ELSEVIER

New Astronomy 6 (2001) 277–291

New Astronomy

www.elsevier.com/locate/newast

NBODY2: A direct *N*-body integration code

Sverre J. Aarseth

Institute of Astronomy, Madingley Road, Cambridge CB3 0HA, UK

Received 1 February 2001; received in revised form 5 May 2001; accepted 5 May 2001
Communicated by L.E. Hernquist

Abstract

We give a full description of the code *NBODY*2 for direct integration of the gravitational *N*-body problem. The method of solution is based on the neighbour scheme of Ahmad and Cohen [J. Comput. Phys. 12 (1973) 389] which speeds up the force calculation already for quite modest particle numbers. Derivations of all the relevant mathematical expressions are given, together with a detailed discussion of the algorithms. The code may be used to study a wide variety of self-consistent problems based on a small softening of the interaction potential. © 2001 Published by Elsevier Science B.V.

PACS: 95.10.Ce; 98.10.+z; 98.65.Fz
Keywords: Methods: *N*-body simulations

1. Introduction

N-body simulations provide a useful tool for exploring a wide range of astronomical problems. By now, direct summation methods have been used to study such diverse topics as planetary formation, few-body scattering, star cluster dynamics, violent relaxation, tidal disruption of dwarf galaxies, interacting galaxies, compact groups and clusters of galaxies, as well as cosmological modelling. The code *NBODY*2 described here is suitable for relatively small *N*-body systems ($20 < N < 10^4$). However, it does not contain any special treatments of close encounters and should therefore only be used for problems where the effect of hard binaries can be neglected (a variety of regularization methods are discussed in Aarseth, 1985).

2. Integration method

In this section, we provide all the relevant mathematical expressions and discuss the basic principles of *N*-body integration.

2.1. Difference formulation

We write the equation of motion for a particle of index i in the form

$$\ddot{\mathbf{r}}_i = -G \sum_{j=1;\, j\neq i}^{N} \frac{m_j (\mathbf{r}_i - \mathbf{r}_j)}{(r_{ij}^2 + \epsilon^2)^{3/2}}. \tag{1}$$

Here G is the gravitational constant and the summation is over all the other particles of mass m_j and coordinates $\mathbf{r}_j$. The introduction of a softening parameter ϵ prevents a force singularity as the mutual separation $r_{ij} \to 0$, thereby making the numerical solutions well behaved. For convenience, we adopt scaled units in which $G = 1$ and define the

E-mail address: sverre@ast.cam.ac.uk (S.J. Aarseth).

1384-1076/01/$ – see front matter © 2001 Published by Elsevier Science B.V.
PII: S1384-1076(01)00060-4

left-hand side of (1) as the force *per unit mass*, $\mathbf{F}$, omitting the subscript. The present difference formulation is based on the notation of Ahmad and Cohen (1973 hereafter AC) and follows closely an earlier treatment (Aarseth, 1985).

Given the values of $\mathbf{F}$ at four successive past epochs t_3, t_2, t_1, t_0, with t_0 the most recent, we write a fourth-order fitting polynomial at time t as

$$\mathbf{F}_t = \big(\big((\mathbf{D}^4(t-t_3)+\mathbf{D}^3)(t-t_2)+\mathbf{D}^2\big)(t-t_1) + \mathbf{D}^1\big)(t-t_0)+\mathbf{F}_0. \tag{2}$$

Using compact notation, the first three divided differences are defined by

$$\mathbf{D}^k[t_0,t_k] = \frac{\mathbf{D}^{k-1}[t_0,t_{k-1}]-\mathbf{D}^{k-1}[t_1,t_k]}{t_0-t_k}, \quad (k=1,2,3) \tag{3}$$

where $\mathbf{D}^0 \equiv \mathbf{F}$ and square brackets refer to the appropriate time intervals ($\mathbf{D}^2[t_1, t_3]$ is evaluated at t_1). The term $\mathbf{D}^4$ is defined similarly by $\mathbf{D}^3[t, t_2]$ and $\mathbf{D}^3[t_0, t_3]$. Conversion of the force polynomial into a Taylor series provides simple expressions for integrating coordinates and velocities. Equating terms in the successive time derivatives of (2) with an equivalent Taylor series and setting $t = t_0$ yields the force derivatives

$$\begin{aligned}
\mathbf{F}^{(1)} &= \big((\mathbf{D}^4 t_3' + \mathbf{D}^3)t_2' + \mathbf{D}^2\big)t_1' + \mathbf{D}^1 \\
\mathbf{F}^{(2)} &= 2!\big(\mathbf{D}^4(t_1't_2' + t_2't_3' + t_1't_3') + \mathbf{D}^3(t_1' + t_2') + \mathbf{D}^2\big) \\
\mathbf{F}^{(3)} &= 3!\big(\mathbf{D}^4(t_1' + t_2' + t_3') + \mathbf{D}^3\big) \\
\mathbf{F}^{(4)} &= 4!\mathbf{D}^4,
\end{aligned} \tag{4}$$

where $t_k' = t_0 - t_k$. Eq. (4) is mainly used to obtain the Taylor series derivatives at $t = t_0$, when the fourth difference is not yet known. Thus the contribution from $\mathbf{D}^4$ to each order is only added at the end of an integration step. This so-called 'semi-iteration' gives increased accuracy at little extra cost (on scalar machines) and no extra memory.

We now describe the initialization procedure, assuming one force polynomial. From the initial conditions $m_j, \mathbf{r}_j, \boldsymbol{v}_j$, the respective Taylor series derivatives are formed by successive differentiations of (1). Introducing the relative coordinates $\mathbf{R} = \mathbf{r}_i - \mathbf{r}_j$ and the relative velocities $\mathbf{V} = \boldsymbol{v}_i - \boldsymbol{v}_j$, all pair-wise interaction terms in $\mathbf{F}$ and $\mathbf{F}^{(1)}$ are first obtained by

$$\begin{aligned}
\mathbf{F}_{ij} &= -\frac{m_j\mathbf{R}}{R^3} \\
\mathbf{F}_{ij}^{(1)} &= -\frac{m_j\mathbf{V}}{R^3} - 3a\mathbf{F}_{ij},
\end{aligned} \tag{5}$$

with $a = \mathbf{R}\cdot\mathbf{V}/R^2$. The total contributions are obtained by summation over N. Next, the mutual second- and third-order terms are formed from

$$\begin{aligned}
\mathbf{F}_{ij}^{(2)} &= -\frac{m_j(\mathbf{F}_i - \mathbf{F}_j)}{R^3} - 6a\,\mathbf{F}_{ij}^{(1)} - 3b\mathbf{F}_{ij} \\
\mathbf{F}_{ij}^{(3)} &= -\frac{m_j(\mathbf{F}_i^{(1)} - \mathbf{F}_j^{(1)})}{R^3} - 9a\mathbf{F}_{ij}^{(2)} - 9b\mathbf{F}_{ij}^{(1)} \\
&\quad - 3c\mathbf{F}_{ij},
\end{aligned} \tag{6}$$

with

$$\begin{aligned}
b &= \left(\frac{V}{R}\right)^2 + \frac{\mathbf{R}\cdot(\mathbf{F}_i - \mathbf{F}_j)}{R^2} + a^2 \\
c &= \frac{3\mathbf{V}\cdot(\mathbf{F}_i - \mathbf{F}_j)}{R^2} + \frac{\mathbf{R}\cdot(\mathbf{F}_i^{(1)} - \mathbf{F}_j^{(1)})}{R^2} + a(3b - 4a^2).
\end{aligned} \tag{7}$$

A second double summation then gives the corresponding values of $\mathbf{F}^{(2)}$ and $\mathbf{F}^{(3)}$ for all particles. This pair-wise boot-strapping procedure provides a convenient starting algorithm, since the extra cost is usually small.

