ハイパフォーマンス
コンピューティング 78－1
（1999．10．7）

FMMによるLegendre陪関数変換の高速化

高見雅保† 須田礼仁† 杉原正顯†

球面調和関数はLegendre陪関数変換とFourier変換からなっている．その計算量はLegendre陪関数変換部分によって決定されるが，我々は切断周波数が M の時に，FMMによる高速多項式内挿法を用いることによって $O(M^2 \log M)$ となる手法を提案した．本稿では数値的に安定な内挿を実現する標本点選出アルゴリズムと，非一様な点の分布に対するFMMの改良を提案する．この手法をLegendre陪関数変換に適用したところ直接法に対して M =1365では25%の高速化が達成された．

Fast associated Legendre function transform by FMM

MASAYASU TAKAMI,† REIJI SUDA† and MASAAKI SUGIHARA†

Spherical harmonics transform consists of associated Legendre function transform and Fourier transform. The computational costs of the spherical harmonics transform is detarmined by that of the associated Legendre function transform. we have proposed a fast algorithm with complexity $O(M^2 \log M)$ for the cut-off frequency M which uses fast polynomial interpolation with FMM (Fast Multipole Method). In this paper we propose an algoritum to choose the sampling points so that the interpolation is numerically stable, and an improvement on FMM for the non-uniformly distributed points. We apply those methods to the associated Legendre function transform and achieve 25% speedup against the direct method for M =1365.

1. はじめに

球面調和関数は2次元球面上の関数近似やスペクトル法などに使われる大変重要なものである．しかしこれは1次元球面上でのFFTのような簡単な高速計算方法を持たず，切断周波数が M の時，計算量が $O(M^3)$ の直接法のアルゴリズムが一般に用いられてきた．これに対し我々は，FMM（Fast Multipole Method）を用いた $O(M^2 \log M)$ のアルゴリズム [4] を提案した．しかし数値的な安定性の問題のため，実装評価は行われていなかった．今回我々は数値的に安定な計算を実現する標本点選出アルゴリズムを見出した．さらにFMMにも改良を加え計算時間の高速化を達成した．

本稿の構成は次の通りである．第2節では我々が提案している高速Legendre陪関数変換のアルゴリズムの概要，第3節はそのアルゴリズムを数値的に安定に計算するための標本点を選ぶアルゴリズム，第4節では非一様に分布した点に対して改良したFMMのアルゴリズムについて説明する．第5節はまとめである．

† 名古屋大学工学研究科計算理工学専攻
Department of Computational Science and Engineering, Graduate School of Engineering, Nagoya University

2. FMMによるLegendre陪関数変換

2.1 Legendre陪関数変換

球面調和関数変換は球面調和関数を Y_n^m とすると

$$g(\lambda,\mu) = \sum_{m=0}^{M} \sum_{n=m}^{M} g_n^m Y_n^m(\lambda,\mu) \qquad (1)$$

と書くことができる．ここでLegandre陪関数 P_n^m を用いると(1)式は次のようにLegendre陪関数変換とFourier変換に分解できる．

$$s^m(\mu) = \sum_{n=m}^{M} g_n^m P_n^m(\mu) \qquad (2)$$

$$g(\lambda,\mu) = \sum_{m=0}^{M} s^m(\mu) e^{im\lambda} \qquad (3)$$

Fourier変換はFFTによって $O(M^2 \log M)$ で計算できるがLegendre陪関数変換には(2)式をそのまま計算すると $O(M^3)$ のアルゴリズムが用いられてきた．しかし我々はFMMによる多項式内挿を用いた $O(M^2 \log M)$ のアルゴリズムを提案した [4]．

2.2 多項式内挿

ある N 個の点 $\{x_i\}$（このような x_i を**標本点**と呼ぶことにする）における $N-1$ 次多項式の値 $f(x_i)$ が分かっているとき $\omega(x)=\prod_{i=1}^{N}\omega(x-x_i)$ と $\omega_i(x)=\omega(x)/(x-x_i)$ を用いると $\{y_j\}$ における $f(y_j)$（このような y_j を**内挿点**と呼ぶことにする）の値は

$$f(y_j)=\omega(y_j)\sum_{i=1}^{N}\frac{1}{y_j-x_i}\frac{f(x_i)}{\omega_i(x_i)} \tag{4}$$

