多倍長計算手法

目次

多倍長計算の手法

最近では扱う問題規模が大きくなり,演算量が増えた事により,倍精度演算のみでは,十分な精度の結果が得られなかったりする事もあり,一部の処理部分をより高精度な演算を使用する事が行われています。

また, $\infty - \infty$ を引いて有限値

を求める(いわゆるくりこみと言われる)計算は計算機から見れば桁落ちの激しい計算となります。

ここでは拡張倍精度(符号部1ビット,指数部15ビット,仮数部64ビット)より多くの仮数部を持つものを多倍長精度と記述します。多倍長精度演算では計算機のアーキテクチャーに依存し,またコンパイラの最適化手法とも係りがあるため,1つの項目にまとめました。

１．4倍精度データ形式

4倍精度には2つのデータ形式があります。

(ア)　4倍精度変数Qを2つの倍精度変数A,Bの和で表現する方式　Q=A+B

更に精度を拡張する場合は倍精度変数の数を増やします。 (3つで6倍精度,4つで8倍精度）

(イ)　IEEE754-2008に準拠したもの

符号部1ビット,指数部15ビット,仮数部112ビット

更に精度を拡張する場合は仮数部のビット数を増やします。 (仮数部176ビットで6倍精度,240ビット8倍精度）

このあとの記述では,(ア)の形式をDD形式,(イ)の形式をIEEE形式とします。

この事に関連した精度に関する2つの簡単な例題をあとの部分で記載しています。

2. DD形式に関する注意事項

DD形式での多倍長演算の基本は

倍精度変数の加減算,乗除算の結果を
２つの倍精度変数の和で表す事です。

a,b,c,d倍精度変数; $a \pm b, a \times b, a \div b = c + d$

加算 : $c = a + b$
$t = c - a$
$d = (a - (c - t)) + (b - t)$

減算 : $c = a - b$
$t = c - a$
$d = (a - (c - t)) - (b - t)$

(加算,減算ともに最適化オプションで括弧をはずした演算順序に
ならない様に注意が必要です。)

乗算(乗加算命令あり)

$c = a * b$
$d = a * b - c$ （最適化オプションにより$d = 0$とならないように
注意が必要です。)

乗算(乗加算命令なし)

$r = 134217729.0d0 \quad (= 2^{27} + 1)$
$c = a * b$
$t1 = a * r$
$a1 = t1 - (t1 - a)$
$a2 = a - a1$
$t2 = b * r$
$b1 = t2 - (t2 - b)$
$b2 = b - b1$
$d = ((a1 * b1 - c) + a1 * b2 + a2 * b1) + a2 * b2$

コンパイラの最適化機能はd=0にならない様にする事と
関連します。
アーキテクチャーは乗加算命令がサポート(かつコンパイラ
のサポート)と関連します。

3. Rump'sの例題

以下の非常に簡単な例題です。

$$a = 77617.0, b = 33096.0$$

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

$$a^2 = 5.5b^2 + 1 \text{から} f = -2 + \frac{a}{2b} = -\frac{54767}{66192} = -0.827396059946...$$

計算機で実行しますと,以下の様に全くでたらめな結果がでてきます。

倍精度演算結果: $f = -0.11805916207174113 \times 10^{22}$

4倍精度演算結果: $f = 1.172603940053.....$

これは以下のA+Bの計算時に121ビットの桁落ちが発生することが原因です。

$$A = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2)$$

=-79171113406689613611011347015429428 50.0

B= $5.5b^8$ = 79171113406689613611011347015429428 48.0

このため,正しい結果を得るには有効桁数が10進37桁必要になります。実際,IEEE形式5倍精度(有効桁数10進43桁),6倍精度(有効桁数10進53桁),DD形式6倍精度(有効桁数10進48桁),拡張倍精度変数を2つつなげた(有効桁数10進39桁)演算を用いると,f=-0.827396059946.....となります。

(注)拡張倍精度変数をつなげるときは乗算で$r = 2^{27} + 1$(倍精度)を

$r = 2^{33} + 1$に変更します。

4.ファイマンループ積分のmassless のケース

これも非常に簡単な二重積分の計算の場合です。

$$I=\int_0^1\int_0^{1-x}\frac{1}{(xy)^{1-\eta}}dydx$$

これは,$x=0,y=0$で特異点(端点)を持つため

二重指数関数型積分を使用して計算します。

求める解は以下の様になります。

$$I=\frac{1}{\eta}\int_0^1 x^{\eta-1}(1-x)^{\eta}=\frac{1}{\eta}B(\eta,\eta+1)=\frac{1}{\eta}\frac{\Gamma(\eta)\Gamma(\eta+1)}{\Gamma(2\eta+1)}=\frac{1}{2\eta}\frac{\Gamma^2(\eta)}{\Gamma(2\eta)}$$

