# An Investigation of Galaxy Clustering Using an Asymptotically Fast N-body Algorithm

Andrew W. Appel

Princeton University

April 24, 1981

The orbit of any one planet depends on the combined motion of all the planets, not to mention the actions of all these on each other. To consider simultaneously all these causes of motion and to define these motions by exact laws allowing of convenient calculation exceeds, unless I am mistaken, the force of the entire human intellect.

Isaac Newton

I would like to thank JCI Data Processing, Inc., for their generous grant of computer time, and Howard Stein in particular for the time he spent running the program.

I would also like to thank my advisor, James Peebles, for his patience and encouragement, and for showing me how to play the game.

## Abstract

It may be that the large clusters and superclusters of galaxies observed today in the universe formed from a randomly distributed arrangement of galaxies in a matter-dominated universe. If this is the case, then the details of their clustering may be investigated numerically in an N-body simulation of gravitational forces.

Current algorithms for N-body integrations require an amount of computation time proportional to the square of the number of bodies, due to the need to compare each body with every other. However, using the fact that the gravitational force felt from a distant clump of bodies is well approximated by considering it as a single object, located at the center of mass of the bodies and having a mass equal to their total mass, the number of calculations may be greatly reduced. An improved algorithm is described which requires computation time proportional to N×log(N) for N bodies.

This algorithm is then used, with suitable parameters, to investigate the behavior of randomly placed galaxies in the Friedman universe.

## Introduction

A reasonable hypothesis for the form of the early universe is that galaxies were originally (shortly after their formation) randomly distributed throughout an isotropic, expanding universe. One reason that this model is attractive is that it proposes conditions very similar to those observed today, differing quantitatively but not qualitatively. Today, though, galaxies are not distributed randomly: on very large scales there is a homogenous distribution of mass, but on a smaller scale, clusters containing thousands of galaxies are observed. These clusters are far larger than would occur by random fluctuations.

It could be, however, that clumps could form spontaneously from a random distribution of galaxies. By random fluctuation, there will be areas where the distribution of mass is slightly denser than in neighboring areas. As the universe expands, these areas will tend to hold together because of their strong internal gravitational forces, and weak external forces. Clusters observed today would have been the same size earlier in the development of the universe, after galaxy formation, but they would have been much closer together, in fact nearly touching.

It is thus useful to know under what conditions a randomly arranged set of galaxies will form clumps due to expansion, and what form these clumps will take.

One way of finding out is to perform a massive gedanken experiment: specify some initial conditions -- a few thousand or tens of thousands of randomly placed mass points -- and parameters for the expansion of the universe, and calculate the final positions of the masses after a suitable time interval.

The N-body problem cannot be done in closed form, however, and so the calculation must be done numerically. That is, at each time t, the gravitational forces of each mass on each of the others may be computed by Newton's laws (for a sufficiently small portion of the universe). This is because the metric of the Friedman solution, for a universe of size comparable to the one which exists now, will not change appreciably over distances of a few million light-years. (The size of our universe, although it is not precisely known, is certainly large enough for this to be true.) Furthermore, in the Schwarzschild solutions which are in some sense imbedded into (or superimposed upon) the universe, the various galaxies are far enough apart so that Newton's laws constitute a very good approximation. Thus, using the inverse-square force law, an approximation to the true acceleration and velocity of each particle over a time dt can be computed. By many iterations of this method, the position of each particle after an arbitrary length of time may be found.

## I. An Algorithm to Solve N-body Problems

Obviously, the numerical calculation of thousands of accelerations would become very tedious without the aid of a computer, and conversely, with a computer it is not difficult to compute the acceleration on mass-point i caused by gravitational attraction of all other mass points j:

$$\ddot{\underline{r}}_i = \sum_j \frac{Gm_j(\underline{r}_j - \underline{r}_i)}{|\underline{r}_j - \underline{r}_i|^3}$$

This formula can be implemented straightforwardly in the following procedure, in which $\underline{X}[i]$ is the position vector of galaxy i, M[i] is the mass of galaxy i, and $\underline{A}[i]$ is the computed acceleration vector:

```
FOR i := 1 to N
 DO FOR j := i+1 to N
     DO BEGIN    r := X[j] - X[i];
                 a := G*r/|r|^3 ;
            A[i] := A[i] + M[j]*a;
            A[j] := A[j] - M[i]*a
        END
```

The Straightforward Algorithm

To get a rough idea of how large N should be for an interesting calculation, one can take the ratio of the size of the universe-patch for which the calculation is made, to the typical intergalactic distance. If N galaxies are put into a cube measuring a*a*a, then the galaxies will be separated by approximately $a/N^3$. Thus, to achieve an accuracy of ten percent in measuring the clustering effect, then as a bare mimimum 1000

galaxies should be simulated. (This is not a rigorous argument, but serves to give some indication of how large N must be.)

Given a number N, it is desirable that the simulation of N galaxies should be feasible on a digital computer. It is clear that the statements between the BEGIN-END will be executed N(N-1)/2 times. For large N, the $N^2$ term will dominate to such an extent that it will be sufficient to note that the running time of this procedure is roughly proportional to $N^2$. Furthermore, this term will be the dominant one even when the cost of all other operations on the mass points is considered. Actually moving the points in space, once their acceleration is computed, will take time proportional only to N, the number of points.

When N is large, it becomes very expensive to compute $N^2$ individual accelerations as required by the straightforward algorithm. At a certain point (for N equal to a few thousand) it will take on the order of a minute to compute the gravitational accelerations for only one iteration, even on the fastest computers. To increase N by an order of magnitude would require that either a hundred times more computational resources be obtained, or that computers become faster by two orders of magnitude. It is conceivable that the efficiency of the computer program to compute the steps between the

BEGIN-END could be improved, but it is difficult to see how two orders of magnitude of improvement could be accomplished.

Reducing the Number of Calculations:

For a real improvement in the speed of the algorithm (that is, changing the dominant term of the running time, rather than the constant of proportionality), it will be necessary to execute the BEGIN-END significantly fewer than const*$N^2$ times. This is quite possible, thanks to the wonderful fact that there are no gravitational dipoles.

Consider the following arrangement of masses, where $\underline{y}$ is the center of mass of the masses $m_i$:

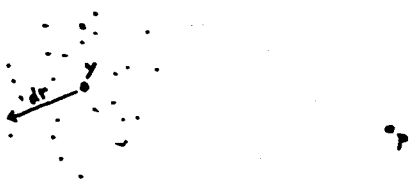



Figure 1: A Clump

We can approximate the strength of the gravitational field at point $\underline{p}$ by $G(\Sigma m_i)/(\underline{y}-\underline{p})^2$, and its direction by $\underline{p}-\underline{y}$. Furthermore, this approximation is accurate to first order in dr/r, since the first term to contribute to the error will be the quadrupole term, with error proportional to $(dr/r)^2$.

The implication of this is that only one calculation (such as that between the BEGIN-END in the straightforward algorithm) needs to be made, no matter how many mass points there are in the cluster about y. What is needed is not a miraculous trick of algebra (other than the simple approximation above), but a way of organizing the calculation to take advantage of the center-of-mass attraction. This will be accomplished by writing down the positions and masses of clumps, just as the positions and masses of indiviual galaxies are written down. A clump will always be defined to contain two subclumps, either of which may be a single galaxy or another clump.

Consider the configuration of galaxies shown in

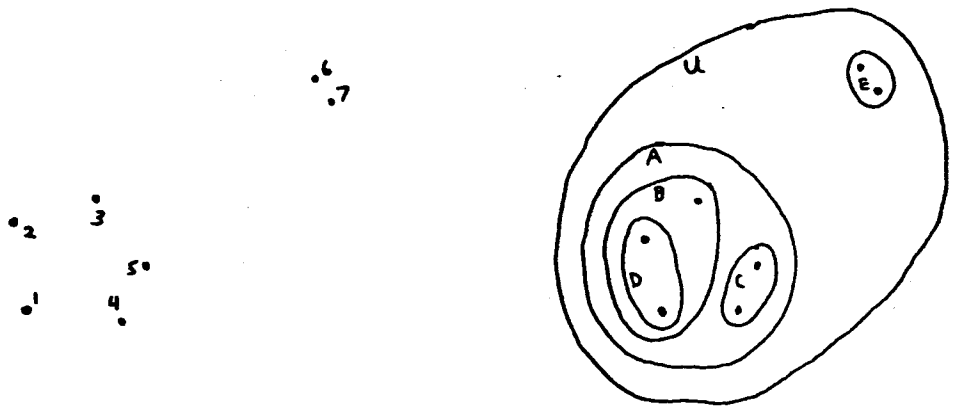



Figure 2: A Clump Structure

figure 2. There is a cluster of five galaxies on the left, and two on the right. These will be represented in clumps as in figure 2b: Clump E contains two galax-

ies, whereas clump A contains two clumps (B and C). Clump B contains one clump and one galaxy. It is also useful to think of the entire picture as one clump U, which contains the two subclumps A and E.

The calculation for figure 2 can now proceed by calculating only once the attraction between the object A (with mass 5) and object E (with mass 2). Then, the internal accelerations of A are computed, and the internal accelerations of E. Furthermore, the internal motion of clump E may be computed in closed form -- since it contains only two bodies -- so that instead of computing an acceleration, a velocity, and a position for each of galaxies 6 and 7 every iteration, a closed-form calculation needs to be computed only on the first iteration. Note that the position of the center of mass of 6 and 7 (that is, the position of clump E) must be computed numerically, every iteration.

A Faster Algorithm:

Assume, for the present, that an arbitrary arrangement of galaxies can be represented as clumps in a natural way. (Note that if the distribution of galaxies is homogenous and not clustered, there will still be portions of the universe which from the point of view of some galaxies will be sufficiently distant relative to their diameter that they may be treated as

clumps.) The procedure for calculating all of the gravitational forces in the universe, while not as simple as the straightforward method, is not much more complicated:

```
Procedure ComputeAccel( B )
   BEGIN    IF B is a nontrivial clump
             THEN BEGIN ComputeAccel(B1);
                        ComputeAccel(B2);
                        TwoNode( B1, B2 )
                  END
   END
```

This procedure's function can be easily expressed in English: To compute all of the accelerations internal to a clump, first compute all the accelerations internal to its first subclump, then compute all the accelerations internal to its second subclump, then compute all the accelerations which involve a galaxy from each of the subclumps. It is clear that of all pairs of galaxies, either both are in the same subclump, in which case their mutual attraction will be calculated in the first or the second line, or they are not both in the same subclump, in which case their attraction will be computed in the third line.

The line "IF B is a nontrivial clump" serves to stop the recursion -- if B is trivial, i.e. a point mass, it has no internal accelerations to be computed.

If ComputeAccel is called to execute upon the clump containing all of the galaxies -- "ComputeAccel(Universe)" -- then the result will be the same as that of invoking the straightforward algorithm.

Finally, the details of the procedure TwoNode, which computes all of the gravitational attractions which involve one member of clump A and one member of

```
Procedure TwoNode( A , B )
  BEGIN d := distance between centers of mass of A and B;
        IF (r_A/d > δ) AND (r_A > r_B)
          THEN BEGIN TwoNode( A_1 , B );
                     TwoNode( A_2 , B )
               END
          ELSE IF r_B/d > δ
                  THEN BEGIN TwoNode( A , B_1 );
                             TwoNode( A , B_2 )
                       END
                  ELSE BEGIN A_A := A_A + G*d*m_B/d^3 ;
                             A_B := A_B - G*d*m_A/d^3
                       END
  END
```

clump B: If the ratios of the sizes of clumps A and B to their separation is smaller than $\delta$, then it is a good approximation to perform a center-of-mass calculation. That is what the last ELSE clause does. Otherwise, it splits up the larger of A and B, for the purposes of this calculation, and calculates all of the attractions between members of the other clump and members of each of the two subclumps.

Suppose, for example, that B is the larger clump (larger in volume, not necessarily in mass), and furthermore that $r_B/d$ is larger than $\delta$. Then the second BEGIN-END clause will be performed. This computes all attractions of galaxies such that one member is in A and the other is in $B_1$. Then it computes all attractions of one member from A and the other from $B_2$ (see figure 3).

