

現代の任意精度浮動小数点演算ライブラリを用いた 悪条件代数方程式の求解

幸谷 智紀^{1,a)}

概要: Chebyshev 積分公式の分点を解として持つ代数方程式は、係数の計算における桁落ちが大きだけでなく、その求解においても次数に応じて必要な桁数が増えることが知られている。この悪条件代数方程式を独自の任意精度計算を用いて小野令美が 1024 次まで解いた結果を 1979 年に報告している。本講演ではこの求解に対して MPFR ライブラリをベースとした MPLAPACK、そして我々の開発した BNCpack を用いてベンチマークテストを行い、その結果について報告する。

キーワード: 代数方程式, 多倍長精度計算, MPFR, MPLAPACK

On solving ill-conditioned algebraic equations using modern arbitrary precision floating-point arithmetic library

Abstract: It is well known that the algebraic equation with their roots as abscissas of Chebyshev integration formula, has coefficients when large loss of significant digits are occurred in the calculation, and that the number of digits required increases with the order. In 1979, Harumi Ono reported the results of solving this algebraic equation up to the 1024th order using an arbitrary precision floating-point arithmetic. In this talk, we will report the results of benchmark tests using the current MPFR library, MPLAPACK, and our BNCpack.

Keywords: Algebraic equation, Multiple precision floating-point arithmetic, MPFR, MPLAPACK

1. 初めに

現状の多倍長精度演算、定評のある QD, MPFR といったライブラリを使用してなされることが多い。こういった 20 世紀末に開発され、20 年以上の使用実績のある高信頼ライブラリの上で、MPLAPACK[7] や我々の BNCpack[5] が成立しており、かつては困難だった悪条件問題も、安々とコンシューマ向けハードウェアで扱えるようになっている。

我々は既に Gauss 型積分公式の分点計算に対して、実三重対角行列の固有値問題として求める方法 (Golub-Welsch 法 [4]) と、代数方程式の零点を求める方法 (山下の方法 [14][16][15]) を適用し、MPFR[9] を用いることで両者どちらでもユーザの要求する精度の分点が得られることを

示した [19]。Golub-Welsch 法では多倍長精度では高速性に難点があるものの丸め誤差に対して頑健であり、低精度でも使用する浮動小数点数の丸め誤差限界程度の近似固有値が得られ、山下の方法では多倍長精度での計算時間は短いものの、収束性を担保するためには良好な初期値を選ぶ必要がある。我々の得た結論は、Binary64 で Golub-Welsch 法を使用し、これで得た近似固有値を山下の方法の初期値として利用し、多倍長精度計算を行って任意の精度の零点を得ることで計算時間の短縮化が図れるというものであった。今の観点では、固有値問題に対する混合精度解法の一つと言える。

今回我々は、この知見に基づき、複素解も含む悪条件の代数方程式に対して、MPLAPACK の実行列用ドライバーである Rgeev と、BNCpack に実装した Durand-Kerner(DK) 法を組み合わせることで、零点計算の高速化が図れるかどうか、数値実験を行った。その例題の一つとして、小野令美が Durand-Kerner 法で実行した Chebyshev

¹ 静岡理科大学
Toyosawa, Fukuroi, Shizuoka 437-8555, Japan
^{a)} kouya.tomonori@sist.ac.jp

積分公式の分点計算 [18][17] を取り上げる。また比較対象として、同じく代数方程式としては悪条件の Wilkinson の例題 [10] も対象とする。

Chebyshev 積分公式の分点を解として持つ代数方程式は、係数の計算における桁落ちが大きだけでなく、その求解においても次数に応じて必要な桁数が増えることが知られている。この悪条件代数方程式を独自の任意精度計算を用いて小野が 1024 次まで解いた結果を 1979 年に、2048 次ものを解いた結果を 1981 年に報告している。また、多倍長精度演算を用いて 20480 次までの係数の計算については益本らが詳細に報告している [13] が、解の導出には至っていない。本講演ではこの代数方程式の求解に対して MPFR ライブラリをベースとした MPLAPACK の Rgeev、そして我々の開発した BNCpack に実装した DK 法を混合精度的に使用することで高速化を図ることができたことを示す。

2. 多倍長精度演算の現状と今後の展開

浮動小数点演算をメインとする計算処理のニーズは、科学技術計算のみならず、深層学習等、増えることはあれ減ることは当面ない。計算処理を行う CPU, GPU といったハードウェアの性能向上は動作周波数の増加よりも

