Computer Sciences Department
The University of Wisconsin
1210 West Dayton Street
Madison, Wisconsin 53706

A NUMERICAL STUDY OF THE MIXING OF FLUIDS WITH AN EXAMPLE OF DISCRETE TURBULENCE

by

Donald Greenspan

Appendix: FORTRAN Program for the Mixing of Fluids by A. B. Schubert

Technical Report #153

April 1972

A NUMERICAL STUDY OF THE MIXING OF FLUIDS WITH AN EXAMPLE OF DISCRETE TURBULENCE

by

Donald Greenspan

Appendix: FORTRAN Program for the Mixing of Fluids

by

A. B. Schubert

ABSTRACT

The diffusion of two fluids is studied from a particle point of view. Only gravity and repulsion are included in the dynamical formulation. Examples illustrating diffusion and turbulence are presented.

A NUMERICAL STUDY OF THE MIXING OF FLUIDS WITH AN EXAMPLE OF DISCRETE TURBULENCE

1. Introduction.

The development of the high speed digital computer has rejuvenated discrete approaches to the study of fluid motions (see, e.g., refs. [1],[2],[4],[5]). Using such an approach, we will formulate and study in this paper a discrete model of the mixing of two fluids. Fundamental to the discussion is the assumption that in dealing with a fluid which consists of, say, 10^{30}, molecules, a discrete model which consists of many fewer particles may be as revealing with regard to fundamental physical mechanisms as a continuous model which consists of an infinite number of particles.

Consider then a square region ABCD, as shown in Figure 1.1. For $b > 0$, let the line $y = b$ meet AD and BC in E and F, respectively. Initially, let fluid L_1 be contained in area R_1 of rectangle CDEF, while fluid L_2 is contained in area R_2 of rectangle ABFE. Under the assumption that L_1 is more dense than L_2, the problem is to describe the resulting mixing motions of L_1 and L_2.

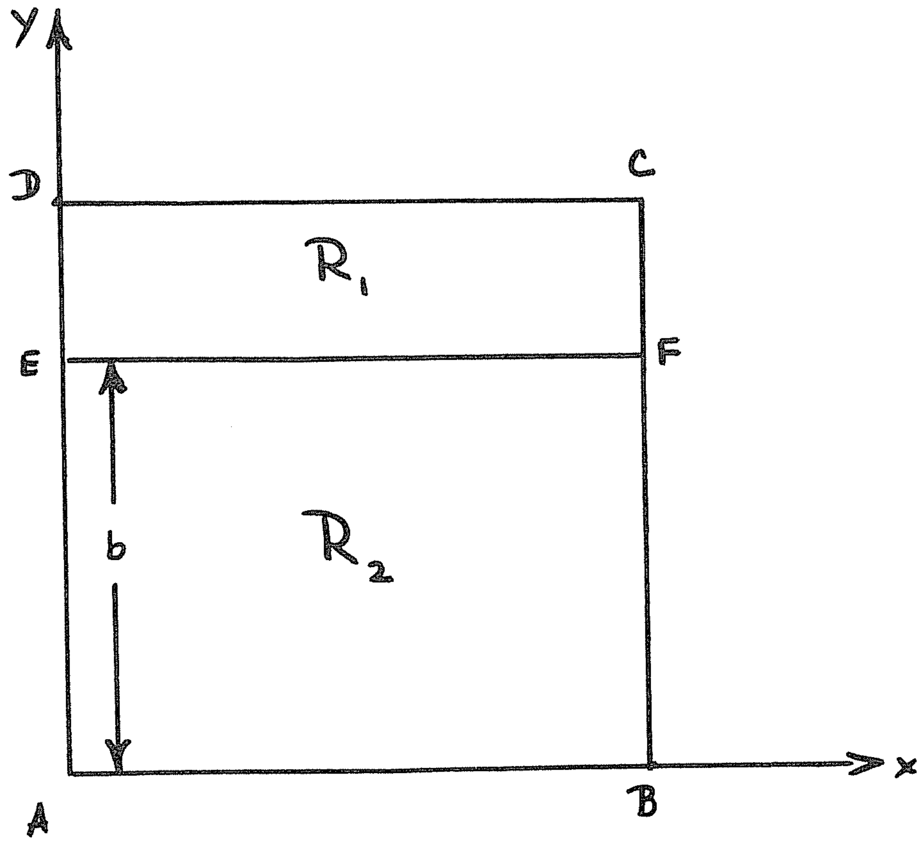

Figure 1.1

2. The Discrete Model.

Let the mass of each particle of L_1 be denoted by $m(L_1)$, while that of each particle of L_2 is denoted by $m(L_2)$. Assume that

$$m(L_1) > m(L_2).$$

Let L_1 consist of n particles $P_1, P_2, \ldots, P_n$, while L_2 consists of $N-n$ particles $P_{n+1}, P_{n+2}, \ldots, P_N$. If the mass of an arbitrary particle P_j is denoted by m_j, then, of course, m_j is necessarily one of $m(L_1)$ or $m(L_2)$.

Initially, let each P_j, $j = 1, 2, \ldots, N$ be located, or, more precisely, have its center of mass located, at $(x_{j,0}, y_{j,0})$, have velocity $(v_{j,0,x}, v_{j,0,y})$, and have acceleration $(a_{j,0,x}, a_{j,0,y})$. For $\Delta t > 0$ and $t_k = k\Delta t$, $k = 0, 1, 2, \ldots$, the position $(x_{j,k}, y_{j,k})$, the velocity $(v_{j,k+\frac{1}{2},x}, v_{j,k+\frac{1}{2},y})$, and the acceleration $(a_{j,k,x}, a_{j,k,y})$ of each P_j for each value of k are assumed to be related by

$$(2.1) \qquad v_{j,k+\frac{1}{2},x} = \begin{cases} v_{j,0,x} + \frac{\Delta t}{2}(a_{j,0,x}); & k = 0,\ j = 1, 2, \ldots, N \\ v_{j,k-\frac{1}{2},x} + (\Delta t)(a_{j,k,x}); & k = 1, 2, \ldots;\ j = 1, 2, \ldots, N \end{cases}$$

$$(2.2)\qquad v_{j,k+\frac{1}{2},y} = \begin{cases} v_{j,0,y} + \frac{\Delta t}{2}(a_{j,0,y}); & k = 0,\ j = 1,2,\ldots,N \\ v_{j,k-\frac{1}{2},y} + (\Delta t)(a_{j,k,y}); & k = 1,2,\ldots;\ j = 1,2,\ldots,N \end{cases}$$

$$(2.3)\qquad x_{j,k+1} = x_{j,k} + (\Delta t)\, v_{j,k+\frac{1}{2},x}\,; \qquad j = 1,2,\ldots,N$$

$$(2.4)\qquad y_{j,k+1} = y_{j,k} + (\Delta t)\, v_{j,k+\frac{1}{2},y}\,; \qquad j = 1,2,\ldots,N.$$

The force $(F_{j,k,x}, F_{j,k,y})$ on P_j at t_k is assumed to be related to the acceleration by the discrete Newtonian equations

$$(2.5)\qquad m_j\, a_{j,k,x} = F_{j,k,x}\,, \qquad j = 1,2,\ldots,N$$

$$(2.6)\qquad m_j\, a_{j,k,y} = F_{j,k,y}\,, \qquad j = 1,2,\ldots,N.$$

Once $F_{j,k,x}$ and $F_{j,k,y}$ are defined, then (2.1)-(2.6) determine explicitly the motion of each P_j from given initial data $x_{j,0}$, $y_{j,0}$, $v_{j,0,x}$ and $v_{j,0,y}$, $j = 1,2,\ldots,N$. Therefore, we proceed next to describe the nature of the forces to be included in the model, namely, gravity and interparticle repulsion.

For $\Delta x > 0$, if P_j is at $(x_{j,k}, y_{j,k})$ at time t_k, then the subset of particles whose centers of mass (x,y) satisfy

$$(x_{j,k} - \Delta x) < x < (x_{j,k} + \Delta x), \quad 0 \le y < y_{j,k}$$

is called the <u>support</u> <u>set</u> of P_j and is denoted by $S(P_j)$. Physically, each particle in the support set of P_j is considered to be, at least in part, "beneath" P_j and thereby can contribute to preventing it from free fall. The gravitational force $g_{j,k}$ acting upon P_j at time t_k is then defined as follows.

Let K be the largest nonnegative integer such that $y_{j,k} \geq K\Delta x$ and let d_j be a positive measure of the width, or volume, of P_j. If $K = 0$ and $y_{j,k} \leq d_j$, then set $g_{j,k} = 0$, while if $K = 0$ and $y_{j,k} > d_j$, let A be the total area of $S(P_j)$ in the rectangular region defined by

$$\begin{cases} \gamma = \max\left[x_{j,k} - \frac{\Delta x}{2}, 0\right] \leq x \leq \min\left[x_{j,k} + \frac{\Delta x}{2}, |AB|\right] = \delta \\ 0 \leq y \leq y_{j,k}, \end{cases}$$

and define $g_{j,k}$ by

$$g_{j,k} = -980 \left[1 - \frac{A}{(\delta-\gamma)y_{j,k}}\right]. \tag{2.8}$$

If $K > 0$, consider the set of K congruent rectangles beneath P_j which are bounded by

$$\begin{cases} \gamma = \max\left[x_{j,k} - \frac{\Delta x}{2}, 0\right] \leq x \leq \min\left[x_{j,k} + \frac{\Delta x}{2}, |AB|\right] = \delta \\ y = p\Delta x;\quad p = 0,1,2,\ldots,K. \end{cases}$$

If the intersection of any one of these rectangles with $S(P_j)$ is empty, set $g_{j,k} = -980$. But, if each of these squares has a nonzero intersection with $S(P_j)$ and if A is the total area of these nonzero intersections, then set

$$(2.9) \qquad g_{j,k} = -980 \left[1 - \frac{A}{(\delta-\gamma)K\Delta x}\right].$$

From (2.9), note that if $A = (\delta-\gamma)K\Delta x$, then P_j has no gravity acting upon it, that is, it is supported fully by particles below it. However, if $A < (\delta-\gamma)K\Delta x$, then $g_{j,k}$ is proportional to how much support is below P_j, while if $A > (\delta-\gamma)K\Delta x$ then many particles have been compressed beneath P_j and the resulting force will be antigravitational. Similar conclusions hold with respect to (2.8).

To simulate repulsion between the particles, whether it be due to collision or electrical forces, we will proceed as follows. Let P_i have mass m_i and be located at $(x_{i,k}, y_{i,k})$ at time t_k. Let P_j have mass m_j and be located at $(x_{j,k}, y_{j,k})$ at time t_k. Let $r_{ij,k}$ be the distance between $(x_{i,k}, y_{i,k})$ and $(x_{j,k}, y_{j,k})$. Then, the force of repulsion on P_j exerted by P_i is defined by

$$(2.10) \quad F_{j,k,x} = \frac{\alpha m_i m_j (x_j - x_i)}{(r_{ij,k}+\epsilon)^p \left(\frac{r_{ij,k}+\epsilon}{d_j}\right)^\beta}, \quad F_{j,k,y} = \frac{\alpha m_i m_j (y_j - y_i)}{(r_{ij,k}+\epsilon)^p \left(\frac{r_{ij,k}+\epsilon}{d_j}\right)^\beta},$$

where α is a nonnegative constant, ϵ is a positive measure of how close the centers of two particles are allowed to be, p is a positive exponent of repulsion, and β is a nonnegative exponent of repulsion which is zero except when $r_{ij,k} < d_j$, at which time it is positive. The effect of β is to greatly increase the force of repulsion when two particles are exceptionally close, as when, for example, they have collided.

