多倍長計算ソフトウェアの開発

吉原　郁夫 [a)]・本田　詩織 [b)]・坂本　亜衣 [b)]・山森　一人 [c)]・棟朝　雅晴 [d)]

Development of Multiple Precision Arithmetic Software

Ikuo YOSHIHARA,　Shiori HONDA,　Ai SAKAMOTO,
Kunihito YAMAMORI,　Masaharu MUNETOMO

Abstract

It is necessary to employ “multiple precision arithmetic” for computing long digit numbers, because numerical representation of computers is usually limited. This paper aims at making prototype software to compute more than one million digit numbers. A long digit number is divided into 2^n short digit numbers, each of which can be calculated by ordinal double precision arithmetic units. The key technique of “multiple precision arithmetic” is based on fast Fourier transform and convolution theorem. The prototype program is verified from the viewpoint of correctness of calculation up to 4×10^6 digits and speed up ratio vs theoretical value. The program is applied to calculation of a 10^7 or more digit π.

Keywords:　Multiple precision arithmetic, fast Fourier transform, convolution

1.　はじめに

計算機の数値表現には限りがあり、あまり桁の大きな数を一度に扱うことはできない。普通の計算機が直接扱うことができるのは、単精度浮動小数型は 32 ビット、倍精度浮動小数型は 64 ビットである [1)]。それらは、10 進数の場合、約 7 桁、約 16 桁に相当する。それ以上の長い桁の計算を行う場合、多倍長計算を行う必要がある。これは、1 つの大きな数を数桁ずつに分割し、処理をする方法である。

多倍長計算は、科学技術計算でしばしば必要となる。そのため、4 倍精度や 8 倍精度演算などのサブルーチンが多く出回っている [2)]。本研究では、4 倍精度や 8 倍精度演算よりもさらに長い、100 万桁以上の多倍長計算ソフトウェアの開発を目的としている。本稿では、多倍長計算ソフトウェアの開発について述べ、例題として実装した円周率の計算については、別稿で坂本が述べる。

2.　多倍長計算

多倍長数aが、p桁ずつ分割される場合、

$$a = \sum_{i=0}^{n-1} (a_i \times 10^{pi}) \qquad (1)$$

と表される。これを、10^p進法、ワード数nの多倍長数aと呼ぶ。

2.1 加減算

加算は、筆算と同様の方法で処理することができる。

10^p進法、ワード数nの多倍長数a、b、cがあるとし、$c = a + b$の計算を行う場合、式は以下のようになる。

$$\begin{aligned} c_0 &= a_0 + b_0 \\ c_1 &= a_1 + b_1 \\ &\vdots \\ c_{n-1} &= a_{n-1} + b_{n-1} \end{aligned}$$

このとき、$c_m (0 \le m \le n-1)$に入っている桁数がpを超えていた場合、桁を揃えるために、正規化が必要となる。

$$c_m = c_m (mod\ 10^p)$$

$$c_{m+1} = c_{m+1} + \frac{c_m}{10^p}$$

減算も、加算と同じように計算することができる。$c = a - b$の計算を行う場合の式は、

$$\begin{aligned} c_0 &= a_0 - b_0 \\ c_1 &= a_1 - b_1 \\ &\vdots \\ c_{n-1} &= a_{n-1} - b_{n-1} \end{aligned}$$

