高速球面調和関数変換法の誤差の解析と制御

須田礼仁[†] 高見雅保[†,☆]

球面調和関数変換はフーリエ変換とルジャンドル陪関数変換とに分解されるが，後者は通常，切断周波数 M に対して計算量が $O(M^3)$ となる直接計算で計算されている．これに対し，我々は計算量が $O(M^2 \log M)$ となる高速変換アルゴリズムを提案し，実装・評価をしてきた．本論文では，この高速変換アルゴリズムに対する誤差の解析と制御の方法を提案する．提案手法では，アプリケーションの特性をある程度誤差解析に反映できるように，重みつきの行列ノルムで誤差を評価する．また，対角スケーリングで一貫性不等式をタイトにすることにより，分離点シフトの見かけの不安定性が解消され，誤差の制御が可能となった．さらに，実用的な誤差制御のアルゴリズムの提案と，数値実験によるこれらの手法の有効性の評価を行った．

Error Analysis and Control of Fast Spherical Harmonics Transform

Reiji Suda[†] and Masayasu Takami[†,☆]

Spherical harmonics transform is divided into Fourier transform and associated Legendre transform, and the latter is usually computed directly in time $O(M^3)$ for cut-off frequency M. We have proposed a fast Legendre transform algorithm which runs in time $O(M^2 \log M)$. In this paper, we propose schemes for error analysis and control of our fast transform algorithm. Our scheme evaluates the error with a weighted matrix norm to reflect some characteristics of application in error analysis. By introducing diagonal scaling, the consistency inequality became tight enough to solve the spurious instability of shift of split points and to enable effective error control. We also present a practical algorithm of error control and evaluations of those schemes through numerical experiments.

1. はじめに

球面調和関数変換はフーリエ変換とルジャンドル陪関数変換に分解される．フーリエ変換の部分は高速フーリエ変換法（FFT）により，切断周波数 M に対して $O(M^2 \log M)$ の計算量で計算することができる．しかしルジャンドル陪関数変換に対しては，単純で効率の良い高速変換法が存在しないため，通常は $O(M^3)$ の計算量がかかる直接計算が用いられている．

これに対し，ルジャンドル陪関数変換を高速に行うアルゴリズムの研究がいくつか行われてきた．なかでも D-H 法と呼ばれるフーリエ変換を利用した高速アルゴリズム[3),5)]と，Mohlenkamp による高速ウェーブレット変換を利用したアルゴリズム[4)]は，ライブラリとして実現されつつある．ただし，D-H 法はそのままでは数値的に不安定であり，Mohlenkamp のアルゴリズムは他の高速アルゴリズムに比べ漸近計算量が大きい．これに対し我々は，高速多重極子展開法（FMM）を用いた計算量が $O(M^2 \log M)$ のルジャンドル陪関数の安定で高速な近似変換アルゴリズムを提案・評価してきた[7),8),10)~13)]．

我々のアルゴリズムは近似アルゴリズムであるため，誤差の解析と制御はアルゴリズムの不可欠な要素である．しかしながら，これまでは FMM の精度を決める展開項数を調整することにより間接的にのみ誤差の制御を行ってきた[8),10)]．このため，(1) 目標の精度を達成するために利用者が試行錯誤をする必要がある，(2) 変換結果に対する誤差の影響力の大きさに関係なく一律の相対精度を FMM に要求するため計算量が無駄になっている，などの問題が生じていた．我々のアルゴリズムが行列の形で表現できることは分かっていたが，3.4 節で述べるように通常の解析方法では我々のアルゴリズムの数値的な安定性すら適切に説明する

† 名古屋大学大学院工学研究科計算理工学専攻
Department of Computational Science and Engineering, Nagoya University
☆ 現在，株式会社クボタエンジン事業部エンジン技術部基礎・環境研究チーム
Presently with Advanced & Emission Section, Engine Engineering Department, Engine Division, Kubota Corp.

ことができないため，有効な誤差の解析と制御の方法の開発が遅れていた．

本論文では，我々の高速変換アルゴリズムの誤差の解析と制御の手法を提案する．解析手法においては，アプリケーションの特性を誤差制御にある程度反映することができるように，変換誤差行列の重みつきノルムで誤差を評価・制御することを提案する．そして，行列積のノルムを評価する一貫性不等式に対角スケーリングを加えるという工夫を行った．この工夫により，行列積のノルムをかなり正確に予測評価できるようになり，我々のアルゴリズムの数値的な安定性と誤差の伝播の様子が適切に説明できるようになった．

さらに，この解析手法に基づいて実際に誤差を制御する手法を提案する．提案する誤差制御手法ではFMMと直接計算に対し，スケーリングを行ったうえで前後の計算の影響を加味して誤差を制御する．さらに行ドロッピングを導入することにより，解析手法で必要となる対角スケーリングが不安定になる危険性を避けることができるようになった．その結果，利用者の指定した精度に対し，数値的な安定性が許す範囲で，できるだけ少ない計算量で計算するようにFMMの展開項数などのパラメータを自動的に調節することができるようになった．

以下，本論文では我々が提案する誤差の解析・制御の手法とその評価について論ずる．次章では我々の高速変換アルゴリズムについて説明する．3章では，行列ノルムを用いた誤差の解析方法と，対角スケーリングの必要性について論ずる．4章では，この解析手法に基づく誤差の制御方法を提案する．5章では，解析の手法の妥当性と誤差制御の成否について，実験により評価を行う．6章はまとめである．

2. 高速ルジャンドル陪関数変換アルゴリズム

ルジャンドル陪関数変換

$$g^m(\mu_i) = \sum_{n=m}^{M} g_n^m P_n^m(\mu_i) \qquad (1)$$

は，行列 $(A)_{ij} = P_{m+j}^m(\mu_i)$ とベクトル $(v)_j = g_{m+j}^m$ との積と見なすことができる．我々の高速変換アルゴリズムは，この行列 A をさまざまな方法で分解・分離し，その一部の計算をFMMで高速化している．この章では，我々の変換アルゴリズムの要点と，その行列表現，前処理のアルゴリズムについて述べる．変換アルゴリズムの詳細については文献7)，10)を参照されたい．

なお，我々のアルゴリズムは，位数 m を固定した式(1)で表される変換を高速に計算するものである．以下では添字表記を見やすくするために，固定された位数 m を添字から省略することがある．