The equations of motion of each P_j are then defined by (2.1)-(2.6) with

$$(2.11) \qquad a_{j,k,x} = \sum_{\substack{i=1 \\ i \neq j}}^{N} \frac{\alpha\, m_i (x_j - x_i)}{(r_{ij,k} + \xi)^p \left(\frac{r_{ij,k} + \xi}{d_j} \right)^{\beta}} , \qquad j = 1, 2, \ldots, N$$

$$(2.12) \qquad a_{j,k,y} = g_{j,k} + \sum_{\substack{i=1 \\ i \neq j}}^{N} \frac{\alpha\, m_i (y_j - y_i)}{(r_{ij,k} + \xi)^p \left(\frac{r_{ij,k} + \xi}{d_j} \right)^{\beta}} ; \quad j = 1, 2, \ldots, N.$$

Collision at the wall will be treated simply by assuming that the angle of incidence is the same as the angle of reflection, and that the reflected speed v_r is related to the incidence speed v_i by

$$(2.13) \qquad |v_r| = \omega |v_i| , \qquad 0 < \omega \leq 1 .$$

The initial velocities of $P_1, P_2, \ldots, P_N$ will be determined as random quantities in the ranges

$$- V \leq v_{j,0,x} \leq V$$

$$- V \leq v_{j,0,y} \leq V \quad ,$$

where V is a fixed positive constant.

3. Examples.

From the large number of examples run on the UNIVAC 1108 at the University of Wisconsin, we will describe now two which are both typical and physically reasonable. In each case, the choices $|AB| = 100$ and $b = 75$ were used for the square shown in Figure 1.1.

Example 1. Consider a sixteen particle configuration with $n = 4$, $N = 16$, $m(L_1) = 25$, $m(L_2) = 10$, $\Delta x = 25$, $\Delta t = 10^{-3}$, $d_j \equiv d = 20$, $\alpha = 1$, $\xi = 0$, $p = 2$, $\beta = 5$, $\omega = 1$, and $V = 100$. The initial positions and initial velocities were

(12.5, 87.5) ,	$v_x = -2.90$,	$v_y = 48.91$
(37.5, 87.5) ,	$v_x = 98.43$,	$v_y = -87.83$
(62.5, 87.5) ,	$v_x = 22.21$,	$v_y = 41.44$
(87.5, 87.5) ,	$v_x = -27.61$,	$v_y = 47.97$
(12.5, 62.5) ,	$v_x = 46.99$,	$v_y = -1.53$
(37.5, 62.5) ,	$v_x = -99.65$,	$v_y = -16.10$
(62.5, 62.5) ,	$v_x = 26.14$,	$v_y = 80.82$
(87.5, 62.5) ,	$v_x = -42.70$,	$v_y = 56.75$
(12.5, 37.5) ,	$v_x = 27.02$	$v_y = -75.35$
(37.5, 37.5) ,	$v_x = 98.66$,	$v_y = -69.15$
(62.5, 37.5),	$v_x = 93.48$,	$v_y = -85.65$

$(87.5, 37.5)$, $v_x = -46.35$, $v_y = 88.20$

$(12.5, 12.5)$, $v_x = -9.09$, $v_y = 90.73$

$(37.5, 12.5)$, $v_x = -6.56$, $v_y = 95.21$

$(62.5, 12.5)$, $v_x = -13.07$, $v_y = -49.69$

$(87.5, 12.5)$, $v_x = 50.50$, $v_y = -13.84$.

Figures 3.1 - 3.9 show the relative positions of L_1 and L_2 at the consecutive times $t_{160+240k}$, $k = 0,1,2,3,\ldots,8$. The particles of L_1 are labeled A while those of L_2 are labeled B. Figure 3.9 shows the complete interchange of the relative positions of L_1 and L_2, so that the "heavier" fluid has settled to the bottom. Thereafter, until t_{3040}, all the particles continue to be in motion, but at least three from L_1 always remain at the bottom. Examples with $N = 16$ were not run past t_{3040} in order to save computer time for larger values of N.

Example 2. Consider a 256 particle configuration with $n = 64$, $N = 256$, $m(L_1) = 1$, $m(L_2) = 0.25$, $\Delta x = 6.25$, $\Delta t = 10^{-2}$, $d_j \equiv d = 5$, $\xi = 0.1$, $p = \beta = 2$, $\omega = 0.9$, $V = 500$. The computation of acceleration components (2.11) and (2.12) was simplified by assuming that each particle was acted upon only by "nearby" particles. This was implimented by defining α as follows:

$$\alpha = \begin{cases} 0\ , & \text{if } r_{ij,k} \geq 2d = 10 \\ 1\ , & \text{if } r_{ij,k} < 2d = 10\ . \end{cases}$$

The initial positions of the particles were fixed at the points $(3.125 + 6.25u_1,\ 3.125 + 6.25u_2)$, $u_1 = 0,1,2,\ldots,15$, $u_2 = 0,1,2,\ldots,15$.

Figure 3.10 shows the initial interaction between L_1 and L_2 at t_2. The particles of L_1 are labeled A, while those of L_2 are labeled B. An arbitrary boundary has been drawn between L_1 and L_2 to indicate the type of motion in progress. Figure 3.11 shows the state of diffusion at time t_{69} by the setting of circles around the particles of L_1. This diffuse character persisted during the entire calculation. At approximately $t = t_{250}$, the small damping effect incorporated in (2.13) became evident in that the particle velocities had decreased noticeably. Figure 3.12, then, shows not only the state of diffusion at time t_{300}, but also shows the onset of a "thinning" of particles in the upper portion of the region and a "condensation" of particles in the lower portion, due probably to a resultant loss of energy in the system. Figure 3.13 shows the position, and resultant motion, at times t_{10k}, $k = 0,1,2,\ldots,30$, for the particle whose initial position was (53.1, 46.9) and whose initial velocity components, generated, of course, at random, were $v_x = 63.0$, $v_y = 133.9$. The figure not only shows the strong effect of gravity, but, as can be seen from the lower right hand corner, also shows the strong effect of repulsion. Figure 3.14 shows the vector field defined by the directions of the particles at time t_{92}. If one interprets the indicated oval areas, with respect to which all nearby vectors have the same relative orientation, as vortices, then these patterns of vortices are in a constant state of formation and decay, due to the relatively independent motion of each particle. This rapid appearance and disappearance of small vortices is, of course, a fundamental characteristic of turbulent motion.

It should be noted that automatic graphing limitations did not allow for the superposition of letters so that, in Figures 3.10-3.12, when two particles were exceptionally close, only one letter, A or B, was printed. In cases where a choice between A and B had to be made, A was always printed.

Finally, note that, for this example, the total running time up to t_{300}, that is, for 30000 time steps, was only 72 minutes.

