ハイパフォーマンス 73−7
コンピューティング
（１９９８．１０．９）

高速球面調和関数変換法

須田礼仁
名古屋大学 工学研究科 計算理工学専攻
reiji@na.cse.nagoya-u.ac.jp

概要

球面調和関数変換は Legendre 陪関数変換と Fourier 変換に分割される。後者は FFT という高速算法を持つが、Legendre 陪関数変換には切断周波数 M に対して通常 $O(M^3)$ の直接法が用いられてきた。本稿では計算量が $O(M^2 \log M)$ となる新しい高速アルゴリズムを提案する。このアルゴリズムは divide-and-conquer によっており、FMM を用いた高速多項式内挿アルゴリズムにより高速計算を実現している。数値的安定性・精度・小規模な問題に対する速度も十分であるこのアルゴリズムは、球面調和関数を用いた数値計算の新しい局面を切り拓くものである。

A Fast Spherical Harmonics Transform Algorithm

Reiji Suda
Dept. of Computational Science and Engineering, Faculty of Engineering, Nagoya University

Abstract

Spherical harmonic transform consists of associated Legendre transform and Fourier transform. While FFT enables fast computation of Fourier transform, the direct method of complexity $O(M^3)$ for cut-of frequency M is used for associated Legendre transform. This paper proposes a new fast associated Legendre transform algorithm with complexity $O(M^2 \log M)$. The algorithm uses the divide-and-conquer approach, and its high speed comes from fast polynomial interpolation with FMM (Fast Multipole Method). The algorithm is also good for stability, accuracy, and computational costs for small problems, and opens new possibility of the spherical harmonics in numerical computation.

1. はじめに

球面調和関数変換は 2 次元球面上での関数近似に欠かせない道具である。1 次元球面（単位円周）上ではこれは Fourier 変換となり、切断周波数を M とすると FFT により計算量 $O(M \log M)$ で変換ができることは良く知られている。しかし球面調和関数変換は FFT のような高速変換を持たず、変数分離による $O(M^3)$ のアルゴリズムが一般に用いられている。これに対し本稿では FMM (Fast Multipole Method) [2] を用いた $O(M^2 \log M)$ の高速球面調和関数変換法を提案する。

1.1. Legendre 陪関数変換

球面調和関数 Y_n^m は Legendre 陪関数 P_n^m と三角関数を用いて

$$Y_n^m(\lambda, \mu) = P_n^m(\mu) e^{im\lambda}$$

と書くことができる。従って、切断周波数 M の球面調和関数変換

$$g(\lambda_i, \mu_j) = \sum_{m=0}^{M} \sum_{n=m}^{M} g_n^m Y_n^m(\lambda_i, \mu_j)$$

は次のように Legendre 陪関数変換と Fourier 変換に分解できる。

$$s^m(\mu_j) = \sum_{n=m}^{M} g_n^m P_n^m(\mu_j)$$

$$g(\lambda_i, \mu_j) = \sum_{m=0}^{M} s^m(\mu_j) e^{im\lambda_i}$$

Fourier 変換は FFT を用いれば $O(M^2 \log M)$ で計算できるが、Legendre 陪関数変換は上記の和を単純にとる $O(M^3)$ のアルゴリズムが一般に用いられている（以下、これを**直接法**と呼ぶ）。従って、球面調和関数変換全体の計算量は Legendre 陪関数変換が決定しており、通常は $O(M^3)$ である。本稿ではこの Legendre 陪関数変換の高速計算について考察する。

1.2. 関連研究

これまでにも Legendre 陪関数変換の高速計算についてさまざまな研究がなされてきた。Orszag[1] は Legendre 陪関数を含む Sturm-Liouville 固有関数変換について、WKB 近似に基づく計算量が $O(M^{3/2})$ および $O(M \log^2 M / \log\log M)$ の 2 つ

のアルゴリズムを提案している。しかし WKB 近似を用いて十分な精度が確保できるのかどうか、計算量の m 依存性と球面調和関数変換全体の計算量がどのようになるかなどの問題を残している。Orszag は Legndre 多項式 $(m=0)$ に対する $O(M^{3/2})$ のアルゴリズムを CRAY-1 に実装して、$M \approx 128$ 程度で直接法よりも速くなったと報告している。また、$O(M\log^2 M/\log\log M)$ のアルゴリズムは実装しておらず、直接法よりも高速になるには少なくとも $M \approx 250$ 程度が必要という見通しを述べている。