と表すことができる．$\{x_i\}$，$\{y_j\}$ が定まっていれば，$\omega(y_j),\omega_i(x_i)$ をあらかじめ計算しておくことができて内挿は以下の 3 ステップで行われることになる．

$$\xi_i=\frac{f(x_i)}{\omega_i(x_i)} \qquad (i=1,\cdots,N) \tag{5}$$

$$\eta_j=\sum_{i=1}^{N}\frac{\xi_i}{y_j-x_i} \qquad (j=1,\cdots,M) \tag{6}$$

$$f(y_j)=\omega(y_j)\eta_j \qquad (j=1,\cdots,M) \tag{7}$$

この時 (5) 式と (7) 式の計算量は各々$O(N),O(M)$ である．(6) 式は FMM を使って $O(M+N)$ で計算できる [3]．

2.3 FMM のアルゴリズム

本節では FMM のアルゴリズムのアウトラインについて説明する．まずそのために必要なキーワード等を挙げる．

2.3.1 キーワードの説明

multipole expansion

ある区間の中心座標値を a として

$$L=\left[a-\frac{L}{2},a+\frac{L}{2}\right] \tag{8}$$

に含まれる標本点 x_i に対し内挿点 y_j が

$$\left|\frac{a-x_i}{a-y_j}\right|<\rho<1 \tag{9}$$

を満たしている時に $c_k=(x_i-a)^{k-1}$ を用いると

$$\frac{\xi_i}{y_j-x_i}=\sum_{l=1}^{\infty}\frac{c_k}{(y_j-a)^l}\xi_i \tag{10}$$

とすることができる．これを multipole expansion と呼び，a を multipole expansion のポールと呼ぶ．

multipole expansion のポールシフト

multipole expansion のポール a は

$$\left|\frac{a'-a}{a'-y_j}\right|<\rho<1 \tag{11}$$

を a' がみたしていれば

$$\sum_{l=1}^{\infty}\frac{c_l}{(y_j-a)^l}\xi_i=\sum_{l=1}^{\infty}\left[\frac{\xi_i}{(y_j-a')^l}\sum_{h=1}^{l}\left\{c_h(a-a')^{l-h}\right\}\binom{l-1}{l-h}\right]$$

とすることでポール a を a' に移動することができる．これを multipole expansion のポールシフトと呼ぶ．

local expansion

内挿点 y_j がある座標値 b に対して

$$\left|\frac{b-y_j}{b-a}\right|<\rho<1 \tag{12}$$

をみたす時，multipole expansion は

$$\sum_{l=1}^{\infty}\frac{c_l}{(y_j-a)^l}\xi_i=\sum_{l=0}^{\infty}\left\{\frac{\xi_i}{(y_j-x_i)^l}\cdot\sum_{h=1}^{\infty}\frac{c_h}{(b-a)^h}\binom{l+h-1}{l}\right\}(b-y_j)^l$$

とすることができる．これを local expansion，b を local expansion のポールと呼ぶ．

local expansion のポールシフト

local expansion のポール b は

$$\sum_{l=0}^{\infty}B_l(b-y_j)^l=\sum_{l=0}^{\infty}\left\{\sum_{h=l}^{\infty}B_h(b-b')^{h-l}\binom{h}{l}\right\}(b'-y_j)^l$$

と b' に移動することができる．これを local expansion のポールシフトと呼ぶ．

展開項数

multipole expansion，local expansion，各々のポールシフトの 4 つの計算は一種のテーラー展開であり，その展開は十分な精度が得られる K で打ち切り $\rho=1/3$ とすると，$K=24$ 程度で倍精度計算で十分な誤差が得られる．この K を展開項数と呼ぶ．

領域分割の設定と level

標本点，内挿点の分布している範囲を 4 等分に分割する，この状態を level1 とする．各々の領域をさらに分割する（level2）さらに各々の領域を分割して level3,4... を設定する．つまり level$l$ は同じ大きさの 2^{l+1} 個の領域が存在している状態を示す．

