社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－ARC－177 (5)
2008－HPC－114 (5)
2008／3／5

Baileyアルゴリズムによる多倍長演算の性能評価

石川 正[†] 濱口信行[††]

多倍長演算は，数学・物理学のある特殊なシミュレーションに必要なばかりか，今後の大規模シミュレーションにおいても必要になってくると考えられる．多倍長変数を倍精度変数の和で表現するBailey アルゴリズムを一般化して演算量を評価した．また，多倍長を配列整数を使って演算する方法（配列整数方式）に関しても演算量の評価を行い，Bailey アルゴリズムと配列整数方式のと性能比較を行った．これらの多倍長演算用プログラムをいくつかの悪条件の例に応用しその結果を報告する．

Estimation of multiple-precision operation using Bailey's algorithm

TADASHI ISHIKAWA[†] and NOBUYUKI HAMAGUCHI[††]

We implement a multiple-precision arithmatic operations using the Bailey's algorithm where the multiple-precision number is treated as a sum of double precison numbers. We compare Bailey's algorithm with the algorithm of the array-integer operation(HMLIB). we show that the alogorithm of array-integer operation is remarkably efficient for long-multiple precision. We also show few application to ustable problems using both algorithms.

1. はじめに

多倍長計算は，数学や物理学などのごく一部の分野において必要なものと思われてきた．今後大規模計算において，長時間の高速実行する際丸め誤差の蓄積などを考えると一般的にシミュレーションを進めていく上では，これまでの倍精度計算より長い仮数部を必要する多倍長計算が必要不可欠になってくると考えられる．

多倍長演算を行うためにBailey らは倍精度変数を並べて行うアルゴリズムを提案している[1]．また藤原は多倍長計算環境を構築しており[2]，多倍長計算の必要性について認識が深まってきている．

濱口は配列整数演算を使って加算・乗算などの多倍長計算を可能とすると共に，内部に整数配列を用意した精度が高くかつ高い性能を有する総和と内積のライブラリ (HMLIB) を作った（以下，配列整数方式と呼ぶ)[3]．

このライブラリを使って素粒子物理学の摂動論シミュレーションにおいて桁落ち情報を得ることによって，プログラム自体の数値的な詳細な分析が行えるようになり，プログラムの信頼性を高めると同時に，これまで困難と思われていた問題を取り組むことができるようになってきている[4]．

本論文においては，これらの論文にあるアルゴリズムを $2n(n=2,\cdots,8)$ 倍精度まで拡張し演算回数の評価を行い，配列演算方式との演算回数の評価についても明らかにする．

Intel-Xeon QuadCore(Xeon X5355 2.66GHz) において測定したものである．コンパイラは Intel 9.1 である．

2. Baileyアルゴリズムでの演算量

$2n$ 倍精度の変数を n 個の倍精度変数の和で表現する．

$$a=\sum_{i=1}^{n} a_i$$

一般には倍精度変数 a,b の加算，乗算の結果は 2 つの倍精度変数で正確に求めることができる．

加算および乗算の演算量を求めるのに，以下の 2 つの場合を考える．仮数部の長さを N_{bit} とする．

(1) 完全保証の場合：nN_{bit} までの精度を保証する，つまり完全に保証する場合．

(2) 準完全保証の場合：$(n-1)N_{bit}$ までの精度は完全に保証する．実際には，最後の数ビットについては丸め誤差によって欠落するが演算量が

† 高エネルギー加速器研究機構
High Energy Accelarator Research Organization
†† 日立製作所 ソフトウェア事業部
Hitachi Ltd., Software Division

少なくなる場合.

Bailey アルゴリズムでは次の基本的な演算処理 two-sum, quick-two-sum, two-prod を元にアルゴリズムを作っている. Power アーキテクチャの場合は, fused-multi-add(FMA) 命令があり, two-prod-fma を使うことができる. これらの基本的な演算の flops 数を表 1 に示す.

表 1 Bailey アルゴリズムで用いる基本演算量

	flop 数
two-sum	6
quick-two-sum	3
two-prod	17
two-prod-fma	3

2.1 $2n$ 倍精度加算の演算量

準完全保証および完全保証の場合の演算量をまとめると以下の表のようになる.