- SIMD 命令の強化、コア数の増加といった並列処理能力の向上
- 半精度・単精度浮動小数点演算の導入・強化といった、処理のニーズに応じた浮動小数点数の仮数部長の増減によって行われるようになっている。ソフトウェア的には、これらのハードウェアの性能向上策を利用し、SIMD 命令ライブラリ、OpenMP, MPI といった並列技法を駆使しつつ、ハードウェアが標準的に備える IEEE 単精度・倍精度 (Binary32, Binary64) と半精度を組み合わせた混合精度技法を用いた性能向上策が取られるようになっている。Binary64 では計算精度が不足する悪条件問題に対して使用される、ソフトウェアライブラリによる多倍長精度浮動小数点演算においても、処理自体が重い分、同様のソフトウェア的性能向上策を積極的に取り入れて行く必要がある。

現状の多倍長精度浮動小数点演算は、無誤差変換技法を用いて Binary32, 64 を複数組み合わせる仮数部長を伸ばすマルチコンポーネント方式と、整数ベースの多数桁方式の 2 種類の実装方式に基づくものが主流である。マルチコンポーネント方式は Bailey らの QD ライブラリが著名であり、多数桁方式では GNU MP(GMP) の任意長自然数カーネル (MPN) を用いた MPFR ライブラリが高速性と信頼性の両方に優れたものとして多数のユーザを抱えている。中田による MPLAPACK はこの 2 つの多倍長精度浮動小数点演算ライブラリの土台の上に開発された LAPACK/BLAS の Fortran コードに基づく多倍長精度線型計算ライブラリであり、2022 年 2 月現在の最新バージョン (Version 1.0.1) では、OpenMP による BLAS コードの並列化に対応し、

LAPACK の主要なドライバルーチンや計算ルーチンを多倍長精度で利用できるようになっている。しかしながら、全てのドライバルーチンがこの高性能 MPBLAS の恩恵を預かるにはまだ時間がかかると思われる。現状では今回使用するドライバルーチン Rgeev は並列化がなされていない。また、任意精度計算で用いている MPFR のラッパー C++ クラスライブラリである mpreal を用いているため、任意精度ルーチン群は、我々が実装した C ネイティブの基本線型計算より低速になることが判明している。

とはいえ、LAPACK において利用頻度の高いドライバルーチンが DD, QD, MPREAL 精度で気軽に利用できるようになったことの恩恵は大きく、MATLAB ベースの Chebfun[3] のような非線形問題を扱うパッケージの多倍長精度化も視野に入ったと言える。特に固有値・特異値計算が手軽に任意精度化できるようになった意義は大きく、例えば非線形方程式の零点計算も、Chebyshev 多項式展開を経由して代数方程式の零点を初期値とした Newton 法を利用する Chebyshev Proxy Rootfinder(CPR)[2] の実装も Rgeev を用いることで任意精度で可能である。これに任意精度 FFT を加えることで、Chebyshev 多項式展開係数も Cosine 変換で効率的に実現できるようになる。

このように、MATLAB の主要機能が多倍長精度で可能になった今、Binary64 では困難だった悪条件な非線形問題も、MPLAPACK の礎の上にヘルパー的な機能を付加しながら、果敢にチャレンジすべきであろう。その第一歩として、悪条件の代数方程式を取り上げ、MPLAPACK だけでは効率の悪い零点計算でも、実装が容易で並列化効率の高い同時反復法を用いることで高速に実行できることを示すのが、本研究の目的である。

3. 代数方程式向けの混合精度解法

我々が今回対象とするのは、実係数多項式 $p_n(x)$

$$p_n(x) = \sum_{i=0}^n a_i x^i \quad (a_i \in \mathbb{R}, a_n \neq 0) \quad (1)$$

を左辺とする n 次代数方程式

$$p_n(x) = 0 \quad (2)$$

である。必ず高々 n 個の解 $\alpha_i \in \mathbb{C}$ に存在することが知られている。今回は方程式 (2) の解について何の情報もないという前提で、複素数解を求めるものとする。また、 $p_n(x)$ を a_n で割った monic な多項式 (3)

$$q_n(x) = x^n + \sum_{i=0}^{n-1} c_i x^i \quad (c_i = a_i/a_n) \quad (3)$$

を $q_n(x)$ とする。

3.1 代数方程式の固有値解法

代数方程式 (2) と同じ解を持つ固有方程式を持つコンパ

ニオン行列 C_n は, $q_n(x)$ の係数を用いて表現でき,

$$C_n = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & \cdots & 0 & 1 \\ -c_0 & -c_1 & \cdots & -c_{n-2} & -c_{n-1} \end{bmatrix} \quad (4)$$

