Star clusters and black holes in galaxies across cosmic time
Proceedings IAU Symposium No. 312, 2014
Y. Meiron, S. Li, F.-K. Liu & R. Spurzem, eds.

doi:10.1017/S1743921315007851

Expansion techniques for collisionless stellar dynamical simulations

Yohai Meiron

Kavli Institute for Astronomy and Astrophysics at Peking University, Beijing 100871, China
email: `ymeiron@pku.edu.cn`

Abstract. We present *ETICS*, a collisionless N-body code based on two kinds of series expansions of the Poisson equation, implemented for graphics processing units (GPUs). The code is publicly available and can be used as a standalone program or as a library (an AMUSE plugin is included). One of the two expansion methods available is the self-consistent field (SCF) method, which is a Fourier-like expansion of the density field in some basis set; the other is the multipole expansion (MEX) method, which is a Taylor-like expansion of the Green's function. MEX, which has been advocated in the past, has not gained as much popularity as SCF. Both are particle-field methods and optimized for collisionless galactic dynamics, but while SCF is a "pure" expansion, MEX is an expansion in just the angular part; thus, MEX is capable of capturing radial structure easily, while SCF needs a large number of radial terms.

Keywords. methods: numerical – stars: kinematics and dynamics

1. Introduction

A stellar system could naively be described as a set of $3N_\star$ coupled, second-order, nonlinear ordinary differential equations, where $N_\star$ is the number of stars. Solving such an equation set numerically is practically only possible at the very low end of the realistic $N_\star$-range, and even so could be very challenging with current computer hardware. Thus, various techniques are used to simplify the mathematical description of the system; these are often designed to fit a particular problem in stellar dynamics and yield unphysical results when applied to another problem.

Direct N-body simulation is one of the main techniques used to study gravitational systems in general and galaxies in particular. In this technique, the distribution function is sampled at $N \ll N_\star$ points in a Monte-Carlo fashion. This N depends on the computational capabilities an may be orders of magnitudes smaller. This simplification can cause problems, as some dynamical processes depend on the number density rather than just the mass density. The most well-known of these processes is two-body relaxation. The relaxation time (the characteristic time for a particle's velocity to change by an order of itself due to encounters with other particles) scales with the crossing time roughly as $N/\ln N$. Thus, the ratio between the relaxation times in a real and a simulated system is of a similar order of magnitude as the undersampling factor.

Galaxies are often described as collisionless stellar systems, which means that the relaxation time is much longer than the timescale of interest (except perhaps at the very center). This property could be very useful. Since a particle's orbit is basically what it would be if it were moving in a smooth gravitational field, we could evaluate the field instead of calculating all of the stellar interactions, which is computationally cheaper. Another useful property is that galaxies are often spheroidal in shape. Even highly flattened galaxies will have a spherical dark halo component. Thus, a spherical shape could be used as a zeroth-order approximation for the gravitational field, and higher-order terms could be written using spherical harmonics.

These two facts are utilized by series expansion techniques such as the multipole expansion (MEX) and the self-consistent field (SCF) methods. They historically come from different ideas and are mathematically distinct. In the context of numerical simulations, however, they serve a similar function: to evaluate the gravitational force on all N particles generated by this same collection of particles in a way that discards spurious small-scale structure (in other words, smooths the field). In Meiron *et al.* (2014) we first presented the *ETICS*† code which encapsulates these techniques, fully describing the mathematics behind it, implementation and accuracy. In this proceeding we briefly summarize this and show basic performance benchmarks.

2. Formalism

Both the MEX and SCF methods are ways of solving the Poisson equation

$$\nabla^2 \Phi(\boldsymbol{r}) = 4\pi\rho(\boldsymbol{r}), \tag{2.1}$$

the formal solution of which is given by the following integral:

$$\Phi(\boldsymbol{r}) = -\int \frac{\rho(\boldsymbol{r}')\mathrm{d}^3r'}{|\boldsymbol{r}-\boldsymbol{r}'|}. \tag{2.2}$$

The expression $|\boldsymbol{r}-\boldsymbol{r}'|^{-1}$ is the Green's function of the Laplace operator in three dimensions and in free space (no boundary conditions), and the integral is over the whole domain of definition of $\rho(\boldsymbol{r})$.