a) 情報システム工学科　教授
b) 情報システム工学科　学生
c) 情報システム工学科　准教授
d) 北海道大学大学院情報科学研究科　准教授

となる。正規化は、

$$c_m = c_m + 10^p$$
$$c_{m+1} = c_{m+1} - 1$$

となる。

加減算の計算量は、$O(n)$である。

2.2 乗算

乗算は、以下のように扱う。

10^p進法、ワード数$N = 2n$の多倍長数a、b、cがあるとする。a、bは、

$$a_m = b_m = 0 \quad (n \le m < 2n)$$

とする。このとき、$c = a \times b$の計算を行う式は、

$$\begin{aligned}
c_0 &= a_0 b_0 \\
c_1 &= a_0 b_1 + a_1 b_0 \\
c_2 &= a_0 b_2 + a_1 b_1 + a_2 b_0 \\
&\vdots \\
c_{n-1} &= a_0 b_{n-1} + a_1 b_{n-2} + \cdots + a_{n-1} b_0 \\
&\vdots \\
c_{2n-3} &= a_{n-1} b_{n-2} + a_{n-2} b_{n-1} \\
c_{2n-2} &= a_{n-1} b_{n-1} \\
c_{2n-1} &= 0
\end{aligned}$$

となり、c の各項は次のように表すことができる。

$$c_{i+j} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j \qquad (2)$$

式(2) は、$k = i + j$とすることで、次のように書き換えられる。

$$c_k = \sum_{i=0}^{N-1} a_i b_{k-i} \quad (k = 0,1,\cdots,N-1) \qquad (3)$$

bの添字$k - i$が負の場合は、Nを加える。この計算は、畳み込み演算と呼ばれる。計算量は、複素計算で$O(N^2)$である。

畳み込み演算は、フーリエ変換を用いることにより、高速に計算することができる。

2.3 除算

除算b/aは、まずaの逆数$1/a$を求め、その値にbをかけて算出する。

逆数$1/a$は、方程式$a - 1/x = 0$の解であり、Newton法

$$x_{n+1} = x_n + x_n(1 - a x_n) \quad (a > 0) \qquad (4)$$

による反復計算で求める。

こうすることにより、除算を、乗算と加減算だけで計算することができる [4]。

3. 高速フーリエ変換を用いた多倍長計算

第2.2節で述べたように、多倍長の乗算は畳み込み演算であり、畳み込み演算は、フーリエ変換を用いることで、周波数空間での内積演算に帰着させることができる。

多倍長数f、g、hがあるとし、

$$h = f \times g \qquad (5)$$

の計算を行うとき、f、g、hをフーリエ変換した値をF、G、Hとすると、

$$H_k = F_k G_k \qquad (6)$$

と計算することができ、その逆変換を行うことで、乗算の結果hを求めることができる。

フーリエ変換は高速フーリエ変換で演算回数を減らすことができ、計算量は、複素計算で$O(N \log N)$である。

3.1 離散フーリエ変換

離散フーリエ変換(Discrete Fourier Transform:DFT)とは、フーリエ変換の式

$$F(y) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x y} \, dx \quad \left(i = \sqrt{-1}\right) \qquad (7)$$

を離散化したものである。

式 (7) の $f(x)$ を離散化した N 個の標本点 $\{f(0), f(1), \cdots, f(N-1)\}$が与えられたとき、

$$\begin{aligned}
F(k) &= \sum_{n=0}^{N-1} f(n) e^{-\frac{2\pi i}{N} k n} \\
&= \sum_{n=0}^{N-1} f(n) \, W^{kn} \\
&(k = 0,1,\cdots,N-1)
\end{aligned} \qquad (8)$$

を離散フーリエ変換という。ここで、Wは捻り因子であり、

$$W = e^{-\frac{2\pi i}{N}} \qquad (9)$$

である [5]。

3.2 高速フーリエ変換

離散フーリエ変換には、$N(N-1)$回の複素加算とN^2回の複素乗算が必要である。この計算量を減らすために、CooleyおよびTukeyが考え出したアルゴリズムが、高速フーリエ変換(Fast Fourier Transform:FFT)である [6]。

