ポテンシャルの高速計算について

吉田 研一，西村 直志 *

1 序

多重極法 (Fast Multipole Method, FMM) は，Rokhlin[1] の 2 次元 Laplace 方程式の積分方程式の数値解の研究に端を発した大規模問題に有利な数値計算手法のひとつであり、Greengard[2] らの多体問題への適用で一躍有名になった。この方法を FFT などと並ぶ 20 世紀の TOP10 アルゴリズムの一つとして紹介している記事 [3] がある事からもわかるように、多重極法は多くの計算科学の分野において大きな影響のあった算法であるといえよう。多重極法を用いると、従来は $O(N^2)$ の計算量を必要としていた N 体問題の相互作用の計算や境界積分方程式法 (境界要素法) の計算は $O(N)$ の演算で済む。

著者らは多重極法を用いた境界積分方程式法の高速化に関する研究を行なっている [4] が、ここでは、多重極法を用いた Coulomb ポテンシャルの計算について紹介する。

2 多重極法について

$\mathbb{R}^3$ の点 $y_i (i = 1, 2, \ldots, N)$ に電荷 q_i を持った粒子が配置されている時、位置 x でのポテンシャル $\Phi(x)$ は

$$\Phi(x) = \sum_{i=1}^{N} \frac{q_i}{|\overrightarrow{y_i x}|} \tag{1}$$

と求められる (図 1)。ただし、ある粒子の位置 y_k が

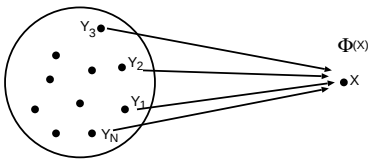

図 1. ポテンシャル $\Phi(x)$

x と一致する時は、式 (1) 中の $i = k$ の項は除くものとする。式 (1) を $x = y_j$ $(j = 1, \ldots, N)$ において評価する場合、定義に従って直接計算を行なうと、計算量は明らかに $O(N^2)$ となる。

多重極法によってある点でのポテンシャルを評価する場合、式 (1) 中の和を x の近傍にある粒子からの寄与とそれ以外の粒子からの寄与の二つに分割する。近傍からの寄与は直接計算し、その他の粒子からの寄与は多重極展開を用いて効率良く計算する。近傍であるか否かを決めるために、全ての粒子を含む立方体を考え、それを順次分割して得られるセルの作るツリー構造を用いる。以下、もう少し具体的に見て行く事とする。

まず、基本解を次のように多重極展開する。

$$\frac{1}{|\overrightarrow{yx}|} = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \overline{S_{n,m}}(\overrightarrow{Ox}) R_{n,m}(\overrightarrow{Oy}), \quad |\overrightarrow{Oy}| < |\overrightarrow{Ox}| \tag{2}$$

ここに $R_{n,m}, S_{n,m}$ は

$$\begin{aligned} R_{n,m}(\overrightarrow{Ox}) &= \frac{1}{(n+m)!} P_n^m(\cos\theta) e^{im\phi} r^n, \\ S_{n,m}(\overrightarrow{Ox}) &= (n-m)! P_n^m(\cos\theta) e^{im\phi} \frac{1}{r^{n+1}} \end{aligned}$$

で定義される体球関数であり、(r, θ, ϕ) は x の極座標、P_n^m は Legendre 陪関数である。$R_{n,m}, S_{n,m}$ は

$$S_{n,m}(\overrightarrow{yx}) = \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} \overline{R_{n',m'}}(\overrightarrow{Oy}) S_{n+n',m+m'}(\overrightarrow{Ox}) \quad |\overrightarrow{Oy}| < |\overrightarrow{Ox}| \tag{3}$$

$$R_{n,m}(\overrightarrow{yx}) = \sum_{n'=0}^{n} \sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{yO}) R_{n-n',m-m'}(\overrightarrow{Ox}) \tag{4}$$

を満たす。点 x が荷電粒子系から十分に離れている時、式 (1)、(2) より、$\Phi(x)$ は次のように多重極展開

* よしだけんいち、にしむらなおし (京都大学 学術情報メディアセンター)

される。

$$\Phi(x) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \overline{S_{n,m}}(\overrightarrow{Ox}) M_{n,m}(O) \qquad (5)$$

