ハイパフォーマンス
コンピューティング　65－6
（１９９７．３．６）

# FFTによる高精度数の乗算

平山　弘

神奈川工科大学　機械システム工学科

二つの高精度数の乗算には、FFT を使うと効率的に計算できることが知られている。この計算には、通常の倍精度実数を使う。そのため、打ち切り誤差の影響を受ける。

本論文では、この打ち切り誤差を調べ、この方法が使える範囲を解析的、数値的方法で求めた。この領域は、僅かな浮動小数点の形式の違いで、すなわち、計算機の違いで、大きく変化することがわかる。

# Multiplication of High Precision Numbers by FFT

Hiroshi Hirayama

Kanagawa Institute of Technology

It is well known that the multiplication of two high precision numbers can be done very effectively by using FFT. This method can be carried on double precision floating-point numbers. In this methods, the calculation can not avoid truncation errors. We must investigate this errors for using of the FFT algorithm.

In this paper, this errors are considered by analytical and numerical methods and gives applicable range of the FFT algorithm. These range change very much when slightly difference of floating-point number format.

## 1. はじめに

計算機の高速化と低価格化に伴って、今まで困難であった高精度数（多倍長数）の計算が容易に行うことができるようになってきている。高精度数でプログラムを作成すると、桁落ちの生じる部分の検出などの作業がなくすことができるため、プログラム開発は容易になり、信頼性も上がる。

高精度数の演算で問題になるのは計算時間である。高速アルゴリズムは不可欠である。その中で、高精度数の乗算は、高速フーリエ変換（FFT）を使うと高速に行うことができることが知られている。$n$桁の数値の乗算には通常$n^2$のオーダーの計算時間が必要であるが、FFTを使うと$n\log n$のオーダーで計算できる。$n$が十分大きな数の場合、FFTを使った計算法は非常に効率的である。多くの計算機で、10進数で約1000桁を越えれば、FFTアルゴリズムを使った計算法が、通常の計算法より高速に計算できると言われている。

FFTを使って、高精度数の積を計算するには、通常の二つの計算方法が考えられる。有限体を使って計算する方法と浮動小数点を使って計算する方法である。有限体を使った計算方法の場合、整数演算だけを使うので、誤差の入らない厳密な計算結果を与える。一方、浮動小数点演算を使う方法は、三角関数などの近似値を扱うために厳密な計算ができない。しかし、最終結果は整数であることが分かるので、誤差が十分に小さいならば、丸め処理によって厳密な計算が可能である。

これまでの多くの計算機では、最も精度の高い整数は、32ビットであり、浮動小数点数は、64ビットである。大型計算機では、128ビットの浮動小数点数もある。このため、計算精度の高い浮動小数点数を使った計算方法が多く使われてきた。最近のマイクロプロセッサでは、64ビット精度の整数を扱うことが出来るので有限体を使った計算法も行われ始めている。

本論文では、浮動小数点数によるFFTを使った高精度数の乗算で生じる誤差を評価し、FFTによる乗算法の利用できる限界を調べた。

## 2. FFTによる高精度数の乗算法

高精度整数$x$を$b$進数$m$桁で表現する。すなわち

$$x = \sum_{k=0}^{m-1} x_k b^k \tag{2.1}$$

とする。このとき高精度整数$y$との積$z$は、

$$z_j = \sum_{k=0}^{j} x_k y_{j-k} \tag{2.2}$$

となる。この計算は畳み込み演算と呼ばれ高速フーリエ変換によって効率よく計算できることが知られている[1]。この場合、倍精度実数を利用したFFTを利用する計算方法がよく使われる。FFTにも多くの計算法があり、今回利用したプログラムには、実数用で基数2と8のFFTプログラムがある。FFTには、複素数用と実数用があるがここで扱うデータが実数であるので、実数用を利用した。実数用は、同じデータ数であるとき複素数用の約2倍高速である。基数2のFFTは、プログラムは短く、よく使われるアルゴリズムである。Bergland[2]って発表された基数8のプログラムを使えばさらに高速に計算できることが知られている。

このようなFFTの計算では、計算の途中で打切り誤差が入る。この誤差は、最後の丸め

処理によって厳密な値になるためには、次のような関係式を満たさなければならない。この式は Henrici [1] によって導かれたものである。$b$進数$m$桁の数値を厳密に乗算できるには、計算精度の相対誤差を$\eta$(マシン・イプシロン)とすると

$$\eta \leq \frac{1}{192m^2(2\log_2 m+7)b^2} \tag{2.3}$$

を満たさなければならない。この式から基数$b$が大きいほど要求精度が高くなることがわかる。桁数$m$も大ききなると要求精度が高くなることがわかる。この式は、相対誤差の２乗以上の高次の項を省略する方法で、誤差を評価しているので、ほぼ十分条件になる。現実にはもっと低い精度でも計算可能である。たとえば、上の式で$b=10000$、$\eta=2.22\times10^{-16}$（IEEE 方式の倍精度浮動小数点）の場合、計算可能桁数は 428 桁となる。実際には 100 万桁の数も計算可能である。