表 2 準完全保証の場合の加算演算量

	flops 数
two-sum	$n(n-1)/2$
quick-two-sum	$n-1$
通常の sum	n
合計の flop 数	$3n^2+n-3$

表 3 完全保証の場合の加算演算量

	flops 数
two-sum	$n(n+1)/2$
quick-two-sum	$2n$
通常の sum	$n-1$
合計の flop 数	$3n^2+10n-1$

$2(n+1)$ 倍精度加減算の準完全保証の flop 数は, $2n^2+7n+1$ であり, $2n$ 倍精度加減算の完全保証の flops 数より, $3n-2$ 少ない.

2.2 $2n$ 倍精度乗算の演算量

FMA 命令がある場合には, 次の表 4 および 5 の演算量が必要となる.

表 4 準完全保証の場合の乗算演算量

	flops 数
two-prod-fma	$n(n-1)/2$
two-sum	$n(n-1)(n-2)/3$
quick-two-sum	$n-1$
通常の sum	n^2
合計の flop 数	$(4n^3-7n^2+11n-6)/2$

FMA 命令がない場合には, two-fma-prod が two-prod に変わるだけであり, 以下の表 6 となる.

表 5 完全保証の場合の乗算演算量

	flops 数
two-prod-fma	$n(n+1)/2$
two-sum	$n(n-1)(n+1)/3$
quick-two-sum	$2n$
通常の sum	$(n+3)(n-1)$
合計の flop 数	$(4n^3+5n^2+15n-6)/2$

表 6 FMA 命令がないときの乗算演算量

	flops 数
準完全保証の場合	$(4n^3+7n^2-3n-6)/2$
完全保証の場合	$(4n^3+19n^2+29n-6)/2$

FMA 命令あるなしに関わらず, $2(n+1)$ 倍精度乗算の準完全保証の場合の flop 数は, $2n$ 倍精度演算の完全保証の場合の flop 数より, $3n-4$ 数少なくなるのが特徴である. いずれにしても flop 数は n が増えると, two-sum の 3 乗の増加が支配的になる.

FMA 命令が有効に働くのは n が小さい場合であり, $n=2$ の準完全保証においては, FMA 命令がない場合に比べ 1.92 倍 flop 数 が少なくなる.

FMA 命令の有効性については, 永井らの論文にて検証されている[5].

3. 配列整数方式の場合の演算量

配列整数方式は, 浮動小数点演算ではなく整数演算を用いるため, 浮動小数点演算とは若干異なり, 以下の様にして演算量を定める. ソースプログラムでの分岐演算, 論理演算 (判定, マスク, シフト), 整数加減算, 乗除算の数とし分岐では各分岐一様に実行されるとし, 小数点が発生すると切り上げて整数値とする.

配列整数方式の場合, 以下の注意点がある.

(1) n 倍精度と仮数部のビット数 m には, $m=32\times n-16$ の関係がある.

(2) 整数 i, j の加算は, 桁上がり値 ic とすると $i+j+ic$ の計算をする事になる.

(3) 整数 i, j の乗算は, 桁上がり値 ic とすると $i\times j+ic$ の計算をする事になる.

仮数部のビット数を m, 多倍長整数を表現する整数配列の 1 要素で使用するビット数を k とする.

A, B は m+1 ビット整数であるので, $\ell=[m/k]$+1 とする.

浮動小数点演算を整数演算で行う場合, 処理は符号部, 指数部, 仮数部に分けて行う.

処理としては,

(1) 整数 $\Leftrightarrow$ 実数変換処理　演算量 $=15\ell$

(2) 大小判定処理　加算のみで演算量 $= 2\ell + 8$

(3) 結果の丸め判定　演算量 $= 5\ell$

(4) 仮数部の多倍長整数演算処理

ここで (1)-(3) の処理は加算，乗算処理とも演算量は n の一次式となる．(4) の演算量は加算と乗算で以下の様になる．

加算　(4) の演算量は整数加算 $(i + j + ic)$ を ℓ 回となる．桁上がり判定をいれて 3ℓ．合計して，加算は演算量 $25\ell + 8$ となる．

乗算　(4) の演算量は整数乗算 $(i \times j + ic)$ を $(\ell^2 + 3\ell - 2)/2$ 回となる．桁上がり判定をいれて $3(\ell^2 + 3\ell - 2)/2$ 回となる．合計して，乗算は演算量 $3(\ell^2 + 3\ell - 2)/2 + 20\ell$ となる．

