

```
      IF(J.GE.JB1)IJTT=IJT
      IF(IFL(IJTT).EQ.2.OR.IFL(IJTT).EQ.3)IJTT=IJT
C
      INDS(IJ,1)=IJR
      INDS(IJ,2)=IJL
      INDS(IJ,3)=IJT
      INDS(IJ,4)=IJB
      INDS(IJ,5)=IJTR
      INDS(IJ,6)=IJTL
      INDS(IJ,7)=IJBR
      INDS(IJ,8)=IJRR
      INDS(IJ,9)=IJTT
30    CONTINUE
      RETURN
      END
C ----------------------------------------ITER
      SUBROUTINE ITER
      INCLUDE 'misb.com'
C
C PERFORM THE ITERATIVE SOLUTION OF DIFFERENCE EQUATIONS OF
C MASS, MOMENTUM AND ENERGY EQUATIONS
C
      LOGICAL MUSTIT
      PARAMETER (NTMAX=400,LMAX=5,OMEGA=1.0)
      MUSTIT=.FALSE.
      DO 200 NIT=1,NTMAX
      NITER(ITX)=NITER(ITX)+1
      DO 100 J=2,JB1
      DO 100 I=2,IB1
      IJ=I+(J-1)*IB2
      IF(IFL(IJ).NE.1)GOTO 100
      LOOP=0
```

```
      KROS=-1
      KV=NTHS(IJ)
      IPJ=INDC(IJ,1)
      IMJ=INDC(IJ,2)
      IJP=INDC(IJ,3)
      IJM=INDC(IJ,4)
      IJR=INDS(IJ,1)
      IJL=INDS(IJ,2)
      IJT=INDS(IJ,3)
      IJB=INDS(IJ,4)
      DG=RLK(KV,IJ)-RLKN(KV,IJ)+DTORDR(I)*(RLFRK(KV,IJ)
     1-RLFRK(KV,IMJ))+DTODZ(J)*(RLFTK(KV,IJ)-RLFTK(KV,IJM))
      TARGET=(1.0-OMEGA)*DG
      ADG=ABS(DG)
      ADGTAR=ABS(DG-TARGET)
      DGORIG=ADG
      IF(ADG.LE.CONV(IJ))GOTO 78
      MUSTIT=.FALSE.
      D3=DG
      P3=P(IJ)
      IF(NIT.EQ.1)GOTO55
10    IF(D3.GT.TARGET)GOTO 11
      D2=D3
      P2=P3
      IF(KROS.EQ.-1)KROS=1
      IF(KROS.EQ.0)KROS=2
      GOTO12
11    D1=D3
      P1=P3
      IF(KROS.EQ.-1)KROS=0
      IF(KROS.EQ.1)KROS=2
12    IF(KROS.EQ.3)GOTO 54
```

```
      IF(KROS.EQ.2)GOTO 13
      D3TAR=D3-TARGET
      DP=-D3TAR*ABETA(IJ)
      DSN=SIGN(1.,D3TAR)
      IF(-DP*DSN.GT.0.25*P3)DP=-0.5*DSN*P3
      P(IJ)=P(IJ)+DP
      GOTO 54
13    P(IJ)=(D1*P2-D2*P1+TARGET*(P1-P2))/(D1-D2)
      ABETA(IJ)=(P1-P2)/(D1-D2)
      KROS=3
54    P3=P(IJ)
55    CONTINUE
      ROG(IJ)=C9+C10*P(IJ)/(C12*TL(0,IJ)+C11*P(IJ))
      IF(ISTW.NE.0)CALL DENSG(IJ)
      RLK(0,IJ)=TH(0,IJ)*ROG(IJ)
      CALL MATS
      CALL VELSK
      CALL MASFK(0,KV-1)
      CALL MASFK(KV+1,NSOLID)
78    THX=0.0
      DO 79 K=0,NSOLID
      IF(K.NE.KV)THEN
      RLK(K,IJ)=RLKN(K,IJ)-DTORDR(I)*(RLFRK(K,IJ)
     1-RLFRK(K,IMJ))-DTODZ(J)*(RLFTK(K,IJ)-RLFTK(K,IJM))
      IF(RLK(K,IJ).LT.1.E-6*RLKMIN(K))RLK(K,IJ)=0.0
      IF(K.EQ.0)THEN
      IF(RLK(0,IJ).GT.ROG(IJ))RLK(0,IJ)=ROG(IJ)
      TH(0,IJ)=RLK(0,IJ)/ROG(IJ)
      ELSE
      IF(RLK(K,IJ).GT.RL(K))RLK(K,IJ)=RL(K)
      TH(k,IJ)=RLK(K,IJ)/RL(K)
      IF(ISTW.NE.0)THEN
```

```
      CALL DENSL(IJ)
      IF(RLK(K,IJ).GT.ROK(IJ))RLK(K,IJ)=ROK(IJ)
      TH(K,IJ)=RLK(K,IJ)/ROK(IJ)
      ENDIF
      ENDIF
      THX=THX+TH(K,IJ)
      ENDIF
79    CONTINUE
      IF(THX.GT.0.8)THEN
      DIV=0.8/THX
      THX=0.8
      DO 81 K=1,NSOLID
      RLK(K,IJ)=RLK(K,IJ)*DIV
81    CONTINUE
      ENDIF
      TH(KV,IJ)=1.-THX
      IF(KV.EQ.0)THEN
      RLK(0,IJ)=TH(0,IJ)*ROG(IJ)
      ELSE
      RLK(KV,IJ)=TH(KV,IJ)*RL(KV)
      IF(ISTW.NE.0)RLK(KV,IJ)=TH(KV,IJ)*ROK(IJ)
      ENDIF
      IF(ADGTAR.LE.CONV(IJ))GOTO100
      CALL MASFK(KV,KV)
      DG=RLK(KV,IJ)-RLKN(KV,IJ)+DTORDR(I)*(RLFRK(KV,IJ)
     1-RLFRK(KV,IMJ))+DTODZ(J)*(RLFTK(KV,IJ)-RLFTK(KV,IJM))
      ADG=ABS(DG)
      ADGTAR=ABS(DG-TARGET)
      IF(ADGTAR.LE.CONV(IJ).AND.ADG.LT.DGORIG)GOTO100
      IF(NIT.EQ.1.AND.LOOP.EQ.0)THEN
      TARGET=(1.-OMEGA)*DG
      DGORIG=ADG
```

```
      ENDIF
      D3=DG
      LOOP=LOOP+1
      IF(LOOP.EQ.LMAX)THEN
      IF(KROS.LT.2)ABETA(IJ)=.5*LMAX*ABETA(IJ)
      GOTO 100
      ENDIF
      IF(KROS.EQ.3)CALL NEWP
      GOTO10
100   CONTINUE
      IF(MUSTIT)RETURN
      MUSTIT=.TRUE.
      IF(NIT.EQ.NTMAX)THEN
      WRITE(6,*)'MAX ITERATION AT TIME = ',TIME
      MAXIT=.TRUE.