となる. 今回は (1) が実係数であることから, LAPACK[6] では例えば xGEEV ドライバルーチンを用いて C_n の固有値を得ることができる. 条件数が小さく, 対角化可能であり, 重複固有値もない場合は丸め誤差による影響も少なく, 使用する浮動小数点の仮数部桁を多く取る必要もない. これを今回は代数方程式 (2) に対する固有値解法と呼ぶことにする.

3.2 代数方程式の同時反復解法

直接, 5 次以上の代数方程式 (2) の解を得る方法として, オートドックスなものとしては Newton 法およびその関連解法がある. 近年特に高次解法が多数提唱されており, Petkovic はコンパクトに 2012 年までの成果をまとめている [8]. しかしながら, 後述するように, 高次解法になればなるほど一回あたりの計算量が増えること, また, 大域的収束性が保証されているわけでもないことから, ここでは全ての解を求めることができる同時反復解法のうち, 古典的な Durand-Kerner 法の 2 次解法と 3 次解法 [1] のみを使用することにする. なお, 初期値の設定法については小澤 [12] があるので, 比較検討は別の機会に行いたい.

どちらも k 回反復後の近似解を

$$\mathbf{z}_k = [z_1^{(k)} \ z_2^{(k)} \ \dots \ z_n^{(k)}]^T \in \mathbb{C}^n$$

とする時, 漸化式は monic 形式 (3) を用いて下記のようになる.

2 次解法

$$z_i^{(k+1)} := z_i^{(k)} - \frac{q_n(z_i^{(k)})}{\prod_{j=1, j \neq i}^n (z_i^{(k)} - z_j^{(k)})} \quad (5)$$

3 次解法

$$z_i^{(k+1)} := z_i^{(k)} - \frac{p_n(z_i^{(k)})}{1 - \frac{p_n(z_i^{(k)})}{p_n'(z_i^{(k)})} \sum_{\substack{j=1 \\ j \neq i}}^n (z_i^{(k)} - z_j^{(k)})^{-1}} \quad (6)$$

Aberth の初期値は下記のようになる.

$$z_i^{(0)} := -\frac{c_{n-1}}{n} + r \exp \left\{ \left(\frac{2(i-1)\pi}{n} + \frac{3}{2n} \right) i \right\} \quad (7)$$

今回は半径 r として, 非零係数 $n_{nz} \leq n$ を用いて

$$r := \max_{0 \leq i \leq (n-1)} |n_{nz} c_i|^{1/(n-i)}$$

とした.

4. 悪条件代数方程式の例

固有値問題とは異なり, 代数方程式の求解問題は, 解の密度が高いほど, 近似解を代入した際には多項式 $p_n(x)$, $q_n(x)$ の値の誤差が大きくなる. そのため, Danilevsky 法 [11] のように, 行列の固有値問題を, コンパニオン行列に変換して固有方程式の係数を直接求める方法を取ると, 固有値問題としては良条件にも関わらず, 代数方程式の求解問題としては悪条件になってしまい, Binary64 では近似解の精度が得られないことも起こり得る. Householder 変換に原点移動を行った QR 法のように, 安定的に高精度な固有値が求められる高速な固有値解法がある現在では推奨されない方法である理由がここにある.

しかしながら, 係数に Binary64 より長い多倍長精度浮動小数点数が必要な悪条件代数方程式の場合, 求解においても多倍長精度浮動小数点演算が不可欠となり, 行列の固有値ルーチンよりも計算量の少ない代数方程式の求解法の方が高速に実行できる可能性が出てくる. 我々が既に行った Gauss 型積分公式の分点計算 [19] でも, 全てを実対称行列の固有値ルーチンを用いて計算するより, Binary64 で求めた近似固有値を初期値とする Newton 法を実行することで, 計算時間が少なく済む事例が多く見られた. この結果の延長線上で考えると, 実非対称のコンパニオン行列を用いることで, 任意の実係数代数方程式に対しても同様の混合精度技法が計算時間の短縮化に効果のあるケースが見つかることが予想される.