以上から，演算量は加算は n の一次式，乗算は n の二次式となる．

以下に多倍長整数演算の測定を図 1 と 2 に示した．

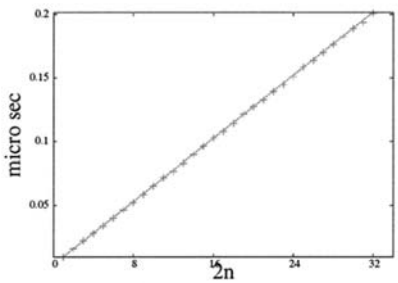

図 1　1 演算あたりの整数加算性能

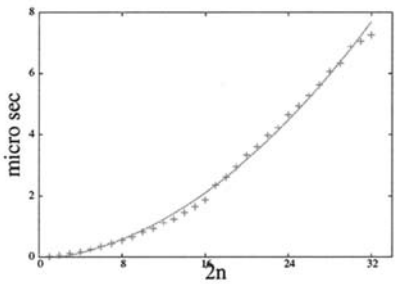

図 2　1 演算あたりの整数乗算性能

4. 多倍長演算のテストと評価

テストプログラムを $2n = 32$ 倍精度まで試作し，検証したのち実測をした．ビット数は Bailey アルゴリズムで $n \times (52 + 1)$ ビット，配列整数方式では，$64 \times n - 16 + 1$ ビットであり，数が違っている．具体的なビット数を表 7 に示す．

表 7　Bailey アルゴリズムと配列整数方式の仮数部ビット数の例

	Bailey	配列整数方式
4 倍	106	113
8 倍	212	241
16 倍	424	497

演算量 10M あたりの性能を図 3 と 4 に示す．

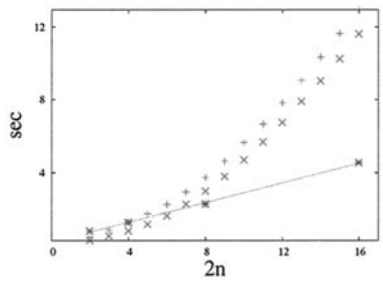

図 3　演算量 10M あたりの加算性能（+:Bailey の完全保証，×:Bailey の準完全保証，∗：配列整数方式（完全保証））

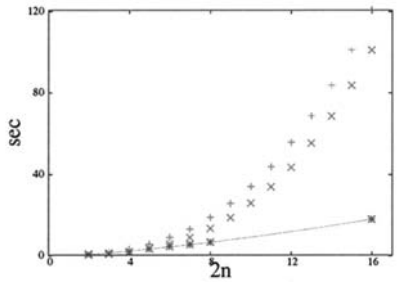

図 4　演算量 10M あたりの乗算性能（+:Bailey の完全保証，×:Bailey の準完全保証，∗：配列整数方式（完全保証））

加算，乗算ともに $n = 2$ では Bailey アルゴリズムによるものの性能が優れている，n が 4 以上になると配列整数方式が性能が良くなる．これは，演算量が Bailey アルゴリズムでは加算で n の 2 次式，乗算で n の 3 次式に対し，配列整数演算方式は加算で n の 1 次式，乗算で n の 2 次式になるからである．

5. 配列整数方式における精度保証総和・内積の評価

精度保証総和・内積演算では，一般の加算，乗算処理と異なり，求める結果が 1 つ，変数の値が 0 の場合処理が不要になるため，大小判定処理，各要素の演算

結果の丸め判定が不要になるため，演算量は見積り易い．今，変数の個数は N とする．また，配列の初期処理 (ゼロクリア)，最終結果作成のオーバーヘッドは考慮しなくても良いほど N は大きいとしている．

総和演算　通常の加算とは異なり，$\ell = [m/k] + 2$ となる．これは，総和計算では，変数の先頭位置が固定されないので，　mod (m,k) のビットが，先頭と最後の 2 つにまたがる場合が発生する事による．これにより，個々の要素の変換処理に要する演算量は，7ℓ となる．個々の要素の加算に要する演算量は，3ℓ となる．また変数の先頭位置を固定するために，個々の要素に対し，シフト命令 1，アンド命令 1，加減算 2，乗算 1，除算 1 が必要でこれを演算量 6 とすると，合計，演算量は $10\ell+6$ となる．

