Modeling Astrophysical Disks Using Nearest Neighbor Perturbations

Jeremy Kepner

Senior Thesis

Submitted in partial fulfillment of the degree of
Bachelor of Arts in Physics at Pomona College

April, 1991

ABSTRACT

A new way to model astrophysical disks based on nearest neighbor perturbations (NNPs) is presented. Key aspects of the NNP model are: chopping up space into a hexagonal grid, imposing differential rotation (as opposed to letting it arise naturally), treating the boundaries as reservoirs, and making the system independent of scale. A Fortran program was written to test the model. The algorithm was fast, and calculations using 10^5 particles could be run on a micro-computer. Simulations of disks with different masses were carried out. The driving force of the model was accretion. The rate of accretion was dependent on the amount of mass in the disk. Spiral arms and ringlet phenomena were observed. The type of pattern depended on the mass in the disk.

ACKNOWLEDGEMENTS

Foremost, I would like to thank my advisor Prof. Robert Chambers for his time and patience. I would also like to thank: Prof. Catalin Mitescu for his help with the programming aspects of the this project, the Geology Department for the use of their computer facilities, Dr. John Hackwell for sparking my interest in disks and nebula, and Prof. Alexandre Chorin for introducing me to the topic of computational physics.

TABLE OF CONTENTS

1. INTRODUCTION

One of the most beautiful phenomena in nature is a pattern that appears in a variety of situations over a vast range of scales. Astrophysical disks—the flat circular structures that can be found around planets, stars, and galaxies—are examples of such a pattern. Their flat twisting curves sharply contrast the spheres that they surround. This beauty along with their physical importance has made astrophysical disks a well-studied subject. Extensive direct and indirect observations have been carried out on the planetary (Smith *et al* 1981), stellar (Snell 1989), and galactic level (Mihalis & Binney 1981). Theoretical and computational work has yielded insight into density waves (Shu & Lissauer 1983), spiral arms (Toomre 1981), ringlets (French *et al* 1988) and other features, but a complete model is still far away (Tremaine 1989).

The emphasis of previous research has been on analytic results. More recently computational modeling has started to play a major role (Sellwood 1989). Standard grid based N-body problem approaches (Sellwood & Carlberg 1984) and Smooth Particle Hydrodynamics codes (Gingold & Monaghan 1982) have been able to reproduce many disk phenomena (Benz *et al* 1990, Artymowicz & Lubow 1989). These methods are strongly motivated by the physics of disks. Although, general results are sometimes difficult to obtain. The work presented here takes a more radical approach that, admittedly, is more difficult to justify, but leads to interesting results never-the-less.

One of the more recent ideas in physics is that small, pervasive forces on a complex system can have a large effect (Lorenz 1979). Keeping with this theme, I am hypothesizing that disks can be simulated with a far simpler model based on nearest neighbor perturbations (NNPs). That is, each particle in the model only "feels" the particles nearest to it. All the other particles are assumed to be far enough away that their net effect is small.

A "typical" disk simulation involves numerically integrating the equations of motion for particles moving around a central mass. Taking into account the gravitational interaction between the particles results in a standard N-body problem. More complex phenomena such as collisions,

non-uniform central fields, resonances with other bodies, etc ... can be added as well. In these models, NNPs are overwhelmed by the more dominant forces of the system. To test my hypothesis, I propose to do a disk simulation in which NNPs dominate.

Two advantages of using such a simple interaction are speed and generality. Dramatic simplifications in the code can be made resulting in programs that can be run on micro-computers. Also, the program can be scaled so that such parameters as central mass, radius, and period have little effect on the behavior. Thus, the results should apply to planets, stars, and galaxies.

§2 discusses the theory behind the NNP model more explicitly. In §3 the details regarding the development of an algorithm and its implementation are given. §4 presents some experimental results, the implications of which are discussed in §5.

2. THEORY

In this section, the physical justification for the NNP model is given. The two fundamental features of the model are: 1) the particles are only perturbed by their nearest neighbors; 2) the overall motion of the particles is Keplerian (although any type of differential rotation would suffice).

To determine nearest neighbors requires that the area of interest of the disk (in this case a ring) be chopped up into a grid. §2.1 describes how this physical space is chopped up into a hexagonal grid, which I call "ring space". Knowing how the ring is gridded makes explaining the interaction between particles easier. The most important part of the NNP interaction is determining what configurations of nearest neighbors will cause a particle to move. This is called the "movement condition" and is explained in §2.2.

The second aspect of the model is Keplerian differential rotation. Since this property is imposed—rather than obtained from integrating the equations of motion—any velocity profile can be chosen. Keplerian motion is used because it is analytic and many computational simplifications can be made. The most important of which is "relative differential rotation". The overall motion in space becomes trivial, and only the relative motion of the particles with respect to each other matters. The theory behind this is given in §2.3.

The simulation can only be carried out over a finite ring of the astrophysical disk, which raises the question of boundary conditions. The implementation of the boundaries is explained in §2.4. Finally, how all this theory ties together to produce a model that scales as well as it does is discussed in §2.5.

2.1 Ring Space

We begin by looking at a ring around a central mass, M_0, with a mean radius R and width W (Fig. 2.1). For the time being, R/W is assumed to be sufficiently large that the space of the ring can be approximated by the rectangle: W x 2πR. Next, the physical space of the ring is chopped-

up into a finite grid of allowable locations—ring space. This griding (or hexing) serves two purposes. First, the nearest neighbors are readily identifiable. Second, it should make the effects of NNPs more prominent by emphasizing changes in position.

The hexagonal grid was chosen for two reasons. The nearest neighbors are more obvious, and the movements are less trivial than in square lattices (Fig. 2.2). Furthermore, increasing the interaction length to the second nearest neighbors is clear as well. The main drawback being the complexity of working with a hexagonal grid.

2.2 Nearest Neighbor Perturbations

The gravitational force on a particle p, with mass m_p, at location i will be defined as

$$\mathbf{F}_p = G\sum_j m_p m_j \frac{\mathbf{d}_{ij}}{d_{ij}^3} \quad . \tag{2.1}$$

The sum being over the nearest neighbors j of i, which for hexagonal ring space reduces to

$$\mathbf{F}_p = \frac{Gm_p}{d^2}\sum_{j=1}^{6}\mathbf{m}_j \quad , \tag{2.2}$$

where $\mathbf{m}_j = m_j\frac{\mathbf{d}_{ij}}{d}$, m_j = total mass in hex j, and d = distance between the centers of two adjacent hexes (Fig. 2.2). Any configuration of particles is only valid for a finite time, Δt, before differential rotation breaks it up. Δt is how long $\mathbf{F}_p$ acts on m_p. The change in position of m_p due to $\mathbf{F}_p$ in this time will be

$$\Delta\mathbf{r}_p = \tfrac{1}{2}\frac{\mathbf{F}_p}{m_p}\,\Delta t^2 \,. \tag{2.3}$$

If $\Delta\mathbf{r}_p \sim d$, then the force on m_p is large enough to move it out of its current hex into the hex conjugate with $\mathbf{F}_p$.

It is convenient to define a quantity

$$\mathbf{f}_p = \sum_{j=1}^{6}\mathbf{m}_j \quad , \tag{2.4}$$

called the "vector mass", which is proportional to the force on m_p. Combining Eqs. (2.4) and (2.5), and noting that for a given simulation, G, d, and Δt are constants, we see that m_p will be perturbed into an adjacent hex if

$$|\mathbf{f}_p| \geq \frac{2d^3}{G\Delta t^2} \equiv C_M = \text{a constant}, \tag{2.5}$$

where C_M is called the "motion constant". Eq. (2.5) is called the "movement condition", and evaluating it for all the particles in the ring is the main purpose of the program.

An important constraint on the previous analysis is that $d << W$. Otherwise, the change in radius will become sufficiently large that angular momentum effects come into play.

2.3 Relative Differential Rotation

The velocity of small particles in circular orbits around a large central mass is given by

$$v(r) = \sqrt{\frac{GM_0}{r}} . \tag{2.6}$$

The NNP model is mainly concerned with the positions of the particles with respect to each other. It is convenient to fix the outermost particles of the ring ($r = R + W/2$). The relative differential rotation then becomes

$$v_{rel}(r) = v(r) - v(R + W/2) . \tag{2.7}$$

where r is the radius of the orbit.

As discussed in the previous section, differential rotation is what breaks up configurations. Let Δy be the vertical distance between hexes in the same column, and Δx be the horizontal distance between hexes in the same row (Fig. 2.3). Note: $d = \Delta y = \Delta x\sqrt{3}$. One way to approximate Δt is to say that it is the time it takes for a particle at $r - \Delta y$ to move Δx relative to a particle at $r + \Delta y$

$$\Delta t(r) = \frac{\Delta x}{v(r - \Delta y) - v(r - \Delta y)} . \tag{2.8}$$

In the above equation Δt is a function of r. Only when $r << W << R$ is Δt a constant. Although these conditions are not strictly true (W is only somewhat less than R), the approximation $\Delta t \approx \Delta t(R)$ is well within the overall accuracy of the model.