(注)二重指数関数型積分法

$$x=\varphi(t),a=\varphi(-\infty),b=\varphi(\infty)$$

$$I=\int_a^b f(x)dx=\int_{-\infty}^{\infty}g(t)dt$$

$$g(t)=f(\varphi(t))\varphi'(t)$$

$$r=\frac{(b-a)}{2},c=\frac{(a+b)}{2}\text{とすると}$$

$$\varphi(t)=r\tanh(\frac{\pi}{2}\sinh(t))+c$$

$$\varphi'(t)=\frac{\pi r\cosh(t)}{2\cosh^2(\frac{\pi}{2}\sinh(t))}$$

で数値積分を行います。このようなケースは
a=0,b=1で変数値と重みのテーブルを作成します。

$\eta = \frac{1}{100}$での実測例を示します。

解析値　　=9998.378865794316870325682352395113
DD形式4倍精度　=9328.220999954985561570967195042031 5
倍精度　　=9328.22099995494682

DD形式の4倍精度は表現できる数値の範囲が倍精度と同じです。このため,この積分計算は演算精度ではなく,表現できる数値の範囲に依存することがわかります。この事からDD形式はさらに6倍精度,8倍精度演算を使用しても正しい積分値が得られない事になります。

解析値　　=9998.378865794316870325682352395113
IEEE4倍精度　=9998.378865794316870325678679367 06
拡張 倍精度　=9998.3788657943164341 9

演算精度の問題か,表現できる数値範囲(アーキテクチャー)の問題かの判断を誤ると,問題解決までに長い時間がかかります。

5. 条件数の大きい線形計算

条件数の大きい線形計算では小さな次元の行列でもその演算精度を挙げる必要が生じます。その典型的な例2つを記載しました。

5.1 非対称Toeplitz行列

Ax=b A:行列,x,bはベクトル

Aを定め,解がすべて1となる様にbを設定しています。その条件数及び演算精度、最大誤差は次ページの結果となっています。

$$A=\begin{bmatrix} 2 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 2 & 1 & 0 & \cdots & 0 & 0 \\ r & 0 & 2 & 1 & \cdots & 0 & 0 \\ 0 & r & 0 & 2 & \cdots & 0 & 0 \\ 0 & 0 & r & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 2 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 2 \end{bmatrix}$$

$$b=\begin{bmatrix} 3 \\ 3 \\ 3+r \\ 3+r \\ 3+r \\ \vdots \\ 3+r \\ 2+r \end{bmatrix}$$

次元数　N=1000

Toeplitz 行列の条件数と最大誤差一覧表

r	条件数	倍精度	4倍精度
2.5	1.04E+07	5.47E-09	5.55E-28
10	1.21E+257	9.17E+239	9.62E+221
25	2.27E+411	————	2.67E+376

r	8倍精度	16倍精度	32倍精度
2.5	4.87E-66	4.59E-143	4.06E-297
10	1.70E+183	1.09E+106	2.34E-49
25	9.70E+336	4.68E+260	2.28E+106

5.2　ヒルベルト行列

$$H(i,j)=\frac{1}{i+j-1}\ (1\le i,j\le n)$$

$$H^{-1}{}_{j,i}=(-1)^{i+j}\frac{(n+i-1)!(n+j-1)!}{(i+j-1)[(i-1)!(j-1)!]^2(n-i)!(n-j)!}$$

条件数 $=\| H \|_\infty \| H^{-1} \|_\infty$

$$\| H \|_\infty = \max(1\le i\le n)\sum_{k=1}^{n} | h_{i,k} |$$

$N=5$　　条件数0.943×10^{6}

$N=10$　　条件数0.353×10^{14}

行列の次元数$N(>>1)$,条件数10^{m}

$m \fallingdotseq 1.5\times N$

$Hx=b, x(i)=1(i=1,2,...N)$となるように

bを設定して連立一次方程式を解きました。

次元数,演算精度、最大誤差は次ページの結果となっています。

ヒルベルト行列最大誤差一覧表

次元数	倍精度	4倍精度	8倍精度
20	51.616	1.44E-07	2.13E-47
40	182.107	62.034	8.72E-16
60	1971.569	62.804	251.132
80	642.074	300.734	252.653
100	201.355	378.428	242.672

