

「アルゴリズムと計算量」ノート

野崎昭弘著，共立出版

田村直之

(神戸大学工学部情報知能工学科)

E-mail: tamura@kobe-u.ac.jp

WWW: <http://bach.seg.kobe-u.ac.jp/tamura.html>

Chapter 0

はじめに

0.1 講義内容

計算量の理論 (Theory of Computational Complexity) について講述する.

0.2 教科書, 参考書

- 野崎昭弘 著
アルゴリズムと計算量, 共立出版
- 伊理正夫, 野崎昭弘, 野下浩平 編著
数学セミナー増刊 計算の効率化とその限界, 日本評論社
- A. V. Aho, J. E. Hopcroft, J. D. Ullman 著, 野崎昭弘, 野下浩平 共訳
アルゴリズムの設計と解析 I-II, サイエンス社
- A. V. Aho, J. E. Hopcroft, J. D. Ullman 著, 大野義夫 訳
データ構造とアルゴリズム, 培風館
- D. E. Knuth 著, 広瀬健 訳
基本算法/基礎概念, サイエンス社
- N. Wirth 著, 浦昭二, 國府方久史 共訳
アルゴリズムとデータ構造, 近代科学社
- 石畑清 著
アルゴリズムとデータ構造, 岩波書店
- 有澤誠 著
アルゴリズムとその解析
- H. S. Wilf 著, 西関隆夫, 高橋敬 共訳
アルゴリズムと計算量入門, 総研出版

Chapter 1

アルゴリズムと計算量

1.1 はじめに

問題 1

乗算のみを使って、与えられた x から x^{23} を計算する手順のうち、最も乗算の回数の少ない手順を示せ.

問題 2

長さ 3 の配列 $x[1], x[2], x[3]$ に格納されている 3 つの異なる実数 a, b, c を下の手順で昇順に並べかえる (つまり手順の実行後, $x[1]$ には a, b, c の最小値, $x[2]$ には中間値, $x[3]$ には最大値が入る). この時, 実行される比較の回数の最大値, 最小値, 平均値を求めよ. ただし a, b, c の順列 ($a < b < c, a < c < b$ など) はすべて等確率で起こるものとする.

- (1) $x[1]$ と $x[2]$ を比較し, $x[1] > x[2]$ なら (2) へ, $x[1] < x[2]$ なら (3) へ
- (2) $x[1]$ と $x[2]$ の内容を交換する
- (3) $x[2]$ と $x[3]$ を比較し, $x[2] > x[3]$ なら (4) へ, $x[2] < x[3]$ なら (6) へ
- (4) $x[2]$ と $x[3]$ の内容を交換する
- (5) (1) へ戻る
- (6) 終わり

問題 1 の解答

x^{23} は次のように 6 回の乗算で計算でき, これが最小である.

$$\begin{aligned}x_2 &\leftarrow x \times x \\x_3 &\leftarrow x_2 \times x \\x_5 &\leftarrow x_3 \times x_2 \\x_{10} &\leftarrow x_5 \times x_5 \\x_{13} &\leftarrow x_{10} \times x_3 \\x_{23} &\leftarrow x_{13} \times x_{10}\end{aligned}$$

問題 2 の解答

$a < b < c, a < c < b, b < a < c, b < c < a, c < a < b, c < b < a$ のそれぞれの場合について、実行される比較の回数を求めると 2, 4, 2, 4, 4, 4 となる。したがって、

$$\begin{aligned} \text{最大値} &= 4 \\ \text{最小値} &= 2 \\ \text{平均値} &= \frac{1}{6}(2 + 4 + 2 + 4 + 4 + 4) = 3.3 \end{aligned}$$

この手順は最良ではない。最良の手順では、最大値 = 3, 最小値 = 2, 平均値 = 2.6 である。

アルゴリズム (algorithm, 算法) とは

明確に定義され、機械的に実行することのできる計算手順 (または処理手順)

計算量 (complexity) とは

アルゴリズムを実行するのに必要なコスト

- **時間計算量 (time complexity)**
実行時間のコスト
- **領域計算量 (space complexity)**
記憶容量のコスト
- **最悪計算量 (最大計算量, worst-case complexity)**
最も計算量が大きくなるデータを与えられた場合の計算量
- **最良計算量 (最小計算量, best-case complexity)**
最も計算量が小さくなるデータを与えられた場合の計算量
- **平均計算量 (expected complexity)**
すべての入力データに対する計算量の平均

計算量の理論 (theory of complexity) とは

次のような問題を取り扱う。

- 与えられたアルゴリズムの計算量を求める。
- より計算量の小さいアルゴリズムを考案する。
- 与えられた問題を解くのに必要な計算量の上界 (upper bound) を得る (具体的なアルゴリズムを考案し、その計算量を求めることによって)。
- 与えられた問題を解くのに必要な計算量の下界 (lower bound) を得る (すべてのアルゴリズムに対して計算量が下界以上であることを証明することによって)。
- アルゴリズムが最良であることを証明する。
- 計算量の大きさに関していろいろな問題をクラス分けし、その性質について論じる (P, NP)。

1.2 直線的プログラム

アルゴリズムとその計算量を、以下で定義する直線的プログラム (straightforward program) により定義する.

A-C. 直線的プログラム

直線的プログラムの構文を定義する.

- 変数

$a, b, c, d, u, v, w, x, y$

$x_0, x_1, x_2, x_3, \dots, y_0, y_1, y_2, \dots$

- 定数

整数 $0, 365, -2$

分数 $1/6, -3/4$

小数 $0.1666\dots, 3.14159, -0.5$

定数記号 π, e

- 代入文 (assignment statement)

$$u \leftarrow v \Delta w$$

$$u \leftarrow v$$

ここで, u は変数, v, w は変数または定数, Δ は $+, -, \times$ のいずれかの演算子.

- 直線的プログラム

いくつかの代入文を並べた列

D. プログラムの解釈

直線的プログラムの意味を定義する.

付値 (valuation)

直線的プログラムで, 第 j 番目の代入文を実行した直後の変数 (または定数) α の内容を表す多項式を値とする関数 $V(\alpha, j)$

- (1) すべての変数, 定数 α について

$$V(\alpha, 0) = \alpha$$

- (2) 第 j 番目 ($j > 0$) の代入文を $u \leftarrow v \Delta w$ とすると

$$V(\alpha, j) = \begin{cases} V(v, j-1) \Delta V(w, j-1) & (\alpha = u) \\ V(\alpha, j-1) & (\alpha \neq u) \end{cases}$$

定義 n 個の代入文から成る直線的プログラムが, 多項式

$$E_1, E_2, \dots, E_t$$

を**計算する**とは, ある変数

$$v_1, v_2, \dots, v_t$$

について, 次の等式が成り立つことをいう.

$$V(v_1, n) = E_1, V(v_2, n) = E_2, \dots, V(v_t, n) = E_t$$

補題 1.1

プログラムの最初の j 個の代入文に乗算 \times が 1 回も現れなかったとする. そのとき, プログラムの中で使われている変数を x_1, x_2, \dots, x_m とすると,

$$V(x_i, j) = k_1 x_1 + \dots + k_m x_m + c$$

ここで, k_1, \dots, k_m は整数定数, c は任意の定数 (k_1, \dots, k_m, c は i ごとに異なってよい).

計算量の定義

- 時間計算量
たとえば, プログラム中の演算の数
- 領域計算量
たとえば, プログラム中に現れる異なる変数の個数

1.3 ベキ乗計算

問題 1 に示したように x^{23} は 6 回の乗算で計算でき, これが最小である. つまり, 乗算の回数を時間計算量の単位とすると, x^{23} を求めるのに必要な最小の時間計算量は 6 である.

定義 x^n (n は 2 以上の整数) を求めるのに必要な最小の時間計算量 (乗算の回数) を $M(n)$ で表す.

例題 1 より $M(23) = 6$ である. この他, $M(2) = 1, M(3) = 2, M(4) = 2, M(5) = 3, M(6) = 3, M(7) = 4$ などであることは, あらゆる直線的プログラムを短いものから順番にしらみつぶしに調べてみれば確かめることができる. しかし, 一般の n に対して $M(n)$ の値を求める公式は存在するのだろうか?

実は, そのような公式は知られていない. しかし $M(n)$ の**上界 (upperbound)** や**下界 (lowerbound)** を求めることはできる.

まず, 上界について考える. 上界を得るには, 具体的な直線的プログラムを示せばよい.

$$\left. \begin{array}{l} y \leftarrow x \times x \\ y \leftarrow y \times x \\ y \leftarrow y \times x \\ \dots \\ y \leftarrow y \times x \end{array} \right\} (n-2) \text{ 回繰り返す}$$

このプログラムは x^n を計算し、その時間計算量は $n - 1$ である。したがって

$$M(n) \leq n - 1$$

がいえる。

A. 2進アルゴリズム

次の漸化式に基づく計算法を2進アルゴリズムという。

$$x^n = \begin{cases} x & (n = 1 \text{ の時}) \\ (x^m)^2 & (n = 2m, m \geq 1 \text{ の時}) \\ (x^m)^2 \times x & (n = 2m + 1, m \geq 1 \text{ の時}) \end{cases}$$

このアルゴリズムの時間計算量 (乗算の回数) を $M_2(n)$ で表す。

$$M_2(n) = \begin{cases} 0 & (n = 1 \text{ の時}) \\ M_2(m) + 1 & (n = 2m, m \geq 1 \text{ の時}) \\ M_2(m) + 2 & (n = 2m + 1, m \geq 1 \text{ の時}) \end{cases}$$

練習問題

1. x^{14} を2進アルゴリズムにより求める直線的プログラムを示せ。

$M_2(n)$ の上限の見積りかた

$n = 2^k - 1$ の時 $M_2(n)$ は $M_2(2), \dots, M_2(n)$ のうち最大である。そこで

$$M_2'(k) = M_2(2^k - 1)$$

とおくと、次の漸化式が得られる。

$$M_2'(k) = \begin{cases} 0 & (k = 1 \text{ の時}) \\ M_2'(k-1) + 2 & (k > 1 \text{ の時}) \end{cases}$$

