第5章　計算量について

> それではあんまりだということで，二十年[a]ほど前から“計算複雑度 (computational complexity)”あるいは “計算量”の理論というのが登場した。(略)「この問題をこのプログラムで得にはこのくらいの手間がかかる」というような形の議論をきちんとやろうというものである。この計算複雑度の理論の中では数値計算の技術に直接関係するような話題が多数取り上げられ，それらに対して，少なくとも理論的には，大変興味ある成果が多く得られている。そうなれば，数値計算に関心のある者としては，そのような成果のうち重要なものは常識として知っていなければならないということになる。
>
> 伊里正夫・藤野和建「数値計算の常識」(共立出版)

[a] この本の初版は 1985 年であるから，それよりも 20 年前という意味であろう。

現在のコンピュータは bit 単位の論理演算を多数組み合わせることで，複雑な浮動小数点演算を実現している。そして，ユーザが直接扱うことの出来る機械語命令は

- byte(or word) 単位の論理演算及び bit 列操作
- 符号付き (signed) もしくは 符号なし (unsigned) 整数演算
- 浮動小数点数演算（四則演算，初等関数）

に大別される [45]。数値計算のアルゴリズムの優劣を論じる時，一つの指標として演算回数の大小がよく使用される。特に 1CPU の PC や WS で，アルゴリズムを逐次的に実行する場合は計算時間の増大を演算回数の order で予測することがある程度は可能である。本章では，数値計算で重要な役割を果たす浮動小数点演算に焦点を絞って，以下の章で紹介する各種アルゴリズムの計算量を考察する基礎を提供する。

5.1　浮動小数点数の四則演算と Landau の O 記号

小学校で習う小数の加減乗除は整数のそれと本質的には同じものである。人間の感覚で言う「面倒くさい」計算は，そのままコンピュータについても当てはまる。面倒な計算は時間を要する。従って四則演算は前章のベンチマークテストからも分かるように，

$$T(\text{加算 (FADD)}) = T(\text{減算 (FSUB)}) \leq T(\text{乗算 (FMUL)}) < T(\text{除算 (FDIV)}) \tag{5.1}$$

という順に計算時間を要すると考えて良い。後で述べる初等関数はこれらの演算を組み合わせて実行されるため，更に時間を要するのが普通である[1]。

従って，数値計算のアルゴリズムの計算時間は加減算の実行回数以上に，乗除算や初等関数の実行回数に左右される。「アルゴリズムの演算回数」という言葉がしばしば後者の意味で使われるのはそのためである。

計算回数に限らず，様々な場面で使用される言葉としてオーダ (order) がある。これは以下に示す Landau の O(ラージオー) と同義である。

定義 5.1.1 (Landau の O 記号)
ある一変数実関数 $f(x), g(x) \in \mathbb{R}$ に対して， $f(x) = O(g(x))$ とは

$$\lim_{x \to \alpha} \frac{f(x)}{g(x)} = \text{定数} \ (\neq 0)$$

となることを意味し，このような $f(x)$ は $g(x)$ のオーダ (order) であると呼ぶ。この $O(g(x))$ を Landau の O(ラージオー) 記号という。α としては 0 もしくは $\pm\infty$ がよく使用される。

また

$$\lim_{x \to \alpha} \frac{f(x)}{g(x)} = 0$$

であるときは特に $f(x) = o(g(x))$ と書き，これを o(スモールオー) 記号と呼ぶ。

以降，オーダという言葉は O(ラージオー) の意味で使用する。$g(x)$ としてよく使用されるのは x の多項式であり，特に $x^2, x^3, ...x^n$ である。$O(x^n)$ はほぼ x^n に比例していることを表しており，直感的に理解しやすいため，様々な場面で使用される。