次元数	16倍精度	32倍精度
20	1.47E-99	1.44E-276
40	4.09E-86	4.30E-243
60	2.66E-60	1.36E-210
80	3.65E-31	1.08E-178
100	2.58E+00	1.41E-150

6. ３次元ポアソン方程式の反復解法

下の３次元ポアソン方程式を反復解法bcg法とcgs法を使用して解いた結果です。反復回数と演算精度により実行時間がもっとも短くて済む、演算精度がきまります。

３次元ポアソン方程式

$$\Delta u + R \times \frac{\partial u}{\partial x} = -f$$

領域 $[0,1] \times [0,1] \times [0,1]$

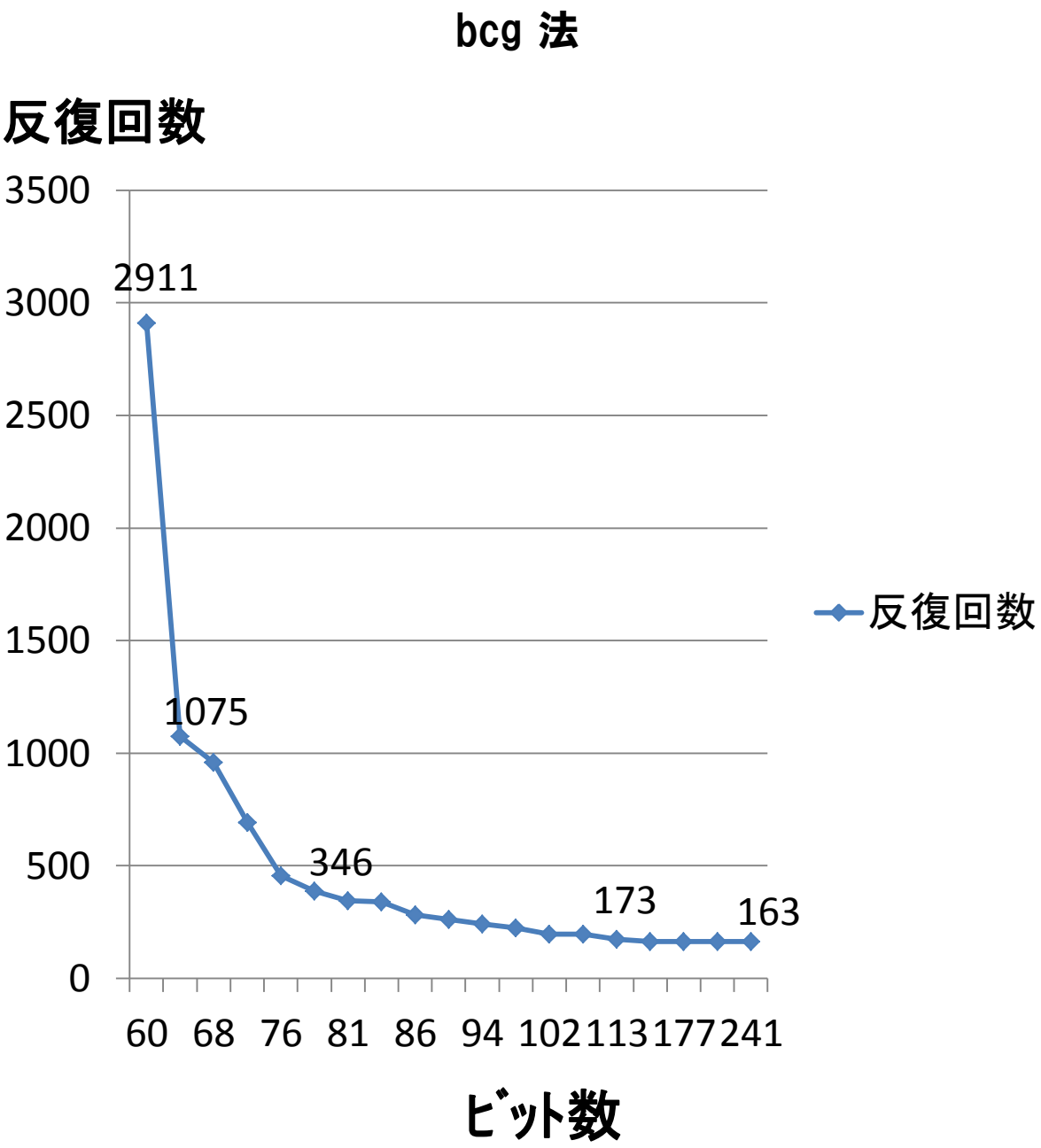
解析解 $u(x,y,z) = e^{xyz} \times \sin(\pi x) \times \sin(\pi y) \times \sin(\pi z)$

