ハイパフォーマンス
コンピューティング 76－9
（1999．5．14）

Orszagの高速Legendre多項式変換法の改良

森 明子†, 須田礼仁‡, 杉原 正顯‡

†株式会社 日立製作所 エンタープライズサーバ事業部
‡名古屋大学　工学研究科　計算理工学専攻

概要

1986年にOrszagはLegendre多項式変換を含むSturm-Liouville固有関数変換の評価計算に対する高速算法を提案した。彼の算法は固有関数のWKB近似を用いて計算の一部でFFTを利用することにより、直接計算で$O(N^2)$かかる計算量が$O(N\log^2 N/\log\log N)$に改善できるというものである。しかし彼の算法は計算の無駄が多く、精度も悪く実用的ではない。我々はLegendre多項式変換の場合についてこのOrszagの算法を改良したので報告する。アルゴリズムの改良により計算量は$O(N\log N)$に改善され、高精度近似公式の採用により、単精度程度の精度で$N \geq 128$, 倍精度程度の精度で$N \geq 256$で直接法よりも高速となることがわかった。

An Improvement on Orszag's Fast Algorithm for Legendre Polynomial Transform

Akiko Mori†, Reiji Suda‡, and Masaaki Sugihara‡

†Enterprise server division, HITACHI, Ltd.
‡Dept. of Computational Science and Engineering,
Graduate School of Engineering, Nagoya University

Abstract

In 1986 Orszag proposed a fast algorithm for Sturm-Liouville eigenfunction transform including the Legendre polynomial transform. His algorithm, which exploits the high speed of FFT through the WKB approximation, runs in time $O(N\log^2 N/\log\log N)$, while the direct computation requires $O(N^2)$ time. His algorithm is, however, not practical because of its low precision and algorithmic unsophisticatedness. We improve Orszag's algorithm in the case of the Legendre polynomial transform. The improved algorithm runs in time $O(N\log N)$, and the Stieltjes's higher order approximation formula enables high precision. Our scheme is faster than the direct method for $N \geq 128$ with the error tolerance $\varepsilon = 10^{-6}$ and for $N \geq 256$ with $\varepsilon = 10^{-12}$.

1. はじめに

直交多項式や特殊関数による関数の変換は計算物理などにおいて重要な手法である。このうちFourier変換には高速に計算するアルゴリズムが存在するが、多くの変換では直接的な計算法が用いられている。これに対して、Orszag [2] は固有関数（Sturm-Liouville問題の固有関数）のWKB近似に基づく、広範な固有関数変換に適用可能な高速算法を提案した。しかし、現在までのところOrszagの算法を用いて実際に応用計算をしたという報告は見当たらない。これは、Orszagの算法は精度が悪く、アルゴリズムも複雑であるためと思われる。

我々は、固有関数変換の一つであるLegendre多項式変換

$$A_j = \sum_{n=0}^{N-1} a_n P_n(\cos((j+\frac{1}{2})\frac{\pi}{N})) \quad (0 \leq j < N) \quad (1)$$

($P_n(x)$ は n 次のLegendre多項式) を取り上げて、Orszagの算法の改良を試みた。その結果、近似公式の改良により高い精度を実現し、アルゴリズムの改良により計算量を削減することができた。本稿ではまずOrszagの算法とその問題点を説明した後、我々の提案する改良法と実装・評価の結果について報告する。

2. Orszagの算法とその問題点

2.1. Orszagの算法

Orszagは Sturm-Liouville 問題

$$\frac{d}{dx}\left(p(x)\frac{d}{dx}\psi_n(x)\right) + [\lambda_n w(x) - q(x)]\psi_n(x) = 0$$

