LA-UR - 85 - 2479

AUG 07 1985
CONF-8504150--2

LA-UR--85-2479

DE85 015707

TITLE: THREE-DIMENSIONAL FREE LAGRANGIAN HYDRODYNAMICS

AUTHOR(S) Harold Trease, X-7

SUBMITTED TO: 1) Conference on Numerical Methods for Fluid Dynamics
University of Reading, UNITED KINGDOM
April 1-4, 1985

2) Conference on Free Lagrangian Methods
Hilton Head Island, South Carolina
March 4-6, 1985

THREE-DIMENSIONAL FREE LAGRANGIAN HYDRODYNAMICS

Dr. Harold E. Trease
Computational Physics Group (X-7)
Los Alamos National Laboratory
Los Alamos, New Mexico

INTRODUCTION:

The purpose of the following discussion is to describe the development of a 3-D free Lagrangian hydrodynamics algorithm. The 3-D algorithm is an outgrowth of the 2-D free Lagrange model that is fully described in Ref. 1. Only the more pertinent issues of the free Lagrange algorithm will be presented, the details of the rest of the code development project are interesting but not appropriate in the context of the free Langrange conference. Let it suffice to say that a complete production code is being developed to support the free Lagrange algorithm to be described. A graphic description that outlines this code development project is presented in Figure 1.

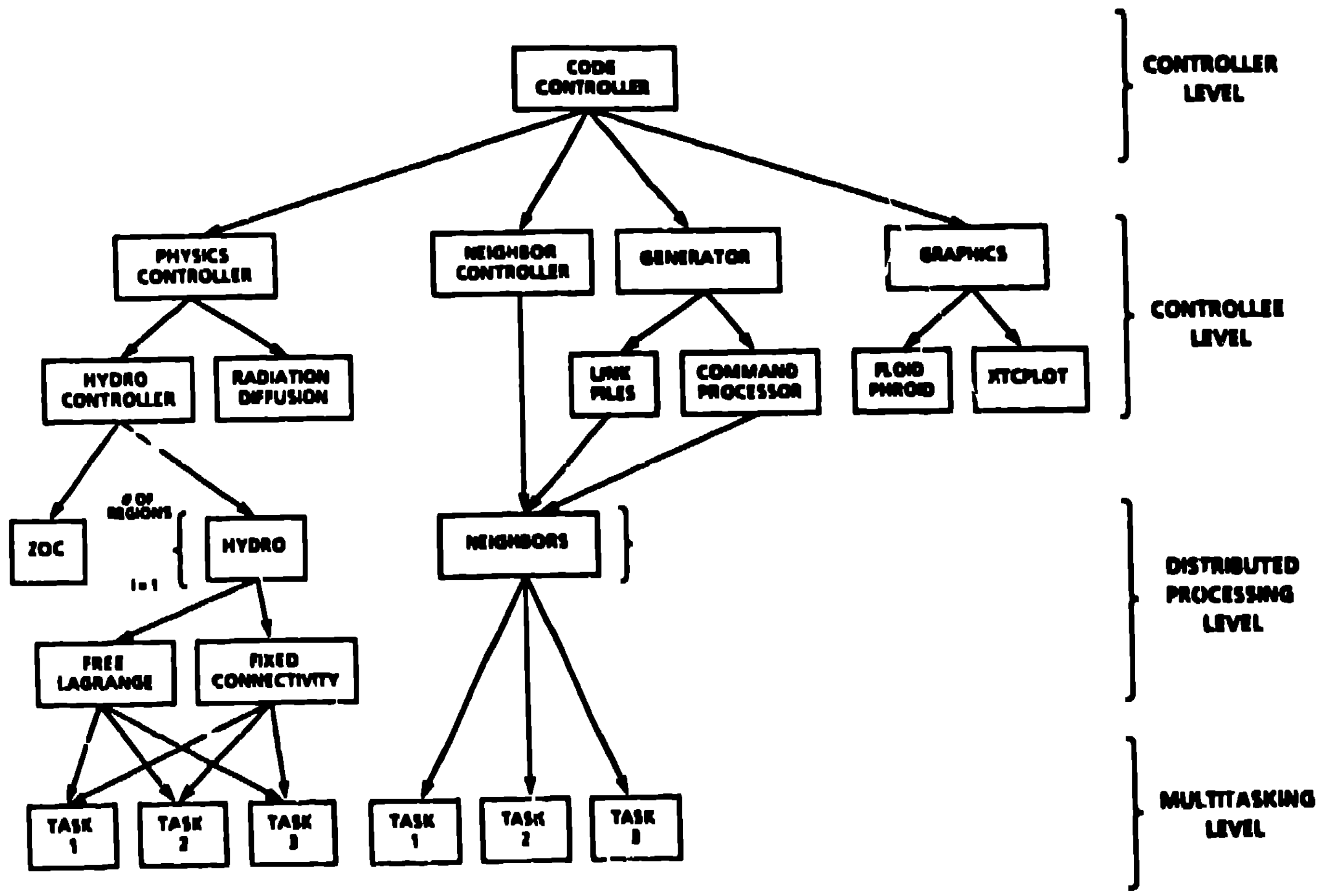


Figure 1. Block structure of the 3-D code, illustrating the four levels that make up the code.

