応用力学論文集Vol.2,PP.101-109（1999年8月）　　土木学会

研究展望：高速多重極展開法 — 粒子法への応用を中心として

REVIEW: Fast Multipole Method — Applications to Particle Systems

牧野淳一郎*・川井敦**
Junichiro MAKINO and Atsushi KAWAI

*非会員 PhD 東京大学助教授 大学院理学系研究科天文学専攻 (〒113-0033 東京都文京区本郷 7-3-1)
**非会員 学術修 東京大学大学院 総合文化研究科 (〒153-8902 東京都目黒区駒場)

We overview the Fast Multipole Method (FMM) and the Barnes-Hut tree method. These algorithms evaluate mutual gravitational interaction between N particles in $O(N)$ or $O(N \log N)$ times, respectively. We present basic algorithms as well as recent developments, such as Anderson's method of using Poisson's formula, the use of FFT, and other optimization techniques. We also summarize the current states of two algorithms. Though FMM with $O(N)$ scaling is theoretically preferred over $O(N \log N)$ tree method, comparisons of existing implementations proved otherwize.

Key Words : *Fast Multipole Method, Barnes-Hut tree, N-body problem*

1. はじめに

ここでは、多数の質点間の重力相互作用や荷電粒子の間の静電相互作用を高速に計算する手法として1980年代中頃に提案された FMM (Fast Multipole Method, 高速多重極展開法) [1]と、それと非常に関連が深く、やはり1980年代に提案されて広く使われるようになったツリー法[2),3)]について、その原理と研究の動向を概説する。以下、問題を質点系の重力相互作用ということにする。別にクーロン相互作用やそれ以外の（$1/r$ ポテンシャルでないような）相互作用でも、距離が大きくなれば大きさが小さくなるようなものであれば原理が変わるわけではない。また、粒子系以外でも、渦糸法による流体計算、偏微分方程式の境界値問題など、積分形で書けるものには広く応用できる。そういった研究もなされている。（例えば[4),5),6)]）ここでは、そのような粒子系以外への適用は他の文献に譲り、粒子系に絞って原理と最近の発展をみていくことにしたい。

まず、基本原理が単純であるツリー法についてその原理を説明し、次に FMM の原理を説明し、その次に、実際に使う上での問題点、最近の研究の動向等についてまとめる。

2. 重力多体問題

解くべき問題は、以下のような常微分方程式系の数値解を求めるということである。

$$\frac{d^2\boldsymbol{x}_i}{dt^2} = G \sum_{j=1,j\neq i}^{N} m_j \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{|\boldsymbol{x}_j - \boldsymbol{x}_i|^3},$$

ここで N は粒子の数、$\boldsymbol{x}_i, m_i$ はそれぞれ粒子 i の位置、質量であり、G は重力定数である。

右辺を評価するもっとも単純な方法は、右辺をそのまま計算することである。粒子数が数百程度であれば、この方法（単に直接計算という）よりも速くするのは難しい。

しかし、直接計算には $O(N^2)$ で計算量が増えていくという問題がある。計算量を減らすには、さまざまな方法がある。従来から知られている方法の多くは、粒子の集まりを質量の連続分布に置き換え、重力ポテンシャルのほうも連続関数として扱うものである。このようにすることで、ポテンシャルは線形なポアソン方程式の解として与えられる。これらは、高速フーリエ変換 (FFT) や球関数展開などを使って解くことができる。

FFT が使える場合には、格子点の数を M として計算量は $O(M \log M)$ となる。従って、粒子数 N に比べて M が同程度あるいは少なくて済むならば、計算量は $O(N \log N)$ ないし $O(N)$ となり、素晴らしく高速である。これに対して、球関数展開の場合には（高速球面調和関数展開といった研究もないわけではないが、）展開項数を M とすれば計算量のオーダーは $O(NM)$ となって、M が大きいと計算量は多い。しかし、粒子分布が球対称に近ければ、$M \ll N$ として良いので計算量を大きく減らすことが出来る。

これらの方法の欠点は、空間分解能や計算精度が、格子間隔や展開項数によって制限されることである。粒子系がもともと連続系の近似として導入されている場合にはこの分解能の制限は問題にならないことが多い。こういった場合にはこれらの方法は非常に有効である。

しかし、多体系で空間分解能の制限が問題になることも多い。典型的な例は分子動力学計算である。ここで

は、計算に使う粒子は原子そのものであり、近接原子間の相互作用が十分な計算精度で求まっていなければならない。仮に FFT を使って計算しようとしたとすると、これは、格子間隔を平均的な原子間距離よりもずっと小さくとらないといけないということを意味する。従って、$M \gg N$ となってしまい、こういった場合には FFT を使うのは現実的ではない。

もっとも、この場合には、相互作用を遠距離成分と近距離成分に切り分け、遠距離成分には FFT を使い近距離成分は直接計算する方法 (Particle-Particle Particle-Mesh あるいは Particle-Mesh Ewald) が有効である。

しかし、粒子分布が一様から遠い場合には、これらの方法ではうまく対応出来なくなる。直接計算する部分の計算量が、密度が高いところで非常に大きくなってしまうからである。また、いうまでもないが FFT を使う場合には境界条件が周期的でないと話が面倒になる。さらに、計算精度を上げるのも大変であり、高い精度を要求すると急速に計算量が増える。

ツリー法や高速多重極展開法は、粒子分布が一様から遠く離れている場合でも適用でき、自由境界（無限遠でポテンシャルが 0）の場合にも適用出来る（周期境界は逆に面倒になるが）。また、高次にしても計算量の増え方が比較的に小さいので、高精度の計算が現実的な計算時間でできる。