      NITER(ITX)=9999
      ENDIF
200   CONTINUE
      RETURN
      END
C  -----------------------------------------KDRAGS
      SUBROUTINE KDRAGS(NPH1,NPH2)
      INCLUDE 'misb.com'
C
C  FLUID-PARTICLE FRICTION COEFFICIENT
C
      IF(TH(KV,IJ).LT.1.E-3)THEN
      DO 5 K=NPH1,NPH2
5     RKPG(K,IJ)=1.0E30
      ELSE
      DO 10 K=NPH1,NPH2
      IF(TH(K,IJ).GT.0.0)THEN
```

```
      DV=0.5*(VK(KV,IJ)-VK(K,IJ)+VK(KV,IJM)-VK(K,IJM))
      IF(I.EQ.1)THEN
      DU=UK(KV,IJ)-UK(K,IJ)
      ELSE
      DU=0.5*(RB(I)*(UK(KV,IJ)-UK(K,IJ))
     1+RB(I-1)*(UK(KV,IMJ)-UK(K,IMJ)))/R(I)
      ENDIF
      VREL(K)=(DU*DU+DV*DV)**0.5
      DENOM=DK(K)*PHI(K)*TH(KV,IJ)
      IF(TH(KV,IJ).GE.0.8)THEN
C  CALCULATE DRAG USING EMPIRICAL CORRELATION (EPfluid >= 0.8)
      REYN=RLK(KV,IJ)*VREL(K)*DENOM/VISCL(KV,IJ)
      IF(REYN.LT.1000.)THEN
      DRCOT=1.+.15*REYN**.687
      RKPG(K,IJ)=18.*DRCOT*TH(K,IJ)*VISCL(KV,IJ)
     1 /(TH(KV,IJ)**1.65*DENOM*DENOM)
      ELSE
      RKPG(K,IJ)=.75*DRCOE*TH(K,IJ)*VREL(K)*RLK(KV,IJ)
     1 /(DENOM*TH(KV,IJ)**1.65)
      ENDIF
      IF(RKPG(K,IJ).GT.1.0E30)RKPG(K,IJ)=1.0E30
      ELSE
C  CALCULATE DRAG USING ERGUN EQUATION (EPfluid < 0.8)
      RKPG(K,IJ)=(150.0*(1.0-TH(KV,IJ))*VISCL(KV,IJ)/DENOM
     1+1.75*RLK(KV,IJ)*VREL(K))*TH(K,IJ)/DENOM
      ENDIF
      IF(MODAB.NE.1)RKPG(K,IJ)=RKPG(K,IJ)/TH(KV,IJ)
C
      ELSE
      RKPG(K,IJ)=0.0
      ENDIF
10    CONTINUE
```

```
      ENDIF
      RETURN
      END
C  ----------------------------------------MASFK
      SUBROUTINE MASFK(NPH1,NPH2)
      INCLUDE 'misb.com'
C
C  CALCULATES MASS FLUXES FOR THE PHASES
C
      DO 10 K=NPH1,NPH2
      IF(UK(K,IMJ).GE.0.)THEN
      RLFRK(K,IMJ)=UK(K,IMJ)*RLK(K,IJL)*RB(I-1)
      ELSE
      RLFRK(K,IMJ)=UK(K,IMJ)*RLK(K,IJ)*RB(I-1)
      ENDIF
      IF(VK(K,IJM).GE.0.)THEN
      RLFTK(K,IJM)=VK(K,IJM)*RLK(K,IJB)
      ELSE
      RLFTK(K,IJM)=VK(K,IJM)*RLK(K,IJ)
      ENDIF
10    CONTINUE
C
      ENTRY MASFKA(NPH1,NPH2)
      DO 20 K=NPH1,NPH2
      IF(UK(K,IJ).GE.0.)THEN
      RLFRK(K,IJ)=UK(K,IJ)*RLK(K,IJ)*RB(I)
      ELSE
      RLFRK(K,IJ)=UK(K,IJ)*RLK(K,IJR)*RB(I)
      ENDIF
      IF(VK(K,IJ).GE.0.)THEN
      RLFTK(K,IJ)=VK(K,IJ)*RLK(K,IJ)
      ELSE
```

```
      RLFTK(K,IJ)=VK(K,IJ)*RLK(K,IJT)
      ENDIF
20    CONTINUE
      RETURN
      END
C ----------------------------------------MATS
      SUBROUTINE MATS
      INCLUDE 'misb.com'
      DIMENSION THKDPR(0:NS),THKDPZ(0:NS)
C
C  CALCULATES THE MATRIX COMPONENTS FOR VELOCITY
COMPONENTS
C
      DPR=DTODRP(I-1)*(P(IJ)-P(IJL))
      DPZ=DTODZP(J-1)*(P(IJ)-P(IJB))
      IF(MODAB.EQ.1)THEN
C  MODEL-A
      DO 1 K=0,NSOLID
      THKDPR(K)=(AR(I-1)*TH(K,IJL)+BR(I-1)*TH(K,IJ))*DPR
1     THKDPZ(K)=(AZ(J-1)*TH(K,IJB)+BZ(J-1)*TH(K,IJ))*DPZ
      ELSE
C  MODEL-B
      DO 2 K=0,NSOLID
      THKDPR(K)=0.0
2     THKDPZ(K)=0.0
      THKDPR(KV)=DPR
      THKDPZ(KV)=DPZ
      ENDIF
C
      DO 130 K=0,NSOLID
      KP=K+1
      BU1(KP)=RUK(K,IMJ)-THKDPR(K)
```

```
      BV1(KP)=RVK(K,IJM)-THKDPZ(K)
      DO 110 KK=1,K
      KS=K*KP/2+KK
      AU1(KP,KK)=AR(I-1)*APP(KS,IJL)+BR(I-1)*APP(KS,IJ)
      AU1(KK,KP)=AU1(KP,KK)
      AV1(KP,KK)=AZ(J-1)*APP(KS,IJB)+BZ(J-1)*APP(KS,IJ)
      AV1(KK,KP)=AV1(KP,KK)
110   CONTINUE
      KS=KP*(KP+1)/2
      AU1(KP,KP)=AR(I-1)*(APP(KS,IJL)+RLK(K,IJL))
     1+BR(I-1)*(APP(KS,IJ)+RLK(K,IJ))
      AV1(KP,KP)=AZ(J-1)*(APP(KS,IJB)+RLK(K,IJB))
     1+BZ(J-1)*(APP(KS,IJ)+RLK(K,IJ))
130   CONTINUE
C
      ENTRY MATSA
      DPR=DTODRP(I)*(P(IJR)-P(IJ))
      DPZ=DTODZP(J)*(P(IJT)-P(IJ))
      IF(MODAB.EQ.1)THEN
C  MODEL-A
      DO 3 K=0,NSOLID
      THKDPR(K)=(AR(I)*TH(K,IJ)+BR(I)*TH(K,IJR))*DPR
3     THKDPZ(K)=(AZ(J)*TH(K,IJ)+BZ(J)*TH(K,IJT))*DPZ
      ELSE
C  MODEL-B
      DO 4 K=0,NSOLID
      THKDPR(K)=0.0
4     THKDPZ(K)=0.0
      THKDPR(KV)=DPR
      THKDPZ(KV)=DPZ
      ENDIF
C
```

```
      DO 230 K=0,NSOLID
      KP=K+1
      BU(KP)=RUK(K,IJ)-THKDPR(K)
      BV(KP)=RVK(K,IJ)-THKDPZ(K)
      DO 210 KK=1,K
      KS=K*KP/2+KK
      AU(KP,KK)=AR(I)*APP(KS,IJ)+BR(I)*APP(KS,IJR)
      AU(KK,KP)=AU(KP,KK)
      AV(KP,KK)=AZ(J)*APP(KS,IJ)+BZ(J)*APP(KS,IJT)
      AV(KK,KP)=AV(KP,KK)
210   CONTINUE
      KS=KP*(KP+1)/2
      AU(KP,KP)=AR(I)*(APP(KS,IJ)+RLK(K,IJ))
     1+BR(I)*(APP(KS,IJR)+RLK(K,IJR))
      AV(KP,KP)=AZ(J)*(APP(KS,IJ)+RLK(K,IJ))
     1+BZ(J)*(APP(KS,IJT)+RLK(K,IJT))
230   CONTINUE
C
      RETURN
      END
C ----------------------------------------MULTI
      SUBROUTINE MULTI
      INCLUDE 'misb.com'
C
C  CALCULATE INTERPHASE MOMENTUM EXCHANGE COEFFICIENT
C
      DO 100 K=1,NSOLID
      DO 90 KK=0,K-1
      KS=K*(K+1)/2+KK+1
      IF(K.EQ.KV)THEN
      APP(KS,IJ)=-RKPG(KK,IJ)*DT
      ELSEIF(KK.EQ.KV)THEN
```

```
      APP(KS,IJ)=-RKPG(K,IJ)*DT
      ELSE
C  CALCULATE PARTICLE TO PARTICLE INTERACTION
      EPSUM=TH(K,IJ)+TH(KK,IJ)
      IF(EPSUM.NE.0.0)THEN
      IF(DK(K).GT.DK(KK))THEN
      K1=K
      K2=KK
      ELSE
      K1=KK
      K2=K
      ENDIF
      XBAR=TH(K1,IJ)/EPSUM
      IF(XBAR.LE.PHILIM(K,KK))THEN
      EPKL=EPSL(K,KK)*XBAR+CPHI(K2)
      ELSE
      EPKL=EPSU(K,KK)*(1.0-XBAR)+CPHI(K1)
      ENDIF
      CEPR=(EPSUM/EPKL)**(1./3.)