(2.3)式より精密な評価を得るために、区間演算などを試みたが、若干適用範囲が増加したが、実用的な範囲にはならなかった。

(2.3)式を誤差の評価式と見たとき、誤差は、基数$b$の２乗、対数部分を省略すると桁数$m$の約２乗に比例することがわかる。この場合の基数$b$とは、１語の中に入る最大の整数という意味になるので、誤差が最大になるのは、各語に$b-1$の数値を入れたとき最大の誤差になることがわかる。$b=10000$とすれば、各語に 9999 を入れたとき最大の誤差が生じることになる。$b=2^n$のときは、各語に$2^n-1$を入れたとき最大の誤差が生じる。

また、このような値を入れたときの計算結果は容易にわかるので、誤差を求めるのは容易である。３節にその結果を示す。

## 3. 最大誤差の計算

前節で述べたことを確かめるために、 IEEE 方式の倍精度浮動小数点をもつ SUN4 を使って誤差を計算した。$b=100$と固定して、桁数$m$を増やす。このときの誤差は、次のようになる。

| m | 8 | 16 | 32 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|
| error | 1.24E-13 | 3.41E-13 | 1.36E-12 | 3.87E-12 | 1.09E-11 | 2.64E-11 |

このときの計算方法は、基数２の FFT である。桁数$m$が２倍になると、誤差は約４倍とり、上の仮定が成り立つ。

次に、桁数$m=256$と固定して、基数$b$を変化させる。このとき、誤差は次のようになる。このときの計算法は、基数２の FFT である。

| b | 8 | 16 | 32 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|
| error | 1.73E-11 | 8.00E-11 | 3.27E-10 | 1.34E-09 | 5.56E-09 | 2.14E-08 |

この計算結果から、誤差は、基数$b$のかなり正確に２乗に比例することがわかる。この誤差が、0.5 より小さければ、実数を使った FFT による高精度の数値の乗算が厳密に行うことができる。このようにして、限界を求めると次のようになる。

### （１） IEEE 方式の倍精度浮動小数点の場合

ここでは、このような計算機として SUN4 を使った。FFT として基数２と基数８のプログラムを利用した。基数$b$として、２のべき乗と 10 のべき乗を使った。その結果は次のよ

うである。

表１．基数２の FFT の場合

| 基数$b$ | 誤差 | 計算可能桁数$m$ | 計算可能桁数(10 進換算) |
|---|---|---|---|
| 1000 | - | >262144 | >786432 |
| 10000 | 0. 174 | 131072 | 524288 |
| 100000 | 0. 500 | 8192 | 40960 |
| 1000000 | 0. 344 | 128 | 768 |

| 基数$b$ | 誤差 | 計算可能桁数$m$ | 計算可能桁数(10 進換算) |
|---|---|---|---|
| 4096 | - | >262144 | >946958 |
| 8192 | 0. 402 | 262144 | 1025871 |
| 16384 | 0. 471 | 131072 | 552392 |
| 32768 | 0. 406 | 65536 | 295925 |
| 65536 | 0. 211 | 8192 | 39457 |
| 131072 | 0. 438 | 4096 | 20961 |
| 262144 | 0. 359 | 1024 | 5548 |
| 524288 | 0. 223 | 256 | 1464 |
| 1048576 | 0. 383 | 128 | 771 |
| 2097152 | 0. 219 | 32 | 202 |
| 4194304 | 0. 312 | 16 | 106 |
| 8388608 | 0. 281 | 8 | 55. 4 |
| 166777216 | 0. 250 | 4 | 28. 9 |

表２．基数８の FFT の場合

| 基数$b$ | 誤差 | 計算可能桁数$m$ | 計算可能桁数(10 進換算) |
|---|---|---|---|
| 1000 | - | >262144 | >786432 |
| 10000 | - | >262144 | >1048576 |
| 100000 | 0. 281 | 16384 | 81920 |
| 1000000 | 0. 203 | 128 | 768 |

| 基数$b$ | 誤差 | 計算可能桁数$m$ | 計算可能桁数(10 進換算) |
|---|---|---|---|
| 16384 | - | >262144 | >1104784 |
| 32768 | 0. 164 | 131072 | 591849 |
| 65536 | 0. 312 | 65536 | 315652 |
| 131072 | 0. 266 | 16384 | 83845 |
| 524288 | 0. 250 | 1024 | 5857 |
| 1048576 | 0. 266 | 256 | 1541 |
| 2097152 | 0. 125 | 32 | 202 |
| 4194304 | 0. 219 | 16 | 106 |
| 8388608 | 0. 203 | 4 | 28 |
| 166777216 | - | <4 | <28 |