Healy ら [7] は計算量が $O(M^2\log^2 M)$ となる Legendre 陪関数変換のアルゴリズムを提案している。彼らのアルゴリズムは Legendre 陪関数が満たす漸化式から導かれる関係式を利用して divide-and-conquer を行ない、各ステップでは FFT と smoothing を用いて高速化を行なっている。私自身 [10] も彼らと似たようなアプローチで、多項式の高速計算法を用いた $O(M^2\log^2 M)$ のアルゴリズムがありうることについて報告した。これらの方法は本質的に Legendre 陪関数展開を多項式展開に変換させているのだが、不幸にも数値的に極めて不安定で、Legendre 多項式だけしか安定に動かない。速度としては Healy らの高速 Legendre 多項式変換では、$M \approx 256$ 程度で直接法よりも高速になるという。

Alpert と Rokhlin [3] は Legendre 多項式変換について $O(M\log M)$ のアルゴリズムを提案している。これは Greengard と Rokhlin が提案した FMM [2] の一般化の研究の一環らしく、本質的には Legendre 多項式展開から Chebyshev 多項式展開への変換を FMM 風の方法で $O(M)$ で行ない、Chebyshev 多項式変換を FFT で $O(M\log M)$ で行なうものである。彼らの実装では $M=64$ ですでに直接法よりも明らかに高速である。Beylkin ら [5] は wavelet を用いてほぼ同じことがすっきりと実現できることを示しているが、この場合はサイズが 2 冪でないとうまく行かないようである。いずれにせよ、これらの方法が Legendre 陪関数変換に拡張できるかどうかについては精度や速度などに関して問題がある。

Mohlenkamp [9] は wavelet の一種である local cosine 変換に基づいた Legendre 陪関数変換のアルゴリズムを 3 つ提案している。計算量はそれぞれ $O(M^{5/2}\log M)$, $O(M^2\log^2 M)$, $O(M^2\log M)$ である。彼は $O(M^{5/2}\log M)$ のアルゴリズムの実装の所用時間を報告しており、$M=256$ で直接法よりも高速であると主張している（彼はデータの読み込みの時間を含めている。読み込みの時間を差っぴくと $M=2048$ にならないと直接法よりも速くならない)。$O(M^2\log^2 M)$ のアルゴリズムについては実装はしたが思ったほど計算量が減らなかったと述べている。$O(M^2\log M)$ のアルゴリズムについてはアウトラインだけを述べるに留まっている。しかし、私の知る限りは、この論文は適当な精度で実際に直接法より高速な Legendre 陪関数変換を実現した唯一の論文である。

Legendre 陪関数変換ではないが、関連する研究がいくつかある。球面調和関数による内挿とは、与えられた μ_j $(1 \le j \le M-m+1)$ での値 $\sum_{n=m}^{M} g_n^m P_n^m(\mu_j)$ がわかっている時、異なる点集合 $\tilde{\mu}_k$ $(1 \le k \le N)$ 上での値 $\sum_{n=m}^{M} g_n^m P_n^m(\tilde{\mu}_k)$ を求める問題である。Boyd [4] はこの問題が Dutt ら [6] による FMM による線形時間の多項式内挿を用いれば $O(M^2+N^2)$ の計算量で計算できることを指摘した。この結果は Legendre 陪関数変換のアルゴリズムに非常に重要な意味を持つ。上記の高速変換アルゴリズムはいずれも特別な μ_j での評価しかできない。しかし内挿が高速にできることから、この μ_j についての制約は実用上何らの問題も生じないという結果が得られるのである。他にも Jakob-Chien と Alpert [8] による spherical filter の高速アルゴリズムの論文がある。

本稿で提案するアルゴリズムは divide-and-conquer に基づき、FMM による高速多項式内挿計算を応用した $O(M^2\log M)$ の計算量を持つアルゴリズムである。この計算量は FFT のものと同じで、これまでに提案されてきた最も高速なアルゴリズムと同じである。安定性と精度も実用上十分であり、また、小さい問題に対しても直接法よりもあまり遅くならない点も重要な長所である。今回は、まだ残っているいくつかの問題のために実装評価は間に合わなかった。次節ではこのアルゴリズムの概要と、安定性や精度などについての予備的評価について報告する。