This procedure assigns acceleration vectors both to clumps and to individual galaxies (which are special cases of clumps, having zero radius and no subclumps). To find the net acceleration of an individual galaxy, the accelerations of all of its enclosing clumps must be summed. If $\delta$ is set to zero, then the TwoNode Procedure will always recur down to the level of individual galaxies, and the accelerations assigned to the nontrivial clumps will be zero. In this case, the accelerations given the individual galaxies will be exactly equal to the accelerations computed by the straightforward method. If $\delta$ is not equal to zero, then the acceleration of a single galaxy found by summing all of the accelerations of the surrounding clumps (together with its own assigned acceleration) will be an approximation to the true acceleration found by the straightforward algorithm. The smaller $\delta$ is, the better this approximation will be, with each two-clump comparison having a possible error of no more than $\delta^2$.

Speed and Accuracy Considerations:

There is a considerable amount of overhead involved in the calculation. Rather than computing N different accelerations, approximately 2N are actually computed, since the number of nontrivial clumps is N-1. Furthermore, the recursion implicit in procedures Com-

puteAccel and TwoNode is somewhat more costly than the looping of the straightforward algorithm. However, the running time of procedure ComputeAccel is proportional to N*logN. Thus, no matter how much overhead is involved, this changes only the constant of proportionality; for large enough N, ComputeAccel will be much faster than the straightforward method.

To see that the number of calculations made will be roughly proportional to N*logN, consider the number of times a particular galaxy X is compared with other clumps for the purposes of adding to an acceleration vector. Suppose there is a spherical shell around X of radius r and thickness $\delta$*r. If this shell is filled with clumps of diameter $\delta$*r, then there will be $4/\delta^2$ clumps in the shell. The smallest shell will have a size such that one would expect to find one galaxy in it, that is, volume equal to $m/\rho$. The largest shell will enclose volume equal to $Nm/\rho$. The quotient of the radii of the largest and smallest shells will be $N^{1/3}$. This will be equal to $(1+\delta)^k$, where k is the number of shells. Then $k=\log N/3\log(1+\delta)$, and the number of clumps for which there must be calculation of acceleration relative to galaxy X is approximately

$$\frac{4}{3\delta}\frac{\log(N)}{\log(1+\delta)}$$

Note that this number overestimates the number of calculations done, in that some of the calculations

will involve not the comparison of X with another clump, but the comparison with an enclosing clump of X with another clump. That calculation would also be counted in this analysis as a calculation for X's sibling clump, and all other subclumps of the encompassing clump. However, this will do no more than change the constant of proportionality: for each of the N galaxies, const*logN calculations must be done, giving a total execution time proportional to N*logN (when $\delta$ is held constant).

When the universe is heavily clustered, the algorithm should speed up, since there will be fewer, denser clumps; the condition that a clump's size must be a factor of $\delta$ smaller than its distance will occur much more often, saving many recursions of the procedure TwoNode.

The parameter $\delta$ is a measure of the accuracy of the calculation. When one clump is compared with another, and the ratio of diameter to separation is less than $\delta$, then the computed acceleration will have a fractional error less than $\delta^2$. When all of the accelerations that clump X feels from other clumps are summed, the error in acceleration should be proportional to $\delta$ squared divided by the square root of the number of clumps compared with. Furthermore, larger clumps will tend to approach some sort of spherically symme-

tric distribution, simply because of the large number of randomly positioned particles. Thus the error in acceleration, on the average, should be significantly smaller than $\delta^2$.

In fact, the distribution of errors, shown in figure 4, is such that there is a maximum absolute error range, such that for most particles the error is less than $\delta^2$. For particles with large accelerations, the proportional error is practically zero.

Moving the Galaxies:

Once the accelerations of all clumps are computed for any time t, the clumps can be moved according to their new velocities. This process takes time proportional to the number of clumps, or to the number of galaxies, as in the "straightforward" version. In this version, note that the clumps are moved according to the accelerations given them by the ComputeAccel procedure: ComputeAccel leaves acceleration vectors at all levels in the clump hierarchy, and the Move procedure (see figure 9) applies them at the level in which they are found -- they are not propagated down to the lowest level (that of individual galaxies), as this happens automatically.

This is accomplished by having the coordinates of each subclump of clump C written relative to the center

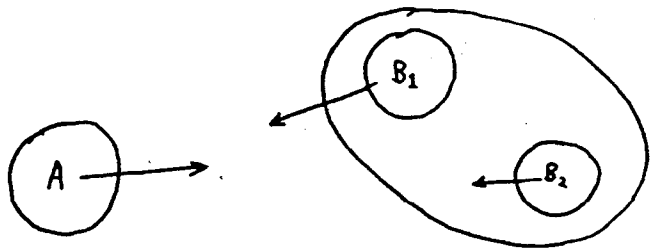



Figure 3: Result of TwoNode when B is large

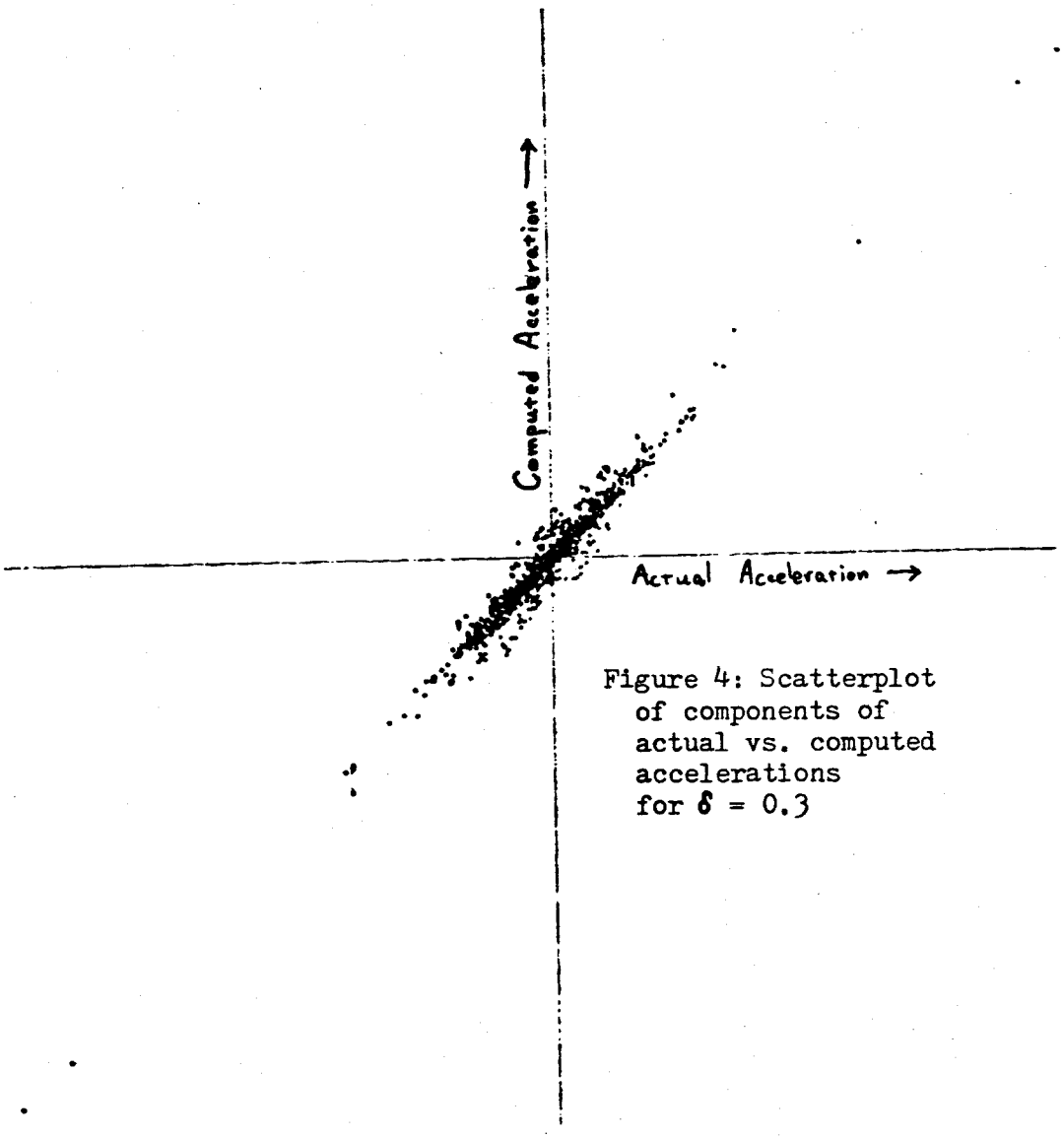



Figure 4: Scatterplot of components of actual vs. computed accelerations for $\delta = 0.3$

of mass of C. The position, velocity, or acceleration vector of $C_1$ -- subclump 1 of clump C -- are stored as the position, velocity, or acceleration as seen by an observer located at the position named $P_C$, having velocity $V_C$, and acceleration $A_C$. Thus when procedure Move changes the velocity and position of C, it de facto changes the position and velocity of all galaxies in C. This in itself does not change the speed of Move by more than a proportionality constant, but it makes many calculations simpler.

Storing coordinates relative to the parent clump does improve the precision of the calculation. Since all computations must be carried out with floating point numbers, which are represented as a k-digit number multiplied by 2 taken to some e-digit number, it is important that no number be required to have a proportional error of less than $2^{-k}$. Suppose, however, that two galaxies are very close to each other and that positions are stored in absolute coordinates. It is quite conceivable that the ratio of the size of the universe to the separation of two galaxies could approach $2^k$. In that case, the two galaxies could not be as close as they are without having the same floating point representation, and the computer would necessarily have to consider them to be at the same point.

In relative coordinates, all of the k digits of the precision can be used to write their position relative to the center of mass. The task of specifying their absolute position in the universe can be left to the position vectors of their enclosing clumps. This way, the absolute error of their center of mass will be the same as before, but the much more important relative position (from which huge accelerations may come, since they are so close) will be specified to many more digits of accuracy. (Note: the procedure TwoNode as defined above does not reflect this detail; this was for purposes of clarity.)

```
Procedure Move( B , dt );
  BEGIN   V_B1 := V_B1 + A_B1 * dt;
          P_B1 := P_B1 + V_B1 * dt;
          IF B1 has subclumps THEN Move(B1);
          V_B2 := V_B2 + A_B2 * dt;
          P_B2 := P_B2 + V_B2 * dt;
          IF B2 has subclumps THEN Move(B2);
          Adjust(B)
  END
```

Figure 9: Details of the Move procedure.

How Big is dt?:

The time increment dt between iterations is determined after each iteration. It is set equal to the minimum over all clumps of the characteristic time of the clumps. The characteristic time for a clump C is the time in which C will move a distance of approxi-

mately $\delta$ times the distance of C from its parent clump's center of mass. This is easy to calculate, since the position vector of C is stored as the vector from (the center of mass of) C's parent. So the characteristic time of C is the smaller of $t_v$ and $t_a$, where

$$\delta \times |P_C| = t_v \times |V_C|$$

$$\delta \times |P_C| = |A_C| \times \tfrac{1}{2} t_a^2$$

In each iteration, the accelerations are computed by ComputeAccel, then the minimum characteristic time dt is found, and then Move calculates the new positions and velocities. Things, unfortunately, are not so simple. Suppose two or three galaxies get into a tight orbit around each other. Their characteristic time may be an order of magnitude shorter than the characteristic time of any other object in the universe. It would be nice to be able to iterate them at shorter time intervals than the rest of the universe, saving a large amount of calculation. This is not too difficult; what is needed, again, is a way to organize this process.

Let such a clump be considered to be one object, indivisible, of nonzero radius. Since any clump is considered as one object -- albeit with internal structure -- already, the first and third qualifications are met trivially. Let indivisibility be defined: a clump is indivisible if for <u>all</u> clumps outside it, its ratio of size to distance is less than $\delta$. What indivisibil-

ity effectively means is that such clumps will never be "split" by procedure TwoNode to calculate accelerations of its subclumps with another clump. This is easy to detect -- simply flag clump A in the first THEN clause or clump B in the second THEN clause of procedure TwoNode. Any clump which is never flagged during the process of computing all the accelerations is indivisible.

Now, procedure Move, procedure ComputeAccel, and the procedure which determines dt will be altered so that they never look at the internal structure of such a clump. Note that TwoNode need not be altered, since the way indivisible clumps are defined is that TwoNode never looks at their internal structure. Now the problem is gone: the small, tight cluster of galaxies has become a point (although with radius!). The time increment dt will be much larger than it could have been otherwise. At the end of the iteration, the internal mechanics of the point can be resolved. This is done relative to the center of mass of the point, which will have been moved, even accelerated -- but all motion and acceleration from external sources will act equally on all the masses within, just due to the restrictions placed on indivisibility. This resolution will usually take several iterations; these iterations involving three or four galaxies are in place of unnecessary iterations which would have involved thousands.