3. ツリー法の原理

ツリー法や高速多重極展開法の基本的な考えは、遠くの粒子を適当にまとめるというものである。例えば遠くの方の粒子のかたまりをそれらの重心で置き換えてもいいし、高い精度が必要なら多重極展開も考えられる。遠くにいくに従ってかたまりを大きくしていけば、直観的には一つの粒子への力を $O(\log N)$ で計算でき、全粒子への力であればその N 倍の $O(N \log N)$ で計算できることになる。

もちろん、このようにまとめられるのは相互作用の性質による。つまり、相互作用が粒子間距離の関数であり、遠くなればゼロに近付く、また、空間微分も同様にゼロに近付くという性質があれば、ある計算精度を達成しようと思えば、遠くはまとめて扱い、近くは細かくみるというやりかたができるわけである。

しかし、ここで問題なのはどうやってうまく粒子をかたまりに分けるかということであろう。ある粒子への力を計算するのに適した分け方を見つけるには、すくなくとも一度は全粒子をみないといけない。従って、その計算量は $O(N)$ より小さくはない。粒子毎に別の分け方をしたら、それだけで計算量が $O(N^2)$ になってしまう。

この問題は、すべての粒子に使える汎用の分け方というのをあらかじめ構成することができれば解決する。それをするのがツリー構造による階層的な空間分割ということになる。

図で説明しよう。3 次元の図を書くのは面倒なので、2 次元で書くことにする（図 1）。まず、平面内の適当な領域に粒子が分布しているとしよう。粒子が有限個なので、そのすべてを含む正方形を考えることが出来る。すべて含んでいればよくて別に最小である必要はないが、普通はそこそこ小さくとる。これを、まず 4 つの小正方形 (セル) にわける。で、そのそれぞれをさらに 4 つにわけるというのを再帰的に繰り返していく。

通常の実装では、セルの中の粒子が 0 個または 1 個になったところで止める。この空間（平面）分割に対応したツリー構造を考えると、根に全体の正方形に対応する節点があり、その下に 4 つの小正方形、さらにその下にそのそれぞれの中の 4 つ $\cdots$ と続いていって、ツリーの葉には粒子 1 つ 1 つがくることになる。この方法では、粒子の数密度が高いところでは細かく、そうでないところでは粗いという意味で、適応的な構造が実現されていることに注意してほしい。3 次元にすれば、正方形 4 個の代わりに立方体 8 個となるが、それ以外は全く同じである。図 2 に、図 1 に対応するツリーの構造を示す。

どうやってこのツリー構造を使って一つの粒子への重力が計算できるかということを考えてみる。このためには、ツリーのある節点が表す立方体内の粒子全体からのある粒子への力を計算する方法を与えればよい。（なお、以下では、節点に対応する立方体、あるいはそのなかの粒子全体のことも単に節点ということがある）系全体からの力は、単に根節点からの力として計算できる。一つの節点からの力は以下のように計算できる：

$$\text{ある節点からの力} = \begin{cases} \text{その節点の重心からの力} \\ \text{(節点が「十分離れている」)} \\ \text{子節点からの力の合計} \\ \text{(それ以外)} \end{cases} \tag{1}$$

これは再帰的に定義されている。簡単にプログラムを書くには、再帰呼びだしが使える言語を使えばいい。再帰呼びだしが使えない Fortran 77 のような言語で書く場合には、アルゴリズムを繰り返しの形に変形すればよい。これについては後で少し触れる。

「十分離れている」かどうかの判定には通常は以下のような基準を使う。

$$\frac{l}{d} < \theta \tag{2}$$

ここで d は粒子と節点の重心の距離、l は節点に対応する立方体の一辺の長さである。パラメータ θ は見込み角といわれる量であり、計算精度と計算量を制御する。高い精度を得たければ θ を小さくすればいいが、漸近的には θ^{-3} に比例して計算量が増える。計算精度をあげるもう一つの方法は、重心だけを考えるのでなく多重極展開を作っておくことである。

原点中心で半径 a の球内にある粒子が球の外に作るポ

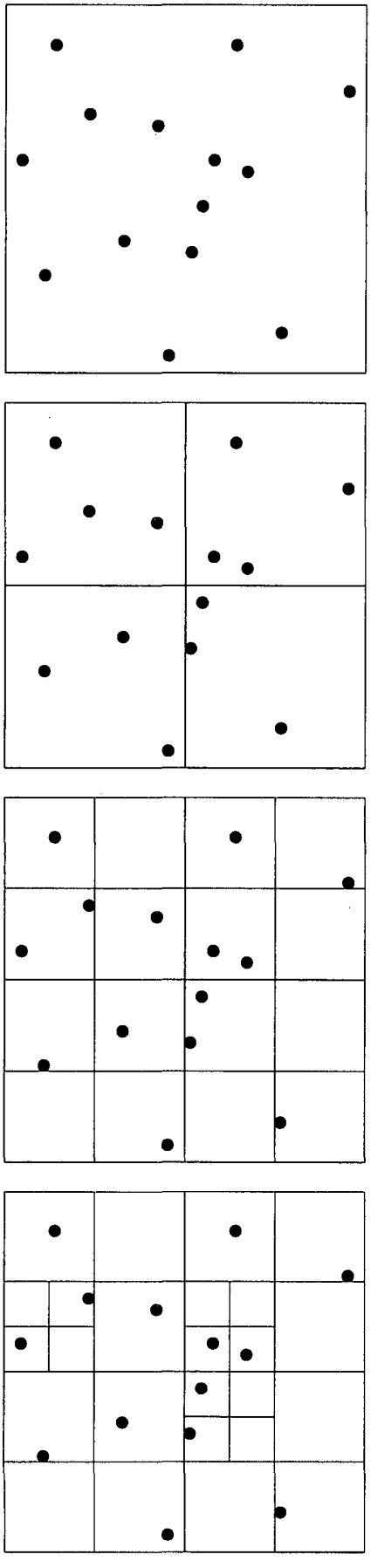
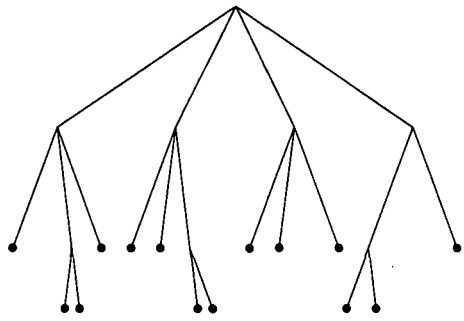