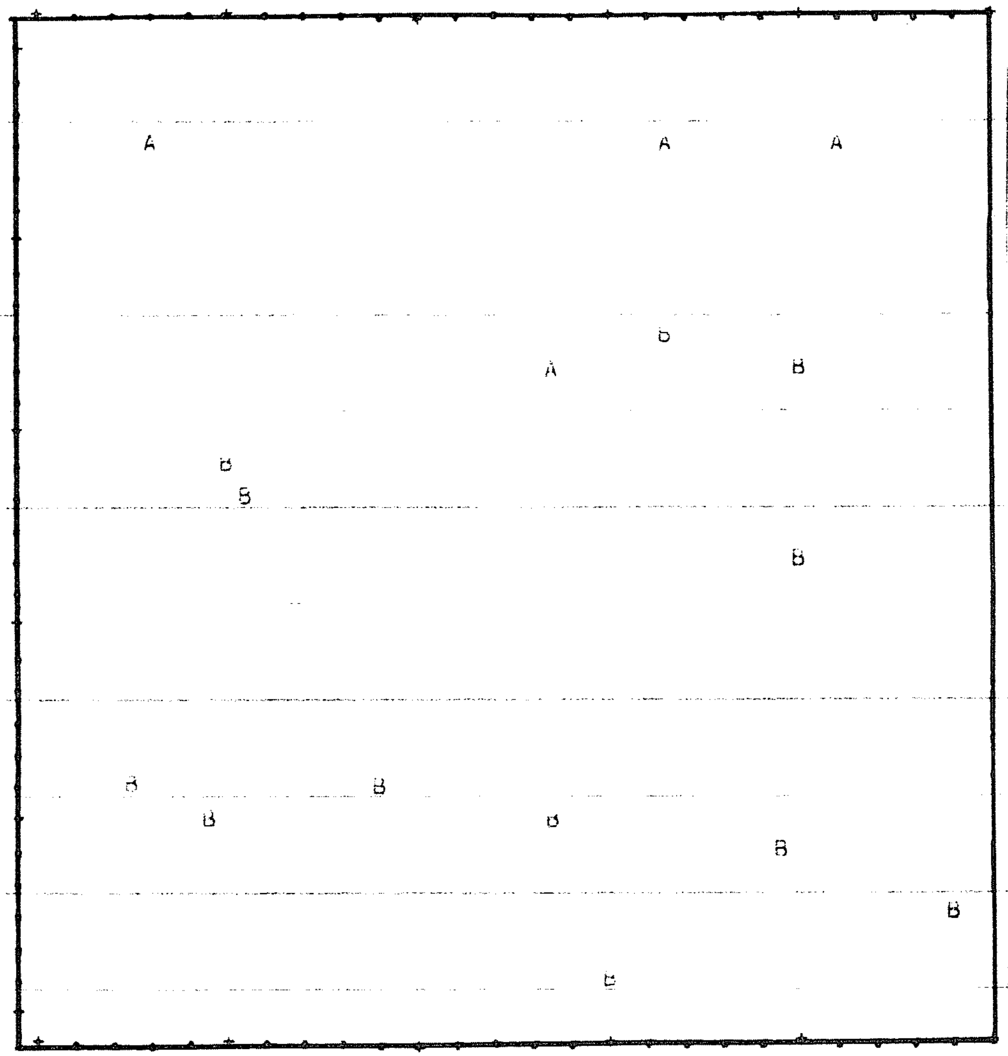

Figure 3.1. $T = t_{160}$

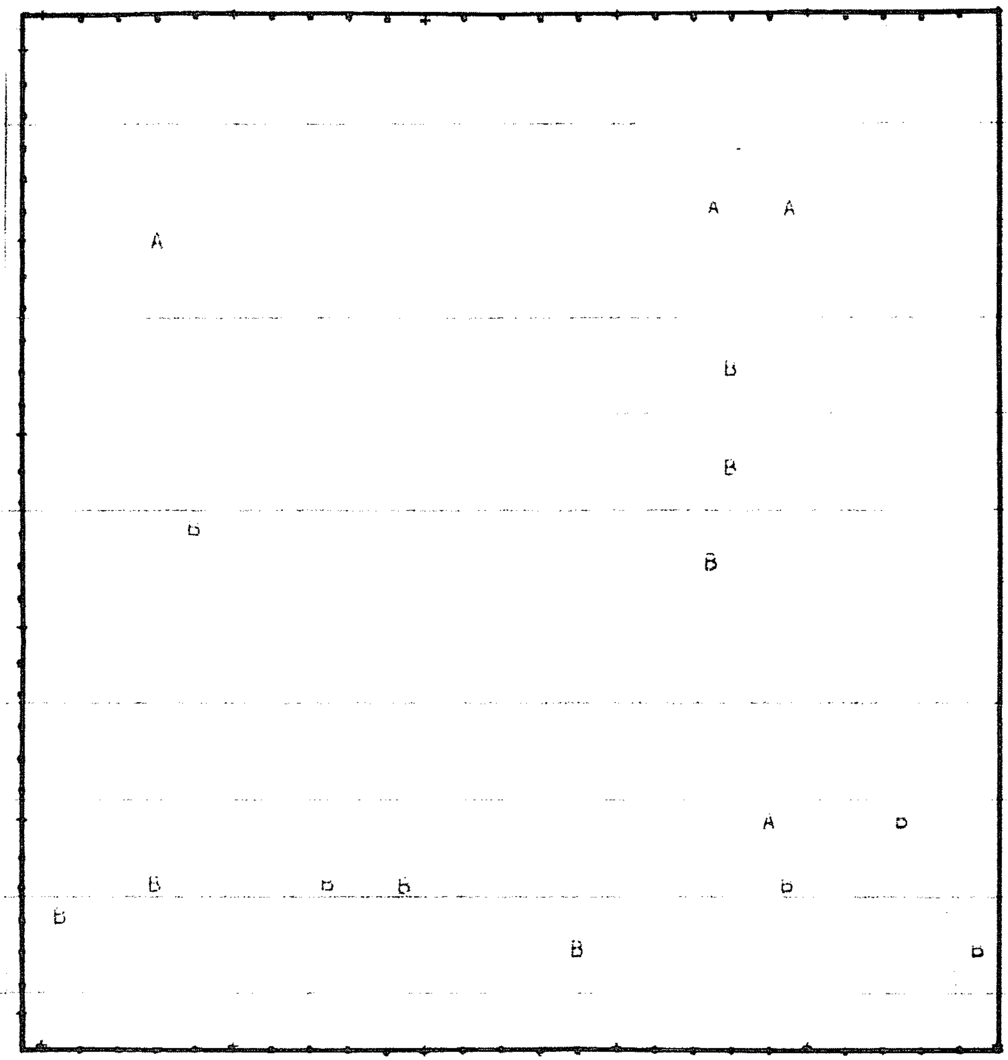

Figure 3.2. $T = t_{400}$

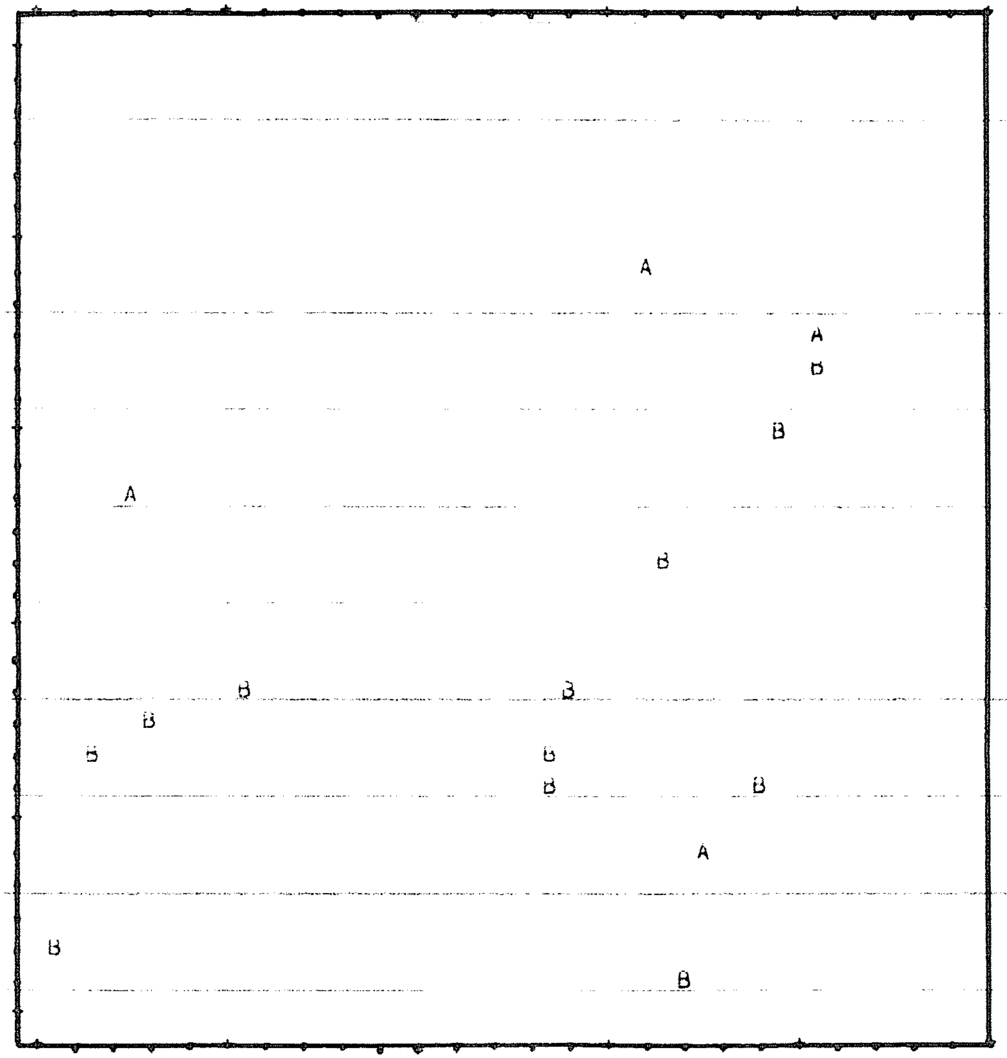

Figure 3.3. $T = t_{640}$

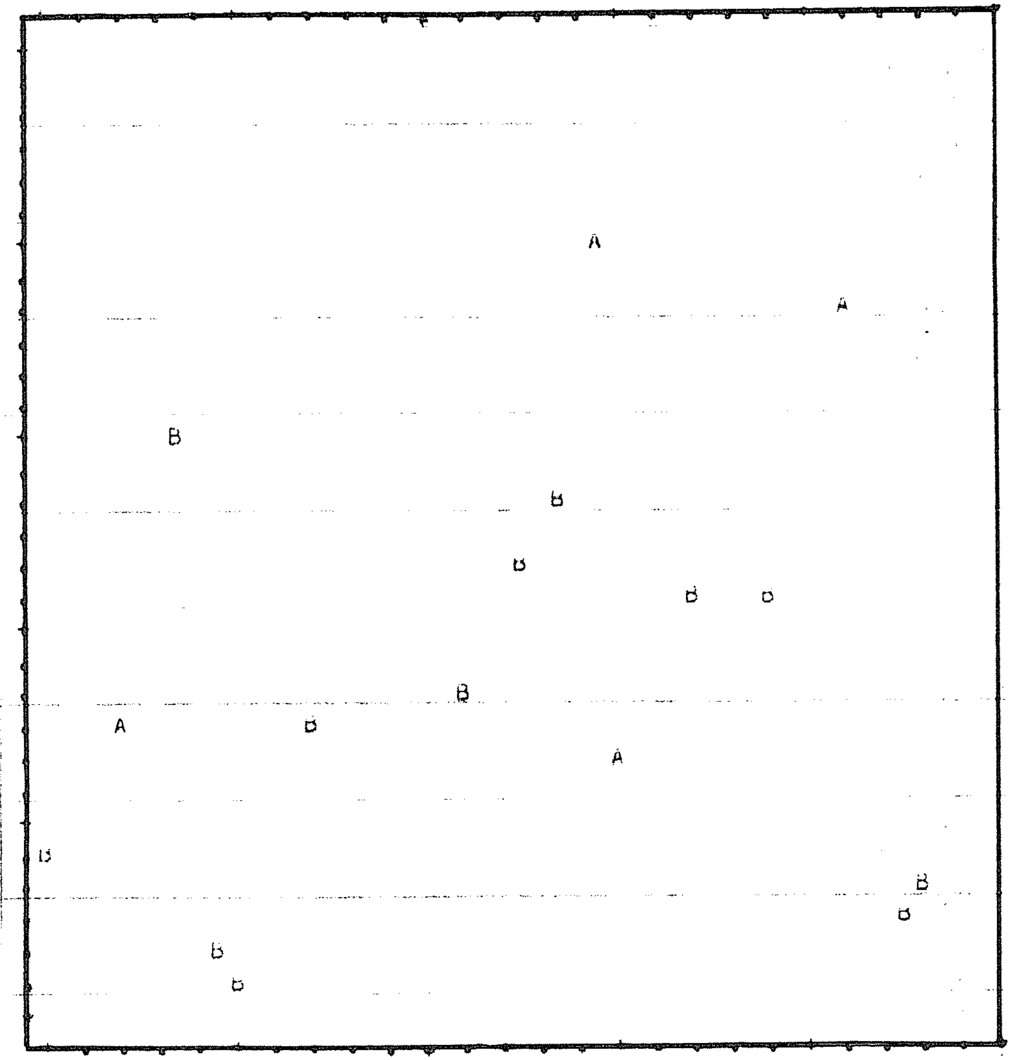

Figure 3.4. $T = t_{880}$

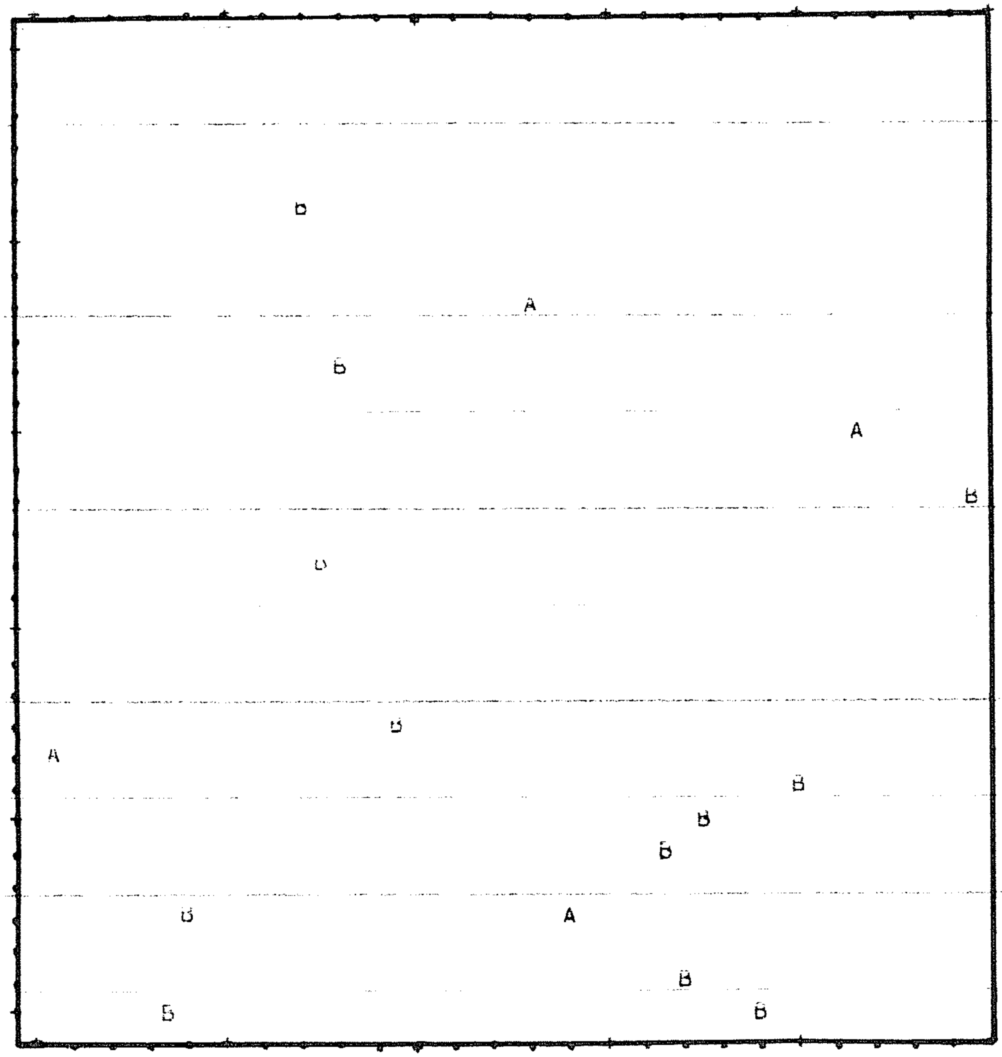

Figure 3.5. $T = t_{1120}$

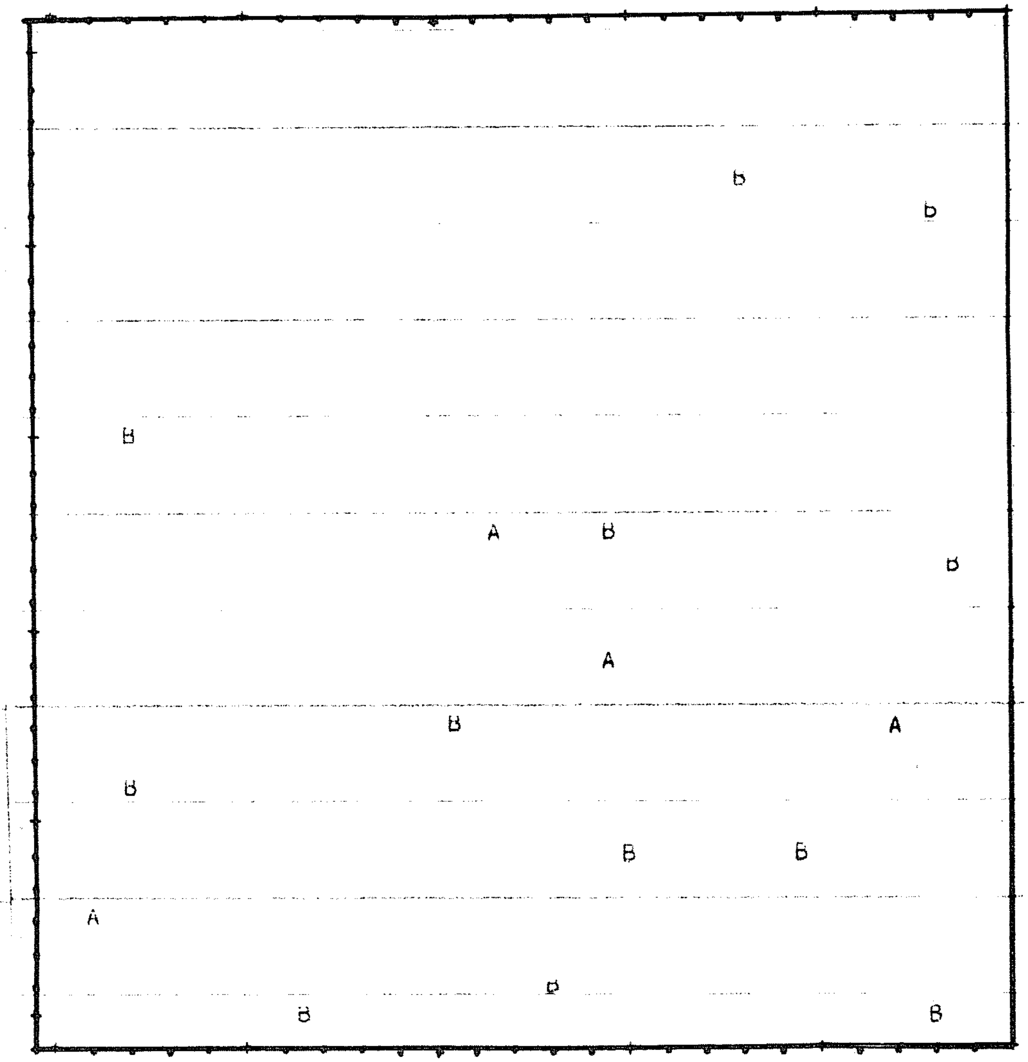

Figure 3.6. $T = t_{1360}$

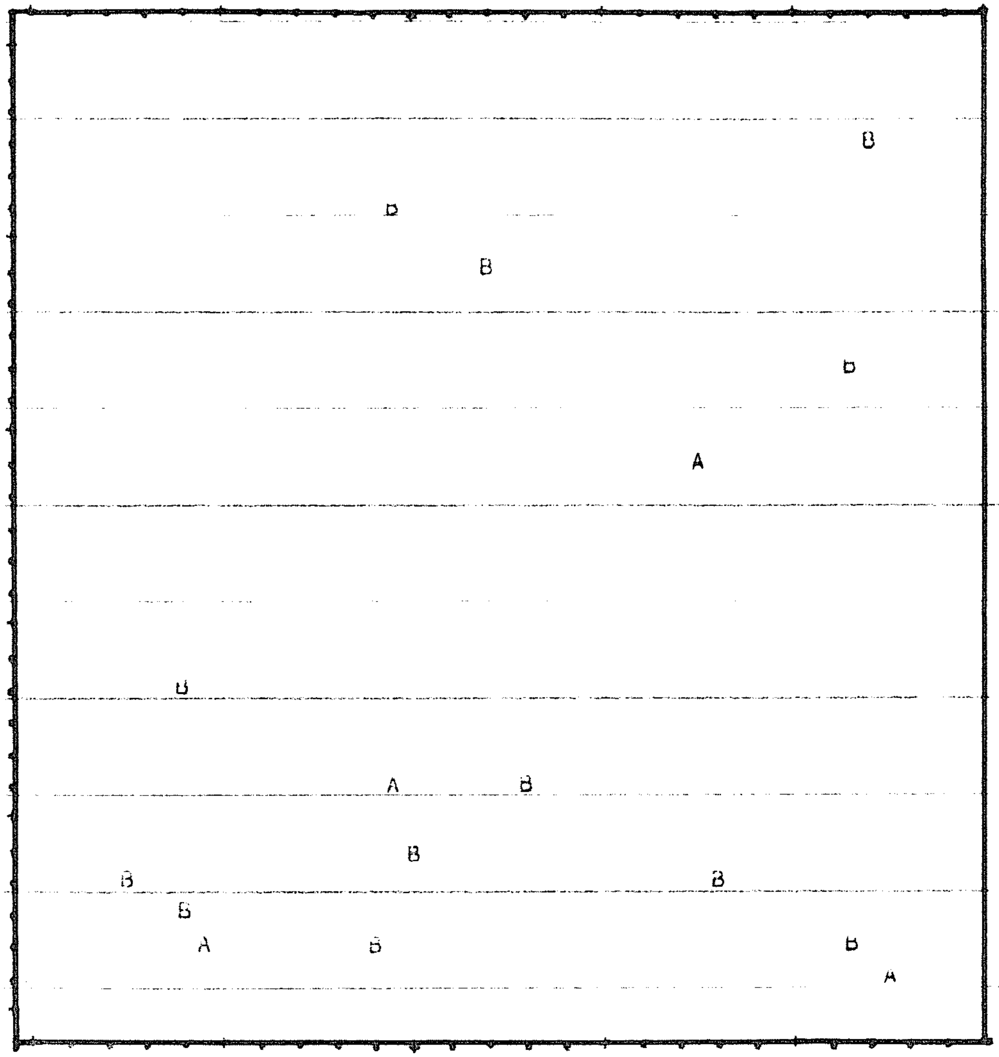

Figure 3.7. $T = t_{1600}$

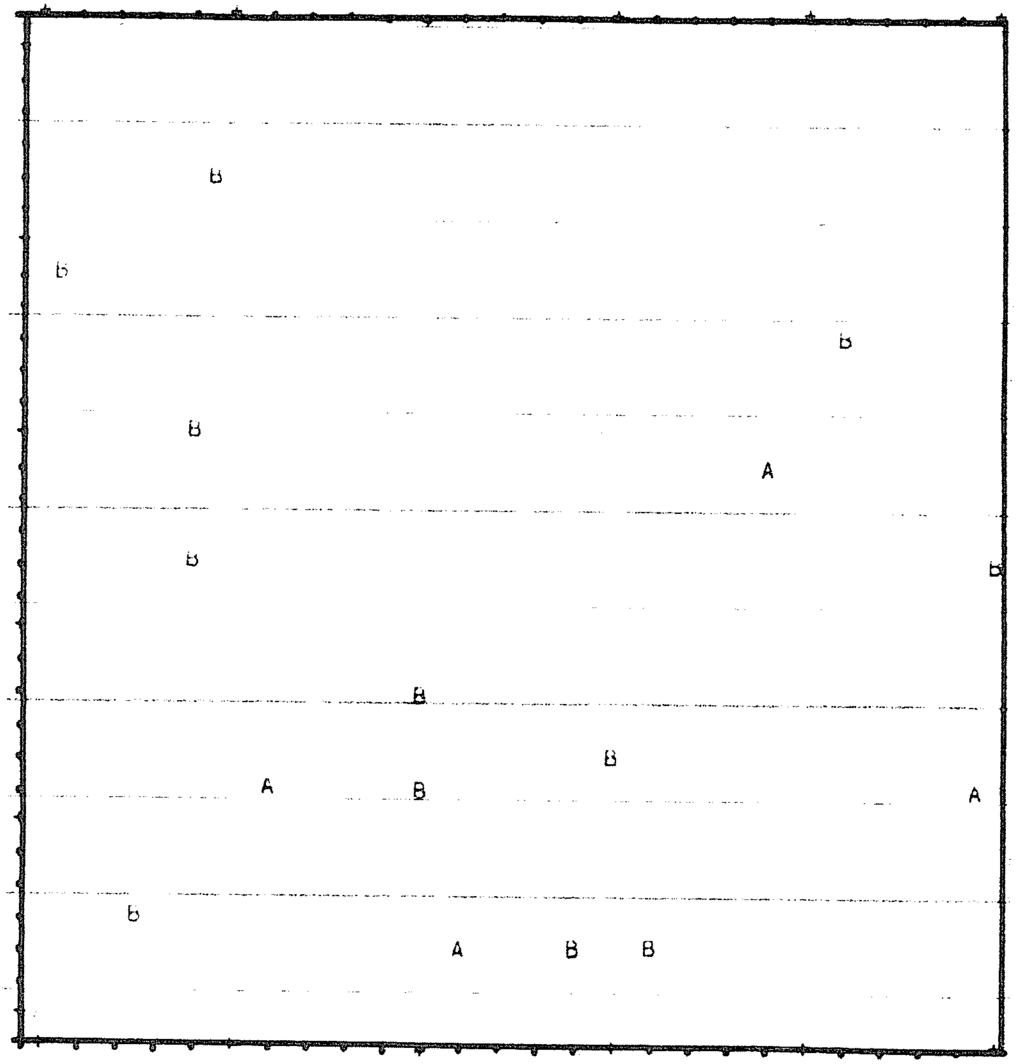

Figure 3.8. $T = t_{1840}$

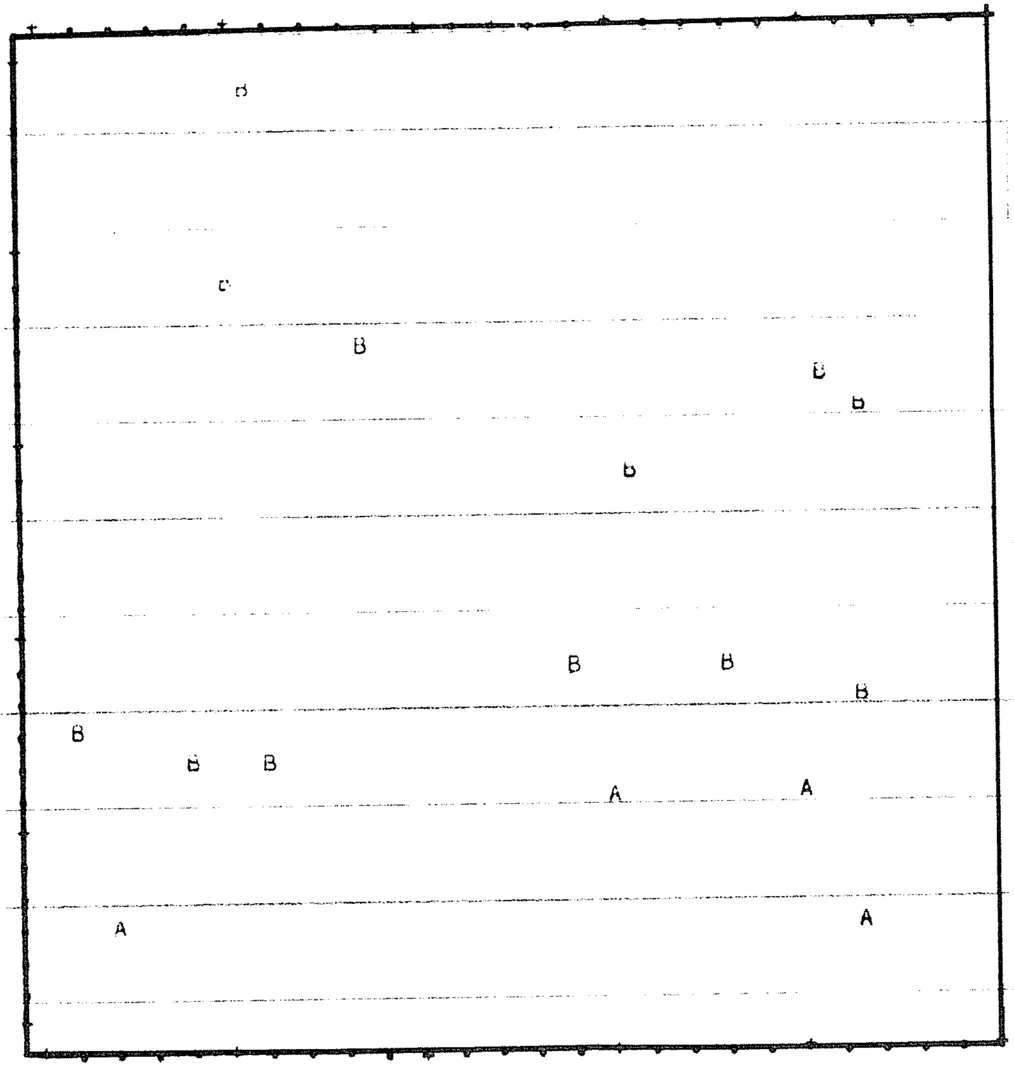

Figure 3.9. $T = t_{2080}$

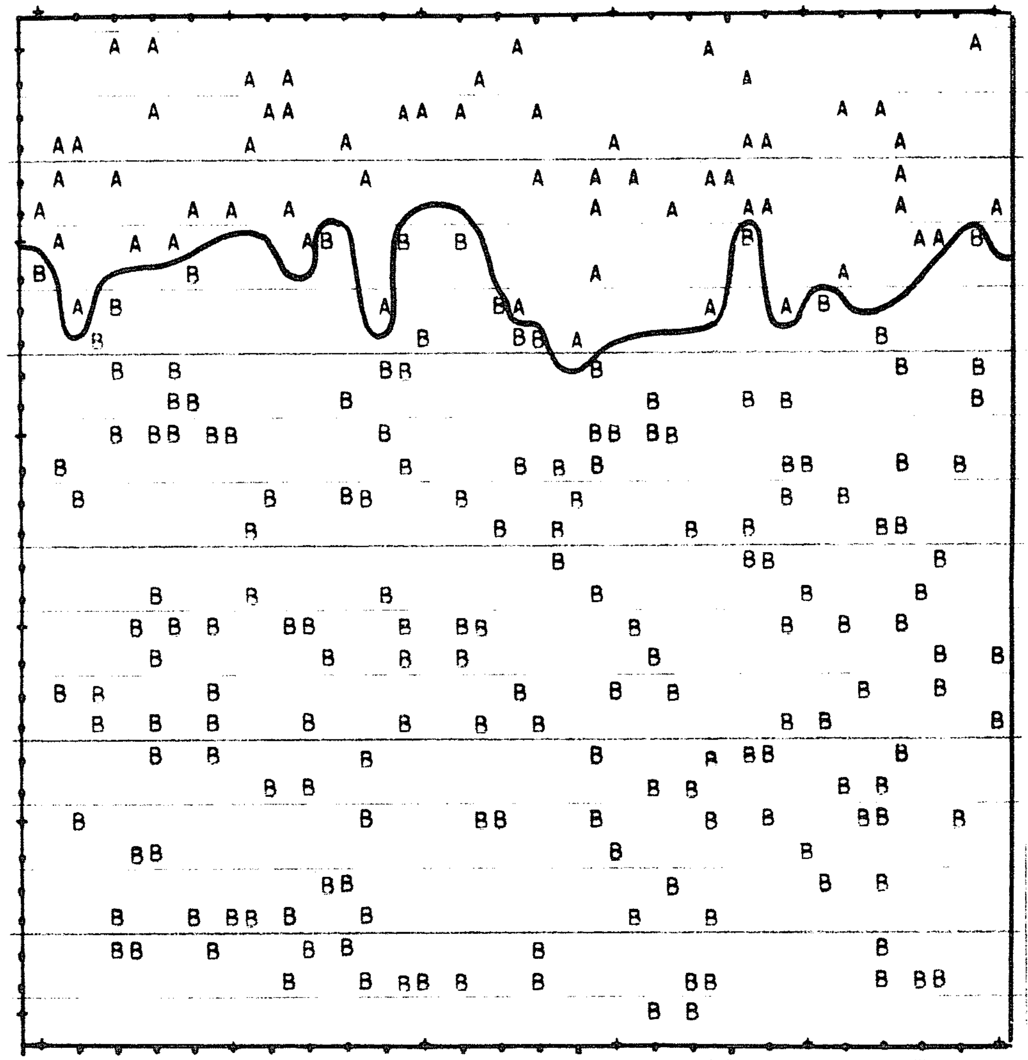

Figure 3.10. $T = t_2$

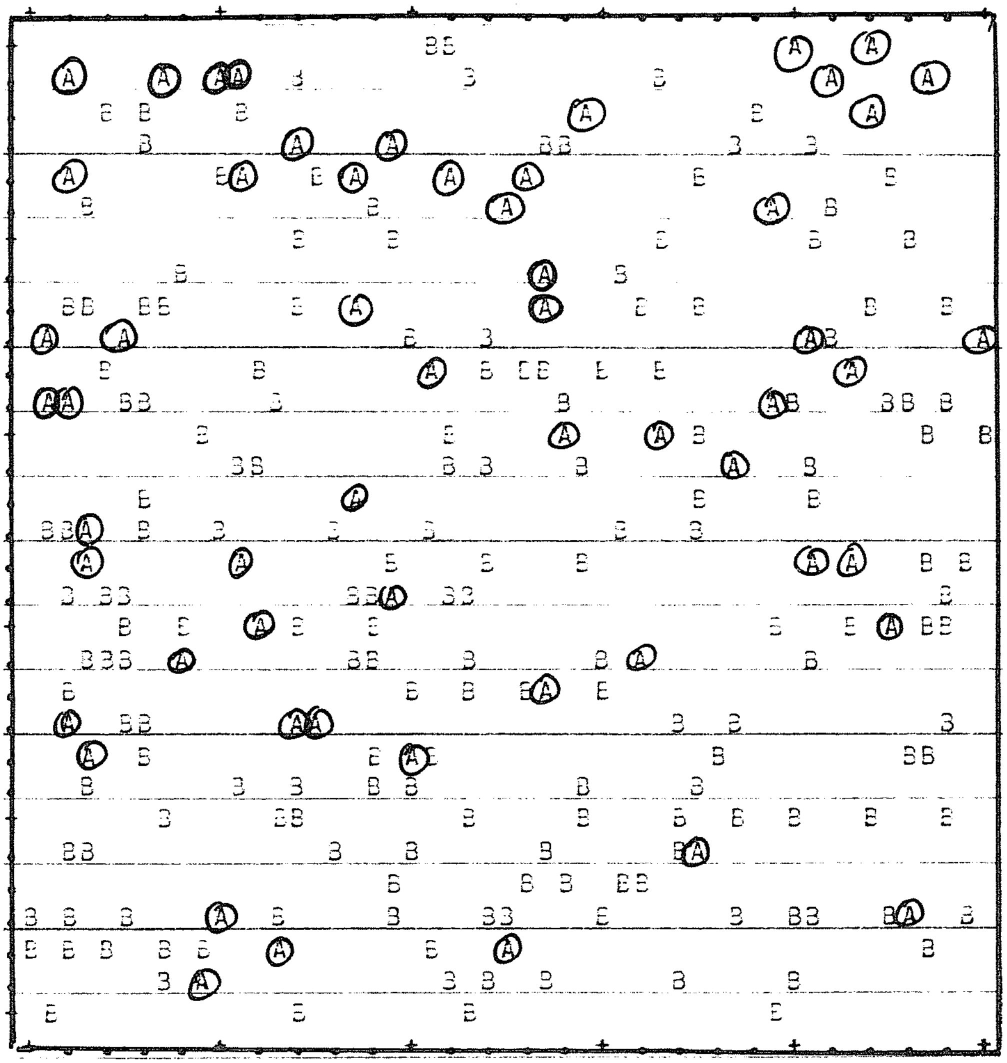

Figure 3.11. $T = t_{69}$