5.2　複素数の四則演算

本書は実数の演算が主体であるが，複素数の演算が必要となる場面に遭遇することもある。ここで復習も兼ねて，複素数の演算とその演算量について若干の考察を行う。

任意の複素数 $c = Re(c) + Im(c)\sqrt{-1} \in \mathbb{C}$ は 2 つの実数の組 $(Re(c), Im(c))$ として表現できる。従って，複素数の四則演算は全て実数のそれを組み合わせることによって実現できる。

$$|a| = \sqrt{(Re(a))^2 + (Im(a))^2} \tag{5.2}$$

$$a \pm b = (Re(a) \pm Re(b)) + \sqrt{-1}(Im(a) \pm Im(b)) \tag{5.3}$$

$$ab = (Re(a)Re(b) - Im(a)Im(b)) + \sqrt{-1}(Im(a)Re(b) + Re(a)Im(b)) \tag{5.4}$$

$$a/b = \frac{a\overline{b}}{|b|^2} \tag{5.5}$$

ここで $\overline{b} = Re(b) - Im(b)\sqrt{-1}$ である。

[1]現在の CPU は内部に積み込んだ高速転送可能なキャッシュ(cache) メモリを持っており，一度メインメモリから読み込んだ値をそこに記憶しておき，2 度目以降のアクセスはそれを取り出すだけで済む。従って，このキャッシュメモリをうまく利用できるようにした線型計算プログラムは，素朴に組んだものより高速になる。よって単純に計算量だけでは計算時間を推定できないこともある。

表 5.1: 複素数演算の計算回数

	加減算	乗算	除算	平方根
$\|a\|$ ((5.2) 式)	1	2	0	1
$\|a\|$ ((5.6) 式)	1	2	1	1
$a \pm b$	2	0	0	0
ab	2	4	0	0
a/b ((5.5) 式)	3	6	2	0
a/b ((5.7) 式)	3	3	3	0

但し，オーバーフローを防止するため，$|a|$ と a/b は次のように計算するのが良いとされている [13]。

$$|a| = \begin{cases} Re(a) & (\text{if } Im(a) = 0) \\ Im(a) & (\text{if } Re(a) = 0) \\ |Re(a)|\sqrt{1 + \left(\frac{Im(a)}{Re(a)}\right)^2} & (\text{if } |Re(a)| \geq |Im(a)| > 0) \\ |Im(a)|\sqrt{1 + \left(\frac{Re(a)}{Im(a)}\right)^2} & (\text{if } |Im(a)| > |Re(a)| > 0) \end{cases} \tag{5.6}$$

$$a/b = \begin{cases} \text{計算不能} & (\text{if } b = 0 \text{ (即ち } Re(b) = Im(b) = 0)) \\ \dfrac{Re(a) + Im(a) \cdot \left(\frac{Im(b)}{Re(b)}\right)}{s} + \dfrac{-Re(a) \cdot \left(\frac{Im(b)}{Re(b)}\right) + Im(a)}{s}\sqrt{-1} & (\text{if } |Re(b)| \geq |Im(b)| \geq 0) \\ & \text{ここで } s = Re(b) + Im(b) \cdot \left(\frac{Im(b)}{Re(b)}\right) \\ \dfrac{Re(a) \cdot \left(\frac{Re(b)}{Im(b)}\right) + Im(a)}{s} + \dfrac{-Re(a) + Im(a) \cdot \left(\frac{Re(b)}{Im(b)}\right)}{s}\sqrt{-1} & (\text{if } |Im(b)| \geq |Re(b)| \geq 0) \\ & \text{ここで } s = Re(b) \cdot \left(\frac{Re(b)}{Im(b)}\right) + Im(b) \end{cases} \tag{5.7}$$

以上の複素数演算の計算回数を表 5.1 にまとめておく。対応する実数の演算と比べて，2〜3 倍の演算量を必要とすることが分かる。従って，複素数の演算は実数のそれに比べてかなり「高くつく」ことを認識しておく必要がある。当然のことながら，必要となるメモリ量も実数の 2 倍になる。