本研究では、通常使われている基数2の高速フーリエ変換を用いる。基数2の高速フーリエ変換は、標本長Nが2のべき乗であるとする。

式(8)は、kの偶数項と奇数項に分けることにより、標本長$N/2$の2つの離散フーリエ変換に分解することができる。

$$F(2r) = \sum_{n=0}^{\frac{N}{2}-1} \left\{ f(n) + f\left(n+\frac{N}{2}\right) \right\} W^{2rn} \qquad (10)$$

$$F(2r+1) = \sum_{n=0}^{\frac{N}{2}-1} \left\{ f(n) - f\left(n+\frac{N}{2}\right) \right\} W^{n} W^{2rn} \qquad (11)$$

$$\left(r = 0,1,\cdots,\frac{N}{2}-1\right)$$

標本長$N/2$の離散フーリエ変換が2式あり、その計算量は$N(N-1)/2$回の加減算と$(N/2)^2$回の乗算で、総計算量$(3N^2-N)/2$回となる。よって、式を2つに分けることによって計算量は約半分になる。この展開を標本長2になるまでの繰り返し回数は$\log N$である。繰り返し1回あたりの計算量は、$N/2$回の加算と、$N/2$回の減算、N回の乗算で、計$2N$回であるから、高速フーリエ変換の計算量は、複素計算で$2N\log N$回である。これを実数計算に換算すると、$7N\log N$回となる。

このアルゴリズムは、図1のような信号流れ図で表すことができる。しかし、この計算法では、出力データの並び順が、入力データと異なるため、本研究では、図2に示すような、入力と出力のデータの並び順が変わらない、Stockham型高速フーリエ変換アルゴリズムを用いる [7]。

3.3 逆離散フーリエ変換

離散フーリエ変換された$F(k)$の逆変換$f(n)$は、以下のようになる。

$$f(n) = \frac{1}{N}\sum_{k=0}^{N-1} F(k) W^{-nk} \qquad (12)$$
$$(n = 0,1,\cdots,N-1)$$

この変換を逆離散フーリエ変換(Inverse Discrete Fourier Transform:IDFT)という。

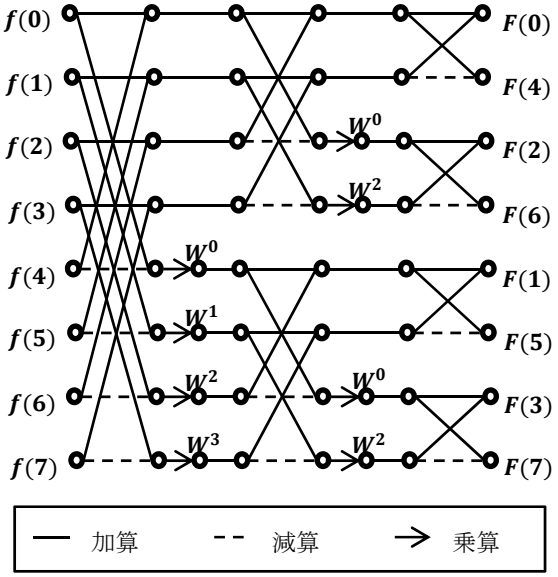

図 1. N=8 の FFT の信号流れ図

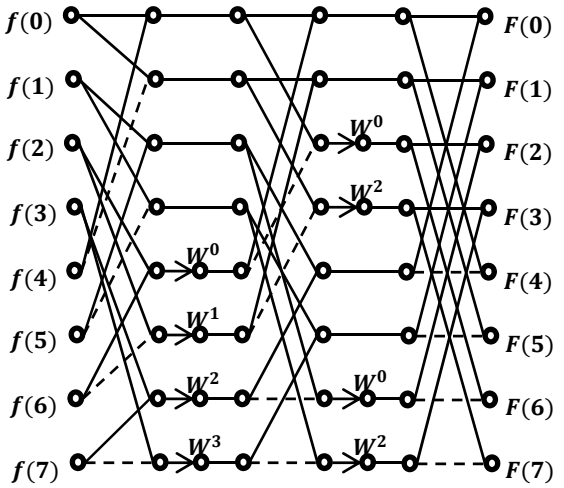

図 2. N=8 の Stockham 型 FFT の信号流れ図

3.4 高速フーリエ変換を用いた多倍長乗算

以下に、高速フーリエ変換を用いた多倍長乗算の計算手順を示す。

step-1 多倍長数$\boldsymbol{f}$、$\boldsymbol{g}$を高速フーリエ変換し、それぞれ$\boldsymbol{F}$、$\boldsymbol{G}$とおく。