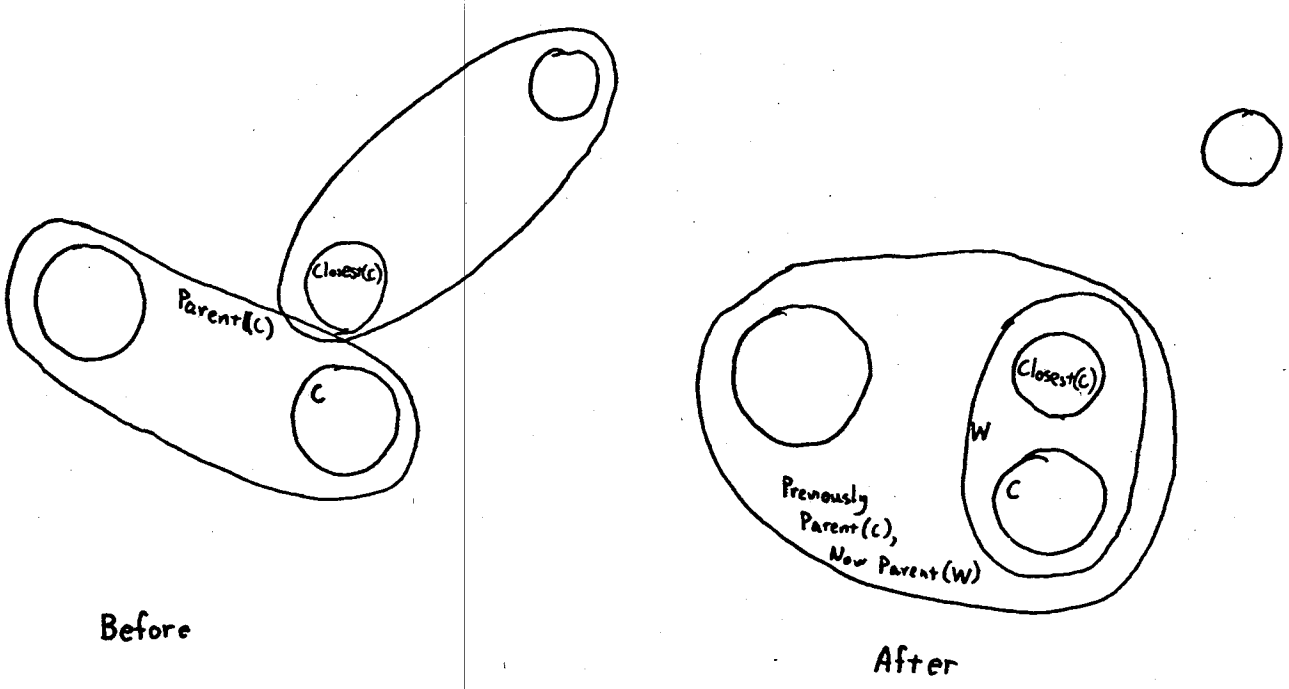



Figure 6: Rearrangement of Clumps

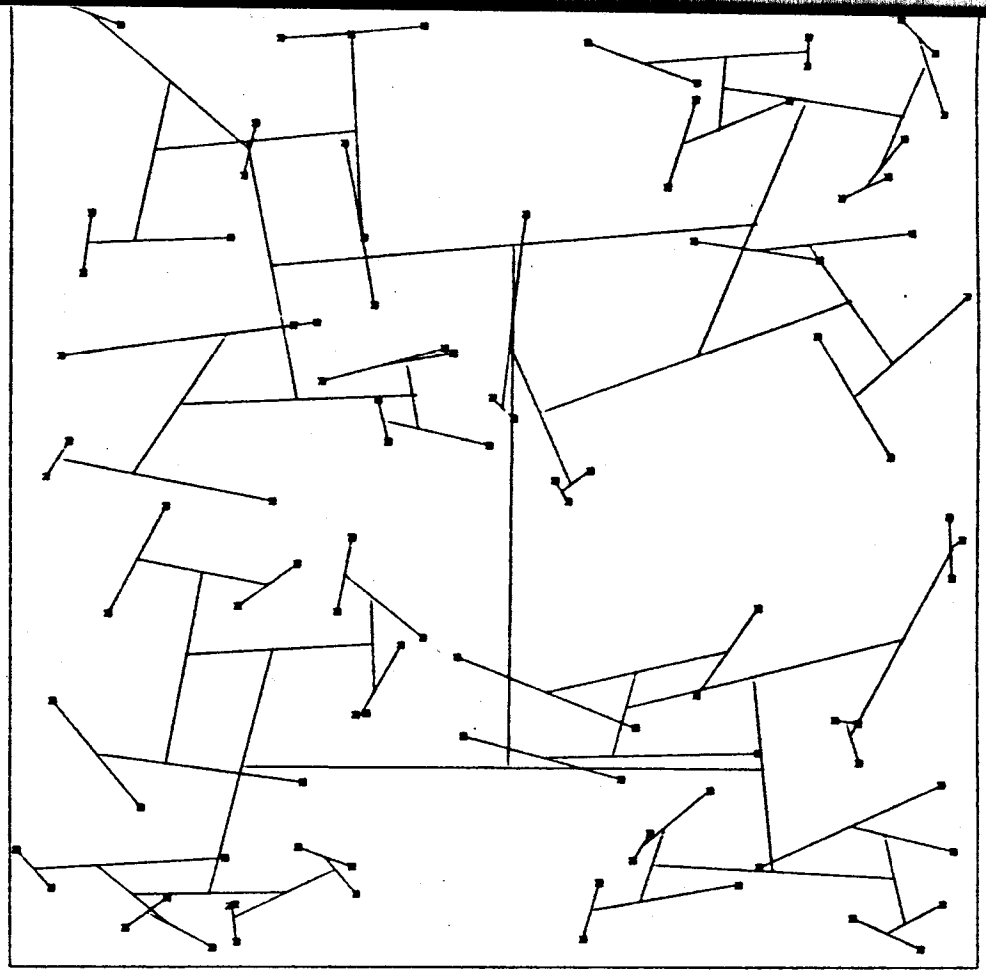
Two-body Solutions:

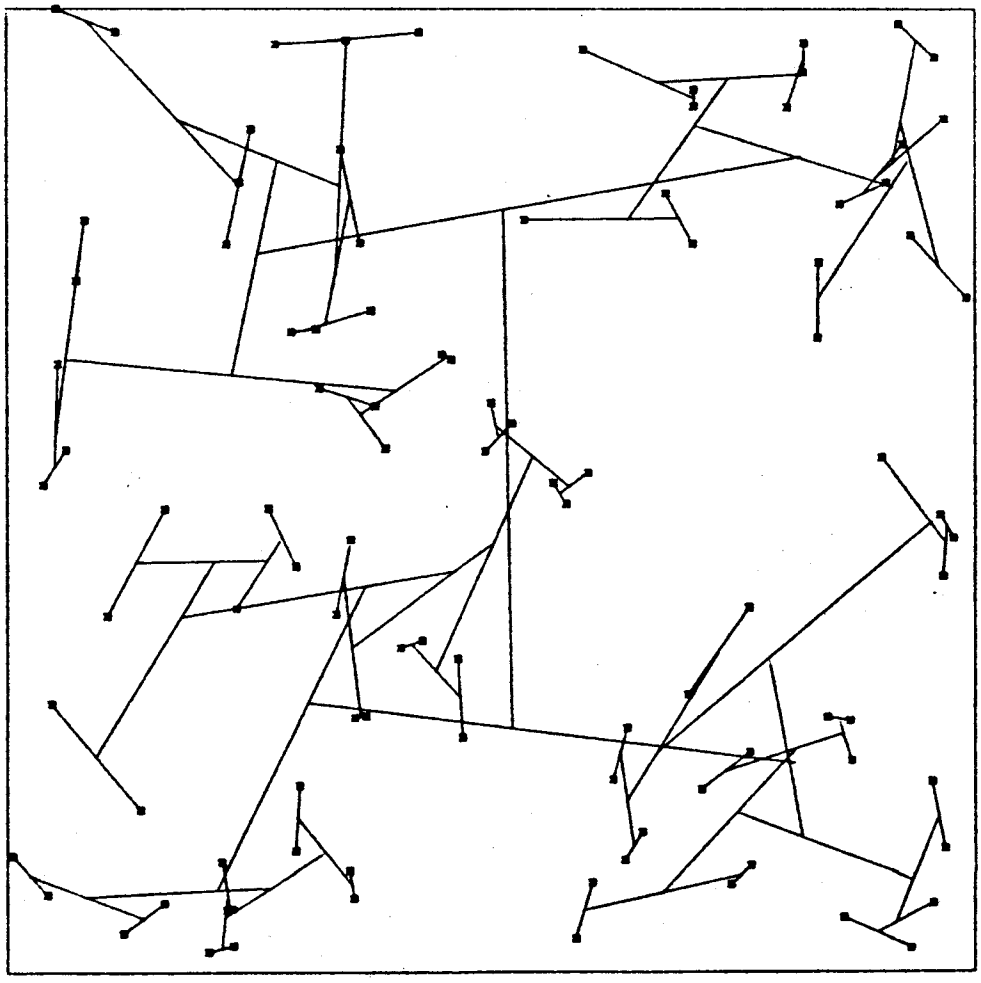
When an indivisible object itself is a clump containing two indivisible subclumps (usually, but not always, simply individual galaxies), then its orbit may be solved in closed form. In this case, the calculations to resolve internal motion may be put off not merely to the end of the global iteration, but until such time as another clump gets near enough to see the object as something with structure. This may be many iterations of the universe later -- and many times more iterations of the tight pair, which typically has a much shorter characteristic time. Only one calculation needs to be made in closed form; furthermore, this calculation will be exceedingly accurate, since no approximations are being made internally to the system. Both elliptical and hyperbolic orbits are treated in closed form. Parabolic orbits will never come up at random, since the chance of this occurring (for 48 bits of precision) is 1 in $10^{13}$.

Thus the universe might contain some dense clumps, some of which contain tight three-body systems and binary galaxies, so that there are many levels of systems, each with a different time increment per iteration -- all while conserving momentum and preserving the validity of the approximation.

Preserving the Clump Structure:

It will usually be the case, with clumps that are not indivisible, that after a Move, the coordinates of a clump will no longer correspond exactly to the center of mass of the two subclumps. This is due to a nearby object attracting one subclump more strongly than the other. It is a simple matter, however, to adjust the position of each clump after its subclumps have been moved. Sometimes, however, another subclump will intrude into a clump so that the clumps no longer represent disjoint clusters. In this case, it is necessary that the clumps be rearranged (while keeping the actual galaxies in the same place). The condition to aim for is this: for all clumps C, the closest clump to C external to C shall be its parent clump. Let Closest(C) be the nearest clump which C is compared with in procedure TwoNode. If the distance from C to Closest(C) is less than the distance from C to its parent (which is simply $|P_C|$), then a new clump W will be formed, which will become the subclump of Parent(C) which C used to be. W will contain as subclumps C and Closest(C). Now the old parent clump of Closest(C) has only one subclump, so it can be liquidated, promoting its subclump. This process is represented in figure 6.





These adjustments (which shall be known as Grabs) take place immediately after procedure ComputeAccel finishes running. Each Grab is a purely local (vis-a-vis the data structure) phenomenon, and preserves the positions, velocities, accelerations, and all other important data of all the clumps involved. The process of Grabbing guarantees that close pairs will be sub-clumps of the same clump, and that the clumps will be close to optimally arranged for quickly computing accelerations. It is probably impossible to find the "best" arrangement in a number of execution steps which is not exponential in the number of galaxies.

Initial Creation of the Clump Structure:

While grabbing is very useful in maintaining the clump structure, it will not be able to create one in the first place from a randomly arranged set of galaxies. This will be done as follows. The universal clump -- which contains all the galaxies -- will be divided initially into two subclumps chosen so that the first contains all galaxies whose X coordinate is less than the median X coordinate, and the other subclump will contain all galaxies with X larger than or equal to the median X.