The main objective of this project is to develop a computer model that can be used to simulate fluid flow in three dimensions. The inspiration for using free Lagrange as a basis for a hydrodynamics code was gained from the work of Crowley (Ref. 2), Fritts and Boris (Ref. 3), and Kirkpatrick (Ref. 4). The 2-D code, described in Ref. 1, was based on this previous work. This two dimensional model showed several attractive features about free Lagrange. First, it showed that free Lagrange can be used to handle fluid flow problems that exhibit strong shearing forces which classically could only be handled by Eulerian type algorithms.

Second, the accuracy of a free Lagrange algorithm was shown to be sufficient, with the irregular mesh, to produce credible solutions. Third, due to the arbitrary connectivity of the free Lagrange logic, meshes can be variably zoned. This allows the user to put the resolution where it is needed.

All three of these features of free Lagrange have been exploited to extend the 2-D model to include the third dimension. In doing so many aspects of large scale code development have been investigated. Several modern software tools have been used to make the manipulation of the arbitrary connectivity matrix, that is associated with free Lagrange, easier. Just for reference these tools include; a dynamic (heap) memory manager, a storage block manager and also a relational data base manager.

DESCRIPTION OF THE FREE LAGRANGE ALGORITHM:

The main features of the free Lagrange algorithm that identifies it from a standard Lagrange algorithm is the connectivity matrix that both defines the nearest neighbors for each point and the shape of the computational cells over which the fluid equations are integrated. A construction technique, that creates a VORONOI mesh, is used to identify nearest neighbors and define the mesh cells. The Voronoi mesh that is constructed has several properties that make it an excellent choice for maintaining the connectivity matrix for the 3-D code, these are:

A) The set of resulting polyhedra map the space defined by the mass points and bounding surfaces uniquely, i.e., none of the polyhedra overlap and nearest neighbors are guaranteed to be reciprocal.

B) Each polyhedron remains convex. This is accomplished by changing the area and the number of faces, i.e., neighbor swapping.

C) The volume and surface area of each polyhedron changes continuously. These and other aspects of the Voronoi mesh will be discussed more thoroughly in later sections.

One of the more important goals of the 3-D code is to be able to couple various hydrodynamic algorithms together. This means that free Lagrangian hydro will be used in regions where the flow field is most distorted. Then, in the (more) well behaved regions we will use an adaptive rezoning technique, with a mesh composed of mass points that have fixed connectivity. These two algorithms will then be coupled through a third algorithm called a ZOC. The free Lagrange and the ZOC algorithms will now be described in detail. The detailed hydrodynamic equations, that are solved by these algorithms, along with their finite difference representations are discussed in detail in Appendix A.

The basic features that describe the free Lagrange algorithm are listed below:

A) All mesh quantities are cell centered.

B) The computational domain is described by an arbitrary distribution of mass points.

C) The code automatically constructs its connectivity matrix.

D) Mass points can be merged and/or added to the mesh.

The code determines its connectivity matrix by constructing a unique polyhedron about each mass point. The resulting polyhedral mesh is known as a Voronoi mesh. The faces of the polyhedron determine the set of "nearest" neighbors with which a mass point interacts. The faces of the polyhedron are represented by intersecting perpendicular, bisecting planes between a given mass point and each of its neighbors. The details of this construction process are described more fully in Appendix B. Figure 2 shows several examples of Voronoi cells. The set of polyhedra that describe the mesh completely and uniquely span the space over which the mass points are distributed. Figure 3 shows a 2-dimensional projection of a 3-dimensional mesh. where the arbitrary polyhedra reduce to arbitrary polygons.

Due to the fact that all physical quantities are carried at cell centered mass points, each point can change the set of "nearest" neighbors that it associates with by changing the shape of the polyhedron surrounding it while still retaining its Lagrangian definition. The neighbor changing process is smooth and continuous because of the integral nature of the algorithm. Two points become neighbors when a face with "epsilon" surface area appears between them. These points will drop each other as "nearest" neighbors when (and if) this face area shrinks below "epsilon".

