数理解析研究所講究録
第1930巻 2015年 26-38

行列Horner法の拡張と効率化

田島 慎一*
SHIN-ICHI TAJIMA
筑波大学大学院 数理物質科学研究科
GRADUATE SCHOOL OF PURE AND APPLIED SCIENCES, UNIVERSITY OF TSUKUBA

小原 功任†
KATSUYOSHI OHARA
金沢大学 理工研究域
FACULTY OF MATHEMATICS AND PHYSICS, KANAZAWA UNIVERSITY

照井 章‡
AKIRA TERUI
筑波大学大学院 数理物質科学研究科
GRADUATE SCHOOL OF PURE AND APPLIED SCIENCES, UNIVERSITY OF TSUKUBA

Abstract

1変数多項式の評価を行う算法として広く用いられているHorner法に対し，1変数多項式に行列を代入して評価するHorner法，すなわち行列多項式に対するHorner法（行列Horner法）の拡張と効率化を提案する．我々が提案する拡張と効率化の特徴は，Horner法を分割して計算することにより，行列の乗算回数を抑える点にある．この方法による効率化は，並列計算を導入せず逐次計算を行った際にもその効果を得ることが可能である．計算機実験では，適切な分割次数のもとでは，計算時間のみならず，メモリ使用量も減少すること，多倍長整数上の行列Horner法のみならず，固定精度（IEEE倍精度）の浮動小数上の行列Horner法でも効果が得られることを示す．

1 はじめに

n 次正方行列 A, n 次正方行列 M もしくは n 次列ベクトル $\boldsymbol{v}$, m 次1変数多項式 $g(\lambda)$ が与えられたときに，行列 $g(A)M$ もしくは列ベクトル $g(A)\boldsymbol{v}$ を効率的に計算する方法について考察する．1変数多項式 $g(\lambda)$ に数値 $\lambda = x_0$ を代入して $g(x_0)$ の値を評価する効率的な算法としてHorner法が古くから知られているが，我々は，数値 x_0 に代えて行列 A を代入して評価するHorner法，すなわち行列多項式に対するHorner法（もしくは行列Horner法）を扱う．

我々は，これまで，レゾルベントの留数解析に基づいて，与えられた行列に対し，スペクトル分解 ([7])，最小消去多項式候補・最小消去多項式やこれらを用いた最小多項式の計算 ([8], [9])，固有ベクトルの計算

*tajima@math.tsukuba.ac.jp
†ohara@air.s.kanazawa-u.ac.jp
‡terui@math.tsukuba.ac.jp

([6]) などを効率的に行う算法を提案しているが，行列多項式の評価は，これらの算法において中心的に用いられる計算の一つであり，行列多項式の評価を効率化することは，算法全体の効率化において重要である．

Horner 法は，もともと逐次的な算法として知られているが，計算機が発達する中で，Horner 法を並列化することにより，複数の演算装置を用いて計算を並列化することによる効率化が提案されてきた ([1], [2], [4], [5]). しかしながら，我々が扱う行列 Horner 法では，行列の乗算が計算コストの主要部を占めるため，行列の乗算回数をいかに減らすかが効率的な算法を作る上での鍵となり，この点において，これまでに提案された，Horner 法の並列化を中心とした効率化とは着想や目的が大きく異なる．また，先行研究の中には，行列多項式に似た形の式を効率的に計算する方法 ([10]) もあるが，この方法は，並列計算によって行列積（もしくはべき乗）の計算を効率化するものであり，Horner 法とは異なるアプローチをとっている．

行列 Horner 法に対し，我々が提案する拡張と効率化の特徴は，Horner 法を分割して計算することにより，行列どうしの乗算回数を抑える点である．与えられた多項式の次数に対し，適切な分割次数を選ぶことにより，計算時間を大幅に抑えることが可能になる．また，実験結果から，適切な分割次数に対しては，メモリ使用量も大幅に抑えられることが示された．これらの計算時間やメモリ使用量の効率化は，並列計算を導入せず逐次計算を行った際にもその効果が得られる点に注意する．

以下，本稿の構成は次の通りである．第 2 章では，Horner 法と行列 Horner 法の概要を述べる．第 3 章では，行列 Horner 法の分割による算法の拡張と効率化を説明する．第 4 章では，計算機実験により，行列 Horner 法の拡張による効果を検証する．実験では，本来我々が計算の目的とする多倍長整数上の行列 Horner 法のみならず，固定精度（IEEE 倍精度）の浮動小数上の行列 Horner 法においても，本稿で提案する算法の効率化による効果が得られることを示す．

2 行列 Horner 法

Horner 法 (Horner's rule) は，1 変数多項式の評価を効率的に行う算法の一つとして知られている ([3]). 以下では K を体とする．Horner 法では，K 上の m 次 1 変数多項式 $f(x)$ に対し，K の元 a を代入した値 $f(a)$ を計算する際の K 上の四則演算の時間計算量は $O(m)$ である（特に乗算の回数が $O(m)$ で与えられることに注意する）．