At this point it is worth noting the most important computational advantage of this algorithm. Using Eq. (2.8) results in time steps that are on order of half a period. Where other algorithms will have 100 time steps per period, NNP will have two or three. The reason for this

large difference is that other algorithms need additional time steps to reproduce the overall differential rotation.

Knowing the velocity and the time step the relative change in position per time step is

$$\Delta \mathbf{r}_x = v_{rel}(r)\, \Delta t \,. \tag{2.9}$$

This is the equation that is used to implement differential rotation on the ring.

2.4 Reservoir Boundary Conditions

To close the ring, a periodic boundary was employed on the vertical sides of the rectangle (Fig. 2.1). The horizontal sides of the rectangle can be given a variety of boundaries. The focus of this project is on the behavior of particles away from the boundaries. Therefore, the boundaries themselves should cause as little disturbance as possible on the rest of the ring. One way to construct such boundaries is to view them as "reservoirs" of particles. *i.e.* Any effect the rest of the ring has on them is considered negligible. Reservoir boundaries have the same properties as the rest of the ring—NNPs and differential rotation—except that they never change. Particles can go in and particles can come out, but within the boundaries the distribution of particles remains constant.

2.5 Scaling

The scaling advantages of this model are subtle and are best understood by observing the execution of the program. In general, there are two ways in which the model scales. The first has to do with the physical parameter M_0. Notice that the only effect of changing M_0 is to change v(r). By Eq. (2.8), Δt is changed as well. However, Eq. (2.9) is what is used to calculate the differential rotation, and in this equation M_0 factors out. The second scaling occurs along geometric lines. It turns out that varying the total number of hexes, H, has no effect on the overall geometry. Even though d varies inversely with H, Δt changes in the opposite sense. Furthermore, the changes balance out so that the movement condition, Eq (2.5), remains constant with respect to H.

The advantages of the model being independent of M_0 and H (and to some extent R and W), means that there are very few variables to observe. In fact, the only really important variables are the number of particles in the system and how they are distributed.

Also along these lines is the question of units. Since the model scales, it doesn't really matter. So we choose distance → A.U., mass → Solar masses, and time → Earth years.

2.6 Figures for §2

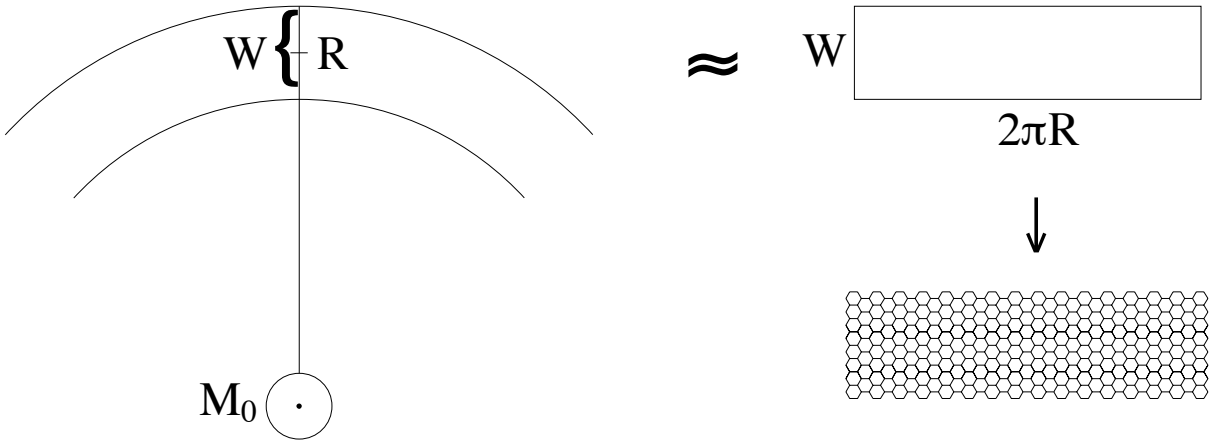


Fig. 2.1: A section of the disc of width W and mean radius R can be approximated as a rectangle. This rectangle is chopped up into a finite number of hexagons which make up "ring space".

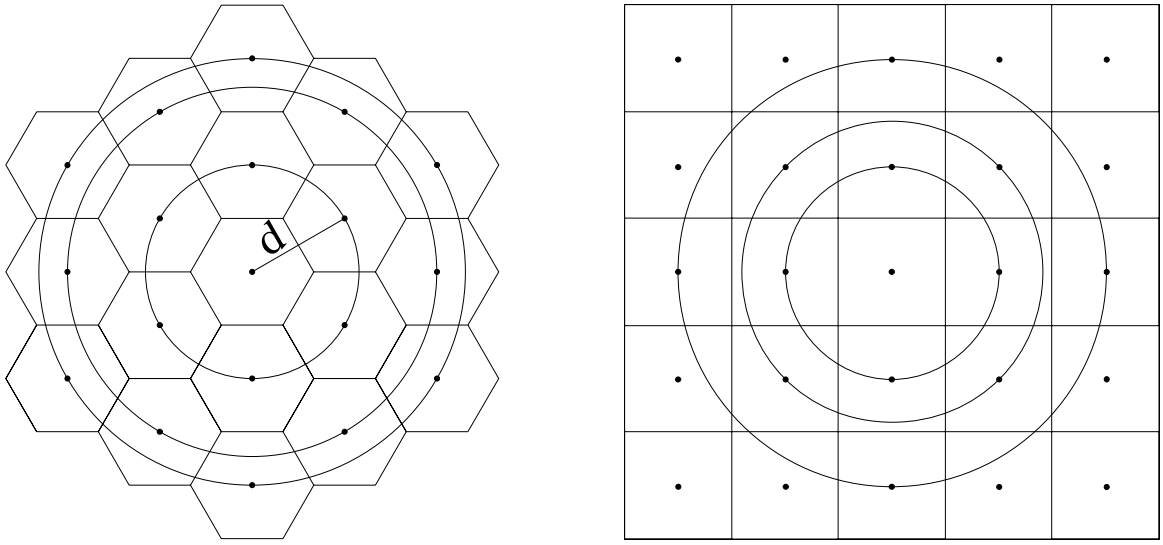


Fig. 2.2: Hexagonal vs square grid. Note: the first and second nearest neighbors in the hexagonal grid are easier to identify than in the square grid.

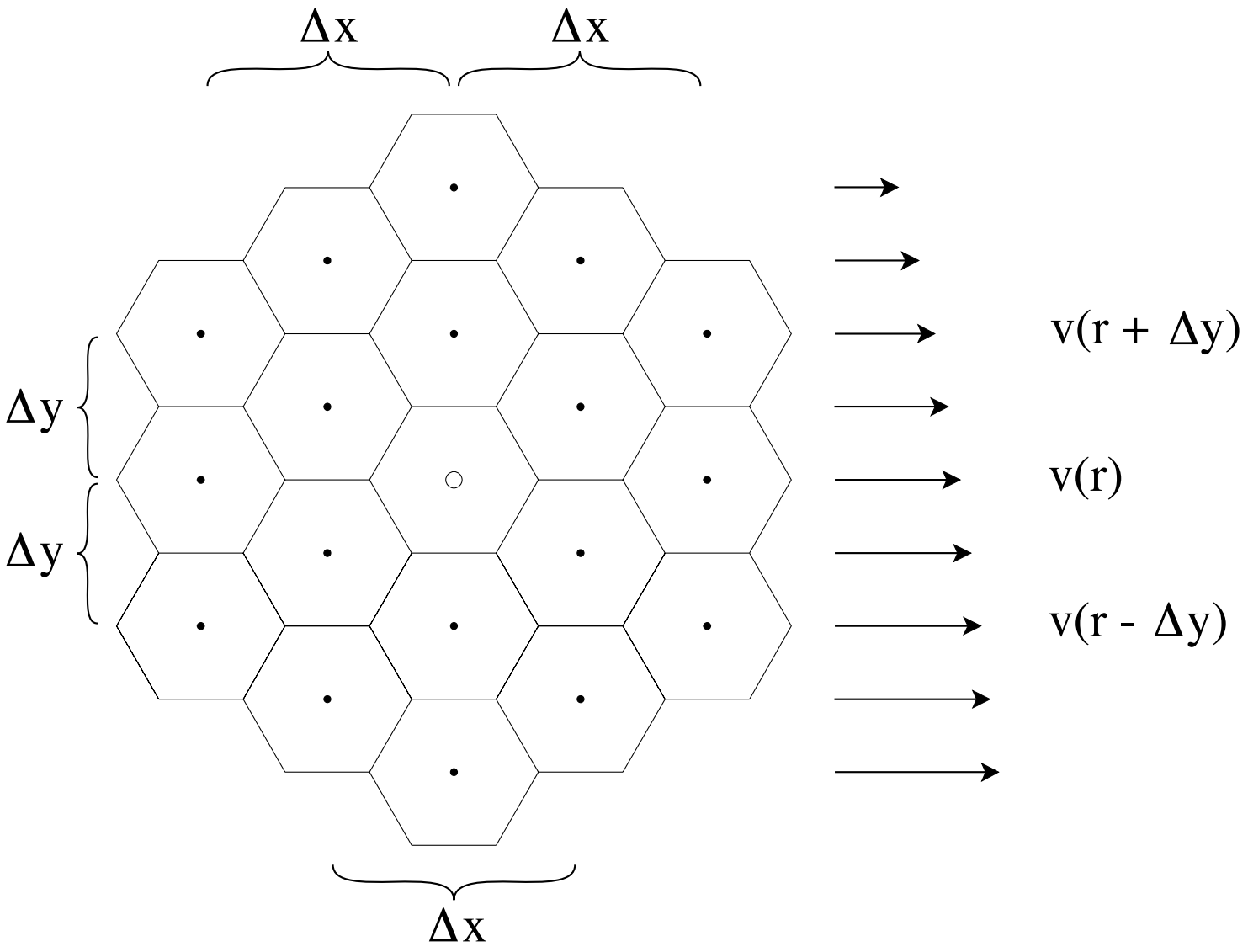


Fig. 2.3: Differential rotation of the hexagonal lattice. Δt is the time is takes for a particle at r - Δy to move Δx relative to a particle at r + Δy.