問題 5.2.1

1. 式 (5.6)，(5.7) がそれぞれ $|a|$ と a/b を計算していることを確認せよ。
2. 前章の多倍長浮動小数点数の四則演算のベンチマークテストの結果を用いて，複素数演算の性能評価を行え。また実際にベンチマークテストを行った結果と比較せよ。

5.3 基本線型計算

現在の数値計算は大規模化が進んでおり，そこではベクトル及び行列の基本線型計算 (linear compuation) が多用される。基本線型計算は次元数が上がるにつれて莫大な計算量を必要とすることを認識しておく必要がある。ここではその一端に触れることにする。

実ベクトル $\mathbf{a} = [a_1\ a_2\ \cdots a_n]^T \in \mathbb{R}^n$ の基本線型計算はそれぞれ次のように行う。

$$\text{スカラー積}\ \ \alpha\mathbf{a} = \begin{bmatrix} \alpha a_1 \\ \alpha a_2 \\ \vdots \\ \alpha a_n \end{bmatrix}\ (\text{ここで}\alpha \in \mathbb{R}) \tag{5.8}$$

$$\text{加減算}\ \ \mathbf{a} \pm \mathbf{b} = \begin{bmatrix} a_1 \pm b_1 \\ a_2 \pm b_2 \\ \vdots \\ a_n \pm b_n \end{bmatrix} \tag{5.9}$$

$$\text{内積}\ \ (\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} a_i b_i \tag{5.10}$$

同様に，実正方行列 $A = [a_{ij}] \in M_n(\mathbb{R})$ $(i, j = 1, 2, ..., n)$ の基本線型計算は次のようにして行われる。

$$\text{ベクトルとの積}\ \ A\mathbf{b} = \begin{bmatrix} \sum_{j=1}^{n} a_{1j} b_j \\ \sum_{j=1}^{n} a_{2j} b_j \\ \vdots \\ \sum_{j=1}^{n} a_{nj} b_j \end{bmatrix} \tag{5.11}$$

$$\text{スカラー積}\ \ \alpha A = \begin{bmatrix} \alpha a_{11} & \alpha a_{12} & \cdots & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} & \cdots & \alpha a_{2n} \\ \vdots & \vdots & & \vdots \\ \alpha a_{n1} & \alpha a_{n2} & \cdots & \alpha a_{nn} \end{bmatrix}\ (\text{ここで}\alpha \in \mathbb{R}) \tag{5.12}$$

$$\text{加減算}\ \ A \pm B = \begin{bmatrix} a_{11} \pm b_{11} & a_{12} \pm b_{12} & \cdots & a_{1n} \pm b_{1n} \\ a_{21} \pm b_{21} & a_{22} \pm b_{22} & \cdots & a_{2n} \pm b_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} \pm b_{n1} & a_{n2} \pm b_{n2} & \cdots & a_{nn} \pm b_{nn} \end{bmatrix} \tag{5.13}$$

$$\text{乗算}\ \ AB = \begin{bmatrix} \sum_{j=1}^{n} a_{1j} b_{j1} & \sum_{j=1}^{n} a_{1j} b_{j2} & \cdots & \sum_{j=1}^{n} a_{1j} b_{jn} \\ \sum_{j=1}^{n} a_{2j} b_{j1} & \sum_{j=1}^{n} a_{2j} b_{j2} & \cdots & \sum_{j=1}^{n} a_{2j} b_{jn} \\ \vdots & \vdots & & \vdots \\ \sum_{j=1}^{n} a_{nj} b_{j1} & \sum_{j=1}^{n} a_{nj} b_{j2} & \cdots & \sum_{j=1}^{n} a_{nj} b_{jn} \end{bmatrix} \tag{5.14}$$

ベクトル演算，行列演算の計算回数を表 5.2，5.3 にまとめておく。

問題 5.3.1

1. $\alpha, \beta \in \mathbb{R}$，$\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ であるとき，$\alpha\mathbf{a} \pm \beta\mathbf{b}$ の計算量を求めよ。