A, M をそれぞれ K 上の n 次正方行列とし，$g(\lambda)$ を K 上 m 次 1 変数多項式とする．本稿では，$g(A)M$ の Horner 法による計算（行列 Horner 法）を取り上げる．この場合，Horner 法で用いられる乗算（行列どうしの乗算）の回数が $O(m)$ で与えられ，n 次正方行列どうしの乗算の計算量は，素朴な行列演算を用いた場合 $O(n^3)$ ([3]) で与えられることから，$g(A)M$ を行列積による Horner 法で計算した場合の K 上の計算量は $O(mn^3)$ となる．もし $m \simeq n$ の場合，計算量は $O(n^4)$ となる．以下にその計算例を示す．

例 1

A, M をそれぞれ体 K 上の n 次正方行列とし，1 変数多項式 $g(\lambda)$ を

$$g(\lambda) = a_4\lambda^4 + a_3\lambda^3 + a_2\lambda^2 + a_1\lambda + a_0, \quad a_i \in K \tag{1}$$

とする．このとき，$g(A)M$ を行列積による Horner 法で計算すると

$$\begin{aligned} g(A)M &= a_4A^4M + a_3A^3M + a_2A^2M + a_1AM + a_0M \\ &= A(A(A(a_4AM + a_3M) + a_2M) + a_1M) + a_0M \end{aligned}$$

となる．

この計算に現われる行列どうしの乗算の回数は 4 回である．素朴な行列演算を用いた場合，本例の Horner 法の計算量は，上記より $O(4n^3)$ と見積もることができる．∎

3 行列Horner法の拡張による効率化

行列Horner法において，我々が提案する効率化のためのアイデアは，Horner法を分割して計算することにより，行列どうしの乗算回数を抑えることである[1]．まず，このアイデアのアウトラインを例題により示す．

例 2
A, M をそれぞれ体 K 上の n 次正方行列とし，1変数多項式 $g(\lambda)$ を

$$g(\lambda) = a_{18}\lambda^{18} + a_{17}\lambda^{17} + \cdots + a_1\lambda + a_0, \quad a_i \in K$$

とする．前章の行列多項式用Horner法をそのまま用いて $g(A)M$ を計算すると

$$\begin{aligned} g(A)M &= a_{18}A^{18}M + a_{17}A^{17}M + \cdots + a_1AM + a_0M \\ &= A(A(\cdots(A(a_{18}AM + a_{17}M) + \cdots) + a_1M) + a_0M \end{aligned}$$

により計算され，この中に行列どうしの乗算が18回現われる．

一方，"分割した"Horner法による計算では，例えば4次ごとに行列多項式を分割する場合，以下の要領で行う．

[Step 1] $A^4 = (A^2)^2$ を，あらかじめ計算して用意しておく．

[Step 2] あらかじめ A^3M, A^2M, AM を計算し，これらに M を加えた4つの行列を用意しておく．

[Step 3] Horner法を以下の通り 4次ごとに分割して加える．

$$\begin{aligned} g(A)M = A^4\{A^4\{A^4\{A^4(&\underline{a_{18}A^2M + a_{17}AM + a_{16}M})\} \\ &+ (\underline{a_{15}A^3M + a_{14}A^2M + a_{13}AM + a_{12}M})\} \\ &+ (\underline{a_{11}A^3M + a_{10}A^2M + a_9AM + a_8M})\} \\ &+ (\underline{a_7A^3M + a_6A^2M + a_5AM + a_4M})\} \\ &+ (\underline{a_3A^3M + a_2A^2M + a_1AM + a_0M}) \end{aligned} \tag{2}$$

式(2)の各行の下線部の計算には，あらかじめ上のStep 2で用意した A^3M, A^2M, AM, M を用いるため，行列の加算と定数倍の演算しか現われないことに注意する．これにより，上記の各ステップに現われる行列どうしの乗算回数は，Step 1が2回，Step 2が3回，Step 3が4回の合計9回となる．従来のHorner法に対し，行列どうしの乗算回数が半分で抑えられていることに注意する．∎

一般の場合，n 次正方行列 A, M および m 次多項式 $g(\lambda) = a_m\lambda^m + a_{m-1}\lambda^{m-1} + \cdots + a_1\lambda + a_0$ に対し，$d = 2^b$ 次（ただし $d \leq m$）ごとに分割したHorner法により，$g(A)M$ の計算を行う手順は，アルゴリズム1の通り表される．

アルゴリズム 1 (行列Horner法)
入力
- A: n 次正方行列，
- M: n 次正方行列もしくは n 次列ベクトル，
- $g(\lambda)$: m 次1変数多項式 $g(\lambda) = a_m\lambda^m + a_{m-1}\lambda^{m-1} + \cdots + a_1\lambda + a_0$，

[1]本稿では，計算の効率化の上での指標として，行列どうしの乗算回数に着目する．我々が本来意図する，多倍長整数上の行列の計算では，多倍長演算の計算量も考慮する必要があるが，この点の考察は今後の課題である．第5章も参照．

- $d=2^b$: 分割次数(ただし $d \leq m$) .