2. 新しいアルゴリズムの概要

2.1. アルゴリズムのアイデア

Legendre 陪関数変換は Legendre 陪関数の線形結合

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

を与えられた $\mathcal{M} = \{\mu_j\}_{j\in\mathcal{J}}$ において評価する問題である。これは「逆変換」であるが、逆に与えられた $\{\mu_j\}_{j\in\mathcal{J}}$ で値が与えられた関数から展開係

数 g_n^m を求める「変換」も、Gauss 積分法を用いてほとんど同じ計算で求めることができる。これらの計算は m ごとに独立であり、提案するアルゴリズムでもこれらの計算を独立に行なうので、今後はある固定した m を考え、必要のない限り添字 m を省略することにする。

今回提案する高速アルゴリズムは、上記の総和の範囲 $m \le n \le M$ を 2 分割し、それぞれの部分和を再帰的に 2 分割して計算する divide-and-conquer に基づいている。これは Healy らのもの [7] や私自身が以前に報告した高速アルゴリズム [10] と同じ方法である。これらの方法では途中で多項式を経由することになるのであるが、このような迂回が数値的な不安定性を引き起こし、実用上必要な精度が得られなくなってしまうことがわかっている。この問題を解決するために今回のアルゴリズムでは関数 $f(\mu)$ の表現として、決められた J 個の評価点集合 $\mathcal{M} = \{\mu_j\}_{j=1}^{J}$ における関数値 $f(\mu_j)$ をならべたベクトル（以下このベクトルを $f(\mathcal{M})$ と表わす。すなわち

$$f(\mathcal{M}) = (\, f(\mu_1) \quad f(\mu_2) \quad \cdots \quad f(\mu_J)\,)^T$$

）を用いることにした。この表現では任意の関数の J 項の線形結合が表わせるので、多項式を経由することなく Legendre 陪関数の部分的な線形結合が直接扱えることになった。数値的な安定性は評価点集合 $\mathcal{M}$ の選択に依存するが、LU 分解の枢軸選択を応用して数値的に安定な評価点集合を決定することができる。

$n \in \mathcal{N}$ の範囲の部分線形結合を

$$s_{\mathcal{N}}(\mu) = \sum_{n \in \mathcal{N}} g_n P_n(\mu)$$

で表わし、これの $\mu \in \mathcal{M}$ における値をならべたベクトルを $s_{\mathcal{N}}(\mathcal{M})$ と表わす。このとき先に述べた通り評価点の数は $|\mathcal{M}| = |\mathcal{N}|$ で十分である。Divide-and-conquer により、右辺の計算は $s_{\mathcal{N}_0}(\mathcal{M}_0)$ と $s_{\mathcal{N}_1}(\mathcal{M}_1)$ に分割されるが、$\mathcal{M}_0$ や $\mathcal{M}_1$ は $\mathcal{M}$ よりも少ない点しか含まないので、conquer の際には内挿をして必要な評価点上での値を求めなければならない。計算量を軽減するために $\mathcal{M}_0 \subset \mathcal{M}$ にとるのは当然であるが、このとき内挿計算に単純なアルゴリズムを用いると、内挿計算は $|\mathcal{M} - \mathcal{M}_0| \times |\mathcal{M}_0|$ の行列とベクトルの積となる。その結果、Legendre 陪関数変換の flop (浮動小数点数演算) 数は直接法と一致する。

しかし、内挿計算に FMM に基づく高速な内挿計算 [6] を用いると内挿の計算量は $O(|\mathcal{M}|)$ となる。部分和の和の計算も $O(|\mathcal{M}|)$ の計算量であるから、conquer に要する計算量はあわせて $O(|\mathcal{M}|)$ となる。Divide-and-conquer の各レベルでの $|\mathcal{M}|$ の総和は $O(M)$ であるから、$O(\log M)$ 段の divide-and-conquer の計算量は $O(M \log M)$ となる。これが $0 \le m \le M$ について計算されるので、合計すると $O(M^2 \log M)$ の計算量となる。