Each of those subclumps will then be divided into two subclumps using the median Y as the splitting criterion. Each lower level of clump will be split on Z, then X, then Y, then Z, until the clumps consist of only one galaxy. Note that this procedure does not require that the number of clumps be a power of two, although that might seem most natural.

The resulting structure will be far from optimal -- nearby objects will not be in the same clump much of the time. However, it turns out that the Grab procedure does a very good job of cleaning up the structure (which could not have been predicted!). (See figure 7.)

---

Figure 7 is on the following page. The top diagram depicts the clump structure as first created, by alternately splitting at the median X, Y, and Z. The bottom diagram shows the structure after several iterations of Grab. Note that the galaxies are in the same positions, but the structure is cleaner -- close pairs are now all linked together.

For purposes of clarity, three things have been modified for this drawing: the universe is two-dimensional instead of three-dimensional, it is not periodic, and the galaxies have not been moved between iterations, as they normally would be.

## II. Simulating a part of the Friedman Universe

We wish to use this machine in order to simulate the development over time of galaxy structures in a real universe. It is impossible to model the entire universe in this fashion; far too many galaxies are involved, and the interaction of very distant galaxies is not well approximated by a Newtonian force law. For these reasons, only a small part of the universe -- a volume element of dimension a*a*a -- will be modelled.

The dominant accelerations on any galaxies in this box will not be from the other galaxies in the box. Rather, it is the rest of the universe whose mass and geometry will have the most effect on the motions of particles in the volume element. Intergalactic attractions within the box, which are being examined in this numerical experiment, are merely the result of local fluctuations in mass density. The net effect of interactions from outside the box will be quite significant: they cause the Hubble expansion to slow down, which has a huge effect on the positions and velocities of the galaxies. In comparison, the local accelerations will cause galaxies possibly to orbit around each other and possibly to cluster, but the absolute position of the galaxies with respect to the universe as a whole will not be much affected by local interactions.

It will be useful to find a way to remove the term coming from the geometry and mass of the entire universe and to focus on the effect from clustering and local mass density fluctuations. Fortunately, the behavior of a universe without fluctuations is reasonably well understood.

Einstein's Equation:

Suppose the universe has a mass density $\rho(t)$ which is the same everywhere. Choose a pair of masses in this universe separated by distance a(t). Then the values a and $\rho$ must satisfy the Einstein equation:

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{8\pi}{3} G\rho - \frac{k}{a^2}$$

The constant k depends on the geometry of the universe, or equivalently, upon the relationship between the mass density and the parameters of expansion. When k is negative, the density of the universe is not sufficient to halt its expansion, and a(t) grows infinitely. When k = +1, the universe at first expands, but ultimately collapses. If k = 0, there will be no collapse, but the expansion rate will approach zero. Present estimates of the mass density of the universe place its value within two orders of magnitude of the value necessary to exactly halt the expansion. Much of the evidence indicates that k is negative, and that there is less mass than would halt the expansion of the

universe, but there are various reasons for believing that k is zero. One reason is that this is a nice symmetry, and such symmetries are not only convenient to make use of, but often physically justified. Performing the N-body calculation with a particular assumption about $\rho$ and k is a means of testing that assumption; if all assumptions are correct, one would expect to see clustering similar to that observed in nature.

In this case the assumption will be made that k=0. The results of the N-body integration, compared with astronomical observations of the real universe, will test whether it is reasonable to believe that the universe is just on the borderline between collapse and infinite explosion. Ideally, one would want to try several different assumptions about the geometry of the universe, and initial positions of the galaxies, but time did not suffice to do these calculations.

When k=0, Einstein's field equation reduces to:

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{8\pi}{3} G\rho$$

Suppose any a×a×a volume element in this (homogenous) universe contains mass M. As a(t) grows, the density will go down, but the mass M in any volume element of size $a^3$ will remain constant. Since the density $\rho$ is then $M/a^3$, we have:

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{8\pi}{3} G\frac{M}{a^3}$$

Comoving Coordinates:

In order to focus on fluctuations from a uniform outward (with respect to any given galaxy) flow, coordinates are chosen such that, in a continuous, fluctuationless universe, there is no motion with respect to these coordinates. (These are called comoving coordinates, since they follow the motion of the mass elements of a homogenous universe.) These coordinates $\underline{x}$ can be placed within the a×a×a box, such that they run from 0 to 1 in each dimension, while the real coordinates $\underline{r}$ would run from 0 to a(t). Now, obviously, if the separation between two mass elements is d a(t) as time progresses, then their separation in the coordinates $\underline{x}$ will be simply d -- i.e., $\underline{r} = a\underline{x}$. Taking the second derivative with respect to time of each side, we obtain the relationship between accelerations in the two coordinate systems:

$$\ddot{\underline{r}} = \ddot{a}\underline{x} + 2\dot{a}\dot{\underline{x}} + a\ddot{\underline{x}}$$

The acceleration $\ddot{\underline{r}}$ is caused by two things: the mass and geometry of the continuous universe, and the local fluctuations:

$$\ddot{\underline{r}} = \underline{g}_u + \underline{g}_f$$

In a homogenous universe, $\underline{g}_f=0$, and $\ddot{\underline{x}}=\dot{\underline{x}}=0$. Thus, $\underline{g}_u = \ddot{a}\underline{x}$. This acceleration would be simulated in the

coordinates $\underline{x}$ by simply leaving each mass element at $\underline{x}$ = const. The "interesting" term, $\underline{g}_f$, is simply the sum of the local gravitational attractions:

$$\underline{g}_f = \sum_{\substack{\text{local} \\ \text{galaxies} \\ i}} \frac{G m_i \underline{r}_i}{r_i^3} = \sum_{\substack{\text{local} \\ \text{galaxies,} \\ i}} \frac{G m_i (a\underline{x}_i - a\underline{x})}{|a\underline{x}_i - a\underline{x}|^3}$$

Egocentric Coordinates:

What is meant by "local galaxies" has not been precisely defined. The galaxies "local" to any galaxy G must be chosen so that, if the universe were indeed homogenous, $\underline{g}_f$ acting on G would be 0. This can be achieved by choosing a cube of dimensions a×a×a, filled with galaxies, and with G at its center. Obviously, if there were a homogenous mass distribution inside the cube, there would be no net force on G from local attractions. Furthermore, there will be only a finite number of galaxies inside the cube, making calculation of $\underline{g}_f$ possible.

The only problem is that each galaxy must be at the center of the cube for this to work. This can be achieved by taking the same cube, and letting each galaxy consider itself to be at the center. That is, the universe will be periodic with period a(t). That would mean that each galaxy is repeated infinitely many times

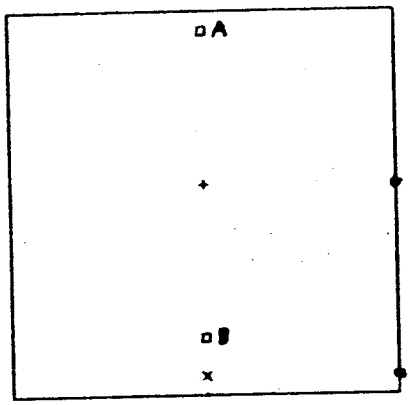



Figure 9: With periodic boundary conditions, two stars may have more than one center of mass. In this case, the average of the coordinates of A and B is at the point marked +. Other candidates for center-of-mass are marked with dots. However, the "true" center of mass -- the one nearest to both of them -- lies at point x. Point x is found by taking the shortest vector between A and B (that is, $\underline{P}(B-A)$ ), and taking its midpoint.

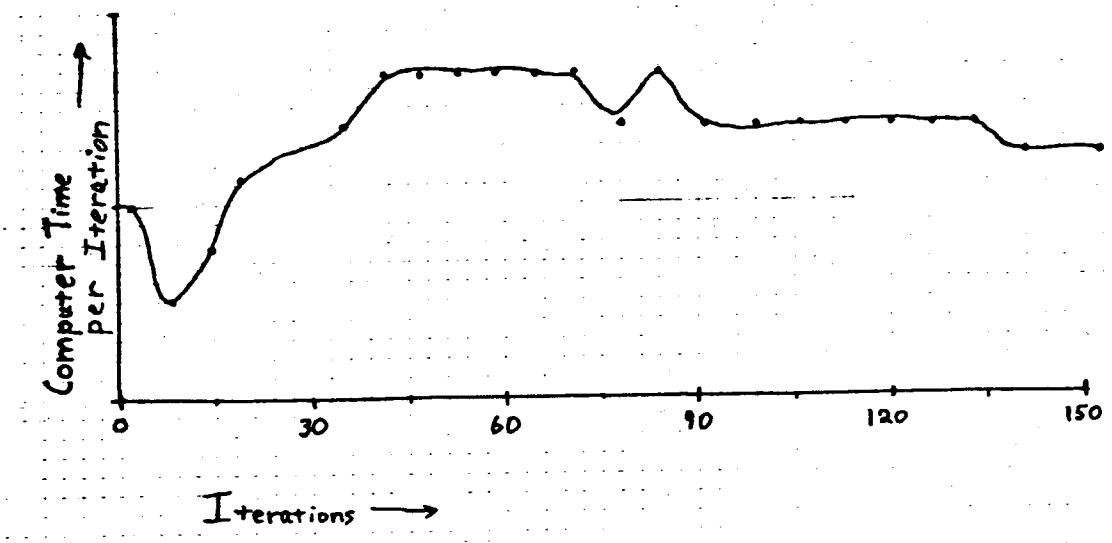



Figure 10: Efficiency of ComputeAccel

in each direction, but each galaxy in the cube "sees" only the galaxies whose x coordinates have values within 1/2 of its own. Thus, as in figure 8, galaxy A sees the same galaxies as galaxy D, but different

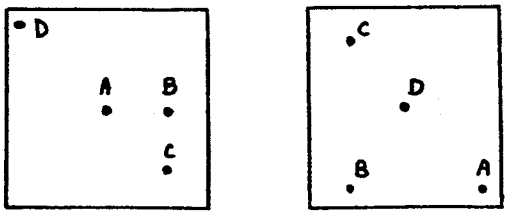



Figure 8: The Periodic Universe

"instances" of them.

The periodicity of the universe is not, of course, a real effect. Furthermore, on scales which approach the size of the box (the period length), no conclusions may be made about any observed clustering. But on any smaller scales, the periodicity will have no effect on small neighborhoods, and should not affect the validity of the result.

So, let a function $\underline{P}(\underline{x})$ be defined: for each component i of $\underline{x}$, $\underline{P}_i$ is the (unique) number between -1/2 and +1/2 which differs from $x_i$ by an integer. Then the summation for $\underline{g}_f$ can be precisely defined:

$$\underline{g}_f = \frac{1}{\alpha^2} \sum_{\substack{\text{all} \\ \text{galaxies} \\ i}} \frac{G m_i \underline{P}(\underline{x}_i - \underline{x})}{|\underline{P}(\underline{x}_i - \underline{x})|^3}$$

Note that the periodicity does inject some subtle complications into any calculation involving centers of mass. Any given pair of objects will actually have eight "centers of mass." That is, there will be eight points where the net gravitational attraction from both objects will be zero. Simply taking the weighted average of the coordinates of the two bodies will yield one of these eight points, but not necessarily the one which will be useful in further calculations. See figure 9 for an example and an explanation.

Transforming between $\underline{x}$ and $\underline{r}$:

To track the motions of the galaxies in $\underline{x}$ coordinates, it is necessary to find $\ddot{\underline{x}}$. From $\ddot{\underline{x}}(t)$ and dt, for any particular galaxy we can compute a value for $\dot{\underline{x}}(t+dt)$, and from $\dot{\underline{x}}$ we can compute $\underline{x}(t+dt)$. Since we know

$$\ddot{\underline{r}} - \underline{g}_u = \underline{g}_f = 2\dot{a}\dot{\underline{x}} + a\ddot{\underline{x}}$$

then

$$\ddot{\underline{x}} = \frac{1}{a^3} \sum \frac{Gm_i \underline{P}(\underline{x}_i - \underline{x})}{|\underline{P}(\underline{x}_i - \underline{x})|^3} - 2\frac{\dot{a}}{a}\dot{\underline{x}}$$

The summation is the result of executing Compute-Accel. Note that TwoNode must be modified to compute $\underline{P}(\underline{x}_A - \underline{x}_B)$ as its first operation, but that otherwise the procedures are unmodified. Let the mass of each galaxy be equal to 1. This means that M, the mass contained

in the volume element a×a×a, is equal to N, the number of galaxies in the volume element. In addition, let G, the gravitational constant, be equal to 1. Then, after executing ComputeAccel to find

$$\underline{A}_j = \sum_i \frac{\underline{P}(\underline{x}_i - \underline{x}_j)}{|\underline{P}(\underline{x}_i - \underline{x}_j)|^3}$$

we can find $\underline{x}_j$:

$$\ddot{\underline{x}}_j = \frac{A_j}{6\pi t^2 N} - \frac{4}{3\tau}\dot{\underline{x}}_j$$

This is a simple calculation to perform (in time proportional to N, the number of galaxies for which the calculation must be done).