内積演算　精度保証が必要なため，乗算 $A(I) \times B(I)$ では $\ell_{dot} = [m/k]+1$ とすると，乗算の演算量は，$3\ell_{dot}^2$ となる．変換に関する演算量は $10\ell_{dot}$ で乗算部分の演算量は $3\ell_{dot}^2+10\ell_{dot}$ となる．加算部分 $T = T+A(I)\times B(I)$ では，$A(I)\times B(I)$ は $2m+1$ ビットの整数であるので，$\ell_{sum} = [2m/k]+2$ とすると演算量は $10\ell_{sum}+4$ となる．総和計算の場合と異なり，加算時には，既に指数部の値はわかっているので，シフト命令 1，アンド命令 1 が不要になる．

総和演算，内積演算の演算量は，配列整数方式はそれぞれ，n の 1 次式，n の 2 次式になる．

Bailey アルゴリズムで精度保証総和・内積を行おうとする場合，加算部分には特殊なハードウエアが無いかぎり配列整数方式をとる必要が生じる．すなわち，2n 倍精度では，総和計算で $\ell = ([52/k]+2)n$，となり，たとえば，$k = 30$ とすると，$n = 3$ では，Bailey 方式では，$\ell = 9$，配列整数方式では，$\ell = 7$ となる．内積計算では，$\ell_{sum} = ([52/k]+1)$ の計算を $2n^2$ 回行う必要が生じる

総和と内積の性能を図 4 と 5 に示した．

n が大きくなると総和計算，内積演算ともに Bailey アルゴリズムと，配列整数方式との性能差が顕著になる．内積演算は Bailey アルゴリズムは n の小さい加算を $2n^2$ 回実行するので，n が大きくなると配列整数演算との性能差がより顕著になる．

精度保証総和・内積の測定値とインライン展開した乗算の性能比較結果を以下に図 7 と 8 に示す．

精度保証内積演算は，$2n$ 倍精度乗算結果を $4n$ 倍精度求め，$4n$ 倍精度加算を行っている．通常の乗算の 2 倍以上の演算量があるにもかかわらず，同じ性能がでている事からも精度保証内積演算は配列整数方式が有効と言える．

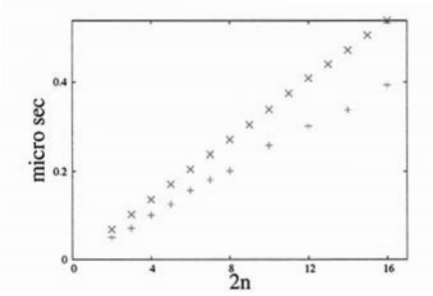

図 5　精度保証総和計算の 1 要素あたりの性能（+：配列整数演算，×：Bailey）

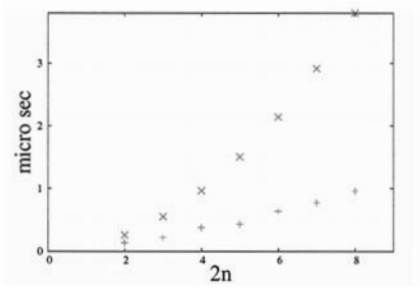

図 6　精度保証内積計算の 1 要素あたりの性能（+：配列整数演算，×：Bailey）

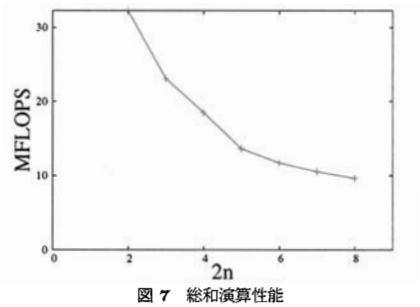

図 7　総和演算性能

6. 多倍長演算への応用例

6.1 基本演算計算

例の最初として，以下の四則演算からなる計算を行った．[6)]

$$333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}$$

但し

$$a = 77617, b = 33096$$

である．

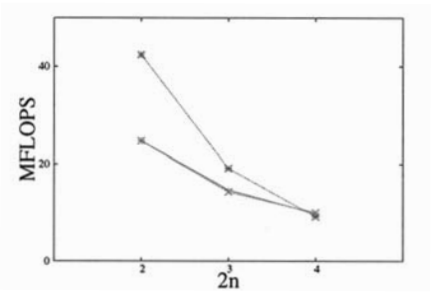

図 8 内積および乗算演算性能（+:配列整数方式の内積，×:配列整数方式の乗算，∗：Bailey の乗算）

4 倍精度での数値計算の結果は

1.17260394005317863185883490452018

となる．正確な解は

-54767/66192

=-0.82739605994682136814116509547982....