これを解いて、

$$M_2'(k) = 2(k-1)$$

$2^{k-1} \leq n \leq 2^k - 1$ なる n に対して、 $k = \lfloor \log_2 n \rfloor + 1$ であることから、

$$M_2(n) \leq 2 \lfloor \log_2 n \rfloor$$

となる。

補題 1.2 任意の $n \geq 1$ に対して

$$M_2(n) \leq 2 \lfloor \log_2 n \rfloor$$

証明 n に関する数学的帰納法で証明する.

i) $n = 1$ の時

右辺 = 0 で成立する.

ii) 1 から $n - 1$ までについて, 補題が成り立つと仮定して,

a) $n = 2m$ ($m \geq 1$) の時

$M_2(n) = M_2(m) + 1$ であるから帰納法の仮定により

$$\begin{aligned} M_2(n) &\leq 2 \lfloor \log_2 m \rfloor + 1 \\ &= 2 \lfloor \log_2 2m \rfloor - 1 \quad (\lfloor x - 1 \rfloor = \lfloor x \rfloor - 1 \text{ より}) \\ &< 2 \lfloor \log_2 n \rfloor \end{aligned}$$

b) $n = 2m + 1$ ($m \geq 1$) の時

$M_2(n) = M_2(m) + 2$ であるから帰納法の仮定により

$$\begin{aligned} M_2(n) &\leq 2 \lfloor \log_2 m \rfloor + 2 \\ &= 2 \lfloor \log_2 2m \rfloor \\ &\leq 2 \lfloor \log_2(2m + 1) \rfloor \quad (\log_2 2m < \log_2(2m + 1) \text{ だから}) \\ &= 2 \lfloor \log_2 n \rfloor \end{aligned}$$

床と天井

$\lfloor x \rfloor$ x の床 (floor) x を越えない最大の整数

$\lceil x \rceil$ x の天井 (ceiling) x 以上で最小の整数

補題 1.2.1

$$M(n) \leq 2 \lfloor \log_2 n \rfloor$$

証明 補題 1.2 より.

補題 1.2.2 任意の $n \geq 1$ に対して

$$M_2(n) \geq \lceil \log_2 n \rceil$$

証明 n に関する数学的帰納法で証明する.

i) $n = 1$ の時

右辺 = 0 で成立する.

ii) 1 から $n - 1$ までについて, 補題が成り立つと仮定して,

a) $n = 2m$ ($m \geq 1$) の時

$M_2(n) = M_2(m) + 1$ であるから帰納法の仮定により

$$\begin{aligned} M_2(n) &\geq \lceil \log_2 m \rceil + 1 \\ &= \lceil \log_2 2m \rceil \quad (\lceil x + 1 \rceil = \lceil x \rceil + 1 \text{ より}) \\ &= \lceil \log_2 n \rceil \end{aligned}$$

b) $n = 2m + 1$ ($m \geq 1$) の時

$M_2(n) = M_2(m) + 2$ であるから帰納法の仮定により

$$\begin{aligned} M_2(n) &\geq \lceil \log_2 m \rceil + 2 \\ &= \lceil \log_2 4m \rceil \\ &\geq \lceil \log_2(2m + 1) \rceil \quad (4m > 2m + 1 \text{ だから}) \\ &= \lceil \log_2 n \rceil \end{aligned}$$

注意

$$M(n) \geq \lceil \log_2 n \rceil$$

は補題 1.2.2 からはいえない。

B. K 進アルゴリズム

次の漸化式に基づく計算法を K 進アルゴリズムという ($K \geq 2$)。

$$x^n = \begin{cases} x_n & (1 \leq n < K \text{ の時}) \\ (x^m)^K & (n = Km, m \geq 1 \text{ の時}) \\ (x^m)^K \times x_d & (n = Km + d, m \geq 1, 1 \leq d < K \text{ の時}) \end{cases}$$

ただし,

$$x_d = x^d \quad (1 \leq d < K)$$

であり, x_d は, x^n の計算の前に前処理としてあらかじめ値が求めてあるものとする. また $(x^m)^K$ の K 乗は 2 進アルゴリズムで求める.

このアルゴリズムの時間計算量 (乗算の回数) を $M_K(n)$ で表す. 前処理の時間計算量を M_K , 前処理を除いた x^n の時間計算量を $M'_K(n)$ とすると,

$$M_K(n) = M_K + M'_K(n)$$

$$M_K = K - 2$$

$$M'_K(n) = \begin{cases} 0 & (1 \leq n < K \text{ の時}) \\ M'_K(m) + q & (n = Km, m \geq 1 \text{ の時}) \\ M'_K(m) + q + 1 & (n = Km + d, m \geq 1, 1 \leq d < K \text{ の時}) \end{cases}$$

$$q = M_2(K)$$

補題 1.3 任意の $n \geq 1$, $K \geq 2$ に対して

$$M_K(n) \leq (K - 2) + (q + 1) \lfloor \log_K n \rfloor$$

ここで $q = M_2(K)$ である.

証明 n に関する数学的帰納法で

$$M'_K(n) \leq (q + 1) \lfloor \log_K n \rfloor$$

を証明する.

i) $1 \leq n < K$ の時

右辺 = 0 で成立する.

ii) 1 から $n - 1$ までについて, 補題が成り立つと仮定して,

a) $n = Km$ ($m \geq 1$) の時

$M'_K(n) = M'_K(m) + q$ であるから帰納法の仮定により

$$\begin{aligned} M'_K(n) &\leq (q + 1) \lfloor \log_K m \rfloor + q \\ &= (q + 1) \lfloor \log_K Km \rfloor - 1 \\ &< (q + 1) \lfloor \log_K n \rfloor \end{aligned}$$

b) $n = Km + d$ ($m \geq 1, 1 \leq d < K$) の時

$M'_K(n) = M'_K(m) + q + 1$ であるから帰納法の仮定により

$$\begin{aligned} M'_K(n) &\leq (q + 1) \lfloor \log_K m \rfloor + q + 1 \\ &= (q + 1) \lfloor \log_K Km \rfloor \\ &\leq (q + 1) \lfloor \log_K (Km + d) \rfloor \\ &= (q + 1) \lfloor \log_K n \rfloor \end{aligned}$$

補題 1.3.1 ($M(n)$ の上界)

$$(1) M(n) \leq 2 \lfloor \log_2 n \rfloor$$

$$(2) M(n) \leq (K - 2) + (M(K) + 1) \lfloor \log_K n \rfloor$$

証明 (1) は補題 1.2.1 と同一. (2) は, K 進アルゴリズムを変更して, K 乗を最良の方法で求めたとすると, 補題 1.3 で $q = M(K)$ とできる.

n	$M(n)$	$M_2(n)$	$M_3(n)$	$M_4(n)$	$M_5(n)$	$M_6(n)$
2	1	1	1	2	3	4
3	2	2	3	2	3	4
4	2	2	4	4	3	4
5	3	3	4	5	6	4
6	3	3	3	5	7	7
7	4	4	4	5	7	8
8	3	3	4	4	7	8
9	4	4	5	5	7	8
10	4	4	6	5	6	8
11	5	5	6	5	7	8
12	4	4	6	4	7	7
13	5	5	7	5	7	8
14	5	5	7	5	7	8
15	5	6	6	5	6	8
16	4	4	7	6	7	8
17	5	5	7	7	7	8
18	5	5	5	7	7	7
19	6	6	6	7	7	8
20	5	5	6	7	6	8
21	6	6	6	8	7	8
22	6	6	7	8	7	8
23	6	7	7	8	7	8
24	5	5	6	7	7	7
25	6	6	7	8	9	8
26	6	6	7	8	10	8
27	6	7	7	8	10	8
28	6	6	8	7	10	8
29	7	7	8	8	10	8
30	6	7	8	8	10	7
31	7	8	9	8	11	8

補題 1.4 ($M(n)$ の下界) 任意の $n \geq 2$ について

$$M(n) \geq \lceil \log_2 n \rceil$$

証明 n についての数学的帰納法で証明する.

i) $n = 2$ の時

右辺 = 1 で成立する.

ii) 2 から $n - 1$ までについて, 補題が成り立つと仮定する. x^n を計算する任意のプログラムについて, 最後の代入文 (t 番目) を

$$u \leftarrow v \times w$$

とする.

$$\begin{aligned} V(u, t) &= x^n \\ V(v, t-1) &= x^p \\ V(w, t-1) &= x^q \\ p+q &= n \end{aligned}$$

と仮定してよい. また $n > p \geq n/2$ と仮定して一般性を失わない.

$$\begin{aligned} t-1 &\geq M(p) && (M(p) \text{ の定義より}) \\ &\geq \lceil \log_2 p \rceil && (\text{帰納法の仮定より}) \\ &\geq \lceil \log_2(n/2) \rceil && (p \geq n/2 \text{ より}) \\ &= \lceil \log_2 n \rceil - 1 \end{aligned}$$

任意のプログラムについて成り立つから

$$M(n) \geq \lceil \log_2 n \rceil$$

系 1

$$\lceil \log_2 n \rceil \leq M(n) \leq 2 \lceil \log_2 n \rceil$$

系 2

$$M(2^q) = q$$

定理 1

$$\lim_{n \rightarrow \infty} \frac{M(n)}{\log_2 n} = 1$$

1.4 多項式の数値計算

A. 1 変数 n 次多項式

1 変数 n 次多項式 ($a_n \neq 0$)

$$\begin{aligned} y &= a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \\ &= \sum_{i=0}^n a_i x^i \end{aligned}$$

を計算する直線的プログラムの計算量について考える.