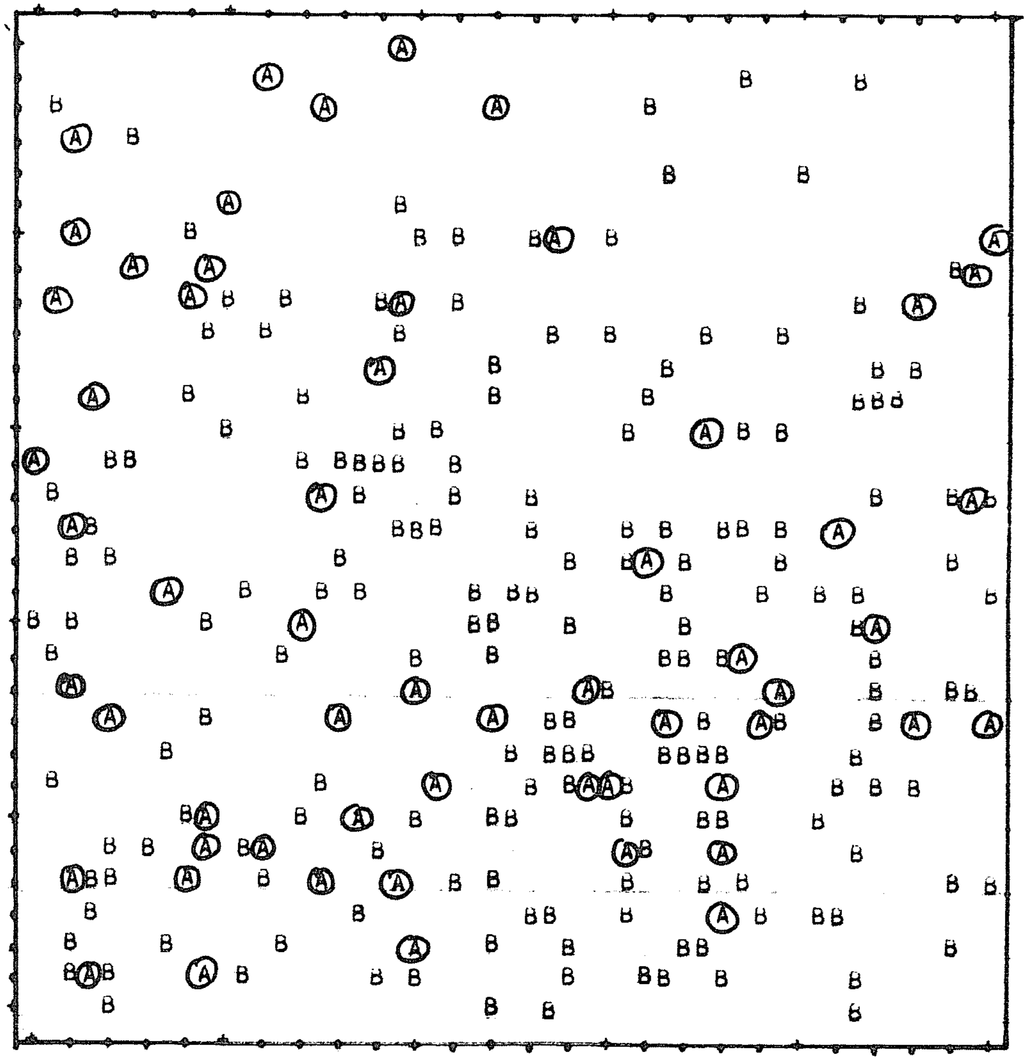

Figure 3.12. $T = t_{300}$

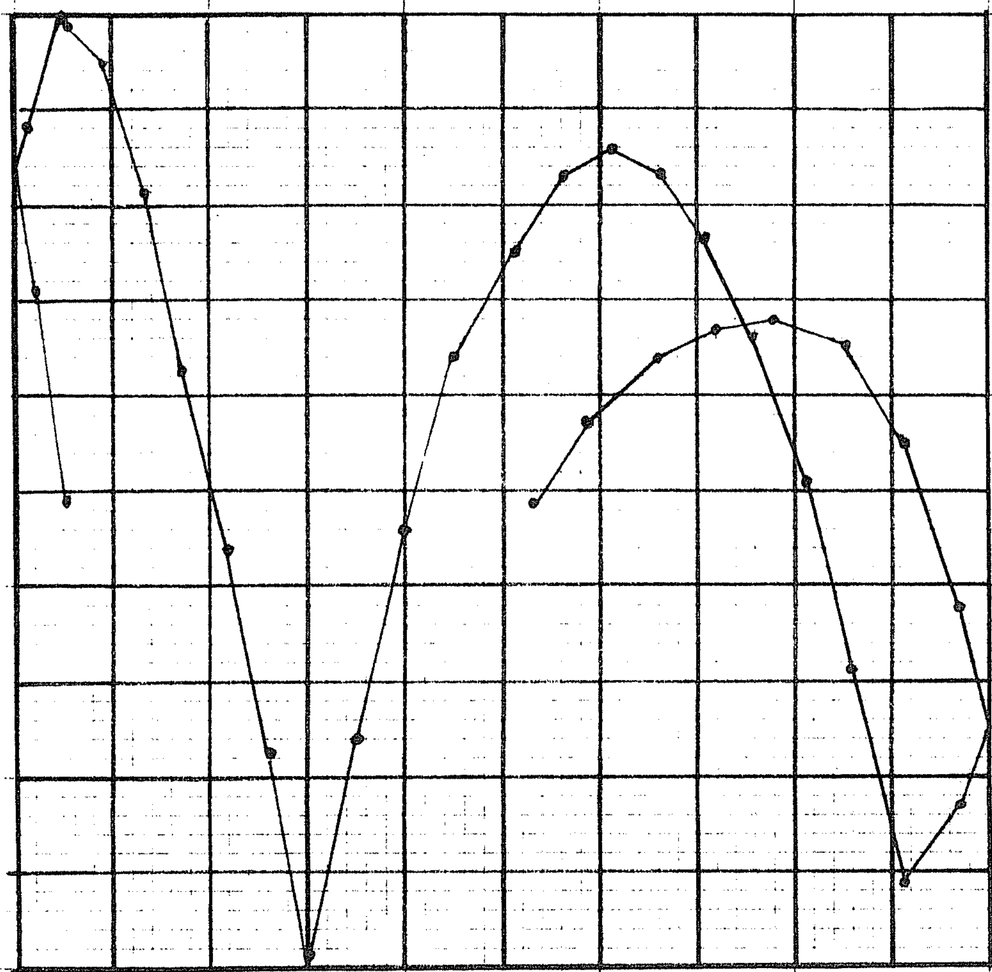

Figure 3.13

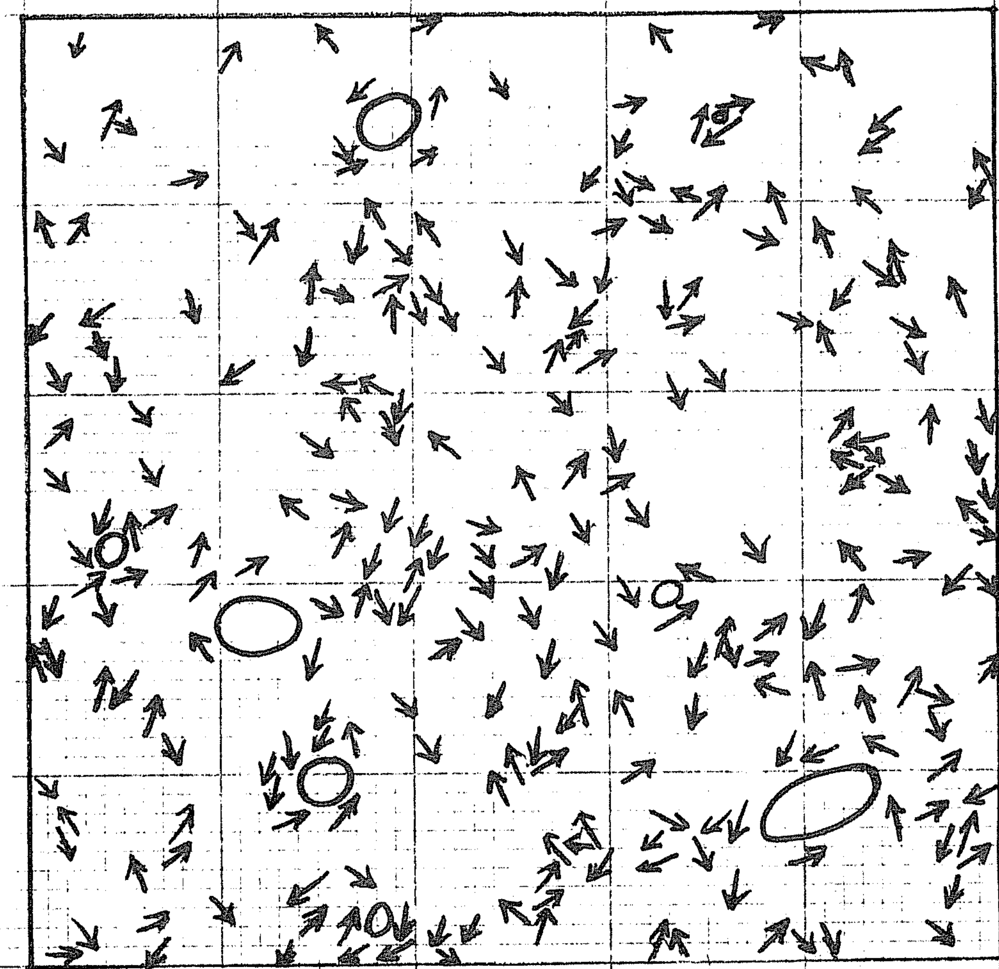

Figure 3.14. $T = t_{92}$

4. Remarks.

Several remarks pertaining to the method and examples of this paper are now in order. Other examples indicated that the character of the fluid motion could be changed in an expected fashion by appropriate variations of the parameters of the equations of motion. Thus, for example, an increase in d invariably led to a decrease in the rate of diffusion, while an increase in β could lead to instability if Δt was not decreased simultaneously. If one wished to choose Δt sufficiently small, then, indeed, one could actually set $d = 1$ and choose p and β so as to agree with