であり，全く数値計算とは異なる．

これを配列演算方式の 8 倍精度でビット落ち情報で計算すると，$[333.75b^6+a^2(11a^2b^2-b^6-121b^4-2)]$ と $5.5b^8$ の加算の部分で 121 ビットの桁落ちが発生することが確認できる．

配列演算方式を使った 8 倍精度の計算においては，121 ビットの桁落ちは回避され，

-0.82739605994682136814116509547982

となる．

Bailey アルゴリズムでは $n=3$ すなわち 6 倍精度（準完全保証）で実行すると，

-0.827396059946821

となる．ここでは除算に関しては結果に大きな影響を与える事はないので，倍精度で求めた．

6.2 連立一次方程式の求解 (1)

条件数の大きい非対称 Toeplitz Matrix 行列を考える．

$$Ax=B, x(i)=1(1\le i\le n)$$

$$A=\begin{pmatrix} 2 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & \dots & 0 & 0 \\ r & 0 & 2 & 1 & 0 & \dots & 0 & 0 \\ 0 & r & 0 & 2 & 1 & \dots & 0 & 0 \\ 0 & 0 & r & 0 & 2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 2 \end{pmatrix}$$

$$B=\begin{pmatrix} 3 \\ 3 \\ 3+R \\ 3+R \\ \vdots \\ 3+R \\ 2+R \end{pmatrix}$$

で，$N=400$，$R=5$ を計算すると誤差は，

25695655046009692393.3744698297561

つまり 10^{19} なので，$19/\log 10(2)=63.1...$ より，4 倍精度に 63 ビット以上を追加する必要がある．

配列演算方式の 8 倍精度は仮数部 240 ビットで 4 倍精度に 128 ビット追加しており誤差は，

3.476971293505648706609618829540524E-0020

となる．

Bailey アルゴリズムを使う場合には，$n=3$ では，ビットが不足するので，$n=4$ で実行する必要がある．ここでは除算は結果に大きな影響を及ぼすので，$n=4$ の除算を作成し，実行を行った．誤差は，$n=4$ で計算した結果の先頭の倍精度の値を出力している．ルーチンはすべて完全精度保証（$n=4$) を使用している．誤差は，

9.802272573073307E-013

となる．

配列演算方式 8 倍精度と，Bailey アルゴリズムの $n=4$ では誤差で 10 進 7 桁以上の差があるが，表 7 にあるように配列演算方式 8 倍精度仮数部が 241 ビット，Bailey アルゴリズムの 212 ビットと精度に関連する有効ビット数の差 $241-212=29$ ビットによるものである．

6.3 連立一次方程式の求解 (2)

需要予測，回帰分析などで最小二乗近似法を使用する場合，正規方程式は，Hilbert 行列 A となり，$Ax=b$ の連立一次方程式を解く事が必要になる．ただ，この連立一次方程式は小さな次元数 ($n=10$ 程度) でも非常に精度が悪くなる事が知られている[7]．実際，条件数を幾つか計算すると，以下の様になっている．

行列の次元数 N, 条件数 10^m には，$m\simeq 1.5\times N$ の関係がある．このため，4 倍精度では $N=20$ 程度までしか精度が出せない事が予想される．多くの変数を扱う需要予測，回帰分析などでは，4 倍精度でも使用する行列の解法の選択には注意を要すると言える．

実 測 結 果

Hilbert 行列 A は $A_{ij}=1.0/(i+j-1)$ のように与えられる．ベクトル $b(i)=\Sigma A_{ij}(j=1,\dots,N)$ と

表 8 Hilbert 行列の条件数

行列サイズ N	条件数	ビット数
10	$10^{13.548}$	45
15	$10^{21.187}$	70
20	$10^{28.798}$	96
25	$10^{36.443}$	121
30	$10^{44.071}$	146
35	$10^{51.715}$	171
40	$10^{59.351}$	197
45	$10^{66.995}$	223
50	$10^{74.637}$	248