図- 1　4 分木の構築。上から順に細かく分割している。



図–2　4 分木構造

テンシャルの多重極展開係数は

$$\alpha_l^m = \sum_{i=1}^{N} m_i \left(\frac{r_i}{a}\right)^l Y_l^{-m}(\theta_i, \phi_i) \qquad (3)$$

で与えられる。ここで、m_i と (r_i, θ_i, ϕ_i) はそれぞれは粒子 i の質量と極座標で表した位置である。また、$Y_l^m(\theta, \phi)$ は l 次の球面調和関数で、ルジャンドル陪関数 P_l^m を使って

$$Y_l^m = (-1)^m \sqrt{\frac{2l+1}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}} P_l^m(\cos\theta) e^{im\phi} \qquad (4)$$

と書ける。この展開係数を使うと、位置 (r, θ, ϕ) $(r > a)$ でのポテンシャルは

$$\Phi(r, \theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \alpha_l^m \frac{a^l}{r^{l+1}} Y_l^m(\theta, \phi). \qquad (5)$$

と書けることになる。

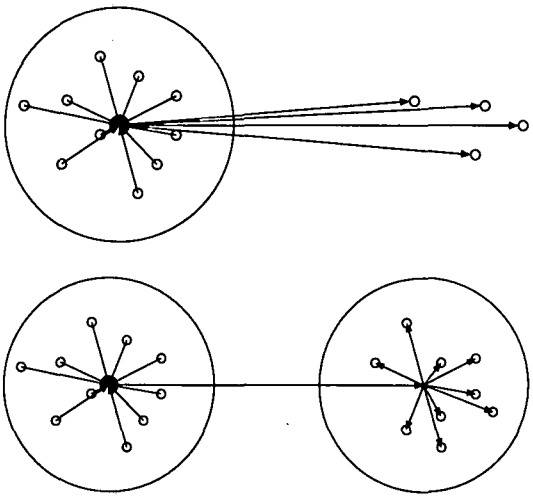
なお、多重極展開を使う場合には、上のような見込み角を使って判定する方法はあまり良くない。誤差項を評価して分割するかどうかを判定するのが望ましい。

ツリー法を実際に使うには、各節点についてその中の粒子全体の重心の位置と質量（あるいは多重極展開の展開係数）をあらかじめ求めておく必要がある。これらはやはり再帰的に定義することができる。つまり、子節点の中の粒子全体の重心の位置と質量がわかっていれば、親節点の情報は計算できる。多重極展開の展開係数についても（計算は繁雑であるが原理的には）おなじことである。

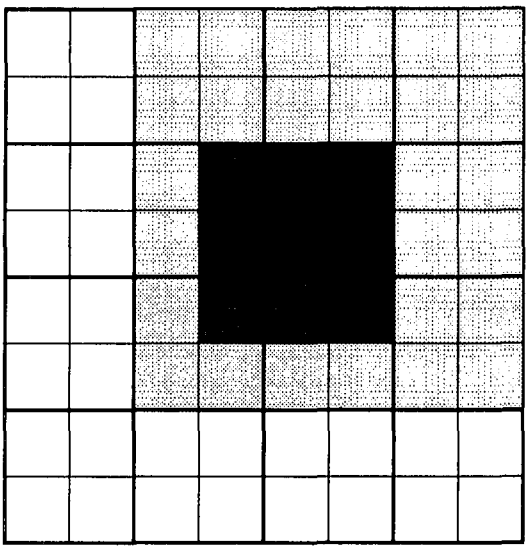
ここではとりあえず原理について述べたが、実際にプログラムにする際には計算量の最適化、ベクトル化、並列化等考えるべきことが多々ある。これらについては後で述べる。

4.　FMM の原理

ツリー法と FMM の違いはそれほど大きなものではない。ツリー法では、ある粒子への力を計算するのに、遠くの粒子からの力はまとめて計算することにした。相互



図–3 ツリー法（上）とFMM（下）



図–4 FMMの概念。もっとも濃く塗ったセルに入っている粒子への力を評価する。自分とその回りの8個のセルからの寄与は直接計算し、さらにその回りの27個は多重極展開を使う。その上のレベルは親セルが面倒を見る。

作用は対称的であるので、逆にある粒子から遠くの粒子への力をまとめて計算するということも考えられる。両方を同時にやれば、ある一群の粒子から別の一群の粒子への力をまとめて計算することになる(図3)。このように、受ける側もまとめてやることによって、以下に説明するようにツリー法の $O(N \log N)$ から $O(N)$ へ計算量のオーダーを下げることができる。このためにこの方法には「高速」多重極展開法という名前がついている。

まとめて評価するというのは実際にはどういうことかというと、セルの中心での重力ポテンシャルのテイラー展開を求めておいて、その展開を各粒子の位置であらためて評価するということになる。もちろん、実際にはポテンシャルは調和関数になっているので、球面調和関数で展開することで項数をテイラー展開に比べて減らすことができる。この展開のことを、以下局所展開と呼ぶ。