ここに、$M_{n,m}(O)$ は O を展開中心とする多重極モーメントで、次のように表される。

$$M_{n,m}(O) = \sum_{i=1}^{N} q_i R_{n,m}(\overrightarrow{Oy_i}) \qquad (6)$$

また、式 (4) より、展開中心を別の点 O' に移した時の多重極モーメントは次式で求められる (図 2)。

$$M_{n,m}(O') = \sum_{n'=0}^{n} \sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{O'O}) M_{n-n',m-m'}(O) \qquad (7)$$

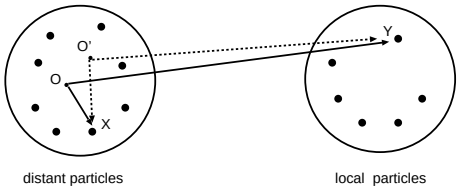

図 2. 多重極展開中心の移動

有名な Barnes と Hut のツリー法 [5] は多重極法と良く似た算法であり、式 (6), (7) を用いてポテンシャルを評価する $O(N \log N)$ のアルゴリズムを実現している。一方、多重極法ではポテンシャルの評価に多重極展開を用いるのではなく、局所展開係数を導入して計算量のオーダーを $O(N \log N)$ から $O(N)$ に軽減する。局所展開係数とは、点 x_0 の近傍の評価点 x でのポテンシャルを調和関数 $R_{n,m}(\overrightarrow{x_0 x})$ で展開した時の係数である。即ち、$\Phi(x)$ を

$$\Phi(x) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} L_{n,m}(x_0) R_{n,m}(\overrightarrow{x_0 x}) \qquad (8)$$

と展開したとき、$L_{n,m}(x_0)$ が x_0 を中心とする局所展開係数である。局所展開係数を計算しておけば、それを用いて x_0 の近傍の多数の評価点 x でのポテンシャルを高速に評価することができる (図 3)。局所展開係数は、多重極モーメント $M_{n,m}(O)$ を用いて、次のように計算できる。

$$L_{n,m}(x_0) =$$

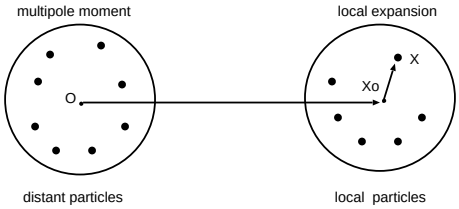

図 3. 局所展開係数の導入

$$\sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} (-1)^n \overline{S_{n+n',m+m'}}(\overrightarrow{Ox_0}) M_{n',m'}(O) \qquad (9)$$

後でわかるように、局所展開係数についても、多重極モーメントと同様に展開中心を移動する式が必要になる (図 4)。局所展開中心を x_0 から x_1 に移した時の局所展開係数は、式 (7) と同様に式 (4) を用いて次式のように得られる。

$$L_{n,m}(x_1) = \sum_{n'=n}^{\infty} \sum_{m'=-n'}^{n'} R_{n'-n,m'-m}(\overrightarrow{x_0 x_1}) L_{n',m'}(x_0) \qquad (10)$$

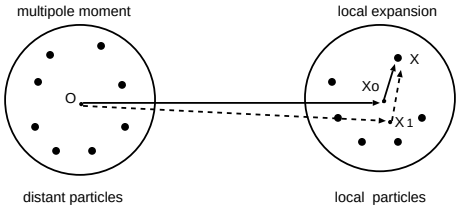

図 4. 局所展開中心の移動

3　多重極法のアルゴリズム

本節では多重極法のアルゴリズムについて簡単に述べる。アルゴリズムや計算コスト評価の詳細については Greengard[2] を参照して頂きたい。

1. (ツリー構造の決定) 全ての粒子を含む立方体を考える。これをレベル 0 のセルとする。この立方体の各辺を 2 等分して、立方体を 8 個の部分

立方体に分割する。このうち、粒子を含む立方体をレベル 1 のセルと呼ぶ。以下同様にレベル i のセルを 8 分割してレベル $i+1$ のセルを作る。また、レベル i のセル C がレベル $i+1$ のセル C' を含む時、C は C' の 親セル、セル C' はセル C の子セルと呼ぶ。セルの分割の際には、粒子を含まないものはそれ以上の分割をやめ、セルの含む粒子数が予め決めた数よりも多い場合はそのセルをさらに分割する。子セルを持たないセルのことをリーフと呼ぶ。