Appropriate initial time-steps Δt_i are now determined, using the general criterion discussed in the next subsection. Setting $t_0 = 0$, the backward times are initialized by $t_k = -k\,\Delta t_i$ $(k = 1, 2, 3)$. Inversion of (4) to third order yields starting values for the divided differences,

$$\begin{aligned}
\mathbf{D}^1 &= \left(\frac{1}{6}\mathbf{F}^{(3)}t_1' - \frac{1}{2}\mathbf{F}^{(2)}\right)t_1' + \mathbf{F}^{(1)} \\
\mathbf{D}^2 &= -\frac{1}{6}\mathbf{F}^{(3)}(t_1' + t_2') + \frac{1}{2}\mathbf{F}^{(2)} \\
\mathbf{D}^3 &= \frac{1}{6}\mathbf{F}^{(3)}.
\end{aligned} \tag{8}$$

2.2. Individual time-steps

Stellar systems are characterized by a range in density which gives rise to different time-scales for significant changes of the orbital parameters. In order to exploit this feature, and economize on the expensive force calculation, each particle is assigned its own time-step which is related to the orbital time-scale. Thus the aim is to ensure the convergence of the force polynomial (2) with the minimum number of force evaluations. Since all interactions must be added consistently in a direct integration method, it is necessary to include a temporary coordinate prediction of the other particles. However, the additional cost of low-order predictions still leads to a significant saving since this permits arbitrarily large time-step ratios.

Following the polynomial initialization discussed above, the integration cycle itself begins by determining the next particle (i) to be advanced. Thus in general, $i = \min_j (t_j + \Delta t_j)$, where t_j is the time of the last force evaluation. It is convenient to define the present epoch (or 'global' time) by $t = t_i + \Delta t_i$, rather than relating it to the previous value.

The individual time-step scheme (Aarseth, 1963) uses two types of coordinates for each particle. We define primary and secondary coordinates, $\mathbf{r}_0$ and $\mathbf{r}_t$, evaluated at t_0 and t, respectively, where the latter are derived from the former by the predictor. In the present treatment where high precision is not normally required, we predict coordinates to order $\mathbf{F}^{(1)}$ by

$$\mathbf{r}_t = ((\tilde{\mathbf{F}}^{(1)}\,\delta t_j' + \tilde{\mathbf{F}})\,\delta t_j' + \mathbf{v})\,\delta t_j' + \mathbf{r}_0, \tag{9}$$

where $\tilde{\mathbf{F}}^{(1)} = \frac{1}{6}\mathbf{F}^{(1)}, \tilde{\mathbf{F}} = \frac{1}{2}\mathbf{F}$ and $\delta t_j' = t - t_j$ (with $\delta t_j' < \Delta t_j$). The coordinates and velocities of particle i are then improved to order $\mathbf{F}^{(3)}$ by standard Taylor series integration (cf. (4)), whereupon the current force may be obtained by direct summation. At this stage the four times t_k are updated to be consistent with the definition that t_0 denotes the time of the most recent force evaluation. New differences are now formed (cf. (3)), including $\mathbf{D}^4$. Together with the new $\mathbf{F}^{(4)}$, these correction terms are combined to improve the current coordinates and velocities to highest order, whereupon the primary coordinates are initialized by $\mathbf{r}_0 = \mathbf{r}_t$.

New time-steps are assigned initially for all particles and at the end of each integration cycle for particle i. We adopt the composite criterion

$$\Delta t_i = \left(\frac{\eta(|\mathbf{F}|\,|\mathbf{F}^{(2)}| + |\mathbf{F}^{(1)}|^2)}{(|\mathbf{F}^{(1)}|\,|\mathbf{F}^{(3)}| + |\mathbf{F}^{(2)}|^2)} \right)^{1/2}, \tag{10}$$

where η is a dimensionless accuracy parameter. For this purpose, only the last two terms of the first and second force derivatives in (4) are included. This expression ensures that all the force derivatives play a role and is also well defined for special cases (i.e. starting from rest or $|\mathbf{F}| \simeq 0$). Although successive time-steps normally change smoothly, it is prudent to restrict the growth by a stability factor (presently 1.2).

In summary, the scheme requires the following 30 variables for each particle: m, $\mathbf{r}_0$, $\mathbf{r}_t$, $\boldsymbol{v}_0$, $\mathbf{F}$, $\mathbf{F}^{(1)}$, $\mathbf{D}^1$, $\mathbf{D}^2$, $\mathbf{D}^3$, Δt, t_0, t_1, t_2, t_3. It is also useful to employ a secondary velocity, $\boldsymbol{v}_t$, for dual purposes.

2.3. Neighbour scheme

Having introduced the basic tools for direct integration, we now turn to the AC neighbour scheme. Here the main idea is to reduce the effort of evaluating the force contribution from distant particles by employing two polynomials based on separate time-scales. Splitting the total force on a given particle into an irregular and a regular component,

$$\mathbf{F} = \mathbf{F}_{\rm irr} + \mathbf{F}_{\rm reg}, \tag{11}$$

we can replace the full N summation in (1) by a sum over the n nearest particles together with a prediction of the distant contribution. This procedure can lead to a significant gain in efficiency, provided the respective time-scales are well separated and $n \ll N$.

To implement the AC scheme, we form a list for each particle containing all members inside a sphere of radius R_s. In addition, we include any particles within a surrounding shell of radius $2^{1/3} R_s$ satisfying $\mathbf{R} \cdot \mathbf{V} < 0.1\, R_s^2/\Delta T_i$, where ΔT_i denotes the regular time-step. This ensures that fast approaching particles are selected from the buffer zone.

The size of the neighbour sphere is modified at the end of each regular time-step when a total force summation is carried out. We have adopted a neighbour criterion mainly based on distance for con-

venience. However, some modification would be desirable for very large mass ratios (say >100). A selection criterion based on the local number density contrast has proved itself for a variety of problems, but may need modification for interacting subsystems or massive particles. To sufficient approximation, the local number density contrast is given by

$$C = \frac{2n_1}{N}\left(\frac{R_h}{R_s}\right)^3, \tag{12}$$

where n_1 is the current membership and R_h is the half-mass radius. In order to limit the range, we adopt a predicted membership

$$n_p = n_{\max}(0.04C)^{1/2}, \tag{13}$$

subject to n_p being within $[0.2n_{\max}, 0.9n_{\max}]$, with $n_{\max}$ denoting the maximum permitted value. The new neighbour sphere radius is then adjusted using the corresponding volume ratio, which gives

$$R_s^{\text{new}} = R_s^{\text{old}}\left(\frac{n_p}{n_1}\right)^{1/3}. \tag{14}$$

An alternative and simpler strategy is to stabilize all neighbour memberships on the same constant value $n_p = n_0$ (AC; Makino and Hut, 1988). A number of refinements are also included as follows:

- In order to avoid a resonance oscillation in R_s, $(n_p/n_1)^{1/6}$ is used in (14) if the predicted membership lies between the old and new value.
- R_s is modified by a radial velocity factor outside the core.
- The volume factor is only allowed to change by 25%, subject to a time-step dependent cut-off if $\Delta T_i < 0.01\,T_{\text{cr}}$ (T_{cr} is the crossing time).
- If $n_1 \leq 3$ and the neighbours are moving outwards, the standard sphere R_s is increased by 10%.