3. THE PROGRAM

The NNP model was implemented in a Fortran program of approximately 1700 lines. The operation centered on the manipulation of three arrays (§3.1). These arrays stored the mass distribution, the coordinates of each particle, and the coordinates of the nearest neighbors. Evaluating the movement condition of each particle was the primary task of the main loop, and considerable effort was put into optimizing this code (§3.2). Differential rotation corresponded to offsetting each row in the the mass distribution array, while also updating the positions of the particles (§3.3). The four top and bottom rows of hexes were devoted to the boundaries. Using a combination of "soft" and "hard" boundaries a reservoir boundary was achieved (§3.4). Two methods were used to distribute particles (§3.5). The random method picked the coordinates of each particle using a random number generator. The uniform method simply placed one particle in each hex. The program ran in a interactive mode, which dramatically reduced the debugging and simulation time (§3.6). Finally, the analogy between the NNP model and a Monte Carlo algorithm is discussed (§3.7).

3.1 Main Arrays

The majority of operations were performed on three large, two byte integer arrays denoted: Rspace, NNmap, and Mlist. These names stand for "Ring space", "Nearest Neighbor map", and "Mass list", respectively.

Rspace was a two dimensional array with each element storing the total mass in one of the hexes. The mapping of the hexagonal grid into memory is shown in Fig. 3.1. Each row in memory represents a row of hexes with the same orbital radius, which made imposing differential rotation much easier.

The rather unorthodox way in which the hexagonal lattice was represented in memory made determining the nearest neighbors more difficult. The nearest neighbors depended upon the evenness of the y coordinate and the position relative to the boundaries. Referencing the nearest

neighbors is probably the most commonly called function in the program. These coordinates were placed in a look-up table: NNmap. NNmap was a three dimensional array that was equivalent to stacking twelve Rspace arrays on top of each other. NNmap stored the coordinates of all six nearest neighbors for each coordinate in ring space. Although this array was large—24 bytes per point—it dramatically increased the speed of the program.

The third array, Mlist, stored the mass and the x and y coordinates of each particle. Storing the mass made it possible to have particles with different (integer) masses. The central loop operated by stepping through Mlist and evaluating the movement condition of each particle.

3.2 Movement Condition

Evaluating the movement condition of a particle consists of two steps. First, computing the value of $|\mathbf{f}_p|$. Second, comparing $|\mathbf{f}_p|$ with C_M and acting upon the result. $|\mathbf{f}_p|$ is obtained from the main arrays in the following manner. The x and y coordinates of a particle in ring space are obtained from Mlist. Plugging these coordinates into the NNmap gives the coordinates of the six nearest neighbors. The mass in each of the nearest neighbors can then be looked-up in Rspace. This process is illustrated by the following Fortran code fragment:

```
C  Get coordinates of particle P.
     I = Mlist(P,X)
     J = Mlist(P,Y)

C  Get mass of each nearest neighbor of P.
     m1 = Rspace(NNmap(I,J,1,X),NNmap(I,J,1,Y))
     m2 = Rspace(NNmap(I,J,2,X),NNmap(I,J,2,Y))
     m3 = Rspace(NNmap(I,J,3,X),NNmap(I,J,3,Y))
     m4 = Rspace(NNmap(I,J,4,X),NNmap(I,J,4,Y))
     m5 = Rspace(NNmap(I,J,5,X),NNmap(I,J,5,Y))
     m6 = Rspace(NNmap(I,J,6,X),NNmap(I,J,6,Y))
```

Where m1,...,m6 are integer variables containing the mass of each nearest neighbor. Knowing these masses, it is a fairly simple matter to calculate $\mathbf{f}_p$. Fig. 3.2 shows an example calculation of $\mathbf{f}_p$. Suffice it to say that both the magnitude and direction of $\mathbf{f}_p$ could be computed with integer operations.

Strictly speaking, $|\mathbf{f}_p|$ should be compared with the value of C_M given in Eq. (2.5). An alternative approach is to define C_M to be a constant (e.g $C_M = 1$), and make d and Δt consistent with it. However, d and Δt are fairly well defined. So what gives? Up to this point, no attention has been paid to the masses of the particles themselves. They are integers, but integer values of how much mass? Setting C_M has the operational effect of fixing the mass of the particles. Algebraically then, Eq. (2.5) can be rewritten as

$$|\mathbf{f}_p| m_0 \geq \frac{2d^3}{G\Delta t^2} \quad \Rightarrow \quad |\mathbf{f}_p| \geq \frac{2d^3}{m_0 G\Delta t^2} \equiv C_M , \tag{3.1}$$

where $|\mathbf{f}_p|$ is now dimensionless. Increasing C_M is then equivalent to reducing the mass of each particle.

Having calculated $|\mathbf{f}_p|$, if $|\mathbf{f}_p| < C_M$, the force on the particle is too small to move it from its hex. If $|\mathbf{f}_p| \geq C_M$, then the particle is moved to the hex conjugate with $\mathbf{f}_p$. This is done by changing the coordinates of the particle in Mlist and adding and subtracting m_p from the appropriate hexes in Rspace.

3.3 Differential Rotation

Motion was imposed by offsetting each row in Rspace. In addition, the position of each particle in the Mlist had to be updated as well. The offset of each row was computed using Eq. (2.9), and was stored in a look-up table. Some typical values are shown in Table I. Table I illustrates three points:

1) Δt is a significant fraction of a year.

2) The velocity of the outermost row is zero.

3) Significant relative offsets do develop.

The numbers in Table I are reals, while the arrays can only be moved in integer amounts. The actual offset comes from rounding off $\Delta\mathbf{r}_x$. To insure the most accurate results, the total integer offset, $\Delta\mathbf{r}_{tot}$, was kept. The integer offset, $\Delta\mathbf{r}_{int}$, for a particular time step, t, was then computed from

$$\Delta\mathbf{r}_{int} = \mathrm{INT}(t\Delta\mathbf{r}_x - \Delta\mathbf{r}_{tot} + 0.5) , \tag{3.2}$$

where INT truncates the expression in parentheses to an integer. $\Delta\mathbf{r}_{tot}$ then becomes

$$\Delta\mathbf{r}_{tot} = \Delta\mathbf{r}_{tot} + \Delta\mathbf{r}_{int} . \tag{3.3}$$

3.4 Boundaries

The desired behavior of the boundaries is discussed in §2.4. The theory is fairly straight forward, but programming reservoir boundaries is another matter. In all, the boundaries were allotted the four top and bottom rows of Rspace (Fig. 3.1). These four rows were further broken down into two rows of hard and soft boundaries. Particles in the hard rows were only represented in the Rspace array and did not exist in the Mlist. Particles in the soft boundaries were represented in both Rspace and Mlist.

The algorithm ran by stepping through Mlist. Thus, the particles in the hard boundaries were never evaluated and never moved except by differential rotation. The purpose of the hard boundaries was to give the particles in the soft boundaries a complete set of nearest neighbors. The particles in the soft boundaries were evaluated. In general only one type of motion was allowed: particles leaving the soft boundary and moving into the "free" zone. Since the boundaries were supposed to be reservoirs, they should be immutable. All intra-boundary motion was ignored. Pushing a particle into the free zone was handled by creating a new particle in the destination hex and adding it on to the end of Mlist. Likewise, pulling a free particle into the soft boundary resulted in subtracting its mass from its original hex and deleting it from Mlist.

Although these boundaries were by no means perfect, they served their function of keeping boundary effects small.

3.5 Adding Particles

There are three distinct regions in Rspace: the upper boundary, the free zone, and the lower boundary. Let the density, ρ, be defined as

$$\rho = N_p/H \,, \tag{3.4}$$

where N_p = the total number of particles. The program was written so that ρ could be set differently for each region. Choosing ρ determines the number of particles in a region, but it says nothing about the distribution. As in the case of the velocity (where there are many different velocity curves to choose from), there are a large number of reasonable distributions. The two used were a random distribution and a uniform distribution. By random, it is meant that the coordinates of each particle were calculated using a random number generator (Park & Miller 1988). One of the nice features of this method was that it easily accommodated different numbers of particles. *i.e.* The position of a new particle does not depend upon the positions of previous particles. The uniform distribution placed one particle in each hex. This distribution was limited to $\rho = 1$.