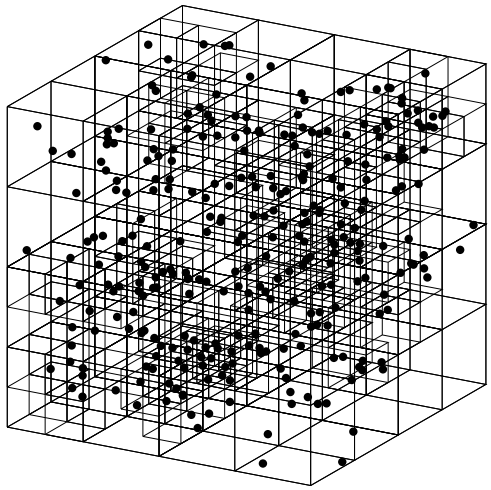

図 5. 粒子 (粒子数 100) とセルによるツリー構造

2. (多重極モーメントの計算) まずリーフであるセルにおいて、その中心回りの多重極モーメントを式 (6) で計算する。次にレベル数をひとつづつ減少させながら、各セルの中心における多重極モーメントを求める。すなわち、リーフでないセル C において、子セルの多重極モーメントを子セルの中心から、セル C の中心へ式 (7) によって移したものを、セル C が含む全ての子セルについて足し合わせる。この操作をレベル 2 のセルに到達するまで行ない、全てのレベルの全てのセルの多重極モーメントを計算する。
3. (局所展開係数の計算) まず、レベル 2 のセルからレベル数をひとつづつ増加させながらセルの中心での局所展開係数を計算していく。レベル i のセル C での局所展開係数は、C にやや近いセルからの寄与と、それより遠いセルからの影響に分けて計算する。前者は、C の親セルと隣接しているレベル $i-1$ のセルの子セルの全体から、C に隣接しているレベル i のセルを除いたものに対して式 (9) から計算したものである。また、後者は C の親セルの局所展開係数を、その展開中心を親セルのものからセル C の中心に式 (10) を使って移動したものを足しあわせて求める。
4. (ポテンシャルの評価) 最終的に各点でのポテンシャルは、リーフにおいて、求めた局所展開係数から式 (8) によって計算される値とセル C に隣接するセルからの影響を直接計算したものの和として評価される。

4 数値計算

ここではランダムな位置と電荷を持つ粒子系の Coulomb ポテンシャルを種々の方法で求め、計算時間の比較を行なう。

4.1 直接計算による計算時間の比較

まず、下に示す種々の計算機環境において直接計算で式 (1) を求め、計算時間を測定した。

(a) DEC 21264 667MHz + DEC Fortran compiler

(b) Intel Xeon 2.8GHz + Intel Fortran compiler

(c) Intel Xeon 2.8GHz(MDGRAPE-2 + BoosterCard) + Intel Fortran compiler

(d) Fujitsu VPP800

比較のために、いずれも 1CPU のみ用いている。Intel Xeon 2.8GHz 搭載の PC には MDGRAPE-2 及び BoosterCard[6] が搭載されており、(c) ではそれを使用して計算を行なっている。図 6 の ALPHA, XEON, MDGMRAPE, VPP800 は上の (a) から (d) にそれぞれ相当する。MDGRAPE-2 とは分子動力学に現れるポテンシャルの計算に特化したパイプライン方式の専用計算機 (PCI card) であり、BoosterCard は MDGRAPE-2 の計算能力を更に向上させる効果がある。なお、VPP800 は今回用いた計算機中最も

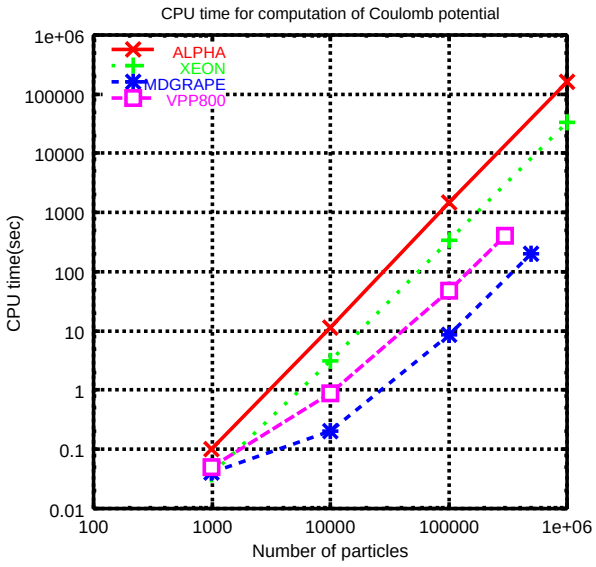

図 6. 計算時間の比較 (直接計算)

