HERMIT4 User Manual

Sverre Aarseth

Email: sverre@ast.cam.ac.uk

Institute of Astronomy, University of Cambridge

1 Introduction

This is a User Manual for HERMIT4, a code that was developed during February 2004. It is a combination of two proven formulations which are studied by the Hermite method. Most particles are integrated by the time-symmetric scheme of Kokubo, Yoshinaga & Makino [1998]. In order to treat close two-body encounters more accurately, the so-called Burdet–Heggie regularization method is employed. For simplicity only one such encounter is treated. This means that the relative motion must be hyperbolic and any rare binaries with small periods are combined into one body as an inelastic collision. The time-symmetric scheme is based on the presence of one dominant central body which allows iteration of the two-body term, thereby reducing the systematic error considerably. Constant (quantized) time-steps for small perturbations also enhance the numerical stability. However, it is still a brute-force method. Hence the CPU requirements limit the maximum particle number to a few hundred, depending on the desired time interval.

2 Code structure

The whole code consists of some 2265 lines including comments and layout space. It is written in FORTRAN and is F77 compliant but also compiles with Intel or F95 (ignore warning messages about non-standard expressions). There are about 32 routines altogether, with mnemonic names of maximum six characters (see Table 1). Likewise, all the FORTRAN statements are in upper case while the comments are in lower case. The coding conforms to a strict style and layout for clarity.

The main code relies on a general common block called commonp.h which contains a number of arrays of maximum size NMAX defined as a parameter, as well as many useful scalars. This enables a calculation to be split into several stages by saving all the common variables after a specified CPU time (or by a 'touch STOP' facility at arbitrary times), followed by a restart. Except for some special situations, this gives rise to reproducible results which are essential for experimental purposes as well as problem investigation.

The code consists of three main parts: input, output and integration, with the latter split into several routines dealing with two different methods. Since the choice is between integration of single particles or one close encounter, the decision-making is simplified and requires very little overheads. Several optional procedures are included for convenience (see routine INPUT for definitions).

3 Getting started

Once the file `hermit4.tar.Z` is downloaded and uncompressed, the routines are extracted by '`tar xvf hermit4.tar`' and copied to directory `Hermit4`. A test input template called `input` is also included.

Before compiling, check the `FORTRAN` directives in the `Makefile`, i.e. whether `f77` or `g77` and use the highest optimization level. Compile by the command '`make hermit4`' which should produce the executable `hermit4`. One possible reason for failure could be the `CPU` timer function `etime` in routine `CPUTIM` which is system dependent. Any other complaints and strange run-time behaviour should be reported after making every effort to ascertain the problem. For frequent usage it is recommended to create separate working directories. It is also a good idea to save the original files when making changes.

To start a test run, place the executable and a template input file `input` in one directory and type the command '`hermit4 < input > output &`'. Although the results are machine- and compiler-dependent, it is expected that the run will finish normally.

The restart facility can be tested in the following way. First specify option #1 = 1 at input time. This will ensure a `common` save on unit `fort.1` if the specified `CPU` time is exceeded or `TIME >= TCRIT`, with `TCRIT` given in the input file. The calculation can then be continued from the `common` save by '`hermit4 < rs >` output2 &', using the restart input file `rs` containing '`KSTART TCOMP`'. Here `KSTART = 1` denotes a new run (followed by the required input) or = 2 for standard restart, and `TCOMP` is the `CPU` time in minutes. If the calculation is terminated at `TIME = TCRIT` with #2 = 2, `fort.2` must be copied to `fort.1` before the restart, in which case a new value of `TCRIT` needs to be prescribed. Also note the possibility of reading some new parameters at restart if `KSTART > 2`. In the current version only one option can be changed but this can readily be generalized.

4 Input parameters

A standard input file consists of four lines defining the `CPU` time, membership, dimensionless time-step factor `ETA`, options and two-body regularization parameters. The overall accuracy is controlled by `ETA`, which enters the time-step criterion in a square root. The appropriate choice depends on many factors and is best decided after extensive testing but for practical work a value around 0.003 seems adequate. Likewise, the options can be selected by consulting routine `INPUT`. Hence only a few values need to be changed in the input template once the first calculation has been performed.