The particles were dropped into Rspace one at a time by region. At the same time, the particles were added to Mlist. The order of particles in Mlist was also the order the algorithm evaluated them. To insure the particles were evaluated randomly Mlist was "shuffled".

3.6 Operation

The aforementioned ideas were combined into an interactive program written in Fortran. The code was very portable and ran successfully on Apple Macintosh, VMS and UNIX computers. The program started by presenting the user with the following list prompt of options and their default values:

```
Type in option or parameter to change.
(1)  Run Program
(2)  Quit Program
(3)  Total Hexes, H = 4000
```

```
(4)  Iterations = 100
(5)  Print Step = 1
(6)  Prompt Step = 10
(7)  Central Mass, Mo = 1.0 (Solar masses)
(8)  Mean Radius, R = 1.0 (A.U.)
(9)  Ring Width, W = 0.45 (A.U.)
(10) Upper Boundary Density = 0.0
(11) Free Region Density = 0.5
(12) Lower Boundary Density = 0.0
```

The most useful menu item for debugging purposes was (6)—Prompt Step. Every specified number of time steps, the program would stop and present the user with another list prompt:

```
Type in option or parameter to change.
(1)     Continue
(2)     Cancel
(3)     Iterations = 500
(4)     Print Step = 1
(5)     Prompt Step = 10
(6)     Show Mass Map
(7)     Write Mass Map To File
```

allowing the user to examine a simulation while it was running. Just the ability to monitor and cancel bad runs resulted in dramatic savings in debugging and simulation time.

The flow of the program is diagrammed in Fig. 3.3. The vast majority of the CPU time was spent in evaluating the movement condition and acting on it.

3.7 Monte Carlo Analogy

Another way to view this algorithm is as a Monte Carlo simulation of particle configurations in ring space. As in thermodynamics calculations, the probability of encountering a particular configuration depends on the interaction energy, or in this case $|\mathbf{f}_p|$.

Computationally, the two algorithms are very similar as well. In fact, a good deal of the computational motivation for the NNP model came from previous work done on 2D Ising systems (Kepner 1990). The concept of generating new configurations using nearest neighbor interactions is directly analogous. However, where a Monte Carlo simulation is more concerned with the long-term statistics of configurations, the NNP model focuses on the short term dynamics. Perhaps the best description of the NNP model is that it is a hybridization of Newtonian and stochastic models (Seiden & Schulman 1990).

3.8 Table and Figures for §3

TABLE I

Differential velocity data from a run with: $M_0 = 1.0$, R = 1.0, W = 0.4, H = 86 x 20 $\Rightarrow \Delta t = 0.39$ years. Multiplying $v_{rel}(r)$ by 0.079 gives the velocity in A.U./yr.

Row	r (A.U.)	$v_{rel}(r)$ (hex/yr)	$\Delta \mathbf{r}_x$ (hexes)
20	1.20	0.00	0.00
19	1.18	0.70	0.27
18	1.16	1.42	0.55
17	1.14	2.15	0.84
16	1.12	2.91	1.14
15	1.10	3.69	1.44
14	1.08	4.50	1.75
13	1.06	5.32	2.08
12	1.04	6.18	2.41
11	1.02	7.05	2.75
10	1.00	7.96	3.10
9	0.98	8.90	3.47
8	0.96	9.86	3.85
7	0.94	10.86	4.24
6	0.92	11.90	4.64
5	0.90	12.97	5.06
4	0.88	14.08	5.49
3	0.86	15.23	5.94
2	0.84	16.43	6.41
1	0.82	17.67	6.89

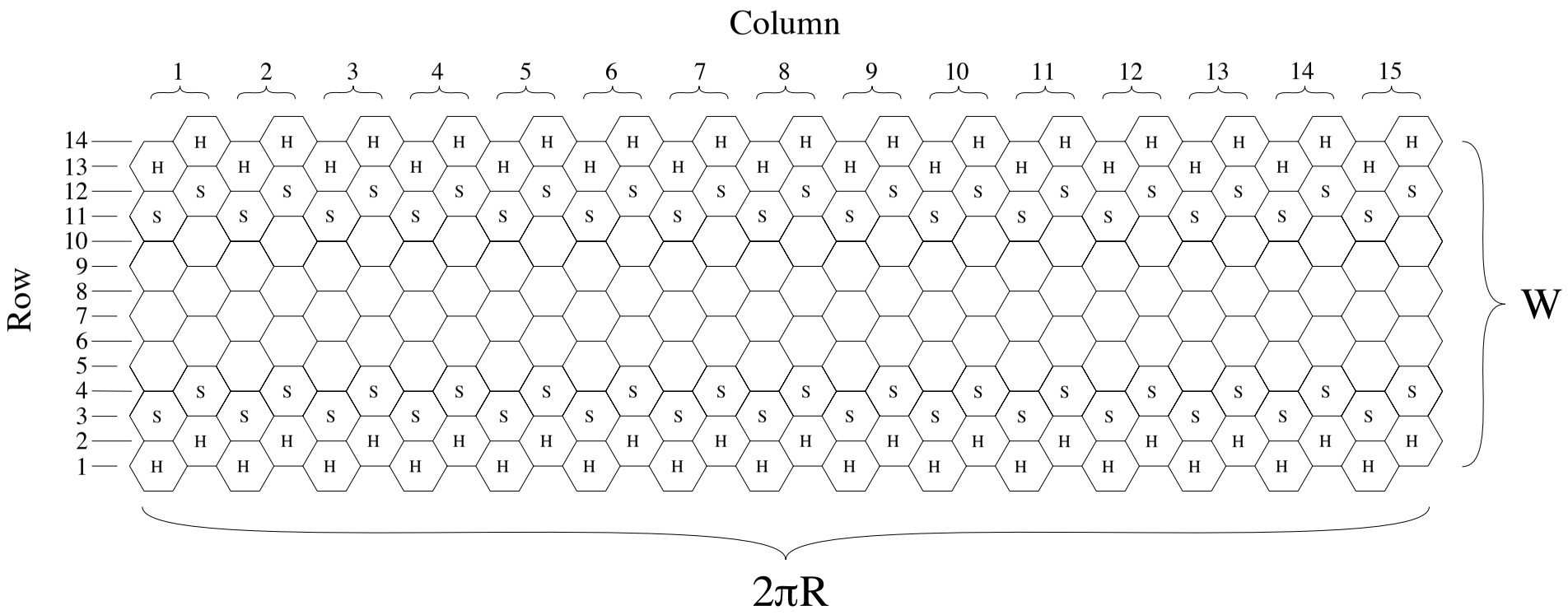


Fig. 3.1: Array referencing scheme for the hexagonal grid with W/R = 1.7 and H = 15x14 (i.e. a very wide ring). H and S denote the type of boundary. H = Hard boundaries—no particles enter or leave. S = Soft boundaries—particles may enter and leave.