ここではよく知られている Wilkinson の例題 (実数解) と, Chebyshev 積分公式の分点を解とする代数方程式 (複素数解) の 2 つを取り上げ, そのような事例があることをベンチマークテストで示す.

Wilkinson の例題

$\alpha_i = i$ となる $p_n(x) = \prod_{i=1}^n (x - i)$ を展開して与える. 係数の絶対値は $O(n!)$ で大きくなり, 例えば $n = 20$ の時は

$$\begin{aligned} a_0 &= 2432902008176640000 \\ a_1 &= -8752948036761600000 \\ &\vdots \\ a_{19} &= -210 \\ a_{20} &= 1 \end{aligned}$$

となる. 今回使用する $n = 128$ の場合, $a_0 = 3.8562 \cdots \times 10^{215}$ となる. これより n が大きくなると Binary64 および DD, QD 精度演算では扱える範囲を超える. 図 1 に, DD, QD, MPREAL 512bits Rgeev で計算した解を示す. QD でも正常な解が得られていないことが一目瞭然である.

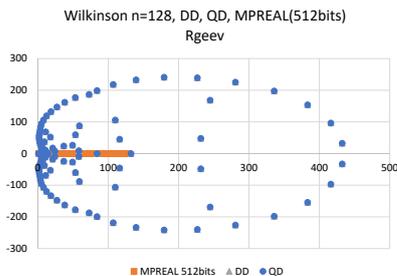


図 1 DD, QD, MPREAL(512bits) Rgeev(MPLAPACK) の結果: Wilkinson の例題 ($n = 128$)

Chebyshev 積分公式から導出された代数方程式

Gauss 型積分公式とは別の系統として Chebyshev の積分公式がある。これは 重みを一定値に固定した場合の積分公式に対して最適な分点を定めるもので、図 2 左図に示すように、 n 個の分点を $n = 8$ および $n > 10$ の時は必ず複素数になることが分かっている。係数は $a_n = 1$ として、以下、次のように求められる。

$$\begin{cases} a_{n-(2k-1)} := 0 \\ a_{n-2k} := -\sum_{j=1}^k a_{n-2(k-j)} \end{cases} \quad (8)$$

ここで、 $k = 1, 2, \dots, \lfloor n/2 \rfloor$ である。

漸化式 (8) を用いて任意次数の係数を求めることは可能だが、次数が大きいくほど桁落ちが増え、正確な係数を求めるには多倍長精度浮動小数点演算が必要となる。益本らの数値実験では $n = 1024$ で定数項 a_0 を 10 進 6 桁以上の有効桁数にするためには 10 進 215 桁以上の歌数部の多倍長精度浮動小数点演算が必要となる。しかし、図 2 右図に示す通り、この解を正確に求めるためには QD 精度の係数でも足りず、より桁数の多い正確な係数が必要となり、10 進 6 桁では不十分である。

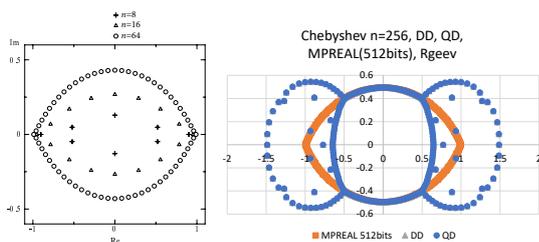


図 2 $n = 8, 16, 64$ の Chebyshev 積分公式の分点 (左), DD, QD, MPREAL(512bits) Rgeev(MPLAPACK) の結果: Chebyshev 積分公式の分点 ($n = 256$, 右)

5. ベンチマークテスト

以上の方針に基づき、与えられた代数方程式 (2) に対して、次のように混合精度計算を利用して求解し、その計算時間 (秒) と得られた解の相対誤差を計測した。

(1) MPREAL(L bits 仮数部) で代数方程式の係数 (a_i , $i = 0, 1, \dots, n$) を求める。

- (2) a_i について MPREAL \rightarrow DD 精度の変換を行い、MPLAPACK の DD 精度 Rgeev を用いて固有値 $\lambda_i^{(DD)} \in \mathbb{C}$ のみ求める。
- (3) 上記で得た固有値 $\lambda_i^{(DD)}$ を初期値 $\mathbf{z}_0 \in \mathbb{C}^n$ として DK 法の 2 次、もしくは 3 次公式を用いて反復を行い、収束値を $\mathbf{z}_{\text{end}} \in \mathbb{C}^n$ とする。