この方法は、Rokhlin と Greengard によってまず2次元の場合に提案され[1]、すぐに3次元に拡張された[7]。考え方は2次元でも3次元でも同じであるので、例によって図は2次元で示す（図4）。ツリー法では適応的なツリー構造を扱ったが、ここではとりあえず一様なツリー構造を考える。適応的なツリーでも話はほぼ同様であるが、アルゴリズムがすこし面倒になる[8]。

一様なツリーの場合、ツリーのレベルを k とすると、元の正方形は 4^k 個の小正方形に分割されている。ツリー法の場合と同様に、あらかじめ各セルに含まれる粒子（一つも無いこともありえるが）が作るポテンシャルの多重極展開は準備しておく。

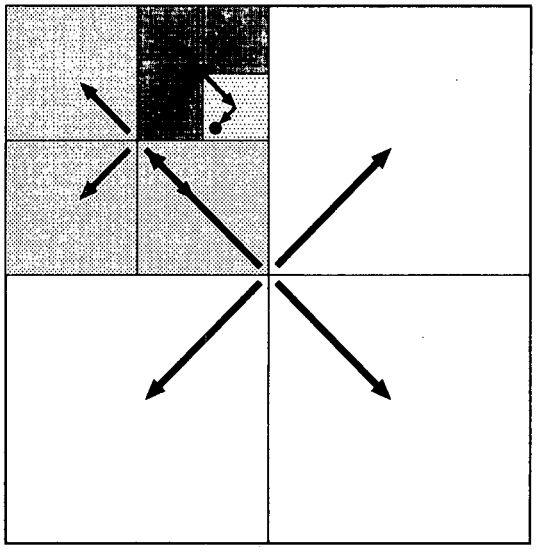
FMMでは、一つの粒子への力を以下のように分割する

$$\boldsymbol{F} = \boldsymbol{F}_d + \boldsymbol{F}_c \tag{6}$$

ここで、$\boldsymbol{F}_d$ は直接計算する分で、自分のいるセルと、それに隣接したセルの粒子からの力である。これらには多重極展開は使えないので、直接計算することになる。残るのが $\boldsymbol{F}_c$ であるが、これは多重極展開で計算する。具体的には、まずツリーのレベル毎に別に計算する。つまり、あるセル i に対し、(i) そのセルと同じレベルにあって、(ii) その親セルとその隣接8セルの合計9セルの子であって、(iii) 問題のセルには隣接していない、27 $(= 36 - 9)$ 個のセルを考える。この27個のセルに含まれる粒子が作る重力ポテンシャルの局所展開の和を、セル i の中心で評価しておく。トップレベル（系全体）と、そのすぐ下のレベルでは、親のレベルでは隣接していて自分のレベルでは隣接していないものは存在しないので、計算の必要はない。

なお、3次元では27個ではなく189個になる。さらに細かいことをいえば、図4では1つおいた先はもう多重極展開をつかってよいことにしているが、これでは多重極展開と局所展開の収束条件ぎりぎりであってあまり精度が良くない。2つおいた先をとることにすると、189個が875個に増える。もっとも、この875個というのはあまりに多いし、不要に精度が高くなっているところもあるので、収束判定をこまめに見て1つ上のレベルのセルの展開をつかっていいところはそれで済まそうといった細かい調節も行なわれている。

このようにして各レベルで重力場を計算しておくと、あとは粒子の位置で、それぞれのレベルでの局所展開を評価して合計すればいい。が、別に粒子毎に全レベルでの展開を評価しなくても、上のレベルの展開は一段下に移してやって（展開の中心をシフトして）まとめておけば、粒子は全部まとめた局所展開を一度だけ評価すればいい。この、局所展開をシフトしながら下に落していくのは、最初に多重極展開をやはりシフトしながら上にあ



図–5 局所展開のシフトと加算

げていくのとちょうど対称的な操作になる。

図5のように、4レベルのセルの中心での展開を合計する場合を考えてみる。まず、一番上のレベルのセルの中心での局所展開を、4つの子セルの中心に展開中心をシフトしてそれぞれのレベルでの重力場の局所展開と足し合わせる。さらに、それぞれの子セルで、合計した局所展開をまたそれらの4つの子セルの中心にシフトし、またそこでの局所展開と足し合わせる。これを最下層のセルに達するまで繰り返すと、各セルの中心での、自分と自分に隣接するセルに入っている粒子を除いた他のすべての粒子による力の局所展開がもとまる。粒子一つへの力は、この最下層での局所展開を粒子の位置で評価したものと、隣接セルの粒子からの力の合計ということになる。

ここで計算量のオーダーを考えてみると、すべての操作の計算量が粒子数またはセルの数にしか比例していないことがわかる。セル数は粒子数に比例するようにとれるので、結局計算量のオーダーが $O(N)$ ということになるわけである。

ただし、ここで注意して欲しいことは、FMMの場合セルの操作（多重極展開から局所展開への変換とか、それらの中心のシフト）の計算量は展開の最高次数を p として p^4 のオーダーになるということである。これは、展開係数が p^2 個あり、さらに展開を変換する行列が密行列になるからである。したがって、あまりなにも考えないでプログラムを書くと、計算量は $O(Np^4)$ になってしまう。これに対してツリー法の場合には、子セルの多重極展開をシフトして親セルの展開を作っていくところは Np^4 であるが、もっとも計算量の多い各粒子に力を及ぼすセルの多重極展開を評価するところは $N \log Np^2$ にしかならない。