The gain or loss of particles is recorded when comparing the old and new neighbour list, following the re-calculation of the regular force. Regular force differences are first evaluated, assuming there has been no change of neighbours. This gives rise to the provisional new regular force difference

$$\mathbf{D}^1 = \frac{\mathbf{F}_{\text{reg}}^{\text{new}} - (\mathbf{F}_{\text{irr}}^{\text{old}} - \mathbf{F}_{\text{irr}}^{\text{new}}) - \mathbf{F}_{\text{reg}}^{\text{old}}}{t - T_0}, \tag{15}$$

where $\mathbf{F}_{\text{reg}}^{\text{old}}$ denotes the old regular force, evaluated at time T_0, and the net change in the irregular force is contained in the middle brackets. In the subsequent discussions, regular times and time-steps are distinguished using upper case characters. All current force components are obtained using the *predicted* coordinates (which must be saved), rather than the corrected values based on the irregular $\mathbf{D}^4$ term since otherwise (15) would contain a spurious force difference. The higher differences are formed in the standard way, whereupon the regular force corrector is applied (if desired).

A complication arises because any change in neighbours requires appropriate corrections of both force polynomials; i.e. using the principle of successive differentiation of (11). The respective Taylor series derivatives (5) and (6) are accumulated to yield the net change. Each force polynomial is modified by first adding or subtracting the correction terms to the corresponding Taylor series derivatives (4) (without the $\mathbf{D}^4$ term), followed by a conversion to standard differences using (8).

Implementation of the AC scheme requires the following additional set of regular variables: $\mathbf{F}_{\text{reg}}$, $\mathbf{D}^1$, $\mathbf{D}^2$, $\mathbf{D}^3$, ΔT, T_0, T_1, T_2, T_3, as well as the neighbour sphere radius R_s and neighbour list L (size $n_{\max}+1$).

3. Program structure

This section describes the structure of the code and defines the main input parameters and some of the options.

3.1. Routines

The program contains some 2307 lines of Fortran statements and another 1203 lines of comments and spaces. It is divided into 38 routines, many of which are optional and not required for standard simulations. All the routines are listed in Table 1, together with the main calling sequence and an outline of the purpose (# denotes an option).

Table 1
Names of routines

Routine	Called by	Description
BLOCK		Block data initialization
BODIES	*OUTPUT*	Output of binaries and particles (# 6 & 9)
CHECK	*OUTPUT*	Error control and restart (# 2 & 11)
CMCORR	*OUTPUT*	Centre of mass corrections (# 18)
COAL	*MAIN*	Inelastic two-body collision (# 12)
CORE	*OUTPUT*	Core radius and density centre (# 8)
CPUTIM	*OUTPUT* & *INTGRT*	CPU time in minutes (machine dependent)
DATA	*START*	Generation of initial conditions
DEFINE	*INPUT*	Definition of input parameters and counters
ENERGY	*OUTPUT* & *SCALE*	Total energy
ESCAPE	*OUTPUT*	Removal of escaping particles (# 13)
FPCORR	*REGINT*	Derivative corrections of force polynomials
*FPOLY*1	*START* & *COAL*	Force and first derivative
*FPOLY*2	*START* & *COAL*	Second and third force derivatives
INPUT	*START*	Main parameter input
INSERT	*INTGRT*	Insert of particle index in sequential list
INTGRT	*MAIN*	Decision-making and flow control
LAGR	*OUTPUT*	Lagrangian radii and half-mass radius (# 7)
MAIN		Master control flow
MODIFY	*MAIN*	Reading modified parameters at restart
MYDUMP	*MAIN* & *INTGRT*	*COMMON* save or copy (# 1 & 2)
NBINT	*INTGRT*	Irregular integration
NBLIST	*START* & *COAL*	Initialization of neighbour list
OUTPUT	*MAIN*	Main output and data save
*RAN*2	*DATA*	Portable random number generator
REGINT	*INTGRT*	Regular integration
REMOVE	*ESCAPE* & *COAL*	Removal of particle from *COMMON* arrays
SCALE	*START*	Scaling to new units
*SORT*2	*INTGRT* & *LAGR*	Sequential sorting of array
START	*MAIN*	Calling sequence for initial setup
STEPI	*INTGRT*	Time-step for irregular force polynomial
STEPS	*START* & *COAL*	Initialization of time-steps and differences
SUBSYS	*START*	Initial subsystems for binary test (# 17)
VERIFY	*INPUT*	Validation of main input parameters
*XTRNL*1	*REGINT* & *FPOLY*1	Force due to Plummer potential (# 15)
*XTRNL*2	*REGINT* & *FPOLY*1	Force due to logarithmic potential (# 15)
XVPRED	*OUTPUT* & *COAL*	Prediction of coordinates and velocities
ZERO	*START*	Initialization of global scalars

The program consists of four separate parts: input, output, initialization and integration. The basic routines are: *START*, *INPUT*, *DATA*, *SCALE* (input); *ENERGY*, *CORE*, *OUTPUT* (output); *NBLIST*, *FPOLY*1, *FPOLY*2, *STEPS* (initialization); and *INTGRT*, *NBINT*, *REGINT*, *STEPI* (integration). The main calling sequence (*START*, *OUTPUT*, *INTGRT*) originates in routine *MAIN*, whereas routine *START* acts as driver for both input and polynomial initialization and routine *INTGRT* controls the subsequent program flow.

To improve the layout and distinguish between code and text, we have adopted upper case characters for the former. At the 'microscopic' level, blank spaces are used to improve readability of the Fortran statements, where blocks of code inside *DO*-loops and logical *IF*'s are indented.

3.2. COMMON variables

In principle, the program can be used for any particle number, subject to available computer mem-

ory and CPU time. The size of all particle arrays is controlled by the parameter *NMAX*, together with *LMAX* which is used for the neighbour lists. These parameters must be specified in the auxiliary header file `params.h` at compile time since most routines communicate via the global *COMMON* blocks, rather than by general-purpose arguments. All the *COMMON* variables are defined in a TEX file for convenience.[1]

The global labelled *COMMON* blocks (header file `common2.h`) are organized in four separate parts denoted *NBODY*, *NAMES*, *COUNTS*, *PARAMS*, which contain all the essential variables. This enables a calculation to be halted temporarily by saving the *COMMON* blocks on disc or tape, whereupon a restart can be made directly from the high-order scheme.

The convention of up to six characters for the names of routines or variables has been adopted, using mnemonic definitions. Standard Fortran 77 has been adhered to as far as possible. The main exceptions are:

- The *INCLUDE* statement is used for *COMMON* blocks.
- Input data are read in free format, which is less prone to typing errors.
- Neighbour lists are declared as *INTEGER**2 to save memory.
- Continuation lines have '*&*' in column 6 (non-standard character).

3.3. Input parameters

To be flexible and useful, a general-purpose program should allow a choice of input data. Since parameter space is large, the permissible range cannot be specified uniquely in advance and every new simulation becomes a test case. The complete list of input parameters is defined in Table 2, where each input record is separated by a space. The corresponding list of options (denoted *KZ*(*J*)) can be found in routine *DEFINE*.