項別計算法

	乗算回数	加減算回数
$y_1 \leftarrow x$	0	0
$y_2 \leftarrow y_1 \times x$	} x^2, \dots, x^n の計算	0
$y_3 \leftarrow y_2 \times x$		
...		
$y_n \leftarrow y_{n-1} \times x$		
$y_1 \leftarrow a_1 \times y_1$	} $a_1 x, \dots, a_n x^n$ の計算	0
$y_2 \leftarrow a_2 \times y_2$		
...		
$y_n \leftarrow a_n \times y_n$		
$y \leftarrow a_0 + y_1$	} 和の計算	n
$y \leftarrow y + y_2$		
...		
$y \leftarrow y + y_n$		

したがって、乗算回数 $G^\times(n)$ と加減算回数 $G^+(n)$ は次の式で表される。

$$G^\times(n) = 2n - 1$$

$$G^+(n) = n$$

乗算は加減算よりも時間がかかるため、時間計算量として乗算の回数だけに注目することがある。

ホーナー (Horner) の方法

$$y = (\dots (a_n \times x + a_{n-1}) \times x + a_{n-2}) \times \dots + a_1) \times x + a_0$$

乗算回数 $H^\times(n)$ と加減算回数 $H^+(n)$ は次のようになる。

$$H^\times(n) = n$$

$$H^+(n) = n$$

B. 前処理を許す場合の計算法

変数 x の値に関係のない、係数のみに関わる式変形 (前処理, preprocessing) を行い、変形した式についての計算量を求める。変形に要するコストは無視する。

4 次式に対するトッド (Todd) の方法

4 次多項式を次のように変形する。

$$y = a_4((x - \alpha)^2 + \beta)((x - \alpha)^2 + x + \gamma) + \delta$$

$\alpha, \beta, \gamma, \delta$ は未定係数法により加減乗除だけで求めることができる。変形した式は、乗算 3 回、加減算 5 回で値を求めることができる。

クヌース (Knuth) の方法

まず, n 次多項式を次のように変形する (ただし, $X = x^2$).

$$\begin{aligned} y &= axP(X) + Q(X) \\ P(X) &= b_s X^s + \cdots + b_1 X + b_0 \\ Q(X) &= c_t X^t + \cdots + c_1 X + c_0 \end{aligned}$$

ここで, $a \neq 0, b_s = 1, c_t \neq 0$ としてよい. また $2s + 1$ と $2t$ の大きい方が n に一致する.

次に, 方程式 $P(X) = 0$ を解いて, 解 $\alpha_1, \dots, \alpha_s$ を求め (どのようにして求めるかは問題としない), 次のようにおく.

$$\begin{aligned} P_j(X) &= (X - \alpha_1) \cdots (X - \alpha_j) \quad (1 \leq j \leq s) \\ P_0(X) &= 1 \end{aligned}$$

$P(X) = P_s(X)$ となる.

次に, $Q_t(X) = Q(X)$ とおき, $1 \leq k \leq s$ なる k について, $Q_{t-k+1}(X)$ を $X - \alpha_{s-k+1}$ で割った商を $Q_{t-k}(X)$, 余りを d_k とする.

$$Q_{t-k+1}(X) = Q_{t-k}(X) \cdot (X - \alpha_{s-k+1}) + d_k \quad (1 \leq k \leq s)$$

$Q_k(X)$ は, $k \geq 1$ なら k 次多項式, $k = 0$ なら定数, $k < 0$ なら 0 となる.

次のように y_j をおく.

$$\begin{aligned} y_{s-k+1} &= axP_{s-k+1}(X) + Q_{t-k+1}(X) \quad (1 \leq k \leq s) \\ y_0 &= ax + Q_{t-s}(X) \end{aligned}$$

すると, y_j について次の漸化式が成り立つ.

$$\begin{aligned} y_j &= y_{j-1} \cdot (X - \alpha_j) + d_{s-j+1} \quad (1 \leq j \leq s) \\ y_0 &= ax + Q_{t-s}(X) \end{aligned}$$

また, $y_s = y$ である.

したがって, 次のようにして多項式の値を計算できる.

	乗算回数	加減算回数
(1) $X = x^2$ の計算	1	0
(2) y_0 の計算		
(ア) $t - s \geq 0$ の場合		
$y_0 = ax + Q_{t-s}(X)$	$t - s + 1$	$t - s + 1$
$Q_{t-s}(X)$ はホーナーの方法で計算		
(イ) $t - s < 0$ の場合		
$y_0 = ax$	1	0
(3) $1 \leq j \leq s$ なる j について		
$y_j = y_{j-1}(X - \alpha_j) + d_{s-j+1}$ の計算	s	$2s$

したがって,

$$\begin{aligned} K^\times(n) &= \begin{cases} t + 2 & (t - s \geq 0) \\ s + 2 & (t - s < 0) \end{cases} \\ K^+(n) &= \begin{cases} t + s + 1 & (t - s \geq 0) \\ 2s & (t - s < 0) \end{cases} \end{aligned}$$

また, n は $2s + 1$ と $2t$ の大きい方だから

$$n = \begin{cases} 2t & (t - s > 0) \\ 2s + 1 & (t - s \leq 0) \end{cases}$$

結局, 次の結果を得る.

$$\begin{aligned} K^\times(n) &= \left\lfloor \frac{n}{2} \right\rfloor + 2 \\ K^+(n) &\leq n \end{aligned}$$

定理 (イブ, J. Eve) 実係数多項式 $F(x)$ に対して, 適当に定数 c を選んで $x = z + c$ とおき, $F(x) = F(z + c) = G(z)$ と変形し, $G(z)$ にクヌースの方法を適用すれば, 現れる解 $\alpha_1, \alpha_2, \dots, \alpha_s$ はすべて実数になる.

証明 下記論文を参照.

J. Eve: The Evaluation of Polynomials, Numerische Mathematik, Vol. 6, pp.17-21 (1964).

練習問題

1. $y = 2x^5 + x^4 - 6x^3 + x^2 + 4x + 1$ を計算する直線的プログラムをクヌースの方法により求めよ.

C. 定数倍を無視してよい場合

変数 x の値に関係のない定数の乗算と, それ以外の乗算を区別し, 計算量を求める. 場合によっては, 定数倍を行う乗算の計算量は無視する.

パターソン-ストックマイヤー (Paterson-Stockmeyer) の方法

まず, n 次多項式を次のように変形する ($X = x^k$).

$$\begin{aligned} y &= \sum_{i=0}^n a_i x^i \\ &= (a_0 + a_1 x + \dots + a_{k-1} x^{k-1}) + \\ &\quad (a_k + a_{k+1} x + \dots + a_{2k-1} x^{k-1}) X + \dots + \\ &\quad (a_{jk} + a_{jk+1} x + \dots + a_{(j+1)k-1} x^{k-1}) X^j + \dots + \\ &\quad (a_{(m-1)k} + a_{(m-1)k+1} x + \dots + a_n x^{n-(m-1)k}) X^{m-1} \end{aligned}$$

ここで, m と k は, $(m-1)k \leq n < mk$ を満たす任意の整数である.

したがって, 次のようにして多項式の値を計算できる.

	乗算	定数倍	加減算
(1) $x^2, \dots, x^{k-1}, x^k = X$ を計算する	$k-1$	0	0
(2) X^j の係数の計算	0	$n-m+1$	$n-m+1$
(3) ホーナーの方法で y を計算する	$m-1$	0	$m-1$
合計	$m+k-2$	$n-m+1$	n

$k = \lceil \sqrt{n+1} \rceil$ とおくと, $m = \lfloor n/k \rfloor + 1$, 乗算回数 $P^\times(n)$, 定数乗算回数 $P^c(n)$, 加減算回数 $P^+(n)$ は次のようになる.

$$\begin{aligned} P^\times(n) &\approx 2\sqrt{n} \\ P^c(n) &\approx n - \sqrt{n} \\ P^+(n) &= n \end{aligned}$$

D. 複数個の多項式の計算

複素数の積

$(a + bi) \times (c + di) = y + zi$ の計算.

直接的な方法

$$\begin{aligned} y &= a \times c - b \times d \\ z &= a \times d + b \times c \end{aligned}$$

乗算 4 回, 加減算 2 回

改良した方法

$$\begin{aligned} u &= a \times c \\ v &= b \times d \\ y &= u - v \\ z &= (a + b) \times (c + d) - u - v \end{aligned}$$

乗算 3 回, 加減算 5 回

1 次多項式の積

$(px + q) \times (rx + s) = ax^2 + bx + c$ の計算.

直接的な方法

$$\begin{aligned} a &= p \times r \\ b &= p \times s + q \times r \\ c &= q \times s \end{aligned}$$

乗算 4 回, 加減算 1 回

改良した方法

$$\begin{aligned} a &= p \times r \\ c &= q \times s \\ b &= (p + q) \times (r + s) - a - c \end{aligned}$$

乗算 3 回, 加減算 4 回

3 次多項式の積

$(a_3x^3 + a_2x^2 + a_1x + a_0) \times (b_3x^3 + b_2x^2 + b_1x + b_0) = \sum_{k=0}^6 c_k x^k$ の計算.

直接的な方法

$$c_k = \sum_{i+j=k} a_i b_j$$

乗算 16 回, 加減算 9 回

改良した方法 まず, X, A_1, A_0, B_1, B_0 を次のようにおく.

$$\begin{aligned} X &= x^2 \\ A_1 &= a_3x + a_2 \\ A_0 &= a_1x + a_0 \\ B_1 &= b_3x + b_2 \\ B_0 &= b_1x + b_0 \end{aligned}$$

すると, 求める式は次のように書ける.

$$\begin{aligned} &(A_1X + A_0) \times (B_1X + B_0) \\ &= (A_1B_1)X^2 + (A_1B_0 + A_0B_1)X + A_0B_0 \end{aligned}$$

これを次のようにして計算する.