Initial Conditions:

In keeping with the assumption that the universe once consisted of randomly distributed galaxies, the N galaxies will be placed in the box with random $\underline{x}$ coordinates. That is, each of the three components of the position vector of each star will be taken from a uniform random distribution between 0 and 1.

Each galaxy will, of course, have a significant initial velocity just from the mechanics of the transformation between $\underline{x}$ and $\underline{r}$ -- if the velocity in $\underline{x}$ coordinates is zero, then the velocity in $\underline{r}$ coordinates is $\dot{a}\underline{x}$. However, it is desirable to give each galaxy additional velocity so that $\dot{\underline{x}}$ will not be zero initially. It is useful to do this so that nearby galaxies do not immediately crash right into each other.

The virial theorem is applicable here, if we assume that we have not chosen a particularly special time to look at the galaxies. In general (for a $1/r^2$ force law), if a set of particles are to maintain a roughly constant distance from each other (for example, two particles in a circular orbit), each particle's kinetic energy will be one-half its potential energy. (The potential energy will, of course, be negative, so actually $T = -V/2$.) Therefore, the peculiar velocity (velocity relative to the comoving coordinates) of each galaxy will be set equal to $\sqrt{V/M}$, where V is the potential energy of a particular galaxy due to local gravitational fields. The direction of the peculiar velocity will be chosen at random from the unit sphere. Note that the potential energies of all galaxies may be computed in time proportional to NlogN by a procedure very similar to ComputeAccel.

## III. Efficiency of the Algorithm as Implemented

Although it is considerably more important to reduce the order of the complexity function (execution time as a function of N) than it is to worry about the constant of proportionality, this does not mean that the constant of proportionality can be ignored. Similarly, the space complexity (amount of memory used as a function of N) must be considered, both in terms of its functional form and its constant of proportionality.

Several things influence the time complexity. The most important, of course, is the parameter $\delta$, whose effect has already been calculated. The periodicity of the cube-shaped patch of space tends to degrade the performance of ComputeAccel, since each galaxy is closer to all of the galaxies that would have been on the other side of the box, and thus the labor-saving approximation can be used less frequently.

One effect which is difficult to predict quantitatively is the influence of actual galaxy clustering on the running time of the program. If the universe becomes highly clustered, then the conditions $r_A/d < \delta$ will be met much more frequently, and TwoNode will have less of a job to do. However, in such a universe, since each galaxy is closer to neighboring galaxies than it would otherwise be, the time increment between iterations will be smaller. These two effects tend to

cancel each other, and it is not obvious which will predominate.

A mysterious effect for which there is no obvious explanation is that the Grab procedure does not seem to work as well in the presence of periodic boundary conditions as it does without them. Due to the nature of the procedure, in which many local changes are made without regard to the global structure (although always preserving the physics of the situation), it is difficult to determine a priori how well the overall structure will be kept. The optimality of the clump structure may be studied by noting the variation in the time required to compute one iteration. Computation time for iteration i, versus i, is plotted in figure 10. One can conclude from this graph that the Grab procedure does work at least well enough that local clustering effects may be taken advantage of, since ComputeAccel does not get much slower after many iterations.

Memory Space:

The amount of memory used by the algorithm is proportional to the number of clumps, which is approximately twice the number of galaxies. The constant of proportionality is larger than that of the straightforward algorithm, however. Figure 11 shows the data stored for each clump.

```
Clump = POINTER TO RECORD
        P,V,A : Vector;
        S1,S2,Parent : Clump;
        U,R2  : Real;
        M : Integer;
        Dclose : Real;
        Sclose : Clump;
        Flags : SET OF (Below,Here,Done,
                        Tracer,Split,Binary);
        ID : Integer;
      END;
```

Figure 11: Definition of a Clump

As can be seen from figure 11, each clump has a position vector, a velocity vector, and an acceleration vector (P, V, and A). Also, it has pointers to its two subclumps (S1 and S2) and to its parent clump (Parent). The variables U and R2 are used for potential energy and approximate clump diameter, and M is simply the mass of the clump. Dclose and Sclose refer to the closest other clump found during the ComputeAccel process, and are used by the Grab procedure to determine exactly which clump to grab. Flags and ID are used for bookkeeping.

All in all, this amounts to 124 bytes of storage per clump, or 248 per galaxy. This is assuming that double-precision (8-byte) real numbers are used. Due to the fact that this algorithm does not heavily rely on extended precision of the variables (since it stores positions and velocities relative to the parent clumps), single-precision (4-byte) real numbers would certainly suffice with no loss of accuracy. This would

reduce the memory demands to 76 bytes per clump, or a total of 152 per galaxy (plus, of course, a constant amount of overhead).

Typical large computers have between 1 and 10 million bytes of fast storage, and much larger amounts of disk storage. Disk storage can be exchanged with fast storage in large chunks with a cost that makes it prohibitively expensive to access the disk for every instruction; it is reasonable, however, to access the disk after each several thousand calculations.

As an example of how the disk can be used, consider running the straightforward algorithm with only enough fast memory to hold one-fifth of the galaxies. The algorithm can still be run efficiently in the following manner. The galaxies will be divided into ten sublists. Then each pair of sublists will be brought into fast memory, and all computations involving pairs of galaxies from each pair will be executed when that pair is in fast memory. The details of this method are not important, but the result is: the straightforward algorithm still takes const*$N^2$ time, and only (on the order of) 100 disk operations are needed.

Using a crude version of the improved algorithm (ComputeAccel), the order in which data from the disk is needed is not systematic, and thus far too many disk operations would be needed. However, if the structure

of clumps is arranged such that entire clumps containing many thousand bodies are in the same area on the disk, then the procedure TwoNode will not require an excessive amount of disk operations. This is certainly worth doing in order to accommodate tens of thousands of galaxies, but for the sake of avoiding excessive complications in this area, it was not implemented. Thus, the version of ComputeAccel which was implemented must run entirely in fast memory.

The compiler used for the program was incapable of dealing with single-precision numbers, and thus, double precision had to be used. The computer which was used has 2 million bytes of fast storage, some of which is taken up by operating system overhead. Thus, the largest number of galaxies which could be simulated, due to memory constraints, was approximately 3000; the largest simulation which was actually run successfully (in the face of logistical problems) contained 1000 galaxies.

The Proportionality Constant:

Each previous mention of the constant of proportionality relating to NlogN vs. $N^2$ has simply dismissed the question. This was necessary because, although the functional form of the time complexity function may be determined from the structure of the algorithm, the constant of proportionality may be determined only from a specific implementation.

In this particular implementation, $t(N) = 0.015 \times N \times \ln(N)$ per iteration. The straightforward algorithm would take, in similar circumstances, $t_s(N) = 0.0001 \times N^2$. (This is for a computer approximately five times slower than Princeton University's IBM 3033.) The breakeven point, where the improved algorithm becomes cheaper than the straightforward algorithm in computing one iteration, is at about 1000 galaxies. However, this does not take into account the fact that the improved algorithm permits a greater dt per iteration, and will thus require fewer iterations to perform a given integration. To compare the running times of the two algorithms for an entire integration rather than just one iteration would require actually executing the straightforward algorithm, which would be somewhat informative but not very interesting.

The time required to compute a specific integration varies, of course, with the behavior of the galaxies; the time required per iteration and the number of iterations cannot be predicted in advance. In an actual run of the program, however, 1000 galaxies were integrated with a $\delta$ parameter of 0.3. For a fifty-fold expansion of the universe, ten hours of computation time were required (the equivalent of two hours on the Princeton University IBM 3033 computer) and 150 iterations were done.

Although the execution times of the two algorithms are comparable for N=1000, and it was not possible to try N much larger than 1000, the improved algorithm should produce a great saving for large N. If we assume that the NlogN term is the dominant term for N > 1000 (which is likely, although not necessarily true -- linear time overhead may still be significant at this size), then there should be a tenfold advantage in running the improved algorithm rather than the straightforward algorithm, for N=10000.

## IV. Computed Behavior of the Galaxies

There are two types of interesting results that can come from this experiment: information on the efficiency of the method, and configurations of galaxies.

The obvious thing to do with the list of galaxies and their positions at time $t=t_f$ is to plot a picture of it. A two-dimensional projection can be taken very easily just by suppressing the z-coordinate of each position vector. Because of the way in which the universe was made periodic, all areas of the picture can be treated in the same way, i.e. there is no preferred area with respect to density. (In contrast, if a non-periodic, spherical group of galaxies had been chosen, then any two-dimensional picture would tend to look denser in the middle.) Figures 12a through 17a show the evolution of a 1000-galaxy patch of space.

Note that clustering is indeed taking place. In the early projections clustering is happening on small scales, but clusters of clusters have not yet formed. By the last projection, where t=374, the one supercluster is almost as big as the box. Any continuation of the integration beyond this point will not be valid, as the periodic edge effects will manifest themselves.

A less obvious, though somewhat more quantitative, approach, is to define a function which measures the typical separations of pairs of galaxies: the two-

point correlation function. Let $f_o(r)$ be the number of pairs of galaxies which are between r and r+dr apart. Let f(r) be $f_o(r)/r^2$, that is, normalized by the amount of volume in the shell between r and r+dr. If there is a uniform distribution of galaxies, then f(r) will be constant for all r (up to the size of the periodic box). Actually, f(r) shall be further normalized by a constant so that in the uniform case, f(r)=1. (Note that the correlation function can be computed in NlogN time by an algorithm similar to ComputeAccel.)

If there is a random distribution of galaxies, f(r) should still be equal to 1 for all r, except for small r, where there would be (on the average) only 1 galaxy in the volume $r^3$, then f(r) will approach zero.

On the other hand, a clustered distribution of galaxies will manifest itself with f(r) much higher than 1 for small r, and somewhat lower than 1 for large r.

Figures 12b through 17b show the two-point correlation functions for the configurations pictured in figures 12a through 17a. They indicate that the scale on which clustering is taking place starts out small, but gradually increases. In the final plot, for t=374, there are some interesting notches in the correlation function which have no obvious explanation. If they are physical, and not a result of the boundary conditions, then something interesting is happening.

Conclusion:

The NlogN algorithm for the N-body problem, as adapted to the special conditions required for the simulation of a patch of the Friedman universe, has several advantages over simpler methods. Its distinguishing characteristic, the grouping of the mass points into a clump data structure, enables not only a reduction in the number of individual forces which must be computed, but makes possible the easy recognition of important features of a situation. Recognizing these features, such as tight two-body or n-body clusters which can be integrated with a different characteristic time, makes possible a further reduction in computation time.

Not only is it thus possible to reasonably efficiently integrate an N-body problem, but the results of doing so indicate that this is an informative thing to do. The assumption that the expansion of a universe initially containing randomly distributed galaxies will lead to clustering seems, in light of the results obtained, to be justified. If this is indeed the case, then N-body integrations can be used to examine quantitatively, rather than just qualitatively, the history of the universe.