2. $\alpha, \beta \in \mathbb{R}, A, B, C \in M_n(\mathbb{R})$ である時，$(\alpha A \pm \beta B)C$ の計算量を求めよ。

表 5.2: ベクトル演算の計算回数

	加減算	乗算
$\alpha\mathbf{a}$	0	n
$\mathbf{a} \pm \mathbf{b}$	n	0
$(\mathbf{a}, \mathbf{b})$	$n-1$	n

表 5.3: 行列演算の計算回数

	加減算	乗算
$A\mathbf{b}$	$n(n-1)$	n^2
αA	0	n^2
AB	$n^2(n-1)$	n^3

3. $\mathbf{a}, \mathbf{b} \in \mathbb{C}^n$ である時，内積 $(\mathbf{a}, \mathbf{b})$ は

$$(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} a_i \overline{b}_i$$

と計算する。この計算量を求めよ。

5.4 収束判定と停止則について

多くの数値計算アルゴリズムは，最終的には収束する無限数列もしくはベクトル列を漸化式に基づいて反復的に生成するという形で表現される。本書で取り上げるアルゴリズムのうち，有限回の演算で終わるものは連立一次方程式の直接法 (第 7 章) だけであり，実用的な観点から，これをわざわざ無限ベクトル列を生成する形に直したアルゴリズム (第 9 章，第 10 章[2]) も使用される。

その際，どこで計算を打ち切るべきか，その機構をプログラム中に予め埋め込んでおく必要がある。この収束を判定する機構を停止則 (stopping rule) と呼ぶ。実際には収束しない計算を行うことも考えられるため，停止則に加えて最大反復回数を指定し，2 重に保険をかけておくのが普通である。

初等関数の近似計算のように，事前に反復回数が分かっているケースもあるが，多くは数列 (ベクトル列) の値そのものを見て判別する必要がある。どのように判別すべきかは，問題やアルゴリズムの数学的，数値的性質や，実用上の必要に応じてさまざまな方法が考えられる。

5.4.1 数列の場合

まず，$s_c (\neq 0)$ へ単調に収束する無限数列 $s_0, s_1, ..., s_n, ...$ を考える。このとき一般項 s_n を

$$s_n = s_c + S(n) \quad (\text{ここで } \lim_{n\to\infty} S(n) = 0) \tag{5.15}$$

と書く。この時 $S(n)$ は単調に 0 に収束することになる。この $S(n)$ を s_n の理論誤差 (打ち切り誤差) と呼ぶ。有限桁の浮動小数点数演算においては，打ち切り誤差が ε_M 以下になれば，誤差全体としてはそれ以下にすることは不可能と考えてよい。よって，$S(n) \approx \varepsilon_M$ になった時には数列の生成を停止する必要がある。この時，$|S(n)| > \varepsilon_M$ となる n の範囲を打ち切り誤差領域，$|S(n)| \leq \varepsilon_M$ となる n の範囲を丸め誤差領域と呼ぶ。ちょうどこの二つの領域の境目となる n において，最小の相対誤差が得られることになる。この n を最適点と呼ぶ。

[2]CG 法は理論的には有限のベクトル列で解に収束するが，丸め誤差の混入する近似計算では無限ベクトル列と同じ扱いが要求される。

一般には最適点における相対誤差になるまで収束させる必要がないことも多く，その場合には要求される相対誤差 (相対許容度 (relative tolerance))$\varepsilon_R > 0$ 以下になった時に停止すればよい。さすれば，十分大きな n に対しては

$$|s_{n+1} - s_n| = |S(n+1) - S(n)| \leq 2\max(|S(n+1)|, |S(n)|) \approx \varepsilon_R |s_c|$$