	乗算	加減算
(1) $A_1B_1 = d_2x^2 + d_1x + d_0$ の計算	4 or 3	1 or 4
(2) $A_0B_0 = e_2x^2 + e_1x + e_0$ の計算	4 or 3	1 or 4
(3) $A_1B_0 + A_0B_1$ $= (A_1 + A_0) \times (B_1 + B_0) - A_1B_1 - A_0B_0$ $= f_2x^2 + f_1x + f_0$ の計算		
(i) $A_1 + A_0$	0	2
(ii) $B_1 + B_0$	0	2
(iii) (i) \times (ii)	4 or 3	1 or 4
(iv) (iii) $- A_1B_1$	0	3
(v) (iv) $- A_0B_0$	0	3
(4) $c_6 = d_2$ $c_5 = d_1$ $c_4 = d_0 + f_2$ $c_3 = f_1$ $c_2 = f_0 + e_2$ $c_1 = e_1$ $c_0 = e_0$		
合計	12 or 9	15 or 24

一般的方法については, 3 章で述べられている.

練習問題

1. $(a_3x^3 + a_2x^2 + a_1x + a_0) \times (b_3x^3 + b_2x^2 + b_1x + b_0)$ を求める直線的プログラムを示せ.

Chapter 2

反復を含むプログラム

2.1 直線的プログラムの拡張

直線的プログラムを拡張し、反復を含むアルゴリズムを記述できるようにする。

文

- 代入文
- 反復文
- 複合文

反復文

for $i = p$ **to** q **do** S

- i は変数, p, q は整数定数, S は文.
- $i = p, p + 1, \dots, q$ なる i に対して, S を繰り返し実行する.
- 反復文は, それと同じことを行う直線的プログラムに展開できる.
- 計算量は展開した直線的プログラムで考える.

複合文

begin $S_1 S_2 \dots S_n$ **end**

- S_i は文.
- $S_1 S_2 \dots S_n$ を実行する.
- 文の間で改行したり, セミコロンで区切ったりしてよい.

代入文の拡張

代入文の右辺に, $v \times w + x$ などの複合式を許す. 除算の使用も許す.

例 3

(2,3) 行列 $A = (a_{ij})$ と 3 次元列ベクトル x の積 c の計算.

```

for  $i = 1$  to 2 do
   $c_i \leftarrow 0$ 
for  $i = 1$  to 2 do
  for  $j = 1$  to 3 do begin
     $t \leftarrow a_{ij} \times x_j$ 
     $c_i \leftarrow c_i + t$ 
  end

```

2.A 行列の和と積

a. 行列の和の計算

(m, n) 行列 $A = (a_{ij})$, $B = (b_{ij})$ の和 $C = (c_{ij}) = A + B$

$$c_{ij} = a_{ij} + b_{ij} \quad (1 \leq i \leq m, 1 \leq j \leq n)$$

プログラム

```

for  $i = 1$  to  $m$  do
  for  $j = 1$  to  $n$  do
     $c_{ij} \leftarrow a_{ij} + b_{ij}$  .....  $mn$ 

```

加減算回数は mn となる.

b. 行列の積の計算

(l, m) 行列 $A = (a_{ik})$ と (m, n) 行列 $B = (b_{kj})$ の積 $C = (c_{ij}) = AB$, C は (l, n) 行列

$$c_{ij} = \sum_{k=1}^m a_{ik} b_{kj} \quad (1 \leq i \leq l, 1 \leq j \leq n)$$

プログラム

```

for  $i = 1$  to  $l$  do
  for  $j = 1$  to  $n$  do begin
     $c_{ij} \leftarrow 0$ 
    for  $k = 1$  to  $m$  do
       $c_{ij} \leftarrow c_{ij} + a_{ik} \times b_{kj}$  .....  $lmn$ 
  end

```

乗除算回数は

$$M(l, m, n) = lmn$$

さらに 3 章で, より良い方法について議論する.

2.B 行列式の計算

n 次正方行列 $A = (a_{ij})$ の行列式は, $|A|$ や $\det A$ などと表され, 次の式で定義される.

$$|A| = \sum (\operatorname{sgn} P) a_{1p_1} a_{2p_2} \cdots a_{np_n}$$

ただし, $P = \begin{pmatrix} 1 & 2 & \cdots & n \\ p_1 & p_2 & \cdots & p_n \end{pmatrix}$ は 1 から n までの番号の置換で, $\operatorname{sgn} P$ は置換の符号 (すなわち, P が偶置換のとき $\operatorname{sgn} P = +1$, 奇置換のとき $\operatorname{sgn} P = -1$) とし, 上記の式における和は $n!$ 個のすべての P をわたるものとする.

a. 定義にしたがって計算する方法

乗除算の回数は次のようになる.

$$D_1(n) = n! \times (n-1)$$

b. ラプラスの展開定理を用いる方法

ラプラスの展開定理を用いると, 任意の i, j について行列式を次のように展開できる.

$$\begin{aligned} |A| &= a_{i1} \tilde{a}_{i1} + a_{i2} \tilde{a}_{i2} + \cdots + a_{in} \tilde{a}_{in} \\ &= a_{1j} \tilde{a}_{1j} + a_{2j} \tilde{a}_{2j} + \cdots + a_{nj} \tilde{a}_{nj} \end{aligned}$$

ここで, \tilde{a}_{ij} は

$$\tilde{a}_{ij} = (-1)^{i+j} \Delta_{ij}$$

であり, A における a_{ij} の余因子と呼ばれ, Δ_{ij} は A の第 i 行および第 j 列を除いてできる $n-1$ 次の小行列式を示す.

乗除算の回数は次の漸化式で表される.

$$D_2(n) = \begin{cases} 0 & (n=1) \\ n(D_2(n-1) + 1) & (n>1) \end{cases}$$

$D_2'(n) = D_2(n)/n!$ とおいてこれを解くと次のようになる.

$$D_2(n) = n! \sum_{k=1}^{n-1} \frac{1}{k!} \approx (e-1)n!$$

c. 上三角行列式に変形して計算する方法

命題 1

行列式のある行に他の行の何倍かを加えても, 行列式の値は変わらない.

命題 2

上三角行列式の値は対角線要素の積に等しい.

上三角行列式に変形して行列式を計算するプログラム

```

for i = 1 to n - 1 do
  for k = i + 1 to n do begin
    t ← aki/aii ..... α
    for j = i + 1 to n do
      akj ← akj - t × aij ..... β
    end
  end
d ← a11
for i = 2 to n do
  d ← d × aii ..... γ

```

ただし、 α の行において常に $a_{ii} \neq 0$ であるものと仮定する。 $a_{ii} = 0$ なら、 $a_{li} \neq 0$ となっているような第 l 行と、第 i 行を交換する。その場合、得られた値は元の行列式の値の -1 倍である。

代入文の実行される回数はそれぞれ、

$$\begin{aligned} \alpha &= \sum_{i=1}^{n-1} (n-i) = \frac{1}{2}n^2 - \frac{1}{2}n \\ \beta &= \sum_{i=1}^{n-1} \sum_{k=i+1}^n (n-i) = \frac{1}{3}n^3 - \frac{1}{2}n^2 + \frac{1}{6}n \\ \gamma &= n-1 \end{aligned}$$

したがって、乗除算の回数は

$$D_3(n) = \alpha + \beta + \gamma = \frac{1}{3}n^3 + \frac{2}{3}n - 1 = O(n^3)$$

n	$D_1(n)$	$D_2(n)$	$D_3(n)$
2	2	2	3
3	12	9	10
4	72	40	32
5	480	205	44
6	2880	1236	75
7	25200	8659	118

練習問題

1. $\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix}$ を求めるとき、どのような求め方が計算量が少ないか。

2.C 逆行列の計算

A が正則行列、すなわち $|A| \neq 0$ のとき、 A は逆行列 A^{-1} を持ち次の式が成り立つ (E は単位行列)。

$$AA^{-1} = A^{-1}A = E$$

代入文の実行される回数はそれぞれ,

$$\alpha = \sum_{i=1}^n (2n - i) = \frac{3}{2}n^2 - \frac{1}{2}n$$

$$\beta + \gamma = \sum_{i=1}^n (n - 1)(2n - i) = \frac{3}{2}n^3 - 2n^2 + \frac{1}{2}n$$

したがって, 乗除算の回数は

$$I_2(n) = \alpha + \beta + \gamma = \frac{3}{2}n^3 - \frac{1}{2}n^2 = O(n^3)$$

練習問題

1. $I_2(3)$ はいくつか? (答: 36)

2.2 連立 1 次方程式の解法

与えられた n 次正則行列 A と n 次列ベクトル \mathbf{b} から, 次の式を満たす n 次列ベクトル \mathbf{x} を求める.

$$A\mathbf{x} = \mathbf{b}$$

a. クラームルの公式により計算する方法

クラームルの公式

$$x_j = \frac{1}{|A|} \begin{vmatrix} a_{11} & \cdots & a_{1j-1} & b_1 & a_{1j+1} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2j-1} & b_2 & a_{2j+1} & \cdots & a_{2n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nj-1} & b_n & a_{nj+1} & \cdots & a_{nn} \end{vmatrix} \quad (1 \leq j \leq n)$$

乗除算の回数は

$$E_1(n) = D_3(n) + n(D_3(n) + 1)$$

$$= \frac{1}{3}n^4 + \frac{1}{3}n^3 + \frac{2}{3}n^2 + \frac{2}{3}n - 1 = O(n^4)$$

練習問題

1. $E_1(3)$ はいくつか? (答: 43)

b. 逆行列を求めて計算する方法

$$\mathbf{x} = A^{-1}\mathbf{b}$$

乗除算の回数は

$$E_2(n) = I_2(n) + M(n, n, 1)$$

$$= \frac{3}{2}n^3 + \frac{1}{2}n^2 = O(n^3)$$

練習問題

1. $E_2(3)$ はいくつか? (答: 45)

A. ジョルダンの方法 (ガウス-ジョルダンの方法, 掃き出し法)

$(a_{ij}) = (A | \mathbf{b})$ に対して基本変形 P_1, P_2, \dots, P_k を行い, $(E | \mathbf{x})$ となるようにする.