There are several advantages and disadvantages associated with free Lagrange hydro. The advantages are obvious to anyone doing hydrodynamic calculations. Due to the arbitrary connectivity of the mesh and the ability to change this connectivity, highly distorted flows can be modeled by using a Lagrangian algorithm. Also, since the mesh maintains itself, <u>no manual rezoning</u> is needed (this is extremely important in a 3-dimensional code). Complex geometries that require variable zoning can be setup relatively easily since the code figures out the connectivity matrix from an arbitrary distribution of mass points. The main disadvantage of this method is the overhead associated with maintaining and processing the connectivity lists, but since the neighbor lists are unique, they are very amenable to a calculation using a multitasking algorithm (i.e., the neighbor searches can be done in any order, but the resulting global connectivity matrix is the same). Also. future machines that support hardware gather-scatter operations

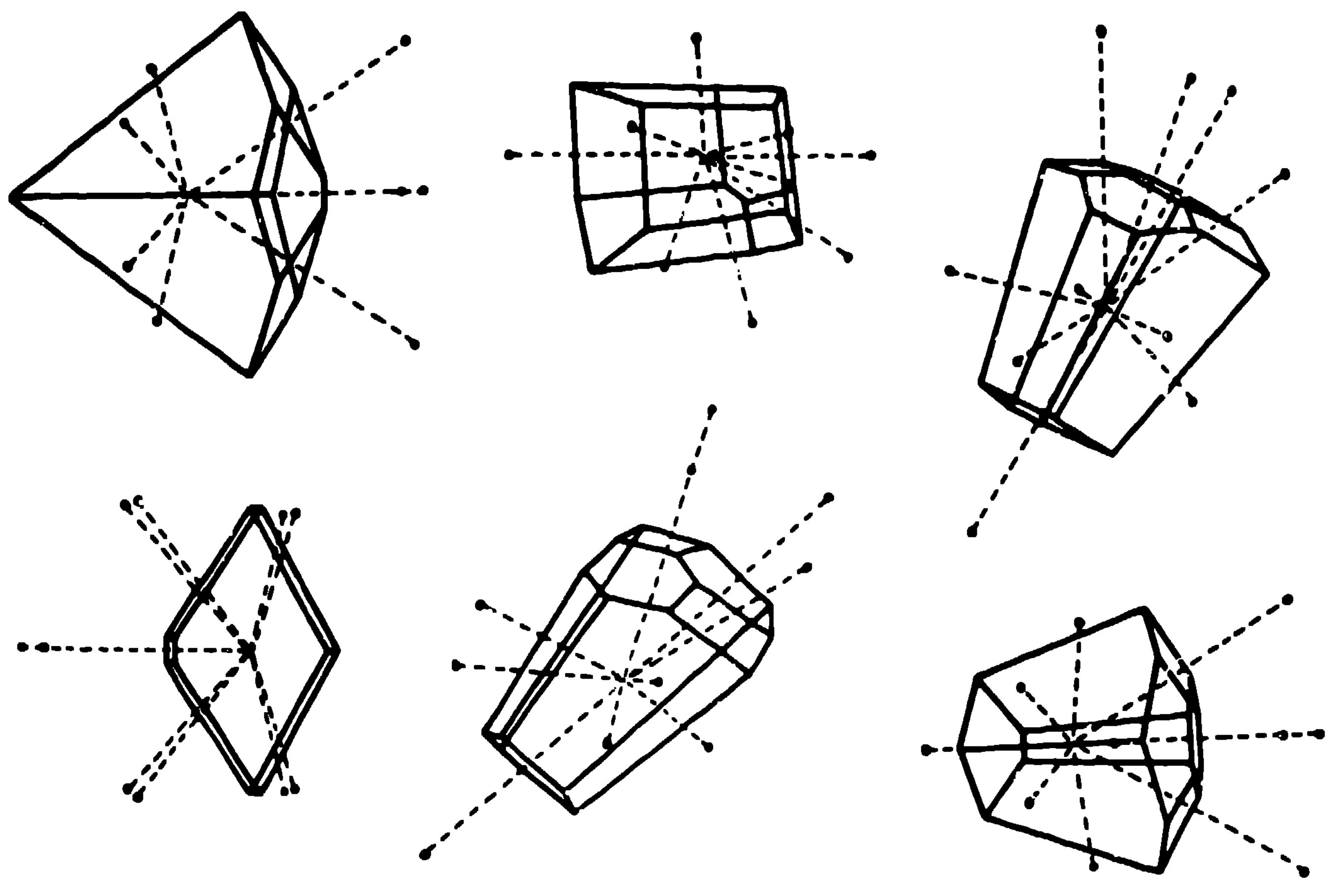


Figure 2. Examples of several Voronoi cells. The Voronoi cells are the polyhedral shaped objects. The straight lines that end at a point represent the connections between the central mass point and its "nearest" neighbors.

will improve the efficiency of this algorithm.