規模の大きい問題に適用可能であるが、センターの需要がピークを迎えるこの時期に業務 machine に負荷をかけるのははばかられたため図 6 の程度の問題規模に留めた。さすがに専用計算機の速さは際だっているが、1999 年に導入された VPP も健闘している。

4.2 直接計算、多重極法、ツリー法の計算時間の比較

次に、Intel Xeon 2.8GHz を搭載した PC でポテンシャルの計算を以下の 4 つの手法で行なった場合の計算時間を比較する。

(a) 直接計算

(b) FMM

(c) FMM with MDGRAPE-2 + BoosterCard

(d) ツリー法

(c) では多重極法で直接計算のサブルーチンが呼び出される度に MDGRAPE-2 及び BoosterCard を使っている。この数値計算では式 (8) の無限級数を 10 項で打ち切り、リーフが含む最大の粒子数を 1000 とした場合の計算時間の比較を行なった。その結果が、図 7 である。図中の DIRECT, FMM, FMM+MDGRAPE, BH は上の (a) から (d) にそれぞれ相当する。図 7 より、FMM の計算は MDGRAPE-2+BoosterCard の使用により更に高速化されているのがわかる。しかし、規模の小さい直接計算で MDGRAPE-2 を使うと逆に遅くなってしまうので、リーフが含む粒子の最大数をあまり小さくすると、専用計算機使用の効果が得られない。実際、リーフが含む粒子の最大数が 100 の時、MDGRAPE-2 を用いると却って遅くなるという結果も得られている。

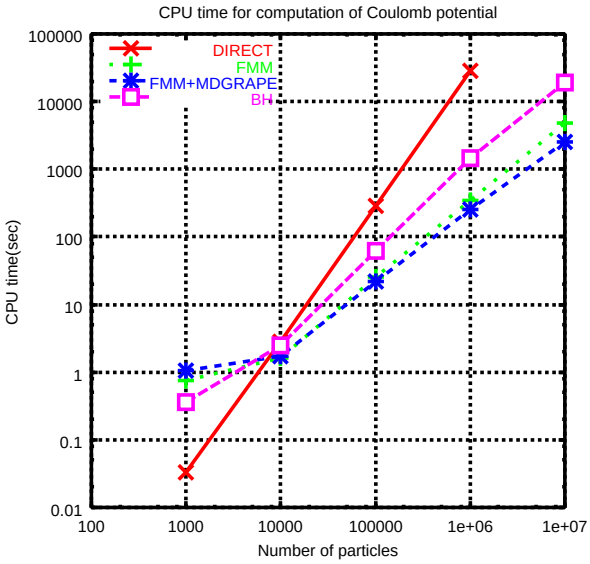

図 7. 計算時間の比較 (FMM, BH など)

図 8 は、図 6、図 7 の結果のうち、Intel Xeon を搭載した PC によって得られた特徴的な結果 (直接計算、MDGRAPE-2 を用いた直接計算、単体の FMM (リーフの最大粒子数 = 1000) を再度 plot したものである。この図からわかるように N が数十万程度までは MDGRAPE-2 による直接計算が速いが、MDGRAPE-2 1 枚で扱える最大の N では単体の FMM が GRAPE に追いつくか、ないしは若干速いと言う結果になった。以上、あくまでも一実験結果に過ぎないし、code 開発や並列化の容易さと言った点は無視した議論ではあるが、FMM と言う手法が大変優れたものである事の傍証とはなっているであろう。

最後に、式 (8) の無限級数の打ち切り項数を 10 で固定し、リーフが含む最大の粒子数を 10, 100, 1000 と変化させた時の多重極及びツリー法の計算時間を比較した (図 9)。MDGRAPE は使っていない。図