して，$Ax = b$ の連立一次方程式 を解くと，理論解は $x(i) = 1.0(i = 1, \ldots, N)$ である．ここで N を変えて，誤差 10^{-6} 以内となる N を求めた．N_1 は 10^{-6} 以下になる最大の整数，N_2 は 10^{-6} 以上になる最小の整数である．4 倍精度のプログラムをそのまま実行すると $N_1 = 20$ （誤差は $10^{-6.917}$）と $N_2 = 21$ （誤差は $10^{-4.690}$）が得られる．表 8 に Bailey アルゴリズムと配列整数方式による N_1，N_2 およびそのときの誤差をまとめた．表 7 にあるように Bailey アルゴリズムと配列整数方式とのビット数が異なることに注意されたい．

表 9 Hilbert 行列の誤差

解法 (精度)	N_1	誤差	N_2	誤差
Bailey(4 倍)	18	$10^{-7.180}$	19	$10^{-5.765}$
Bailey(6 倍)	29	$10^{-7.062}$	30	$10^{-5.130}$
Bailey(8 倍)	40	$10^{-6.538}$	41	$10^{-5.422}$
配列整数方式 (5 倍)	27	$10^{-6.679}$	28	$10^{-4.726}$
配列整数方式 (6 倍)	33	$10^{-6.490}$	34	$10^{-5.127}$
配列整数方式 (7 倍)	39	$10^{-7.600}$	40	$10^{-5.390}$
配列整数方式 (8 倍)	45	$10^{-6.968}$	46	$10^{-5.328}$

この表から 最大誤差を ϵ_{max}，仮数部のビット数を M_{bit} 条件数を 10^m とすると，

$$-\log_{10}(|\epsilon_{max}|) = M_{bit}\log_{10}(2) - m$$

の関係が見られる．

7. おわりに

多倍長計算に使われる Bailey アルゴリズムと配列整数演算方式による性能比較を行った．4 倍精度までなら，Bailey アルゴリズムが優れているが，それ以上になると配列整数演算方式が性能的に優位なる．これは演算量が Bailey アルゴリズムでは加算で n の 2 次式，乗算で n の 3 次式に対し，配列整数演算方式は加算で n の 1 次式，乗算で n の 2 次式になるからである．

また，濱口の開発した配列整数方式においては，加算の桁落ちの検査が可能であると共に，また完全精度保証した総和・内積計算が可能である．

4 倍精度計算およびそれ以上の多倍長精度計算は，巨大しつつあるプログラムのシミュレーションの結果を計算機自体で保証するために必須であり，高速計算が必要が必要である．

謝　辞

様々な助言を頂いた高エネルギー加速器研究機構の湯浅富久子准教授に感謝いたします．

参考文献

1) Y. Hida,, X.S. Li, and D.H. Bailey:Algorithm for quad-double precision floating point arithmetic, *Proc. 15th IEEE Symbosium on Computer Arithmetic* pp.287–302 (2001).
Yozo Hida, Xiaoye S. Li and David H. Bailey: "Quad-Double Arithmetic: Algorithms, Implementation, and Application", Technical Report LBNL-46996, Lawrence Berkeley National Laboratory (2000)

2) 藤原宏志：科学技術計算に適した多倍長数値計算環境の構築と数値的に不安定なスキームの直接計算の実装，
情報処理学会論文誌：コンピューティングシステム，Vol.48 No.SIG13(ACS 18) pp.22–30 (2007).

3) 濱口信行：配列整数演算の精度，性能評価，情報処理学会：研究報告，IPSJ SIG Technical Report 2007-ARC-172(23)/2007-HPC-109(23) pp.133–137(2007).

4) F.Yuasa et.al.:Numerical Evaluation of the scalar one-loop integrals with the infrared divergence, hep-ph arXir:0709.0777v2 (2007).

5) 永井貴博，吉田仁，黒田久泰，金田康正:SR11000 モデル J2 における 4 倍精度積和演算の高速化，情報処理学会論文誌：コンピューティングシステム，Vol.48 No.SIG13(ACS 19) pp.214–222 (2007).

6) C.Hu, S.Xu and X.Yang:A review on Interval Computation:Software and Applications, *Int. J. Comput. Numer. Anal. Appl.* 1, No.2, 149-162 (2002)

7) 杉原正顯，室田一雄：数値計算法の数理，岩波書店，2003.