One of the weakest aspects of the free Lagrange method just described is the treatment of continuous interfaces. This results from the fact that the edges of the computational volumes are arbitrarily defined to be midway between two "nearest" neighbors. This definition, while consistent with the Lagrangian equations, leads to a poor definition of a continuous interface. The realization of this fact suggested that an interface tracking algorithm was needed to follow the motion of interfaces. We will now describe the algorithm that is being used and how we intend to develop it into a full blown interface tracking algorithm, along with some of the positive side effects, in relation to distributed processing and slip-line treatment.

A little reflection on two key properties of interfaces will help the reader's understanding of the algorithm to be described. First, a continuous interface separates what could be considered immiscible fluids, i.e., fluid "A" remains distinct from fluid "B" even though interpenetration may occur. Second, an interface can be described in a space that is one dimension less than the rest of the problem. In one dimension an interface is a point, in two dimensions it is a line, and in three dimensions it is a surface. Generalizing this idea we can

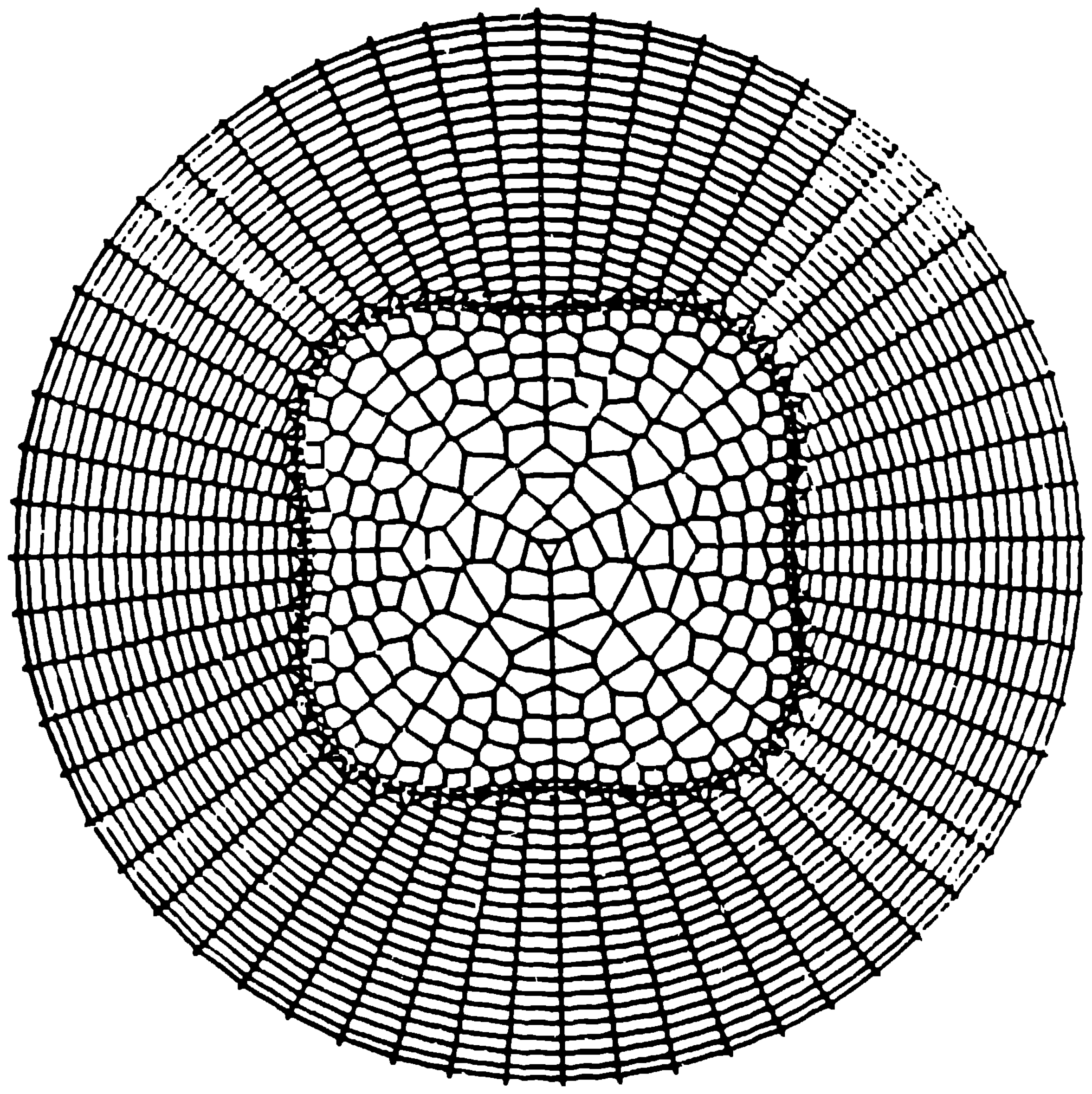


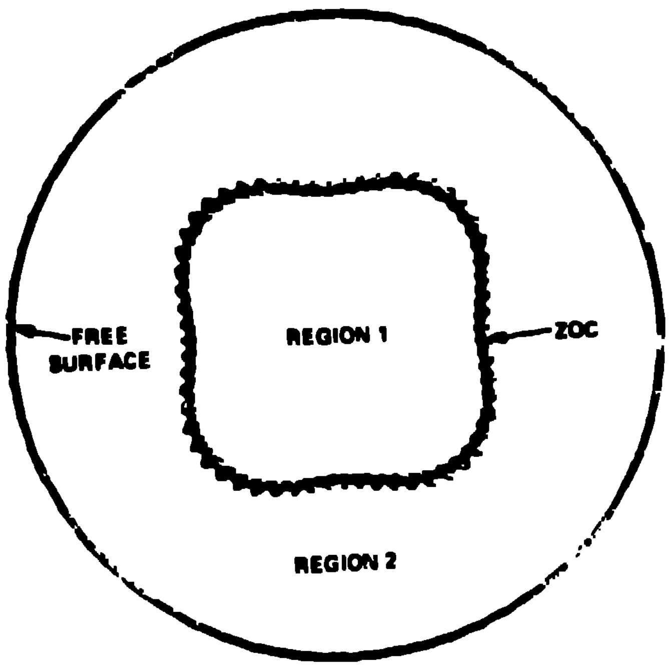
Figure 3. 2-dimensional projection of a 3-dimensional Voronoi mesh.

represent an interface as a (N-1) construct in a N-dimensional space. Taking these two concepts into consideration we came up with an interface tracking construct called a ZOC. An example of a ZOC is shown in Figure 4, where several observations can be made. First, we can see that the interface separating the two fluids is distinct. Also, we notice that zoning away from the interface in the two regions is discontinuous with respect to the other region and the interface.