$$\begin{array}{ccc} & \mathcal{F} & \\ f & \rightarrow & F \\ & \mathcal{F} & \\ g & \rightarrow & G \end{array} \qquad (13)$$

step-2 $\boldsymbol{F}$と$\boldsymbol{G}$の積を、$\boldsymbol{H}$とおく。

$$H = F * G \tag{14}$$

step-3 $\boldsymbol{H}$を逆高速フーリエ変換し、$\boldsymbol{h}$とおく。これにより、$\boldsymbol{f}$と$\boldsymbol{g}$の多倍長乗算$\boldsymbol{h}$が求まる。

$$H \xrightarrow{\mathcal{F}^{-1}} h \tag{15}$$

4. 検証実験

4.1 高速フーリエ変換を用いた多倍長計算の高速化の検証実験

高速フーリエ変換を用いた多倍長計算が、理論計算量に沿って高速化されていることを検証する。

(1) 実験方法

適当に与えた2つの多倍長数の乗算を、畳み込み演算を用いた多倍長計算(以下:筆算法)プログラムと、高速フーリエ変換を用いた多倍長計算(以下:FFT法)プログラムで行う。標本長(多倍長数のワード数)毎にそれぞれの計算時間を測り、その加速率を求める。

$$\text{加速率} = \frac{\text{筆算法での処理時間}}{\text{FFT 法での処理時間}}$$

その理論値と実測値の比が一定ならば、FFT法は理論計算量に沿って高速化されているといえる。

オーバーフローを回避するために、10^2～10^5進法とする。

(2) 加速率

まず、筆算法とFFT法の加速率の実測値と理論値を求める。

加速率(実測値)

実測値は、以下に示すように、筆算法の計算時間からFFT法の計算時間を割った値とする。

$$\text{加速率(実測値)} = \frac{\text{筆算法の計算時間(秒)}}{\text{FFT法の計算時間(秒)}} \tag{16}$$

加速率(理論値)

筆算法の計算量は、第2.2節より、$O(N^2)$であり、FFT法の計算量は、第3.2節より、$O(N\log N)$である。よって、筆算法の計算量をFFT法の計算量で割った値である加速率は、式(17)の通りであり、これの理論値と実験値がそれぞれ一定となることが示されればよい。

$$\begin{aligned}\text{加速率(理論値)} &= \frac{\text{筆算法の計算量}}{\text{FFT法の計算量}} \\ &= \frac{O(N^2)}{O(N\log N)}\end{aligned} \tag{17}$$

以下に、標本長に対する、加速率(理論値)を示す。

表1. 各標本長に対する加速率(理論値)

標本長	加速率(理論値)
64	35.4
128	60.7
256	106.3
512	189.0
1024	340.2
2048	618.5
4096	1133.9
8192	2093.3
16384	3887.6
32768	7256.9
65536	13606.6
131072	25612.5
262144	48379.1
524288	91665.6

(3) 実験結果

10^5進法の場合の、計算時間と加速率(実測値)を表1、表2、および図3に示す。同図から加速率の実測値と理論値が比例することが確認できた。

表 2. 10^5進法の場合の計算時間と加速率(実測値)

標本長	計算時間(秒)		加速率(実測値)
	筆算法	FFT 法	
64	0.000021	0.000709	0.03
128	0.000066	0.001295	0.05
256	0.000233	0.002381	0.10
512	0.000881	0.004556	0.19
1024	0.003462	0.008919	0.39
2048	0.013577	0.017673	0.77
4096	0.053983	0.035995	1.50
8192	0.215449	0.073726	2.92
16384	0.866797	0.150388	5.76
32768	3.466491	0.308473	11.24
65536	13.848631	0.635505	21.79
131072	56.752992	1.289477	44.01
262144	238.253806	2.628806	90.63
524288	943.757652	5.333200	176.96