The question of the initial particle distribution must be decided by the user and only a few light bodies have been included for code testing purposes. It is recommended to generate initial masses, coordinates and velocities by a separate program, to be read in by routine `DATA`. Note that the calculations are performed in heliocentric coordinates, with unit mass assumed for the massive central body. The code is not suitable for studying high eccentricities because the indirect terms may be large, giving rise to unfavourable fluctuations. For central distances well outside the unit length, ensure that the maximum time-step `DTMAX` (set in `DATA`) is sufficiently large.

Input parameters for regularization should be consistent with the particle distribution. Thus the proper value of the close encounter time-step, `DTMIN`, should match the critical two-body separation, `RMIN`. Because the solar perturbation also plays a role (and its effect is included), it is recommended to use a conservative value of `RMIN` much smaller than the Hill (or Roche) radius. Since most encounters treated will be hyperbolic, the choice of the regularized time-step parameter `ETAU` is less important. For a binary, there are $2\pi/$`ETAU` steps per orbit, while a constant `DTAU` determined at `R = RMIN` is appropriate for hyperbolic motion. In practice, the regularization step counter should be much less than the number of standard integration steps.

Since planetary systems are intended, the output time interval and termination time, `DELTAT` and `TCRIT`, are specified in years as input. Scaling the time to other systems, such as a planet with satellites, can readily be made in terms of the square root of the solar mass ratio.

The code allows for integration of an additional component of mass-less particles. This is achieved by first allocating `NMASS` massive planetesimals in the sequential arrays, followed by `N - NMASS` particles of zero mass. Hence all particles are advanced in the same way, with the force summation restricted to `NMASS`. At present, `NMASS = N` and this feature has not been fully tested.

5 Implementation

The integration of single particles employs the time-symmetric Hermite block-step scheme [Kokubo *et al.* 1998]. At each cycle, a list of particles due to be advanced is determined. The perturbations due to other planetesimals is evaluated separately from the dominant two-body term. This allows an extra iteration after the corrector has been applied which results in a significant accuracy gain (as can be seen by suppressing the iteration loop).

Close encounters are characterized by small time-steps. A search is therefore made for the dominant neighbour if the time-step is below `DTMIN`. The candidate particle is accepted if `R` $<$ `RMIN` and the relative velocity is negative. Masses, coordinates and velocities for the two particles are then placed in the two first locations of the respective arrays. Relevant quantities for the new pair are initialized according to the equations of motion [cf. Aarseth 2003] and the centre of mass (c.m.) is introduced as a fictitious particle at location `NTOT = N + 1`. Finally, the force polynomial and time-step for the c.m. are evaluated in the single particle approximation with reduced time-step.

The subsequent integration of the c.m. particle is formally similar to that of single particles, except that the force is obtained by vectorial summation over the components unless the c.m. approximation applies. Note that since heliocentric coordinates are used, the so-called indirect force terms must be included in all the equations of motion. Conversely, the force on single particles due to a regularized pair is treated more carefully within a factor ($\simeq 100$) of the two-body separation. This entails inverting the regularized time interval by a Taylor series expansion and transforming to physical coordinates and velocities before summing over the individual components. Hence the direct force loop starts at `IFIRST = 2*NPAIRS + 1`, where `NPAIRS = 1` during regularization, in which case the corresponding contribution is added afterwards.

The equations of motion for two-body regularization are initialized by boot-strapping, yielding explicit derivatives and also advanced by the Hermite integration scheme. Each regularized time-step is converted to physical units by a high-order Taylor series based on known coefficients. Hyperbolic regularizations are terminated when the initial separation is exceeded. In order to improve the accuracy and permit larger new quantized time-steps, the relative motion is advanced up to the end of the next block-step. This is followed by standard initialization of polynomials and time-steps for both components.