2.1 FMMによる高速内挿

我々のアルゴリズムで最も重要な部分は，**FMMによる高速内挿**である．ルジャンドル陪関数 $P_n^m(x)$ はある $n-m$ 次多項式 $q_n^m(x)$ を用いて

$$P_n^m(x) = P_m^m(x) q_n^m(x)$$

のように表現できる．このためルジャンドル陪関数変換(1)から $g^m(x)/P_m^m(x)$ を計算すると $M-m$ 次の多項式になる．したがって，適当な $M-m+1$ 点上で関数 g^m の値が分かっていれば，任意の点での関数値は多項式内挿により計算することができる．ラグランジュの内挿公式によれば，この内挿は

$$g(y_i) = P_m^m(y_i) \sum_{j=0}^{M-m} \frac{\omega_j(y_i)}{\omega_j(x_j)} \frac{g(x_j)}{P_m^m(x_j)} \qquad (2)$$

と表現できる．ここで

$$\omega(x) = \prod (x - x_j)$$
$$\omega_j(x) = \omega(x)/(x - x_j)$$

である．この内挿計算はFMM[1),2),6)]を用いて高速に計算することができる．

FMMによる高速内挿アルゴリズムを用いると，ルジャンドル陪関数変換を2ステップで計算することにより高速化が実現できる．まず準備として，ルジャンドル陪関数変換を評価する点の集合 $\mathcal{M} = \{\mu_i\}$（**評価点集合**と呼ぶ）から適当な $M-m+1$ 点を選んで**標本点集合** $\mathcal{X} = \{x_j\}$ とし，残りの点を**内挿点集合** $\{y_i\} = \mathcal{Y} = \mathcal{M} - \mathcal{X}$ とする．そのうえで，変換の第1ステップでは式(1)により標本点 $x_j \in \mathcal{X}$ 上でのルジャンドル陪関数変換 $g(x_j)$ を計算し，第2ステップでは式(2)によりFMMによる高速内挿を用いて内挿点 y_i 上での値 $g(y_i)$ を計算する．

これを行列表示するために

$$A_{ik} = P_{m+k}^m(\mu_i) \quad \mu_i \in \mathcal{M},\ 0 \le k \le M-m$$
$$\bar{A}_{jk} = P_{m+k}^m(x_j) \quad x_j \in \mathcal{X},\ 0 \le k \le M-m$$
$$\Theta_{ij} = \frac{P_m^m(y_i)\omega_j(y_i)}{P_m^m(x_j)\omega_j(x_j)} \quad x_j \in \mathcal{X},\ y_i \in \mathcal{Y}$$

を定義すると，内挿によるルジャンドル陪関数変換は

$$A = \begin{pmatrix} I \\ \Theta \end{pmatrix} \bar{A}$$

と表現できる☆．ここで I は単位行列であり，Θ はFMMにより高速計算がされ，$\bar{A}$ はもとの A よりも小さな行列になっているので，内挿を用いることによ

り直接 A で計算するよりも計算量が減るのである．

2.2 分割統治法

さらに $\bar{A}$ の計算も高速化するために，**分割統治法**を利用する．そのために次数 n の範囲をほぼ中央 ν で分割する．この ν を**分割点**と呼ぶ．分割された小行列

$$(A_0)_{jk} = P^m_{m+k}(x_j) \quad 0 \le k < \nu - m$$
$$(A_1)_{jk} = P^m_{\nu+k}(x_j) \quad 0 \le k \le M - \nu$$

を定義すると分割は

$$\bar{A} = \begin{pmatrix} A_0 & A_1 \end{pmatrix} \tag{3}$$

と表現できる．分割された 2 つの行列のうち，次数の範囲が下半分となる A_0 については，再び内挿

$$A_0 = \begin{pmatrix} I \\ \Theta_0 \end{pmatrix} \bar{A}_0$$

を適用することができ，さらに $\bar{A}_0$ には再帰的に分割統治法を適用することができる．しかし一方の A_1 の方は，次節で示す**分離**をしなければ内挿をすることができない．

2.3 分離と分離内挿

分割された行列 A_1 にはそのままでは多項式内挿が適用できない．これは式 (2) の内挿の際に行ったようにルジャンドル陪関数変換 $g(x)$ を $P^m_m(x)$ で割っても，低い次数の多項式にならないためである．これを解決するには，**分離ルジャンドル関数**を導入する必要がある．ルジャンドル陪関数は

$$P^m_n(x) = P^{m,0}_{n,\nu}(x) + P^{m,1}_{n,\nu}(x)$$
$$P^{m,l}_{n,\nu}(x) = P^m_{\nu+l}(x) q^{m,l}_{n,\nu}(x)$$

のように分離することができる[7]．ここで，$q^{m,l}_{n,\nu}(x)$ は次数が $|n-\nu+l-1|-1$ 次の多項式である．これに従いルジャンドル陪関数変換も

$$g(x) = g^0(x) + g^1(x)$$
$$g^l(x) = P^m_{\nu+l}(x) \sum_n g^m_n q^{m,l}_{n,\nu}(x)$$

のように分離される．ここでの ν を**分離点**と呼ぶ．分離点は一般的には分割点と異なってもよいが，一致している方が数値的に安定である．

ルジャンドル陪関数変換の分離は，

$$(A^l_1)_{jk} = P^{m,l}_{\nu+k,\nu}(x_j)$$

という行列を用いると，

$$A_1 = \begin{pmatrix} I & I \end{pmatrix} \begin{pmatrix} A^0_1 \\ A^1_1 \end{pmatrix}$$

と表現できる．

分離されたルジャンドル陪関数変換 $g^l(x)$ は $P^m_{\nu+l}(x)$ で割ることにより多項式になるので，FMM による高速内挿が適用できる．これを行列で表すと

$$\begin{pmatrix} A^0_1 \\ A^1_1 \end{pmatrix} = \begin{pmatrix} I & 0 \\ \Theta^0_1 & 0 \\ 0 & I \\ 0 & \Theta^1_1 \end{pmatrix} \begin{pmatrix} \bar{A}^0_1 \\ \bar{A}^1_1 \end{pmatrix}$$

となる．ただし，内挿行列は

$$(\Theta^l_1)_{ij} = \frac{P^m_{\nu+l}(y_i)\omega_j(y_i)}{P^m_{\nu+l}(x_j)\omega_j(x_j)}$$

であり，FMM を適用する $\omega_j(y_i)/\omega_j(x_j)$ の部分は $l = 0, 1$ に対して共通である．

2.4 分離点のシフト

さらに，分離された $\bar{A}^0_1$ と $\bar{A}^1_1$ に対して分割統治法と内挿を適用することを考える．そこで先と同様に次数の範囲をほぼ中央の ν_1 で分割して

$$\begin{pmatrix} \bar{A}^0_1 \\ \bar{A}^1_1 \end{pmatrix} = \begin{pmatrix} A^0_{10} & A^0_{11} \\ A^1_{10} & A^1_{11} \end{pmatrix}$$