      IF(CEPR.GE.1.0)THEN
      CON=4.E10*RLK(K,IJ)*RLK(KK,IJ)*DT*DKF(K,KK)
      ELSE
      CON=DT*RLK(K,IJ)*RLK(KK,IJ)*DKF(K,KK)
     1*(CEPR+3.)/(1.-CEPR)
      ENDIF
      IF(MODAB.NE.0)CON=CON/TH(KV,IJ)
      DV=(VK(KK,IJ)-VK(K,IJ)+VK(KK,IJM)-VK(K,IJM))*0.5
      IF(I.EQ.1)THEN
      DU=UK(KK,IJ)-UK(K,IJ)
      ELSE
      DU=0.5*(RB(I)*(UK(KK,IJ)-UK(K,IJ))
     1+RB(I-1)*(UK(KK,IMJ)-UK(K,IMJ)))/R(I)
```

```
      ENDIF
      VRELP=(DU*DU+DV*DV)**0.5
      APP(KS,IJ)=-CON*VRELP
      ELSE
      APP(KS,IJ)=0.0
      ENDIF
      ENDIF
90    CONTINUE
100   CONTINUE
C
      DO 200 K=0,NSOLID
      SUM=0.0
      DO 110 KK=0,K-1
      KS=K*(K+1)/2+KK+1
110   SUM=SUM+APP(KS,IJ)
      DO 120 KK=K+1,NSOLID
      KS=KK*(KK+1)/2+K+1
120   SUM=SUM+APP(KS,IJ)
      KS=(K+1)*(K+2)/2
200   APP(KS,IJ)=-SUM
      RETURN
      END
C ----------------------------------------NEWP
      SUBROUTINE NEWP
      INCLUDE 'misb.com'
C
C CALCULATE NEW ESTIMATES OF ADVANCED TIME PRESSURE
C FROM THREE (P, D) POINTS
C
      IF(D1.NE.D3)THEN
      PA=(D1*P3-D3*P1+TARGET*(P1-P3))/(D1-D3)
      ELSE
```

```
      PA=0.5*(P2+P3)
      ENDIF
      IF(D2.NE.D3)THEN
      PB=(D2*P3-D3*P2+TARGET*(P2-P3))/(D2-D3)
      ELSE
      PB=0.5*(P1+P3)
      ENDIF
      IF((D1-TARGET)*(D3-TARGET).GT.0.)THEN
      IF(PA.LT.P2.OR.PA.GT.P3)PA=0.5*(P2+P3)
      ELSE
      IF(PB.LT.P3.OR.PB.GT.P1)PB=0.5*(P1+P3)
      ENDIF
      P(IJ)=0.5*(PA+PB)
      RETURN
      END
C ----------------------------------------PROG
      SUBROUTINE PROG
      INCLUDE 'misb.com'
C
C  CONTROL THE PROGRAM FLOW AND OUTPUT
C
      TDUMP1=TIME
      TPRI=TIME
      TPRII=TIME
      WRITE(16,600)
1     CONTINUE
      TPDT=TIME+0.1*DT
      IF(MAXIT)CALL VARDT
C
C  SET BOUNDARY AND OBSTACLE CELLS
      CALL BDRY
C
```

```
C  SAVE VALUES AT PREVIOUS TIME STEP
      DO 10 J=2,JB2
      DO 10 I=2,IB2
      IJ=I+(J-1)*IB2
      IF(IFL(IJ).NE.2.OR.IFL(IJ).NE.3)THEN
      IMJ=INDC(IJ,2)
      IJM=INDC(IJ,4)
      RLX(IJ)=RLK(0,IJ)
      PN(IJ)=P(IJ)
      VISFN(IJ)=VISCL(0,IJ)
      DO 5 K =1,NSOLID
5      RLX(IJ)=RLX(IJ)+RLK(K,IJ)
      DO 6 K=0,NSOLID
6      RLKN(K,IJ)=RLK(K,IJ)
      IF(IFL(IJ).EQ.4.OR.IFL(IJ).GT.7)THEN
      KV=NTHS(IJ)
      CALL KDRAGS(0,NSOLID)
      CALL MULTI
      ENDIF
      ENDIF
10    CONTINUE
C
      IF(TPDT.GE.TPRII)THEN
      TPRII=TPRII+TPRR
      IF(NIMBR.NE.1)THEN
C     calculate the mass withdraw and quality ratio for outlet/inlet
      W1=0.0
      W2=0.0
      W3=0.0
      THG1=0.0
      THG2=0.0
      THG3=0.0
```

```
      IF(NIMBR.EQ.0)THEN
C     for branching tee
      J=1
      DO 31 I=1,IB2
      IJ=I+(J-1)*IB2
      IF(IFL(IJ).EQ.5)THEN
      DO 32 K=0,1
32    W1=W1+RLK(K,IJ)*VK(K,IJ)*DR(I)
      THG1=THG1+RLK(0,IJ)*VK(0,IJ)*DR(I)
      ENDIF
31    CONTINUE
      ELSE
C     for impacting
      I=1
      DO 30 J=1,JB2
      IJ=I+(J-1)*IB2
      IF(IFL(IJ).EQ.5)THEN
      DO 11 K=0,1
11    W1=W1+RLK(K,IJ)*UK(K,IJ)*DZ(J)          !w1---mass flux at inlet
      THG1=THG1+RLK(0,IJ)*UK(0,IJ)*DZ(J)
      ENDIF
30    CONTINUE
      ENDIF
      X1=THG1/W1                              !x1---quality at inlet
      J=LCAT1
      DO 12 I=1,IB2
      IJ=I+(J-1)*IB2
      IF(IFL(IJ).EQ.3)GOTO 12
      DO 13 K=0,1
13    W2=W2+RLK(K,IJ)*VK(K,IJ)*DR(I)          !W2---MASS FLUX AT
OUTLET II
      THG2=THG2+RLK(0,IJ)*VK(0,IJ)*DR(I)
```

```
12    CONTINUE
      IF(W2.EQ.0.0)THEN
      X2=0.0
      ELSE
      X2=THG2/W2                    !X2---QUALITY AT OUTLET II
      ENDIF
      IF(NIMBR.EQ.0)THEN
C     for branching tee at branch outletIII
      I=LCAT2
      DO 33 J=1,JB2
      IJ=I+(J-1)*IB2
      IF(IFL(IJ).EQ.3)GOTO 33
      DO 34 K=0,1
34    W3=W3+RLK(K,IJ)*UK(K,IJ)*DZ(J)
      THG3=THG3+RLK(0,IJ)*UK(0,IJ)*DZ(J)
33    CONTINUE
      ELSE
C     for impacting tee
      J=LCAT2
      DO 14 I=1,IB2
      IJ=I+(J-1)*IB2
      IF(IFL(IJ).EQ.3)GOTO 14
      DO 25 K=0,1
25    W3=W3+RLK(K,IJ)*VK(K,IJ)*DR(I)                  !W3---MASS FLUX AT
OUTLET III
      THG3=THG3+RLK(0,IJ)*VK(0,IJ)*DR(I)
14    CONTINUE
      ENDIF
      IF(W3.EQ.0.0)THEN
      X3=0.0                              !X3---QUALITY AT OUTLET III
      ELSE
      X3=THG3/W3
```

```
      ENDIF
      W31=W3/W1                  !w31---mass withdraw ratio for outletIII/inlet
      W21=W2/W1                  !w21---mass withdraw ratio at outletII
      X31=X3/X1                 !x31---quality ratio for outletIII/inlet
      X21=X2/X1                 !x21---quality ratio for outletII/inlet
      WRITE(16,610)TIME,W21,W31,X21,X31
      ENDIF
      ENDIF
      IF(TPDT.GE.TPRI)THEN
      WRITE(6,547)TIME
      WRITE(12,547)TIME
      WRITE(13,547)TIME
      TPRI=TPRI+TPR
      IF(TIME.GT.-0.1*DT)THEN
      IF(NIMBR.NE.1)THEN
      WRITE(6,400)W1,W2,W3
      WRITE(6,410)W21,W31
      WRITE(6,420)X21,X31
      WRITE(6,430)X1,X2,X3
      ENDIF
      WRITE(6,548)
      DO 325 IJ=IB2JB2,IB2,-IB2
325   WRITE(6,550)(P(IL),IL=IJ-IB1,IJ)
      DO 450 IJ=IB2JB2,IB2,-IB2
450   WRITE(12,550)(TH(0,IL),IL=IJ-IB1,IJ)
      DO 440 IJ=IB2JB2,IB2,-IB2
440   WRITE(13,550)(TH(1,IL),IL=IJ-IB1,IJ)
      DO 337 k=0,NSOLID
      WRITE(6,556)K,K
      DO 336 IJ=IB2JB2,IB2,-IB2
336   WRITE(6,550)(VK(K,IL),IL=IJ-IB1,IJ)
      WRITE(6,557)K,K
```

```
      DO 337 IJ=IB2JB2,IB2,-IB2
337   WRITE(6,550)(UK(K,IL),IL=IJ-IB1,IJ)
      ENDIF
      ENDIF
C
C  WRITE DATA ON DISK FOR RESTART
      IF(TPDT.GT.TSTOP.OR.TPDT.GT.TDUMP1)THEN
      ITINT=ITINT+1
      CALL TAPEWR
      IF(ITX.NE.1)THEN
      ITXN=ITX-1
      IF(ITINT.GE.IMX(ITX).OR.NITER(ITXN).LT.