以下では、このアルゴリズムのさらに詳しい説明と安定性や精度に関する予備評価の結果や今後の課題などについて、3 段階のアルゴリズムに分けて説明を行なう。

2.2. アルゴリズム 1: FMM 内挿法

この節では、$\mathcal{M}$ は最終的に Legendre 陪関数変換を評価する評価点集合、$\mathcal{N}$ は全範囲 $\{n : m \le n \le M\}$ とする。また、$\mu \in \mathcal{M}$ と $n \in \mathcal{N}$ に対する $P_n(\mu)$ をならべた行列を $P_{\mathcal{N}}(\mathcal{M})$ で、g_n をならべた列ベクトルを $g_{\mathcal{N}}$ で表わす。従って、$s_{\mathcal{N}}(\mathcal{M}) = P_{\mathcal{N}}(\mathcal{M}) g_{\mathcal{N}}$ である。

行列 $P_{\mathcal{N}}(\mathcal{M}')$ が正則となるように $\mathcal{M}' \subset \mathcal{M}$ を適当に選ぶことにする。このとき $\mathcal{M}'$ 上で Legendre 陪関数変換 $s_{\mathcal{N}}(\mathcal{M}')$ が評価されていれば残りの点集合 $\mathcal{M} - \mathcal{M}'$ における値は

$$\begin{aligned} s_{\mathcal{N}}(\mathcal{M} - \mathcal{M}') &= P_{\mathcal{N}}(\mathcal{M} - \mathcal{M}') g_{\mathcal{N}} \\ &= P_{\mathcal{N}}(\mathcal{M} - \mathcal{M}') P_{\mathcal{N}}(\mathcal{M}')^{-1} s_{\mathcal{N}}(\mathcal{M}') \end{aligned}$$

として計算することができる。これは評価点集合 $\mathcal{M}'$ から $\mathcal{M} - \mathcal{M}'$ 上への「内挿」である。以下、$\mathcal{M}$ の要素を**評価点**、$\mathcal{M}'$ の要素を**参照点**、$\mathcal{M} - \mathcal{M}'$ の要素を**内挿点**と呼ぶことにする。内挿の計算は上の式から明らかなように、行列とベクトルの積の形をしている。この行列を**内挿行列**と呼び、

$$Q_{\mathcal{N}}[\mathcal{M}'/\mathcal{M}] = P_{\mathcal{N}}(\mathcal{M} - \mathcal{M}') P_{\mathcal{N}}(\mathcal{M}')^{-1}$$

と表わす。この行列・ベクトル積を単純なアルゴリズムで計算すると $O(|\mathcal{M}'||\mathcal{M} - \mathcal{M}'|)$ の計算量となる。

しかし、多項式の内挿は FMM を用いることにより $O(|\mathcal{M}'| + |\mathcal{M} - \mathcal{M}'|) = O(|\mathcal{M}|)$ の計算量で計算することができる [6] ことが知られている。今回の場合は Legendre 陪関数の線形結合であるが、Legendre 陪関数 P_n^m は P_m^m と超球多項式の積である。従って、$s_{\mathcal{N}}(\mu)$ を $s_{\mathcal{N}}(\mu)/P_m^m(\mu)$ のように「スケーリング」してやれば、問題は超球多項式の内挿となる。従って、Legendre 陪関数展開の内挿は線形オーダーで計算できる。

この内挿を用いることにより、漸近的なオーダーは変わらないものの、若干の計算量が節約できる。Legendre 陪関数の対称性から、$n-m$ が偶数の場合と奇数の場合を別々に扱うのが有利である。このとき、通常は評価点数 J は $J=3M/4$ 程度となる。この場合、直接法では $M^2J/2$ の積和計算が必要であるが、FMM 内挿を用いれば $M^3/6+O(M^2+MJ)$ の計算量になる。従って、FMM 内挿を用いることにより計算量の M^3 の係数は直接法の 4/9 になる。注意深く実装すれば、小さい問題でも flop 数が直接法を上回ることはない。

以下ではこのアルゴリズムを「アルゴリズム 1」と呼ぶ。このアルゴリズム 1 には以下のような検討課題が残っている。

参照点集合 $\mathcal{M}'$ は、$P_{\mathcal{N}}(\mathcal{M}')^{-1}$ ができるだけ安定に計算できるように選ぶのが望ましい。これは、内挿行列の計算に行枢軸選択付きの LU 分解または LQ 分解を用いることで実現できる。このうち LU 分解の方を試験的に実装してみたが、実用上問題になるような数値的な不安定性は見つかっていない。内挿計算に伴う誤差のため、丸め誤差よりも数桁大きな誤差が混入するが、倍精度で大体 10^{-12} くらいに収まっている。参照点集合 $\mathcal{M}'$ の選択方法については今後理論と実験の両面から詰めていく必要があると考えている。