とする．ここで次数の範囲が下半分になっている左側はやはり先と同様に内挿が適用できて

$$\begin{pmatrix} A^0_{10} \\ A^1_{10} \end{pmatrix} = \begin{pmatrix} I & 0 \\ \Theta^0_{10} & 0 \\ 0 & I \\ 0 & \Theta^1_{10} \end{pmatrix} \begin{pmatrix} \bar{A}^0_{10} \\ \bar{A}^1_{10} \end{pmatrix}$$

のようにすることができる．

しかし，次数の範囲が上半分となっている右側については 2.2 節の A_1 と同様の状況であり，そのままでは高速内挿が行えない．そこで，ν で分離されている A^l_{11} から ν_1 で分離されている $\hat{A}^l_{11}$:

$$(\hat{A}^l_{11})_{jk} = P^{m,l}_{\nu_1+k,\nu_1}(x)$$

に変換をする．これを**分離点のシフト**と呼んでいる．変換行列

$$T^{l,\lambda}_{\nu,\nu_1} = \mathrm{diag}(P^{m,l}_{\nu,\nu_1+\lambda}(x_i)/P^m_{\nu_1+\lambda}(x_i)) \tag{4}$$

を定義すれば

$$\begin{pmatrix} A^0_{11} \\ A^1_{11} \end{pmatrix} = \begin{pmatrix} T^{0,0}_{\nu,\nu_1} & T^{0,1}_{\nu,\nu_1} \\ T^{1,0}_{\nu,\nu_1} & T^{1,1}_{\nu,\nu_1} \end{pmatrix} \begin{pmatrix} \hat{A}^0_{11} \\ \hat{A}^1_{11} \end{pmatrix}$$

☆ 厳密には，標本点に対応する行を上半分に，内挿点に対応する行を下半分にそれぞれ集める行の入れ替え行列が必要であるが，記述を簡単にするため省略した．以下も同様である．

により，この分離点のシフトをすることができる．このように分離点をシフトすることにより，高速内挿

$$\begin{pmatrix} \hat{A}^0_{11} \\ \hat{A}^1_{11} \end{pmatrix} = \begin{pmatrix} I & 0 \\ \Theta^0_{11} & 0 \\ 0 & I \\ 0 & \Theta^1_{11} \end{pmatrix} \begin{pmatrix} \bar{A}^0_{11} \\ \bar{A}^1_{11} \end{pmatrix}$$

が可能となる．

2.5 前処理とそのアルゴリズム

我々のアルゴリズムはこのような分割統治により，FMM による高速内挿を繰り返し適用して高速性を実現している．分割を繰り返してゆくと部分問題が小さくなり，あるところで FMM による高速内挿計算よりも**直接計算**の方が速くなる．そこで分割統治を停止し，直接計算により部分問題を計算することにする．

分離・分割などの処理や，内挿計算に必要な $P^m_m(x)$ や $\omega_j(y_i)$ などの係数は**前処理**としてあらかじめ計算しておく．そして FMM の展開項数を一定にとると仮定すると，球面調和関数変換の計算量は $O(M^2 \log M)$ となる[7),10)]．

図 1 に，前処理アルゴリズムの概要を擬似プログラムの形で示す．関数 fast_pp(m, n_0, n_1, $\mathcal{M}$, c) は位数が m，次数 n が $n_0 \le n < n_1$ の範囲における評価点集合 $\mathcal{M}$ 上での高速ルジャンドル陪関数変換の前処理を行う．この関数では分枝限定法で計算量の最小化を行っており，引数の c はこの変換にかけることができる計算量の上限を与えるものとし，この変換にかかる計算量を関数 fast_pp の返り値とする．また，関数 dir_pp は fast_pp と同様の引数をとり，直接計算の計算量を返す関数とする．

ここでは，(1) 内挿によらず評価点集合 $\mathcal{M}$ 上の計算をすべて直接計算で行う方法，(2) 標本点集合 $\mathcal{X}$ 上の変換を直接計算で行ってから内挿を行う方法，(3) 標本点集合 $\mathcal{X}$ 上の変換を分割統治法により再帰的に計算してから内挿を行う方法の 3 つの方法を考え，最も計算量の少ない方法を選択する．直接計算による計算量 c_D を上限，分離・シフト・内挿の計算量の合計 c_i を内挿を行う場合の計算量の下限として分枝限定法を適用している．さらに，標本点上の直接計算の計算量 c_d や，分割の片方の子ノードの計算量 c_0 も考慮して，計算量の上限を逐次更新している．なお，4 行目にある行ドロッピングについては 4.4 節で説明する．

```
int fast_pp(m, n0, n1, M, c)
  c_D = dir_pp(m, n0, n1, M)
  c = min{c, c_D} // 上限値を更新
  行ドロッピングを行う // 4.4 節参照
  必要ならば分離・シフトを行う
  標本点集合 X ⊂ M を選択する
  内挿計算の前処理を行う
  分離・シフトと内挿の計算量を c_i とする
  if (c_i > c)
    return c_D // 内挿なしの直接計算を採用
  else // 内挿を試みる
    c_d = dir_pp(m, n0, n1, X)
    c_t = min{c - c_i, c_d} // 上限値を更新
    ν = (n0 + n1)/2 // 分割点を決定
    c_0 = fast_pp(m, n0, ν, X, c_t)
    c_1 = fast_pp(m, ν, n1, X, c_t - c_0)
    if (c_0 + c_1 < c_d かつ c_i + c_0 + c_1 < c)
      return c_i + c_0 + c_1 // 分割統治+内挿を採用
    else if (c_i + c_d < c)
      return c_i + c_d // 直接計算+内挿を採用
    else
      return c_D // 内挿なしの直接計算を採用
```

図 1 前処理アルゴリズムの擬似プログラム
Fig. 1 Pseudo-code of preprocessing algorithm.