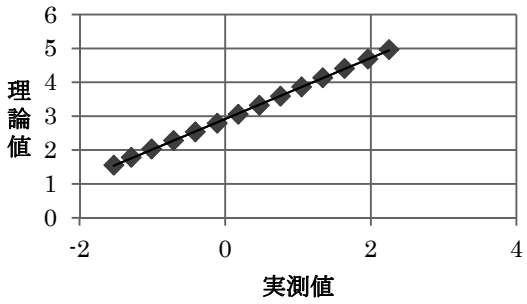

図 3. 10^5進法の場合の加速率

(4) 考察

実験結果より、標本長2048の計算までは筆算法の方が速いが、それ以上は、FFT法の方が速くなった。また、図3より、加速率の実測値と理論値が比例しているので、今回作成した高速フーリエ変換を用いた乗算プログラムは、理論通り高速化されていることが検証できた。

4.2 計算可能桁数の検証実験

正確な計算が可能な桁数を実験的に求める。

(1) 実験方法

標本長を増加しながら、適当に与えた2つの多倍長数の乗算を行い、正確な計算が可能な範囲を求める。

オーバーフローを回避するために、10^2～10^5進法とする。

(2) 実験結果

以下に、正確な計算が可能な限界標本長と総桁数をまとめた表を示す。

表3. 計算可能な限界標本長と総桁数

記数法	限界標本長	総桁数
10^2進法	1048576	2097152
10^3進法	1048576	3145728
10^4進法	1048576	4194304
10^5進法	524288	1572864

(3)考察

実験結果より、今回作成したプログラムで、約419万桁の計算まで正確に行えることが分かった。

5. おわりに

100万桁以上の多倍長計算ソフトウェアの開発を目的とし、まず、畳み込み演算を用いた計算法から、離散フーリエ変換を用いた計算法を導いた。次に、離散フーリエ変換を高速化する方法を示した。そして、畳み込み演算法で作成したプログラムと、高速フーリエ変換を用いた計算法で作成したプログラムの、高速化や計算可能範囲の検証実験を行った。

実験を行った結果、標本長2048以上の乗算から、畳み込み演算法よりも、高速フーリエ変換を用いた計算法の方が、計算時間が短くなることがわかった。また、約419万桁の計算を理論計算量通りに高速に、かつ、正確に行うことができた。

今後の課題として、計算可能な桁数範囲を広げる工夫や、高速フーリエ変換の更なる高速化などが挙げられる。

謝辞

本研究の一部は、北海道大学情報基盤センターとの共同研究の一環として行われたもので、関係各位に感謝いたします

参考文献

1) ㈱システム計画研究所編：C 言語によるプログラミング-基礎編-第 2 版、株式会社オーム社、2001.

2) 小武守恒、藤井昭宏、長谷川秀彦、西田晃：”反復法ライブラリ向け 4 倍精度演算の実装と SSE2 を用いた高速化”、情報処理学会論文誌「コンピューティングシステム」、Vol.1、No.1、pp.73-84、2008.

3) 高橋大介、金田康正：”分散メモリ型並列計算機による円周率の 515 億桁計算”、情報処理学会論文誌、Vol.39、No.7、pp.2074-2083、1998.

4) 大浦拓哉：”円周率公式の改良と高速多倍長計算の実装”、日本応用数理学会論文誌、Vol.9、No.4、pp.165-1724、 1999.

5) 佐川雅彦、貴家仁志：高速フーリエ変換とその応用, 昭晃堂、1992.

6) J. W. Cooley, and J. W. Tukey：”An Algorithm for the Machine Calculation of Complex Fourier Series”, Mathematics of Computation, Vol.19, No.1965, pp.297-301, 1965.

7) D. H. Bailey：”A High-Performance FFT Algorithm for Vector Supercomputers”, International Journal of Supercomputer Applications, Vol.2, No.1, pp.82-87, 1988.