FMM 内挿の性能を決定する重要な問題が 2 つある。いわゆる「多重極」の数と行列の再帰的な分割の仕方である。おそらくあまり異常なものを選ばなければ漸近的なオーダーには影響がないと思われるが、計算量の定数係数や小規模な問題に対する速度はかなりの影響を受けるはずである。オリジナルの論文 [6] で多項式の内挿が線形オーダーの計算量となることの証明には、参照点と評価点の数が同じで空間的に一様に分布していることが仮定されているが、これは線形オーダーの計算量を得るためには必要ないと予想している。この点も含め、FMM については今後さらに精密に検討する必要があると考えている。

2.3. アルゴリズム 2：分割統治法

アルゴリズム 1 では、まず $\mathcal{M}'$ 上での Legendre 陪関数変換の評価を行ない、それを $\mathcal{M}$ に内挿した。計算量は前半部分が $O(M^3)$ で、後半部分が $O(M^2+MJ)$ である。次に、この前半部分を更に高速化する「アルゴリズム 2」を考える。

アルゴリズム 2 はいわゆる divide-and-conquer によっている。まず、n の範囲である $\mathcal{N}$ を $\mathcal{N}_0$ と $\mathcal{N}_1$ のように分解する。基本的には $\mathcal{N}$ の要素の小さい方から半分を集めて $\mathcal{N}_0$ とし、残りを $\mathcal{N}_1$ とするのであるが、それ以外の選び方も可能である。$s_{\mathcal{N}}(\mathcal{M})$ の計算は

$$
\begin{aligned}
s_{\mathcal{N}_0}(\mathcal{M}') &= P_{\mathcal{N}_0}(\mathcal{M}')g_{\mathcal{N}_0} \\
s_{\mathcal{N}_1}(\mathcal{M}') &= P_{\mathcal{N}_1}(\mathcal{M}')g_{\mathcal{N}_1} \\
s_{\mathcal{N}}(\mathcal{M}') &= s_{\mathcal{N}_0}(\mathcal{M}') + s_{\mathcal{N}_1}(\mathcal{M}') \\
s_{\mathcal{N}}(\mathcal{M}) &= Q_{\mathcal{N}}[\mathcal{M}'/\mathcal{M}]s_{\mathcal{N}}(\mathcal{M}')
\end{aligned}
$$

という 4 つのステップに分解される。

第 1 ステップは再帰計算となる。すなわち、$\mathcal{N}_0$ をさらに分割して $\mathcal{N}_{00}$ と $\mathcal{N}_{01}$ とし、$|\mathcal{M}_0|=|\mathcal{N}_0|$ となる評価点集合 $\mathcal{M}_0$ を決めたうえで、

$$
\begin{aligned}
s_{\mathcal{N}_{00}}(\mathcal{M}_0) &= P_{\mathcal{N}_{00}}(\mathcal{M}_0)g_{\mathcal{N}_{00}} \\
s_{\mathcal{N}_{01}}(\mathcal{M}_0) &= P_{\mathcal{N}_{01}}(\mathcal{M}_1)g_{\mathcal{N}_{01}} \\
s_{\mathcal{N}_0}(\mathcal{M}_0) &= s_{\mathcal{N}_{00}}(\mathcal{M}_0) + s_{\mathcal{N}_{01}}(\mathcal{M}_0) \\
s_{\mathcal{N}_0}(\mathcal{M}') &= Q_{\mathcal{N}_0}[\mathcal{M}_0/\mathcal{M}']s_{\mathcal{N}_0}(\mathcal{M}_0)
\end{aligned}
$$

のように計算を行なう。第 2 ステップも同様である。第 3 ステップは単にベクトルの和をとっているだけであるから、$|\mathcal{M}'|$ 回の加算で計算できる。第 4 ステップは前節で論じた通りの内挿計算である。