Most of the technical details of maintaining a ZOC will not be discussed, but some of the more general aspects of this interface tracking concept may be interesting. These are listed below:

A) A ZOC is essentially a special free Lagrange region that uses its connectivity matrix to connect to the surrounding regions.

B) Points can be added or subtracted from a ZOC to maintain the interface.



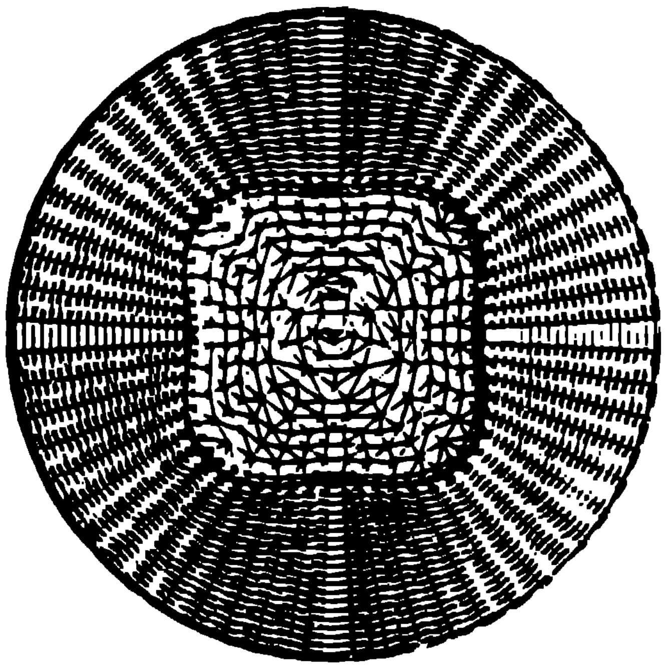


Figure 4. Example of ZOC. The left figure shows the ZOC in relation to the free surface. The right figure shows the corresponding grid.

This process is made especially easy since a connectivity matrix is used to connect points and thus the mesh reconnections account for the fact that a point has been added or subtracted. The process of adding new points is trivial because the Voronoi mesh indicates when a new point should be added and where the new point should be located. The rest of the details of adding a point involves the redistribution of mass, momentum, and energy in a local region of space.

C) A ZOC will work in two dimensions, to maintain a line interface, just as well as in three dimensions.

D) The treatment of slip-lines should be automatic with a ZOC since there is no restriction on the tangential velocity of the fluid on either side of the interface.

E) This interface will be used to connect regions that use different hydro algorithms. This means a free Lagrange algorithm can be used on one side of the interface and a fixed mesh algorithm on the other side.

F) A ZOC makes a natural communication buffer for connecting two separate algorithms that are running as distributed processes. This is where the dimensionality of the ZOC becomes important because the data transfer between processes must be kept to a minimum. The separate region processes are N-dimensional data structures and a ZOC is a (N-1) data structure, which means the amount of information being communicated between the regions is small compared to the regions themselves.