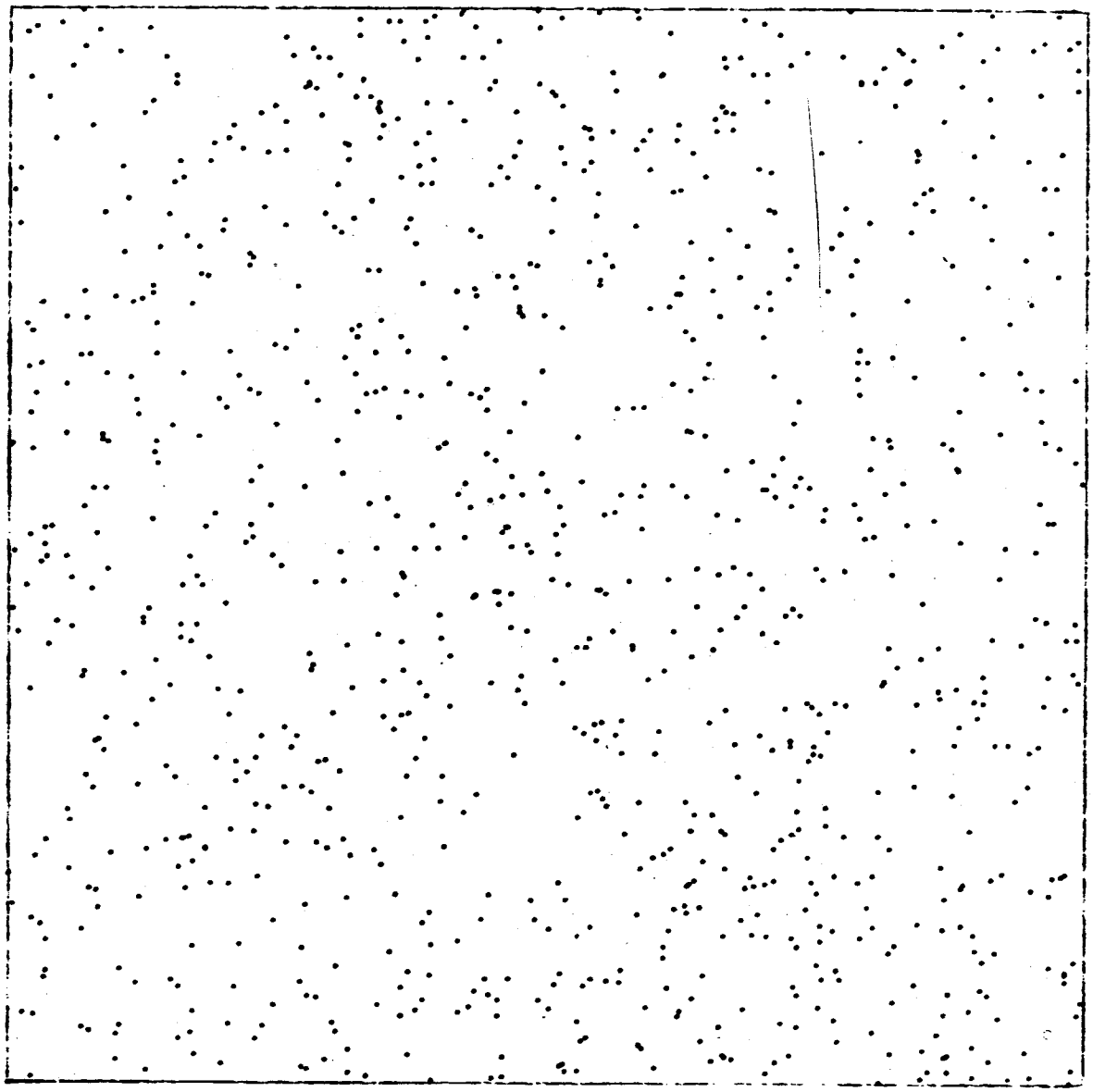
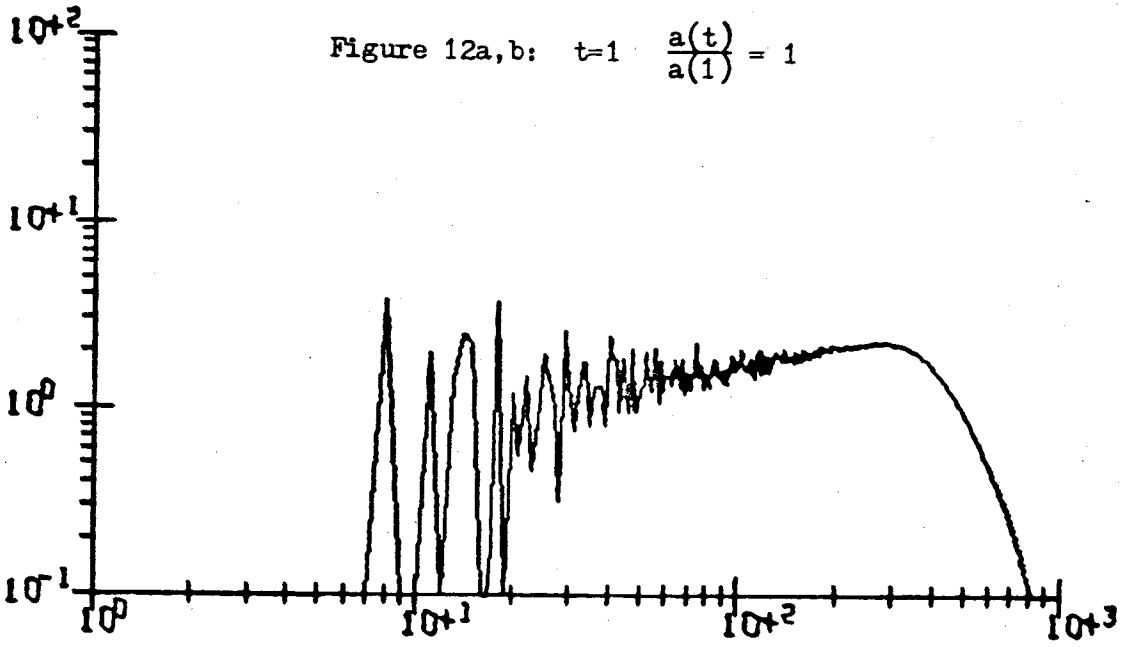


Figure 12a,b: t=1 $\frac{a(t)}{a(1)} = 1$



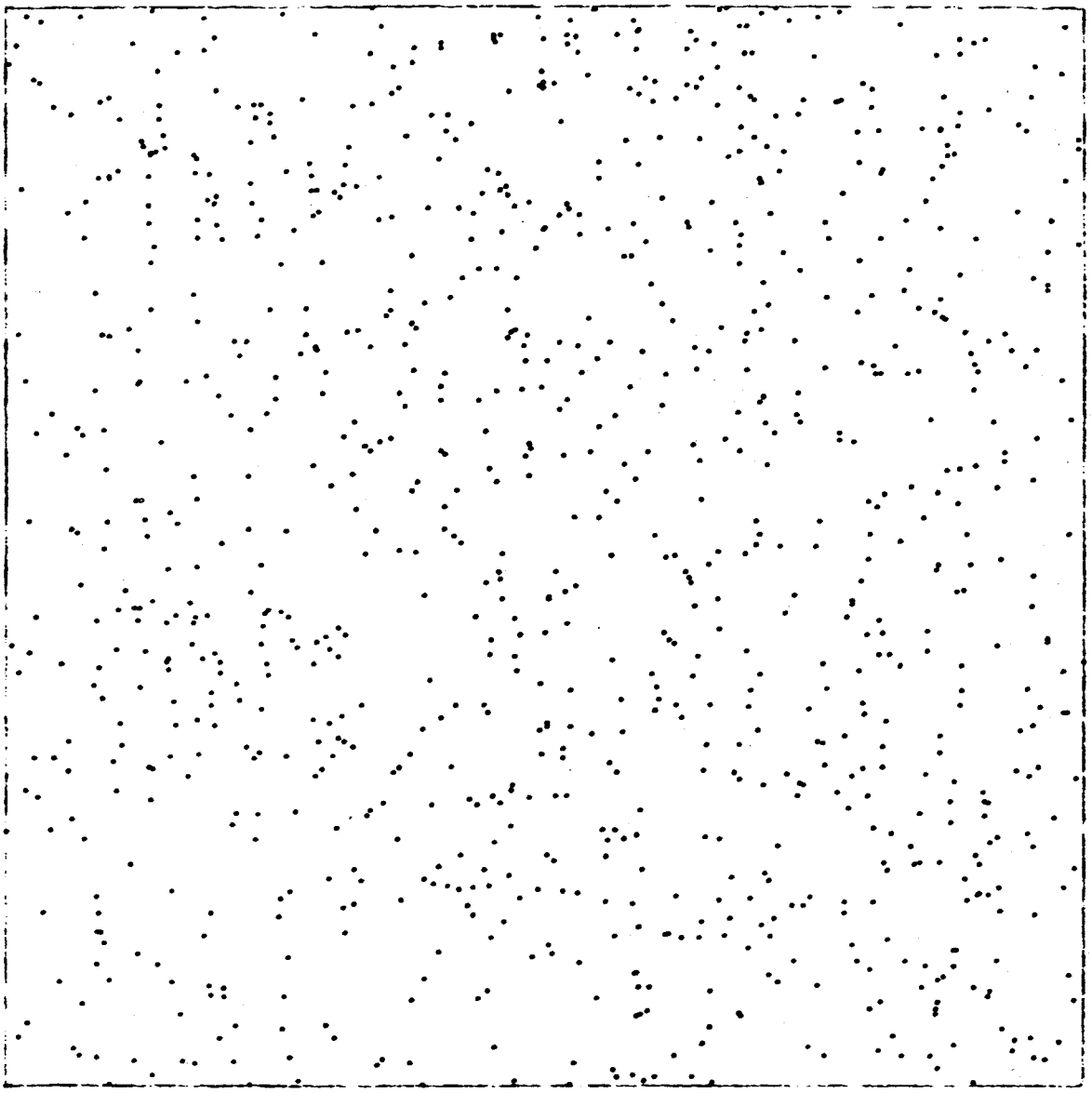
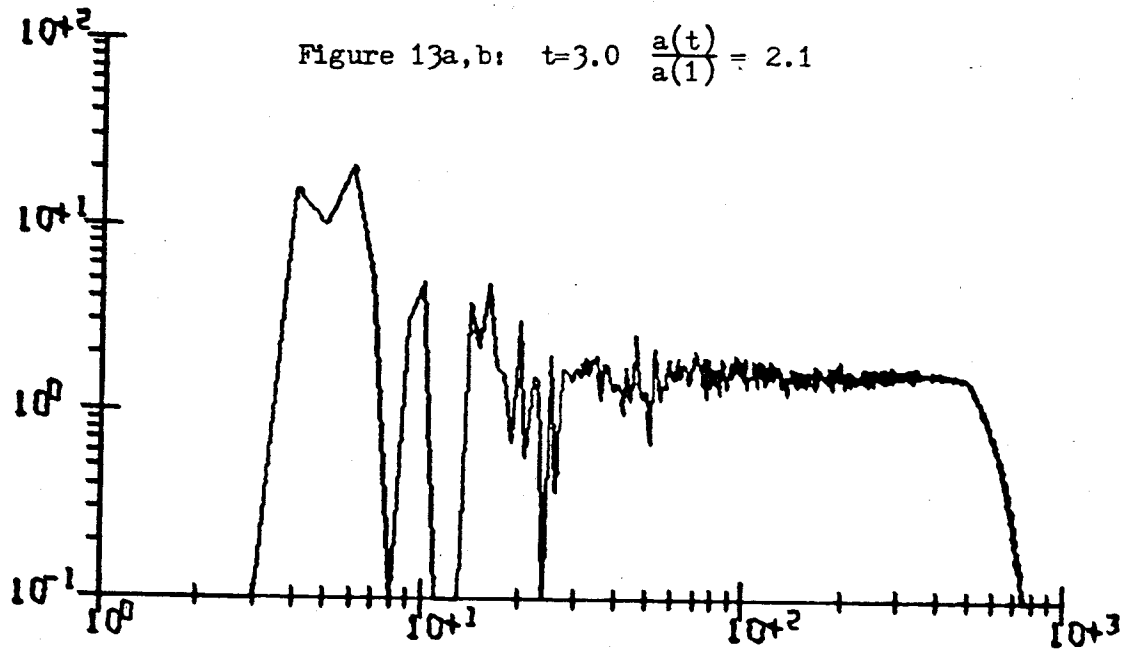
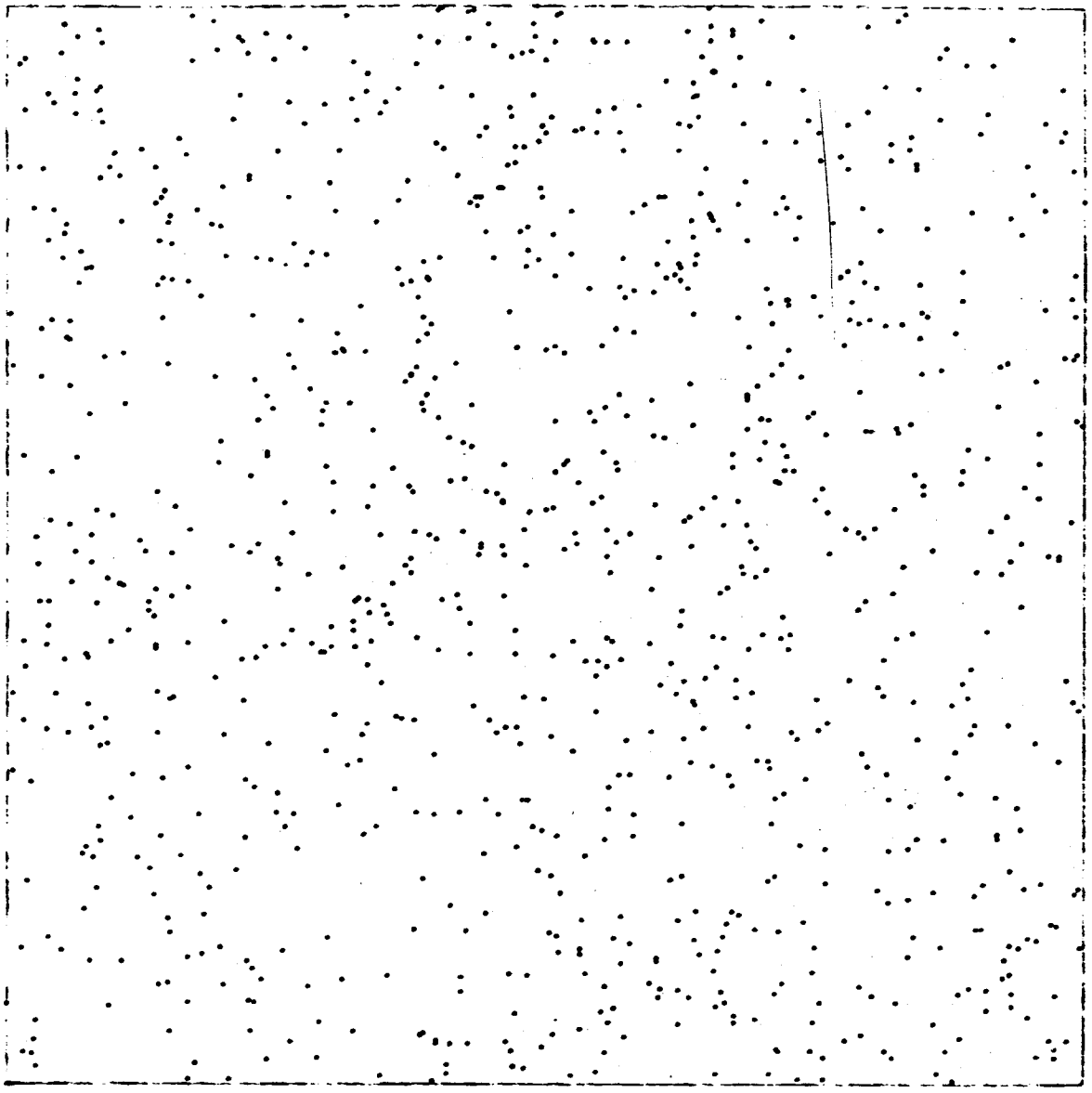