Four main types of input may be distinguished. The integration parameters $ETAI = 0.03$ and $ETAR = 0.06$ usually give reasonable results, although slightly smaller values are preferable. Further discussions of time-step criteria and accuracy in the AC method have been given elsewhere (Makino, 1991a; Makino and Aarseth, 1992). The optimum size of the neighbour sphere depends on the problem; usually $NNBMAX \simeq 10 + N^{1/2}$ is sufficient for small N. For larger N (> 1000), Makino and Hut (1988) suggest $NNBMAX \simeq (N/8)^{3/4}$. However, see Spurzem (1999) for a different point of view.

An estimate of the initial neighbour radius (*RS*0) for routine *NBLIST* should be consistent with the scaled density distribution. For homogeneous systems (12) gives $R_s = (2n_1/N)^{1/3} R_h$, where n_1 is the desired membership (say *NNBMAX*/2). If necessary, the neighbour sphere is doubled or reduced by the volume factor until the membership falls in the acceptable range [1, *NNBMAX*].

Another set of parameters specify the initial conditions. The main decision-making is controlled by time intervals (*DELTAT*, *TCRIT*) and error checking (*QE*) for conservative systems. Among the optional features are rotation, external potentials and the creation of two initial subsystems. To guard against typing errors, routine *VERIFY* checks some input parameters.

4. NBODY2 algorithms

In the following subsections, we discuss some of the main procedures and algorithms used by the code *NBODY*2, including several optional features.

4.1. Initial conditions

N-body simulations usually employ quite specific initial conditions. However, for illustrative purposes we include a brief discussion of routine *DATA*.

The present version provides a choice (option 5) between an artificial system of constant density and a centrally concentrated Plummer model. Local input parameters are first read (exponent α_s, maximum and minimum mass m_1 and m_N). If $\alpha_s = 1$ or $m_1 = m_N$, all the individual masses are taken to be unity, otherwise they are selected from a smooth power-law

[1] The code can be downloaded from http://www.ast.cam.ac.uk/~sverre.

Table 2
Input parameters

Variable	Definition
KSTART	Control index (1: new run; >1: restart; >2: new parameters)
TCOMP	Maximum computing time in minutes (saved in *CPU*)
N	Total particle number ($\leq NMAX$)
NFIX	Output frequency of data save or binaries ($\neq$ 3 & 6)
NRAND	Random number sequence skip
NNBMAX	Maximum number of neighbours ($<LMAX$)
NRUN	Run identification index
ETAI	Time-step parameter for irregular force polynomial
ETAR	Time-step parameter for regular force polynomial
*RS*0	Initial neighbour sphere radius
DELTAT	Output time interval in units of the crossing time
TCRIT	Termination time in units of the crossing time
QE	Energy tolerance (restart if $DE/E>5*QE$ and $\neq 2>1$)
EPS	Softening parameter (square saved in *EPS*2)
KZ(*J*)	Non-zero options for alternative paths
*XTPAR*1	Mass of external Plummer model ($\neq 15=1$; scaled units)
*XTPAR*2	Length scale for Plummer model
ZMGAS	Mass scale for external logarithmic potential ($\neq 15=2$)
RGAS	Length scale for logarithmic potential
ALPHAS	Power-law index for initial mass function (routine *DATA*)
*BODY*1	Maximum particle mass before scaling
BODYN	Minimum particle mass before scaling
Q	Virial ratio (routine *SCALE*; $Q=0.5$ for equilibrium)
VXROT	*XY*-velocity scaling factor (>0 for solid-body rotation)
VZROT	*Z*-velocity scaling factor (not used if $VXROT=0$)
RBAR	Virial radius in pc (for scaling to physical units)
ZMBAR	Mean mass in solar units (for scaling)
XCM	Displacement of subsystem (routine *SUBSYS*; $\neq$ 17)
ECC	Eccentricity of relative motion for subsystem ($ECC\leq 1$)

distribution, where $\alpha_s = 2.35$ gives a Salpeter-type function.

In the simplest case ($KZ(5)=0$), a uniform spherical system is produced by choosing x, y, z at random inside a unit sphere with corresponding isotropic velocities having randomized magnitudes in [0, 1]. The second choice ($KZ(5)=1$) provides for a Plummer model with isotropic velocity distribution (cf. Aarseth et al., 1974 for a detailed algorithm).

4.2. Scaling

Results of *N*-body simulations are often presented in a variety of units which hinders comparison with other work. In the present code, a convenient set of units are introduced in routine *SCALE*; however, all scaling can be bypassed if desired (cf. option 16). Once the initial conditions have been generated, all coordinates and velocities are first expressed with respect to the centre of mass rest frame.

We introduce the so-called 'standard' units recommended by Heggie and Mathieu (1986), in which $G=1$, $M_0=1$, $E_0=-1/4$, where M_0 is the total mass and E_0 is the initial energy. These units are suitable for most types of bound systems. This choice corresponds to a virial equilibrium radius $R_v=1$, rms velocity $V_v=\sqrt{2}/2$ and a mean crossing time $T_{\rm cr}=2R_v/V_v=M_0^{5/2}/(2\,|E_0|)^{3/2}=2\sqrt{2}$. The scaling to internal units is carried out in four separate stages:

1. Scale all the masses by the old total mass, $M_0^{\rm old}$, to give $M_0 = 1$.
2. Calculate the total kinetic, potential and virial energies.
3. Scale the velocities by the specified virial theorem ratio (Q).
4. Rescale coordinates and velocities to yield the desired total energy.

Having transformed to scaled units, the output interval (*DELTAT*) and termination time (*TCRIT*) are re-defined in terms of the crossing time. Using the input parameters for the virial radius (*RBAR*) in pc and mean mass (*ZMBAR*) in solar masses, we also introduce scaling factors for conversion to physical units. This gives rise to the time unit *TSTAR* (in 10^6 yr), velocity unit *VSTAR* (km s^{-1}) and a re-scaled mass unit *ZMBAR* ($M_\odot$).

4.3. Integration algorithm

We begin by summarizing the starting procedure of the AC method. Given the initial conditions m_i, $\mathbf{r}_i$, $\boldsymbol{v}_i$ and a suitable value of R_s, the two force polynomials are first initialized according to (5) while the neighbour list is constructed. We also form the total force (11) and its first derivative, to be used for coordinate predictions and neighbour corrections. During the second stage, the next two orders (6) are calculated for each component, using the neighbour list for identification. The initialization procedure is completed by specifying irregular and regular time-steps, Δt_i and ΔT_i, according to (10), whereupon the divided differences are formed by (8). The main integration cycle consists of the following significant steps:

1. Select the next particle, i, to be advanced and define the current epoch by $t = t_i + \Delta t_i$.
2. Make a new sorted time-step list if $t > t_L$ and adjust Δt_L.
3. Compare $t + \Delta t_i$ with $T_0 + \Delta T_i$ to decide between a regular force prediction [case (1)] or a new total force summation [case (2)].
4. Predict coordinates of neighbours [case (1)] or all particles [case (2)].
5. Combine polynomials for i and predict $\mathbf{r}_t$ and $\boldsymbol{v}_t$ to order $\mathbf{F}^{(3)}$.
6. Evaluate the irregular force $\mathbf{F}_{\rm irr}^{\rm old}$ and update the times t_k.
7. Form new irregular differences and include the $\mathbf{D}^4$ term.
8. [Case (1) only.] Extrapolate the regular force and first derivative to give $\mathbf{F}_t$ and $\mathbf{F}_t^{(1)}$; go to step 15.
9. Obtain the new irregular and regular force, $\mathbf{F}_{\rm irr}^{\rm new}$ and $\mathbf{F}_{\rm reg}^{\rm new}$, and form a temporary neighbour list. Check for $n_1 = 0$ or $n_1 > n_{max}$.
10. Adjust the neighbour sphere R_s and update the times T_k.
11. Construct new regular differences and include the $\mathbf{D}^4$ term.
12. Set $\mathbf{F}_t$ from (11) and $\mathbf{F}_t^{(1)}$ using (4) for both types.
13. Identify the loss or gain of neighbours and accumulate derivative corrections; update the neighbour list and convert to differences by (8)
14. Prescribe the new regular time-step ΔT_i.
15. Update the new irregular time-step Δt_i.
16. Repeat the cycle at step 1 until termination or output.