APPENDIX A
THE HYDRODYNAMIC EQUATIONS AND THEIR FINITE DIFFERENCE REPRESENTATION

The hydrodynamic equations that are solved on this free Lagrange system of mass points are given below. These equations represent the conservation of mass, momentum, and specific internal energy, respectively;

Continuity Equation;

$$\frac{1}{\rho}\frac{D\rho}{Dt} = -\bar{\nabla}\cdot\bar{u} \quad , \qquad \text{(Eq. 1)}$$

Conservation of Momentum;

$$\rho\frac{D\bar{u}}{Dt} = -\bar{\nabla}p - [\bar{\nabla}\cdot\bar{\bar{T}}] \quad , \qquad \text{(Eq. 2)}$$

Conservation of Internal Energy;

$$\rho\frac{D\varepsilon}{Dt} = -p(\bar{\nabla}\cdot\bar{u}) - (\bar{\bar{T}}:\nabla u) \quad , \qquad \text{(Eq. 3)}$$

where,

ρ = fluid density ,

p = fluid pressure ,

ε = internal energy/unit mass ,

$\bar{u}$ = fluid velocity vector ,

$\bar{\bar{T}}$ = total stress tensor , and

$= \bar{\bar{\tau}} + \bar{\bar{Q}}$,

where,

$\bar{\bar{\tau}}$ = stress tensor and

$\bar{\bar{Q}}$ = artificial viscosity tensor .

$$\bar{\bar{Q}} = \begin{cases} \ell^2\rho\,\mathrm{div}(\bar{u})\{(\overline{\nabla u}) - \frac{1}{3}\mathrm{div}(\bar{u})\bar{\bar{e}}\} & , \text{ if } \mathrm{div}(\bar{u}) < 0 \\ 0 & , \text{ if } \mathrm{div}(\bar{u}) \geq 0 \end{cases} \qquad \text{(Eq. 4)}$$

where,

$\mathrm{div}(\bar{u})$ = divergence of velocity vector $\bar{u}$,

∇u = dyadic product of the differential operator $\bar{\nabla}$ and the velocity vector $\bar{u}$.

$\bar{e}$ = unit tensor , and

ℓ = constant * local grid spacing .

The algorithm used to solve these equations can be outlined as follows. First, Equations 1 through 3 are integrated over an arbitrary volume (in reference to the code we integrate over a computational cell). The volume integrals are transformed to surface integrals by using the divergence theorem. The mean-value-theorem from calculus is used to obtain average quantities. These resulting equations are then cast into finite difference form as shown below. The following notation will be used in writing the finite difference representation of the fluid flow equations;

i = mass point i, spatial position at (XI,YI,ZI) ,

j = j^{th} nearest neighbor of mass point i, spatial position at (XJ,YJ,ZJ) ,

J = total number of nearest neighbors associated with mass point i ,

n = present time step (time = t) ,

$n + 1$ = next time step (time = t + Δt) ,

$R_{i,j}$ = distance from mass point i to nearest neighbor j ,

$A_{i,j}$ = area of polygon face separating mass point i from nearest neighbor j ,

$V_{i,j}$ = volume of the polyhedron associated with mass point i and nearest neighbor j ,

M_i = mass of fluid associated with mass point i ,

$P_{i,j}$ = fluid pressure at the face associated with mass point i and nearest neighbor j ,

ρ_i = fluid density in cell i ,

$\bar{U}_{i,j}$ = fluid velocity at the face associated with mass point i and nearest neighbor j , and

$\hat{n}_{i,j}$ = normal vector to the face associated with mass point i and nearest neighbor j .

$$V_i^{n+1} \left(\frac{D\rho}{Dt}\right)_i = -\rho_i^n \sum_{j=1}^{J} \hat{n}_{i,j} \cdot \bar{U}_{i,j}^{n+1} A_{i,j} \qquad \text{(Eq. 5)}$$

$$M_i\left(\frac{D\bar{U}}{Dt}\right)_i = -\sum_{j=1}^{J} P_{i,j}\, \hat{n}_{i,j}\, A_{i,j} - \sum_{j=1} \hat{n}_{i,j} \cdot \bar{\bar{T}}_{i,j}\, A_{i,j} \qquad \text{(Eq. 6)}$$

$$M_i\left(\frac{D\varepsilon}{Dt}\right)_i = -P_i \sum_{j=1}^{J} \hat{n}_{i,j} \cdot \bar{U}_{i,j}^{n+1} A_{i,j} - \sum_{j=1} \hat{n}_{i,j} \cdot [\bar{\bar{T}} \cdot \bar{U}]_{i,j}\, A_{i,j} +$$

$$+ \bar{U}_{i,j} \cdot \sum_{j=1}^{J} \hat{n}_{i,j} \cdot \bar{\bar{T}}_{i,j}\, A_{i,j} \qquad \text{(Eq. 7)}$$

APPENDIX B
MESH CONNECTIVITY
(NEAREST NEIGHBOR CALCULATIONS)

The purpose of this Appendix is to describe the manner in which the connectivity matrix for the free Lagrange algorithm is calculated. The connectivity matrix contains the "nearest" neighbors for all the mass points. These connections are used for calculating surface areas and volumes of the Voronoi cells that make up the computation mesh. In addition to describing the geometry of the cells the connectivity matrix indicates which cells will interact hydrodynamically.