2.3.2 アルゴリズムと計算量

本節では FMM のアルゴリズムを順を追って説明し，計算量も併せて記す．

まず最も深い level（以下**最適 level** と呼ぶ）を

level$\mathcal{L}$ とすると，以下のような step で行われる．

step 1 計算量 NK
level$\mathcal{L}$ のみにおける全領域の multipole expansion を計算する．

step 2 計算量 $4(2^{\mathcal{L}}-4)K^2$
level2 から level$\mathcal{L}-1$ の全領域で multipole expansion のポールシフトを行う．

step 3 計算量 $(4\cdot 2^{\mathcal{L}}+10\mathcal{L}-12)K^2$
(a) $l=1$ にする．
(b) level$l$ の全領域で local expansion を行う．
(c) level$l$ の全領域で local expansion のポールシフトを行う．
(d) $l=l+1$ にして (b) に戻る．$l=\mathcal{L}-1$ になったらやめる．

step 4 計算量 MK
level$\mathcal{L}$ の全領域で local expansion を評価する．

step 5 計算量 $MN(3\cdot 2^{\mathcal{L}}+4)/2^{2\mathcal{L}+2}$
最適 level でも local expansion が行えなかった部分の直接計算を行う．

これらの手順を行うときに重要になるのは最適 level の決定である．つまり領域分割を繰り返して level を下げるときに 1 領域の標本点が少なくなり，multipole expansion 等の近似計算をおこなうより直接計算したほうが計算量が少なくなるポイントが存在するのである．2) では最適 level は標本点，内挿点が同数かつ一様分布であれば $\mathcal{L}=\log_2 N/4K$ で決定できるとしている．これは最適 level における 1 領域あたりの標本点数が $2K$ 程度にの状態であり，この時の計算量は $O(M+N)$ となる．この計算量は点の分布が非一様でも同じである [3]．

2.4 多項式内挿を用いた高速 Legedre 陪関数変換

Legendre 陪関数は $n-m$ 次多項式 $q_n^m(x)$ を用いて

$$P_n^m(x)=q_n^m(x)P_m^m(x) \tag{13}$$

と表すことができるので (2) 式は

$$s^m(\mu)=P_m^m(\mu)\sum_{n=m}^{M} g_n^m q_n^m(\mu) \tag{14}$$

となる．この時 $s^m(\mu)/P_m^m(\mu)$ は $(M-m)$ 次多項式であるから多項式内挿が使える [1]．(14) 式は (4) 式から $N=M-m+1$ として

$$s^m(\mu_j)=P_m^m(\mu_j)\omega(\mu_j)\sum_{i=1}^{N}\frac{1}{\mu_j-\mu_i}\frac{s^m(\mu_i)}{P_m^m(\mu_i)\omega_i(\mu_i)} \tag{15}$$

となる．

今，評価したい M 個の点（これを**評価点**と呼ぶ）$\{\mu_t\}_{t=1}^{M}$ の集合を $\mathcal{M}$ とする．この時以下の要領で多項式内挿を使って全ての評価点を計算することができる．

step 1 N 個の点 $\{\mu_i\}_{i=1}^{N}\in\mathcal{M}$ を評価点から選び (2) 式をそのまま計算して $s^m(\mu_i)$ の値を得る．

step 2 得られた $s^m(\mu_i)$ から多項式内挿を用いて，残った点 $\{\mu_j\}_{j=1}^{M-N}$ $(\mu_i\neq\mu_j)$ における $s^m(\mu_j)$ を計算し全ての評価点の $s^m(\mu_t)$ を得る．

ここで，step 1 の計算量は $O(N^2)$，step 2 の計算量は FMM を用いれば $O(N+M)$ となる．直接 (2) 式から全ての評価点での $s^m(\mu)$ を求めると $O(NM)$ の計算量が必要であるから，N が M より十分小さいとこの方法は直接法より速くなる．

2.5 問 題 点

このアルゴリズムには以下のような問題点がある．

- (15) 式は無作為に標本点（内挿点）を取った場合，数値的に不安定となる．数値的安定性は主に標本点の取り方に依存しており，安定となるように標本点を選ぶ必要がある．
- 評価点の分布が一様でないため，FMM が最も効率よく機能するという状況にならない．

そこで我々は，数値的に安定となるように標本点を選出する方法と，非一様分布な評価点に対する FMM アルゴリズムの改良を提案する．

3. 標本点の選出

この節では，論文 4) で未解決であった標本点（内挿点）の選出のアルゴリズムについて論じる．さらに高速 Legendre 陪関数変換の実装評価を行う．

3.1 標本点の選びかた

内挿計算の数値的安定性は標本点と内挿点の集合（主に標本点集合）に依存している．

$$L_{ji}=\frac{P_m^m(\mu_j)\omega(\mu_j)}{P_m^m(\mu_i)\omega_i(\mu_i)} \tag{16}$$