     1(NITER(ITX)-4))THEN
      CALL VARDTI
      GOTO 100
      ENDIF
      ENDIF
      IF(ITX.NE.ITXMX)THEN
      ITXN=ITX+1
      IF(NITER(ITXN).LT.(NITER(ITX)-2))CALL VARDTI
      ENDIF
100   NITER(ITX)=0
      REWIND(9)
      TDUMP1=TDUMP1+TDUMP
      ENDIF
C
      IF(TPDT.LT.TSTOP)THEN
      CALL TILDE
      CALL BETAS
      CALL ITER
      IF(.NOT.(MAXIT).OR.ITXMX.EQ.1)THEN
C  CALCULATE NEW PHYSICAL PROPERTIES IN FLUID CELLS
```

```
      DO 20 J=2,JB2
      DO 20 I=2,IB2
      IJ=I+(J-1)*IB2
      IF(IFL(IJ).EQ.1)THEN
      ROG(IJ)=C9+C10*P(IJ)/(C12*TL(0,IJ)+C11*P(IJ))
      IF(ISTW.NE.0)CALL DENSG(IJ)
      RLK(0,IJ)=TH(0,IJ)*ROG(IJ)
      DO 15 K=0,NSOLID
15    VISCL(K,IJ)=PVISC(K)*TH(K,IJ)
      ENDIF
20    CONTINUE
      ENDIF
      TIME=TIME+DT
      GOTO 1
      ENDIF
      RETURN
C
547   FORMAT(1X,//,1X,'@ TIME = ',1PE12.5,' secs')
548   FORMAT(1X,/,1X,'FLUID PRESSURE, P (dynes/cm^2)'/)
549   FORMAT(1X,/,1X,'VOLUME FRACTION (PHASE- ',I1
     1,'), TH',I1/)
550   FORMAT(1X,30(1X,G10.4))
555   FORMAT(1X,/,1X,'TEMPERATURE (PHASE-'
     1,I1,'), TL',I1,' (Kelvin)'/)
556   FORMAT(1X,/,1X,'VELOCITY - Z (or Y) component,',
     1 ' (PHASE-',I1,'), VK',I1,' (cm/s)'/)
557   FORMAT(1X,/,1X,'VELOCITY - R (or X) component,',
     1 ' (PHASE-',I1,'), UK',I1,' (cm/s)'/)
400   FORMAT(1X,/,1X,'INLET FLUX=',G11.4,3X,'RUN FLUX=',G11.4,3X,
     1 'BRANCH FLUX=',G11.4)
410   FORMAT(1X,/,1X,'RUN/INLET MASS RATIO=',G11.4,3X,
     1 'BRANCH/INLET MASS RATIO=',G11.4)
```

```
420  FORMAT(1X,/,1X,'RUN/INLET QUALITY RATIO=',G11.4,3X,
   1 'BRANCH/INLET QUALITY RATIO=',G11.4)
430  FORMAT(1X,/,1X,'INLET QUALITY X1=',G11.4,2X,
   1  'RUN QUALITY X2=',G11.4,2X,'BRANCH QUALITY X3=',G11.4)
600  FORMAT(1X,/,7X,'TIME',8X,'W21',9X,'W31',9X,'X21',9X,
   1   'X31')
610  FORMAT(1X,5(1X,G11.4))
     END
C  ----------------------------------------QESOL
     SUBROUTINE QESOL(AP0,AP1,AP2,XSOL)
C
C  SOLVE QUADRATIC EQUATION
C
C  SCALING OF COEFFICIENTS
     AAP0=ABS(AP0)
     AAP1=ABS(AP1)
     AAP2=ABS(AP2)
     APMAX=AAP2
     IF(APMAX.LT.AAP1)APMAX=AAP1
     IF(APMAX.LT.AAP0)APMAX=AAP0
     AP0=AP0/APMAX
     AP1=AP1/APMAX
     AP2=AP2/APMAX
C
     ENTRY QESOL1(AP0,AP1,AP2,XSOL)
     IF(AP0.EQ.0.0)THEN
     IF(AP2.EQ.0.0)THEN
     XSOL=0.0
     ELSE
     XSOL=-AP1/AP2
     IF(XSOL.LT.0.0)XSOL=0.0
     ENDIF
```

```
      ELSE
      IF(AP1.EQ.0.0)THEN
      IF(AP2.EQ.0.0)THEN
      XSOL=0.0
      ELSE
      DISC=-AP0/AP2
      IF(DISC.LE.0.0)THEN
      XSOL=0.0
      ELSE
      XSOL=DISC**0.5
      ENDIF
      ENDIF
      ELSE
      IF(AP2.EQ.0.0)THEN
      XSOL=-AP0/AP1
      IF(XSOL.LT.0.0)XSOL=0.0
      ELSE
      GOTO 10
      ENDIF
      ENDIF
      ENDIF
      RETURN
C
      ENTRY QESOL2(AP0,AP1,AP2,XSOL)
10    CONTINUE
      SAP1=AP1*AP1
      DISC=SAP1-4.0*AP0*AP2
      IF(DISC.LT.-1.E-4*SAP1)THEN
      XSOL=0.0
      ELSE
      IF(DISC.LT.0.0)DISC=0.0
      IF(AP1.LT.0.0)THEN
```

```
      XSOL=(-AP1+DISC**0.5)/(2.0*AP2)
      ELSE
      XSOL=-2.0*AP0/(AP1+DISC**0.5)
      ENDIF
      ENDIF
      RETURN
      END
C  ----------------------------------------SETC
      SUBROUTINE SETC
      INCLUDE 'misb.com'
C
C  INITIALIZES CONSTANTS AND FUNCTIONS
C
      SQTPI=PI**0.5
      RSQTP=1./SQTPI
C  INITIALIZE STATIC SOLIDS PRESSURE AND COHESIVE FORCE
      DO 15 I=0,1000
      THX=I/1000.
      IF(ICOH.EQ.0)THEN
      COHF(I)=0.0
      ELSE
      COHF(I)=10.**(-C13*THX+C14)
      ENDIF
C  SOLIDS STRESS (ELASTIC MODULUS G) REGIME
      GTH(I)=10.**(-C15*THX+C16)
15    CONTINUE
C
C  CALCULATE VOLUME FRACTION FOR (RLIM particles/cm^3)
      DO 20 K=0,NSOLID
      VOLP=PI*DK(K)*DK(K)*DK(K)/6.0
      RLKMIN(K)=RLIM*RL(K)*VOLP
20    CONTINUE
```

```
C
      DO 50 K=NCONT,NSOLID
      VISDIL(K)=5.0*SQTPI*RL(K)*DK(K)/96.0
C     GCDIL(K)=(15.0/4.0)*VISDIL(K)
50    CONTINUE
C
      MAXIT=.FALSE.