the repulsive part of any of the commonly accepted molecular potential functions ([3],[4]). Unfortunately, economic considerations limited the number of particles and running times of the examples which could be explored. Nevertheless, even the relatively simple examples described in Section 3 serve to illustrate the viability of a direct particle approach to the generation and study of turbulence, which is the most common, and yet least understood, type of fluid motion [6]. Finally, because of the experimental nature of the present work, the computer program used is made available in [7], so that every phase of the discussion can be reconstructed by the reader.

References

1. B. J. Alder, "Studies in molecular dynamics III. A mixture of hard spheres," Jour. Chem. Phys., 40, 1964.

2. D. Greenspan, "A discrete numerical approach to fluid dynamics," Proc. IFIPS 71, North-Holland, Amsterdam, 1972.

3. J. O. Hirschfelder, C. F. Curtiss and R. B. Bird, Molecular Theory of Gases and Liquids, Wiley, N. Y., 1954.

4. A. K. MacPherson, "The formulation of shock waves in a dense gas using a molecular dynamics type technique," Jour. Fluid Mech., 45, 1971.

5. J. R. Pasta and S. Ulam, "Heuristic numerical work in some problems of hydrodynamics," MTAC, 13, 1959.

6. P. G. Saffman, "Lectures on homogeneous turbulence," in Topics in Nonlinear Physics, N. J. Zabusky, Ed., Springer, N. Y., 1968.

7. A. B. Schubert, "FORTRAN program for the mixing of fluids," Technical Report #153, Dept. of Comp. Sci., Univ. of Wis., Madison, 1972.