Here both force polynomials are integrated to the same order; however, two separate time-step parameters are used in (10). Note that the decision to re-calculate the regular force is based on the next *estimated* irregular time-step. Force polynomials and their derivatives must be combined with care. Thus the second regular difference required at step 5 (and at output times) is obtained from a differentiation of (4) at $t \neq T_0$ which yields an extra term in $\mathbf{D}^3$. Likewise for step 8, the regular force and its first derivative are evaluated to highest order at an intermediate time before the contributions are added to the respective irregular parts.

Various stratagems may be used to increase the efficiency of determining the next particle to be advanced. The earlier scheduling algorithm of searching a list of $\simeq N^{1/2}$ pre-selected members has been modified to include sequential sorting. At time $t \geq t_L$ (with $t_L = 0$ initially), a list is formed of all the particles due to be advanced during the interval $[t_L, t_L + \Delta t_L]$ and the list is then ordered sequentially.

To include the case of repeated small time-steps and ensure that the global time increases monotoni-

cally, we have adopted an insert procedure to maintain the sequential ordering. Thus the index of a particle satisfying $t + \Delta t_i < t_L$, evaluated at the end of the cycle, is inserted at the appropriate sequential location (routine *INSERT*). An updating of the sorted list is made if $t > t_L$, whereupon the interval Δt_L is stabilized on a membership chosen as a compromise between the cost of sorting the quantities $t_j + \Delta t_j$ and inserting a small number of particles in the sequential list. This is achieved by employing two variable stabilization factors (in the range 0.25–4.0) which are used to determine the optimal membership ($\propto NNBMAX$). In spite of this complication, comparison with the so-called 'heap-sort' algorithm advocated by Press (1986) favoured the present algorithm above $N \simeq 100$, although the cost of both procedures is $\propto N \log_2 N$.

4.4. Force polynomials

At several stages, it is necessary to combine force polynomials and their derivatives. The prediction of the regular force is straightforward, using (2) with appropriate time arguments $t - T_k$ (cf. routine *NBINT*).

The first force derivatives are usually evaluated at the end of an integration step of either type, since coordinates or velocities at a general time $t \neq t_0$ or $t \neq T_0$ are obtained by an expansion with respect to the interval $t - t_0$. In the case of the irregular component, $t = t_0$ and (4) applies directly; likewise for the regular component at an end-point when $t = T_0$. However, the prediction of the regular force derivative at the end of a sub-step which does not coincide with a total force calculation requires special care. Applying one time differentiation of (2), we obtain

$$\mathbf{F}_t^{(1)} = \mathbf{D}^3(t_0' t_1' + t_0' t_2' + t_1' t_2') + \mathbf{D}^2(t_0' + t_1') + \mathbf{D}^1, \tag{16}$$

where again $t_k' = t_0 - t_k$ denotes the relevant time intervals (since $t_0 = t$ here). Setting $t_k = T_k$ then yields the predicted regular force derivative which is added to its irregular counterpart, obtained by (4), to form the total force derivative, $\mathbf{F}^{(1)}$, employed in the subsequent predictions.

The analogous expression for the second regular force derivative contains an extra term $t_0' = t_0 - T_0$ in the corresponding coefficient of $\mathbf{D}^3$ (cf. (4)). It is required for the prediction of particle i at the beginning of an irregular step as well as at output and other high-order predictions.

Force polynomials based on divided differences result in relatively simple code. However, the integration is most conveniently carried out in terms of an equivalent Taylor series, with (optimized) prediction to order $\mathbf{F}^{(3)}$ given by

$$\begin{aligned}
\mathbf{r}_t &= \left(\left(\left((0.6\tilde{\mathbf{F}}^{(3)} t' + \tilde{\mathbf{F}}^{(2)})\frac{1}{12} t' + \tilde{\mathbf{F}}^{(1)}\right) t' + \tilde{\mathbf{F}}\right) t' \right. \\
&\quad \left. + \mathbf{v}_0\right) t' + \mathbf{r}_0 \\
\mathbf{v}_t &= \left(\left((0.75\tilde{\mathbf{F}}^{(3)} t' + \tilde{\mathbf{F}}^{(2)})\frac{1}{9} t' + \tilde{\mathbf{F}}^{(1)}\right) 1.5t' \right. \\
&\quad \left. + \tilde{\mathbf{F}}\right) 2t' + \mathbf{v}_0.
\end{aligned} \tag{17}$$

Here the integration factorials in (9) and factorials in (4) are absorbed in $\tilde{\mathbf{F}}^{(k)}$, and $t' = t - t_0$ represents the integration interval.

The coordinate and velocity increments of (17) due to the corrector $\mathbf{D}^4$ contain four terms since all the lower derivatives are also modified in (4). Consequently, we combine the corresponding time-step factors for $\mathbf{r}_t$ and $\boldsymbol{v}_t$ to yield (in code notation)

$$\begin{aligned}
s_6 &= \left(\left(\left(\frac{2}{3} t' + s_5\right) 0.6t' + s_4\right)\frac{1}{12} t' + \frac{1}{6} s_3\right) t' \\
s_7 &= \left((0.2t' + 0.25s_5)t' + \frac{1}{3} s_4\right) t' + 0.5s_3,
\end{aligned} \tag{18}$$

where again all factorials are absorbed in the force derivatives. The coefficients are defined by $s_3 = t_1' t_2' t_3'$, $s_4 = t_1' t_2' + t_1' t_3' + t_2' t_3'$, $s_5 = t_1' + t_2' + t_3'$, respectively, where the old definition of t_k' still applies. The extra factor t'^2 omitted from (18) is included in $\mathbf{D}^4$. The semi-iteration is applied to both polynomials here. However, the regular corrections may not be beneficial if there are large force derivatives, as in the case of small softening parameters (i.e. $\epsilon < R_h/N$).

Although the AC scheme is based on the concept of nearest neighbours, there are situations when this is less useful. Thus the neighbour sphere determination for a distant particle may lead to undesirable oscillations and consequent time-step reduction. This

problem has been circumvented by including a nominal small central mass in the irregular force if there are no neighbours inside a large neighbour sphere ($R_s > 50\,R_h$); hence adopting $n_1 = 0$ still permits large irregular time-steps.

4.5. External potentials

We have included two optional external potentials, which are treated in a similar way. The first case ($KZ(15) = 1$) is the Plummer potential

$$\Phi_1 = -\frac{G\,M_b}{(r^2 + a^2)^{1/2}}\,, \tag{19}$$

where the background mass M_b and length scale a are specified as input (cf. *XTPAR*1, *XTPAR*2). The corresponding force is evaluated in routine *XTRNL*1 and added to the regular component. For initialization, the same routine adds the force and first derivative to the regular terms. The appropriate contributions are added to the potential and virial energies (routine *ENERGY*).