しかし、再帰計算での内挿は FMM を用いて線形時間で計算できるとは限らない。部分的な線形結合の場合には多項式の内挿に帰着できないからである。いわゆる「多重極子展開」も数値的に行なわなければならないが、これについては目新しい問題ではない [6]。問題は計算量であって、現在のところこの内挿計算にどれだけの計算量が必要であるかわかっていない。予備的に数値実験を行なったところ、部分線形結合の N 点から M 点への内挿の計算量が $O(N+M)$ から $O((N+M)^{3/2})$ 程度と推測される結果が得られている。前者の場合には Legendre 陪関数変換の計算量は $O(M^2\log M)$ に、後者の場合には $O(M^{5/2})$ となる。内挿計算が $O(NM)$ であれば定数倍の高速化しか得られない。なお、この方法も M が小さくて FMM 内挿が効かない場合でも直接法よりも flop 数が多くなることはない。

上記の通り、アルゴリズム 2 では FMM の計算量が最大の問題である。特に、FMM の計算量を決定付ける「多重極子」の数と行列の再帰的分割の方法について十分に検討を加えなければならない。また、divide 段階の $\mathcal{N}$ の分割も等分割でない方が高速である可能性があり、これについても研究が必要である。

2.4. アルゴリズム 3：シフト多項式

アルゴリズム 2 においては、多項式でない関数に対する FMM 内挿の計算量が明らかでないために全体の計算量がわからなかった。仮に FMM 内挿が線形時間で計算できても、多項式の場合に比べて定数係数が大きくなることは避けられない様子である。本節では部分線形結合の内挿を多項式内挿に帰着させるアルゴリズム 3 を提案する。

鍵になるのは、任意の n と ν (ただし $n \equiv \nu \pmod 2$) に対して Legendre 陪関数 P_n と P_ν, $P_{\nu+2}$ の間に成り立つ

$$P_n(\mu) = R^{(0)}_{n,\nu}(\mu^2)P_\nu(\mu) + R^{(1)}_{n,\nu}(\mu^2)P_{\nu+2}(\mu)$$

という式である。ここで、$R^{(l)}_{n,\nu}(\mu^2)$ $(l = 0,1)$ は μ^2 についてのある多項式である (Healy ら [7] は P_n, P_ν, $P_{\nu+1}$ の間で成立する類似の関係式にでてくる多項式を shifted Legendre polynomial と呼んでいるようである)。多項式の次数は、$n > \nu$ の場合は $R^{(l)}_{n,\nu}$ が $(n-\nu)/2 + l$ 次となり、$n \leq \nu$ の場合は $(\nu - n)/2 + 1 - l$ 次となる。但し $R^{(l)}_{n,\nu}$ は $\mu \approx \pm 1$ で極端に大きな値を取る多項式で、これを用いた計算は安定ではない。

一方、$\bar{P}^{(l)}_{n,\nu} = R^{(l)}_{n,\nu}P_{\nu+2l}$ を用いれば

$$P_n = \bar{P}^{(0)}_{n,\nu} + \bar{P}^{(1)}_{n,\nu}$$

のように P_n を 2 つの関数の和に分割できる。この $\bar{P}^{(l)}_{n,\nu}$ の方は下記の例外を除いて数値的に安定である。相対誤差という意味では $T^{(0l)}_{n,\nu} = \bar{P}^{(l)}_{n,\nu}/P_n$ の安定性が問題となるが、これも μ が特別な値でない限りはかなり安定である。例外は n が $m \leq n < 2m$ の範囲で $\nu > n$ の場合で、この場合は数値的に極めて不安定である。しかしこの場合も $\nu \leq n$ にとれば安定に計算することができる。

上記の式を利用すると、$s_{\mathcal{N}}(\mathcal{M})$ は

$$\begin{aligned} s^{(0)}_{\mathcal{N},\mathcal{M}} &= \bar{P}^{(0)}_{\mathcal{N},\nu} g_{\mathcal{N}} \\ s^{(1)}_{\mathcal{N},\mathcal{M}} &= \bar{P}^{(1)}_{\mathcal{N},\nu} g_{\mathcal{N}} \\ s_{\mathcal{N}}(\mathcal{M}) &= s^{(0)}_{\mathcal{N},\mathcal{M}} + s^{(1)}_{\mathcal{N},\mathcal{M}} \end{aligned}$$