ジョルダンの方法のプログラム

```

for i = 1 to n do begin
  for j = i + 1 to n + 1 do
     $a_{ij} \leftarrow a_{ij} / a_{ii}$  .....  $\alpha$ 
  for k = 1 to i - 1 do
    for j = i + 1 to n + 1 do
       $a_{kj} \leftarrow a_{kj} - a_{ki} \times a_{ij}$  .....  $\beta$ 
    for k = i + 1 to n do
      for j = i + 1 to n + 1 do
         $a_{kj} \leftarrow a_{kj} - a_{ki} \times a_{ij}$  .....  $\gamma$ 
  end

```

代入文の実行される回数はそれぞれ,

$$\alpha = \sum_{i=1}^n (n - i + 1) = \frac{1}{2}n^2 + \frac{1}{2}n$$

$$\beta + \gamma = \sum_{i=1}^n (n - 1)(n - i + 1) = \frac{1}{2}n^3 - \frac{1}{2}n$$

したがって, 乗除算の回数は

$$E_3(n) = \alpha + \beta + \gamma = \frac{1}{2}n^3 + \frac{1}{2}n^2 = O(n^3)$$

練習問題

1. $E_3(3)$ はいくつか? (答: 18)

B. ガウスの方法

$(a_{ij}) = (A | \mathbf{b})$ に対して基本変形 P_1, P_2, \dots, P_k を行い, $(a'_{ij}) = (A' | \mathbf{b}')$ で, A' が対角要素がすべて 1 の上三角行列となるようにする (前進消去). その後, 次の漸化式

$$\begin{aligned}
x_n &= a'_{n, n+1} \\
x_i &= a'_{i, n+1} - a'_{i, i+1}x_{i+1} - a'_{i, i+2}x_{i+2} - \dots - a'_{i, n}x_n \\
&= a'_{i, n+1} - \sum_{j=i+1}^n a'_{ij}x_j \quad (1 \leq i < n)
\end{aligned}$$

により, \mathbf{x} を求める (後退代入).

プログラム

```

for  $i = 1$  to  $n$  do begin
  for  $j = i + 1$  to  $n + 1$  do
     $a_{ij} \leftarrow a_{ij} / a_{ii}$  .....  $\alpha$ 
  for  $k = i + 1$  to  $n$  do
    for  $j = i + 1$  to  $n + 1$  do
       $a_{kj} \leftarrow a_{kj} - a_{ki} \times a_{ij}$  .....  $\beta$ 
    end
  for  $k = 1$  to  $n - 1$  do begin
     $i \leftarrow n - k$ 
    for  $j = i + 1$  to  $n$  do
       $a_{in+1} \leftarrow a_{in+1} - a_{ij} \times a_{jn+1}$  .....  $\gamma$ 
    end

```

代入文の実行される回数はそれぞれ,

$$\alpha = \sum_{i=1}^n (n - i + 1) = \frac{1}{2}n^2 + \frac{1}{2}n$$

$$\beta = \sum_{i=1}^n \sum_{k=i+1}^n (n - i + 1) = \frac{1}{3}n^3 - \frac{1}{3}n$$

$$\gamma = \sum_{k=1}^{n-1} (n - (n - k)) = \frac{1}{2}n^2 - \frac{1}{2}n$$

したがって, 乗除算の回数は

$$E_4(n) = \alpha + \beta + \gamma = \frac{1}{3}n^3 + n^2 - \frac{1}{3}n = O(n^3)$$

練習問題

1. $E_4(3)$ はいくつか? (答: 17)

C. LU 分解

次のような手順で計算する.

- (1) A を, 下三角行列 L と, 対角要素がすべて 1 の上三角行列 U に分解する (LU 分解).

$$A = LU$$

$$L = \begin{pmatrix} l_{11} & & & 0 \\ \vdots & \ddots & & \\ l_{n1} & \cdots & l_{nn} & \end{pmatrix}$$

$$U = \begin{pmatrix} 1 & u_{12} & \cdots & u_{1n} \\ & \ddots & \ddots & \vdots \\ & & \ddots & u_{n-1n} \\ 0 & & & 1 \end{pmatrix}$$

- (2) $Ly = b$ を満たす y を求める.
 (3) $Ux = y$ を満たす x を求める.

LU 分解の方法

$A = (a_{ij})$ が 1 次なら

$$\begin{aligned} L &= (a_{11}) \\ U &= (1) \end{aligned}$$

である. 2 次以上なら, A を基本変形により

$$\begin{pmatrix} 1 & a_{12}^* & \cdots & a_{1n}^* \\ 0 & & & \\ \vdots & & A^* & \\ 0 & & & \end{pmatrix}$$

の形に変形する. ここで,

$$A^* = \begin{pmatrix} a_{22}^* & \cdots & a_{2n}^* \\ \vdots & \ddots & \vdots \\ a_{n2}^* & \cdots & a_{nn}^* \end{pmatrix}$$

また,

$$\begin{aligned} a_{1j}^* &= a_{1j}/a_{11} & (2 \leq j \leq n) \\ a_{ij}^* &= a_{ij} - a_{i1} \frac{a_{1j}}{a_{11}} & (2 \leq i \leq n, 2 \leq j \leq n) \end{aligned}$$

である.

A^* に対してさらに LU 分解を行い

$$A^* = L^*U^*$$

となったとする. この時,

$$\begin{aligned} L &= \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & & & \\ \vdots & & L^* & \\ a_{n1} & & & \end{pmatrix} \\ U &= \begin{pmatrix} 1 & a_{12}^* & \cdots & a_{1n}^* \\ 0 & & & \\ \vdots & & U^* & \\ 0 & & & \end{pmatrix} \end{aligned}$$

とおくと, LU の各成分 b_{ij} は

$$\begin{aligned} b_{i1} &= a_{i1} & (1 \leq i \leq n) \\ b_{1j} &= a_{11}a_{1j}^* = a_{1j} & (2 \leq j \leq n) \\ b_{ij} &= a_{i1}a_{1j}^* + a_{ij}^* & (2 \leq i \leq n, 2 \leq j \leq n) \\ &= a_{i1} \frac{a_{1j}}{a_{11}} + a_{ij} - a_{i1} \frac{a_{1j}}{a_{11}} \\ &= a_{ij} \end{aligned}$$

となり, A の各成分に一致する. したがって, L と U は A の LU 分解である.

LU 分解のプログラム

```

for  $i = 1$  to  $n - 1$  do begin
  for  $j = i + 1$  to  $n$  do
     $a_{ij} \leftarrow a_{ij} / a_{ii}$  .....  $\alpha$ 
  for  $k = i + 1$  to  $n$  do
    for  $j = i + 1$  to  $n$  do
       $a_{kj} \leftarrow a_{kj} - a_{ki} \times a_{ij}$  .....  $\beta$ 
end

```

代入文の実行される回数はそれぞれ,

$$\alpha = \sum_{i=1}^{n-1} (n-i) = \frac{1}{2}n^2 - \frac{1}{2}n$$

$$\beta = \sum_{i=1}^{n-1} \sum_{k=i+1}^n (n-i) = \frac{1}{3}n^3 - \frac{1}{2}n^2 + \frac{1}{6}n$$

したがって, 乗除算の回数は

$$LU_1(n) = \alpha + \beta = \frac{1}{3}n^3 - \frac{1}{3}n$$

 $Ly = b$ を解く方法

漸化式

$$y_1 = \frac{b_1}{l_{11}}$$

$$y_i = \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij} y_j \right) \quad (1 < i \leq n)$$

により, y を求める.

 $Ly = b$ を解くプログラム

```

for  $i = 1$  to  $n$  do begin
  for  $j = 1$  to  $i - 1$  do
     $a_{in+1} \leftarrow a_{in+1} - a_{ij} \times a_{jn+1}$  .....  $\alpha$ 
   $a_{in+1} \leftarrow a_{in+1} / a_{ii}$  .....  $\beta$ 
end

```

代入文の実行される回数はそれぞれ,

$$\alpha = \sum_{i=1}^n (i-1) = \frac{1}{2}n^2 - \frac{1}{2}n$$

$$\beta = n$$

したがって, 乗除算の回数は

$$LU_2(n) = \alpha + \beta = \frac{1}{2}n^2 + \frac{1}{2}n$$

$Ux = y$ を解く方法

ガウスの方法の後退代入と同一の方法で解ける. したがって, 乗除算の回数は

$$LU_3(n) = \frac{1}{2}n^2 - \frac{1}{2}n$$

よって, 全体の乗除算の回数は

$$\begin{aligned} E_5(n) &= LU_1(n) + LU_2(n) + LU_3(n) \\ &= \frac{1}{3}n^3 + n^2 - \frac{1}{3}n = O(n^3) \end{aligned}$$

であり, ガウスの方法の乗除算回数に一致する.

D. コレスキー法

A が対称行列の場合, より計算量の少ない方法がある.

命題 3

対称行列 A を基本変形により

$$\begin{pmatrix} 1 & a_{12}^* & \cdots & a_{1n}^* \\ 0 & & & \\ \vdots & & A^* & \\ 0 & & & \end{pmatrix}$$

の形に変形したとする (ただし, $a_{11} \neq 0$ と仮定する). ここで,

$$\begin{aligned} A^* &= \begin{pmatrix} a_{22}^* & \cdots & a_{2n}^* \\ \vdots & \ddots & \vdots \\ a_{n2}^* & \cdots & a_{nn}^* \end{pmatrix} \\ a_{1j}^* &= a_{1j}/a_{11} & (2 \leq j \leq n) \\ a_{ij}^* &= a_{ij} - a_{i1} \frac{a_{1j}}{a_{11}} & (2 \leq i \leq n, 2 \leq j \leq n) \end{aligned}$$

である. このとき, A^* は対称行列になる.