とした時に $\max|L_{ji}|$ が大きくなりすぎると内挿が不

安定となる．そこで，これができるだけ小さくなるように以下のような簡単なアルゴリズムを用いて，2.4節で説明したように M 個の点からなる評価点集合 $\mathcal{M}$ から $N = M - m + 1$ 個の標本点を選ぶ．

step 1 $h_0(\mu) = P_m^m(\mu)$ $(\mathcal{M} \ni \mu)$ とする．

step 2 $i = 1$ にする．

step 3 $|h_{i-1}(\mu_i)| = \max_j |h_{i-1}(\mu_j)|$ となるように μ の中から μ_i を選び標本点集合に加える

step 4 $h_i(\mu_j) = (\mu_j - \mu_i)h_{i-1}(\mu_j)$ $(j = 1, \ldots, M : j \neq i)$ とする．

step 5 $i = i + 1$ にして step3 に戻る．i が M になったら終了する．

このアルゴリズムで $h_i(\mu)$ は $P_m^m(\mu)\omega(\mu)$ の一部を計算している．この値が大きいものを標本点として選ぶことにより，この式が L_{ji} の分母に現れるようになり，L_{ji} が大きくなることを防いでいる．

3.2 実装評価

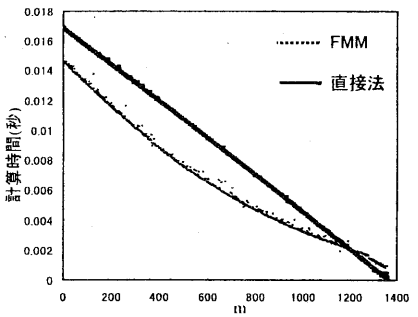

図 1 直接法と FMM アルゴリズムの計算時間の比較 ($M = 1365$)

上記のアルゴリズムを用いて実際に高速 Legendre 陪関数変換を行った．切断周波数 $M = 1365$，評価点は Gauss-Legendre 積分公式の節点である．

プログラム言語は C でコンパイラは DEC cc，最適化オプションは-O7．CPU は COMPAQ alpha EV6 500 MHz，OS は DEC UNIX V4.0E で数値実験を行った．展開項数は 24 とした，これは倍精度を得るのに十分な値である．

図 1 は FMM を用いたプログラムと直接法のみの計算時間を比較したものである．横軸は Legendre 陪関数の位数 m，縦軸は計算時間（秒）を示している．

FMM を用いたアルゴリズムはトータルで 17.8 秒だったのに対し，直接法のみでは 23.1 秒かかった．前者の方が約 23%速くなっている．m ごとに各々の計算時間を比較してみると，だいたい $m = M/3$ 程度で最も効率良く機能していて，その後徐々にその効力は落ちて $m = 1200$ あたりから直接法の方が速くなっている．この時の相対誤差はフロベニウスノルムで 10^{-13} 以下となり数値的安定性が実現されていることが分かる．

3.3 標本点の分布傾向

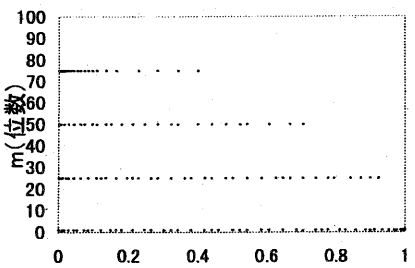

図 2 標本点の分布

先のアルゴリズムで選んだ標本点の分布を図 2 に示す（切断周波数は 100）．横軸は標本点の座標値，縦軸は Legendre 陪関数の位数 m を表している．m が切断周波数に近づくにつれて標本点が原点の付近に集まっていくのが分かる．これは $P_m^m(\mu) \propto (1 - \mu^2)^{m/2}$ であるためで，この傾向は切断周波数によらない．従って，一様分布な点に対して最も効力を発揮する従来の FMM のアルゴリズムを改良する必要がある．

4. FMM の改良

4.1 従来の FMM の欠点

3.3 節で述べたように位数 m が切断周波数 M に近づくと標本点は原点に向かって分布が偏るという傾向がある．そうなった場合，従来の FMM のアルゴリズムでは以下のような欠点がある．

- 標本点が全く存在しない領域であっても multipole expansion やそのポールシフトの計算を行っている．
- 新たに local expansion ができる標本点領域が現れないのに local expansion のポールシフトに関