出力 $g(A)M$: n 次正方行列(M が行列の場合) もしくは n 次列ベクトル(M がベクトルの場合) .

[Step 1] $A^d=A^{2^b}=(\cdots(A^2)^2\cdots)^2$ を, あらかじめ計算して用意しておく.

[Step 2] あらかじめ $A^{d-1}M, A^{d-2}M, \ldots, AM$ を計算し, これらに M を加えた d 個の行列を用意しておく.

[Step 3] Horner 法を以下の通り d 次ごとに分割して加える.

$$\begin{aligned} g(A)M = A^d\{\cdots\{A^d(a_mA^rM+\cdots+a_{qd+1}AM+a_{qd}M)\} \\ +(a_{qd-1}A^{d-1}M+\cdots+a_{(q-1)d+1}AM+a_{(q-1)d}M)\} \\ +\cdots \\ +(a_{d-1}A^{d-1}M+\cdots+a_1AM+a_0M), \end{aligned}$$

ここに q および r はそれぞれ m を d で割った商および剰余を表す. ∎

アルゴリズム 1 の各ステップに現われる行列どうしの乗算回数を b, d, m を用いて表すと, Step 1 で b 回, Step 2 で $d-1$ 回, Step 3 で $\lfloor m/d \rfloor$ 回より, 合計すると

$$b+d+\lfloor m/d \rfloor -1 \tag{3}$$

となる. $d=2^b$ を用いて行列どうしの乗算回数を $T(b,m)$ で表すと, (3) より

$$T(b,m)=b+2^b+\lfloor m/2^b \rfloor -1 \tag{4}$$

と表される.

3.1 分割次数 d の最適値の見積もり

行列 Horner 法の分割次数 d の最適値は, 以下の通り 見積もることができる.

ここでは, アルゴリズム 1 に現われる行列の乗算回数のうち, Step 1 を除いた, Horner 法にかかるものの回数のみを考慮する [2]. このときの行列の乗算回数は, (3) より, およそ

$$\frac{m}{d}+d-1 \tag{5}$$

と見積もることができる. 多項式の次数 m を固定した場合の (5) の最小値は, m/d と d の相加・相乗平均の関係を用いると, m/d と d が等しいときに (5) が最小値をとる. 方程式 $m/d=d$ を d について解くと, $d=\sqrt{m}$ を得る. $d=2^b$ より, d の最適値は, $\sqrt{m}$ を越えない 2^b の最大値か, あるいは $\sqrt{m}$ に最も近い 2^b の値と見積もることができる.

---

[2] 我々が意図している行列 Horner 法の応用(最小消去多項式候補や最小消去多項式等の計算) では, A^d の計算を 1 回だけ行うのに対し, 同じ多項式に対する行列 Horner 法の計算を, 与えられた行列の次元程度(特性多項式の既約因子の次数程度) の回数繰り返す. よって, 全計算時間に占める A^d の計算時間は十分小さいとみなす.

4 実験

本章では，前章に述べた行列 Horner 法の効率化の効果を確かめるために行った実験の結果を示す．

本章の各実験において，A, M を n 次正方行列，$\boldsymbol{v}$ を n 次列ベクトル，$g(\lambda)$ を m 次 1 変数多項式とし，$g(A)M$（もしくは $g(A)\boldsymbol{v}$）を行列多項式に対する Horner 法で計算した．我々が提案した算法の計算量は，上述のとおり，Horner 法の分割次数に依存するため，各実験で与えられた問題に対し，Horner 法の分割次数を変えて行列多項式の評価を行い，計算時間と使用したメモリ容量を比較した．

各算法の実装は数式処理システム Risa/Asir 上で行い，実験を行った．実験環境は以下の通り：Quadcore AMD Opteron 2350 at 2.0 GHz, RAM 4GB, Linux 2.6.26-amd64.