が成立した時ということになるので，$|s_n| \approx |s_c|$ であるから，実用的には

$$|s_{n+1} - s_n| \leq \varepsilon_R |s_n| \tag{5.16}$$

となった時に停止すれば，$rE(s_{n+1}) \approx \varepsilon_R$ と考えられる。

例題 5.4.1
IEEE754 倍精度計算で

$$s_n = \sum_{i=0}^{n} \left(-\frac{1}{2}\right)^i \tag{5.17}$$

を計算する例を考える。この時 $s_c = 2/3 = 0.666\cdots$ である。横軸に n を取り，$|s_{n+1} - s_n|, |s_n|, rE(s_n)$ をプロットしたのが図 5.1 である。ちょうど $|s_{n+1} - s_n|$ が $rE(s_n)$ と連動して小さくなっていること

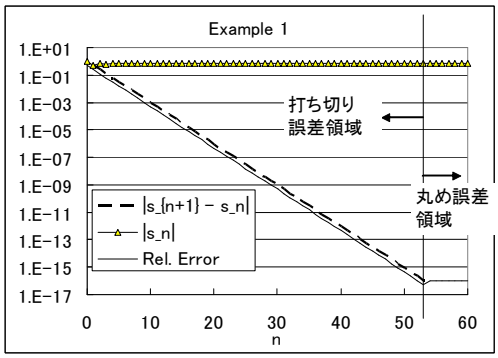

図 5.1: 数列 (5.17) の収束状況

が分かる。また，十分大きな n に対しては $|s_n| \approx |s_c|$ であるから，$|s_{n+1} - s_n| \leq \varepsilon_R |s_n|$ という停止則は，$rE(s_n) \approx \varepsilon_R$ である所でうまく停止させることができることが分かる。また，$\varepsilon_R < \varepsilon_M$ となる丸め誤差領域では，ε_M 以下の相対誤差となる収束値を得ることが不可能なことも理解できよう。

次に，$s_c = 0$ の場合を考える。この場合は $|S(n)| \approx \varepsilon_M$ となる所が最適点となるので，絶対許容度 $\varepsilon_A > \varepsilon_M$ を使用して

$$|s_{n+1} - s_n| \leq \varepsilon_A \tag{5.18}$$

となるところで停止する必要がある。これを例で示す。

例題 5.4.2
0 へ単調に収束する数列

$$s_n = \sum_{i=0}^{n} \left(-\frac{1}{2}\right)^i - \frac{2}{3} \tag{5.19}$$

を考える。前述の例題と同様に，横軸に n を取り，$|s_{n+1} - s_n|, |s_n|, rE(s_n)$ をプロットしたグラフを図 5.2 に示す。

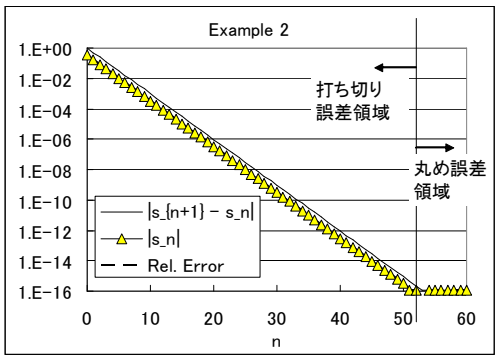

図 5.2: 数列 (5.19) の収束状況

0 に収束する数列であるから，当然十分大きな n に対しては $|s_n| \approx 0$ となり，他の全ての値も 0 に近づいていく。よって絶対許容度 ε_A が相対誤差の指標となる。これは真値が 0 の時の相対誤差が絶対誤差と同じであることと同義である。

以上の例により，0 を含む収束値を持つ単調数列を停止させるには，(5.16), (5.18) の両方の条件式の機能を持たせる必要がある。よって，ユーザには相対許容度 $0 < \varepsilon_R < \varepsilon_M$ と絶対許容度 $0 < \varepsilon_A$ の両方を指定させ，

$$|s_{n+1} - s_n| \leq \varepsilon_R |s_n| + \varepsilon_A \tag{5.20}$$

が満足した時に停止させるようにすることが望ましい。ε_A は一般的には前述の通り $\varepsilon_A > \varepsilon_M$ とすべきだが，$0 \neq s_c < \varepsilon_M$ であるケースも考えられるので，問題やアルゴリズムの持つ性質を十分見極めた上で，$\varepsilon_A < \varepsilon_M$ とする場合も考えられる。