We also include an external logarithmic potential of the type

$$\Phi_2 = \frac{GM_g}{R_g \log_2(r/R_g)}\,, \tag{20}$$

where M_g is the mass scale and R_g is the length scale of the continuous mass distribution ($KZ(15) = 2$). A second routine (*XTRNL*2) is constructed in a similar way as above, with $C_g = M_g/R_g$ (denoted *CGAS*).

4.6. Inelastic collisions

An optional procedure (option 12) has been included for the coalescence of two particles. The neighbour scheme itself facilitates the determination of a collision candidate since a search can be made during the irregular force loop. Having identified a close neighbour, the osculating semi-major axis, a, may be used in the collision criterion. If accepted, the appropriate global index is saved until the end of the integration cycle. The main steps of the coalescence procedure (routine *COAL*) are as follows:

1. Improve the collision candidate (j) to order $\mathbf{F}^{(3)}$.
2. Obtain the potential energy with respect to both components.
3. Form the two-body binding energy $E_b = -m_i m_j/2a$.
4. Define new mass, coordinates and velocities using the index min(i, j).
5. Calculate potential energy with respect to the composite body.
6. Subtract the two-body energy and tidal potential energy from E_t.
7. Reduce N by 1 and update all *COMMON* arrays (routine *REMOVE*).
8. Predict coordinates and velocities of neighbours to order $\mathbf{F}^{(2)}$.
9. Initialize force polynomials and time-steps for the new body.

This algorithm maintains fairly good energy conservation with respect to the newly updated total energy (E_t). Since the *COMMON* arrays are addressed by global indices (rather than pointers), the removal of variables associated with a given particle (j) entails a compressing of arrays for all indices $k \geq j$, performed in routine *REMOVE*. In addition, the neighbour lists require reduction by 1 for all index references $> j$, as well as removal of the specified particle.

4.7. Escaper removal

Routine *ESCAPE* (option 13) contains procedures for the removal of escaping particles. A nominal tidal radius, R_t, is used as distance criterion, with $R_t = 10R_h$. The escape algorithm can be summarized as follows:

1. Select a candidate (i) for escape if $|\mathbf{r}_i| > 2R_t$.
2. Evaluate the binding energy $E_i = \frac{1}{2}v_i^2 + \phi_i$; accept escaper if $E_i > 0$.
3. Check whether nearest particle is escaping or has large perturbation.
4. Subtract $m_i E_i$ from the total energy E_t and m_i from M_t.
5. Correct the irregular force and first difference for each neighbour.
6. Reduce N by 1 and update all *COMMON* arrays (routine *REMOVE*).

7. Re-define coordinates and velocities in the c.m. frame and correct E_t.

In a second stage, routine *CMCORR* (option 18) adjusts all coordinates and velocities to the centre-of-mass rest frame. Appropriate modification of the total energy involves a correction to the new kinetic energy, and the primary integration variables, v_0 and $\mathbf{r}_0$ are updated consistently.

4.8. Density centre

Models of centrally concentrated systems are most conveniently analysed with respect to a well defined cluster centre, in accordance with standard observational practice. Following analytical estimates and extensive numerical experiments, Casertano and Hut (1985) have proposed an operational definition of the density centre and corresponding core radius.

The original definition of the core radius has been modified slightly in order to obtain a convergent result using a smaller (central) sample ($n \simeq N/2$). Thus the core radius is determined by the rms expression

$$R_c = \left(\frac{\sum_{i=1}^{n} |\mathbf{r}_i - \mathbf{r}_d|^2 \rho_i^2}{\sum \rho_i^2} \right)^{1/2} , \tag{21}$$

where $\mathbf{r}_d$ denotes the coordinates of the density centre and $\rho_i = 15/(4\pi r_6^3)$ is the density estimator defined with respect to the sixth nearest particle, r_6. We also generalize the density estimator to include the actual mass of the *five* nearest neighbours (Spurzem, 1991) in order to be consistent with the original expression for equal masses (cf. Eq. (V.3) of Casertano and Hut). The density centre and core radius are determined in routine *CORE* (option 8) according to the schematic algorithm:

1. Select all particles inside $\max(3R_c, R_h)$ as data sample.
2. Form a list of the six nearest neighbours for each selected particle.
3. Sort the list of six square distances to find the largest (r_6^2).
4. Obtain the total mass (M_5) of the *five* nearest neighbours.
5. Define individual mass densities $\rho_i = 3M_5/(4\pi r_6^3)$.
6. Form the density centre $\mathbf{r}_d = \Sigma\ \mathbf{r}_i \rho_i / \Sigma\ \rho_i$.
7. Obtain core radius R_c and average core density, $\langle \rho \rangle = \Sigma\ \rho_i^2 / \Sigma\ \rho_i$.

In the AC scheme, central particles tend to have the largest neighbour densities, n_1/R_s^3; hence an appropriate averaging would also yield a well defined density centre. Where relevant, any discussions (and algorithms) involving the central distance should be interpreted as referring to the density centre (i.e. using $|\mathbf{r}_i - \mathbf{r}_d|$), which is updated at output times.

4.9. Error checking

In standard simulations, energy conservation may be employed as a check to safeguard the results from any spurious procedures such as force discontinuities or inappropriate program usage. Even systems which include collisions, escape and/or analytical external potentials may be treated in this way, since the total energy is adjusted accordingly.

By saving all *COMMON* variables at specified intervals (usually each output), the last interval may be re-calculated with reduced time-steps if the energy change exceeds the prescribed tolerance. Since unit 1 (option 1) is reserved for *COMMON* save at termination, unit 2 is employed for this purpose. Thus if $KZ(2) = 1$, a *COMMON* save is performed every output, provided the relative energy error *DE* (scaled by the kinetic or potential energy) satisfies $DE < 5{*}QE$, where *QE* is the specified tolerance.

Modifications of the accuracy parameters take place if $KZ(2) = 2$. Thus if $DE > 5{*}QE$, the previous *COMMON* variables are read from unit 2, whereupon *ETAI* and *ETAR* are reduced by a factor of 2, and the current time-steps are reduced partially (subject to $\Delta t_i \geq t - t_i$). A second restart is carried out if this procedure does not satisfy the accuracy criterion (using the counter *NDUMP*). If a restart is not required but $DE > QE$, the integration parameters are decreased by $(QE/DE)^{1/2}$, whereas a small increase is allowed if $DE < 0.2{*}QE$ and $ETAI <$ *ETA*0 (initial value of *ETAI*).

5. Practical aspects

The final section is devoted to practical aspects of *N*-body simulations, and also summarizes some features of a test calculation.

5.1. Performance and Accuracy

The litmus test of a good code lies in its ability to produce satisfactory solutions in a given time. The CPU time requirement is particularly crucial for *N*-body simulations, which are often designed to make full use of the available resources, as in the case of dedicated workstations. In view of the time-consuming nature of such calculations, code design must inevitably be based on a trade-off between speed and accuracy.

In the present version, we have made a compromise by defining the basic variables $\mathbf{r}_0$, $\mathbf{r}_t$, $\mathbf{v}_0$, t_0 (and global time) in full double precision, with all other variables in standard precision. This gives rise to a faster force calculation while the dominant terms of the Taylor series are retained to higher accuracy at little extra cost. Assuming a Fortran 77 compiler (or better), the precision of the code may readily be changed by suitable declarations, taking care to match the array sizes of the saved *COMMON* blocks (routine *MYDUMP*). Moreover, on some types of hardware, the CPU time may not depend significantly on the choice of precision.