3. 誤 差 解 析

本論文の主題は前章で説明した高速変換アルゴリズムに対する誤差の解析と制御の手法の提案である．この章では誤差の解析について，次章では誤差の制御について論ずる．

以下では，一般に行列 A で表される演算に対して，近似が含まれた計算に対応する行列を $\tilde{A}$ で表現するものとする．また，丸め誤差は近似の誤差よりも十分小さく，無視できるものとする．さらに，誤差の解析と制御は（変換のときではなく）前処理の際に行われると仮定し，計算量については論じないこととする．

3.1 誤差の評価の方法

計算の誤差を評価する方法はいくつもある．ルジャンドル陪関数変換は行列として表現できるが，それでも絶対誤差，相対誤差，ノルムなどさまざまな指標が考えられる．誤差解析の最初の重要な決断は，どういう指標で誤差を解析するかということである．ルジャンドル陪関数変換を利用するアプリケーションにはさまざまなものがある．入力となる係数 g^m_n の大きさにどのような傾向があるのか，また変換結果 $g^m(\mu_i)$ はどのように使用されるのか，といったことはアプリケーションごとに異なるものと思われる．誤差行列の「要素ごとの相対誤差」で評価できれば，こういった

違いは吸収できるかもしれない．しかしルジャンドル陪関数はアンダーフローが生じるほど値の変化が激しいため，我々の近似アルゴリズムは行列要素ごとの相対誤差を保証することはできない．

そこで我々は**誤差行列の重みつきノルム**を用いて誤差を評価することを提案する．ルジャンドル陪関数変換を行列 A で，近似変換を $\tilde{A}$ で表すとする．ここで，適当な対角行列 W_R と W_C を用いて重みづけされた誤差ノルム $\|W_R(\tilde{A}-A)W_C\|$ を近似変換の精度の指標に用いるのである．W_R に大きな要素があれば，誤差行列 $\tilde{A}-A$ の対応する行が拡大されて評価される．したがって，W_R は変換結果 $g^m(\mu_i)$ の重みを表現するものと解釈することができる．同様に，W_C は入力係数 g_n^m の重みを表す．これらの重みづけ行列を適切に選択することにより，アプリケーションに対する変換誤差の影響の仕方の違いをある程度反映できるものと期待される．

ただし，最初から変換行列 A に重みづけ行列がかけられていると仮定しても一般性を失わない．そこで，本節ではこれ以降，特に必要のない限り，重みづけ行列 W_R と W_C は明記しないことにする．

以下，行列のノルム解析を用いて誤差の解析を行うが，行列とベクトルのノルムはすべて**一貫性条件**

$$\|AB\| \le \|A\|\|B\| \tag{5}$$

を満たしているものを仮定する．本論文ではこの不等式を**一貫性不等式**と呼ぶ．

3.2 誤差解析の対象

前節で説明したように，我々のアルゴリズムには (1) 内挿，(2) 分割，(3) 分離，(4) シフトがあり，これに (5) 直接計算を加えた 5 つが主な計算である．以下，それぞれの誤差について考察する．

分割：前節の分割を表す式 (3) は

$$\bar{A} = \begin{pmatrix} A_0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & A_1 \end{pmatrix}$$

のように，行列の和に書くことができる．演算 $A+B$ に対して，近似の誤差は

$$\|(\tilde{A}+\tilde{B})-(A+B)\| \le \|\tilde{A}-A\|+\|\tilde{B}-B\| \tag{6}$$

となるから，分割されたそれぞれの演算について誤差を独立に解析してやればよい．分割自身による誤差を考える必要は（丸め誤差を無視すれば）ない．

分離：分離についても，行列表現が $\begin{pmatrix} I & I \end{pmatrix}$ であるから，それ自身の誤差を考える必要はない．

シフト：分離点のシフトについても近似は入れないので，丸め誤差まで正しい計算をしていると仮定することができる．ただし，変換行列 (4) の要素はきわめて大きな値をとりうるので，これが他の原因で生じた誤差をどのように伝播（拡大）するかについて検討を要する．

直接計算：直接計算には近似が入る．これは，ルジャンドル陪関数 $P_n^m(x)$ の次のような性質によるものである．ルジャンドル陪関数の絶対値 $|P_n^m(x)|$ は x が 1 に近づくと，およそ $(1-x)^{m/2}$ の速さでダンプする．この現象は特に次数が $n \le 2m$ の範囲で顕著に現れ，大規模な問題では倍精度浮動小数では関数値を表現しきれずにアンダーフローしてしまうし，組立除法で変換計算を行うと計算途中でオーバフローを起こしてしまう．

しかし，それほど $P_n^m(x)$ が小さくなってしまえば，0 だと思ってもかまわないはずである．そこで，$|P_n^m(x)|$ がある閾値よりも小さくなる (m,n,x) の範囲に関する変換計算を省略することにする．この計算の省略を以下では**ドロッピング**と呼ぶが，これにより上述のアンダーフローやオーバフローの問題は解決できる．直接計算では，このドロッピングで発生する誤差の大きさと，発生した誤差の変換結果への伝播について，解析と制御を行う必要がある．

内挿：内挿では FMM を用いて近似計算をしているので誤差が発生する．さらに，内挿はつねに計算の途中にあるので，他の計算で発生した誤差を伝播する．このように，内挿に関しては誤差の発生と誤差の伝播の両方について考慮しなければならない．

3.3 誤差解析の範囲と誤差の分解

以上の考察から，誤差の発生を考える演算は内挿と直接計算であることが明らかとなった．また式 (6) から分割された部分の誤差は個別に扱えばよいことになるので，考えている演算の入出力に直接関係する計算だけを考慮すれば足りる．

以上の考察に基づき，以下では直接計算と内挿の誤差を解析するための基本的な式を導く．

直接計算：ある直接計算 D を考える．直接計算の入力は係数ベクトル v であるから，この計算 D に関係する他の計算は，D の変換結果から最終的な変換結果を得るまでの部分だけである．この部分の計算は適当な行列 A で表現できるはずである．この A は直接計算 D の変換結果を内挿によりすべての評価点の上での値を求める計算に相当する．

直接計算 D に関する誤差の発生とその伝播を解析するときには，A は厳密な内挿ではなく，近似を含む内挿 $\tilde{A}$ となることに注意する．すなわち，解析すべき誤差は $\tilde{A}\tilde{D}-AD$ となる．しかし，$\tilde{D}-D$ を解析・制御するのに比べて行列積を含む $\tilde{A}\tilde{D}-AD$ を解析・制御するのは格段に面倒である．

しかし，この誤差は

$$\|\tilde{A}\tilde{D}-AD\| \le \|\tilde{A}(\tilde{D}-D)\|+\|(\tilde{A}-A)D\| \quad (7)$$

のように**分解**することができる．右辺第 1 項は，直接計算の誤差（$\tilde{D}-D$）を制御するときには，近似を含む内挿（$\tilde{A}$）に基づいて行うべきことを示しており，右辺第 2 項は，内挿の誤差（$\tilde{A}-A$）を制御するときには，右につく直接計算は厳密な計算（D）を仮定して制御すべきことを示している．このように近似計算と厳密な計算とを適切に区別して用いることにより，誤差をともなう誤差伝播を正確に扱うことができる．