の固有関数 $\psi_n(x)$ で展開された関数の評価計算

$$A_j = \sum_{n=0}^{N-1} a_n \psi_n(x_j) \quad (0 \le j < N) \tag{2}$$

に対する高速算法を提案した。評価点 x_j が固有関数 $\psi_N(x)$ によって決まるある特別な点のとき、固有関数 $\psi_n(x)$ に対する WKB 近似:

$$\psi_n(x_j) \approx (w(x_j)p(x_j))^{-1/4} \cos(n\pi(j+1)/N) \tag{3}$$

はある十分大きい定数 n_0 に対して

$$\min\{nj, n(N-j)\} > n_0 \tag{4}$$

なる条件のもとで成立することが知られている。条件 (4) を満たす (j,n) の集合を S とすると、計算 (2) は

$$\begin{aligned} A_j \approx & (w(x_j)p(x_j))^{-1/4} \sum_{(j,n)\in S} a_n \cos\frac{n(j+1)\pi}{N} \\ & \sum_{(j,n)\notin S} a_n \psi_n(x_j) \end{aligned} \tag{5}$$

のように近似できる。Orszag の算法は、式 (5) の右辺第 1 項の一部に FFT を適用することによって高速化を実現するものである。

FFT を適用する具体的方法として、Orszag は以下の方法を提案した。まず適当な M と K を設定して S に含まれる長方形

$$S^{(0)} = \{(j,n) \mid M < n < N, K \le j < N-K\}$$

を定義し、計算 (2) を

$$\begin{aligned} A_j \approx & (w(x_j)p(x_j))^{-1/4} \sum_{(j,n)\in S^{(0)}} a_n \psi_n(x_j) \\ & + \sum_{(j,n)\notin S^{(0)}} a_n \psi_n(x_j) \end{aligned} \tag{6}$$

のように近似するのである。計算量が最小になるように M と K をとると (6) の計算量は $O(N^{3/2})$ となる。

上記の方法は、近似が成立する領域 S よりもかなり狭い $S^{(0)}$ に FFT の適用を制限している。Orszag はこの欠点の改善方法として、次のような方法を提案した。まず M_i, K_i $(i = 1, \cdots, k)$ を用いて S に含まれる k 個の長方形

$$S^{(i)} = \{(j,n) \mid M_{i-1} < n \le M_i, K_i \le j < N-K_i\}$$

を定義して、計算 (2) を

$$\begin{aligned} A_j &\approx B_j^{(1)} + B_j^{(2)} + \cdots + B_j^{(k)} + C_j^{(k)} \\ B_j^{(i)} &= (w(x_j)p(x_j))^{-1/4} \sum_{(j,n)\in S^{(i)}} a_n \psi_n(x_j) \\ C_j^{(k)} &= \sum_{(j,n)\notin \cup_i S^{(i)}} a_n \psi_n(x_j) \end{aligned} \tag{7}$$

のように近似するのである。このようにすれば $\cup_i S^{(i)}$ に対して FFT を適用できる。ここで先と同様に長方形 $S^{(i)}$ の取り方を最適化することにより、漸近計算量が $O(N\log^2 N/\log\log N)$ となるという理論的な結果が得られる。

Orszag はこの方法を実装していないようであるが、論文中では CRAY-1 上で行ったいくつかの評価から $O(N^{3/2})$ の方法による Legendre 多項式変換について $N = 128$ 程度で直接法よりも高速になりそうだと述べている。$O(N\log^2 N/\log\log N)$ の方法については、$N = 250$ よりも小さい N では高速になりそうにないとコメントしているだけである。精度については論文に十分な記述がないが、少なくとも単精度程度を想定しているように思われる。

2.2. Orszag の算法の問題点

我々は Orszag の算法を Legendre 多項式変換 (1) に適用しようとした。しかし、次のような問題が生じたため使用に耐えなかった。

精度— 前述の通り、Orszag の論文は精度についてほとんど触れていない。そこで、近似式 (3) でどの程度の近似精度が得られるかを $N = 128$ から $N = 1024$ まで調べた。その結果、この範囲の N に対しては誤差がほとんど 10^{-2} 以上となり、実用精度からは程遠いことが分かった。従って、この範囲の N では WKB 近似はまだ十分な精度で成立していないと見られる。