      ITINT=0
      DO 30 IX=1,ITXMX
      IMX(IX)=105-5*IX
      NITER(IX)=0
30    CONTINUE
C
      DRCOE=0.44
      DO 55 K=1,NSOLID
      DO 55 KK=0,K-1
      IF(DK(K).GT.DK(KK))THEN
      K1=K
      K2=KK
      ELSE
      K1=KK
      K2=K
      ENDIF
      DRATX=(DK(K2)/DK(K1))**0.5
      PHILIM(K,KK)=CPHI(K1)/(CPHI(K1)+(1.0-CPHI(K1))*CPHI(K2))
      EPSL(K,KK)=(CPHI(K1)-CPHI(K2)+(1.0-DRATX)*(1.0-CPHI(K1))
     1*CPHI(K2))*(CPHI(K1)+(1.0-CPHI(K2))*CPHI(K1))/CPHI(K1)
      EPSU(K,KK)=(1.0-DRATX)*(CPHI(K1)
     1+(1.0-CPHI(K1))*CPHI(K2))
55    CONTINUE
      RETURN
      END
```

```
C ----------------------------------------SETPRE
      SUBROUTINE SETPRE
      INCLUDE 'misb.com'
      DIMENSION RLSUM(NO)
C
C SET INITIAL PRESSURE PROFILE
C
      DO 10 N=1,NFL
      RLSUM(N)=0.0
      IF(IPRES.EQ.1)THEN
      DO 5 K=1,NSOLID
5      RLSUM(N)=RLSUM(N)+THIO(K,N)*RL(K)
      ENDIF
10    CONTINUE
      POM=PIO(1)
      RGSUM=(C9+C10*PIO(1)/(C12*TEMIO(0,1)
     1+C11*PIO(1)))*THIO(0,1)
      RLSUMO=RLSUM(1)
C
      DO 30 N=1,NFL
      RLSUMN=RLSUM(N)
      CS1=0.5*THIO(0,N)*(-GRAVY)
      DO 20 J=IOB(4,N),IOB(3,N),-1
      CS2=POM+0.5*(DZ(J)*(RGSUM+RLSUMO)+DZ(J+1)
     1 *(C9*THIO(0,N)+RLSUMN))*(-GRAVY)
      AP0=-CS2*C12*TEMIO(0,N)
      AP1=C12*TEMIO(0,N)-CS2*C11-CS1*DZ(J+1)*C10
      AP2=C11
      IF(C10.EQ.0.0)THEN
      POM=CS2
      ELSEIF(C11.EQ.0.0)THEN
      POM=-AP0/AP1
```

```
      ELSE
      CALL QESOL2(AP0,AP1,AP2,POM)
      ENDIF
      ROGT=(C9+C10*POM/(C12*TEMIO(0,N)+C11*POM))
      IF(ISTW.NE.0)THEN
      PPC=220.55E+06
      VVC=3.10559
      AAO=0.85188
      AA1=-0.26786667
      AA2=-0.10696316
      AA3=-0.015826285
      AA4=-9.0169534E-4
      CC1=ALOG(POM/PPC)
      CC=EXP(AAO+AA1*CC1+AA2*CC1**2.+AA3*CC1**3.+
     1AA4*CC1**4.)
      RC=1./CC
      ROGT=POM*RC/PPC/VVC
      ENDIF
      RGSUM=ROGT*THIO(0,N)
      RLSUMO=RLSUMN
      DO 20 I=1,IB2
      IJ=I+(J-1)*IB2
      IF(IFL(IJ).EQ.1.OR.IFL(IJ).EQ.6)THEN
      P(IJ)=POM
      ROG(IJ)=ROGT
      RLK(0,IJ)=RGSUM
      ENDIF
20    CONTINUE
30    CONTINUE
      RETURN
      END
C ----------------------------------------SETRZ
```

```
      SUBROUTINE SETRZ
      INCLUDE 'misb.com'
C
C  INITIALIZE THE RADII AND ALL THE FIELD VARIABLES;
C  DEFINE THE VARIABLE COMPUTING MESH R, Z, DR AND DZ.
C
      IF(ITD.NE.1)THEN
      DO 1 I=1,IB1
      DRP(I)=0.5*(DR(I)+DR(I+1))
      RDR(I)=1.0/DR(I)
      RDRP(I)=1.0/DRP(I)
      AR(I)=0.5*DR(I+1)*RDRP(I)
1     BR(I)=1.0-AR(I)
      DRP(IB2)=DR(IB2)
      RDR(IB2)=1.0/DR(IB2)
      RDRP(IB2)=RDR(IB2)
      AR(IB2)=0.5
      BR(IB2)=0.5
      DO 2 J=1,JB1
      DZP(J)=0.5*(DZ(J)+DZ(J+1))
      RDZ(J)=1.0/DZ(J)
      RDZP(J)=1.0/DZP(J)
      AZ(J)=0.5*DZ(J+1)*RDZP(J)
2     BZ(J)=1.0-AZ(J)
      DZP(JB2)=DZ(JB2)
      RDZ(JB2)=1.0/DZ(JB2)
      RDZP(JB2)=RDZ(JB2)
      AZ(JB2)=0.5
      BZ(JB2)=0.5
      IF(ITC.EQ.1)THEN
      RTC=RST-0.5*DR(1)
      RTB=RST
```

```
      R(1)=RTC**ITC
      RB(1)=RTB**ITC
C     IF(RTC.LE.0.0.AND.ITC.EQ.2)R(1)=-R(1)
      IF(RB(1).LT.1.E-8)THEN
      RRB(1)=0.0
      ELSE
      RRB(1)=1.0/RB(1)
      ENDIF
      DO 3 I=2,IB2
      RTC=RTB+0.5*DR(I)
      RTB=RTB+DR(I)
      R(I)=RTC**ITC
      RB(I)=RTB**ITC
      RRB(I)=1./RB(I)
3     CONTINUE
C
      ELSE
      DO 8 I=1,IB2
      R(I)=1.
      RB(I)=1.
      RRB(I)=1.
8     CONTINUE
      ENDIF
      DO 11 I=1,IB2
      RRIDR(I)=RDR(I)/R(I)
      RRIDRP(I)=RRB(I)*RDRP(I)
11    CONTINUE
      ENDIF
      DO 15 I=1,IB2
      DTODR(I)=DT*RDR(I)
      DTODRP(I)=DT*RDRP(I)
      DTORDR(I)=DT*RRIDR(I)
```

```
      DTOBDR(I)=DT*RRIDRP(I)
15    CONTINUE
      DO 16 J=1,JB2
      DTODZ(J)=DT*RDZ(J)
16    DTODZP(J)=DT*RDZP(J)
      RETURN
      END
C ----------------------------------------SETUP
      SUBROUTINE SETUP
      INCLUDE 'misb.com'
C
C  DEFINES THE COMPUTING MESH FLUID VARIABLE
C  INITIAL CONDITIONS FROM INPUT DATA
C
      CALL SETC
      CALL SETRZ
      IF(ITD.NE.1)THEN
      WRITE(6,660)
      DO 10 IJ2=IB2JB2,IB2,-IB2
      IJ1=IJ2-IB1
10    WRITE(6,650)(IFL(IKPR),IKPR=IJ1,IJ2)
      WRITE(6,660)
C
C  CALCULATE INDICES
      CALL INDX
C
      IF(ITD.NE.2)THEN
      DO 60 N=1,NCAL
      UGY=UIO(0,N)/THIO(0,N)
      DO 60 J=IOB(3,N),IOB(4,N)
      DO 60 I=IOB(1,N),IOB(2,N)
      IJ=I+(J-1)*IB2
```

```
      IF(IFL(IJ).NE.2.AND.IFL(IJ).NE.3)THEN
      IPJ=INDC(IJ,1)
      P(IJ)=PIO(N)
      TL(0,IJ)=TEMIO(0,N)
      TH(0,IJ)=THIO(0,N)
      IF(IFL(IJ).EQ.4.OR.IFL(IJ).EQ.5.OR.IFL(IJ).GE.7)THEN
      ROG(IJ)=C9+C10*P(IJ)/(C12*TL(0,IJ)+C11*P(IJ))
      IF(ISTW.NE.0)CALL DENSG(IJ)
      RLK(0,IJ)=ROG(IJ)*TH(0,IJ)
      ENDIF
      UK(0,IJ)=UIO(0,N)
      IF(IFLZB.EQ.1.AND.IFL(IJ).NE.5.AND.