なお，同様な分解として

$$\|\tilde{A}\tilde{D}-AD\| \le \|(\tilde{A}-A)\tilde{D}\|+\|A(\tilde{D}-D)\|$$

を考えることもできる．しかし 2.5 節の前処理アルゴリズムでは，分割統治法によって近似行列が左側から順に決定されて行く．したがって，式 (7) のように左側を近似計算とする分解の方が適しているのである．

内挿：同様に，ある内挿計算 C の誤差にかかわる計算は，適当な行列 A，B を用いて ACB と表すことができる．このうち B は内挿の入力となる標本点上での関数値を計算する部分，A は内挿の結果をさらに内挿してすべての評価点での値を計算する部分に相当する．分離や分離点シフトがあれば，それも A に含まれることになる．

誤差はやはり $\tilde{A}\tilde{C}\tilde{B}-ACB$ の形で評価する必要がある．ここでも直接計算のときと同じように

$$\begin{aligned}\|\tilde{A}\tilde{C}\tilde{B}-ACB\| &\le \|\tilde{A}\tilde{C}(\tilde{B}-B)\| \\ &\quad +\|\tilde{A}(\tilde{C}-C)B\|+\|(\tilde{A}-A)CB\| \quad (8)\end{aligned}$$

のようにそれぞれの誤差の項に分離することができる．

3.4 一貫性不等式の利用

前小節の解析方法を用いると，直接計算の誤差 $\tilde{D}-D$ については $\|\tilde{A}(\tilde{D}-D)\|$ を，内挿の誤差 $\tilde{C}-C$ については $\|\tilde{A}(\tilde{C}-C)B\|$ を評価すればよいことになる．しかしながら，行列の積の計算は計算量が大きいため，この形のままで誤差の解析や制御を行うのは不便である．

ここで，行列ノルムの**一貫性不等式**を利用すると，たとえば直接計算の誤差は

$$\|\tilde{A}(\tilde{D}-D)\| \le \|\tilde{A}\|\|\tilde{D}-D\| \quad (9)$$

と評価できる．こうすれば誤差 $\tilde{D}-D$ とその伝播 $\tilde{A}$ を分離して扱うことが可能になる．

しかし，一般に式 (9) のような一貫性不等式は非常にゆるい．実際にこのようにして誤差を解析しようとすると，我々の高速変換アルゴリズムは数値的にきわめて不安定であるような結果が導かれてしまう．これは分離点でのルジャンドル陪関数 $P^m_{\nu_1+\lambda}(x)$ が（たまたま x が零点に近いために）0 に近いときに，これを分母に持つ変換行列 (4) のノルムが非常に大きくなってしまうことによっている．ところがそのような非常に大きな要素を含んでいるにもかかわらず，我々の変換アルゴリズムは D-H 法[3),5)] などに比べて数値的にかなり安定である．普通に行列のノルムをとって誤差とその伝播を解析していたのでは，我々のアルゴリズムの本質をとりこぼしてしまうのである．

3.5 対角スケーリングの導入

この問題を解決するため，我々は**対角スケーリング**を用いることを提案する．すなわち，ある対角行列 S_A を定義し，直接計算における誤差とその伝播を

$$\|\tilde{A}(\tilde{D}-D)\| \le \|\tilde{A}S_A^{-1}\|\|S_A(\tilde{D}-D)\| \quad (10)$$

という形で分離するのである．スケーリング行列 S_A を適切に選べば，不等式のゆるさをかなり緩和できると期待される．内挿の誤差の解析においても，同様にスケーリング行列 S_A，S_B を導入して

$$\begin{aligned}&\|\tilde{A}(\tilde{C}-C)B\| \\ &\quad \le \|\tilde{A}S_A^{-1}\|\|S_A(\tilde{C}-C)S_B\|\|S_B^{-1}B\| \quad (11)\end{aligned}$$

のように誤差を評価する．

スケーリング行列 S_A，S_B の具体的な選択はいくつか考えられる．我々は報告 9) で 2 種類のスケーリングについて考察した．第 1 のスケーリングは**アルファ・ベータ・スケーリング**と呼んでいるもので，$\tilde{A}$ の第 i 列のノルム（行列ノルムと一貫しているベクトルノルム）を α_i，B の第 i 行のノルムを β_i として，S_A は α_i を並べた対角行列，S_B は β_i を並べた対角行列とする．これは $\tilde{A}S_A^{-1}$ の列および $S_B^{-1}B$ の行のノルムを一様にするもので，行列の条件を改善するスケーリングとしてはよく知られたものである．アルファ・ベータ・スケーリングによる一貫性不等式がどの程度タイトであるかについては，5 章で実験結果を示す．

第 2 のスケーリングは**分離関数スケーリング**と呼んでいるもので，直接計算や内挿といった特定の演算に限らず，一般的に適用することを目指している．証明されているわけではないが，分離関数スケーリングを用いると，分離シフト行列の不安定性が見かけだけであるということが説明できるらしいという実験結果が得られている[9)]．

いずれにせよ，適当なスケーリングを行うことにより，一貫性不等式 (10)，(11) を用いて誤差の解析と制御を行うことが可能となる．次章では本章で提案した誤差解析手法に基づいて，具体的な誤差の制御の方法を提案する．

なお，スケーリング (10)，(11) と 3.1 節で導入した重みつきノルム $\|W_R(\tilde{A}-A)W_C\|$ とは，対角行列

をかけてノルムをとる共通の形をしている．しかし，スケーリングは一貫性不等式による誤差の分離のために内部的に用いられるのに対して，重みづけはアプリケーションによる要求精度の定義のために用いられるものである．このため本論文では両者に異なる名前をつけて区別をすることとした．

4. 誤差制御の方法

本章では，前章で導入されたスケーリングつきの一貫性不等式による誤差解析をベースとした誤差制御アルゴリズムを提案する．

ここで考えている誤差制御の目的は 2 つある．第 1 の目的は指定した精度で高速変換を行うこと，第 2 の目的はできるだけ少ない計算量で指定した精度を実現することである．また本論文では調節可能なパラメータとして，直接計算のドロッピングの閾値と FMM の精度を決める展開項数の 2 つを考える．これ以外にも，分割統治法の分割における次数の分割点，分離ルジャンドル関数の分離点，評価点集合からの標本点集合の選択などが精度に影響する．しかし，これらの要素は誤差よりもその伝播に関するものであるので，本論文では計算量と数値的安定性を考慮して適切に決められているものと仮定する．

4.1 精度の分配

分割統治法による行列の分割に対して，誤差の分離の式 (6) が成り立つ．これにより，誤差の解析は分割された 2 つの要素に対して独立に評価してその和をとることにより行うことができる．直接計算と内挿の誤差の相互関係を示す式 (7) と式 (8) の場合も行列の和であるから同様である．