アルゴリズム— Orszag の算法では、実際に近似が成立している領域 S よりも狭い範囲にしか FFT を適用していないため、近似を十分に生かしきれない。また、(7) のように多くの小領域に対して FFT を適用することにより計算が複雑になり、計算量も多くかかってしまう。さらに、実際に計算するためには $S^{(i)}$ の具体的な設定が必要であるが、Orszag の論文にはこの具体的な方法ついての明確な記述がない。

3. Orszag の算法の問題点の克服

Orszag の算法には上記のような問題があることがわかった。そこで我々は Sturm-Liouville 固有関数から Legendre 多項式を取り上げて Orszag の算法の改良を試みた。本節では改良算法とその結果について報告する。

3.1. 高精度近似式

WKB 近似式では精度が十分ではないので、Legendre 多項式 $P_n(x)$ の高精度近似式で、WKB 近似式のような FFT が適用できるものを調べた。その結

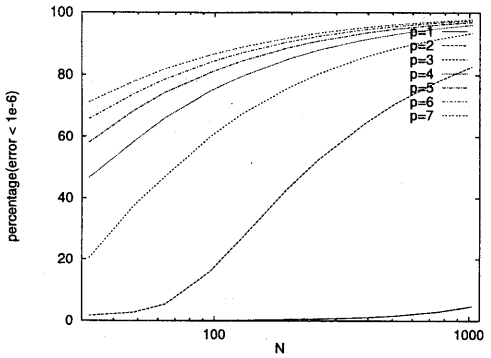

図 1. 各 p での $R_p(n,\theta_j) < 10^{-6}$ となる割合

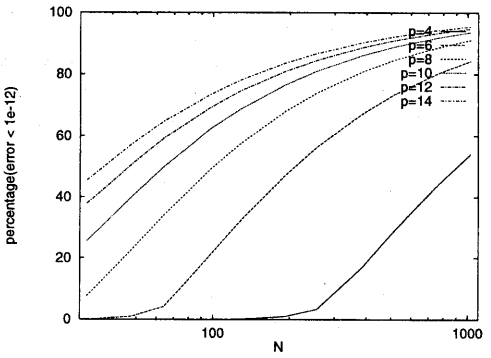

図 2. 各 p での $R_p(n,\theta_j) < 10^{-12}$ となる割合

果、次の Stieltjes による漸近展開式 [1] を見出した。これは近似の次数 p を高めることで精度をあげることができ、また近似誤差もわかっているという特徴がある。

$$P_n^{(p)}(\cos\theta) = \frac{4}{\pi}\sum_{\nu=0}^{p-1}\frac{[(2\nu-1)!!]^2(2n)!!}{(2\nu)!!(2n+2\nu+1)!!} \cdot \frac{\cos\{(n+\nu+1/2)\theta-(\nu+1/2)\pi/2\}}{(2\sin\theta)^{\nu+1/2}} \tag{8}$$

とすると、$0<\theta<\pi$ に対して以下の近似が成立する。

$$P_n(\cos\theta) = P_n^{(p)}(\cos\theta) + O((n\sin\theta)^{-p-1/2})$$

この近似式 $P_n^{(p)}(\cos\theta)$ の具体的な近似度を調べるため、$\theta_j=(j+1/2)\pi/N$ として、p を変化させ、誤差

$$R_p(n,\theta_j) = P_n(\cos\theta_j) - P_n^{(p)}(\cos\theta_j)$$

が 10^{-6} より小さくなる (n,j) の割合を調べた (図 1)。次数 p があがるほど近似がよくなっており、$N\geq 128$ では $p=5$ で 8 割以上の要素が 10^{-6} の精度を持つようになることがわかる。