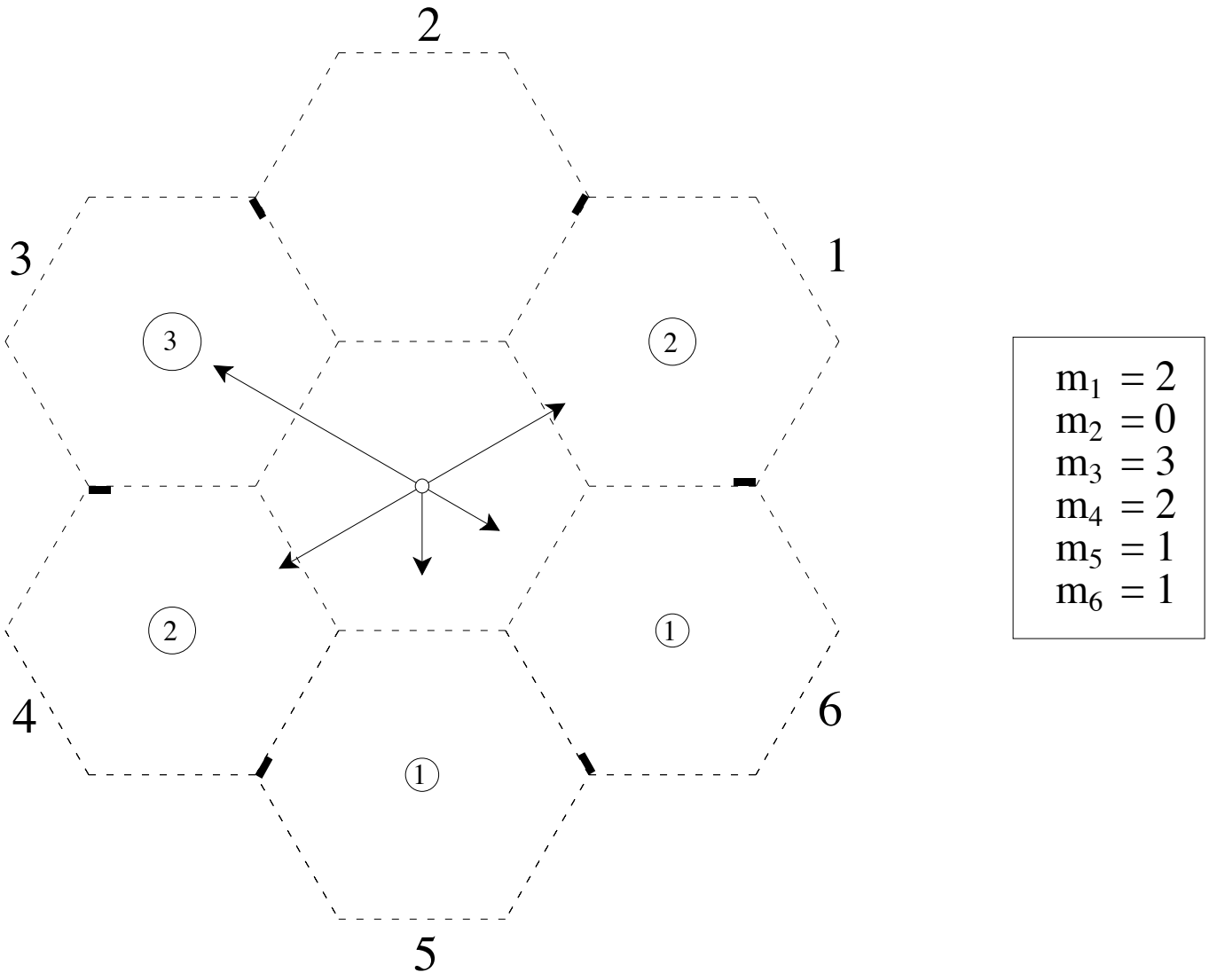


Fig. 3.2: Typical configuration of particles. The larger numbers on the outside are the nearest neighbor labels. In this case the vector mass on a particle in the center hex would be f = 1.414 in the -x direction. f lies on the border between 3 and 4. The convention for resolving border disputes is given by the black flags lying on the boundaries between hexes.

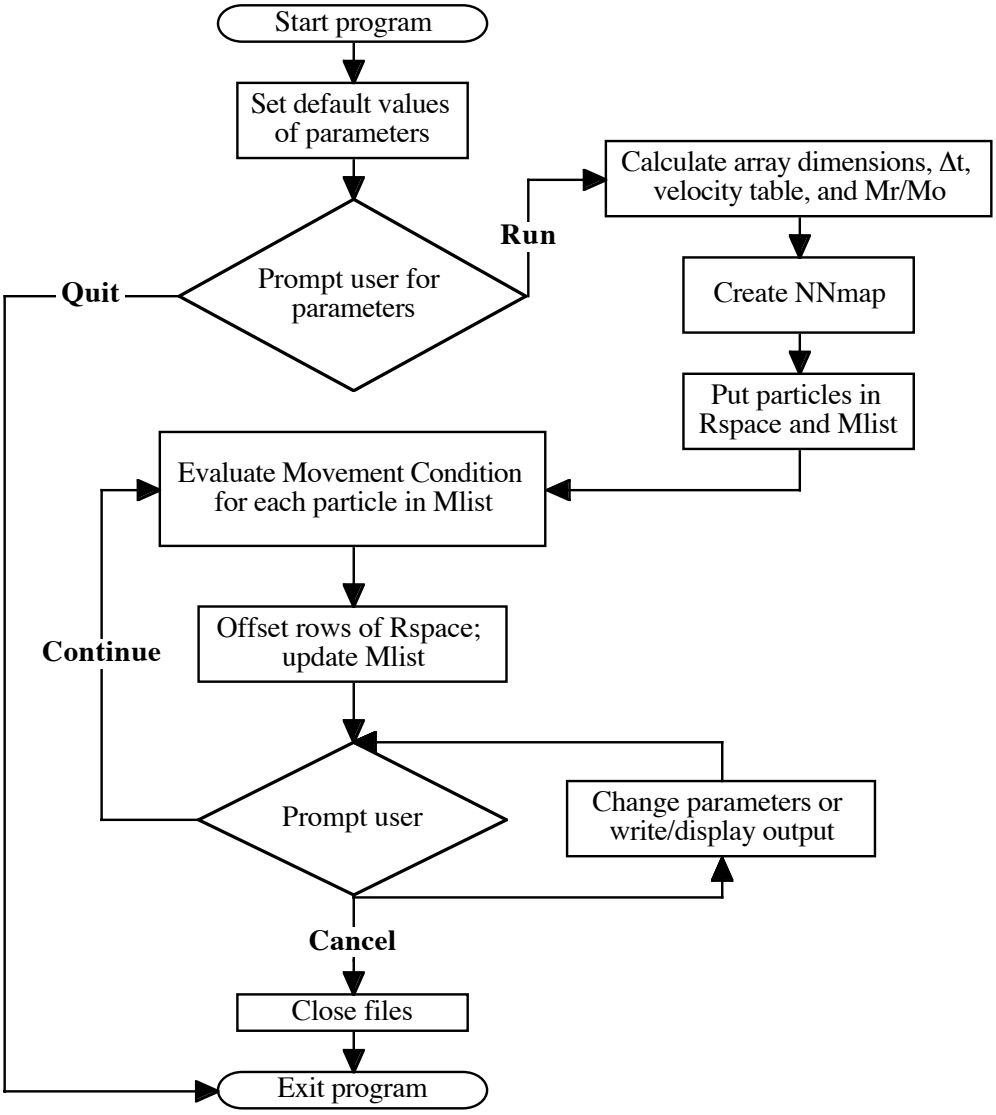


Fig. 3.3 : Flow chart of NNP program.

4. RESULTS

The program was executed on an Apple Macintosh SE/30 micro-computer and a VAX 6310 mainframe. The experiments can be broken up into three groups. The first set of experiments were of a general kind. This survey (§4.1) determined what types of astrophysical situations the program could address. The other two sets focused on particular aspects of the model. §4.2 describes the exploration of the quantitative outputs as a function of the density of particles. §4.3 presents a qualitative examination of how a disturbance propagates through the ring.

4.1 General Behavior

The first set of experiments were of a broad nature to determine the overall behavior of the algorithm. These preliminary runs helped sort out the wide variety of data that the program produced. Two main outputs were examined, the mean force per particle, $\langle|\mathbf{f}|\rangle$, and the transfer ratio, T_r. The mean force is calculated by taking the average of all the $|\mathbf{f}_p|$ at a given time step. The transfer ratio is the fraction of the total number of particles that moved during a particular time step. In other words, the number of particles that satisfied the movement condition divided by N_p.

Before examining $\langle|\mathbf{f}|\rangle$ and T_r, it is important understand the qualitative behavior of the algorithm. One way to explore the evolution is by decomposing the model. For example, asking the question "what would happen if there was no differential rotation?" The answer is "not much." Starting with an initial configuration of randomly distributed particles with $m_p = 1$ and empty boundaries† (Fig. 4.1a), the system evolves to a static situation in three time steps (Fig. 4.1b). Such a fast equilibrium indicates that motion is necessary for complex dynamics to occur. Adding differential rotation gives a slightly more interesting result. After approximately fifty

†This was the standard starting configuration for two reasons. First, the random distribution was a very general and very reproducible distribution. Second, whether or not the boundaries were filled or empty made no difference with this distribution.

time steps, the hexes with more particles have accreted most of the free particles (Fig. 4.1c). At this point the algorithm looses its physical validity. The large concentrations of particles would undoubtedly exert significant long range forces and the NNPs would be of little importance. It is interesting to note that the process of accretion is intrinsic to the algorithm.

Accretion is exhibited quantitatively as well. Fig 4.2 shows $\langle|\mathbf{f}|\rangle$ as a function of the time step††. For the first thirty time steps the system evolves in a consistent manner with $\langle|\mathbf{f}|\rangle$ increasing logarithmically. After accretion has occurred all that is left is a statistical system governed by the random encounters of the particle concentrations. The probability of encountering a particular configuration of particles then becomes a function of $\langle|\mathbf{f}|\rangle$ (Fig 4.3). Since this is beyond the physically valid regime, we will devote no more discussion to this long term behavior.

That the algorithm is related to a Monte Carlo simulation is partially demonstrated by the behavior of T_r (Fig. 4.4). T_r follows the logarithmic decay curve that is typical of Monte Carlo algorithms (Binder 1984).

Finally, I would like to say a few words about computational performance. One of the goals was to write a program that was linear in N_p. Often in dealing with these kind of phenomena, the programs are $O(N_p^2)$ or $O(N_p \text{ Log } N_p)$ (Monaghan 1990)†, which can get costly very quickly. Fig 4.5 demonstrates that indeed the program is linear and efficient. Especially when one considers that it was quite feasible to do calculations involving 100,000 particles on a micro-computer.

The general behavior of the algorithm can be summarized as follows. Starting with a random distribution, the configurations steadily move towards a situation in which most of the particles

---

††The computational parameters (M_0, R, W, and H) given for Fig 4.2 were the same as those that were used throughout the experiments of §4.1 and §4.2. These parameters were used to generate the data shown in Figs. 4.3-4.8.

are found in a few hexes. This process of accretion shows up as a logarithmically increasing force per particle. Accretion is completed in about fifty time steps. The concentration of particles causes the model to loose its physical validity. Thus, there is no physical justification in studying its behavior beyond this point.