     1 IFL(IPJ).NE.2.AND.IFL(IPJ).NE.3)UK(0,IJ)=UGY
      VK(0,IJ)=VIO(0,N)
      DO 56 K=1,NSOLID
      TL(K,IJ)=TEMIO(K,N)
      UK(K,IJ)=UIO(K,N)
      VK(K,IJ)=VIO(K,N)
      TH(K,IJ)=THIO(K,N)
      IF(ISTW.NE.0)THEN
      CALL DENSL(IJ)
      RLK(K,IJ)=ROK(IJ)*THIO(K,N)
      ELSE
      RLK(K,IJ)=RL(K)*THIO(K,N)
      ENDIF
56    CONTINUE
      ENDIF
60    CONTINUE
C
      CALL SETPRE
      ENDIF
      ENDIF
```

```
C
C  INITIALIZE PHYSICAL PROPERTIES FLUID, INFLOW
C  AND OUTFLOW CELLS
      DO 70 J=1,JB2
      DO 70 I=1,IB2
      IJ=I+(J-1)*IB2
      IF(IFL(IJ).NE.2.AND.IFL(IJ).NE.3)THEN
      IMJ=INDC(IJ,2)
      IJP=INDC(IJ,3)
      IJM=INDC(IJ,4)
      IJR=INDS(IJ,1)
      IJT=INDS(IJ,3)
C
      IF(ITD.EQ.0)THEN
      IF(IFLZB.EQ.1.AND.IFL(IJ).NE.5.AND.IFL(IJP).NE.2
     1.AND.IFL(IJP).NE.3)VK(0,IJ)=ROG(IJ)*VK(0,IJ)
     1/(AZ(J)*RLK(0,IJ)+BZ(J)*RLK(0,IJT))
      ELSE
      ROG(IJ)=C9+C10*P(IJ)/(C12*TL(0,IJ)+C11*P(IJ))
      IF(ISTW.NE.0)CALL DENSG(IJ)
      RLK(0,IJ)=TH(0,IJ)*ROG(IJ)
      DO 65 K=1,NSOLID
      IF(ISTW.NE.0)THEN
      CALL DENSL(IJ)
      RLK(K,IJ)=TH(K,IJ)*ROK(IJ)
      ELSE
      RLK(K,IJ)=TH(K,IJ)*RL(K)
      ENDIF
65    CONTINUE
      ENDIF
      CALL MASFKA(0,NSOLID)
      DO 66 K=0,NSOLID
```

```
      VISCL(K,IJ)=PVISC(K)*TH(K,IJ)
66    VISBL(K,IJ)=0.0
      ENDIF
C
C  FIND CONTINUOUS PHASE
      KV=0
      IF(NCONT.EQ.2)THEN
      CALL KDRAGS(1,1)
      IF(ISWIT.EQ.1)THEN
      KV=1
      CALL KDRAGS(0,0)
      IF(RKPG(0,IJ).GT.RKPG(1,IJ))KV=0
      RKPG(KV,IJ)=0.0
      ENDIF
      ENDIF
      NTHS(IJ)=KV
      CALL KDRAGS(NCONT,NSOLID)
      CALL MULTI
70    CONTINUE
650   FORMAT(1X,78I1)
660   FORMAT(//)
      RETURN
      END
C  ----------------------------------------TAPERD
      SUBROUTINE TAPERD
      INCLUDE 'misb.com'
C
C  READ INPUT DATA FROM TAPE
C
      READ(9)TIME
      IF(ITD.EQ.2)THEN
      READ(9)DT,ITX,TDUMP1,TPRI,TPRII,TPDT,
```

```
     1 (NITER(IX),IX=1,ITXMX)
      ELSE
      READ(9)
      ENDIF
      READ(9)(P(IJ),IJ=1,IB2JB2)
      READ(9)((TH(K,IJ),UK(K,IJ),VK(K,IJ),TL(K,IJ),
     1 K=0,NSOLID),IJ=1,IB2JB2)
      RETURN
      END
C ----------------------------------------TAPEWR
      SUBROUTINE TAPEWR
      INCLUDE 'misb.com'
C
C WRITE INPUT DATA TO TAPE
C
      WRITE(9)TIME,IB2,JB2,NSOLID
      WRITE(9)DT,ITX,TDUMP1,TPRI,TPRII,TPDT,
     1 (NITER(IX),IX=1,ITXMX)
      WRITE(9)(P(IJ),IJ=1,IB2JB2)
      WRITE(9)((TH(K,IJ),UK(K,IJ),VK(K,IJ),TL(K,IJ),
     1 K=0,NSOLID),IJ=1,IB2JB2)
      RETURN
      END
C ----------------------------------------TILDE
      SUBROUTINE TILDE
      INCLUDE 'misb.com'
      DIMENSION RKR(0:NS),RKZ(0:NS),RKRAB(0:NS),RKZAB(0:NS)
      DIMENSION RLGR(0:NS),RLGZ(0:NS)
C
C CALCULATE MOMENTA DUE TO CONVECTION, GRAVITY,
C VISCOUS STRESS, SOLIDS PRESSURE AND COHESIVE STRESS
C
```

```
      DO 100 J=2,JB1
      DO 100 I=2,IB1
      IJ=I+(J-1)*IB2
      IF(IFL(IJ).NE.1)GOTO 100
      IPJ=INDC(IJ,1)
      IMJ=INDC(IJ,2)
      IJP=INDC(IJ,3)
      IJM=INDC(IJ,4)
      IMJP=INDC(IJ,6)
      IPJM=INDC(IJ,7)
      IJR=INDS(IJ,1)
      IJL=INDS(IJ,2)
      IJT=INDS(IJ,3)
      IJB=INDS(IJ,4)
      IJTR=INDS(IJ,5)
      IJTL=INDS(IJ,6)
      IJBR=INDS(IJ,7)
      IJRR=INDS(IJ,8)
      IJTT=INDS(IJ,9)
C  FIND CONTINUOUS PHASE
      KV=0
      KVN=1
      THAVRN=0.0
      THAVZN=0.0
      IF(NCONT.EQ.2)THEN
      CALL KDRAGS(1,1)
      IF(ISWIT.EQ.1)THEN
      KV=1
      CALL KDRAGS(0,0)
      IF(RKPG(0,IJ).GT.RKPG(1,IJ))KV=0
      RKPG(KV,IJ)=0.0
      ENDIF
```

```
      KVN=1-KV
      THAVRN=AR(I)*TH(KVN,IJ)+BR(I)*TH(KVN,IJR)
      THAVZN=AZ(J)*TH(KVN,IJ)+BZ(J)*TH(KVN,IJT)
      ENDIF
      NTHS(IJ)=KV
C
      RLGR(0)=AR(I)*ROG(IJ)+BR(I)*ROG(IJR)
      RLGZ(0)=AZ(J)*ROG(IJ)+BZ(J)*ROG(IJT)
      DO 5 K=1,NSOLID
      RLGR(K)=RL(K)
      RLGZ(K)=RL(K)
      IF(ISTW.NE.0)THEN
      RLGR(K)=AR(I)*ROK(IJ)+BR(I)*ROK(IJR)
      RLGZ(K)=AZ(J)*ROK(IJ)+BZ(J)*ROK(IJT)
      ENDIF
5     CONTINUE
C
      THAVR=AR(I)*TH(KV,IJ)+BR(I)*TH(KV,IJR)
      THAVZ=AZ(J)*TH(KV,IJ)+BZ(J)*TH(KV,IJT)
      IGKU=1000*(THAVR+THAVRN)
      IGKV=1000*(THAVZ+THAVZN)
      IGK=TH(0,IJ)*1000
      IGKR=TH(0,IJR)*1000
      IGKT=TH(0,IJT)*1000
C
      DO 10 K=0,NSOLID
      RKR(K)=AR(I)*RLK(K,IJ)+BR(I)*RLK(K,IJR)
      RKZ(K)=AZ(J)*RLK(K,IJ)+BZ(J)*RLK(K,IJT)
10    CONTINUE
C
      IF(MODAB.EQ.2)THEN
C  MODEL-B (MODIFIED)
```

```
      RKRAB(KV)=RLGR(KV)
      RKZAB(KV)=RLGZ(KV)
      IF(NSOLID.EQ.1)THEN
      RKRAB(KVN)=(1.0-RLGR(KV)/RLGR(KVN))*RKR(KVN)
      RKZAB(KVN)=(1.0-RLGZ(KV)/RLGZ(KVN))*RKZ(KVN)
      ELSE
      DO 20 K=0,NSOLID
      IF(K.NE.KV)THEN
      RKRAB(K)=RKR(K)*(1.0-(AR(I)*RLX(IJ)+BR(I)*RLX(IJR))