Each Voronoi cell is made of an arbitrary number of intersecting planes. These planes construct a convex polyhedron with an arbitrary number of "faces" about each mass point. Each face forms a polygon with an arbitrary number of "edges". The trick is to come up with an algorithm that can calculate the connectivity matrix from an arbitrary distribution of points. As we proceed through this discussion the two following definitions should be kept in mind:

A) "face" neighbor: Any two points that are "nearest" neighbors are separated, in 3 dimensions, by a polygon shaped "face". A "face" neighbor therefore refers to the "nearest" neighbor that is across a given "face" from a given central point "I". There is a one-to-one correspondence between the number of "face" neighbors that are associated with a point and the number of "faces" that make up the polyhedral cell surrounding that point.

B) "edge" neighbor: Each of the "faces" of a Voronoi cell is a polygon. Each vertex of a given polygon "face" is found to be the center of a sphere that passes through four points. The four points are: the central point "I", its "face" neighbor "j", and two other points that are called "edge" neighbors (they are referred to as "k-1" and "k"). Also, since each polyhedron is a closed figure each "edge" neighbor must also be a "face" neighbor. It should be noted that there is a one-to-one correspondence between the number of "edges" on a given polygonal "face" and the number of "edge" neighbor associated with that "face".

The main idea in discovering the "nearest" neighbors of a point is to identify the list of "edge" neighbors for each "face" of the polyhedron surrounding that point. As each set of "edge" neighbors is discovered they are put on a stack of "face" neighbors, then the next "face" neighbor to be looked at is pulled from this stack. When the stack of "face" neighbors is empty the Voronoi polyhedron is complete and the connectivity for this given point has been found.

The process of constructing the connectivity matrix for a given point is described [illegible] completely in the following steps:

A) Assemble a list of "possible" neighbors. This process depends on whether or not the point has been calculated before.

a) If the point hasn't been calculated before.

1) Use any "logical" neighbor information to select as many potential neighbors as possible.

2) Use a proximity rule to select a set of closest points based on filters such as distance between and material type.

b) If the point has been calculated before.

1) Recall the old connectivity matrix for a given point.

2) Form the list of possible "nearest" neighbors from the old "nearest" neighbors plus the "nearest" neighbors of the old "nearest" neighbors.

B) From this list of "possible" neighbors we identify the 1st "face" neighbor by the procedure outlined below.

a) First all points are translated so the coordinate axes are centered at the central mass point, point "I", by using the following equation,

$$\overset{nkin}{\underset{\ell=1}{}} \{\vec{X}_\ell = \vec{X}_\ell - \vec{X}_I\} \qquad \text{(Eq. 8)}$$

where,

$\vec{X}_\ell$ = coordinates of a possible neighbor "ℓ" ,

$\vec{X}_I$ = coordinates of point "I" , and

nkin = number of points on the list of "possible" neighbors for point "I" .

b) Select the first "face" neighbor by using the following equation,

$$N_1 = \text{Index of } [\min_{\ell=1}^{nkin} \{|\vec{X}_\ell|\}] \qquad \text{(Eq. 9)}$$

where,

$|X_\ell|$ = the distance to point ℓ and

N_1 = index of the 1st "nearest" neighbor (this is the index into the global mesh arrays) .

c) Select the first "edge" neighbor for the first "face" by using the following equation.

$$E_{1,1} = \text{Index of } [\min_{\ell=1}^{nkin} \{W_1 |(\vec{X}_V)_1 \times (\vec{X}_V)_\ell| + W_2 \cos((\vec{X}_N)_1, \vec{X}_\ell) + W_3 |\vec{X}_\ell|\}] \qquad \text{(Eq. 10)}$$

where,

$E_{1,1} \neq N_1$,

$E_{k,j}$ = the kth "edge" neighbor of the jth "face" neighbor ,

$(\vec{X}_N)_j$ = the coordinates of the "face" neighbor "N_j" ,

W_i = weighting parameters
$W_1 > W_2 > W_3$.