4.2 Quantitative Analysis

In this section a closer look was given to the short term values of $\langle|\mathbf{f}|\rangle$ and T_r. As mentioned in the previous section, both $\langle|\mathbf{f}|\rangle$ and T_r are approximately logarithmic during the first twenty to thirty time steps (Fig 4.6). How these curves change with ρ was the focus of the next set of experiments. ρ and the mass ratio, M_r/M_0, are related by

$$M_r/M_0 = \alpha\rho \,, \tag{4.1}$$

where $\alpha = 6.3 \times 10^{-2}$ hexes/particle for this series of simulations. Varying ρ is then equivalent to varying the mass of the ring, M_r.

Ten runs were carried out over the range $0.1 \le \rho \le 1.9$ ($0.63 \times 10^{-3} \le M_r/M_0 \le 12.0 \times 10^{-3}$). These data were fit to

$$\langle|\mathbf{f}|\rangle = A + B\,\mathrm{Log}(t) \,, \tag{4.2a}$$

and

$$T_r = C - D\,\mathrm{Log}(t) \,. \tag{4.3a}$$

A and B, and C and D are plotted in Fig. 4.7. From Fig. 4.7a it is fairly clear that $A(\rho) \approx B(\rho) \approx 4.3\,\rho$. Using this result, it is possible to simplify Eq. (4.2a)

$$\langle|\mathbf{f}|\rangle = 4.3\,\rho\,(1 + \mathrm{Log}(t)) \,. \tag{4.2b}$$

Unfortunately, there is no such simplification for Eq. (4.3a). Although, as ρ becomes large, it is safe to say that

$$T_r \rightarrow \approx 1.0 - 0.5\,\mathrm{Log}(t) \,. \tag{4.3b}$$

---

†A notable exception is the fast multipole method which is O(Np) (Greengard & Rokhlin 1987).

Eqs. (4.2b) and (4.3b) suggest that while the mean force on the particles increases linearly as more mass is added, the transfer ratio levels off.

Up to this point, all the experiments have been carried out with a motion constant of $C_M = 1$. Suppose C_M were varied. To explore this possibility, four runs at $\rho = 0.9$ were carried out for $C_M = 1$, 2, 3, and 4. The transfer ratios for the first thirty time steps of each run are shown in Fig. 4.8. The effect of increasing C_M is to reduce the initial transfer rate. It is even possible to reduce C_M to the point that the transfer rate starts out increasing rather than decreasing.

4.3 Uniform Distribution with a Disturbance

In contrast to the previous section, which examined the quantitative behavior of a large number of randomly distributed particles, this section will focus on the qualitative dynamics of a small number of orderly distributed particles. The goal will be to determine how a uniformly distributed ring responds to a small disturbance. Fig 4.9 shows the initial configuration. Every hex, including the boundaries, is filled with one particle. Three additional particles are located in a hex corresponding to the point in space: $r = R$, $\theta = 0$. Figs. 4.10, 4.11, 4.12† show how the ring evolved for three different motion constants: $C_M = 1$, 2, and 3. In the first case ($C_M = 1$), the disturbance causes accretion to propagate quickly through the disk, even outrunning the differential rotation. In the second case ($C_M = 2$), the disturbance moves more slowly, allowing the differential rotation to create a spiral pattern. In the third case ($C_M = 3$), the disturbance creeps along. Only those particles that are adjacent to the disturbance are perturbed, and a ringlet develops.

4.5 Figures for §4

†The computational parameters used to generate the data shown in Figs. 4.9, 4.10, and 4.11 were the same as those used in Fig. 4.1.

Fig 4.2: Mean "force", $\langle|\mathbf{f}|\rangle$, for 1,000 time steps. Parameters were as follows: M_0 = 1.0 Solar masses, R = 1.0 A.U., W = 0.45 A.U., H = 601x150, and $\rho = 0.9 \Rightarrow$ N_p = 76,000, Δt = 0.38 years, and M_r/M_0 = 0.0060. Initially, $\langle|\mathbf{f}|\rangle$ increases logarithmically with time. After accretion has occurred, the system becomes a statistical one.

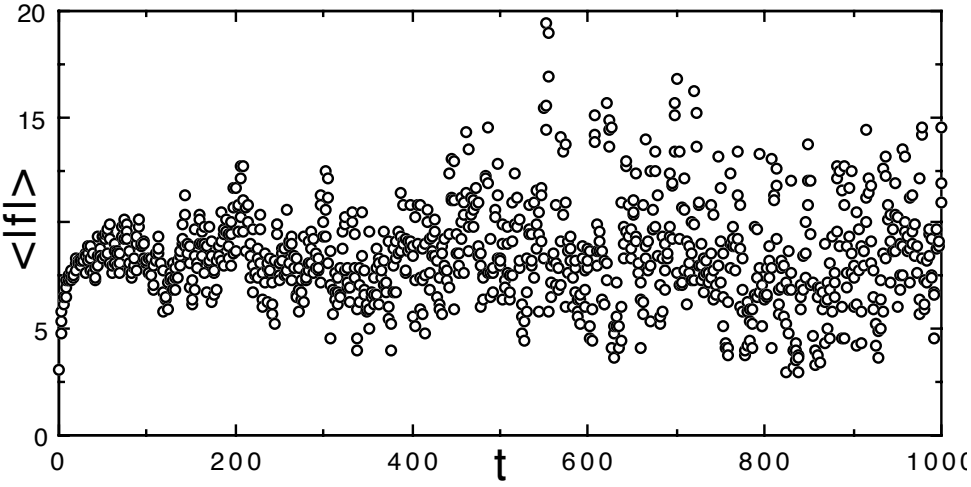

20
15
5
0
<|f|>
0
200
400
600
800
1000
t

Fig 4.3: Histogram of the first 1100 configurations illustrating that the algorithm becomes probabilistic after accretion has occurred. With the probability of encountering a particular configuration, C, being some statistical function such as

$$\text{Prob}(C) \sim \exp[\, -(\langle|\mathbf{f}|\rangle - f_{avg})^2 \,] ,$$

where $\langle|\mathbf{f}|\rangle$ is the mean vector mass of C and f_{avg} is the mean value of all the $\langle|\mathbf{f}|\rangle$.

Fig 4.4: Transition ratio, T_r, vs. time step. The number of particles satisfying the movement condition decreases logarithmically with time for the first 200 time steps before it begins to level off.

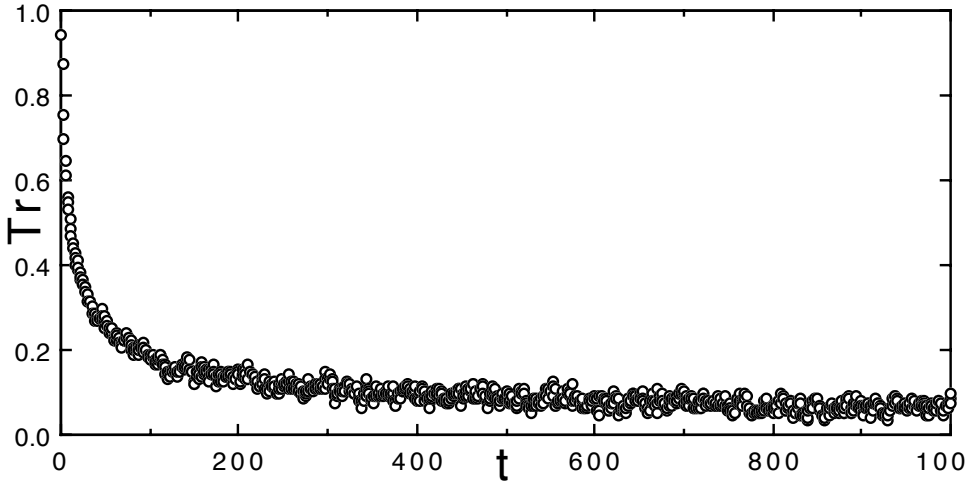

1.0
0.8
0.6
0.4
0.2
0.0
Tr
0
200
400
t
600
800
100

Fig 4.5: Computation time vs. N_p. Data taken from runs done on a Macintosh SE/30 with 4 MBytes of RAM. The graph shows that CPU time is proportional to N_p.