4.1 実験 1: 行列どうしの積の Horner 法の場合

本実験は，以下の要領で行った．

- $g(\lambda)$ の次数は 64 次 ($m = 64$) とし，係数は長さ 64 ビットの整数で無作為に与えた．
- 行列 A, M は 64 次正方行列 ($n = 64$) とし，各要素は長さ 64 ビットの整数で無作為に与えた．
- 以上の A, M, $g(\lambda)$ に対し，$g(A)M$ を行列 Horner 法で計算し，計算時間とメモリ使用量を測定した．
  - Horner 法の分割次数 d は 1（なし），2, 4, 8, 16, 32 と変化させた．
  - 行列 A として異なるものを 10 個用意し，$g(A)M$ の計算を，各分割数あたり 1 回ずつ，計 10 回行い，計算時間とメモリ使用量の平均値を計算した．これらの計算において，行列 M は同じものを用いた．

実験結果のうち，計算時間を表 1 および図 1 に示す．表 1 において，d は Horner 法の分割次数を表す．“Time (Horner)”, “Time (power)” および “Time (total)” は，それぞれ，Horner 法の計算時間，A^d の計算時間および両者の合計を秒単位で表したものである．図 1 は表 1 を棒グラフで表したものである．横軸は計算時間（秒）を表し，縦軸は Horner 法の分割次数 d を表す．グラフのうち，塗りつぶしの部分および網かけの部分は，それぞれ表 1 の “Time (Horner)”, “Time (power)” を表す．

実験結果より，計算時間は $d = 8 = \sqrt{64}$ のときに最も小さいが，Horner 法のみの計算時間は $d = 16$ のときに最も小さく，理論的な行列の乗算回数の最適値の見積もりである $d = 8$（第 3.1 節参照）よりも大きな値で計算時間が最も小さくなっている．

次に，メモリ使用量を表 2 および図 2 に示す．表 2 において，d は表 1 と同じく Horner 法の分割次数を表す．“Memory (Horner)”, “Memory (power)” および “Memory (total)” は，それぞれ，Horner 法の計算，A^d の計算で使用したメモリの容量，および両者の合計をバイト単位で表したものである．これらの数値において “aeb” (a は実数，b は整数) は $a \times 10^b$ を表す．図 2 は表 2 を棒グラフで表したものである．横軸はメモリ使用量（バイト）を表し，縦軸は Horner 法の分割次数 d を表す．グラフのうち，塗りつぶしの部分および網かけの部分は，それぞれ表 2 の “Memory (Horner)”, “Memory (power)” を表す．

メモリ使用量は，d が 1 から 8 まで d に反比例して減少していることがわかる．

4.2 実験 2: 行列の次元と多項式の次数を大きくした場合

本実験では，実験 1 と比較して，より高次元の行列，および，より次数の大きな多項式を用いた．

- $g(\lambda)$ の次数は 128 次 ($m = 128$) とし，係数は長さ 64 ビットの整数で無作為に与えた．

d	Time (Horner)	Time (Power)	Time (total)
1	33.32	N/A	33.32
2	20.47	0.11	20.58
4	14.14	0.29	14.43
8	11.75	0.53	12.28
16	11.14	2.09	13.23
32	17.60	2.39	19.99

表 1: 実験 1 の計算時間. 詳細は第 4.1 章を参照.

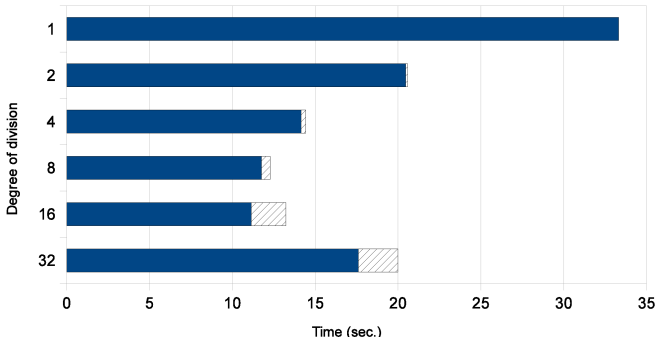

図 1: 表 1 のグラフ. 詳細は第 4.1 章を参照.

d	Memory (Horner)	Memory (Power)	Memory (total)
1	1.19e10	N/A	1.19e10
2	6.08e9	3.03e7	6.11e9
4	3.28e9	7.00e7	3.35e9
8	2.04e9	1.28e8	2.16e9
16	1.99e9	2.28e8	2.22e9
32	3.80e9	4.07e8	4.20e9

表 2: 実験 1 のメモリ 使用量. 詳細は第 4.1 章を参照.

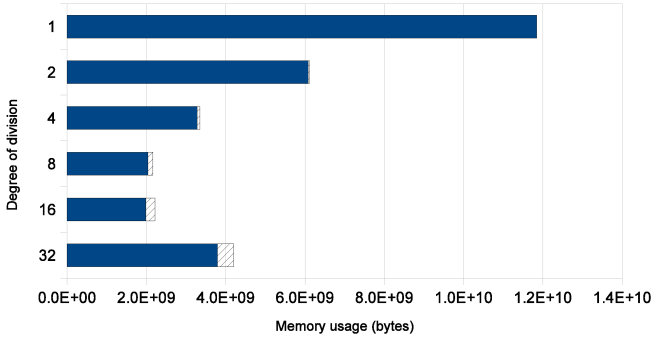

図 2: 表 2 のグラフ. 詳細は第 4.1 章を参照.