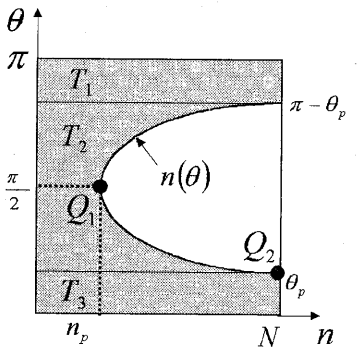

図 3. $R_p(n,\theta_j) > \varepsilon$ となる (n,θ) の範囲

さらに、誤差限界を 10^{-12} にして同様の実験を行った結果が図 2 である。N が小さい場合には p を大きくしてもほとんど精度は上がらないが、$N\geq 128$ では $p=8$ で半分以上の要素が 10^{-12} の精度を持つようになった。このように、Stieltjes の近似公式は大規模な問題に対しては倍精度程度の精度を実現できることがわかる。

3.2. アルゴリズムの改良

Orszag は近似が適用できる範囲を複数の長方形に分割して個別に FFT を適用したが、これは複雑であるし無駄でもある。我々は Orszag の算法を改良して、次のような計算方法を考えた。

$$A_j \approx \sum_{n=0}^{N-1} a_n P_n^{(p)}(\cos\theta_j) + \sum_{R_p(n,\theta_j)>\varepsilon} a_n R_p(n,\theta_j) \tag{9}$$

ここで右辺第 1 項に FFT を適用し、第 2 項は直接計算する。ε は精度を決定する閾値である。このアルゴリズムは FFT を全域に適用することで計算を単純かつ高速にしており、近似のよい範囲に直接計算を持ち込まないため無駄がないのが特徴である。しかも、以下に述べるように漸近的な計算量も $O(N\log N)$ に改善される。

3.3. 提案算法の計算量

次に近似式の次数 p を固定したとき、この算法の計算量が $O(N\log N)$ であることを示す。式 (9) の右辺第 1 項については $O(N\log N)$ であることは自明であるので、右辺第 2 項の計算量、すなわち $R_p(n,\theta_j)>\varepsilon$ になる範囲の面積が $O(N\log N)$ であることを示せばよい。近似式 (8) の誤差が $O((n\sin\theta)^{-p-1/2})$ となることから、ある定数 n_p があって

$$n \geq n_p/\sin\theta$$

を満たす範囲では必ず誤差が閾値以下になることが分かる。従って、これに含まれない、図3で網掛けで示した領域の面積を評価すればよい。

この領域を図のように $T_1+T_2+T_3$ に分けてそれぞれの面積を評価する。曲線 $n(\theta)$ と、その $n=N$ での切片 θ_p はそれぞれ

$$n(\theta)=n_p/\sin\theta,\quad \theta_p=\arcsin(n_p/N)$$

のように計算されるから、それぞれの面積は

$$T_1=T_3=n_pN/\pi$$
$$T_2=(N/\pi)\int_{\theta_p}^{\pi-\theta_p}(n_p/\sin\theta)d\theta$$

のように表わされることになる。T_1 と T_3 は $O(N)$ であり、T_2 の面積も積分を実行すれば $O(N\log N)$ となることは容易に分かる。以上により、全体の計算量は $O(N\log N)$ であることが証明された。

3.4. 近似公式の計算方法

近似式 (8) による変換の計算は FFT によって行うことができる。しかし、この計算内容は通常の FFT そのものではない。本節では今回の実装に用いた算法について述べる。但し、評価点数 N は 2 の冪乗と仮定する。

高速計算が必要なのは、特殊な cosine 変換

$$g_j=\sum_{n=0}^{N-1}G_n\cos\left(\left(n+\nu+\frac{1}{2}\right)\theta_j-\left(\nu+\frac{1}{2}\right)\frac{\pi}{2}\right)\tag{10}$$