最後に，単調でない収束列の場合を述べておく。一般には収束値も，収束の状況も不明であるとあり得る訳で，その際には (5.20) 式の停止則では真に相対誤差に近い値が望めないこともある。例えば，方程式 $f(x) = 0$ を満足する解に収束する数列 $x_0, x_1, ..., x_n, ...$ が得られた時は，数列の値そのものを見るよりは，いかにその値が方程式を満足しているか，すなわち残差 (residual) $f(x_n)$ が $f(x_n) \approx 0$ であるかを調べる方が合理的である。この場合は

$$|f(x_n)| < \varepsilon_A \tag{5.21}$$

を満足していればそこで停止する，という条件を使うべきであろう。あるいは，収束値にある程度近い出発値 s_0 が取れるとすれば，出発値よりは ε_R 程度の残差の改善が見込めるときに停止させる

$$|f(x_n)| < \varepsilon_R |f(x_0)| \tag{5.22}$$

という条件も考えられる。もちろんこの場合も，(5.20) と同様にして

$$|f(x_n)| < \varepsilon_R |f(x_0)| + \varepsilon_A \tag{5.23}$$

という形にしておいた方が汎用性が高まることになる。

5.4.2　ベクトル列，行列列の場合

収束する m 次元ベクトル列 $\mathbf{s}_0, \mathbf{s}_1, ..., \mathbf{s}_n, ...$ あるいは行列の列についても前述の停止則の議論がそのまま適用できる。行列についてはベクトルの場合と同様に議論できるので，以下ではベクトル列のみ扱う。

ベクトルを一つの塊と見てよい場合は，ベクトルノルム (第 8 章参照) を用いて，単調収束の場合は (5.20) 式より

$$\|\mathbf{s}_{n+1} - \mathbf{s}_n\| \leq \varepsilon_R \|\mathbf{s}_n\| + \varepsilon_A \tag{5.24}$$

となるし，方程式 $\mathbf{f}(\mathbf{x}) = 0$ の解に収束する場合に残差を用いるのであれば，(5.23) 式より

$$\|\mathbf{f}(\mathbf{x}_n)\| \leq \varepsilon_R \|\mathbf{f}(\mathbf{x}_0)\| + \varepsilon_A \tag{5.25}$$

としてよい。

しかし実用上，$\mathbf{s}_n = [s_1^{(n)}\ s_2^{(n)}\ ...s_m^{(n)}]^T$ の特定の要素，あるいは全要素が収束条件 (5.20)，あるいは (5.23) を満足していなければならないこともあり得る。ことに要素の絶対値の大きさが格段に異なる場合，一番絶対値の大きな要素の収束だけをチェックしただけでは不十分なこともある。

繰り返しになるが，停止則の選択は，ことにベクトル列の収束判定においては，問題の数学的性質，アルゴリズムの数値的性質，実用上の要求 (精度) の 3 要素を全て吟味して，それに応じたものを選択する必要がある。

問題 5.4.1

行列 A が

$$A = \begin{bmatrix} 1/2 & 0 \\ 1 & 1/3 \end{bmatrix} \in M_2(\mathbb{R})$$

である時，行列の列 S_n, P_n, Q_n を次のように定義する。

1. $S_n = I + \sum_{i=1}^{n} A^i$
2. $P_n = I + \sum_{i=1}^{n} (1/i!) A^i$
3. $Q_n = \sum_{i=1}^{n} ((-1)^i / i) A^i$

この時，$S = \lim_{n\to\infty} S_n$, $P = \lim_{n\to\infty} P_n$，$Q = \lim_{n\to\infty} Q_n$ の近似値を，次の条件を満足するように求めよ。また，その停止則に必要な計算量も算出せよ。