- 行列 A, M は 128 次正方行列 ($n = 128$) とし，各要素は長さ 64 ビット の整数で無作為に与えた．
- 以上の A, M, $g(\lambda)$ に対し，$g(A)M$ を行列 Horner 法で計算し，計算時間とメモリ 使用量を測定した．
  - Horner 法の分割次数 d は 1(なし)，2, 4, 8, 16 と変化させた．d が 32, 64 の場合の実験も 試みたが，それまでの実験と 比較して計算時間が大幅に増加したので，計算を途中で打ち切った．
  - 行列 A として異なるものを 10 個用意し，$g(A)M$ の計算を，各分割数あたり 1 回ずつ，計 10 回行い，計算時間とメモリ 使用量の平均値を計算した．これらの計算において，行列 M は同じものを用いた．

実験結果のう ち，計算時間を表 3 および図 3 に示し，メモリ 使用量を表 4 および図 4 に示す．以下，表およびグラフの要領は実験 1 の場合と 同様である．Horner 法のみの計算時間とメモリ 使用量の，分割次数 d に対する減少の様子は実験 1 の場合と 同様である．

行列のべき 乗と Horner 法を合わせた計算時間とメモリ 使用量の合計を見ると，実験 1 では，d が多項式の次数の平方根，すなわち $\sqrt{m} = \sqrt{64} = 8$ を越えた値 ($d = 16$) において，$d = 8$ の場合から 増加に転じているのに対し，実験 2 では，$d = 16$ の場合の方が小さく なっている．実験 2 では，行列のべき 乗の計算が全体の計算に占める 割合が，実験 1 の場合に比べて小さく なっており，Horner 法の(時間／空間) 計算量が全体の計算により 大きな 影響を与えていることがわかる．

4.3 実験 3: 行列・ベクトル積の Horner 法の場合

上の実験では，行列どうしの積の Horner 法を扱ったが，本実験では，行列・ ベクト ル積の Horner 法を取り 上げた．実験の要領は以下の通り である．

- $g(\lambda)$ の次数は 64 次 ($m = 64$) とし，係数は長さ 64 ビット の整数で無作為に与えた．
- 行列 A は 64 次正方行列，ベクト ル $\boldsymbol{v}$ は 64 次 ($n = 64$) とし，各要素は長さ 64 ビット の整数で無作為に与えた．
- 以上の A, $\boldsymbol{v}$, $g(\lambda)$ に対し，$g(A)\boldsymbol{v}$ を行列 Horner 法で計算し，計算時間とメモリ 使用量を測定した．
  - Horner 法の分割次数 d は 1(なし)，2, 4, 8, 16, 32 と変化させた．
  - 行列 A として異なるものを 10 個用意し，$g(A)\boldsymbol{v}$ の計算を，各分割数あたり 1 回ずつ，計 10 回行い，計算時間とメモリ 使用量の平均値を計算した．これらの計算において，ベクト ル $\boldsymbol{v}$ は同じものを用いた．

実験結果のう ち，計算時間を表 5 および図 5 に示し，メモリ 使用量を表 6 および図 6 に示す．本実験では行列・ ベクト ル積の計算が中心で，計算時間，メモリ 使用量とも，Horner 法が全体の計算に占める 割合は，行列どうしの乗算による Horner 法(実験 1, 2) に比べて小さい．逆に，Horner 法を分割した際に計算する行列 A のべき 乗が全体の計算に占める 割合が，分割次数 d が増える程，増加している．

分割次数が $d = 2^b$ から $2d = 2^{b+1}$ に増加するとき，A^{2d} を計算する 際のメモリ 使用量と 計算時間は，A^d の場合のそれらと 比較して，それぞれ以下の通り 見積もることができる．

- A^{2d} の要素は A^d の要素どうしの積をとるため，A^{2d} の要素の大きさは A^d の要素の大きさのほぼ 2 乗，すなわち各要素の(計算機上の) ワード 長は 2 倍程度と 見積もることができる．

d	Time (Horner)	Time (Power)	Time (total)
1	1059.75	N/A	1059.754
2	646.45	0.95	647.40
4	435.01	3.34	438.35
8	343.27	6.49	349.76
16	283.83	11.75	295.58

表 3: 実験 2 の計算時間．詳細は第 4.2 章を参照．

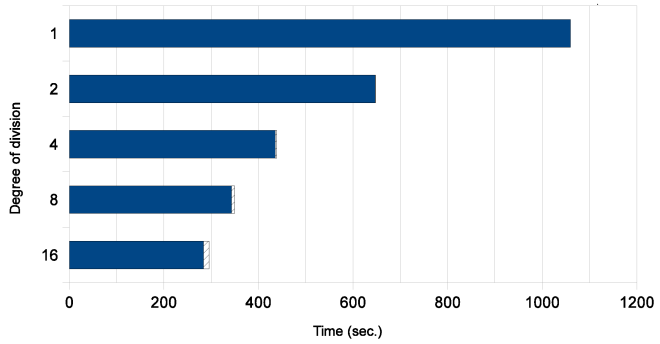

図 3: 表 3 のグラフ．詳細は第 4.2 章を参照．

d	Memory (Horner)	Memory (Power)	Memory (total)
1	3.93e11	N/A	3.93e11
2	1.99e11	2.44e8	1.99e11
4	1.03e11	5.69e8	1.03e11
8	5.48e10	1.04e9	5.59e10
16	3.55e10	1.90e9	3.73e10