In both MEX and SCF, the integrand above is expanded as a series of terms, each of which is more easily numerically integrable; this is done in two different ways, lending the two methods quite different properties. In brief, MEX is a Taylor-like expansion of the Green's function, while SCF is a Fourier-like expansion of the density. Another way to look at it is that in both methods the integrand is written as a series of functions (of $\boldsymbol{r}$) with coefficients: in MEX, one uses the given density to evaluate the functions, while their coefficients are known in advance; in SCF, one evaluates coefficients, while the functions are known in advance. The standard form of MEX in three dimensions is

$$\Phi(\boldsymbol{r}) = -\sum_{l=0}^{\infty}\frac{4\pi}{2l+1}\sum_{m=-l}^{l}\left[q_{lm}(r)r^{-(l+1)} + p_{lm}(r)r^{l}\right]Y_{lm}(\theta,\phi) \tag{2.3}$$

$$q_{lm}(r) = \int_{r'<r} r'^{l}\rho(\boldsymbol{r}')Y^{*}_{lm}(\theta',\phi')\mathrm{d}^3r' \tag{2.4}$$

$$p_{lm}(r) = \int_{r<r'} r'^{-(l+1)}\rho(\boldsymbol{r}')Y^{*}_{lm}(\theta',\phi')\mathrm{d}^3r'. \tag{2.5}$$

Since in practice the density field is made of N discrete points, they must be sorted by r in order for the above integrals to be evaluated in one pass.

The standard form of SCF in three dimensions is

$$\Phi(\boldsymbol{r}) = \sum_{n=0}^{\infty}\sum_{l=0}^{\infty}\sum_{m=-l}^{l} A_{nlm}\Phi_{nl}(r)Y_{lm}(\theta,\phi) \tag{2.6}$$

$$A_{nlm} = \int \rho(\boldsymbol{r}')\Phi_{nl}(r')Y^{*}_{lm}(\theta',\phi')\mathrm{d}^3r'. \tag{2.7}$$

† `https://github.com/sahmes/etics`

$\Phi_{nl}(r)$ are functions forming an orthogonal set, the radial basis, which is a key difference between MEX and SCF. The choice of basis is not unique, and the basis functions themselves need not represent physical potentials, but it is convenient to take the zeroth term ($n = l = 0$) to represent some physical system, and to construct the rest of the set by some orthogonalization method. Hernquist & Ostriker (1992) constructed a radial basis using Gegenbauer polynomials that at zeroth order was a Hernquist (1990) model, which they argued was well suited to study galaxies; this is the basis we adopt in *ETICS*.

3. Implementation

GPUs are powerful and cost-effective devices for high-performance parallel computing. They are used to accelerate many scientific calculations, especially in astrophysics, such as the dynamics of dense star clusters and galaxy centers (see review by Spurzem *et al.* 2012). The main issue when implementing a mathematical technique for GPUs is to make the algorithm parallel. In many cases there is already a parallel implementation of the method, but a GPU is different from a multi-CPU parallel environment, and the parallelization scheme is not always transferable. A full description of the implementation would be very long and technical, thus we only briefly overview it here; the full details are found in Meiron *et al.* (2014).

Both the MEX and SCF algorithms have two distinct steps: calculating the expansion from the particles, and using the expansion to calculate the forces on the particles. These two parts are done consecutively in *ETICS*, while each part is done in parallel on the GPU. While in SCF the full coefficient list is calculated before it is used for force calculation, in MEX, due to use of very large arrays in the global memory, the force is calculated in parts, i.e., first the zeroth (monopole) order force is calculated, then the $l = 2$ correction is added to it, and so forth.

The current implementation of the MEX method relies on *Thrust* (Bell & Hoberock 2011), a C++ template library of parallel algorithms which is part of the CUDA framework. The particles are first sorted by radius using a *Thrust* subroutine; this is followed by a loop over l, inside of which the spherical harmonics are calculated; these are used to calculate the contributions of the particle to q_{lm} and p_{lm}, which are saved in global memory. Finally, *Thrust* subroutines are dispatched to perform forward and backward cumulative sums.

For SCF, no sorting or cumulative summation are done. In principle, a similar algorithm could be used, i.e., the contribution of each particle to each coefficient is stored in global memory, and later the contributions are summed. The problem is that there are $\frac{1}{2}(n_{\max} + 1)$ more SCF coefficients than there are MEX functions. There is a better scheme that does not require such heavy use of the global GPU memory. Since a normal (rather than cumulative) summation is needed, it is fairly easy to use the very fast "shared memory" (another memory type in the GPU, available to all threads in a single block) to accumulate the particles' contributions, and then perform parallel reductions within the block (later the CPU finalizes by summing the contribution from each block). The parallel SCF algorithm of Hernquist *et al.* (1995) could not be used here due to the difference between how the GPU and CPU access and cache memory (it could however be used for the inter-node level parallelization).