1. 全成分の相対誤差が 10^{-5} 以下
2. Frobenius ノルム ((8.10) 式参照)$\|\cdot\|_F$ に基づく相対誤差が 10^{-5} 以下
3. 1 ノルム $\|\cdot\|_1$ に基づく相対誤差が 10^{-5} 以下

演習問題

1. 複素数 $a = \pi + e\sqrt{-1}, b = \sqrt{2} + \sqrt{-3}$ について次の問いに答えよ。

   (a) 10 進 5 桁の浮動小数点数を用いて，a, b を表わせ。その際発生する実数部，虚数部の相対丸め誤差をそれぞれ求めよ。

   (b) 上で求めた近似値をそれぞれ $\widetilde{a}, \widetilde{b}$ とする時，$\widetilde{a}+\widetilde{b}, \widetilde{ab}, \widetilde{a}/\widetilde{b}$ をなるべく最小の計算量で済むようなアルゴリズムで計算し，10 進 5 桁で表示せよ。その際，計算において発生する相対丸め誤差もそれぞれ求めよ。

   (c) 上の計算で必要な計算量を求めよ。

   (d) (b) の計算をソフトウェアを用いて実行し，その手順と結果を説明せよ。

2. ベクトル $\mathbf{a} = [\pi\ e\ \sqrt{2}\ \sqrt{3}]^T$，$\mathbf{b} = [1/3\ \ 1/7\ \ 1/9\ \ 1/11]^T \in \mathbb{R}^4$ とする時，次の問いに答えよ。

   (a) 10 進 5 桁の浮動小数点数を用いて，$\mathbf{a}, \mathbf{b}$ を表わせ。その際発生する各成分ごとの相対丸め誤差を求めよ。

   (b) 上で求めた近似値をそれぞれ $\widetilde{\mathbf{a}}, \widetilde{\mathbf{b}}$ とする時，$\widetilde{\mathbf{a}}+\widetilde{\mathbf{b}}, (\widetilde{\mathbf{a}}, \widetilde{\mathbf{b}})$ をなるべく最小の計算量で済むようなアルゴリズムで計算し，10 進 5 桁で表示せよ。その際，計算において発生する相対丸め誤差も各成分ごとに求めよ。

   (c) 上の計算に必要な計算量を答えよ。

   (d) (b) の計算をソフトウェアを用いて実行し，その手順と結果を説明せよ。

3. 行列 A，$B \in M_2(\mathbb{R})$ が次のような成分を持つとき，次の問いに答えよ。

$$A = \begin{bmatrix} e & \pi \\ \pi & \sqrt{2} \end{bmatrix},\ B = \frac{1}{\pi^2 - e\sqrt{2}} \begin{bmatrix} -\sqrt{2} & \pi \\ \pi & -e \end{bmatrix}$$

   (a) 10 進 5 桁の浮動小数点数を用いて，A, B を表わせ。その際発生する各成分ごとの相対丸め誤差を求めよ。

   (b) 上で求めた近似値をそれぞれ $\widetilde{A}, \widetilde{B}$ とする時，$\widetilde{A}+\widetilde{B}, \widetilde{A}\,\widetilde{B}$ をなるべく最小の計算量で済むようなアルゴリズムで計算し，10 進 5 桁で表示せよ。その際，計算において発生する相対丸め誤差も各成分ごとに求めよ。

   (c) 上の計算に必要な計算量を答えよ。

   (d) (b) の計算をソフトウェアを用いて実行し，その手順と結果を説明せよ。

参考図書

数値計算にはある程度，アルゴリズムの計算量という観点が必要である。このことは伊理 [13] も自著の中で触れている。大学初年度向けのテキストは多数出版されているが，例えば

アルゴリズムの設計と解析 I, II
A.V.Aho et al./野崎昭弘・野下浩平 訳
サイエンス社
1977 年

などを，結果だけでも良いから通読しておくと参考になる。
　数値計算も含めた各種のアルゴリズムをスマートに実装したプログラム満載の

Java によるアルゴリズム事典
奥村晴彦ほか
技術評論社
2003 年

は，座右の書として便利である。