Appendix - FORTRAN Program For The Mixing Of Fluids - A. B. Schubert

```
@RUN SCHUBERT,ETC.

      ASSIGN INPUT AND/OR OUTPUT TAPE FILES TO BE USED THIS RUN.
      TAPES ARE READ AND WRITTEN IN BINARY MODE AT HIGH DENSITY(800BPI).

@ASG,TH    10.,T,0158
@ASG,TH    11.,T,$1320
@REWIND    10.
@REWIND    11.

@FOR,ISZ    ,MIX
C     PARAMETERS TO BE SET FOR EACH RUN

C        N = TOTAL NO. OF PARTICLES IN THE CONTAINER
C     NROW = TOTAL NO. OF ROWS OF PARTICLES IN THE CONTAINER

      PARAMETER N=256,NM1=N-1,NP1=N+1,NH=N*NM1/2,        NROW=16,NRM1=
     * NROW-1,N4P10=4*N+10

      INTEGER P,BETA,OUTPUT,OUTTAF
      REAL MASS(N),ML1,ML2
      EQUIVALENCE(DUMMY,XO)
      DIMENSION XO(N),YO(N),VXO(N),VYO(N),X(N,2),Y(N,2),VX(N,2),VY(N,2),
     * GRAV(N),ACX(N),ACY(N),AM(N),DYK(NROW),R(NH),DUMMY(N4P10)
      DIMENSION AL(N),AR(N),BL(N),BR(N)

C     DEFINITIONS OF DATA VARIABLES

C      EPS = SMALL POSITIVE CONSTANT USED TO AVOID DIVISION BY ZERO
C            AND AS A RESOLUTION VALUE FOR FLOATING POINT UNIQUENESS
C     GCON = MEAN ACCELERATION DUE TO GRAVITY, IN (CM/SEC)/SEC

      DATA EPS,GCON/1.E-3,-980./

C     DEFINITIONS OF X,Y,VX,VY ARRA

C     FOR I=1,2,....,N
C        X(I,1) = X-COMPONENT OF POSITION OF PARTICLE I AT PREVIOUS
C                 TIME STEP
C        X(I,2) = SAME AS ABOVE, EXCEPT AT CURRENT TIME STEP
C       VX(I,1) = X-COMPONENT OF VELOCITY OF PARTICLE I
C       VX(I,2) =   WITH DEFINITION OF SECOND SUBSCRIPT
C                     SIMILAR TO THAT GIVEN FOR X ABOVE
C        Y(I,1) = SAME AS ABOVE EXCEPT FOR
C        Y(I,2) =   Y-COMPONENTS OF
C       VY(I,1) =     POSITION AND VELOCITY
C       VY(I,2) =       OF PARTICLE I

C     READ INFREQUENTLY-VARYING PROBLEM DATA.

C      NL1 = NO. OF PARTICLES COMPRISING LIQUID 1
```

```
C       AX =  LEFT BOUNDARY OF CONTAINER IN X-DIRECTION
C       BX = RIGHT BOUNDARY OF CONTAINER IN X-DIRECTION
C       AY =  LEFT BOUNDARY OF CONTAINER IN Y-DIRECTION
C       BY = RIGHT BOUNDARY OF CONTAINER IN Y-DIRECTION

    3 READ(5,99) NL1,AX,BX,AY,BY

C     COMPUTE NO. OF PARTICLES COMPRISING LIQUID 2.

      NL2=N-NL1

C     COMPUTE CONTAINER SUB-BLOCK DIMENSIONS.

      BLKX=(BX-AX)/FLOAT(NCOL)
      BLKX2=.5*BLKX
      BLKY=(BY-AY)/FLOAT(NROW)
      BLKY2=.5*BLKY

C     COMPUTE Y-COORDINATES OF LOWER BOUNDARY OF EACH ROW OF SUB-BLOCKS.

      DO 4 I=1,NROW
    4 DYK(I)=FLOAT(I-1)*BLKY

      VELNP=-2.*EPS
      NCOL=N/NROW

C     INITIALIZE TAPE I/O VARIABLES.

      NREC=0
      INTAP=0
      OUTTAP=0

C     INPUT DATA DEFINING ONE PROBLEM CASE

C     NMAX = MAX. TIME STEP FOR WHICH POSITIONS AND VELOCITIES ARE TO
C            BE COMPUTED FOR THIS CASE
C    INCPR = TIME STEP INCREMENT FOR PRINTING OF POSITIONS AND
C            VELOCITIES
C   INCPLT = TIME STEP INCREMENT FOR PLOTTING OF PARTICLE POSITIONS
C    INPUT = CONTROL VARIABLE FOR SOURCE OF INITIAL DATA
C                INPUT = 0 IMPLIES TAKE INITIAL POSITIONS TO BE UNIFORMLY
C                          SPACED IN THE CONTAINER AND GENERATE INITIAL
C                          VELOCITIES RANDOMLY
C                INPUT = 1 IMPLIES TAPE INPUT. READ PROBLEM-DEFINING
C                          PARAMETERS AND INITIAL POSITIONS AND
C                          VELOCITIES FROM TAPE (FILE 10) IN BINARY MODE
C               INPUT.GT.1 IMPLIES CARD INPUT. READ PROBLEM-DEFINING
C                          PARAMETERS AND INITIAL POSITIONS AND
C                          VELOCITIES FROM CARDS IN THE FORMAT (3I5,8E8.3
C                          /(8E10.5))
C   OUTPUT = VARIABLE CONTROLLING DISPOSITION OF FINAL DATA FOR THIS
C            CASE
C             OUTPUT = 0 IMPLIES DON'T SAVE FINAL DATA FOR THIS CASE
C             OUTPUT = 1 IMPLIES TAPE OUTPUT. WRITE PARAMETERS
```

```
C                         DEFINING THIS CASE AND FINAL POSITIONS AND
C                         VELOCITIES TO TAPE IN BINARY MODE
C              OUTPUT.GT.1 IMPLIES CARD OUTPUT. PUNCH PARAMETERS DEFINING
C                         THIS CASE AND FINAL POSITIONS AND VELOCITIES
C                         ON CARDS IN THE FORMAT (3I5,8E8.3/(8E10.5))
C         NTO = STARTING TIME STEP FOR THIS CASE THIS RUN
C      P,BETA = EXPONENTS IN REPULSION LAW (SEE TEXT OF REPORT)
C          DT = TIME STEP
C       ALPHA = COEFFICIENT OF REPULSION
C        DAMP = VELOCITY DAMPING FACTOR IN WALL COLLISIONS
C               DAMP = 0 IMPLIES TOTAL DAMPING
C               DAMP = 1 IMPLIES    NO DAMPING
C        VELN = MAXIMUM NORM OF ABSOLUTE VALUE OF INITIAL VELOCITIES
C               WHICH ARE GENERATED RANDOMLY
C         ML1 = MASS OF EACH PARTICLE IN LIQUID 1
C         ML2 = MASS OF EACH PARTICLE IN LIQUID 2
C          XI = CONSTANT ADDED TO DISTANCE BETWEEN PARTICLES IN REPULSION
C               LAW TO PREVENT ZERO DISTANCE BETWEEN PARTICLES

    5 READ(5,38,END=40) NMAX,INCPR,INCPLT,INPUT,OUTPUT,NTO,P,BETA,DT,
     * ALPHA,DAMP,VELN,ML1,ML2,XI,DIAM
      IF(OUTPUT.EQ.1) OUTTAP=1

C     GET INITIAL POSITIONS AND VELOCITIES.

      CALL INITAL
      WRITE(6,94) NL1,NL2,ML1,ML2,DIAM,AX,BX,AY,BY
      WRITE(6,93) DT,DAMP,VELN,ALPHA,P,BETA,XI

C     TEST FOR FAILURE TO FIND INPUT DATA ON TAPE IF EXPECTED TO BE
C     FOUND THERE.

      IF(IFLAG.EQ.0) GO TO 1005
      WRITE(6,87)
      GO TO 5

C     SET MASSES OF PARTICLES IN THE TWO LIQUIDS.  THE FIRST NL1
C     PARTICLES ARE ASSUMED TO COMPRISE LIQUID 1.

 1005 DO 105 I=1,N
      IF(I.LE.NL1) MASS(I)=ML1
      IF(I.GT.NL1) MASS(I)=ML2
      AM(I)=ALPHA*MASS(I)
  105 CONTINUE
      RAD=.5*DIAM
      DIAM2=2.*DIAM
      DT2=.5*DT
      NT=NTO

C     TRANSFER INITIAL DATA INTO WORKING ARRAYS.

      DO 8 I=1,N
      X(I,1)=X0(I)
      Y(I,1)=Y0(I)
```

```
      VX(I,1)=VXO(I)
    8 VY(I,1)=VYO(I)
C     PRINT OUT INITIAL PARTICLE POSITIONS.
      CALL PRINT(1)
C     COMPUTE INITIAL ACCELERATION,VELOCITIES AT 'TIME STEP 1/2'
C     AND POSITIONS AT TIME STEP 1.
      NT=NT+1
      CALL ACCEL
      DO 9 J=1,N
      VX(J,2)=VX(J,1)+DT2*ACX(J)
      VY(J,2)=VY(J,1)+DT2*ACY(J)
      X(J,2)=X(J,1)+DT*VX(J,2)
    9 Y(J,2)=Y(J,1)+DT*VY(J,2)
C     ADJUST PARTICLE POSITIONS AND VELOCITIES AT FIRST TIME STEP
C     IN CASE OF COLLISIONS WITH CONTAINER WALLS.
      CALL WALCOL
C     TEST FOR PRINTING AND PLOTTING OF POSITIONS AND VELOCITIES AT
C     FIRST TIME STEP.
      IF(INCPR.EQ.1) CALL PRINT(2)
      IF(INCPLT.EQ.1) CALL PLOT
C     BEGIN TIME STEP LOOP.
C     UPDATE TIME STEP COUNTER AND PARTICLE POSITIONS AND VELOCITIES
C     FOR PREVIOUS TIME STEP.
   10 NT=NT+1
      DO 12 I=1,N
      X(I,1)=X(I,2)
      Y(I,1)=Y(I,2)
      VX(I,1)=VX(I,2)
   12 VY(I,1)=VY(I,2)
C     COMPUTE ACCELERATION AT PREVIOUS TIME STEP AND CURRENT VELOCITIES
C     AND POSITIONS.
      CALL ACCEL
      DO 15 J=1,N
      VX(J,2)=VX(J,1)+DT*ACX(J)
      VY(J,2)=VY(J,1)+DT*ACY(J)
      X(J,2)=X(J,1)+DT*VX(J,2)
   15 Y(J,2)=Y(J,1)+DT*VY(J,2)
C     ADJUST POSITIONS AND VELOCITIES OF PARTICLES WHICH HAVE COLLIDED
C     WITH THE CONTAINER WALLS.
```

```
      CALL WALCOL
C     TEST FOR PRINTING OF CURRENT POSITIONS AND VELOCITIES.
      IF(MOD(NT,INCPR).EQ.0) CALL PRINT(2)
C     TEST FOR PLOTTING OF CURRENT POSITIONS AND ALSO TEST TO ENSURE
C     THAT POSITIONS TO BE PLOTTED ARE FIRST PRINTED.
      IF(MOD(NT,INCPLT).EQ.0 .AND. MOD(NT,INCPR).NE.0) CALL PRINT(2)
      IF(MOD(NT,INCPLT).EQ.0) CALL PLOT
C     TEST FOR MAXIMUM TIME STEP FOR THIS DATA CASE.
      IF(NT.LT.NMAX) GO TO 10
C     TEST FOR TAPE OUTPUT OF FINAL POSITIONS AND VELOCITIES FOR THIS
C     CASE.
      IF(OUTPUT.EQ.0) GO TO 5
      IF(OUTPUT.EQ.1) GO TO 20
C     PUNCH FINAL DATA ON CARDS.
      PUNCH 89, NT,P,BETA,DT,ALPHA,DAMP,VELN,ML1,ML2,XI,DIAM,(X(I,2),
     * Y(I,2),VX(I,2),VY(I,2),I=1,N)
      GO TO 5
C     WRITE FINAL DATA OUT TO TAPE IN BINARY MODE.
   20 WRITE(11) NT,P,BETA,DT,ALPHA,DAMP,VELN,ML1,ML2,XI,DIAM,(X(I,2),
     * Y(I,2),VX(I,2),VY(I,2),I=1,N)
C     UPDATE COUNTER OF OUTPUT TAPE RECORDS AND WRITE MESSAGE TO
C     PRINTER INDICATING THAT FINAL DATA FOR THIS CASE WAS WRITTEN
C     OUT TO TAPE.
      NREC=NREC+1
      WRITE(6,88) NREC
      GO TO 5
C     TERMINATION POINT FOR PROGRAM. CONTROL REACHES HERE UPON
C     ATTEMPTING TO READ PAST LAST DATA CARD.
C     IF THERE WAS NO TAPE OUTPUT, TERMINATE EXECUTION.
C     OTHERWISE, WRITE END-OF-FILE ON AND REWIND OUTPUT TAPE.
   40 IF(NREC.EQ.0) STOP
      END FILE 11
      REWIND 11
C     IF THERE WAS TAPE INPUT, REWIND THE INPUT TAPE.
      IF(INTAP.NE.0) REWIND 10
      STOP
```

```
C     FORMAT STATEMENTS FOR MAIN PROGRAM SEGMENT

   99 FORMAT(I5,10E5.0)