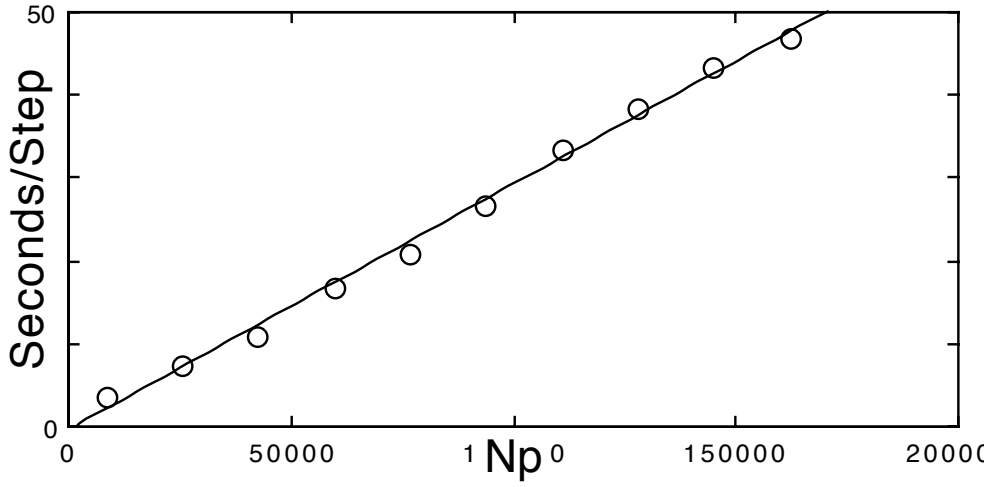

50
0
Seconds/Step
0
50000
1
Np
0
150000
20000

Fig. 4.6: Short term behavior of the NNP model ($\rho = 0.7$, 20 steps). **a**) $\langle|\mathbf{f}|\rangle$ vs. t fit to A + B Log(t). **b**) T_r vs. t fit to C - D Log(t).

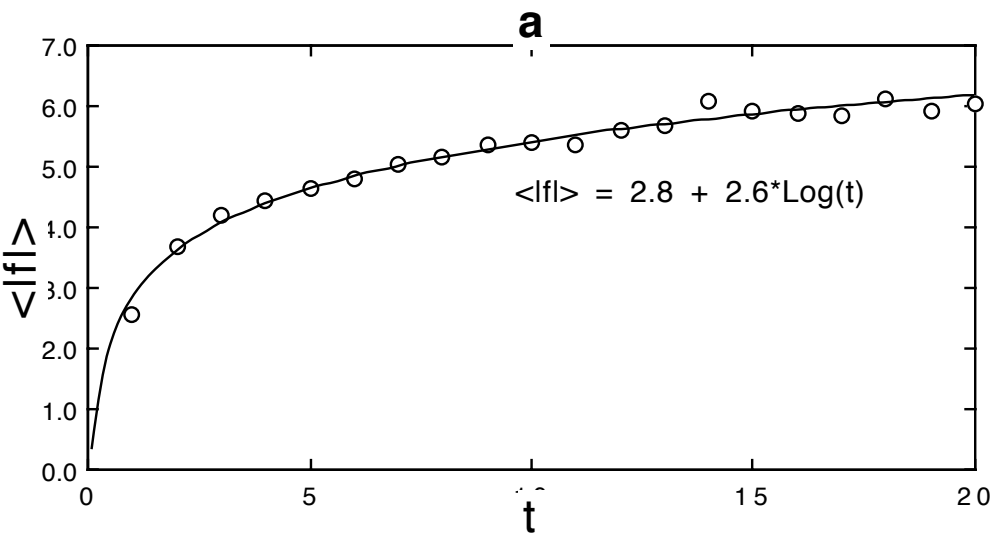

a
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0.0
<|f|>
<|f|> = 2.8 + 2.6*Log(t)
0
5
15
20
t

Fig. 4.7: Curve fit values as a function of density. **a**) A and B vs. ρ. **b**) C and D vs. ρ.

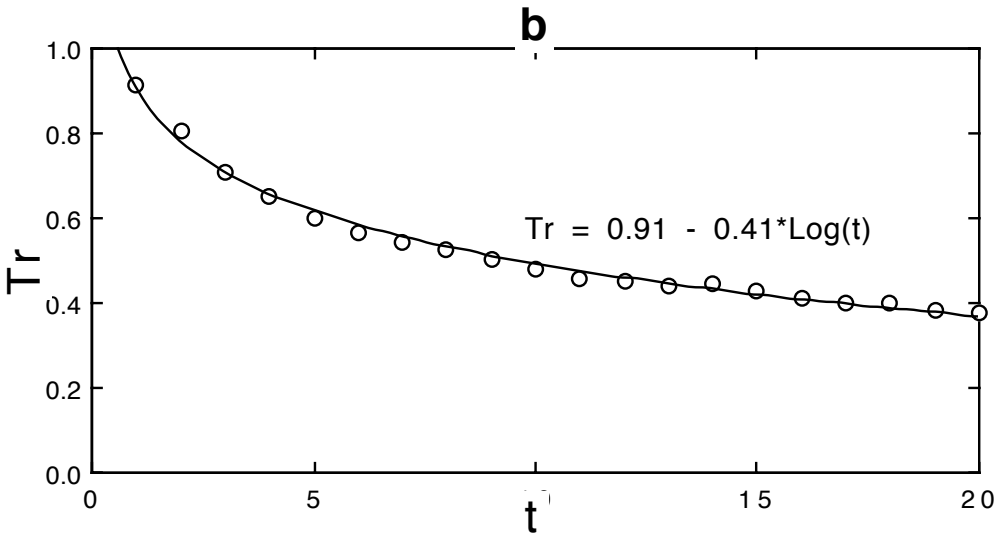
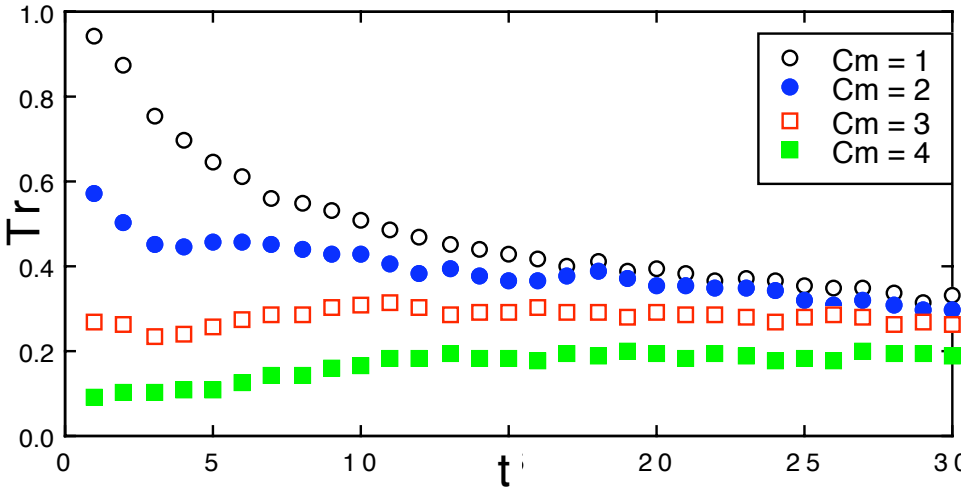

b
1.0
0.8
0.6
0.4
0.2
0.0
Tr
Tr = 0.91 - 0.41*Log(t)
0
5
15
20
t

Fig. 4.8: T_r vs. t ($\rho = 0.9$) for different motion constants: $C_M = 1, 2, 3, 4$.


1.0
0.8
0.6
0.4
0.2
0.0
Tr
0
5
10
20
25
Cm = 1
Cm = 2
Cm = 3
Cm = 4

5. DISCUSSION

In this section, the results of §4 are placed into a physical context. First, a general evaluation of the NNP algorithm is given with a discussion of the advantages and disadvantages (§5.1). §5.2 applies the data presented in §4.2 to the accretion rates of disks around stars. §5.3 briefly states some of the implications of §4.3. Throughout §5 a rather liberal approach is taken in comparing the NNP model to what has been observed. The thin physical basis of the model means that it is very difficult to rigorously prove its correspondence with nature. But this was never the goal. The goal has always been to see what new ideas the NNP model could provoke.

5.1 Properties of the Model

Computationally, the NNP algorithm has several advantages over standard Newtonian methods. The most significant being large values of Δt, making it possible to see a considerable amount of evolution with relatively few steps. Yet, Δt is small enough that the simulations maintain their continuity from step-to-step. Other time saving features of the algorithm are:

- Calculations are linear with the number of particles *i.e.* $O(N_p)$.
- Integer basis increases speed and reduces memory usage.
- Grid representation provides faster I/O.

Methodologically, the algorithm has several properties that give a different perspective on astrophysical disks. The utility of the scaling of ring space has already been discussed. Imposing the velocity curve also has interesting ramifications. The experiments carried out in this paper only required differential rotation. Many additional experiments could be carried out exploring different velocity curves. The idea of treating the boundaries as reservoirs is not an integral part of the NNP model, but the ability to simulate a small portion of a disk *is*. This feature allows a range of phenomena to be addressed that would otherwise require much larger systems.

The price of these new tools is sacrificing the concept of angular momentum transport. This has traditionally been an anchor of Newtonian modeling techniques, and a great deal has been learned by just applying conservation of angular momentum (Tremaine 1989). Additionally, limiting the interaction to nearest neighbors means that the model looses its validity when concentrations of mass develop. This limits the number of time steps in a typical simulation to ~ 50.

5.2 Interpretation of Quantitative Experiments

The main result of the experiments carried out in §4 is that the behavior of the NNP model is driven by accretion. Figs. 4.1 b and c show that motion is a key part of this process. Accretion pushes the system in such a way that the fraction of particles being perturbed steadily decreases (Fig. 4.4 and Fig. 4.6b). This is due to the sweeping up of individual particles by larger concentrations of particles. More concentrations causes the mean force on the particles to steadily increase (Fig. 4.6a).