こう書くと、一見ツリーのほうが得なようにも見えるが、実は FMM の計算量は Np^2 に下げることが出来る。すなわち、最初に空間をセルに分割してツリー構造を作る時に、粒子1個まで分割しないで p^2 個（のオーダー）の粒子が入っているところで止めてしまうのである。こうすればセルの総数が N/p^2 個になるので、セルに関係する計算量は Np^2 で済む。もちろん、そのかわり粒子間相互作用を直接計算する分の計算量がもともと p に依存しなかったのが Np^2 になる。つまり、近傍を直接計算する分を増やして、多重極展開で計算する分の計算量が等しくなるようにツリーの深さを最適化することで、計算量を Np^4 から Np^2 に減らせるわけである。

というわけで、ツリーと FMM の計算量の違いはあまり大きくなく、理論的にどちらが優れているというのは難しい。以下では、実験的な比較や、高速化のためのいくつかの手法を見ていくことにする。

5. ツリー法、FMM の高速化

ツリー法、FMM のどちらにしても、直接計算よりも速く計算できるというところに値打ちがある。アルゴリズムの理論的な研究としては、計算量のオーダー評価とか理論的なオペレーション数を数えるとかして少なければよい方法であるということになるが、実際に使う側から見れば、理論的に良いかどうかよりも

- 容易にプログラムが作成、デバッグでき
- 現実に使える高速な計算機の上に実装して速い

ということが重要になる。以下では、それぞれどのような研究がなされていて、どの程度の成果をあげているかを簡単にみていきたい。

5.1 理論的に良い方法

ツリー法については、多重極展開が p^2 個あってそれを評価するというのはもうどうしようもないので、あまり改良の余地はない。これに対し、FMM では特に多重極展開を局所展開に変換するところでさまざまな方法が提案されている。いくつか見ていくことにしよう。

多重極展開から局所展開への変換は、結局展開係数のベクトルに変換行列を掛ける操作になる。Greengard と Rokhlin [7]はこの変換を FFT を使って高速化できることを示唆した。Elliott と Board はこの方法を実現し、詳細な性能評価を行なった[9]。例えば $p = 16$ までとったときには変換自体は3倍ほど高速化されている。ただし、上に述べた直接計算とのトレードオフがあるので、計算全体の高速化は1.7倍程度ということになる。また、通常は $p = 16$ というようなところまでとることはまずあり得ないので、実際の効果はさらに小さい。

FFT を使う方法は、理屈はともかく実装は非常に大変で、原論文を読むと苦労がうかがえるのと同時にあまり自分でやりたいという気は起きなくなる。これに対し、「コロンブスの卵」的なアイディアで変換の計算量

を p^4 から p^3 に減らす方法も提案されている。この方法では、変換を一度に行なうのではなく、まず z 軸を移動の方向に持ってくるような回転を行なう。すると、ϕ 方向の次数 m が違うものはカップリングがなくなるので、変換行列は変換前と変換後で m が同じ p^3 個以外はゼロになり、計算量が減るわけである。変換前と変換後に回転する必要があるが、この回転の計算量自体も p^3 である。したがって、変換は p^3 で済み、最適化すれば $p^{3/2}$ となる。これは FFT よりも漸近的には計算量が大きいが、実用的な p の値では同じ程度の効果があり、実装はずっと簡単である。

この他にも同じようなアイディアの方法がいくつも提案されているが、効能も大差ないようなのでここでは省略する。

5.2 プログラム作成を容易にするような方法

FMM を実装する上での大きな障害の一つは、定式化、特に球面調和関数のシフトや多重極展開から局所展開への変換が複雑であり、プログラムの開発や高速化が難しいということであった。

Anderson [10]は実装をずっと単純にする画期的な方法を提案した。この方法を使うとプログラムが簡単になるだけでなく実行速度の観点からもメリットがあるという結果が得られている[11]。以下、この方法を簡単に紹介しよう。

基本的なアイディアは、球面調和関数の展開係数をデータとして使う代わりに球面でのポテンシャルの値そのものを使おうというものである。半径 a の球の中の粒子が外に及ぼすポテンシャルは、球の表面でのポテンシャルの値がわかっていればラプラス方程式の境界値問題を解けば与えられる。解はポアソンの公式を使って以下のように書ける

$$\Psi(\boldsymbol{x}) = \frac{1}{4\pi}\int_S \left[\sum_{n=0}^{\infty}(2n+1)\left(\frac{a}{r}\right)^{n+1} \mathrm{P}_n(\boldsymbol{s}\cdot\boldsymbol{x}/|x|)\right] \Psi(a\boldsymbol{s})d\boldsymbol{s} \qquad (7)$$

ここで P_n は n 次ルジャンドル関数である。この積分を、球面上での適当な標本点を使った数値積分に置き換える。2 次元であれば、円を等分割すればいい。、3 次元では、数値積分が球面調和関数の直交性を満たしている必要がある。そのような置き方は良く研究されていて、文献[12),13)]を見れば数値的にそのような点が求められているので、それを使えばよい。なお、技術的な注意としては、エリアシングによる誤差を防ぐために数値積分で表現出来ている次数に応じて中の無限和を適当なところで打ち切る必要がある。

子セルの外の多重極展開をその一つ上のセルでのそれに変換するには、上の式に従って親セルを含む球面上でのポテンシャルを求めればいい。また、局所展開も、$1/r$ での展開が r での展開に変わるだけで上と同じ式である。さらに、多重極展開を局所展開に変換するには、上の式をつかって局所展開する球面上でポテンシャルを評価するればよい。つまり、上の式だけで、FMM に必要な数学がすべて表現されていることになる。これに対し、オリジナルの FMM のための変換式は全部書くと 1 ページいっぱいに式が並ぶようなものであった。

Anderson の方法では別に計算量のオーダーが減るわけではなく、変換の計算量は p^4 のままであるので理論的には前節で述べた方法に比べて良い点はない。しかし、式が簡単になっていることでプログラムの最適化やチューニングがずっと容易になるという現実的なメリットはけっして小さくない。