Figure 13a,b: t=3.0 $\frac{a(t)}{a(1)} = 2.1$





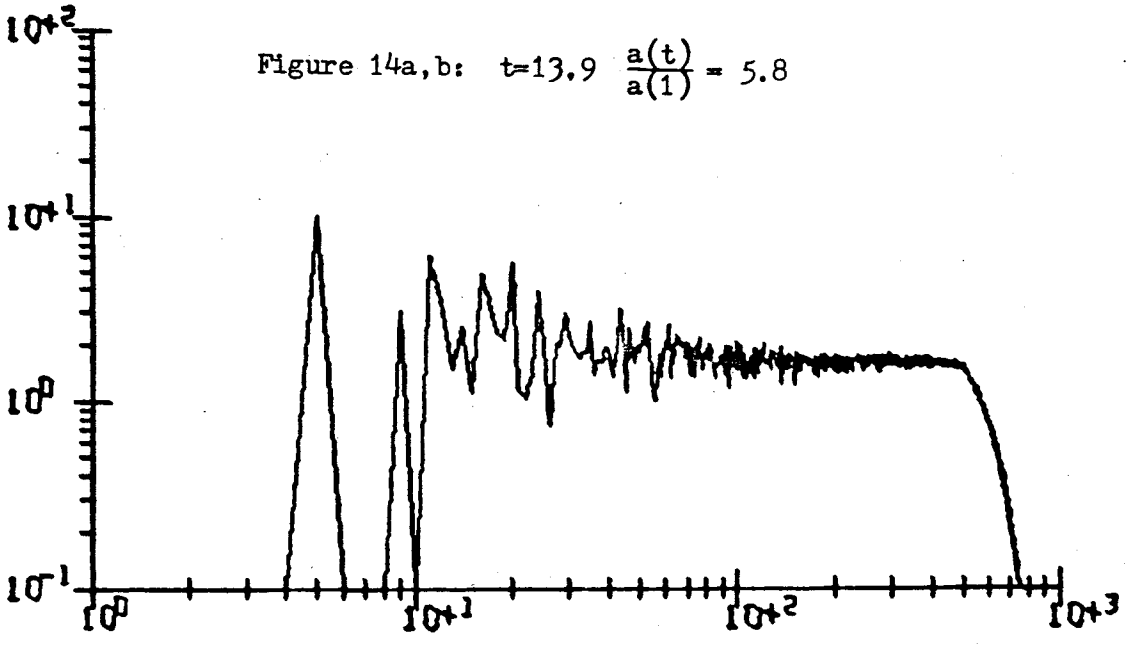



Figure 14a,b: t=13.9 $\frac{a(t)}{a(1)} = 5.8$

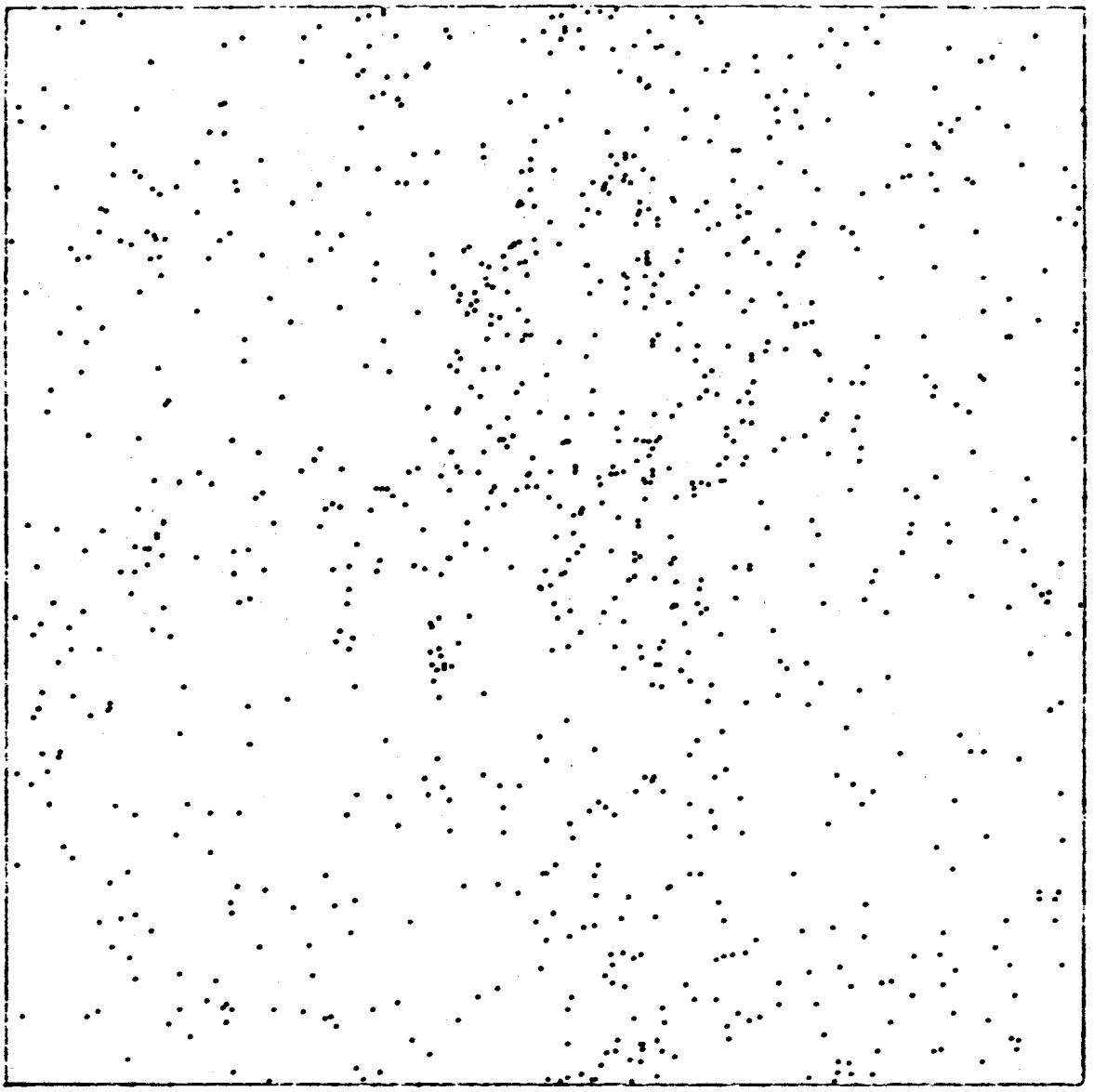
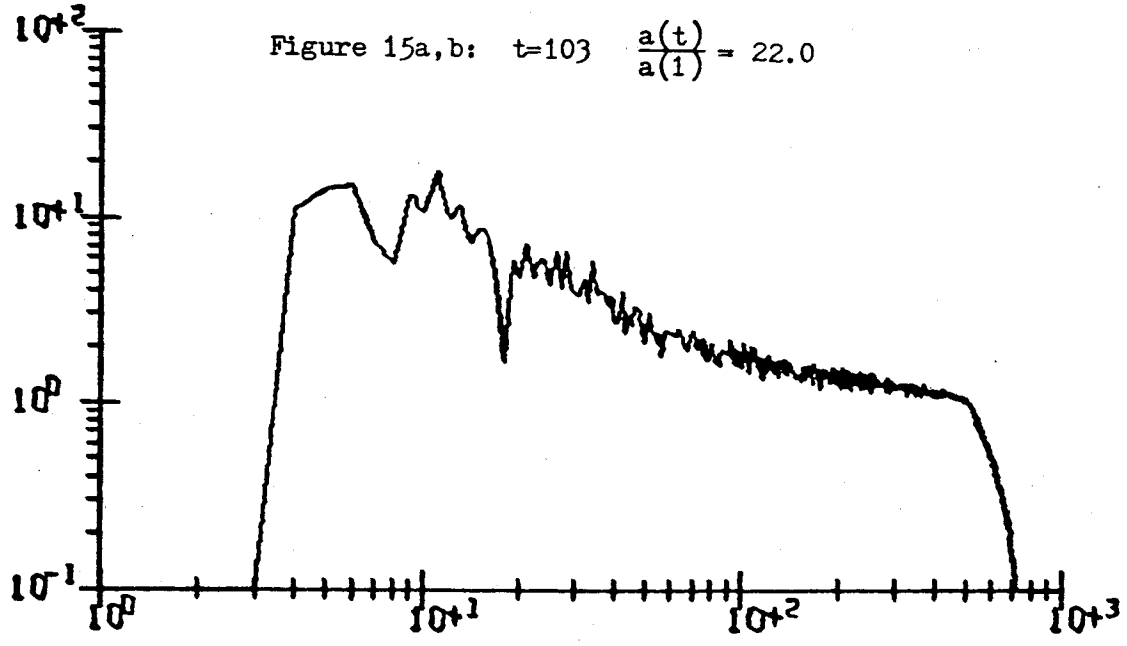