The key parameter determining the rate at which accretion occurs appears to be the mass in the ring. Experiments were conducted over the range $0.63 \times 10^{-3} \leq M_r/M_0 \leq 12.0 \times 10^{-3}$, with $M_0 = 1.0\ M_o$, R = 1 AU, W = 0.45 AU, and $C_M = 1$. These parameters correspond to part of an accretion disk around a star. Observed values for T Tauri stars ($M_0 \sim 1.0\ M_o$) are $M_{disk} \sim 0.01 - 0.1\ M_o$ and have radii from 10 - 100 AU (Strom, Edwards & Strom 1989). Assuming that the density drops off as roughly $1/r^2 \Rightarrow M(r) \propto r$ (a gross assumption), the typical mass for a 0.45 AU ring is $\sim 0.45 \times 10^{-3}\ M_o$. Thus, these experiments correspond to slightly denser than average disks.

T_r indicates what fraction of the total particles moved, and is a measure of the rate of accretion. At the lower end, adding mass causes the rate of accretion to speed up (Fig. 4.7b). The mean force between particles increases linearly with the mass (Fig. 4.7a). What is unexpected is that the accretion rate levels off. Whether or not this reflects a physical situation is difficult to say. It would suggest an upper limit on how quickly accretion can take place.

Increasing C_M is equivalent to decreasing the mass in the ring. To understand how the system behaves with even less mass, it is easiest to vary C_M. Transition ratios for $M_r \approx 0.57 \times 10^{-3}\ M_o$, $0.28 \times 10^{-3}\ M_o$, $0.19 \times 10^{-3}\ M_o$, and $0.14 \times 10^{-3}\ M_o$ are shown in Fig. 4.7. As the mass decreases, the initial rate of accretion drops dramatically. In fact, T_r gets to the point where it starts out increasing. It is interesting to note that this occurs in the observed range of disk masses. Interpreting these results to their fullest suggests that accretion disks form when less mass is available and the rate of evolution is level or gently increasing. Adding mass causes disks to accrete too quickly to be observed. Finally, there is an upper limit for how quickly the disk can be swept up.

These results can be scaled to planetary disks as well. In the galactic case, there is the problem of the phase transition due to star formation. How this should be handled is not exactly clear, and the above interpretations should be applied with care.

5.3 Interpretation of Qualitative Experiments

The effects of varying C_M on the propagation of a disturbance is shown in Figs. 4.10, 4.11, and 4.12. In each case the ring evolves in a different way. In Fig. 4.10, the particles are swept up quickly, illustrating how easily the process of accretion is initiated when there is enough mass. Lowering the mass by increasing C_M causes the disturbance to be sheared by differential rotation. Hence the characteristic spiral pattern. The rate of propagation is slow in real terms as well. The spiral in Fig. 4.11b took approximately six years, or periods, to develop. In a larger system, it would not be inconceivable for this time frame to multiplied by an order of magnitude. In other words, the spirals live long enough that they could be seen on galactic time scales. Furthermore, adding arms to the spiral is trivial, just add more disturbances. In Fig. 4.12 C_M is so large that very few particles are perturbed. Very slowly the particles nearest to the disturbance are swept up. The final ringlet that develops is quite reminiscent of the structures seen in planetary rings (Smith *et al* 1981). Using combinations of strategically placed disturbances, it is not difficult to imagine how a whole array of ringlets, arcs, gaps and double rings might be produced.

These results indicate that a wide variety of phenomena can be produced by changing only one parameter, C_M. That C_M corresponds to the mass is very thought provoking, implying a link between accretion disks, the spiral arms of galaxies, and planetary rings.

6. CONCLUSIONS

The answer to the question of can NNPs effect the behavior of disks is an unqualified "yes". In fact, many disk phenomena can be simulated with the NNP model. Accretion, spiral arm structure, and finer scale planetary ring structure can all be obtained, and are related by the amount of mass in the disk. Also tied to the disk mass is the rate at which accretion occurs. Less massive disks accrete slowly, while more massive ones accrete quickly, but only to a point. From a computational standpoint the NNP algorithm produces these results quite efficiently. Complete simulations with 10^5 particles can be performed in minutes on a micro-computer.

This paper also addresses the question of how basic a model is too basic. Claims of rigorous results from simple models should be heavily scrutinized. This is not to say that such models cannot be useful tools for provoking new insights. Never-the-less, as is shown here, the very simplest of models can give impressive results.

As far as further research is concerned, imposing different velocity curves is an obvious topic. Other possibilities include:

- Studying different mass distributions.
- Changing the geometry to encompass whole disks.
- Extending the interaction length to cover second nearest neighbors.
- Optimizing the code to its theoretical limit (6 bytes/particle) and running simulations of $O(10^7)$ particles.
- Combining the NNP model with a phase transition model (Seiden & Schulman 1990) to get a more complete picture of galaxies.

It is apparent that there are many situations to which NNP be can be applied. In this light, the research here can be viewed as a "proof-of-concept", and warrants exploring the further application of this model.

7. REFERENCES

ARTYMOWICZ, P. & Lubow, S. H., 1989, "The Smoothed Particle Hydrodynamics of Galactic Discs", In *Dynamics of Astrophysical Discs*, edited by J. A. Sellwood, Cambridge University Press, Cambridge, p. 211.

BENZ, W., BOWERS, R. L., CAMERON, A. G. W. & PRESS, W. H., 1990, "Dynamic Mass Exchange in Doubly Degenerate Binaries I. 0.9 and 1.2 M_O Stars", *Astrophysical Journal*, **348**, No. 2, Part 4, p. 647.

BINDER, K., 1984, "A Simple Introduction to Monte Carlo Simulations and Some Specialized Topics", In *Applications of the Monte Carlo Method in Statistical Physics*, edited by K. Binder, Springer-Verlag, Berlin, p. 1.

FRENCH, R. G., ELLIOT, J. L., FRENCH, L. M., KANGAS J. A., MEECH, K. J., RESSLER, M. E., BUIE, M. W., FROGEL, J. A., HOLBERG, J. B., FUENSALIDA, J. J. & JOY, M., 1988, "Uranian Ring Orbits from Earth-Based and Voyager Occultation Observations", *Icarus*, **73**, p. 349.

GINGOLD, R. A. & MONAGHAN, J. J., 1982, "Kernel Estimates as a Basis for General Particle Methods in Hydrodynamics", *Journal of Computational Physics*, **46**, p. 429.

GREENGARD, L. & ROKHLIN, V., 1987, "A Fast Algorithm for Particle Simulations", *Journal of Computational Physics*, **73**, p. 325.

KEPNER, J. V., 1990, "Canonical vs. Micro-Canonical Sampling Methods in a 2D Ising Model", Lawrence Berkeley Lab report, LBL-30056 (December 1990).

LORENZ, E., 1979, "Predictability: Does the Flap of a Butterfly's Wings in Brazil Set Off a Tornado in Texas", Address at the annual meeting of the American Association for the Advancement of Science (29 December 1979).

MIHALIS, D. & BINNEY, J., 1981, *Galactic Astronomy—Structure and Kinematics*, 2nd edition, W. H. Freeman and Company, New York, p. 285.

MONAGHAN, J. J., 1990, "Modelling the Universe", *Proceedings of the Astronomical Society of Australia*, **8**, No. 3, p. 233.

PARK, S. K. & MILLER, K. W., 1988, "Random Number Generators: Good Ones are Hard to Find", *Communications of the ACM*, **31**, No. 10, p. 1192.

SELLWOOD, J. A., 1989, "Spiral Instabilities in N-Body Simulations", In *Dynamics of Astrophysical Discs*, edited by J. A. Sellwood, Cambridge University Press, Cambridge, p. 155.

SELLWOOD, J. A. & CARLBERG, R. G., 1984, "Spiral Instabilities Provoked by Accretion and Star Formation", *Astrophysical Journal*, **282**, p. 61.

SHU, F. H., CUZZI, N. C. & LISSAUER, J. J., 1983, "Bending Waves in Saturn's Rings", *Icarus*, **53**, p. 118.

SMITH, B. A. *et al*, 1981, "Encounter with Saturn: Voyager 1 Imaging Science Results", *Science*, **212**, p. 163.

SNELL, R. L., 1989, "Observations of Discs Around Protostars and Young Stars", In *Dynamics of Astrophysical Discs*, edited by J. A. Sellwood, Cambridge University Press, Cambridge, p. 49.

STROM, S. E., EDWARDS, S. & STROM, K. M., 1989, "Constraints on the Properties and Environment of Primitive Stellar Nebulae from the Astrophysical Record Provided by Young Stellar Objects", In *The Formation and Evolution of Planetary Systems*, edited by H. A. Weaver and L. Danly, Cambridge University Press, Cambridge, MA, p. 91.

TOOMRE, A., 1981, "What Amplifies the Spirals?", In *Structure and Evolution of Normal Galaxies*, edited by S. M. Fall and D. Lynden-Bell, Cambridge University Press, Cambridge, p. 111.

TREMAINE, S., 1989, "Common Processes and Problems in Disc Dynamics", In *Dynamics of Astrophysical Discs*, edited by J. A. Sellwood, Cambridge University Press, Cambridge, p. 231.