ここで、著者らが最近提案した $\mathrm{P^2M^2}$ 法 (pseudo-particle multipole method)[14),15)] にも簡単に触れさせていただきたい。基本的な考え方は Anderson の方法とほとんど同じものであるが、違いは球面上でのポテンシャル分布を使うのではなく球面上の質量分布によって多重極展開を表現するというものである。定式化は Anderson の方法と同様に行なえ、

$$m_j = \sum_{i=1}^{N} m_i \sum_{l=0}^{p} \frac{2l+1}{K}\left(\frac{r_i}{r}\right)^l \mathrm{P}_l(\cos\gamma) \qquad (8)$$

という形で実粒子の分布から仮想粒子の質量が表現される。ここで m_i, m_j はそれぞれ実粒子、仮想粒子の質量であり、γ は展開の中心から見て実粒子と仮想粒子がなす角度、r_i は実粒子の原点からの距離、r は仮想粒子を置く球の半径、K は仮想粒子の数である。子セルの展開を親セルの展開にシフトするには、子セルの仮想粒子を実粒子であると思えば同じ式が使える。

局所展開については、Anderson の方法を使ってもよいし、上と同様に仮想粒子を置くという形で表現することもできる。現在のところ、我々はツリー法との組み合わせでしか実装していないので、こちらについては省略する。

普通のプログラマブルな計算機の上では、$\mathrm{P^2M^2}$ 法には Anderson の方法同様にプログラムが簡単であるということ以上になにかメリットがあるわけではない。しかし、専用ハードウェアとの組み合わせでは大きなメリットがある。我々は GRAPE[16),17)] という重力相互作用だけを高速に計算する専用計算機を開発しており、これと組み合わせると $\mathrm{P^2M^2}$ 法はセルからの重力も専用計算機で計算出来て画期的な高速化ができるのである。

5.3 高速な計算機への実装

ツリー法や FMM が提案されたのは 1980 年代中頃であり、大規模計算にはベクトル計算機や場合によっては並列計算機が使われるようになったころである。特に実装が簡単なツリー法は提案後すぐにベクトル化、並列化が精力的に研究された。

まず、ベクトル化について簡単にまとめておこう。並列化についても、共有メモリ型の機械では同じアイディ

アで行なえる。以下、アルゴリズムのなかでもっとも計算量が多いツリーを再帰的にたどって粒子への重力を計算するところに話を絞る。全体については文献[18]や原論文[19),20),21)]を参照されたい。

ベクトル化のためには、アルゴリズムの並列性を生かす必要がある。ツリーをたどるところでは2種の並列性がある。一つは別の粒子への重力は独立に計算出来るという並列性であり、もう一つはある粒子への重力のうち、ツリーのなかで上下関係にないセルからの力は独立に計算できるというものである。

前者の並列性を使うには、粒子全部を一度に見るような形にプログラムを書き換えればよい。これには、まず再帰によって表現されたアルゴリズムをスタックやポインタを使う明示的な繰り返しループに変形し、さらに一番外側の粒子を順番に処理するというループを一番内側に持ってくる。これはかなり繁雑な作業になるが、ツリー法の場合はもともとのアルゴリズムが単純であるので注意すれば出来る。後者の並列性を使うには、やはり再帰を明示的な横型探索 (Breadth First Search) のループに書き換え、ツリーの1階層づつをベクトル化されたループで処理していく。前者の方法のほうがベクトル長は長く、一般には高い性能が出るようである。

分散メモリの並列計算機の場合には、アルゴリズムの並列性だけでは十分ではなく、どのように空間分割をして粒子と領域をプロセッサに割り付けるかが問題となる。逆にいえば、それが決まれば並列化そのものはそれほど複雑ではない。メッセージパッシングの機械では Warren と Salmon による ORB（再帰的直交分割）や Morton ordering による領域分割が良く知られている[22),23)]。物理的に共有メモリである機械や、分散共有メモリの場合は、Morton ordering によって粒子をソートしておくことで自動的に空間分割が実現でき、かなり良い性能が得られている[24)]。また、最近は HPF などのデータ並列型の言語を使った並列化も試みられている[25)]。

前節で触れたが、専用計算機との組み合わせについてもここでまとめておこう。

筆者らのグループでは、GRAPE という名前で粒子間相互作用を計算することだけに特化した計算機を開発し、実際に天文学の研究に使っている[17),16),26)]。FMM やツリー法のような速い方法があるのに、そもそもこういった計算機を作ることに意味があるのだろうか。

もちろん、我々は意味があるから作っているつもりであるが、その意味は実は2つあると考えている。一つは、粒子毎にタイムスケールが大きく違うような問題では、ツリー法や FMM を使うのは非常に難しい、特に並列計算機で使えるような実現法はまだ存在しないということである。直接計算の場合には、多重時間刻みに適した高精度な積分公式が知られていて、これは並列化や専用計算機への実装も可能である。しかし、ツリー法や FMM の場合、特に並列化では多数の粒子への重力を並列に計算できるということを利用しており、多重時間刻みにするのは難しい。また、このような問題では要求される計算精度も高いのが普通であり、直接計算に対する優位性がそもそも小さいのである。

もうひとつの理由は、ツリー法も FMM も、実は直接計算の部分を高速化するだけで非常に速くなるということである。例えばツリー法の場合、それほど計算精度がいらない場合には、セルからの力はその重心からの力（モノポール）と思って良い。これはモノポールといっても実際には1次の項まで入っているし、セルが立方体なので高次の項の寄与は想像するより小さくなる傾向があるからである。この時には、粒子からの重力が計算出来ればよい。また、実際に高次の項まで入れる場合でも、先に述べた疑似粒子多重極法を使えば粒子で表現できる。