Figure 15a,b: t=103 $\frac{a(t)}{a(1)} = 22.0$



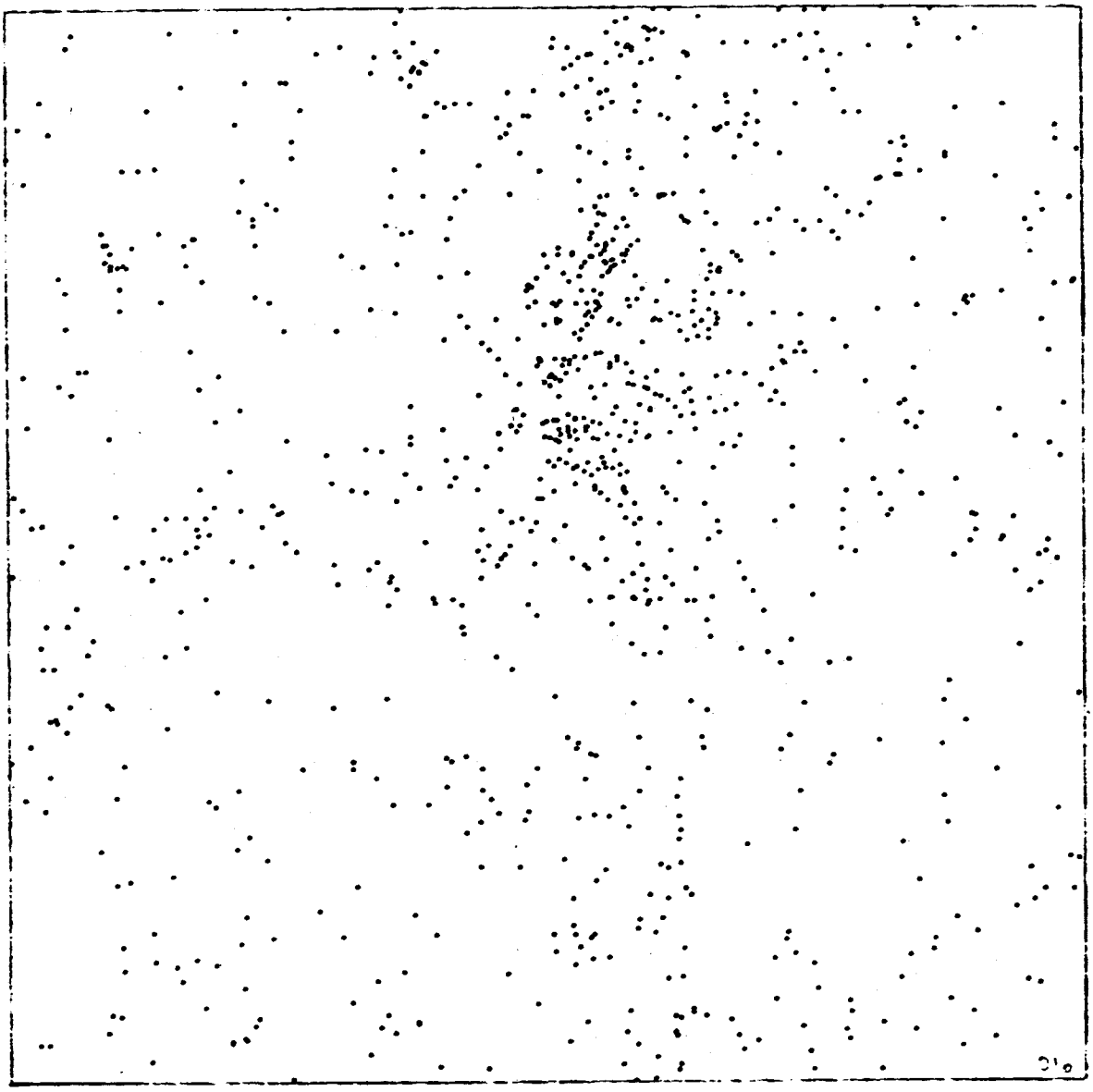
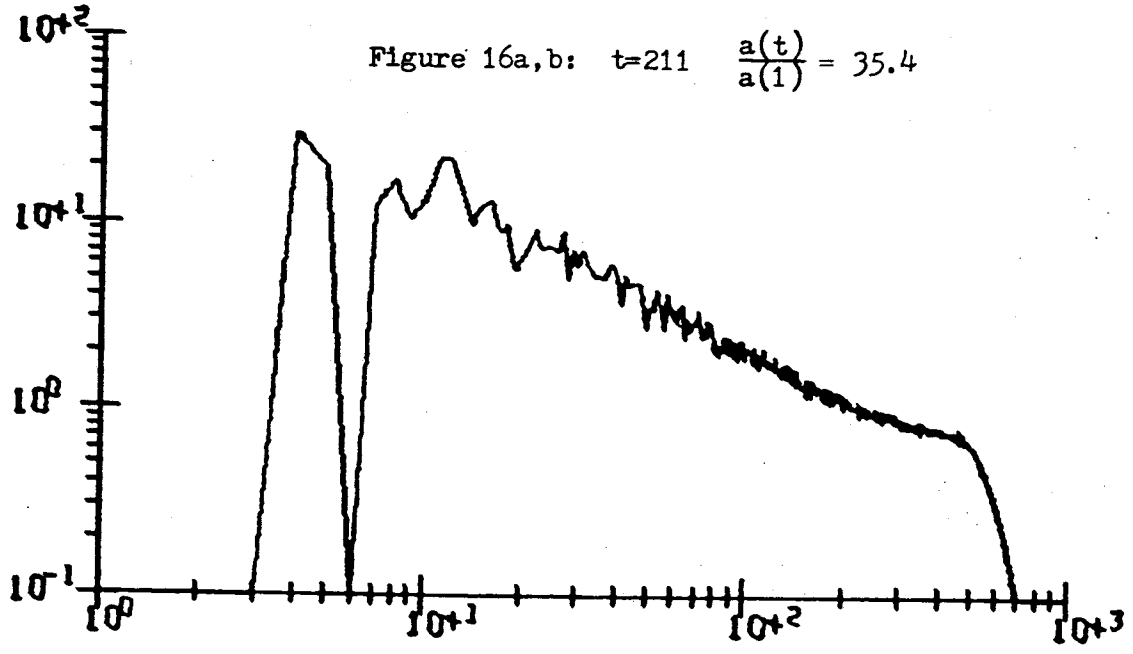
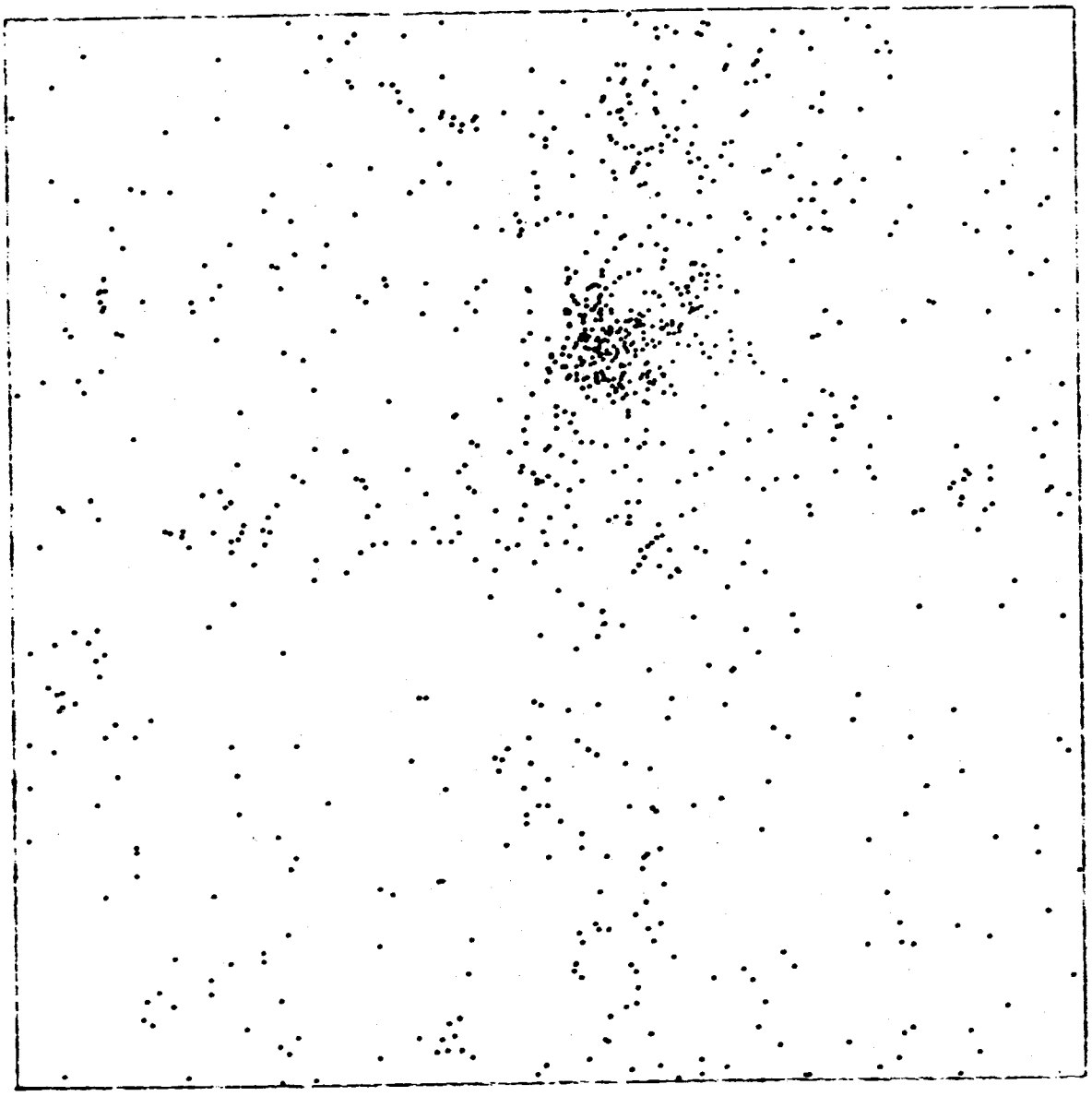


Figure 16a,b: t=211 $\frac{a(t)}{a(1)} = 35.4$





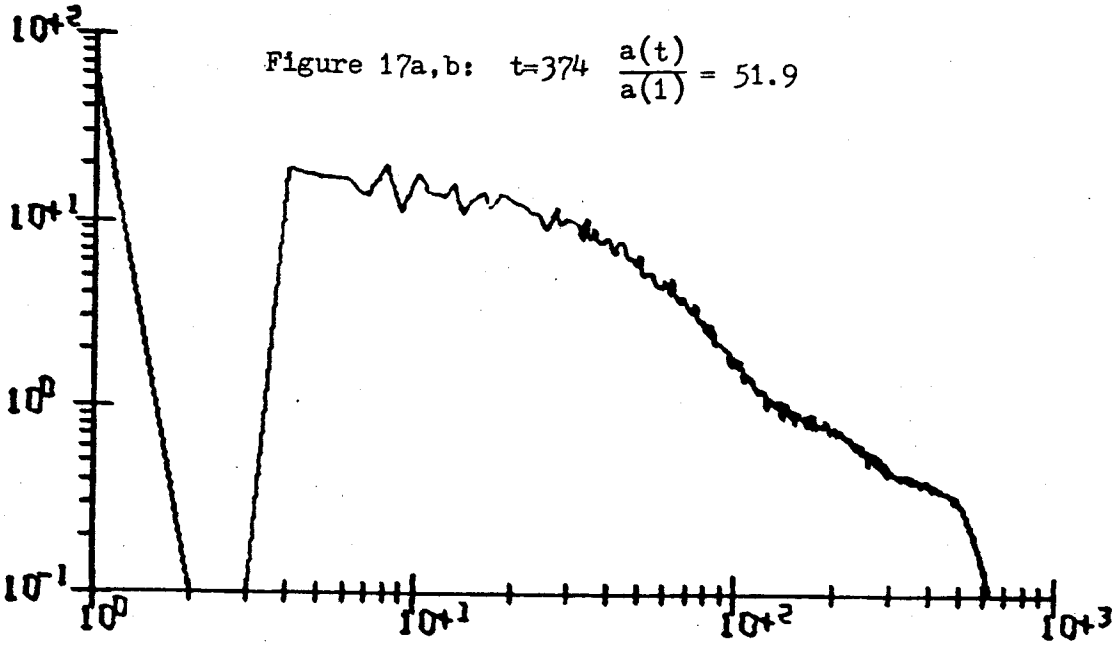



Figure 17a,b: t=374 $\frac{a(t)}{a(1)} = 51.9$