証明

$2 \leq i \leq n, 2 \leq j \leq n$ なる i, j について

$$\begin{aligned} a_{ij}^* &= a_{ij} - a_{i1} \frac{a_{1j}}{a_{11}} \\ &= a_{ji} - a_{1i} \frac{a_{j1}}{a_{11}} \\ &= a_{ji} - a_{j1} \frac{a_{1i}}{a_{11}} \\ &= a_{ji}^* \end{aligned}$$

コレスキーの方法による LU 分解のプログラム

```

for  $i = 1$  to  $n - 1$  do begin
  for  $j = i + 1$  to  $n$  do
     $a_{ij} \leftarrow a_{ji} / a_{ii}$  .....  $\alpha$ 
  for  $k = i + 1$  to  $n$  do
    for  $j = i + 1$  to  $k$  do
       $a_{kj} \leftarrow a_{kj} - a_{ki} \times a_{ij}$  .....  $\beta$ 
  end

```

A^* の下三角の部分だけを計算している.

代入文の実行される回数はそれぞれ,

$$\alpha = \sum_{i=1}^{n-1} (n-i) = \frac{1}{2}n^2 - \frac{1}{2}n$$

$$\beta = \sum_{i=1}^{n-1} \sum_{k=i+1}^n (k-i) = \frac{1}{6}n^3 - \frac{1}{6}n$$

したがって, 乗除算の回数は

$$LU'_1(n) = \alpha + \beta = \frac{1}{6}n^3 + \frac{1}{2}n^2 - \frac{2}{3}n$$

E. 仮定と if 文

プログラムの拡張.

if 文

if C **then** S_1 **else** S_2

- C は条件式.
- S_1, S_2 は文.
- C が成り立てば S_1 を, 成り立たなければ S_2 を実行する.

if C **then** S_1

- C は条件式.
- S_1 は文.
- C が成り立てば S_1 を実行する.

go to 文

go to L

- L はラベル.
- ラベル L の付いている文に制御を移す.

ラベル付き文

$$L : S$$

- L はラベル, S はラベル無しの文.

反復文の拡張

$$\text{for } i = p \text{ downto } q \text{ do } S$$

while 文

$$\text{while } C \text{ do } S$$

例

行の交換を行う.

```

if  $a_{ii} \neq 0$  then
  go to set
for  $k = i$  to  $n$  do
  if  $a_{ki} \neq 0$  then begin
    for  $j = 1$  to  $n + 1$  do begin
       $t \leftarrow a_{ij}$ 
       $a_{ij} \leftarrow a_{kj}$ 
       $a_{kj} \leftarrow t$ 
    end
  end
  go to set
end
go to alarm
set :

```

2.3 文字列照合

比較が実行される回数を計算量とする.

A. 素朴な方法

```

t ← 1
while t ≤ N - m + 1 do begin
  for j = m downto 1 do
    if T(t + j - 1) ≠ p(j) then ..... α
    go to continue
  go to atta
continue: t ← t + 1
end

```

まず, **for** 文の中についてだけ考える. $T(t + j - 1) = p(j)$ が成立する確率を p とすると, α がちょうど n 回実行される確率 $p(n)$ は

$$p(n) = \begin{cases} p^{n-1}(1-p) & (1 \leq n < m) \\ p^{m-1} & (n = m) \\ 0 & (n > m) \end{cases}$$

である. したがって, **for** 文中については

$$\begin{aligned}
\text{ave } \alpha &= \sum_{n=1}^{\infty} np(n) \\
&= \sum_{n=1}^{m-1} (np^{n-1}(1-p)) + mp^{m-1} \\
&= (1-p)(1 + 2p + \cdots + (m-1)p^{m-2}) + mp^{m-1} \\
&= (1-p)(1 + p + \cdots + p^{m-1})' + mp^{m-1} \\
&= (1-p) \left(\frac{1-p^m}{1-p} \right)' + mp^{m-1} \\
&= \frac{-mp^{m-1}(1-p) + (1-p^m)}{1-p} + mp^{m-1} \\
&= \frac{1-p^m}{1-p}
\end{aligned}$$

atta の処理のあと戻ってきて続きを行うとすると, **for** 文自身が $N - m + 1$ 回繰り返される. したがって,

$$\begin{aligned}
\min \alpha &= N - m + 1 \\
\text{ave } \alpha &= \frac{1-p^m}{1-p} (N - m + 1) \\
\max \alpha &= m(N - m + 1)
\end{aligned}$$

B. Boyer-Moore の方法

簡易 BM 法のプログラム

```

t ← 1
while t ≤ N - m + 1 do begin
  for j = m downto 1 do
    if T(t + j - 1) ≠ p(j) then ..... α
      go to continue
  go to atta
continue:  t ← t + d(T(t + m - 1))
end

```

for 文内での α の平均実行回数は

$$\frac{1 - p^m}{1 - p}$$

である。 $d(T(t + m - 1))$ の平均値を d とすると、

$$\begin{aligned} \min \alpha &= \frac{N - m + 1}{m} \\ \text{ave } \alpha &= \frac{1 - p^m}{1 - p} \frac{N - m + 1}{d} \\ \max \alpha &= m(N - m + 1) \end{aligned}$$

BM 法のプログラム

```

t ← 1
while t ≤ N - m + 1 do begin
  for j = m downto 1 do
    if T(t + j - 1) ≠ p(j) then ..... α
      go to continue
  go to atta
continue:  if j = m
            then t ← t + d(T(t + m - 1))
            else t ← t + dd(m - j)
end

```

Chapter 3

アルゴリズムの高速化

3.1 高速化の手法

A. 分割統治法

1章の最後の3次多項式の計算法の応用.

n 次多項式の積

二つの多項式, $\sum_{i=0}^n a_i x^i$, $\sum_{i=0}^n b_i x^i$ の積 $\sum_{i=0}^{2n} c_i x^i$ を求める.

直接的な方法

$$\begin{aligned} c_l &= \sum_{i+j=l} a_i b_j \quad (0 \leq l \leq 2n) \\ &= \begin{cases} a_0 b_l + a_1 b_{l-1} + \cdots + a_l b_0 & (l \leq n) \\ a_{l-n} b_n + a_{l-n+1} b_{n-1} + \cdots + a_n b_{l-n} & (l > n) \end{cases} \end{aligned}$$

乗除算と加減算の回数は

$$\begin{aligned} P_0^\times(n) &= \sum_{l=0}^n (l+1) + \sum_{l=n+1}^{2n} (2n-l+1) \\ &= (n+1)^2 \\ P_0^+(n) &= \sum_{l=0}^n l + \sum_{l=n+1}^{2n} (2n-l) \\ &= n^2 \end{aligned}$$

分割統治法 (divide and conquer)

$n = 2m + 1$ とする. X, A_1, A_0, B_1, B_0 を次のようにおく.

$$X = x^{m+1}$$

$$\begin{aligned}
 A_1 &= \sum_{i=0}^m a_{m+i+1} x^i \\
 A_0 &= \sum_{i=0}^m a_i x^i \\
 B_1 &= \sum_{i=0}^m b_{m+i+1} x^i \\
 B_0 &= \sum_{i=0}^m b_i x^i
 \end{aligned}$$

すると、求める式は次のように変形できる。

$$\begin{aligned}
 \sum_{l=0}^{2n} c_l x^l &= \sum_{i=0}^{2m+1} a_i x^i \times \sum_{i=0}^{2m+1} b_i x^i \\
 &= (A_0 + A_1 X) \times (B_0 + B_1 X) \\
 &= A_0 B_0 + (A_1 B_0 + A_0 B_1) X + (A_1 B_1) X^2
 \end{aligned}$$

これを次のようにして計算する。 $A_0 B_0$ などの積は直接的な方法で求めるとする。

	乗算	加減算
(1) $C_0 = A_0 B_0$ の計算	$P_0^\times(m)$	$P_0^+(m)$
(2) $C_1 = A_1 B_1$ の計算	$P_0^\times(m)$	$P_0^+(m)$
(3) $C_2 = A_0 + A_1$ の計算	0	$m + 1$
(4) $C_3 = B_0 + B_1$ の計算	0	$m + 1$
(5) $C_4 = C_2 C_3$ の計算	$P_0^\times(m)$	$P_0^+(m)$
(6) $C_5 = C_4 - C_0$ の計算	0	$2m + 1$
(7) $C_6 = C_5 - C_1$ の計算	0	$2m + 1$
(8) $C_1 X^2 + C_6 X + C_0$ の同類項の整理	0	$2m$
合計	$3P_0^\times(m)$	$3P_0^+(m) + 8m + 4$

よって、乗除算と加減算の回数は

$$\begin{aligned}
 P_1^\times(n) &= 3P_0^\times((n-1)/2) \\
 &= \frac{3}{4}(n+1)^2 = O(n^2) \\
 P_1^+(n) &= 3P_0^+(\frac{n-1}{2}) + 8 \frac{n-1}{2} + 4 \\
 &= \frac{3}{4}n^2 + \frac{5}{2}n + \frac{3}{4} = O(n^2)
 \end{aligned}$$

B. 再帰的応用

$A_0 B_0$ などの積の計算に分割統治法を再帰的に応用する。ただし、 $n = 2m + 1 = 2^k - 1$ とする。

	乗算	加減算
(1) $C_0 = A_0 B_0$ の計算	$P_2^\times(m)$	$P_2^+(m)$
(2) $C_1 = A_1 B_1$ の計算	$P_2^\times(m)$	$P_2^+(m)$
(3) $C_2 = A_1 + A_0$ の計算	0	$m + 1$
(4) $C_3 = B_1 + B_0$ の計算	0	$m + 1$
(5) $C_4 = C_2 C_3$ の計算	$P_2^\times(m)$	$P_2^+(m)$
(6) $C_5 = C_4 - C_1$ の計算	0	$2m + 1$
(7) $C_6 = C_5 - C_0$ の計算	0	$2m + 1$
(8) $C_1 X^2 + C_6 X + C_0$ の同類項の整理	0	$2m$
合計	$3P_2^\times(m)$	$3P_2^+(m) + 8m + 4$