表 4: 実験 2 のメモリ 使用量．詳細は第 4.2 章を参照．

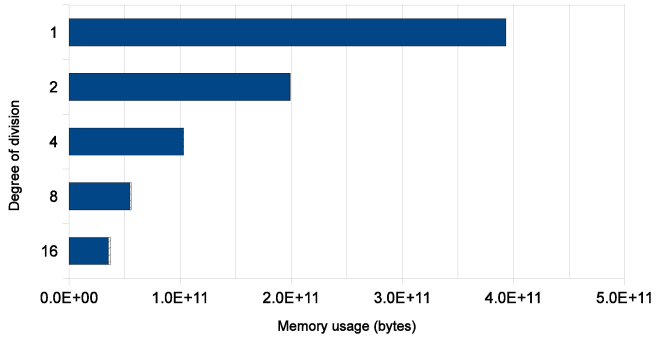

図 4: 表 4 のグラフ．詳細は第 4.2 章を参照．

- 多倍長整数どうしの素朴な乗算の場合，乗数および被乗数のワード長が等しいとすると，計算時間はそれらのワード長の2乗に比例する ([3]). よって，メモリ使用量の見積もりにより，A^{2d} の計算時間は A^d の計算時間の4倍程度と見積もることができる.

さて，表5および図5より計算時間を見ると，Horner法の計算時間は，$d=4$ で $d=1$ の場合の半分程度になり，以後，$d=8,16,32$ においてほとんど変化がない．A^d の計算時間は，分割次数の増加に対応して増加しているが、増加の程度は上述の見積もりよりも小さいと見ることができる.

表6および図6よりメモリ使用量を見ると，Horner法の計算におけるメモリ使用量は，分割次数 d が1から8まで，d に反比例して減少している．一方で，A^d の計算におけるメモリ使用量は，上述の見積もりにほぼ従って増加していると見ることができる.

4.4 実験4: 浮動小数を要素とする行列を用いた場合

これまでの実験では，整数行列やベクトルのHorner法を扱ったが，本実験では，浮動小数を要素とする行列どうしの乗算によるHorner法を取り上げ，浮動小数に対しても我々が提案する算法が有効であることを示す．実験の要領は以下の通りである.

- $g(\lambda)$ の次数は128次 ($m=128$) とし，係数はIEEE倍精度浮動小数(仮数部52ビット，指数部11ビット，符号部1ビット) で無作為に与えた.
- 行列 A, M は64次正方行列 ($n=64$) とし，各要素はIEEE倍精度浮動小数で無作為に与えた.
- 以上の A, M, $g(\lambda)$ に対し，$g(A)M$ を行列Horner法で計算し，計算時間とメモリ使用量を測定した.
  - Horner法の分割次数 d は1(なし)，2, 4, 8, 16, 32, 64と変化させた.
  - 行列 A として異なるものを10個用意し，$g(A)M$ の計算を，各分割数あたり1回ずつ，計10回行い，計算時間とメモリ使用量の平均値を計算した．これらの計算において，行列 M は同じものを用いた.

実験結果のうち，計算時間を表7および図7に示し，メモリ使用量を表8および図8に示す．今回用いた浮動小数は固定精度であるため，分割次数 $d=2^b$ に対し，行列 A^d の計算では，計算時間，メモリ使用量とも b にほぼ比例して増加していることがわかる．一方，Horner法の計算では，計算時間，メモリ使用量とも，$d=8$ においてほぼ最小値をとっており，理論的な行列の乗算回数の見積もり (4) にほぼ従っていることがわかる.

5 まとめ

本稿では，行列Horner法の計算において，Horner法を分割して計算する拡張により，計算を効率化する方法を提案した．計算機実験では，以下の知見が得られた.