   98 FORMAT(8I5, 8E5.5)
   95 FORMAT(1X 8E16.6)
   94 FORMAT(    '1NO. OF PARTICLES--IN L1 =' I4,2X'IN L2 =' I4,2X
     * 'MASS OF PARTICLE--IN L1 =' F6.2 ,2X'IN L2 =' F6.2 ,2X'PARTICLE DI
     *AMETER =' F6.2 /'0CONTAINER BOUNDARIES--X =' F6.1 ,2X'TO' F6.1 ,5X
     * 'Y =' F6.1 ,2X 'TO' F6.1 )
   93 FORMAT('0DT =' F7.5 ,                    4X'DAMPING FACTOR =' F7.3 ,4X
     * 'VELOC. NORM =' F6.1,4X 'ALPHA =' F6.2,4X 'P =' I4,4X 'BETA ='
     * I4,4X 'XI =' F4.1//)
   91 FORMAT(2I5,10E5.0)
   90 FORMAT(1X 10F8.4)
   89 FORMAT(3I5,8E8.3/(8E10.5))
   88 FORMAT('0FINAL POSITIONS AND VELOCITIES FOR THIS CASE WERE OUTPUT
     *TO TAPE AS RECORD NO.' I4/)
   87 FORMAT('0INITIAL DATA FOR THIS CASE NOT FOUND ON TAPE. GO TO NEXT
     *DATA CASE.'/)

C     INTERNAL SUBROUTINE FOR COMPUTING DISTANCES BETWEEN PARTICLES

      SUBROUTINE DIST
      K=0
      DO 5 I=1,NM1
      IP1=I+1
      DO 5 J=IP1,N
      K=K+1
    5 R(K)=SQRT((X(I,1)-X(J,1))**2+(Y(I,1)-Y(J,1))**2)
      RETURN

C     INTERNAL SUBROUTINE FOR COMPUTING ACCELERATION OF PARTICLES.
C     RESULTS STORED IN ACX(J),ACY(J),J=1,...,N.

      SUBROUTINE ACCEL
      CALL DIST
      CALL SUPORT
      DO 1 J=1,N
      ACX(J)=0.
    1 ACY(J)=GRAV(J)
      K=0
      DO 2 J=1,NM1
      JP1=J+1
      DO 2 I=JP1,N
      K=K+1

C     COMPUTE NON-ZERO REPULSION BETWEEN TWO PARTICLES ONLY IF THEY
C     ARE SEPARATED BY A DISTANCE LESS THAN TWO DIAMETERS.

      IF(.NOT.(R(K).LT.DIAM2))GO TO 2
```

```
C     TEST IF TWO PARTICLES ARE SEPARATED BY A DISTANCE LESS THAN ONE
C     DIAMETER, IN WHICH CASE A HIGHER DEGREE OF REPULSION IS ASSUMED
C     IN EFFECT THAN IF SEPARATED BY A DISTANCE GREATER THAN ONE
C     DIAMETER.

      IF(.NOT.(R(K).LT.DIAM)) D=(R(K)+XI)**P
      IF(R(K).LT.DIAM) D=(R(K)+XI)**P*((R(K)+XI)/DIAM)**BETA
      DINV=1./D
      TX=(X(J,1)-X(I,1))*DINV
      TY=(Y(J,1)-Y(I,1))*DINV
      ACX(J)=ACX(J)+AM(I)*TX
      ACX(I)=ACX(I)-AM(J)*TX
      ACY(J)=ACY(J)+AM(I)*TY
      ACY(I)=ACY(I)-AM(J)*TY
    2 CONTINUE
      RETURN

C     THIS ROUTINE COMPUTES THE GRAVITY TERM IN THE FORMULA
C     FOR ACCELERATION IN THE Y-DIRECTION.

      SUBROUTINE SUPORT
      DO 101 J=1,N
      AL(J)=AMAX1(AX,X(J,1)-RAD)
      AR(J)=AMIN1(BX,X(J,1)+RAD)
      BL(J)=AMAX1(AY,Y(J,1)-RAD)
  101 BR(J)=AMIN1(BY,Y(J,1)+RAD)
      DO18 JP=1,N
      XL=AMAX1(AX,X(JP,1)-BLKX2)
      XU=AMIN1(BX,X(JP,1)+BLKX2)
      DO11 K=1,NRM1
      KAY=K
      IF(Y(JP,1).LT.DYK(K+1)) GO TO12
   11 CONTINUE
      KAY=NROW
      GO TO16
   12 IF(KAY.GT.1) GO TO16
      IF(Y(JP,1).GT.DIAM) GO TO13
      GRAV(JP)=0.
      GO TO18
   13 AS=ARINT(JP,XL,XU,AY,Y(JP,1))
      IF(AS.GT.0.) GO TO15
   14 GRAV(JP)=GCON
      GO TO18
   15 GRAV(JP)=GCON*(1.-AS/(Y(JP,1)*(XU-XL)+EPS))
      GO TO18
   16 AA=0.
      DO17 K=2,KAY
      AS=ARINT(JP,XL,XU,DYK(K-1),DYK(K))
      IF(.NOT.(AS.GT.0.)) GO TO14
   17 AA=AA+AS
      GRAV(JP)=GCON*(1.-AA/(DYK(KAY)*(XU-XL)+EPS))
   18 CONTINUE
```

```
      RETURN

C     THIS ROUTINE COMPUTES THE TOTAL AREA OF INTERSECTION OF ALL THE
C     PARTICLES WITH THE SHADOW REGION FOR A SPECIFIED PARTICLE.

      FUNCTION ARINT(JP,X1,X2,Y1,Y2)
      ARINT=0.
      DO 1 I=1,N
      IF(I.EQ.JP) GO TO 1
      IF(AR(I).LT.X1.OR.AL(I).GT.X2.OR.Y(I,1).GT.Y(JP,1)) GO TO 1
      F1=AMIN1(BR(I),Y2)-AMAX1(BL(I),Y1)
      IF(.NOT.(F1.GT.0.)) GO TO 1
      F2=AMIN1(AR(I),X2)-AMAX1(AL(I),X1)
      IF(.NOT.(F2.GT.0.)) GO TO 1
      ARINT=ARINT+F1*F2
    1 CONTINUE
      RETURN

C     THIS ROUTINE MODIFIES THE NEWLY-COMPUTED PARTICLE POSITIONS AND
C     VELOCITIES TO ACCOUNT FOR WALL COLLISIONS BY REFLECTING, WITH
C     DAMPING, FROM THE WALLS ANY PARTICLES WHOSE COMPUTED POSITIONS
C     LIE OUTSIDE THE BOUNDARIES OF THE CONTAINER.

      SUBROUTINE WALCOL
      DO77 J=1,N
   22 CONTINUE
      IF(.NOT.(X(J,2).GT.BX)) GO TO 33
      TANTH=(Y(J,2)-Y(J,1))/(X(J,2)-X(J,1)+SIGN(EPS,X(J,2)-X(J,1)))
      X(J,2)=BX-DAMP*(X(J,2)-BX)
      X(J,1)=BX
      Y(J,1)=Y(J,2)+(X(J,2)-BX)*TANTH
      Y(J,2)=Y(J,1)-DAMP*(X(J,2)-BX)*TANTH
      GO TO 66
   33 IF(.NOT.(X(J,2).LT.AX)) GO TO 44
      TANTH=(Y(J,2)-Y(J,1))/(X(J,2)-X(J,1)+SIGN(EPS,X(J,2)-X(J,1)))
      X(J,2)=AX-DAMP*(X(J,2)-AX)
      X(J,1)=AX
      Y(J,1)=Y(J,2)+(X(J,2)-AX)*TANTH
      Y(J,2)=Y(J,1)-DAMP*(X(J,2)-AX)*TANTH
      GO TO 66
   44 IF(.NOT.(Y(J,2).GT.BY)) GO TO 55
      TANTH=(X(J,2)-X(J,1))/(Y(J,2)-Y(J,1)+SIGN(EPS,Y(J,2)-Y(J,1)))
      Y(J,2)=BY-DAMP*(Y(J,2)-BY)
      Y(J,1)=BY
      X(J,1)=X(J,2)+(Y(J,2)-BY)*TANTH
      X(J,2)=X(J,1)-DAMP*(Y(J,2)-BY)*TANTH
      GO TO 66
   55 IF(.NOT.(Y(J,2).LT.AY)) GO TO 77
      TANTH=(X(J,2)-X(J,1))/(Y(J,2)-Y(J,1)+SIGN(EPS,Y(J,2)-Y(J,1)))
      Y(J,2)=AY-DAMP*(Y(J,2)-AY)
      Y(J,1)=AY
      X(J,1)=X(J,2)+(Y(J,2)-AY)*TANTH
```

```
      X(J,2)=X(J,1)-DAMP*(Y(J,2)-AY)*TANTH
   55 SPEED=DAMP*SQRT(VX(J,2)**2+VY(J,2)**2)
      DISINV=1./(SQRT((X(J,2)-X(J,1))**2+(Y(J,2)-Y(J,1))**2)+EPS)
      VX(J,2)=SPEED*(X(J,2)-X(J,1))*DISINV
      VY(J,2)=SPEED*(Y(J,2)-Y(J,1))*DISINV
      GO TO 22
   77 CONTINUE
      RETURN

C     THIS ROUTINE INPUTS THE PARTICLE MASSES AND COMPUTES OR
C     READS THE INITIAL POSITIONS AND VELOCITIES FROM CARDS OR TAPE.

      SUBROUTINE INITAL
  199 FORMAT(3I5,8E8.3/(8E10.5))
      IFLAG=0

C     TEST FOR POSSIBLE CARD INPUT.

      IF(INPUT.LE.1) GO TO 201
      READ(5,199) NTO,P,BETA,DT,ALPHA,DAMP,VELN,ML1,ML2,XI,DIAM,(XO(I),
     * YO(I),VXO(I),VYO(I),I=1,N)
      RETURN

C     TEST FOR NEW PROBLEM CASE VS. ONE TO BE CONTINUED FROM AN EARLIER
C     RUN.

  201 IF(INTAP.EQ.0) GO TO 11201
10201 IF(INPUT.NE.1) GO TO 3
      GO TO 2201
11201 IF(OUTTAP.EQ.0) GO TO 10201
      INTAP=1
      REWIND 10
      REWIND 11

C     COPY ENTIRE INPUT TAPE FILE OUT TO OUTPUT TAPE.

 1201 READ(10,END=2201) DUMMY
      WRITE(11) DUMMY
      NREC=NREC+1
      GO TO 1201
 2201 REWIND 10
 3201 IF(NTO.EQ.0) GO TO 3

C     SEARCH INPUT TAPE FOR INITIAL DATA FOR PROBLEM CASE TO BE
C     CONTINUED FROM AN EARLIER RUN.

  301 READ(10,END=102) INTO,IP,IBETA,XDT,XALPHA,XDAMP,XVELN,XML1,XML2,
     * XXI,XDIAM,(XO(I),YO(I),VXO(I),VYO(I),I=1,N)
      IF(INTO.EQ.NTO .AND. IP.EQ.P .AND. IBETA.EQ.BETA .AND. ABS(XDT-DT)
     * .LT.DELTA .AND. ABS(XALPHA-ALPHA).LT.DELTA .AND. ABS(XDAMP-DAMP)
     * .LT.DELTA .AND. ABS(XVELN-VELN).LT.DELTA .AND. ABS(XML1-ML1).LT.
     * DELTA .AND. ABS(XML2-ML2).LT.DELTA .AND. ABS(XXI-XI).LT.DELTA
     * .AND. ABS(XDIAM-DIAM).LT.DELTA) GO TO 202
```

```
      GO TO 301
C     SET FLAG INDICATING INITIAL DATA FOR CURRENT PROBLEM CASE WAS
C     NOT FOUND ON INPUT TAPE.
  102 IFLAG=1
  202 REWIND 10
      RETURN
C     IF THIS IS THE START OF A NEW PROBLEM CASE, COMPUTE UNIFORMLY
C     DISTRIBUTED INITIAL POSITIONS AND GENERATE INITIAL VELOCITIES
C     RANDOMLY.
    3 IF(ABS(VELN-VELNP).LT.EPS) RETURN
      VELNP=VELN
      K=0
      DO 5 I=1,NROW
      DO 5 J=1,NCOL
      K=K+1
      XO(K)=BLKX*FLOAT(J-1)+BLKX2
      YO(K)=BLKY*FLOAT(NROW-I)+BLKY2
      VXO(K)=VELN*(2.*RANUN(.5)-1.)
      VYO(K)=VELN*(2.*RANUN(.5)-1.)
    5 CONTINUE
      RETURN
C     THIS ROUTINE PROVIDES A PRINTOUT OF PARTICLE POSITIONS AND
C     PARTICLE VELOCITIES AT A SPECIFIED TIME STEP (NT).
      SUBROUTINE PRINT(L)
   99 FORMAT(/1HO I4,16F7.1/5X 16F7.1)
   98 FORMAT( 1HO 4X 16F7.1/5X 16F7.1)
   97 FORMAT(1H1 'VELOCITIES')
      K1=1
      K2=NCOL
      WRITE(6,99) NT,(Y(I,L),I=K1,K2),(X(I,L),I=K1,K2)
      DO 5 J=1,NRM1
      K1=K1+NCOL
      K2=K2+NCOL
    5 WRITE(6,98) (Y(I,L),I=K1,K2),(X(I,L),I=K1,K2)
      WRITE(6,97)
      K1=1-NCOL
      K2=0
      DO 10 J=1,NROW
      K1=K1+NCOL
      K2=K2+NCOL
   10 WRITE(6,98) (VY(I,L),I=K1,K2),(VX(I,L),I=K1,K2)
      RETURN
C     THIS ROUTINE PROVIDES A PRINTER PLOT OF THE POSITIONS
C     OF THE PARTICLES, WITH DIFFERENTIATION BETWEEN PARTICLES
C     COMPRISING LIQUID 1 AND THOSE COMPRISING LIQUID 2.
```

```
      SUBROUTINE PLOT
      DIMENSION TITLE(8),XTITLE(8),YTITLE(8),YPR(N)
      DATA  (TITLE(I),I=1,8)/'A INDICATES LIQUID 1     B INDICATES LIQUI
     *D 2 ..'/,XTITLE(1),YTITLE(1)/6HY..    ,6HX..    /
      DO 1 I=1,N
    1 YPR(I)=-Y(I,2)
      CALL GRPH2N(YPR    ,'R',X(1,2),'R',-NL1,'5X5',-100.,20.,0.,20.,
     * TITLE,XTITLE,YTITLE,'A')
      CALL GRPH2V(YPR(NL1+1),'R',X(NL1+1,2),'R',-NL2,'SAME','B')
      CALL GRPHND
      RETURN
      END


@XQT
   64    0. 100.    0. 100.
   60     2     2     1     1     0     2     2  .01    1.    1. 500.    2.  .25   .1  1


      FINAL REWINDING OF INPUT AND OUTPUT TAPES AND RELEASING OF UNITS.

@REWIND   10.
@FREE   10.
@REWIND  11.
@FREE   11.

@FIN
```