もっとも、順番にツリーをたどっていくような計算自体をハードウェアで実現するのは、不可能ではないかもしれないが面倒である。我々は、Barnes によるベクトル化の方法[21)]を応用して、汎用計算機の方でツリーをたどるがその回数を減らすような工夫をして使っている。

専用計算機の効果であるが、以下に実例をあげてみよう。昨年度の Gordon-Bell 賞のコスト・パフォーマンス部門で、DEC Alpha のクラスター 70 台を並列に使うという計算が、賞をとったわけではないが最終選考まで残った。これは 533MHz の Alpha 70 台で並列ツリー法のプログラムを走らせ、実効 6.8 Gflops を達成したというものである。これに対し、我々は今年度の Gordon Bell 賞に昨年完成した GRAPE-5 を使った同じツリー法の計算で応募した。我々の計算は、DEC (Compaq) Alpha のホスト一台に GRAPE-5 ボード2枚をつけた計算機で、昨年の Alpha 70 台とほぼ同じ実効速度を得た。（理論ピーク性能も同程度である）コストパフォーマンスでは数倍良くなっているし、占有スペースや消費電力ではほぼ2桁良い。これは一例であるが、問題にあまりよらないでこの程度の性能を得ることができ、専用計算機のメリットは非常に大きい。

6. それぞれの方法の歴史と現状

FMM とツリー法のどちらも、まだ10年少々の歴史しかない新しい方法である。しかし、特に 90年代に入ってから急速に研究が進み、すでに簡単に述べたように応用分野も広まりつつある。基本的な考え方は、遠くとの相互作用はまとめるという単純なものなので、重力・クーロン相互作用に限らず非常にさまざまな応用がありえるわけである。

ここでは、特に筆者の専門である重力多体系の観点から過去の歴史を簡単に振り返ってみたい。

ツリー法を初めて実現したのは、プリンストン大学の

学部生 A. Appel であった[2]。これは 1985 年になるまで公表論文にはならなかった。彼の方法は、ツリーを最近接粒子をまとめていくことによってボトムアップに構成していくものであり、現在広く使われている 8 分木を使うものではなかった。8 分木を使うツリー法を提案したのは当時プリンストン高等研究所にいた J. E. Barnes と P. Hut である[3]。この方法に基づいて、その後の数年間に既に述べたベクトル型スーパーコンピュータへの効率的な実装についての研究と分散メモリ型計算機への実装の研究が精力的に進められ、銀河や銀河団、あるいは宇宙の大規模構造などの形成、進化のシミュレーションに広く使われるようになった。特に分散メモリの超並列計算機では、10^9 を越えるような粒子数のシミュレーションが可能になってきている。

このように広く使われることになった理由の一つは、実装が比較的簡単であったことである。ツリー法の実装のほとんどは、4 重極モーメントまでしか扱わない。このため、多重極展開や球面調和関数といってもそれほど面倒な計算式が出てくるわけではない。高い計算精度が必要であれば、離れたセルもある程度まで分割して相互作用を計算する。もちろん、非常に高い計算精度が必要になれば、より高次のモーメントをとり入れた方が計算量が減る。しかし、天文学の応用の多くでは、粒子数が有限であることからくる particle noise で計算精度がきまっているので、相互作用の計算精度をあげるよりも、計算精度を下げても粒子数を増やしたほうがよりよい結果が得られるのである。このために、かなり広い応用で、4 重極モーメントよりも精度をあげることには意味がない。

このように、理論的にも実験的にも複雑なことをして精度をあげる必要がなかったために、ベクトル化や並列化、特に適応的な木構造の実現やロードバランシングなどに努力が傾注されてきた。これらの努力により高速で実用的なプログラムが利用可能になってきている。

FMM はツリー法とほぼ同時期にイェール大学の Greengard と Rokhlin により提案された[1]。これは多重極展開と局所展開をまともに使うものであった。その後、計算法の改良について非常に多様な提案がなされてきた。しかし、このような多様な提案があるということは、逆にいえばまだ決定版というべきものはなく、多数の改善案のうち一つが他のものよりも圧倒的によいというわけではないということでもある。さらに、このような方法は一般に高い精度を要求する場合にしか改善につながらない。

実際、適応的な木構造の場合にツリー法と FMM の計算速度を比較した研究[11]では、比較的精度が低い（ポテンシャルの相対精度が 10^{-3} 程度）場合、FMM はツリー法に比べて有意に遅いという結果になっている。他の人の実験でも同様な結果になっており、現在のところ天文シミュレーションにおいてはツリー法は広く使われているが FMM は全く使われていない。

一様に近い分布の場合については、精度が必要でなければ PM に対抗出来ないし、また高い精度が必要な場合でも、Ewald 法や P^3M 法[27]（これは最近は Particle-Mesh Ewald とよばれることもあるが、本質的には同じものである）との優劣が問題になる。実験的には P^3M 法のほうが優勢であるようである[28]。

なお、周期境界をおかない場合には FMM のほうが FFT に基づく方法よりも有利になりうることはいうまでもない。また、2 次元の場合には 3 次元に比べて FMM の計算量は大きく減り、FMM は実用性の高い方法となる。これは、多重極展開の項数が p^2 から p に減るだけでなく、幾何学の違いのためにセル間相互作用の数が減るためである。3 次元の場合は一つのセルが $6^3 - 3^3 = 189$ 個のセルと相互作用するが、2 次元の場合は $6^2 - 3^2 = 27$ 個であり、7 倍計算量が違うことになる。

なお、後者の幾何学の違いは、空間が 3 次元であっても実際に計算する領域は 2 次元である境界要素法の場合にも有利に働くことを注意しておきたい。このために、FMM やツリー法は境界要素法には特に有効である。