する計算を行っている．

これらはいずれも無駄な計算である．そこで我々はこの問題を解決するため，従来のFMMとは異なる領域分割を提案する．

4.2 FMMのアルゴリズムの改良

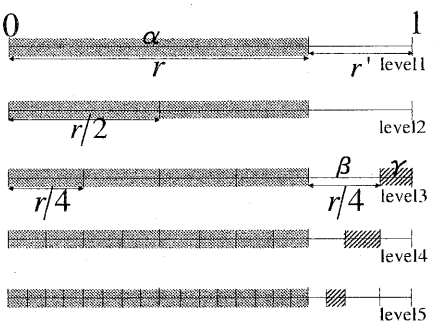

図3 改良FMMの領域分割設定

先に述べた問題を解決するために以下のような標本点，内挿点の領域分割を考える．

領域αの設定と分割方法

まず，全領域$[0,1]$を標本点のなかで最も座標値が大きい点$r=\max_i\{\mu_i\}$で分割する．こうしてできた領域$[0,r]$をαと呼ぶ（図3）．αは従来のFMMのアルゴリズムの通りlevelが1層深くなるごとに等分割していく．

領域β，γの設定と分割方法

$r'=1-r$とする．levelを深くして領域αを分割していった時の1領域分の大きさはlevel$l$において$r/2^{l-1}$となっている（図3）．あるlevelζで$r'>r/2^{\zeta-1}$となったら，領域$[r',1]$を$r+r/2^{\zeta-1}$で分割し，領域$[r,r+r/2^{\zeta-1}]$をβ，領域$[r+r/2^{\zeta-1},1]$をγと名付ける．図3の例ではlevel3で$r'>r/4$となったので，そこで領域$[r',1]$を分割し，βが領域$[r,r+r/4]$，γが$[r+r/4,1]$となる．

βは標本点が分布している領域に隣接している領域のみ分割する．γは分割しない．図3の斜線部が分割しない領域である．

multipole expansion，local expansionについては以下のようにする．

αやその分割によって派生する領域）

従来のFMMのアルゴリズムで計算する．

βやその分割によって派生する領域）

標本点を持たないのでmultipole expansionに関する計算は行わない．local expansionは全ての領域で計算する．local expansionのポールシフトは，分割を既にやめている領域は計算しない．つまりαとβが同じ大きさなら，標本点が存在する範囲（図3のアミ掛けした部分）に隣接する領域のみ計算する．

γの領域）

local expansionのみ計算し，multipole expansion，そのポールシフト，local expansionのポールシフトについては計算しない．

従来のアルゴリズムは，改良されたアルゴリズムのβ，γにあたる部分に標本点が入っていなくても，先にあげた無駄な計算を行っている．従って，改良されたアルゴリズムは標本点の分布の偏りが大きくなって領域β，γが大きくなる程，有効であると考えられる．また切断周波数が大きくなることによって標本点，内挿点が増えると，最適levelが深くなるため従来のFMMは無駄なポールシフトの計算が増えることになり，この場合でも改良されたアルゴリズムは効力を発揮すると考えられる．

4.3 計 算 量

ここではFMMアルゴリズムを改良したことによって計算量がどれだけ減ったかを論じる．

まず$r=1/2$でN個の標本点と$2N$個の内挿点が一様分布していた場合，展開項数をKとしたとき，従来のFMMでは計算量が

$$19NK-6K^2\log_2\frac{N}{2K}-32K^2 \tag{17}$$

であるのに対し改良されたFMMでは

$$13NK+2K^2\log_2\frac{N}{K}-14K^2 \tag{18}$$

となり$N\to\infty$では，計算量の比が13/19になっている．

さらに$r=1/4$でN個の標本点と$4N$個の内挿点が一様分布していた場合，従来のFMMでは

$$31NK-6K^2\log_2\frac{N}{K}-32K^2 \tag{19}$$

であるのに対し改良されたFMMでは

$$15NK+2K^2\log_2\frac{N}{K}-13K^2 \tag{20}$$

となり，計算量の比は約1/2に減っておりその効果はさらに顕著である．

4.4 実装評価

4.4.1 数値実験 1

表 1 改良前と改良後の計算時間（秒）の比較 1（M=1365）

m	668	945	1170
改良前	2.91×10^{-3}	2.47×10^{-3}	1.99×10^{-3}
改良後	2.66×10^{-3}	2.13×10^{-3}	1.48×10^{-3}