$(\vec{X}_v)_\ell$ = the coordinates of the ℓth Voronoi point. This point is found as the center of a circle which passes through three points. These points are: the central point "I", the "face" neighbor "N_j" and the next possible neighbor "ℓ" ($N_\ell \neq N_j$). The equations, in matrix form, that determine the (X,Y,Z) - coordinates of the ℓth Voronoi point are:

$$\begin{bmatrix} (\vec{X}_N)_j \\ \vec{X}_\ell \\ (\vec{X}_N)_j \times \vec{X}_\ell \end{bmatrix} \cdot \begin{pmatrix} (\vec{X}_V)_\ell \end{pmatrix} = -\tfrac{1}{2} \begin{pmatrix} |(\vec{X}_N)_j|^2 \\ |\vec{X}_\ell|^2 \\ 0 \end{pmatrix}$$

NOTE: This step describes the "nearest" neighbor algorithm in 2 dimensions (Z=0). Here we just use it to "start" the 3-dimensional algorithm described below.

d) The process of selecting the rest of the "nearest" neighbors for point "I" serves a dual purpose. First, we finish calculating the rest of the "face" neighbors and we also discover the list of "edge" neighbors that make up the polygonal "face" that separates Points "I" and "j". An important point to notice is that the list of "nearest" neighbors for point "I" are contained within the sets of "edge" neighbors, i.e., the "nearest" neighbors are a subset of the "edge" neighbors. Therefore, by sifting the "edge" neirhbors we obtain a list of unique points that represent the "face" neighbors for point "I". The "edge" and "face" neighbor lists will bootstrap each other to completely describe the polyhedral cell surrounding point "I". The 3-D "nearest" neighbor selection algorithm is described below.

1) We already know the first "face" neighbor, N_1, and the first "edge" neighbor, $E_{1,1}$, for face one for point "I". These were found in Steps (B.b) and (B.c)

2) Now we calculate the "edge" neighbors for "face" j.

$$
\begin{aligned}
E_{k,j} = \text{Index of } [\min_{\ell=1}^{nkin} \{ & W_1 (\vec{X}_N)_j \cdot \\
& \cdot [[(\vec{X}_V)_{k-1} - \tfrac{1}{2}(\vec{X}_N)_j] \times \\
& \times [(\vec{X}_V)_\ell - \tfrac{1}{2}(\vec{X}_N)_j]] + \\
& + W_2 \cos((\vec{X}_N)_j, \vec{X}_\ell) + \\
& + W_3 \cos(\vec{X}_{k-1}, \vec{X}_\ell) + \\
& + W_4 |\vec{X}_\ell| \}]
\end{aligned}
\qquad \text{(Eq. 11)}
$$

where,

$E_{k,j} \neq N_j$,

$E_{k,j} \neq N_{1,j}$,

j = index of the current face ,

k = 2 to K (until $E_{k+1,j} = E_{1,j}$, where K=number of "edge" neighbors) ,

W_i = weighting parameters

$W_1 > W_2 > W_3 > W_4$,

$(\vec{X}_N)_j$ = coordinates of nearest j ,

$(\vec{X}_V)_\ell$ = coordinates of the ℓth Voronoi point and ,

$$
\left\{\begin{matrix} (\vec{X}_N)_j \\ \vec{X}_{k-1} \\ \vec{X}_\ell \end{matrix}\right\} \cdot \begin{pmatrix} (\vec{X}_V)_\ell \end{pmatrix} = -\tfrac{1}{2} \begin{pmatrix} |(\vec{X}_N)_j|^2 \\ |\vec{X}_{k-1}|^2 \\ |\vec{X}_\ell|^2 . \end{pmatrix}
$$

3) From the list of "edge" neighbors, (E_k, k=1, K), we add the unique indices to the list of "face" neighbors, N_j, j=1, J).

4) Increment j and continue with Step (d.2). This continues until all the "face" neighbors have been calculated, i.e., J > j. J is the number of "face" neighbors that are associated with point "I".

REFERENCES:

1. Trease, H. E. (1981), A Two-Dimensional Free Lagrangian Hydrodynamics Model, Ph.D. Thesis, University of Illinois, Urbana-Champaign.

2. Crowley, W. P. (1971), "FLAG: A Free Lagrange Method for Numerically Simulating Hydrodynamic Flow in Two Dimensions," Proceedings of the Second International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics, Vol. 8, pp. 37-43, Springer-Verlag, New York.

3. Fritts, M. J. and Boris, J. P. (1979), "The Lagrangian Solution of Transient Problems in Hydrodynamics using a Triangular Mesh," Journal of Computational Physics, Vol. 3, pp. 319-343, Academic Press, New York.

4. Kirkpatrick, R. C. (1976), "FREE FLOW HYDRO," Internal Report (TD-2), Los Alamos National Laboratory, Los Alamos, New Mexico.