- 我々が提案した効率化法は，特に行列どうしの積によるHorner法の計算において効果を発揮した．さらに，分割次数が適切な範囲内では，計算時間のみならず，メモリ使用量も減少することがわかった(実験1, 2).
- 行列・ベクトル積のHorner法を分割した場合は，分割次数の増加に伴い，行列のべき乗を計算するコストは増加するが，Horner法自体の計算に関しては，分割次数を適切に与えると，計算時間，メモリ使用量とも減少することがわかった.(実験3).

d	Time (Horner)	Time (Power)	Time (total)
1	0.47	N/A	0.47
2	0.32	0.12	0.44
4	0.22	0.28	0.50
8	0.21	0.52	0.73
16	0.21	1.06	1.27
32	0.26	2.31	2.57

表 5: 実験 3 の計算時間．詳細は第 4.3 章を参照．

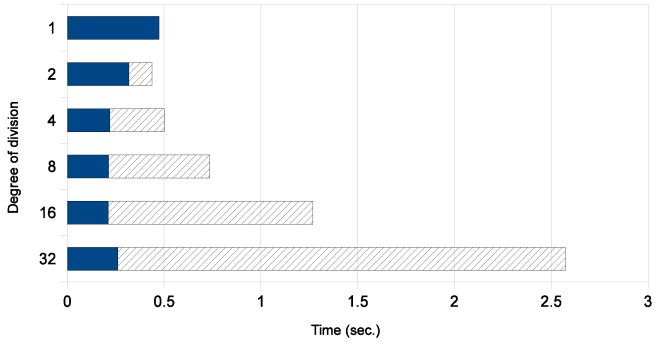

図 5: 表 5 のグラフ．詳細は第 4.3 章を参照．

d	Memory (Horner)	Memory (Power)	Memory (total)
1	1.91e8	N/A	1.91e8
2	9.77e7	3.03e7	1.28e8
4	5.26e7	7.00e7	1.23e8
8	3.23e7	1.28e8	1.61e8
16	3.05e7	2.28e8	2.59e8
32	5.94e7	4.07e8	4.67e8

表 6: 実験 3 のメモリ 使用量．詳細は第 4.3 章を参照．

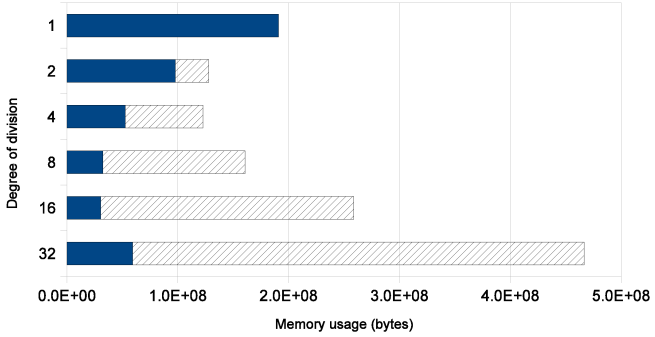

図 6: 表 6 のグラフ．詳細は第 4.3 章を参照．

d	Time (Horner)	Time (Power)	Time (total)
1	46.85	N/A	46.85
2	25.81	0.34	26.16
4	14.33	0.81	15.14
8	10.01	1.31	11.32
16	10.96	1.76	12.72
32	16.85	2.12	18.96
64	30.67	2.44	33.11

表 7: 実験 4 の計算時間．詳細は第 4.4 章を参照．

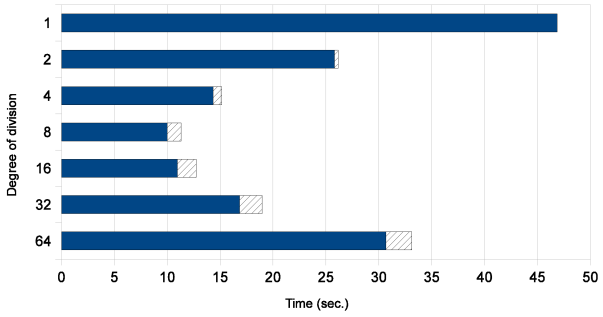

図 7: 表 7 のグラフ．詳細は第 4.4 章を参照．

d	Memory (Horner)	Memory (Power)	Memory (total)
1	1.30e10	N/A	1.30e10
2	6.68e9	1.00e8	6.78e9
4	3.66e9	2.01e8	3.86e9
8	2.46e9	3.01e8	2.76e9
16	2.46e9	4.02e8	2.86e9
32	3.66e9	5.02e8	4.16e9
64	6.68e9	6.03e8	7.28e9

表 8: 実験 4 のメモリ 使用量．詳細は第 4.4 章を参照．

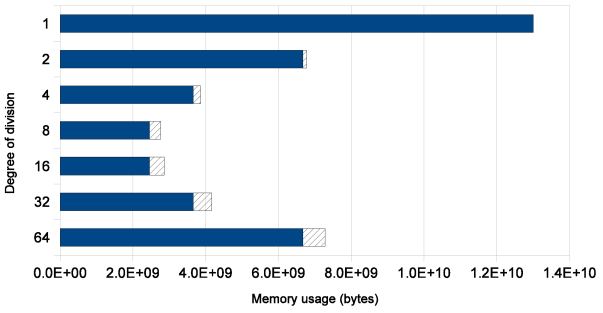

図 8: 表 8 のグラフ．詳細は第 4.4 章を参照．

- 我々が提案した効率化法は，多倍長整数上の行列 Horner 法のみならず，固定精度（実験では IEEE 倍精度）の浮動小数上の行列 Horner 法でも効果が得られた（実験 4）．