7. まとめ

本稿では、1980 年代後半に提案されて急速に研究が進んだツリー法と FMM についてその原理と現状を、特に多体シミュレーションに使う側の観点から簡単にまとめた。天文シミュレーションでは、精度の要求が低いこともあって実装が容易なツリー法が広く使われている。FMM が実用上ツリー法や P^3M 法と対抗できるようになるかどうかは依然未知数である。

なお、筆者の能力不足もあり、多体シミュレーション以外の分野への応用についてはほとんど触れることができなかった。特に境界要素法への応用では、幾何学の違いもあって適応的な FMM で非常によい結果が得られているようである。こちらの最近の進展については、先に紹介した文献[6),5),4)]等を御覧いただきたい。

参考文献

1) Greengard L. and Rokhlin V.: A fast algorithm for particle simulations. *Journal of Computational Physics*, Vol.73, pp.325–348, 1987.
2) Appel A. W.: An efficient program for many-body simulation, *SIAM Journal on Scientific and Statistical Computing*, Vol.6, pp.85–103, January 1985.
3) Barnes J. and Hut P.: A hiearchical O(n log n) force calculation algorithm, *Nature*, Vol.324, pp.446–449, 1986.
4) 福井卓雄, 服部純一, 土居野優: 高速多重極法の境界要素解析への応用, 構造工学論文集, Vol.43, pp.373–382 1997.
5) 西村直志, 吉田研一, 小林昭一: 多重極積分方程式法による3次元クラック問題の解析について, 境界要素法論文集, Vol.14, pp.37–42, 1997.
6) Greengard L. and Wandzura S.: Guest Editor's Introduction: Fast Multipole Methods, *Computational Science and Enginnering*, Vol.5, pp.16-18, 1998.
7) Greengard L. and Rokhlin V.: Rapid evaluation of potential fields in three dimensions, In C. Anderson and C. Greengard (ed) *Voltex Methods*, pp.121–141, Springer-Verlag, Berlin, 1988.
8) Board J. A., Jr, Hakura Z. S., Elliott W. D., and Rankin W. T.: Scalable variants of multipole-based algorithms for molecular dynamics applications. In D. H. Bailey *et al.* (ed) *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, pp.295–300. SIAM, Philadelphia, 1995.
9) Elliott W. D. and Board J. A., Jr.: Fast fourier transform accelerated fast multipole algorithm. *SIAM Journal on Scientific Computing*, Vol.17, pp.398–415, 1996.
10) Anderson C. R.: An implementation of the fast multipole method without multipoles. *SIAM Journal on Scientific and Statistical Computing*, Vol.13, pp.923–947, 1992.
11) Blackston D. and Suel T.: Highly portable and efficient implementations of parallel adaptive n-body methods. In *Proceedings of SC97*, (CD-ROM). ACM, 1997.
12) McLaren A. D.: Optimal numerical integration on a sphere. *Math. Comput.*, Vol.17, pp.361–383, 1963.
13) Hardin R. H. and Sloane N. J. A.: Mclaren's improved snub cube and other new spherical designs in three dimensions, *Discrete and Computational Geometry*, Vol.15, pp.429–441, 1996.
14) Makino J.: Yet another fast multipole method without multipoles — pseudo-particle multipole method. *Journal of Computational Physics*, Vol.151, pp.910–920, 1999.
15) Kawai A. and Makino J.: A simple formulation of the fast multipole method: Pseudo-particle multipole method. In *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, CD-ROM, SIAM, Philadelphia, 1999.
16) Sugimoto D., Chikada Y., Makino J., Ito T., Ebisuzaki T., and Umemura M.: A special-purpose computer for gravitational many-body problems, *Nature*, Vol.345, pp.33-45, 1990.
17) Makino J. and Taiji M.: *Scientific Simulations with Special-Purpose Computers — The GRAPE Systems.* John Wiley and Sons, Chichester, 1998.
18) Pfalzner S. and Gibbon P.: *Many-body tree method in physics.* Cambridge University Press, Cambridge, 1996.
19) Makino J.: Vectorization of a treecode. *Journal of Computational Physics*, Vol.87, pp.148–160, 1990.
20) Hernquist L.: Vectorization of tree traversals. *Journal of Computational Physics*, Vol.87, pp.137–147, 1990.
21) Barnes J. E.: A modified tree code: don't laugh; it runs, *Journal of Computational Physics*, Vol.87, pp.161–170, 1990.
22) Warren M. S. and Salmon J. K.: Astrophysical N-body simulations using hierarchical tree data structures. In *Supercomputing '92*, pp.570–576. IEEE Comp. Soc., Los Alamitos, 1992.
23) Warren M. S. and Salmon J. K.: A parallel hashed oct-tree N-body algorithm. In *Supercomputing '93*, pp.12–21. IEEE Comp. Soc., Los Alamitos, 1993.
24) Holt C. and Singh J. P.: Hierarchical n-body methods on shared address space multiprocessors. In D. H. Bailey, *et al.* (ed) *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, pp.313–318. SIAM, Philadelphia, 1995.
25) Hu Y., Jonsson S. L., and Teng S.-H.: A data-parallel adaptive n-body method. In *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing.* SIAM, 1997.
26) Hut P. and Makino J.: Astrophysics on the grape family of special-purpose computers. pp *Science*, Vol.283, pp.501–505, 1999.
27) Hockney R. W. and Eastwood J. W.: *Computer Simulation Using Particles.* IOP Publishing, Ltd., Bristol, 1988.
28) Board J. A. Jr, Humphres C. W., Lambert C. G., Rankin W. T., and Toukmaji A. Y. : Ewald and multipole methods for periodic n-body problems. In *Proceedings of the Eigth SIAM Conference on Parallel Processing for Scientific Computing*, SIAM, Philadelphia, 1997.

(1999 年 4 月 23 日 受付)