これらの式に基づいて厳密に誤差を制御する場合には，指定された精度 ϵ を**分配**し，それぞれの要素に割り振ることになる．たとえば式 (6) の場合には，$\epsilon = \epsilon_A + \epsilon_B$ のように精度を分配して

$$\|\tilde{A} - A\| \leq \epsilon_A$$
$$\|\tilde{B} - B\| \leq \epsilon_B$$

のように誤差を制御しなければならない．ここで計算量の最小化を目指すとすると，要求精度 ϵ をどのように ϵ_A と ϵ_B とに分割するかが問題となるが，これを最適化するのは一般には容易ではない．

しかし，実験的には次のようなことが分かっている．(1) ノルムとして 2 ノルムを採用すると,「行列の和のノルム」は「行列のノルムの和」よりも「行列のノルムの最大値」に近い挙動を示す．したがって，それぞれの近似に対して要求精度に一定の**安全係数**をかけた一律な精度を要求することでほとんど足りる．(2) 高速内挿に比べると，直接計算の方が要求精度に対する計算量の影響が少ない．したがって，直接計算へ分配する精度を高めにとることにより内挿計算に要求される精度を低くするほうが有利である．

この (1) のような簡単化をすると，厳密に誤差をバウンドすることはできなくなる．しかし次節で示すように，実際にこのような制御の方法により，実用上支障ない程度に目標の精度に近い結果を出すことができる．

4.2 直接計算の誤差制御

前節の誤差解析に基づき，直接計算 D の誤差は式 (10) で評価する．対角スケーリングとしては**アルファ・ベータ・スケーリング**を用いる．また，行列のノルムとしては **2 ノルム**を用いる．行列 $\tilde{A}$ が陽に求まっている場合には 2 ノルムよりもフロベニウスノルムなどの方が計算が容易であるが，実際には $\tilde{A}$ は高速変換の一部であって，アルゴリズムとしては実現されているが各要素の値は与えられていない．このためべき乗法を用いて 2 ノルムを推定する方が効率的なのである．

しかし，直接計算のドロッピングの判定のために何度も誤差行列 $S_A(\tilde{D} - D)$ の 2 ノルムを計算するのは面倒である．そこで，**フロベニウスノルム**を用いて

$$\|S_A(\tilde{D} - D)\|_2 \leq \|S_A(\tilde{D} - D)\|_F \qquad (12)$$

のように再評価する．そして要求精度 ϵ に対して

$$\|S_A(\tilde{D} - D)\|_F \leq \epsilon / \|\tilde{A} S_A^{-1}\|_2$$

となるようにドロッピングの閾値を制御する．スケーリングによる一貫性不等式 (10) と 2 ノルムのフロベニウスノルムでの代用 (12) の 2 つの不等式は，後述するように比較的タイトである．

厳密に誤差行列のフロベニウスノルムを制御することは難しくはないが，ここでも安全係数を掛けて誤差要素の最大値で置き換えてやってもよいようである．前述のように，直接計算の計算量は要求精度に対して比較的敏感ではないので，大き目の安全係数を掛けておけばよい．

直接計算の誤差制御は，2.5 節の擬似プログラムでは関数 dir_pp の中で行われることになる．

4.3 FMM の誤差制御

内挿 C についても，前節で導いた式 (11) によって誤差を制御する．要求精度を ϵ とすると

$$\|S_A(\tilde{C} - C)S_B\| \leq \epsilon / (\|\tilde{A} S_A^{-1}\| \|S_B^{-1} B\|)$$

のように FMM の精度を制御すればよい．FMM は展開項数 K に対して指数関数的に精度が向上するので，その誤差の挙動は適当な定数 σ と ρ を用いて

$$\|S_A(\tilde{C} - C)S_B\| \approx \sigma \rho^K \|S_A C S_B\| \qquad (13)$$

で近似できると期待される．これに従えば，

$$\gamma = \|\tilde{A}S_A^{-1}\|\|S_ACS_B\|\|S_B^{-1}B\|$$
$$K \approx \log_\rho \epsilon/(\sigma\gamma) \quad (14)$$

のように最適な展開項数 K を決定することができると期待される．

実際には，内挿に関する一貫性不等式 (11) はかなりゆるく，FMM の精度の予測 (13) もかなり精度が悪いため，σ や ρ といったパラメータを調節しても，目標の精度から 1〜2 桁ずれることが多い．しかし FMM は精度に対して計算量が敏感に反応するので，できるだけ目標精度ぎりぎりの精度を達成するのが望ましい．そこで我々の実装では，いったん式 (14) で K の初期値を決めておき，そこから誤差ノルム $\|\tilde{A}(\tilde{C}-C)B\|_2$ を計算し，それが目標精度にあうように K を**修正**することにしている．必要ならば修正を**反復**することにより，要求精度を満たす最小の K を決定することができる．

FMM の誤差制御は，2.5 節の擬似プログラムにおいては「内挿計算の前処理」の部分で行われることになる．

4.4 行ドロッピング

以上で誤差制御のアルゴリズムの説明を終わるが，ここでアルファ・ベータ・スケーリングを計算機に実装する際に生じる 1 つの問題について述べておかなければならない．それは，ルジャンドル陪関数のダンプのために，B のいくつかの行の全要素がアンダーフローを起こしてしまうことである．このとき β_i は数値的に 0 となり，S_B^{-1} はオーバフローしてしまい，そのままでは適切な処理ができない．この問題を避けるためには，以下に述べる行ドロッピングが不可欠である．内挿計算 ACB において CB の部分を取り出すと，これはある直接計算 D で表現することができるはずである．このとき，$ACB = AD$ となり，直接計算の誤差解析・制御に現れた行列と同じ形になる．そこで，仮想的に D にドロッピングを適用し，行全部がドロッピングされるような D の行を数え上げる．ドロッピングが適切になされていれば，これらの行に対応する評価点を内挿計算から除外してしまっても変換の誤差は要求精度以内に収まるはずである．このようにして一部の評価点に対する内挿計算を省いてしまうのが**行ドロッピング**である．

行ドロッピングを行ったあとの B は，ドロッピングされなかった行を集めてきた行列になる．したがって，β_i はアンダーフローを起こすような極端に小さな値にはなりえない．このように行ドロッピングを導入することにより，アンダーフローを起こすほど絶対値が変化するルジャンドル陪関数変換に対してでも，通常の浮動小数点数を用いてアルファ・ベータ・スケーリングが使えるようになるのである．