と分解できる。以下では ν を**分割点**と呼ぶことにする。このように分割した $s^{(l)}_{\mathcal{N},\mathcal{M}}$ はそれぞれ関数と多項式の積の形になっているので、内挿計算が FMM を用いて線形時間で計算できるようになる。Legendre 陪関数変換の計算量としては $O(M^2 \log M)$ となる。

この方法には内挿計算の回数自体が 2 倍になるというデメリットがある。このデメリットは $n \geq 2m$ の範囲では解決することができる。ν を $\mathcal{N}$ の範囲の中央値にとると、$s^{(l)}_{\mathcal{N},\mathcal{M}}$ は $P_{\nu+2l}$ とおよそ $|\mathcal{N}|/2$ 次の多項式の積になるからである。これにより評価点数が半減し、内挿計算の回数の倍増をキャンセルできる。但し、$n < 2m$ の範囲では数値的な安定性を得るために ν を $\mathcal{N}$ の範囲の最小値にとらなければならないため、この範囲では計算量の倍増を避けることができない。この計算量の倍増は、問題サイズが小さい時の計算量に影響を及ぼす。内挿計算の FMM がまったく効かないと仮定すると直接法の $M^2J/2$ よりも $M^3/24$ 多い。これは $J = 3M/4$ とすると 1/9 のオーバーヘッドとなる。

分割したことで生じるもう一つの問題は、ν の移動の処理が必要となることである。分割点が ν_0 の $s^{(l)}_{\mathcal{N}_0,\mu}$ と分割点が ν_1 の $s^{(l)}_{\mathcal{N}_1,\mu}$ との和を分割点 ν の分割表現にする計算は

$$\begin{aligned} s^{(0)}_{\mathcal{N},\mu} &= T^{(00)}_{\nu_0,\nu}(\mu)s^{(0)}_{\mathcal{N}_0,\mu} + T^{(01)}_{\nu_0,\nu}(\mu)s^{(1)}_{\mathcal{N}_0,\mu} \\ &\quad + T^{(00)}_{\nu_1,\nu}(\mu)s^{(0)}_{\mathcal{N}_1,\mu} + T^{(01)}_{\nu_1,\nu}(\mu)s^{(1)}_{\mathcal{N}_1,\mu} \\ s^{(1)}_{\mathcal{N},\mu} &= T^{(10)}_{\nu_0,\nu}(\mu)s^{(0)}_{\mathcal{N}_0,\mu} + T^{(11)}_{\nu_0,\nu}(\mu)s^{(1)}_{\mathcal{N}_0,\mu} \\ &\quad + T^{(10)}_{\nu_1,\nu}(\mu)s^{(0)}_{\mathcal{N}_1,\mu} + T^{(11)}_{\nu_1,\nu}(\mu)s^{(1)}_{\mathcal{N}_1,\mu} \end{aligned}$$

のようになる。ここで変換係数は

$$T^{(l\lambda)}_{n,\nu} = \bar{P}^{(\lambda)}_{n+2l,\nu}/P_{n+2l}$$

で定義される (変換係数に表れる除算は、評価点集合の選択が適切であれば数値的な安定性に問題を生じない)。内挿計算に比べればかなり少ないとはいえ、この計算のほとんどは直接法では必要のないオーバーヘッドである。

さらに、$n \geq 2m$ の範囲で ν を $\mathcal{N}$ の範囲の中央にとることによっていくつか問題が生じる。まず、分割された $s^{(0)}_{\mathcal{N},\mathcal{M}}$ と $s^{(1)}_{\mathcal{N},\mathcal{M}}$ とで多項式の次数が一般には異なるということである。このため内挿に必要な参照点の数が異なるので注意が必要である。また、内挿行列を計算する際の LU 分解が時々途中で止まってしまう。しかしそこで分解を打ち切ってもかなりよい精度で内挿計算自体はできている。これが実用性に問題を及ぼすような数値的な不安定性を示すものであるかどうか検討が必要である。

最後に、アルゴリズムが大変に複雑になるのもこのアルゴリズムの短所である。これは $\mathcal{N}$ の範囲によって表現方法が異なることが原因である。もし $m \leq n < 2m$ となる n が $\mathcal{N}$ に含まれていなければ分割点 ν は $\mathcal{N}$ の中央の要素が選ばれ、その