乗除算の回数は

$$\begin{aligned} P_2^\times(0) &= 1 \\ P_2^\times(2^k - 1) &= 3P_2^\times(2^{k-1} - 1) \end{aligned}$$

加減算の回数は

$$\begin{aligned} P_2^+(0) &= 0 \\ P_2^+(2^k - 1) &= 3P_2^+(2^{k-1} - 1) + 8(2^{k-1} - 1) + 4 \\ &= 3P_2^+(2^{k-1} - 1) + 4(2^k - 1) \end{aligned}$$

$P'^\times(k) = P_2^\times(2^k - 1)$, $P'^+(k) = P_2^+(2^k - 1)$ とおくと

$$\begin{aligned} P'^\times(0) &= 1 \\ P'^\times(k) &= 3P'^\times(k-1) \\ P'^+(0) &= 0 \\ P'^+(k) &= 3P'^+(k-1) + 4(2^k - 1) \end{aligned}$$

これを解いて

$$\begin{aligned} P'^\times(k) &= 3^k \\ P'^+(k) &= \sum_{i=1}^k 3^{k-i} \cdot 4(2^i - 1) \\ &= 4 \cdot 3^k \left(\sum_{i=1}^k \frac{2^i}{3^i} - \sum_{i=1}^k \frac{1}{3^i} \right) \\ &= 4 \cdot 3^k \left(\frac{1 - (2/3)^{k+1}}{1 - 2/3} - \frac{1 - (1/3)^{k+1}}{1 - 1/3} \right) \\ &= 2 \cdot 3^{k+1} - 4 \cdot 2^{k+1} + 2 \end{aligned}$$

定理 1

乗除算の回数 $P_2^\times(2^k - 1)$ と加減算の回数 $P_2^+(2^k - 1)$ は次のようになる。

$$\begin{aligned} P_2^\times(2^k - 1) &= 3^k \\ P_2^+(2^k - 1) &= 2 \cdot 3^{k+1} - 4 \cdot 2^{k+1} + 2 \end{aligned}$$

また ($k = \log_2(n+1)$ より)

$$\begin{aligned} P_2^\times(n) &= 3^{\log_2(n+1)} \\ &= (n+1)^{\log_2 3} = (n+1)^{1.58496\dots} \\ &= O(n^{1.58496\dots}) \end{aligned}$$

3.2 高速フーリエ変換

a. フーリエ級数

実数値関数 $f(t)$ が区間 $(-\pi, \pi)$ で積分可能なとき

$$\begin{aligned} f(t) &= a_0 + \sum_{k=1}^{\infty} (a_k \cos kt + b_k \sin kt) \\ a_0 &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) dt \\ a_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \cos kt dt \\ b_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \sin kt dt \end{aligned}$$

を $f(t)$ のフーリエ級数という.

ここで, $c_0 = a_0$, $c_k = (a_k - ib_k)/2$, $c_{-k} = \overline{c_k}$ とおくと

$$\begin{aligned} f(t) &= \sum_{k=-\infty}^{\infty} c_k e^{ikt} \\ c_k &= \frac{1}{2\pi} \int_{-\pi}^{\pi} f(t) e^{-ikt} dt \end{aligned}$$

である. これを複素型のフーリエ級数という.

$f(t)$ の周期が T であるとする

$$\begin{aligned} f(t) &= \sum_{k=-\infty}^{\infty} c_k e^{2\pi i \frac{kt}{T}} \\ c_k &= \frac{1}{T} \int_0^T f(t) e^{-2\pi i \frac{kt}{T}} dt \end{aligned}$$

A. 離散型フーリエ変換

関数 $f(t)$ が周期 T の周期関数であり, T/n の等間隔分点 t_m ($0 \leq m < n$) に対する $f(t_m)$ の値を y_m とする. このとき, 第 k フーリエ係数 a_k (複素型フーリエ級数での c_k) を数値積分すると,

$$a_k = \frac{1}{n} \sum_{m=0}^{n-1} y_m e^{-2\pi i \frac{mk}{n}}$$

ω を 1 の原始 n 乗根のうちの一つ $e^{\frac{2\pi i}{n}}$ とすると

$$a_k = \frac{1}{n} \sum_{m=0}^{n-1} y_m \omega^{-mk} \quad (1)$$

となる. これを y_m について解くと

$$y_m = \sum_{k=0}^{n-1} a_k \omega^{mk} \quad (2)$$

式(2)により, n 次元ベクトル $\mathbf{a} = (a_0, a_1, \dots, a_{n-1})$ から n 次元ベクトル $\mathbf{y} = (y_0, y_1, \dots, y_{n-1})$ を求めることを, n 次の離散型フーリエ変換, 式(1)により, n 次元ベクトル $\mathbf{y} = (y_0, y_1, \dots, y_{n-1})$ から n 次元ベクトル $\mathbf{a} = (a_0, a_1, \dots, a_{n-1})$ を求めることを, n 次の離散型逆フーリエ変換という.

離散型フーリエ変換は結局与えられた n 次ベクトルを係数ベクトルとする $n-1$ 次多項式

$$f(x) = a_0 + a_1 x + \dots + a_{n-1} x^{n-1}$$

について, n 次ベクトル

$$\mathbf{y} = (f(\omega^0), f(\omega^1), \dots, f(\omega^{n-1}))$$

あるいは W を n 次正方行列 ($\omega^{(i-1)(j-1)}$) とすると

$$\mathbf{y} = W\mathbf{a}$$

を求めることに相当する.

直接的方法

$\omega^0, \omega^1, \dots, \omega^{n-1}$ は求められているものとして, n 個の $n-1$ 次多項式 $f(\omega^0), f(\omega^1), \dots, f(\omega^{n-1})$ をホーナーの方法で計算すると, 乗除算, 加減算ともに $n(n-1)$ 回である.

折りたたみによる方法

$n = 2u$ の時,

$$\begin{aligned} y_m &= \sum_{k=0}^{n-1} a_k \omega^{mk} \\ &= \sum_{k=0}^{u-1} a_k \omega^{mk} + \sum_{k=0}^{u-1} a_{k+u} \omega^{m(k+u)} \\ &= \sum_{k=0}^{u-1} (a_k + \omega^{mu} a_{k+u}) \omega^{mk} \quad (0 \leq m < n) \end{aligned}$$

ω^{mu} は1の平方根だから $+1$ または -1 である. したがって, 係数 $a_k + \omega^{mu} a_{k+u}$ は, すべての y_m について $a_k + 1 \cdot a_{k+u}$ または $a_k - 1 \cdot a_{k+u}$ だから, 乗除算 $n/2$ 回 ($1 \cdot a_{k+u}$ の計算), 加減算 n 回で求められる.

したがって, n 次ベクトル \mathbf{y} を求めるには, $b = a_k + 1 \cdot a_{k+u}$, $c = a_k - 1 \cdot a_{k+u}$ とすると,

$$\begin{aligned} g(x) &= b_0 + b_1 x + \dots + b_{u-1} x^{u-1} \\ h(x) &= c_0 + c_1 x + \dots + c_{u-1} x^{u-1} \end{aligned}$$

について, 次の二つの u 次ベクトル

$$\begin{aligned} &(g(\omega^0), g(\omega^1), \dots, g(\omega^{u-1})) \\ &(h(\omega^0), h(\omega^1), \dots, h(\omega^{u-1})) \end{aligned}$$

を求めて、それを並べ換えて \mathbf{y} を求めれば良い.

$$\mathbf{y} = (g(\omega^0), h(\omega^0), g(\omega^1), h(\omega^1), \dots, g(\omega^{u-1}), h(\omega^{u-1}))$$

二つの u 次ベクトルをホーナーの方法で求めるとすると、 n 個の $n/2 - 1$ 次多項式を計算することになるから、全体で乗除算 $n/2 + n(n/2 - 1) = n(n - 1)/2$ 回、加減算 $n + n(n/2 - 1) = n^2/2$ 回である.

B. 高速フーリエ変換 (FFT)

n が 2 のべき乗 2^k のとき、折りたたみによる方法を再帰的に応用する. すなわち、二つの $n/2$ 次ベクトル $(g(\omega^0), g(\omega^1), \dots, g(\omega^{u-1}))$, $(h(\omega^0), h(\omega^1), \dots, h(\omega^{u-1}))$ を同様の方法で求める.

すると、乗除算回数 $F^\times(n)$ と加減算回数 $F^+(n)$ は、次の漸化式で表される.

$$\begin{aligned} F^\times(1) &= 0 \\ F^\times(2^k) &= 2^{k-1} + 2F^\times(2^{k-1}) \\ F^+(1) &= 0 \\ F^+(2^k) &= 2^k + 2F^+(2^{k-1}) \end{aligned}$$

これを解いて

$$\begin{aligned} F^\times(2^k) &= 2^{k-1}k \\ &= \frac{1}{2}n \log_2 n \\ F^+(2^k) &= 2^k k \\ &= n \log_2 n \end{aligned}$$

C. 高速フーリエ変換の応用

逆フーリエ変換

逆フーリエ変換は与えられた n 次ベクトル

$$\mathbf{y} = (f(\omega^0), f(\omega^1), \dots, f(\omega^{n-1}))$$

から $n - 1$ 次多項式

$$f(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1}$$

の係数ベクトル \mathbf{a} を決定することに相当する.