5. 数値実験

この章では，本論文で提案している誤差の解析と制御のアルゴリズムが実際に適切に働くかどうか，実装を通して評価を行う．実装に用いたプログラミング言語は C である．評価点はガウス点で，エイリアス除去のために一般に行われている方法に従い，評価点の数は切断周波数 M のおよそ 3/2 倍とした．また，ルジャンドル陪関数の対称性を利用して計算量を約半分に節約する工夫を採用しているので，ルジャンドル陪関数変換は偶数次と奇数次の 2 つの変換を別々に実行する形になっている．全体のアルゴリズムは 2.5 節で示した擬似プログラムに従っており，分割点 ν は次数の範囲の中点，分離点は次数の範囲の最小値 n_0 を用いている．標本点選択のアルゴリズムには対角スケーリングの効果を取り入れ，式 (14) などに現れる $\|S_ACS_B\|$ をできるだけ小さくするように工夫をしている[9]．FMM はテーラー展開に基づく近似を用いている[11]．要求精度 ϵ は丸め誤差が問題にならない範囲であればどのように設定しても同じような結果になるので，本論文では $\epsilon = 10^{-10}$ の結果のみを示す．なお，実験の計算時間を短縮するために，位数 m を等間隔に 10 個サンプリングした．

重みづけ行列 W_R と W_C は次のように決めている．W_C は 2 種類準備した．1 つ目の W_C は各列のノルムが一定になるようにしている．これはこれまでの報告[8)~10)]と同じである．一般には係数 g_n^m は次数 n が大きくなるに従ってダンプしてゆくと期待されるが，この W_C はそれを加味しない，いわば「最も厳しい」評価基準になっている．2 つ目の W_C は 1 つ目の W_C の n 番目の対角要素を $(n+1)^{-2}$ 倍したものである．高次数の係数は値が小さくなることを期待した重みづけになっている（なお，指数に用いた -2 という数値は特定の物理シミュレーションを念頭に置いたものではない．これまで展開係数の大きさに関するこの種の検討はあまり行われていなかったらしく，参照できる数値が見当たらない．現在シミュレーションの研究者とともに検討中である）．一方，W_R は 1 つ目の W_C を用いたときに $(W_RAW_C)^T(W_RAW_C) = I$ となるようにとった．順変換の結果を逆変換すればもとに戻るはずであるので，これを誤差評価の基準に利用したのである．

各部分への要求精度は，前章の提案に従い，安全係数を掛けた一律の値とした．今回の実験では，和のノ

表 1 達成精度とバウンドのゆるさ
Table 1 Achieved precision and looseness of bounding inequality.

M	K	W_C	全体誤差		高速内挿			直接計算		行ドロップ	
			誤差	予測比	全体	一貫性	FMM	一貫性	Fr	一貫性	Fr
341	19.5	1.09	1.1e-10	2.5	438.1	12.0	79.4	4.2	5.7	4.2	1.0
511	20.2	1.16	1.2e-10	2.6	1320.4	26.3	570.1	5.6	7.2	5.6	1.0
682	20.1	1.18	1.5e-10	2.6	1946.5	31.72	160.5	4.7	7.6	6.1	1.0
1023	20.4	1.26	1.2e-10	4.1	6526.7	158.6	359.9	6.1	7.5	7.9	1.0
1365	20.2	1.32	1.4e-10	5.6	3418.8	98.3	755.5	9.7	7.8	9.7	1.0
2047	21.0	1.43	1.6e-10	6.3	14734.4	246.4	440.9	17.0	7.4	18.3	1.0
2730	20.9	1.52	1.2e-10	5.4	1884.3	181.2	726.6	17.2	7.7	24.2	1.0
4095	21.6	1.66	5.4e-9	6.1	10325.6	1066.3	1499.0	61.3	9.4	72.5	1.0

ルムがノルムの最大値に近い挙動をすることを確認しやすいように，直接計算，内挿，行ドロッピングのいずれに対しても同じ 0.8ϵ という要求精度を与えている．

表 1 は数値実験の結果をまとめたもので，各欄の数値は以下のとおりである．M は切断周波数，K は FMM の展開項数の（内挿サイズで重みづけした）平均値，W_C は 2 つの W_C の設定による計算量の比である．全体誤差，高速内挿，直接計算，行ドロッピングに対しては，誤差制御の不等式や近似式のゆるさを，両辺の比の最大値を計算することによって評価した．全体誤差の「誤差」欄は誤差そのものすなわち誤差行列の重みつきノルム,「予測比」は行列のノルムの和と行列の和のノルムの比で，いずれも最大値である．高速内挿については，式 (14) により決定される K に対する評価を行った．要求精度と式 (14) の K で得られた精度との比を「全体」欄に，一貫性不等式 (11) のゆるさを「一貫性」欄に，FMM の精度予測 (13) の悪さを「FMM」欄に，それぞれ示した．直接計算と行ドロッピングについては，一貫性不等式 (10) のゆるさを「一貫性」欄に，2 ノルムとフロベニウスノルムの比 (12) を「Fr」欄に，それぞれ示した．なお，W_C 欄以外はいずれも基準が厳しい 1 つ目の W_C における結果である．

以下，実験結果について考察する．FMM に要求された平均展開項数 K はほとんど一定であるが，M の増加に従ってきわめて緩やかに増加しているようにも思われる．式 (14) で K の挙動がおよそ決まるとすると，右辺に現れるノルムの積の値が M の増加に従い大きくなっているものと思われる．この式 (14) の右辺の値は標本点集合により決まり，アルゴリズムの数値的な安定性に大きくかかわっている．標本点選択のアルゴリズムの改良により平均展開項数 K をもっと小さくできるかどうか，現在検討中である．

重みづけ W_C の違いによる計算量の比を見ると，サイズが大きくなるほど開いてくることから，ダンピングした重みづけによりゆるめられた要求精度を利用して計算量を下げることに成功していることが分かる．実は，ダンピングした重みづけでは FMM の平均展開項数 K が M の増加に従ってはっきりと減少している．なお，重みづけノルムによる誤差制御は高次数に対して計算精度を低くするが，これがアプリケーションに与える影響は自明ではない．この問題については今後アプリケーション側の研究者と協力して慎重に検討をし，評価を行わなければならないと考えている．

変換全体の誤差については，目標精度にほぼ一致した値をつねに出している．この「誤差」の値に「予測比」の値を掛けたものが,「行列の和のノルム」を「行列のノルムの和」でバウンドしたときの誤差の予測値であり，実際の値よりもかなり大きくなってしまう．このように，むしろ「ノルムの最大値」で近似することでうまく誤差が制御できることが分かる．$M = 4095$ では目標の精度が達成されていないが，これは誤差制御が失敗したのではなく，数値的な不安定性のために倍精度計算では目標精度が実現できないためであり，数値的な安定性のさらなる改善が必要であると考えられる．