である。ここで、評価する点は $\theta_j=(j+1/2)\pi/N$ であるから、

$$\alpha_j=\exp\left(\frac{\pi i}{2N}(2\nu+1)(2j+1-N)\right)$$
$$\gamma_j=\sum_{n=0}^{N-1}G_n\exp\left(\frac{\pi i}{N}n(j+1/2)\right)$$

を定義すると、

$$g_j=\Re[\alpha_j\gamma_j]$$

となる。

ここで $1\le M=2^\mu\le N$, $0\le l<M$, $0\le j<N/M$ に対して

$$\gamma_j^{l/M}=\sum_{m=0}^{N/M-1}G_{Mm+l}\exp\left(\frac{\pi i}{N}(Mm+l)\left(j+\frac{1}{2}\right)\right)$$

を定義する。明らかに

$$\gamma_j^{l/1}=\gamma_j$$
$$\gamma_0^{l/N}=G_l\exp\left(\frac{\pi i}{2N}l\right)$$

である。さらに、簡単な計算により $0\le j<N/(2M)$ に対して

$$\begin{aligned}\gamma_j^{l/M}&=(\gamma_j^{l/(2M)}+\gamma_j^{(l+M)/(2M)})\\ \gamma_{N/M-j-1}^{l/M}&=e^{\pi il/M}(\overline{\gamma_j^{l/(2M)}}-\overline{\gamma_j^{(l+M)/(2M)}})\end{aligned}\tag{11}$$

となることがわかる。(11) を再帰的に用いることにより、$\gamma_0^{l/N}$ から $\gamma_j^{l/1}$ までを計算することができる。

この計算の浮動小数点数演算 (flop) 数は以下のように評価できる。再帰式 (11) の計算は実数演算に還元して 10 flop である。j, l, M の動く範囲を考慮すると、再帰計算部分の flop 数は $5N\log N$ であることがわかる。さらに $\gamma_0^{l/M}$ の計算に $2N$ flop、γ_j から g_j を計算するのに $3N$ flop かかるので、高速変換 (10) 全体は $5N(\log N+1)$ flop で計算されることになる。

この計算に対してこのアルゴリズムが最適であるかどうかはまだわかっていない。入出力の情報がともに N 個の実数値であるのに再帰計算 (11) は複素数を扱っており、実数換算で $2N$ 個のデータ間の演算になっているのは無駄のように思われる。これを N 個の実数データ間の演算に帰着させることができれば、およそ半分の計算量になることが期待される。この問題については今後さらに検討を加えたいと考えている。

3.5. 実装による提案算法の評価

本節では提案算法の実装による評価結果について報告する。サイズ N は 128, 256 および 512 とし、閾値 ε を 10^{-3} から 10^{-14} まで変化させた。それぞれの N, ε に対して p を 3 から 9 まで変化させて計算量・計算時間を計測し、得られた計算量・計算時間の最小値を求めた。計算時間の計測では合計計算時間が約 20 秒になるまで変換計算を繰り返し、変換一回あたりの計算時間を算出した。使用した計算機は、CPU が alpha 21264 の 500 MHz, OS は Digital UNIX, コンパイラとオプションは `gcc -O2 -funroll-loops` である。計算はすべて倍精度で行った。

図4は、提案手法と直接計算の浮動小数点数演算回数を比較したものである。縦軸が精度にあたる閾値 ε の設定値、横軸は演算量 (kflop) で、提案算法の計算量を点でプロットしている。直接法の計算量を示す

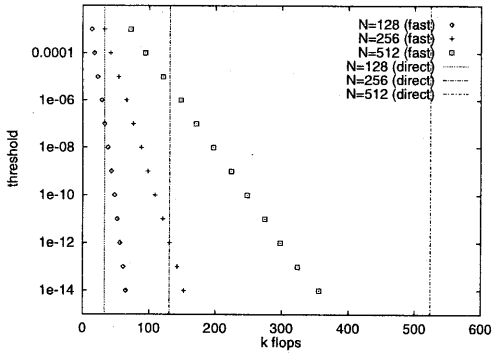

図 4. 直接計算と提案算法の浮動小数点数演算数

縦の直線と比較してみると、$N = 128$ ではおよそ単精度程度の精度となる $\varepsilon = 10^{-7}$ で、$N = 256$ ではおよそ倍精度程度の精度となる $\varepsilon = 10^{-12}$ で、それぞれ直接計算とほぼ同じ計算量となっている。$N = 512$ では $\varepsilon = 10^{-14}$ でも直接計算よりも 30% 程度速い。