     1/RLGR(K))/THAVR
      RKZAB(K)=RKZ(K)*(1.0-(AZ(J)*RLX(IJ)+BZ(J)*RLX(IJT))
     1/RLGZ(K))/THAVZ
      ENDIF
 20   CONTINUE
      ENDIF
      ELSE
C  MODEL-A
      DO 30 K=0,NSOLID
      RKRAB(K)=RKR(K)
      RKZAB(K)=RKZ(K)
 30   CONTINUE
      ENDIF
C
      DO 40 K=0,NSOLID
      CALL ULMOMF
      RUK(K,IJ)=RKR(K)*UK(K,IJ)+RKRAB(K)*GRAVX*DT
     1-DTOBDR(I)*(ULFR(K)-ULFL(K))
     1-DTODZ(J)*(ULFT(K)-ULFB(K,I))-DT*SULC(K)
      ULFL(K)=ULFR(K)
      ULFB(K,I)=ULFT(K)
      CALL VLMOMF
      RVK(K,IJ)=RKZ(K)*VK(K,IJ)+RKZAB(K)*GRAVY*DT
```

```
     1-DTORDR(I)*(VLFR(K)-VLFL(K))
     1-DTODZP(J)*(VLFT(K)-VLFB(K,I))
      VLFL(K)=VLFR(K)
      VLFB(K,I)=VLFT(K)
40    CONTINUE
C
      DO 50 K=NCONT,NSOLID
      RUK(K,IJ)=RUK(K,IJ)-DTODRP(I)*(GTH(IGKU)*(TH(K,IJR)
     1-TH(K,IJ))+PS(K,IJR)-PS(K,IJ)-COHF(IGKR)+COHF(IGK))
      RVK(K,IJ)=RVK(K,IJ)-DTODZP(J)*(GTH(IGKV)*(TH(K,IJT)
     1-TH(K,IJ))+PS(K,IJT)-PS(K,IJ)-COHF(IGKT)+COHF(IGK))
50    CONTINUE
      CALL KDRAGS(NCONT,NSOLID)
      CALL MULTI
100   CONTINUE
C
C  CALCULATE VELOCITY ESTIMATES
      DO 200 J=2,JB1
      DO 200 I=2,IB1
      IJ=I+(J-1)*IB2
      IF(IFL(IJ).NE.1)GOTO 200
      IPJ=INDC(IJ,1)
      IJP=INDC(IJ,3)
      IJR=INDS(IJ,1)
      IJT=INDS(IJ,3)
      KV=NTHS(IJ)
      CALL MATSA
      CALL VELSK2
      CALL MASFKA(KV,KV)
200   CONTINUE
      RETURN
      END
```

```
C ---------------------------------------ULMOMF
      SUBROUTINE ULMOMF
      INCLUDE 'misb.com'
C
C  CALCULATE FLUXES FOR RADIAL MOMENTUM FOR THE PHASES
C
      CS=0.5*(RB(I)*UK(K,IJ)+RB(I+1)*UK(K,IPJ))
      IF(CS.GE.0.)THEN
      ULFR(K)=(AR(I)*RLK(K,IJ)+BR(I)*RLK(K,IJR))*UK(K,IJ)*CS
      ELSE
      ULFR(K)=(AR(I+1)*RLK(K,IJR)+BR(I+1)*RLK(K,IJRR))
     1*UK(K,IPJ)*CS
      ENDIF
      CS=AR(I)*VK(K,IJ)+BR(I)*VK(K,IPJ)
      IF(CS.GE.0.)THEN
      ULFT(K)=(AR(I)*RLK(K,IJ)+BR(I)*RLK(K,IJR))*UK(K,IJ)*CS
      ELSE
      ULFT(K)=(AR(I)*RLK(K,IJT)+BR(I)*RLK(K,IJTR))
     1*UK(K,IJP)*CS
      ENDIF
      IF(IFL(IMJ).NE.1)GOTO 1
      IF(IFL(IJM).NE.1)GOTO 2
      CALL ULVSB
      RETURN
C
1     CS=0.5*(RB(I)*UK(K,IJ)+RB(I-1)*UK(K,IMJ))
      IF(CS.GE.0.)THEN
      ULFL(K)=(BR(I-1)*RLK(K,IJ)+AR(I-1)*RLK(K,IJL))
     1*UK(K,IMJ)*CS
      ELSE
      ULFL(K)=(AR(I)*RLK(K,IJ)+BR(I)*RLK(K,IJR))*UK(K,IJ)*CS
      ENDIF
```

```
      IF(IFL(IJM).NE.1)GOTO2
      CALL ULVSA
      RETURN
C
2     CS=AR(I)*VK(K,IJM)+BR(I)*VK(K,IPJM)
      IF(CS.GE.0.)THEN
      ULFB(K,I)=(AR(I)*RLK(K,IJB)+BR(I)*RLK(K,IJBR))
     1*UK(K,IJM)*CS
      ELSE
      ULFB(K,I)=(AR(I)*RLK(K,IJ)+BR(I)*RLK(K,IJR))*UK(K,IJ)*CS
      ENDIF
      CALL ULVS
      RETURN
      END
C  ----------------------------------------ULVS
      SUBROUTINE ULVS
      INCLUDE 'misb.com'
C
C  CALCULATE TH*SIGMA(R,Z) AT I+1/2, J-1/2
      SULB(K)=((VK(K,IPJM)-VK(K,IJM))*RDRP(I)
     1+(UK(K,IJ)-UK(K,IJM))*RDZP(J-1))*(AZ(J-1)
     1*(AR(I)*VISCL(K,IJB)+BR(I)*VISCL(K,IJBR))
     1+BZ(J-1)*(AR(I)*VISCL(K,IJ)+BR(I)*VISCL(K,IJR)))
      ULFB(K,I)=ULFB(K,I)-SULB(K)
C
C  CALCULATE R*TH*SIGMA(R,R) AT I, J
      ENTRY ULVSA
      IF(IFL(IMJ).NE.1)THEN
      SULL(K)=2.*VISCL(K,IJ)*(UK(K,IJ)-UK(K,IMJ))*RDR(I)
     1 +(VISBL(K,IJ)-(2./3.)*VISCL(K,IJ))
     1 *(RRIDR(I)*(RB(I)*UK(K,IJ)-RB(I-1)*UK(K,IMJ))
     1 +(VK(K,IJ)-VK(K,IJM))*RDZ(J))*R(I)
```

```
      ULFL(K)=ULFL(K)-SULL(K)
      ENDIF
C
C  CALCULATE TH*SIGMA(R,Z) AT I+1/2, J+1/2
C         R*TH*SIGMA(R,R) AT I+1, J
C         (TH/R)*SIGMA(PHI,PHI) AT I+1/2, J
      ENTRY ULVSB
      SULT(K)=((VK(K,IPJ)-VK(K,IJ))*RDRP(I)
     1+(UK(K,IJP)-UK(K,IJ))*RDZP(J))*(AZ(J)
     1*(AR(I)*VISCL(K,IJ)+BR(I)*VISCL(K,IJR))
     1+BZ(J)*(AR(I)*VISCL(K,IJT)+BR(I)*VISCL(K,IJTR)))
      SULR(K)=2.*VISCL(K,IJR)*(UK(K,IPJ)-UK(K,IJ))*RDR(I+1)
     1+(VISBL(K,IJR)-(2./3.)*VISCL(K,IJR))
     1*(RRIDR(I+1)*(RB(I+1)*UK(K,IPJ)-RB(I)*UK(K,IJ))
     1+(VK(K,IPJ)-VK(K,IPJM))*RDZ(J))*R(I+1)
      ULFT(K)=ULFT(K)-SULT(K)
      ULFR(K)=ULFR(K)-SULR(K)
      IF(ITC.NE.0)THEN
      SULC(K)=2.*(AR(I)*VISCL(K,IJ)+BR(I)*VISCL(K,IJR))
     1*RRB(I)*UK(K,IJ)+((AR(I)*VISBL(K,IJ)
     1+BR(I)*VISBL(K,IJR))-(2./3.)*(AR(I)*VISCL(K,IJ)
     1+BR(I)*VISCL(K,IJR))) *(0.5*RRIDRP(I)*(RB(I+1)*UK(K,IPJ)
     1-RB(I-1)*UK(K,IMJ))+(AR(I)*(VK(K,IJ)-VK(K,IJM))
     1+BR(I)*(VK(K,IPJ)-VK(K,IPJM)))*RDZ(J))
      ELSE
      SULC(K)=0.0
      ENDIF
      RETURN
      END
C ----------------------------------------VARDT
      SUBROUTINE VARDT
      INCLUDE 'misb.com'
```

```
C
C  ADJUSTS TIME INCREMENT (DT) DURING EXECUTION OF THE
PROGRAM
C
C  IF MAX. ITERATION OCCURS, REDUCE DT
    ITD=1
    ITX=ITX+1
    NITER(ITX)=0
    MAXIT=.FALSE.