高速内挿に関しては，予測精度の悪さが目を引く．式 (11) で一貫性不等式が 2 度使われていることと，FMM の展開項数と内挿の精度の関係が式 (13) のように単純でないことが相乗的に影響している．したがって，内挿においては 4.3 節で説明したような FMM の展開項数の再調整が必須である．

一方，直接計算と行ドロッピングはほぼ同じ落ち着いた挙動を示している．サイズ N の行列に対して 2 ノルムとフロベニウスノルムとの比は最大で $\sqrt{N}$ になるはずであるが，実際の違いはずっと小さい．特に行ドロッピングではノルムの違いは 5%にも達しない．これはルジャンドル陪関数のダンピングが急速であるため，ごく少数の大きな誤差要素がノルムを支配して

いることによるものと思われる．しかし，一貫性不等式はサイズ M に比例するような勢いでゆるくなっているので，もっと大きな問題では FMM のような誤差制御の再調整が必要となるかもしれない．

また，今回の誤差制御アルゴリズムでは W_C のダンピングをさらに加速して $(n+1)^{-3}$ にすると，大規模な問題では目標精度が達成されなくなってしまった．これは $n \approx 2150$ で $(n+1)^{-3}$ が目標精度である 10^{-10} に達してしまうため，それ以上の次数のルジャンドル陪関数展開に意味がなくなってしまうことが原因である．このため極端な計算の省略が行われ，上述の「ごく少数の大きな誤差要素がノルムを支配している」状況が崩れてしまうのである．このような場合にも安全係数を再調整することにより目標精度を確実に達成することは容易であるが，実用的にはむしろ切断周波数が高すぎて無駄な展開を行おうとしていると警告することが必要であろう．

6. まとめと課題

本論文では，我々が提案してきている高速ルジャンドル陪関数変換法の誤差の解析と制御の方法を提案した．また，提案手法の有効性を数値実験を通して確認した．誤差の解析においては，適切に対角スケーリングを行うことにより，一貫性不等式を用いて誤差とその伝播を分離して考察できるようになった．また誤差の制御においては，問題を簡単にするいくつかの近似を導入することにより，比較的容易に誤差を制御する手法を提案した．提案手法のそれぞれの構成要素は新規性が非常に高いとはいえないかもしれないが，誤差解析の対象のアルゴリズムが独自のものであるということを除いても，分離シフト行列の見かけの不安定性やアンダーフロー・オーバフローなどの微妙な問題を含む変換アルゴリズムに対するトータルな誤差の解析と制御の手法の提案としては新規性を主張できるものと考えている．ただし，今回提案した誤差の制御がアプリケーションにおいて適切に働くかどうかについては慎重な検討が必要である．特に，重みづけで仮定するダンピングの速さおよび切断波数と要求精度の関係は自明ではなく，個々のアプリケーションで研究と評価を行っていかなければならない．

また，アルゴリズムの基本的な問題として，安定性の最適化が残されている．今回はスケーリングを考慮した標本点選択アルゴリズムを用いたが，残念ながら $M = 4095$ においてやや不安定な挙動が見られた．この不安定性は分離ルジャンドル関数に起因していることが分かっている．現在，分離における数値的な不安定性を最小限に抑える手法について研究を行っている．

謝辞 あらゆる意味で私たちの研究を支えてくださっている杉原正顯教授に深く感謝いたします．また，有益なコメントをくださっている下記の未来開拓プロジェクトの皆様，特に金田行雄教授とポスドクの赤堀浩司様に感謝いたします．また，論文をよりよくするための貴重なコメントをくださった査読者の皆様にも深く感謝をいたします．

この研究の一部は，学振未来開拓プロジェクト（地球規模流動現象解明のための計算理工学）および文部科学省科研費の支援を受けています．

参 考 文 献

1) Boyd, J.P.: Multipole expansions and pseudospectral cardinal functions: A new generation of the fast Fourier transform, *J. Comput. Phys.*, Vol.103, pp.184–186 (1992).
2) Greengard, L. and Rokhlin, V.: A Fast algorithm for particle simulations, *J. Comput. Phys.*, Vol.73, pp.325–348 (1987).
3) Healy, Jr., D.M., Rockmore, D., Kostelec, P.J. and Moore, S.S.B.: FFTs for 2-Sphere — Improvements and Variations, Tech. Rep. PCS-TR96-292, Dartmouth Univ. (1996).
4) Mohlenkamp, M.J.: A Fast Transform for Spherical Harmonics, *J. Fourier Anal. Appl.*, Vol.2, pp.159–184 (1999).
5) Potts, D., Steidl, G. and Tasche, M.: Fast and stable algorithms for discrete spherical Fourier transforms, *Linear Algebra Appl.*, pp.433–450 (1998).
6) Reif, J.H. and Tate, S.R.: N-body Simulation I: Fast Algorithms for Potential Field Evaluation and Trummer's Problem, Tech. Rep. N-96-002, Univ. of North Texas, Dept. of Computer Science (1996).
7) 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).
8) 須田礼仁，高見雅保：高速球面調和関数変換法の精度と速度，情報処理学会研究報告 2000-HPC-83，pp.19–24 (2000).
9) 須田礼仁，高見雅保：高速球面調和関数変換法の誤差の解析と制御，情報処理学会研究 2000-HPC-84, pp.7–12 (2000).
10) Suda, R. and Takami, M.: A Fast Spherical Harmonics Transform Algorithm, *Math. Comp.*（掲載予定）
11) 高見雅保，須田礼仁，杉原正顯：FMM による Legendre 陪関数変換の高速化，情報処理学会研究報告 99-HPC-78，pp.1–6 (1999).
12) 高見雅保，須田礼仁：Legendre 陪関数変換の高速計算にともなう FMM の改良，豊田研究報告，

Vol.53, pp.77–84 (2000).
13) 高見雅保，都竹隆広，須田礼仁：FMM のChebyshev 近似による高速化とその球面偏微分方程式の解法への応用，豊田研究報告，Vol.54, pp.1–7 (2001).

(平成 13 年 4 月 30 日受付)
(平成 13 年 8 月 24 日採録)

須田 礼仁（正会員）

1968 年生．1991 年東京大学理学部情報科学科卒業．1993 年同大学院理学系研究科情報科学専攻修士課程修了．1996 年同博士（理学）．現在名古屋大学大学院工学研究科計算理工学専攻助教授．日本応用数理学会会員．

高見 雅保

1975 年生．1999 年名古屋大学工学部応用物理学科卒業．2001 年同大学院工学研究科計算理工学専攻修士課程修了．現在クボタ（株）エンジン事業部エンジン技術部基礎・環境研究チーム勤務．