通常、代数方程式の反復解法における収束条件は $|p_n(x)|$ に基づいて行われるが、今回は任意精度計算を行っていることから、 k 回及び $k+1$ 回反復時の近似解 $z_i^{(k)}$, $z_i^{(k+1)}$ と、正の実数として与える相対許容度 ε_{rel} , 絶対許容度 ε_{abs} を用いて

$$|z_i^{(k+1)} - z_i^{(k)}| \leq \varepsilon_{\text{rel}} |z_i^{(k)}| + \varepsilon_{\text{abs}} \quad (9)$$

を満足した時を収束と判断した。相対誤差は、MPREAL の Rgeev で計算した値と比較して算出している。

使用した計算環境は下記の EPYC マシンである。MPLAPACK および我々の BNCpack は GCC でネイティブ環境でコンパイルしたものを使用している。GNU MP(GMP) と MPFR は MPLAPACK に同梱されているものを使用した。

CPU AMD EPYC 7402P (1.5 GHz, 24 cores)

RAM 64 GB

S/W Ubuntu 18.04.5, GCC 7.5.0, MPLAPACK 1.0.1, BNCpack 0.8

並列化は OpenMP で行い、GOMP を用いてコンパイルして実行した。

まず Wilkinson の例題 ($n = 128$) の結果を表 1 に示す。ここで、‘DK(2nd)’, ‘DK(3rd)’ は Aberth の初期値を用いた時の反復回数と各スレッド数の時の計算時間 (秒) を示し、‘2nd+DD’, ‘3rd+DD’ は初期値に DD 精度の Rgeev を用いた時の反復回数と計算時間を示している。反復回数はその下に記述してある現状では MPLAPACK の Rgeev は並列化されておらず、512 bits の MPREAL Rgeev では約 5.2 秒を要する。これに対し、DK 法で十分な精度の近似解を得るため 1024bits 計算し、収束判定には $\varepsilon_{\text{rel}} := 7.5 \times 10^{-145}$, $\varepsilon_{\text{abs}} := 1.0 \times 10^{-300}$ を使用している。この結果、10 進 59 桁から 148 桁まで正しい近似解が得られている。

表 1 Wilkinson の例題: $n = 128$, DK 法 1024bits

	DK(2nd)	DK(3rd)	2nd+DD	3rd+DD
Iter.	1374	691	526-555	508-551
1 Th.	81	72.4	32.1	52.2
2	41.6	36.4	15.6	26
4	20.3	18	7.7	13
8	10.2	9.71	4.13	6.56
16	5.18	4.93	2	3.57
24	3.92	3.45	1.57	2.54

次に、Chebyshev 積分公式の分点問題 ($n = 256$) を解

いた結果を表 2 に示す. MPREAL(256 bits) を用いて係数を (8) 式に基づいて導出し, MPREAL Rgeev では 55.8 秒を要した. DK 法は 512bits 計算を用い, 収束判定には $\varepsilon_{\text{rel}} := 8.6 \times 10^{-68}$, $\varepsilon_{\text{abs}} := 1.0 \times 10^{-300}$ を使用している. この結果, 10 進 32 桁から 43 桁まで正しい近似解が得られている.

表 2 Chebyshev 積分公式の分点: $n = 256$, DK 法 512bits

Iter.	DK(2nd)	DK(3rd)	2nd+DD	3rd+DD
	1344	671	298-328	253-266
1 Th.	206	188	48.5	71.8
2	103	93.3	24	35.1
4	51.5	46.1	11.3	18
8	25.9	25	5.79	10
16	13.3	11.9	3.21	4.91
24	9.51	8.45	2.11	3.34

表 1 と表 2 の結果より, 並列化性能向上率をプロットしたものを図 3 に示す. 2 次解法, 3 次解法, どちらもほぼ理想的な性能向上率を示していることが分かる.

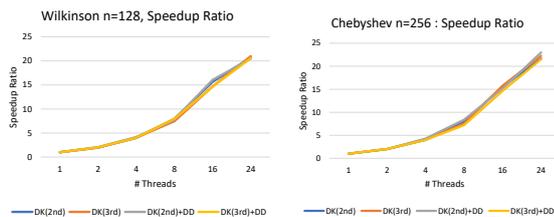


図 3 同時反復解法の並列化効率: Wilkinson の例題 ($n = 128$, 左), Chebyshev 積分公式の分点 ($n = 256$, 右)

固有値問題としては良条件な問題ではあっても, 並列化がなされていない場合は低精度な計算結果を高速に求め, それを初期値とする同次反復解法をより高精度で実行することで高速化が容易になされる例が存在することを, この 2 例では示している. もちろん, 全ての悪条件代数方程式に対して有用であるとは言えないが, 実装が容易で並列化効率の高い同次反復解法の応用範囲はそれなりの広さを持っている可能性がある.