Actual CPU times are problem dependent and any empirical relation is therefore only a rough guide. In the absence of unique conversion factors for other machines, there is also further uncertainty in estimating CPU times. Timing tests have been carried out for Plummer models near equilibrium over one crossing time with N in the range [100, 2000] and softening parameter $\epsilon = 4R_v/N$. A least square fit including all points gave a relation for the computing time per crossing time as $T_c \propto N^{1.91}$ with a correlation coefficient exceeding 0.9997. Excluding $N = 100$ increased the exponent to 1.96. This result may be compared with an older test for N in [25, 250] which gave a value of 1.6 (Aarseth, 1985).

The question of accuracy must also be considered, although it is not an easy one. It is well known that the numerical solutions for the point-mass problem diverge on a relatively short time-scale (e.g. Miller, 1964; Goodman et al., 1993). However, in view of the chaotic nature of the *N*-body problem, it is not justified to aim for the highest possible accuracy at the expense of curtailing the integration time or particle number. This is particularly relevant when using an approximate model for the interaction potential.

The most noticeable systematic errors are connected with the integration of binaries, and this effect is also present when using a softening parameter. We can measure the characteristic error of the two-body motion by studying binaries in isolation, using the equivalent formulation with one force polynomial (program *NBODY*1). For a binary with eccentricity 0.8 (and no softening) the systematic error in semi-major axis per revolution is $\delta a/a = -3 \times 10^{-5}$ with $\eta = 0.02$ (or -6×10^{-6} with $\eta = 0.01$), which corresponds to about 140 (200) integration steps for each component.

5.2. Softening

The force law (1) is based on the softened potential $-G m_j/(r_{ij}^2 + \epsilon^2)^{1/2}$ which was introduced in order to model the interaction between galaxies (Aarseth, 1963). This potential represents the Plummer distribution, where the softening size is related to the half-mass radius by $R_h \simeq 1.3\,\epsilon$. We note that the maximum force occurs at $r = \epsilon/\sqrt{2}$, inside which the interaction tends to a harmonic oscillator.

The introduction of a softened potential is a convenient artifact for modelling larger systems by relatively small values of N, thereby reducing close encounter effects. Considerations based on uniform systems in virial equilibrium lead to a close encounter definition $r_{\rm cl} \simeq 4\,R_h/N$ for a 90 degree deflection between two typical particles. On the other hand, the need for sufficient resolution places an upper limit on the amount of softening for a given problem. A general discussion of these aspects has been given by several authors (e.g. White, 1976; Gerhard, 1981; Governato et al., 1991). Although the present code has been designed for softened potentials, it also works formally for the case $\epsilon = 0$ as long as there are no critical encounters or persistent binaries of high eccentricity.

In view of the slow asymptotic convergence of the

standard softened potential, it is desirable to consider expressions with a steeper r-dependence in the transition region. One such alternative is $\phi_4 = -Gm/(r^4 + \epsilon^4)^{1/4}$ which has been used for simulations of dwarf galaxies orbiting the Milky Way (Oh et al., 1995). Although the force evaluation is now more expensive, the resulting force is a better approximation to Newton's Law for $r > \epsilon$. In the AC scheme, the steeper fall-off actually permits the point-mass expression to be used for the regular force. Implementation is straightforward, following identification of all expressions involving ϵ^2.

The interpretation of the softening as an effective half-mass radius has been subject to some confusion. Although it has been traditional to generalize the interaction potential for two unequal-mass bodies by replacing ϵ^2 with $\epsilon_i^2 + \epsilon_j^2$ (e.g. White, 1976), the corresponding force does not describe correctly the motion of two overlapping Plummer spheres. The actual force law of such interactions is rather complicated (cf. Wielen, 1979), but alternatives have been discussed (Dyer and Ip, 1993). However, this still leaves softened potentials as a useful tool for reducing two-body relaxation. For example, collisionless simulations of collapsing systems may relate softening to half-mass radius (cf. Aarseth et al., 1988).

5.3. Special features

It is often desirable to perform large simulations in several stages by saving all *COMMON* variables on disc or tape. This can be done by specifying the run time at input (*TCOMP*), together with option 1. Alternatively, on Unix machines the command 'touch *STOP*' creates a file (*STOP*) which triggers termination at the next timing check (every *NMAX* steps).

A standard restart is made after reading the saved *COMMON* file from unit 1 (using *KSTART* = 2). For increased flexibility, some of the input parameters can be changed at restart time. Depending on the value of the control index, routine *MODIFY* reads one or two input lines (with the convention that only non-zero variables are altered):

- *DELTAT*, *TNEXT*, *TCRIT*, *QE*, *J*, *KZ*(*J*) (*KSTART* = 3 or 5).
- *ETAI*, *ETAR* (*KSTART* = 4 or 5).

Among the various types of initial conditions, we include an example of generating two separate subsystems (routine *SUBSYS*, option 17) which may be used for interacting binary experiments. Here the second system is a replica of the first (after it has been scaled in the usual way) and the two clumps are assigned initial displacements ($\pm XCM$) with specified orbital eccentricity (*ECC*), whereupon the particle number and total mass are updated. In this case, there is no well defined density centre, and the modification of the neighbour sphere by a radial velocity factor outside the core should be omitted (cf. option 10 in routine *REGINT*). Likewise, the determination of the density centre and the half-mass radius should be modified accordingly (i.e. by using two separate procedures), or suppressed altogether.

The general algorithm of the AC scheme requires current values of all the positions at each total force evaluation. However, the frequency of full N coordinate predictions may be reduced considerably without unduly affecting the accuracy (option 14). This is permissible because typical neighbours are predicted a number of times at each irregular force summation, thus reducing the need for frequent updating of all members. Consequently, the full N predictor loop is replaced by a neighbour prediction if the membership n_1 *exceeds* the average neighbour number (updated in *KZ*(14) every output, provided the initial value is non-zero). Another reduction of the numerical effort is made during initialization by omitting distant contributions to the second and third force derivatives in (6) (i.e. $R > 3\,R_s$).

5.4. Data analysis

The emphasis of the code is on providing a versatile integration tool, leaving the question of data analysis to the individual user. The main facility here is the data bank (option 3) which creates a file (unit 3) of the basic variables (m, $\mathbf{r}$, $\boldsymbol{v}$ and the identity) for each particle at output times *TNEXT* (frequency *NFIX*). This data may be read by a separate program, with the first record (*N*, *MODEL*, *NRUN*) containing the size (*N*) of the subsequent arrays.

The general output on unit 6 (routine *OUTPUT*) provides a summary of the main diagnostics. Most of this output refers to *COMMON* variables. Among the other quantities are the time expressed in crossing

times (*TC*) and in 10^6 yr (*T*6), half-mass radius ($\langle R\rangle$), maximum neighbour density contrast (*CMAX*), time-weighted neighbour number ($\langle Cn\rangle$), centre-of-mass displacements (*RCM*, *VCM*) and z angular momentum (*AZ*).

Further optional diagnostic output is provided on unit 6 by routine *BODIES*. Here an individual line is printed for each particle (using option 9 to control the amount). Likewise, a search for significant binaries is made at each output time or at a specified frequency, *NFIX* (*KZ*(6) = 1 or 2). Each output line contains the identity of components and their masses, binding energy per unit mass, semi-major axis, mean motion, separation, central distance, eccentricity and neighbour number.