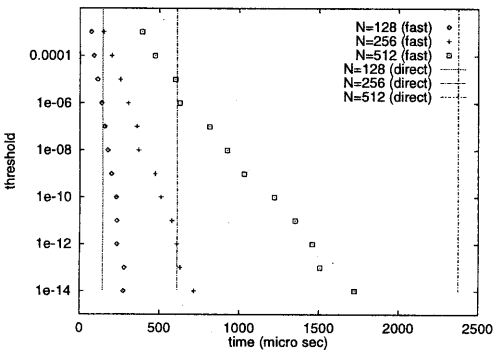

図 5. 直接計算と提案算法の CPU 時間

図 5 は、計算時間を図 4 と同様の方法でプロットしたもので、横軸は所要時間 (μs) である。所要時間は計算量に比べ揺らぎが大きいが、およそ図 4 と同じ結果が得られている。

上記の 2 つのグラフにおいて、計算量·計算時間と精度との関係はほぼ直線、つまり $O(N \log N \log(1/\varepsilon))$ となっているようである。この計算量の内訳を調べてみると以下のようになる。

図 6 は、計算量を最小にする近似次数 p と ε との関係を同様の方法でプロットしたものである。これを見ると、最適な近似次数 p も要求精度 $\log(1/\varepsilon)$ に比例しているように思われる。このことは、近似アルゴリズム (9) の右辺第 1 項の計算量が $\log(1/\varepsilon)$ にほぼ比例していることを示している。

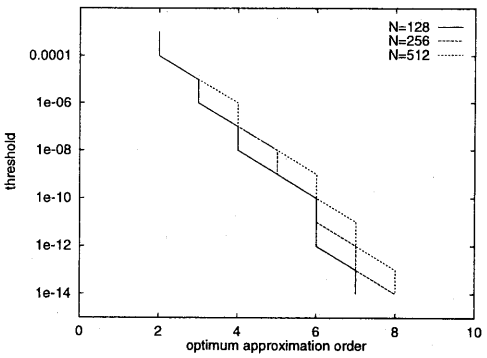

図 6. 計算量最小の近似次数

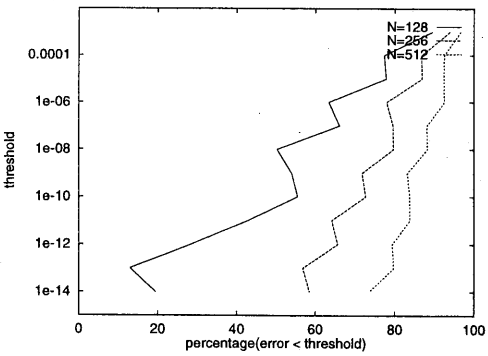

図 7. 誤差が閾値未満となる要素の割合

図 7 は、近似式の誤差が閾値 ε 未満となる (n, j) の割合を ε の関数としてプロットしたものである。N が大きくなるほど近似の程度が良くなることがすぐに見て取れる。グラフがギザギザしているのは近似の次数 p がそれぞれの N, ε で計算量が最小になるものを選んであるためで、p が変化するところでグラフが逆行しているのである。しかし全体的な傾向としては、閾値 ε を小さくするほど近似が成立している範囲が狭くなってゆくことがわかる。すなわち近似アルゴリズム (9) の右辺第 2 項の部分の計算量が増えてゆくのであるが、これもおよそ直線に乗っており、おおよそ $\log(1/\varepsilon)$ に比例しているようである。