表 2 改良前と改良後の計算時間（秒）の比較 2（M=2730）

m	1073	1913	2353
改良前	6.49×10^{-3}	6.25×10^{-3}	4.73×10^{-3}
改良後	6.28×10^{-3}	4.76×10^{-3}	3.26×10^{-3}

表 1 と表 2 は改良前と改良後の FMM の計算時間の比較である．切断周波数はそれぞれ 1365, 2730 で，m は各々$r \approx 3/4, 1/2, 1/4$ になっている値である．CPU 等の計算機，コンパイラに関する環境設定は 3.2 節の実装評価と同じとした．

改良されたアルゴリズムは，標本点の分布範囲が偏るにつれて効力を発揮していることが分かる．また切断周波数が大きくなって評価点の数が増えると，その効果も大きくなっている．

4.4.2 数値実験 2

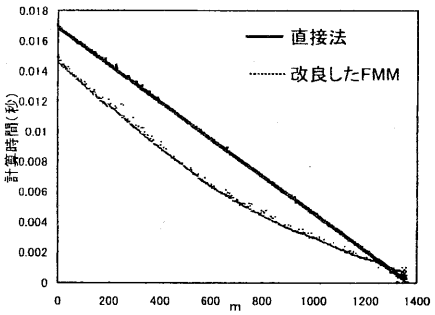

図 4 直接法と改良後の FMM の計算時間の比較 ($M = 1365$)

図 4 は Legendre 陪関数変換に対する，改良した FMM と直接法の計算時間の比較である．なお CPU，コンパイラ等の環境設定は 3.2 節と同じとした．改良されたアルゴリズムでは，トータルの計算時間は 17.3 秒で直接法に比べ 25%計算時間が短縮された．改良前と比べ大きな高速化とまでは至らなかった．しかし標本点が偏る m が 500 から 1365 に注目すると，改良前は 6.79 秒であったが改良後は 6.24 秒となり 9.2%の時間短縮が見られた．

5. ま と め

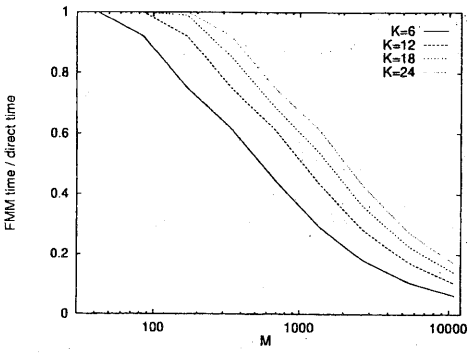

図 5 FMM アルゴリズムと直接法の計算時間比率の見積り

本稿は，既に提案されている Legendre 陪関数変換の高速計算法の実装評価と，その改良に関する報告を行った．その結果 $M = 1365$ で，直接法に比べ 25%高速に計算できるという結果が得られた．しかし，FMM アルゴリズムはまだまだ改善の余地がある．図 5 は点の分布が一様で，FMM が最もうまく働いた場合の高速 Legendre 陪関数変換法と，直接法との計算時間比の見積りである．これによると $M = 1365$ 程度なら展開項数 $K = 24$ で 30%以上の計算時間の短縮が可能と見られている．また chebyshev 近似を用いれば $K = 6$ で単精度の誤差範囲となるが，この K でならば $M \approx 100$ で直接法より高速である．

今後は非一様に散らばる標本点，内挿点に対する領域分割の改良，FMM の基礎となっている近似計算方法の改良により，FMM のアルゴリズムをさらに改善していくことが課題である

参 考 文 献

1) A. Dutt, M. Gu, and V. Rokhlin: *Fast algorithms for polynomial interpolation, integration, and differentiation*, SIAM J. Num. Anal., Vol. 33, No. 5, pp. 1689-1711, (1996).
2) L. F. Greengard: *The Rapid Evaluation of Potential Fields in Particle Systems*, The MIT Press (1988)
3) J. H. Reif and S. R. Tate: *N-body Simulation I: Fast Algorithms for Potential Field Evaluation and Trummer's Problem*, Tech. Rpt. N-96-002, Univ. of North Texas, Dept. of Computer Science (1996).
4) 須田礼仁: 高速球面調和関数変換法，情報処理学会研究報告，98-HPC-73, pp. 37-42, (1998).