    IF(ITX.LE.ITXMX)THEN
    DT=DTNXT(ITX)
C  READ RESTART FILE TO START AGAIN WITH LOWER DT
    REWIND(9)
    CALL TAPERD
    TDUMP1=TDUMP1-TDUMP
    CALL SETUP
    ELSE
    ITX=ITXMX
    ENDIF
    GOTO 100
C
    ENTRY VARDTI
C  IF CONVERGENCE IS ACHIEVED FOR MORE THAN IMX TIMES
C  FOR A DT, INCREASE DT
    ITX=ITXN
    DT=DTNXT(ITX)
    ITD=1
    CALL SETRZ
100  CONTINUE
    ITINT=0
    RETURN
    END
```

```
C ----------------------------------------VELINV
      SUBROUTINE VELINV(NS,NP,A,B)
      DIMENSION A(NP,NP),B(NP)
C
C     USE GAUSS-DOLITTLE METHOD FOR SYMMETRIC MATRIX
INVERSION
      DO 136 K=2,NP
      IF(ABS(A(K,K)).GE.1.E-6)GOTO 136
      DO 135 KK=1,NP
      A(K,KK)=0.0
135   A(KK,K)=0.0
      B(K)=0.0
136   CONTINUE
C
      DO 160 K=1,NP
      IF(A(K,K).EQ.0.0)GOTO 160
      KP1=K+1
      DIV=1./A(K,K)
      DO 140 KJ=KP1,NP
140   A(K,KJ)=A(K,KJ)*DIV
      B(K)=B(K)*DIV
      DO 150 KI=KP1,NP
      AMUL=A(KI,K)
      DO 145 KJ=KP1,NP
145   A(KI,KJ)=A(KI,KJ)-AMUL*A(K,KJ)
150   B(KI)=B(KI)-AMUL*B(K)
160   CONTINUE
      DO 170 K=NS,1,-1
      KP1=K+1
      DO 170 KI=KP1,NP
170   B(K)=B(K)-B(KI)*A(K,KI)
      RETURN
```

```
      END
C  ----------------------------------------VELSK
      SUBROUTINE VELSK
      INCLUDE 'misb.com'
C
C  CALCULATE (8) VELOCITIES ON THE FOUR BOUNDARIES OF THE
CELL
C
      IFLL=IFL(IMJ)
      IF(IFLL.EQ.2.OR.IFLL.EQ.3.OR.IFLL.EQ.5)GOTO 200
      CALL VELINV(NSOLID,NPHASE,AU1,BU1)
      DO 165 K=0,NSOLID
165   UK(K,IMJ)=BU1(K+1)
200   CONTINUE
      IFLB=IFL(IJM)
      IF(IFLB.EQ.2.OR.IFLB.EQ.3.OR.IFLB.EQ.5)GOTO 300
      CALL VELINV(NSOLID,NPHASE,AV1,BV1)
      DO 265 K=0,NSOLID
265   VK(K,IJM)=BV1(K+1)
C
      ENTRY VELSK2
300   CONTINUE
      IFLR=IFL(IPJ)
      IF(IFLR.EQ.2.OR.IFLR.EQ.3.OR.IFLR.EQ.5)GOTO 400
      CALL VELINV(NSOLID,NPHASE,AU,BU)
      DO 365 K=0,NSOLID
365   UK(K,IJ)=BU(K+1)
400   CONTINUE
      IFLT=IFL(IJP)
      IF(IFLT.EQ.2.OR.IFLT.EQ.3.OR.IFLT.EQ.5)RETURN
      CALL VELINV(NSOLID,NPHASE,AV,BV)
      DO 465 K=0,NSOLID
```

```
465  VK(K,IJ)=BV(K+1)
     RETURN
     END
C  ----------------------------------------VLMOMF
     SUBROUTINE VLMOMF
     INCLUDE 'misb.com'
C
C  CALCULATES FLUXES OF AXIAL MOMENTUM FOR THE PHASES
C
     CS=0.5*(VK(K,IJ)+VK(K,IJP))
     IF(CS.GE.0.)THEN
     VLFT(K)=(AZ(J)*RLK(K,IJ)+BZ(J)*RLK(K,IJT))*VK(K,IJ)*CS
     ELSE
     VLFT(K)=(AZ(J+1)*RLK(K,IJT)+BZ(J+1)*RLK(K,IJTT))
    1*VK(K,IJP)*CS
     ENDIF
     CS=(AZ(J)*UK(K,IJ)+BZ(J)*UK(K,IJP))*RB(I)
     IF(CS.GE.0.)THEN
     VLFR(K)=(AZ(J)*RLK(K,IJ)+BZ(J)*RLK(K,IJT))*VK(K,IJ)*CS
     ELSE
     VLFR(K)=(AZ(J)*RLK(K,IJR)+BZ(J)*RLK(K,IJTR))
    1*VK(K,IPJ)*CS
     ENDIF
     IF(IFL(IMJ).NE.1)GOTO 1
     IF(IFL(IJM).NE.1)GOTO 2
     CALL VLVSB
     RETURN
C
1    CS=(AZ(J)*UK(K,IMJ)+BZ(J)*UK(K,IMJP))*RB(I-1)
     IF(CS.GE.0.)THEN
     VLFL(K)=(AZ(J)*RLK(K,IJL)+BZ(J)*RLK(K,IJTL))
    1*VK(K,IMJ)*CS
```

```
      ELSE
      VLFL(K)=(AZ(J)*RLK(K,IJ)+BZ(J)*RLK(K,IJT))*VK(K,IJ)*CS
      ENDIF
      IF(IFL(IJM).NE.1)GOTO2
      CALL VLVSA
      RETURN
C
2     CS=0.5*(VK(K,IJM)+VK(K,IJ))
      IF(CS.GE.0.)THEN
      VLFB(K,I)=(BZ(J-1)*RLK(K,IJ)+AZ(J-1)*RLK(K,IJB))
     1*VK(K,IJM)*CS
      ELSE
      VLFB(K,I)=(AZ(J)*RLK(K,IJ)+BZ(J)*RLK(K,IJT))*VK(K,IJ)*CS
      ENDIF
      CALL VLVS
      RETURN
      END
C ----------------------------------------VLVS
      SUBROUTINE VLVS
      INCLUDE 'misb.com'
C
C  CALCULATE TH*SIGMA(Z,Z) AT I, J
      CS=(VK(K,IJ)-VK(K,IJM))*RDZ(J)
      SVLB(K)=2.*VISCL(K,IJ)*CS+(VISBL(K,IJ)
     1-(2./3.)*VISCL(K,IJ))*(CS+RRIDR(I)*(RB(I)
     1*UK(K,IJ)-RB(I-1)*UK(K,IMJ)))
      VLFB(K,I)=VLFB(K,I)-SVLB(K)
C
C  CALCULATE R*TH*SIGMA(R,Z) AT I-1/2, J+1/2
      ENTRY VLVSA
      IF(IFL(IMJ).EQ.1)THEN
      SVLL(K)=((VK(K,IJ)-VK(K,IMJ))*RDRP(I-1)
```

```
     1+(UK(K,IMJP)-UK(K,IMJ))*RDZP(J))*RB(I-1)
     1*(AZ(J)*(BR(I-1)*VISCL(K,IJ)+AR(I-1)*VISCL(K,IJL))
     1+BZ(J)*(BR(I-1)*VISCL(K,IJT)+AR(I-1)*VISCL(K,IJTL)))
      VLFL(K)=VLFL(K)-SVLL(K)
      ENDIF
C
C  CALCULATE TH*SIGMA(Z,Z) AT I, J+1
C          R*TH*SIGMA(R,Z) AT I+1/2, J+1/2
      ENTRY VLVSB
      CS=(VK(K,IJP)-VK(K,IJ))*RDZ(J+1)
      SVLT(K)=2.*VISCL(K,IJT)*CS+(VISBL(K,IJT)
     1-(2./3.)*VISCL(K,IJT))*(CS+RRIDR(I)
     1*(RB(I)*UK(K,IJP)-RB(I-1)*UK(K,IMJP)))
      SVLR(K)=((VK(K,IPJ)-VK(K,IJ))*RDRP(I)
     1+(UK(K,IJP)-UK(K,IJ))*RDZP(J))*RB(I)
     1*(AZ(J)*(AR(I)*VISCL(K,IJ)+BR(I)*VISCL(K,IJR))
     1+BZ(J)*(AR(I)*VISCL(K,IJT)+BR(I)*VISCL(K,IJTR)))
      VLFT(K)=VLFT(K)-SVLT(K)
      VLFR(K)=VLFR(K)-SVLR(K)
      RETURN
      END
C  ----------------------------------------VRELS
      SUBROUTINE VRELS
      INCLUDE 'misb.com'
C
C  CALCULATE SQUARE OF RELATIVE VELOCITY BETWEEN FIELDS
      DO 10 K=0,NSOLID
      IF(K.NE.KV)THEN
      DV=0.5*(VK(KV,IJ)-VK(K,IJ)+VK(KV,IJM)-VK(K,IJM))
      DU=0.5*(RB(I)*(UK(KV,IJ)-UK(K,IJ))
     1+RB(I-1)*(UK(KV,IMJ)-UK(K,IMJ)))/R(I)
      SVREL(K)=DU*DU+DV*DV
```

```
      ENDIF
10    CONTINUE
      RETURN
      END
```

CURRICULUM VITAE

Name: Suttipong Songprawat

Date of Birth: May 25, 1979

Nationality: Thai

University Education:

1997-2000 Bachelor Degree of Science (1st class honors) in Department of Chemical Technology (Chemical Engineering/Fuel Technology), Faculty of Science, Chulalongkorn University, Bangkok, Thailand.