定理 3

$$y_m = \sum_{k=0}^{n-1} a_k \omega^{mk}$$

の時

$$a_k = \frac{1}{n} \sum_{m=0}^{n-1} y_m \omega^{-mk}$$

証明

$$\begin{aligned}
\text{右辺} &= \frac{1}{n} \sum_{m=0}^{n-1} \left(\sum_{l=0}^{n-1} a_l \omega^{ml} \right) \omega^{-mk} \\
&= \frac{1}{n} \sum_{l=0}^{n-1} \left(\sum_{m=0}^{n-1} \omega^{m(l-k)} \right) a_l \\
&= a_k
\end{aligned}$$

逆フーリエ変換は $\omega' = \omega^{-1}$ とおくと,

$$a_k = \sum_{m=0}^{n-1} \frac{y_m}{n} \omega'^{mk}$$

だから, 1 の原始 n 乗根 ω' に基づく, n 次元ベクトル $\mathbf{y} = (y_0/n, y_1/n, \dots, y_{n-1}/n)$ のフーリエ変換とみなせる. したがって, FFT を利用できる.

 n 次多項式の積

2 つの $n = 2^k - 1$ 次多項式 $f(x)$, $g(x)$ の積を求める.

これは FFT を利用して次のように行なえる.

- (1) 0 の係数を付け加えて, $f(x)$, $g(x)$ を $2^{k+1} - 1$ 次式として考える. 1 の原始 2^{k+1} 乗根 ω について, $f(x)$, $g(x)$ に対してフーリエ変換を行ない, 次の二つの 2^{k+1} 次ベクトルを求める

$$\begin{aligned}
&\left(f(\omega^0), f(\omega^1), \dots, f(\omega^{2^{k+1}-1}) \right) \\
&\left(g(\omega^0), g(\omega^1), \dots, g(\omega^{2^{k+1}-1}) \right)
\end{aligned}$$

- (2) 積 $h_j = f(\omega^j) \cdot g(\omega^j)$ を $0 \leq j \leq 2^{k+1} - 1$ について求め, 次の 2^{k+1} 次ベクトルを得る

$$\mathbf{h} = (h_0, h_1, \dots, h_{2^{k+1}-1})$$

- (3) \mathbf{h} に対して, 逆フーリエ変換を行なう.

$$(c_0, c_1, \dots, c_{2^{k+1}-1})$$

これは, $h(x) = f(x) \cdot g(x)$ の係数ベクトルになっている

この計算量は

	乗算回数	加減算回数
(1)	$2 \cdot 2^k(k+1)$	$2 \cdot 2^{k+1}(k+1)$
(2)	2^{k+1}	0
(3)	$2^{k+1} + 2^k(k+1)$	$2^{k+1}(k+1)$
合計	$2^{k+1}(3k+7)$	$3 \cdot 2^{k+1}(k+1)$

よって

$$\begin{aligned}
P_3^\times(2^k - 1) &= 2^{k+1}(3k+7) \\
P_3^+(2^k - 1) &= 3 \cdot 2^{k+1}(k+1)
\end{aligned}$$

また ($k = \log_2(n+1)$ より)

$$\begin{aligned} P_3^\times(n) &= 2 \cdot 2^{\log_2(n+1)}(3 \log_2(n+1) + 7) \\ &= 2(n+1)(3 \log_2(n+1) + 7) \\ &= O(n \log n) \end{aligned}$$

k	$P_1^\times(2^k - 1)$	$P_2^\times(2^k - 1)$	$P_3^\times(2^k - 1)$
1	3	3	40
2	12	9	104
5	768	243	1,408
10	786,432	59,049	75,776
15	805,306,368	14,348,907	3,407,872
20	824,633,720,832	3,486,784,401	140,509,184
30	864,691,128,455,135,232	205,891,132,094,649	208,305,913,856

例

$1 + x + x^2 + x^3$ と $-1 + x - x^2 + x^3$ の積を求める. $(1, 1, 1, 1, 0, 0, 0, 0)$ のフーリエ変換は

$$\begin{aligned} (4, 1 + i + e^{\pi i/4} + e^{3\pi i/4}, 0, 1 - i + e^{3\pi i/4} + e^{9\pi i/4}, 0, \\ 1 + i + e^{5\pi i/4} + e^{15\pi i/4}, 0, 1 - i + e^{7\pi i/4} + e^{21\pi i/4}) \end{aligned}$$

であり, $(-1, 1, -1, 1, 0, 0, 0, 0)$ のフーリエ変換は

$$\begin{aligned} (0, -1 - i + e^{\pi i/4} + e^{3\pi i/4}, 0, -1 + i + e^{3\pi i/4} + e^{9\pi i/4}, -4, \\ -1 - i + e^{5\pi i/4} + e^{15\pi i/4}, 0, -1 + i + e^{7\pi i/4} + e^{21\pi i/4}) \end{aligned}$$

である. 積は

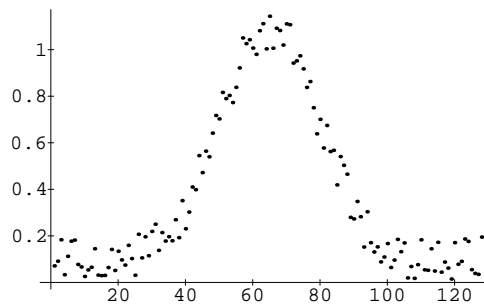
$$(0, -2 - 2i, 0, -2 + 2i, 0, -2 - 2i, 0, -2 + 2i)$$

となり, これを逆フーリエ変換した結果は

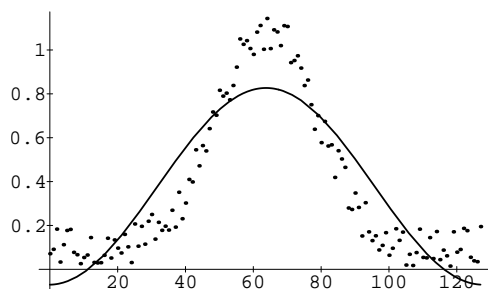
$$(-1, 0, -1, 0, 1, 0, 1, 0)$$

となり, $-1 - x^2 + x^4 + x^6$ が得られる.

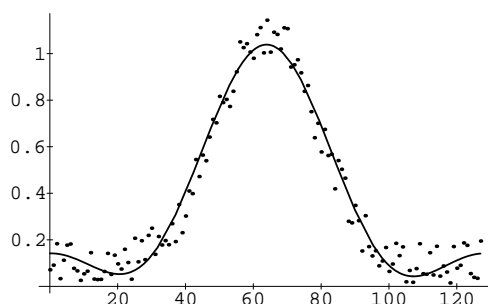
パラメータの推定



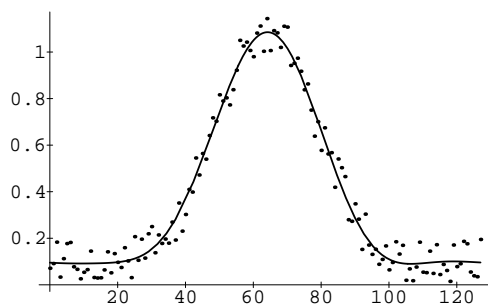
$$0.378358 - 0.448369 \cos\left(\frac{2\pi x}{128}\right) + 0.00532727 \sin\left(\frac{2\pi x}{128}\right)$$



$$0.378358 - 0.448369 \cos\left(\frac{2\pi x}{128}\right) + 0.00532727 \sin\left(\frac{2\pi x}{128}\right) \\ + 0.211793 \cos\left(\frac{4\pi x}{128}\right) + 0.000307794 \sin\left(\frac{4\pi x}{128}\right)$$



$$0.378358 - 0.448369 \cos\left(\frac{2\pi x}{128}\right) + 0.00532727 \sin\left(\frac{2\pi x}{128}\right) \\ + 0.211793 \cos\left(\frac{4\pi x}{128}\right) + 0.000307794 \sin\left(\frac{4\pi x}{128}\right) \\ - 0.0466242 \cos\left(\frac{6\pi x}{128}\right) - 0.00717946 \sin\left(\frac{6\pi x}{128}\right)$$



3.3 行列の積の計算

A. ウィノグラードの方法

B. シュトラッセンの方法

Appendix A

付録

A.1 漸化式の解法

A. タイプ 1

次のタイプの漸化式

$$a_n = a_{n-1} + c_n \quad (n \geq 1)$$

解

$$a_n = a_0 + \sum_{k=1}^n c_k$$

例

$$\begin{cases} a_0 = 1 \\ a_n = a_{n-1} + 2n \quad (n \geq 1) \end{cases}$$

は次のようになる.

$$\begin{aligned} a_n &= 1 + \sum_{k=1}^n 2k \\ &= 1 + 2 \sum_{k=1}^n k \\ &= 1 + 2 \cdot \frac{n(n+1)}{2} \\ &= n^2 + n + 1 \end{aligned}$$

B. タイプ 2

次のタイプの漸化式

$$a_n = ba_{n-1} + c \quad (n \geq 1)$$

解

$a_n = b^n a'_n$ とおくと

$$\begin{cases} a'_0 = a_0 \\ a'_n = a'_{n-1} + \frac{c}{b^n} \quad (n \geq 1) \end{cases}$$

であり、タイプ 1 に帰着し次のように解ける.

$$a'_n = a_0 + \sum_{k=1}^n \frac{c}{b^k}$$

よって、 a_n は

$$\begin{aligned} a_n &= a_0 b^n + c \sum_{k=1}^n b^{n-k} \\ &= a_0 b^n + c(1 + b + \cdots + b^{n-1}) \\ &= a_0 b^n + \frac{c(b^n - 1)}{b - 1} \end{aligned}$$

例

$$\begin{cases} a_0 = 1 \\ a_n = 2a_{n-1} + 1 \quad (n \geq 1) \end{cases}$$

は次のようになる.

$$\begin{aligned} a_n &= 1 \cdot 2^n + \frac{1 \cdot (2^n - 1)}{2 - 1} \\ &= 2^{n+1} - 1 \end{aligned}$$

C. タイプ 3

次のタイプの漸化式

$$a_n = ba_{n-1} + ca_{n-2} \quad (n \geq 2)$$

解

例