表 1. 高速 Legendre 陪関数変換アルゴリズムの比較

論文著者	計算量	精度	安定性	実装評価
須田 [本論文]	$O(M^2 \log M)$	O	O	予備評価のみ
Mohlenkamp[9]	$O(M^2 \log M)$	O?	?	なし
Alpert & Rokhlin[3]	$O(M \log M)$	O?	X?	$m=0$ のみ
Orszag[1]	$O(M \log^2 M / \log\log M)$	X?	O?	なし
Mohlenkamp[9]	$O(M^2 \log^2 M)$	O	O	速度評価はなし
Healy ら [7]	$O(M^2 \log^2 M)$	O	X	不安定
須田 [10]	$O(M^2 \log^2 M)$	O	X	不安定
Orszag[1]	$O(M^{3/2})$	X?	O?	$m=0$ のみ
Mohlenkamp[9]	$O(M^{5/2} \log M)$	O	O	あり

ような n が $\mathcal{N}$ に含まれていれば ν は $\mathcal{N}$ の最小要素となる。しかしいずれの場合でも $\mathcal{N}$ が m または $m+1$ を含んでいれば分割しなくても多項式計算に帰着できるので分割は行なわない。従って、3 種類の方法で線形結合の部分和を表現することになる。さらにこれらの間での変換や併合などを行なう必要があり、アルゴリズムは大変に複雑になってしまうのである。

3. まとめと今後の課題

球面調和関数変換は球面上での関数近似の重要な道具であるが、変換に $O(M^3)$ という計算量がかかることが問題点であった。本稿では FMM を用いた $O(M^2 \log M)$ の高速変換法を提案し、その概要を説明した。このアルゴリズムは線形結合を部分和の問題に再帰的に分割する divide-and-conquer によっていて、部分和の表現として決められた評価点集合上での関数値をならべたベクトルを用いることで数値的な安定性を、FMM に基づいた線形時間の内挿計算を利用することにより高速性を実現したものである。これまでに提案されている高速 Legendre 陪関数変換のアルゴリズムについてまとめると、表 1 のようになる。

今後取り組むべき課題はいくつも残っている。参照点集合の決定方法、FMM のための多重極子数と行列の再帰的分割の決定方法、アルゴリズム 2 の一般化 FMM 内挿の具体的な方法と計算量の評価などは本文でも触れた。これ以外に、球面調和関数展開の微分の評価、効率的な並列処理の手法、アルゴリズムの完全な実装と精度・速度の評価、そしてアプリケーションへの応用について取り組んで行かなければならないと考えている。

References

[1] S. A. Orszag, "Fast Eigenfunction Transforms", Academic Press: Science and Computers, Advances in Mathematics Supplementary Studies, Vol. 10, pp. 23-30, 1986.

[2] L. Greengard and V. Rokhlin, "A Fast algorithm for particle simulations", J. Comp. Phys., Vol. 73, pp. 325–, 1987.

[3] B. K. Alpert and V. Rokhlin, "A Fast Algorithm for the Evaluation of Legendre Expansions", SIAM J. Sci. Stat. Comput., Vol. 12, No. 1, pp. 158-179, 1991.

[4] G. Beylkin, R. R. Coifman, and V. Rokhlin, "Fast Wavelet Transforms and Numerical Algorithms I", Comm. Pure Appl. Math. No. 44, pp. 141-183, 1991.

[5] J. P. Boyd, "Multipole expansions and pseudospectral cardinal functions: A new generation of the fast Fourier transform", J. Comp. Phys., Vol. 103, pp. 184–186, 1992.

[6] 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.

[7] D. M. Healy Jr, D. Rockmore, P. J. Kostelec, and S. S. B. Moore, "FFTs for 2-Sphere — Improvements and Variations", Technical Report PCS-TR96-292, Dartmouth University, 1996.

[8] R. Jakob-Chien and B. K. Alpert, "A Fast Spherical Filter with Uniform Resolution", J. Comp. Phys., Vol. 136, pp. 580-584, 1997

[9] M. J. Mohlenkamp, "A Fast Transform for Spherical Harmonics", Ph.D dissertation, Yale University, 1997.

[10] 須田礼仁,「球面調和関数変換の高速計算について」, 情報処理学会 研究報告 HPC-98-71, pp. 7-12, 1998.