### （2） 拡張 IEEE 方式の倍精度浮動小数点の場合

拡張 IEEE 方式の倍精度浮動小数点を使うコンピュータとは、レジスタでは 80 ビット、メモリの中では 64 ビットで倍精度実数を表すコンピュータである。ここでは、このような計算機として SUN3 を使った。FFT として基数２と基数８のプログラムを利用した。基数$b$として、10 のべき乗を使った。その結果は次のようである。この結果は、利用した計算機だけでなくコンパイラにも依存する。

表３．基数２の FFT の場合

| 基数$b$ | 誤差 | 計算可能桁数$m$ | 計算可能桁数(10 進換算) |
|---|---|---|---|
| 10000 | - | >262144 | >1048576 |
| 100000 | 0.422 | 8192 | 40960 |
| 1000000 | 0.328 | 128 | 768 |
| 10000000 | 0.500 | 8 | 64 |

表４．基数８の FFT の場合

| 基数$b$ | 誤差 | 計算可能桁数$m$ | 計算可能桁数(10 進換算) |
|---|---|---|---|
| 10000 | - | >262144 | >1048576 |
| 100000 | 0.323 | 16384 | 81920 |
| 1000000 | 0.438 | 256 | 1536 |

### (3) IBM 方式の倍精度浮動小数点の場合

IBM 方式の倍精度浮動小数点を使うコンピュータとして、ここでは、HITAC M-880 を使った。FFT として基数２と基数８のプログラムを利用した。基数$b$として、10 のべき乗を使った。その結果は次のようである。

表 5. 基数２の FFT の場合

| 基数$b$ | 誤差 | 計算可能桁数$m$ | 計算可能桁数(10 進換算) |
|---|---|---|---|
| 1000 | - | >262144 | >786432 |
| 10000 | 0.144 | 16384 | 65536 |
| 100000 | 0.309 | 2048 | 10240 |
| 1000000 | 0.430 | 128 | 768 |
| 10000000 | 0.188 | 4 | 32 |

表 6. 基数８の FFT の場合

| 基数$b$ | 誤差 | 計算可能桁数$m$ | 計算可能桁数(10 進換算) |
|---|---|---|---|
| 10000 | - | >262144 | >1048576 |
| 100000 | 0.383 | 32768 | 163840 |
| 1000000 | 0.156 | 256 | 1536 |
| 10000000 | 0.234 | 4 | 28 |

計算効率から考えると、基数$b$はなるべく大きく取り、桁数$m$はなるべく小さくした方が能率的である。このようにして上の表を見ると、基数８の FFT を利用する方が効率的であることがわかる。基数８の FFT は、単に計算時間が短いだけでなく、誤差も小さいことがわかる。

レジスタが 80 ビットである場合、すべて 64 ビットである場合にくらべ、大きな数値を小さい桁数$m$で計算できる。

仮数部分のビット数が長い、IBM 方式の浮動小数点を使う計算機が、この計算の場合、

かなり効率的である。

基数$b$が小さいときは、FFT のプログラムの制限から、実際の誤差を求めていないが、上の表からどの程度の計算が可能か推定できる。(2.3)式から、この分母が一定であるはずであるから、基数$b$が1桁小さな値になれば、$\log_2 m$を含む項を一定だとすると桁数$m$はおよそ10倍にできる。このように考えれば、およその値が推定できる。

上の誤差は、最悪の場合の誤差であり、通常は、この 10 分の1程度の誤差である。したがって、円周率の計算のようにある特別な数値だけしか計算しない場合には、上の限界を超えた大きな桁数の数値を掛け算できる場合がある。

## 4. 終わりに

FFTを使って高精度整数の乗算を行う場合のその適用限界を決定する実際的な方法を述べた。この方法は、誤差解析の式から最も誤差の大きくなる場合を求め、その場合の誤差を実際に計算し、その適用限界を計算する方法である。この方法によれば、同じ64ビットの浮動小数点数でも、浮動小数点の表現方法が異なれば、その適用限界が変化することがわかる。ほんの僅かな形式の違いが、適用限界の大きな違いとして現れる。

FFTを使った高精度計算プログラムは、参考文献の[4]を見てください。また、ガウス体を使った整数論的FFTを使ったプログラムについては、APFLOATを参照してください。
(http://www.hut.fi/~mtommila/apfloat)

### 参考文献


[1] Henrici P., Applied and Computational Complex Analysis, Vol. 3, Chap. 13, John Wiley & Sons, New York(1986)

[2] Bergland G. D., A Radix-Eight Fast Fourier Transform Subroutine for Real-Valued Series, IEEE Trans. A. E., AU17.2, pp.138-144

[3] 森、名取、鳥居, 数値計算（岩波講座情報科学１８）, 5章, 岩波書店(1982)

[4] 平山　弘, 多倍長計算プログラムパッケージMPPACKの利用の手引き, 東大計算センター(1992)