このように、アルゴリズム (9) の右辺の両項ともが $\log(1/\varepsilon)$ に比例しているようである。しかし、この現象についての理論的な説明はまだ得られていない。今後検討してゆきたいと考えている。

最後に変換における誤差について報告する。入力ベクトル a_n として $[0, -1)$ の一様乱数を用い、直接計算の結果と提案手法の計算結果との相対誤差を 2 ノルムで評価したところ、およそ ε よりも 1 桁小さい値となった。これにより、提案手法が実際に良好な精

度で安定に計算されていることが確認できた。

Stieltjes による近似式 (8) は分母に $\sin(\theta)^{\nu+1/2}$ という係数を含んでいる。$0 \leq \nu < p$ であるから、次数 p が高く θ が 0 または π に近いところではこの項が非常に大きくなってしまうことが予想される。このため、実装においては計算量を増加させない範囲で安定性が最大になるように、図 3 の T_1 と T_3 にあたる部分については近似式を適用していない。それでも単精度で計算すると $p = 6$ あたりから誤差の増大が見られる。倍精度計算では主記憶容量の限界である $N = 2048$ まで実験を行ったが、特に数値的な不安定性は見られなかった。しかし N が大きくなるほど θ が 0 や π に近いところまで近似が成り立つため、より大きい N では精度に影響を与える可能性もある。

4. まとめと考察

本稿では、Legendre 多項式変換の場合について Orszag の方法を改良した。計算量は $O(N \log N)$ に改善され、単精度程度の精度なら $N \geq 128$ で、倍精度程度の精度なら $N \geq 256$ で直接法よりも高速となることがわかった。また、精度と計算量の関係が $O(N \log N \log(1/\varepsilon))$ となると思われる実験結果が得られた。これにより、実用的な精度と計算速度を持つ高速 Legendre 多項式変換法が実現できた。

本稿の方法は等分点上での関数評価の場合にしか適用できないが、Boyd の方法 [5] と組み合わせることにより他の任意の点での値を評価することができるようになる。この場合でも計算量は $O(N \log N)$ である。また、標本点上での関数値から Legendre 多項式展開の係数を評価することも、提案手法に Gauss 積分と Boyd の方法を組み合わせることによって可能である。この場合も計算量は $O(N \log N)$ である。

本稿と同じような議論は近似公式として WKB 近似 (3) を用いた場合もできるものと思われる。従って、本稿で提案した改良アルゴリズム (9) は WKB 近似を用いた一般の Sturm-Liouville 固有関数変換に対しても Orszag の方法の計算量を改善するものと思われる。しかし実際の速度と精度については、近似式 (8) に対応する高精度近似式が得られるかどうかにかかっている。近似度のよい解析的な式がない場合、最適近似を数値的に計算しそれを近似式に当てることも考えられる。

計算を Legendre 多項式変換に限ると、高速算法は Orszag のもの以外にもいくつか [3, 4, 6, 7, 8, 9] 知られており、なかには計算量が $O(N \log N)$ で本稿で提案した算法よりも高速と思われるものもある。しかし、他の算法に比べかなり簡単な計算で高速計算が実現でき、その考え方が他の Sturm-Liouville 固有関数変換にも適用が可能であるという点において、本算法は他の算法と一線を画するものである。今後は、本算法の考え方の基づく他の高速固有関数変換法の開発や近似式の最適化などについて研究を進めて行く予定である。

謝辞

本研究の一部は日本学術振興会未来開拓学術研究推進事業による。

参考文献

[1] G. Szegö, *Orthogonal Polynomials*, Chap. 8, AMS, New York, 1959.

[2] S. A. Orszag, "Fast Eigenfunction Transforms", in: G. C. Rota ed. *Science and Computers*, Academic Press, pp. 23-30, 1986.

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

[9] 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.