The calculation of the forces (or potentials) at the last step is very similar between MEX and SCF (except, as mentioned before, the MEX force is calculated for each multipole level separately), as each particle only needs to know its own coordinates and the series expansion and there is no dependency between the threads.

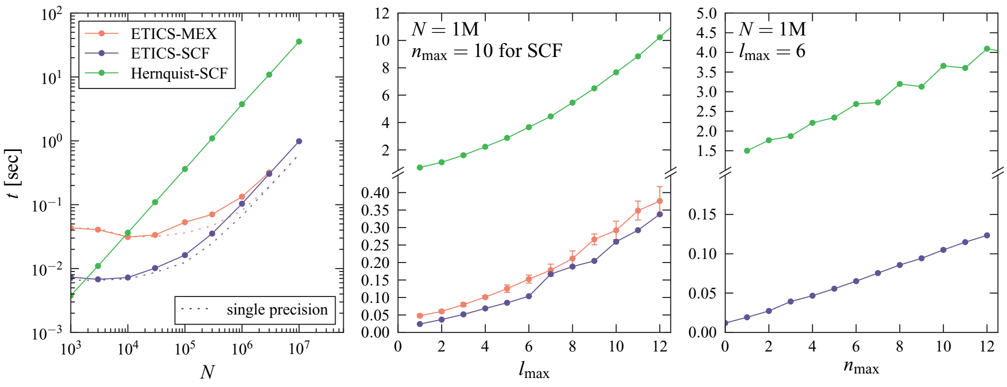

Figure 1. Scaling of one full force calculation time. Hernquist's SCF code (in green) is a CPU and *ETICS* is a GPU code with both MEX (red) and SCF (blue) methods; for the GPU codes, dotted lines show the performance in single-precision mode. The parameter which is held constant is indicated on each panel. The CPU code shows some erratic behavior due to compiler optimization. Note that the CPU and GPU tests use different hardware.

4. Performance

We tested the performance of *ETICS* (both MEX and SCF) on a single Nvidia Tesla K20 GPU on the Laohu supercomputer at the NAOC in Beijing. For comparison, we also tested the Fortran CPU SCF code by Lars Hernquist on the ACCRE cluster at Vanderbilt University in Nashville, Tennessee (we used a node with an Intel Xeon E5520 CPU). Figure 1 shows the time it takes to do one full force calculation as a function of N, $l_{\max}$, and $n_{\max}$. Note that the timing only depends on the number of particles (and expansion cutoffs) and not on their spatial distribution.

The CPU and GPU SCF codes are both theoretically $\mathcal{O}(l_{max}^2 n_{max} N)$. At low N, the GPU is not fully loaded and *ETICS* performance seems superlinear with N. *ETICS*-MEX is theoretically $\mathcal{O}(l_{max}^2 N\ logN)$, but this again is an asymptotic behavior which is not observed. The lack of good GPU load for $N \lesssim 10^6$ is much more evident than the $N logN$ nature of the algorithm. The GPU global memory was the limiting factor in how many particles could be used with both methods. All codes should scale quadratically with $l_{\max}$, but as the middle panel of Figure 1 shows, this behavior is not as clear for *ETICS*-MEX. This is due to the extensive memory access this code requires, which rivals the calculation time. Memory latency on GPUs is not easy to predict; due to caching and the way memory is copied in blocks, and the latency depends not only on the amount of memory accessed but also on the memory access pattern.

References

Bell, N. & Hoberock, J. 2011, in *GPU Computing Gems Jade Edition*, ed. W.-M W. Hwu (Waltham, MA: Morgan Kaufmann), 359
Hernquist, L. 1990, *ApJ*, 356, 359
Hernquist, L. & Ostriker, J. P. 1992, *ApJ*, 386, 375
Hernquist, L., Sigurðsson, S., & Bryan, G. L. 1995, *ApJ*, 446, 717
Meiron, Y., Li, B., Holley-Bockelmann, K., & Spurzem, R. 2014, *ApJ*, 792, 98
Spurzem, R., Berczik, P., Berentzen, I., *et al.* 2012, in *Large-Scale Computing Techniques for Complex System Simulations*, ed. W. Dubitzky, K. Kurowski, & B. Schott (Hoboken, NJ: John Wiley & Sons), 35