Collisions are defined by `R < RCOLL`, where `RCOLL` is currently defined by a data statement in routine `BHINT` (but can readily be included as input, provided it is added in `commonp.h`). For practical reasons, the implementation of a collision (option #7) is again delayed until the end of the block-step, leaving the c.m. as a new particle after removal of the components.

Particles are removed (option #9) from the calculation at output time for two different conditions. This occurs if the eccentricity exceeds a critical value (defined by `ECRIT` in routine `OUTPUT`) and the central distance is suitably large (i.e. `RESC`) or the orbital eccentricity is less than 1. The latter case deals with sun-grazing orbits which would otherwise need many small time-steps. Other criteria may be introduced to suit the requirements.

Eccentric binaries are treated as close encounters in the pericentre region. However, in order to restrict the treatment to just one regularization for simplicity, the components of binaries with period below `TMERGE` (in days, defined in routine `BHINT`) are combined inelastically as for physical collisions. Note that the critical period must at least be large enough to include a circular binary of semi-major axis `RMIN`, otherwise an inconsistency may arise. In fact, this requirement needs to be satisfied by a considerable margin in order to handle hyperbolic encounters at the appropriate times.

Note that the calculation can be terminated at any time by typing 'touch STOP' whereupon a `common` save will occur on `fort.1`, from which a restart can be made.

References

Aarseth, S.J. [2003], *Gravitational N-Body Simulations* (Cambridge University Press).

Burdet, C.A. [1967], 'Regularization of the two-body problem', *Z. Angew. Math. Phys.* **18**, 434–438.

Heggie, D.C. [1973], 'Regularization using a time-transformation only', in *Recent Advances in Dynamical Astronomy*, ed. B.D. Tapley & V. Szebehely (Reidel, Dordrecht), 34–37.

Kokubo, E., Yoshinaga, K. & Makino, J. [1998], 'On a time-symmetric Hermite integrator for planetary N-body simulation', *Mon. Not. R. Astron. Soc.* **297**, 1067–1072.

6 Appendix

Table 1 contains a list of all the subroutines, the main calling structure and a brief description of their purpose. The subroutine names are constructed in analogy with the large N-body codes.

Table 1: *Subroutine definitions.*

Routine	Called by	Description
BHINIT	BHREG	Initialization of two-body regularization.
BHINT	INTGRT	Integration of two-body motion.
BHLIST	BHINIT	Perturber selection.
BHPERT	BHINIT & BHINT	Perturbing forces.
BHREG	HERMIT4	New Burdet-Heggie regularization.
BHTERM	HERMIT4	Termination of regularization.
BLOCK		Block data initialization.
BODIES	OUTPUT	Output of single particles and binaries.
CMF	NBINT	Force on c.m. particle.
CPUTIM	INTGRT & OUTPUT	Elapsed CPU time in minutes.
DATA	START	Generation of initial conditions.
ENERGY	OUTPUT & BHTERM	Evaluation of the total energy.
FPERT	SEARCH	Perturbing force on dominant bodies.
FPOLY1	START & BHINIT	Initialization of force polynomial.
HERMIT4		Master routine.
IBLOCK	START	Initialization of block steps.
INPUT	START	Parameter input.
INTGRT	HERMIT4	Main integrator flow control.
MYDUMP	INTGRT & OUTPUT	COMMON save or read.
NBINT	INTGRT	N-body integration with iteration.
OUTPUT	HERMIT4	Energy check, output and escape removal.
RAN2	DATA	Random number generator.
REMOVE	BHINT & OUTPUT	Particle removal.
RESOLV	BHINT & NBINT	Two-body coordinates and velocities.
SEARCH	NBINT	Close encounter search.
START	HERMIT4	Initialization of data and polynomials.
STEPI	NBINT	Fast time-step expression (option #5).
STEPK	STEPS	Truncation to commensurate block-step.
STEPS	START & BHINIT	Initialization of time-steps.
TSTEP	NBINT	Standard time-step criterion.
XVPRED	OUTPUT & BHINT	Prediction of coordinates and velocities.
ZERO	START	Initialization of global scalars.