6. 結論と今後の課題

2 例の悪条件代数方程式に対し, 初期値を低精度の固有値解法で求め, 高精度かつ高効率な同次反復解法を並列計算で高速化できることを示した. 反復計算と解の安定性を考えると, 固有値解法のもの的高速化を, 例えば低精度計算による近似固有値を高精度計算時の原点移動量として使用する等の方法が考えられるが, 計算量, 並列性, そして実装の容易さを持つ代数方程式の直接求解法の利用も多倍長精度計算環境では有用なことがあると言える.

従って, 今後の課題としては, 数値計算アルゴリズムの

面からは

- (1) より高次の悪条件代数方程式への適用
- (2) 3 次より高次の解法の適用と計算効率の比較
- (3) Chebyshev Proxy Rootfinder による非線型方程式への適用

が挙げられる. これらをマルチコンポーネント方式の多倍長精度環境と, MPFR ベースの多数桁方式の任意精度計算環境で比較していきたい.

謝辞 本研究の一部は科学技術研究費 20K11843 の助成を利用して行われた. また静岡理科大学研究支援費の支援も受けている. 関係各位に感謝する.

参考文献

- [1] Oliver Aberth. Iteration methods for finding all zeros of a polynomial simultaneously. *Math. Comp.*, 27:339–344, 1973.
- [2] John P. Boyd. Finding the zeros of a univariate equation: Proxy rootfinders, chebyshev interpolation, and the companion matrix. *SIAM Review*, 55(2):375–396, 2013.
- [3] T. A. Driscoll, N. Hale, L. N. Trefethen. *Chebfun Guide*. Pafnuty Publications, 2014.
- [4] G.H.Golub and J.H.Welsch. Calculation of Gauss quadrature rules. *Mathematics of Computations*, 23:221–230, 1969.
- [5] Tomonori Kouya. BNCpack. <https://na-inet.jp/na/bnc/>.
- [6] LAPACK. <http://www.netlib.org/lapack/>.
- [7] MPLAPACK/MPBLAS. Multiple precision arithmetic LAPACK and BLAS. <https://github.com/nakatamaho/mplapack>.
- [8] M.S.Petkovic, Beny Neta, L.D.Petkovic, and J.Dzunic. *Multipoint Methods for Solving Nonlinear Equations*. Elsevier, 2013.
- [9] MPFR Project. The MPFR library. <https://www.mpfr.org/>.
- [10] J. H. Wilkinson. *Rounding Errors in Algebraic Process* (Reprinted Edition). Dover, 1994.
- [11] ファジューエフ, ファジューエバ, 小国 力訳. 線型代数の計算法 (上, 下). 産業図書, 1970.
- [12] 小澤一文. Durand-kerner 法の効率的な初期値の簡単な設定法. *日本応用数学会論文誌*, 3, 1993.
- [13] 益本博幸, 藤野清次, 小野令美, 児島彰. ある 20480 次代数方程式の係数の計算に対する多倍長演算の並列化. *情報処理学会論文誌*, 40(12):4159–4168, 1999.
- [14] 山下真一郎. Gauss の数値積分公式の分点と重率の決定. *情報処理*, 5:206–215, 1964.
- [15] 山下真一郎, 佐竹誠也. Hermite-gauss の数値積分公式の分点と重率の決定. *情報処理*, 5:266–270, 1965.
- [16] 山下真一郎, 佐竹誠也. Laguerre-gauss の数値積分公式の分点と重率の決定. *情報処理*, 4:216–220, 1965.
- [17] 小野令美. Durand-Kerner 法と Aberth 法を用いた超高次方程式の数値計算. *情報処理学会論文誌*, 20(5):399–404, 1979.
- [18] 小野令美. Durand-Kerner-Aberth 法を用いたある種の超高次方程式の解の数値計算. *情報処理学会論文誌*, 22(2):165–168, 1981.
- [19] 幸谷智紀. 実用的な古典的誤差評価法の提案と gauss 型積分公式の分点計算への応用について. *情報処理学会論文誌*, 48(SIG18(ACS20)):1–11, 2007.