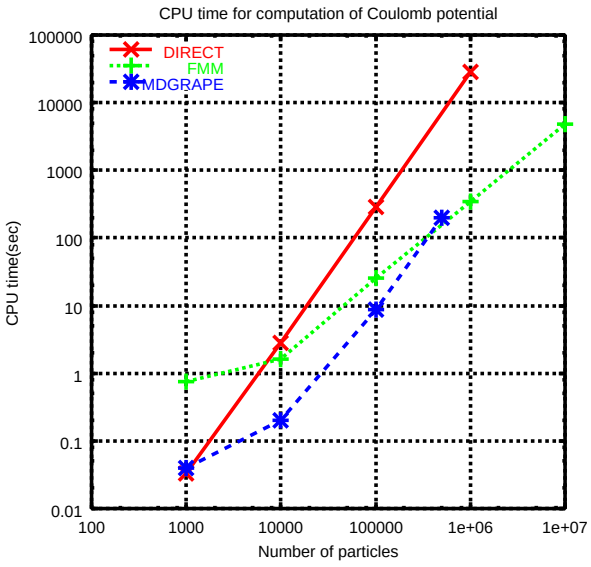

図 8. 計算時間の比較 (FMM, MDGRAPE)

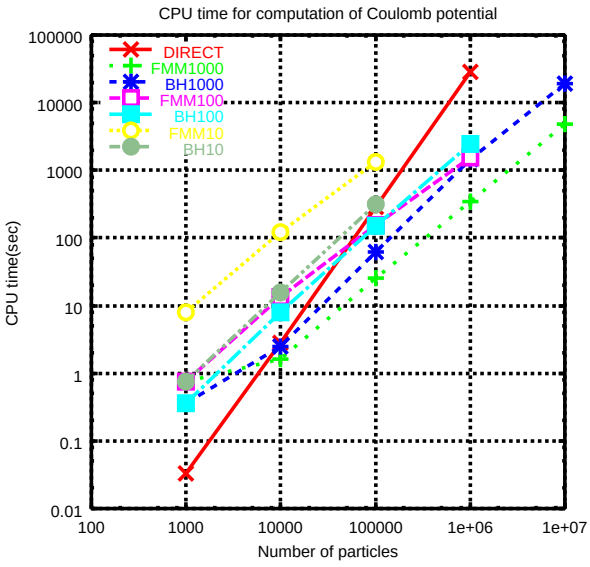

図 9. 計算時間の比較 (リーフの粒子数の影響)

中の FMMn 及び BHn はそれぞれ多重極法及びツリー法でリーフが含む最大の粒子数を n とした時の計算時間である。$n = 10$ の時はツリー法の方が速いが、n の値を大きくすると多重極法がツリー法に対して速くなっていくことがわかる。

なお、天文学などの分野ではツリー法が多重極法より速いと言われているが [7]、多重極展開の打ち切り項数などの条件によっては、ここで得られた結果と異なった傾向が得られる事も十分あり得る。また、8 分木構造以外の更に計算効率の良いデータ構造を用いたツリー法も試みられている (例えば [8])。

5 終りに

以上、最も簡単な Coulomb ポテンシャルの場合の多重極法について述べてきた。鬼才 Rokhlin の生みだしたこの強力な手法の魅力の一端でも伝える事ができたのであれば著者にとって幸いである。

多重極法は多体問題に限らず、電磁気学、音響学、弾性学などの様々な分野で研究され、その有効性が報告されている。特に境界積分方程式法への多重極法の適用については Nishimura[9] を参照して頂きたい。今後機会があれば、多重極法を用いた境界積分方程式法の並列化や MDGRAPE-2 などの利用についても検討・報告したい。

参考文献

[1] V. Rokhlin : Rapid solution of integral equations of classical potential theory, J. Comp. Phys., 60, 1985, 187–207.

[2] L. Greengard : The rapid evaluation of potential fields in particle systems, The MIT Press, 1987.

[3] J. Board and K. Schulten : The fast multipole algorithm, IEEE Comp. Sci. Eng., 2(1), 2000, 76–79.

[4] `http://gspsun1.gee.kyoto-u.ac.jp/~nchml/papers.html`
`http://gspsun1.gee.kyoto-u.ac.jp/~yoshida/research.html`

[5] J. Barnes and P. Hut : A hierarchical force calculation algorithm, Nature, 324, 1986, 446–449.

[6] `http://www.peta.co.jp`

[7] J. Makino : Fast multipole algorithm, letters to the editors, IEEE Comput. Sci. Eng., 2(3), 2000, 4.

[8] R.J. Anderson : Tree data structures for N-body simulation, SIAM J. Comput., 28, 1999, 1923–1940.

[9] N. Nishimura : Fast multipole accelerated boundary integral equation methods, Appl. Mech. Rev., 55, 2002, 299–324