行列・ベクトル積の Horner 法の実験結果（実験 3）は，一見，分割次数が増加するほど行列のべき乗を計算するコストが増加しており，算法全体が効率的でなくなるように見える．しかしながら，我々が今後，本算法を，最小消去多項式候補や最小消去多項式の計算に応用する際には，以下の理由により，本算法の効果がもたらされることを強調したい．

最小消去多項式候補や最小消去多項式の計算においては，与えられた行列 A と，算法の中で計算する多項式 $g(\lambda)$ に対し，ベクトル $\boldsymbol{v}$ をいくつも変えて $g(A)\boldsymbol{v}$ を計算する．そのため，行列 A のべき乗は一度計算して使い回せばよいので，A^d を計算するコストは 1 回分しかかからないのに対し，$\boldsymbol{v}$ は，$g(A)\boldsymbol{v}$ の計算の都度取り替える．よって，計算全体で用いる $\boldsymbol{v}$ の個数が多ければ多いほど，A^d を計算するコストが全体の計算コストに占める割合は小さくなり，Horner 法の部分の効率化が，計算全体の効率化に寄与するのである．

我々が本稿で行った計算機実験の対象は，行列の次元，多項式の次数ともに数十次の “中程度” の大きさの問題にとどまっているが，我々が提案する効率化は，行列の次元，多項式の次数ともに数百次から数千次のような大きな問題に対して，より効果を発揮すると考えている．よって，より大きな問題に対して計算機実験を行い，その効果を検証することは，今後の課題である．また，行列 Horner 法の分割次数 d は，現在のところ，2 のべき乗で与えているが，より大きな問題に対し，最も効率的に計算が行われるような d の値（2 のべき乗で十分か，あるいは 2^b と 2^{b+1} の間により適した分割次数が存在するか）を決定することも，今後の課題である．

今後は，本稿で提案した算法に対し，並列計算等，実装による計算のさらなる効率化を行いたいと考えている．計算の効率化を評価する上では，適切な計算量の導入が重要である．本稿では，単純な行列の乗算回数を基準に計算効率の評価を行い，実験でも理論的見積もりにほぼ従った効果が見られたが，我々が意図する行列 Horner 法の応用（最小消去多項式候補や最小消去多項式等の計算）では，多倍長整数上の Horner 法を念頭に置いている．よって，多倍長演算も含めた計算量の評価を行い，それらに適した効率化を検討することも，今後の課題である．さらに，我々が行ってきた行列の最小消去多項式や固有ベクトルの算法に応用し，それぞれの算法の効率化を行っていきたいと考えている．

参 考 文 献

[1] W. S. Dorn. Generalizations of Horner's Rule for Polynomial Evaluation. *IBM Journal of Research and Development*, Vol. 6, No. 2, pp. 239–245, April 1962.

[2] Michael L. Dowling. A Fast Parallel Horner Algorithm. *SIAM Journal on Computing*, Vol. 19, No. 1, pp. 133–142, February 1990.

[3] D. Knuth. *The Art of Computer Programming*, Vol. 2: Seminumerical Algorithms. Addison-Wesley, Third edition, 1998.

[4] Kiyoshi Maruyama. On the Parallel Evaluation of Polynomials. *IEEE Transactions on Computers*, Vol. C-22, No. 1, pp. 2–5, January 1973.

[5] Ian Munro and Michael Paterson. Optimal algorithms for parallel polynomial evaluation. *Journal of Computer and System Sciences*, Vol. 7, No. 2, pp. 189–198, April 1973.

[6] 照井章，田島慎一．行列の最小消去多項式候補を利用した固有ベクトル計算．In *Computer Algebra*

— *Design of Algorithms, Implementations and Applications*, 数理解析研究所講究録, 第 1815 巻, pp. 13–22. 京都大学数理解析研究所, October 2012.

[7] 田島慎一. 微分作用素を用いたレゾルベントの留数解析と行列のスペクトル分解. In *Computer Algebra — Design of Algorithms, Implementations and Applications*, 数理解析研究所講究録, 第 1814 巻, pp. 17–28. 京都大学数理解析研究所, October 2012.

[8] 田島慎一, 奈良洸平. 最小消去多項式候補とその応用. In *Computer Algebra — Design of Algorithms, Implementations and Applications*, 数理解析研究所講究録, 第 1815 巻, pp. 1–12. 京都大学数理解析研究所, October 2012.

[9] 田島慎一, 奈良洸平, 小原功任. 行列の最小多項式計算について. In *Computer Algebra — Design of Algorithms, Implementations and Applications*, 数理解析研究所講究録, 第 1814 巻, pp. 1–8. 京都大学数理解析研究所, October 2012.

[10] 丸山清, David J. Kuck. 行列の式の並列計算について. 情報処理, Vol. 15, No. 5, pp. 335–341, 1974.