Additional information about the radii of specified mass percentiles, including the half-mass radius, is provided by routine *LAGR* (option 7) and is printed on unit 6 and/or 7, depending on the value of the option. Further analysis or plotting routines may be called from routine *OUTPUT*, since all the current coordinates and velocities (**r**, $\dot{\mathbf{r}}$) are known at this stage.

5.5. Test calculation

It is instructive to illustrate the general workings of the code by quoting a test calculation, although strict reproducibility may not be achieved on different machines. We choose input parameters for a collapsing system of 250 particles (cf. test input file). Some selected variables are displayed in Table 3 at every other output time, using code notation.

Comparison of the irregular and regular integration steps (Columns 2 and 3) shows a ratio of about 6, for typical average neighbour numbers (Column 4) of 10. In spite of the violent relaxation, the total energy (Column 7) is quite stable, with fairly small relative energy errors (Column 8). The main error occurred at $t = 2T_{\rm cr}$ when the time-step parameters were reduced to 0.012 and 0.024 (from 0.02 and 0.04), respectively; this was followed by a small increase. Note that the half-mass radius (*RSCALE*) should be updated fairly frequently during phases of violent relaxation (cf. (12)).

The test calculation required a total CPU time of 35 s on a Sun UltraSparc 1 workstation, with clock speed 140 MHz and SPEC95 rating of 7.9. Although the AC method gains in efficiency with respect to a single polynomial formulation as N increases, it can also be used for relatively small systems. Thus a second similar example with $N = 25$ conserved the total energy to about 3×10^{-5} (using *NNBMAX* = 10 and $\epsilon = 0.25$) after 10389 and 5134 irregular and regular steps, respectively.

5.6. Program modifications

The variety of problems suitable for direct N-body simulations is so large that one code cannot deal with all the requirements. However, the basic solution method is sufficiently flexible for new procedures to be included.

Although the present code has been designed for scalar machines, it can readily be modified to exploit special features available on vector processors. The main speed-up occurs for long loops, where the optimum size is sometimes 'quantized' to powers of 64 (e.g. the CRAY supercomputer). Thus it may be advantageous to increase the size of the neighbour array; further discussion can be found elsewhere (Makino, 1986; Aarseth and Inagaki, 1986). The total force loop itself should be simplified by omitting the velocity-dependent part which becomes less effective with increasing neighbour number. Even for scalar machines, the implementation of a hierar-

Table 3
Summary of test calculation

TIME	NSTEPI	NSTEPR	⟨NB⟩	Q	⟨R⟩	E	DE
0.0	0	0	13	0.0	1.87	−0.25008	0.000000
2.8	44387	5010	13	0.38	1.27	−0.25008	0.000007
5.7	227263	37293	7	0.59	0.76	−0.25003	0.000054
8.5	353397	60890	9	0.58	0.66	−0.25002	0.000011
11.3	476792	83805	9	0.58	0.65	−0.25001	0.000005
14.1	601234	105861	10	0.62	0.61	−0.25000	0.000009

chical time-step scheme would be beneficial (McMillan, 1986; Makino, 1991b).

Since the square root calculation is by far the most time-consuming part of a direct N-body code, it may be desirable to consider alternative procedures. A fast algorithm based on a non-uniform look-up table for $1/r^3$ has proved effective on scalar machines, where indirect addressing is not a problem (cf. Aarseth et al., 1993). This algorithm would also be suitable for the regular force calculation without significant loss of precision.

The code may readily be modified to include a population of massless test particles. Now all N particles may be integrated in the standard way, with the force loops performed over a smaller number of massive bodies. When modelling close interactions, it is sometimes desirable to include partially inelastic effects. The collision determination would be similar to the coalescence procedure but the velocity of each component should be modified by the coefficient of restitution (Aarseth et al., 1993).

The effect of an external potential may also be studied as a perturbation of the internal motions. This description may be used for star clusters inside a galaxy or dwarf galaxies orbiting a large galaxy. In the case of open clusters, the assumption of circular motion near the plane of symmetry permits simplified expressions for the tidal force (cf. Aarseth, 1985). For more general types of orbits, it is convenient to employ local rotating coordinates with respect to a guiding centre (Oh et al., 1992).

Cosmological modelling by direct N-body methods is limited to much smaller particle numbers than can be studied by FFT or tree codes. The standard AC code can be used for cosmological simulations as it stands (i.e. $Q > 1$). However, it is more efficient to employ comoving equations of motion. Such a formulation has been described elsewhere (Aarseth, 1985) and is available as a separate program called *COMOVE*.

Finally, we mention the generation of output for computer movies. This requires monitoring an additional time-scale after each integration cycle. All relevant particle coordinates can then be written to a separate file, using scaled integer format of two bytes to save memory. The actual movie production depends on available software; by now it is possible to make nice movies on a PC using standard graphics packages.

References

Aarseth, S.J., 1963. MNRAS 126, 223.

Aarseth, S.J., 1985. In: Brackbill, J.U., Cohen, B.I. (Eds.), Multiple Time Scales. Academic Press, New York, p. 377.

Aarseth, S.J., Inagaki, S., 1986. In: Hut, P., McMillan, S. (Eds.), The Use of Supercomputers in Stellar Dynamics. Springer-Verlag, New York, p. 203.

Aarseth, S.J., Hénon, M., Wielen, R., 1974. A&A 37, 183.

Aarseth, S.J., Lin, D.N.C., Papaloizou, J., 1988. ApJ 324, 288.

Aarseth, S.J., Palmer, P.L., Lin, D.N.C., 1993. ApJ 403, 351.

Ahmad, A., Cohen, L., 1973. J. Comput. Phys. 12, 389.

Casertano, S., Hut, P., 1985. ApJ 298, 80.

Dyer, C.C., Ip, P.S.S., 1993. ApJ 409, 60.

Gerhard, O.E., 1981. MNRAS 197, 179.

Goodman, J., Heggie, D.C., Hut, P., 1993. ApJ 415, 715.

Governato, F., Bhatia, R., Chinkarini, G., 1991. ApJ 371, L15.

Heggie, D.C., Mathieu, R.D., 1986. In: Hut, P., McMillan, S. (Eds.), The Use of Supercomputers in Stellar Dynamics. Springer-Verlag, New York, p. 233.

Makino, J., 1986. In: Hut, P., McMillan, S. (Eds.), The Use of Supercomputers in Stellar Dynamics. Springer-Verlag, New York, p. 151.

Makino, J., 1991a. ApJ 369, 200.

Makino, J., 1991b. PASJ 43, 859.

Makino, J., Aarseth, S.J., 1992. PASJ 44, 141.

Makino, J., Hut, P., 1988. ApJS 68, 833.

McMillan, S.L.W., 1986. In: Hut, P., McMillan, S. (Eds.), The Use of Supercomputers in Stellar Dynamics. Springer-Verlag, New York, p. 156.

Miller, R.H., 1964. ApJ 140, 250.

Oh, K.S., Lin, D.N.C., Aarseth, S.J., 1992. ApJ 386, 506.

Oh, K.S., Lin, D.N.C., Aarseth, S.J., 1995. ApJ 442, 142.

Press, W.H., 1986. In: Hut, P., McMillan, S. (Eds.), The Use of Supercomputers in Stellar Dynamics. Springer-Verlag, New York, p. 184.

Spurzem, R., 1991. Personal communication.

Spurzem, R., 1999. J. Comp. Appl. Maths. 109, 407.

White, S.D.M., 1976. MNRAS 177, 717.

Wielen, R., 1979. MitAG 45, 16.