$$R = 100, nx = ny = nz = 65$$

収束判定値は共役残差 10^{-12}

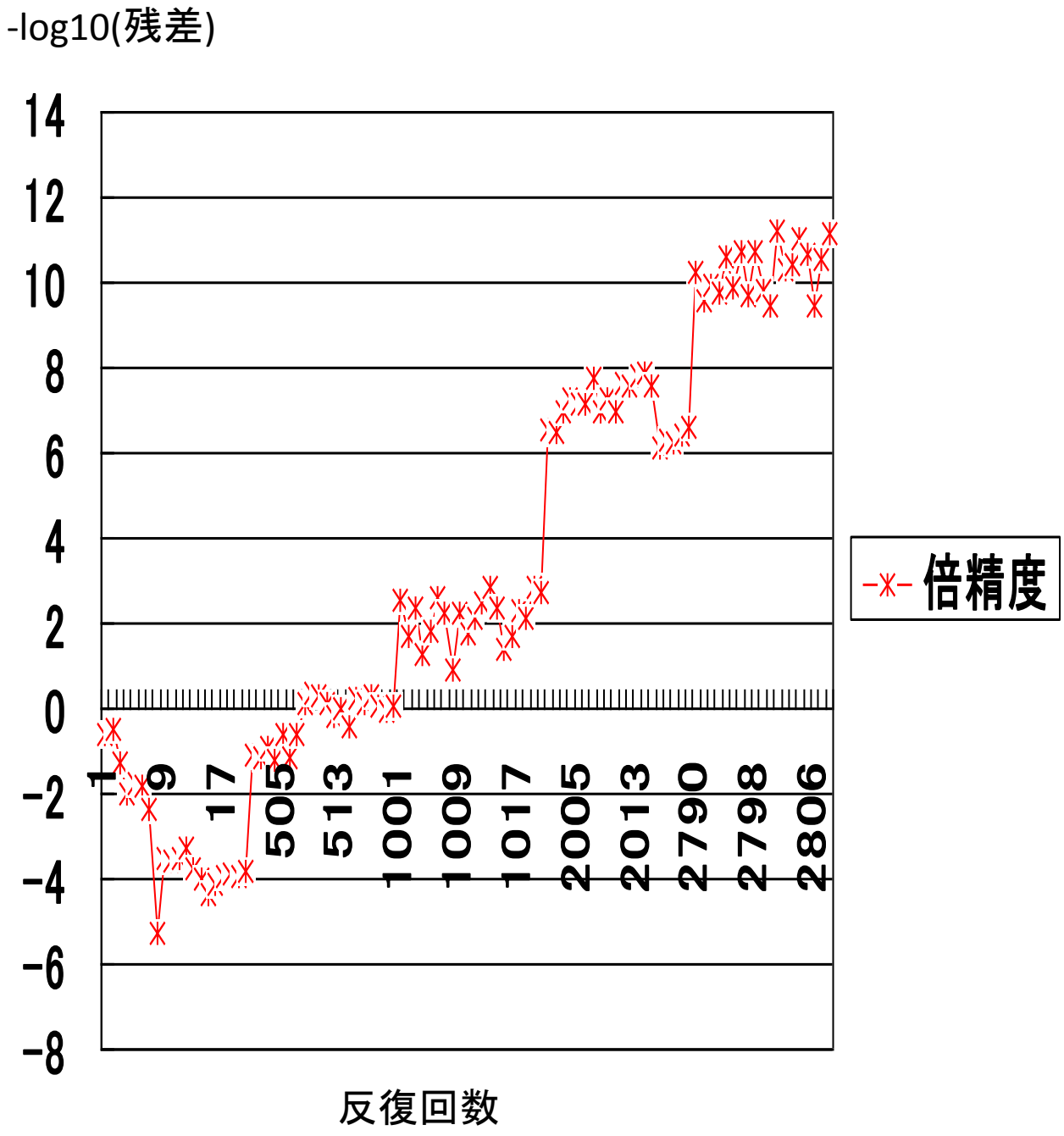
6.1 bcg 法

演算精度と反復回数は以下の様になっています。
これより,4倍精度演算が良い事がわかります。
またそれぞれの演算精度の誤差状況を次ページ以降に記載しました。



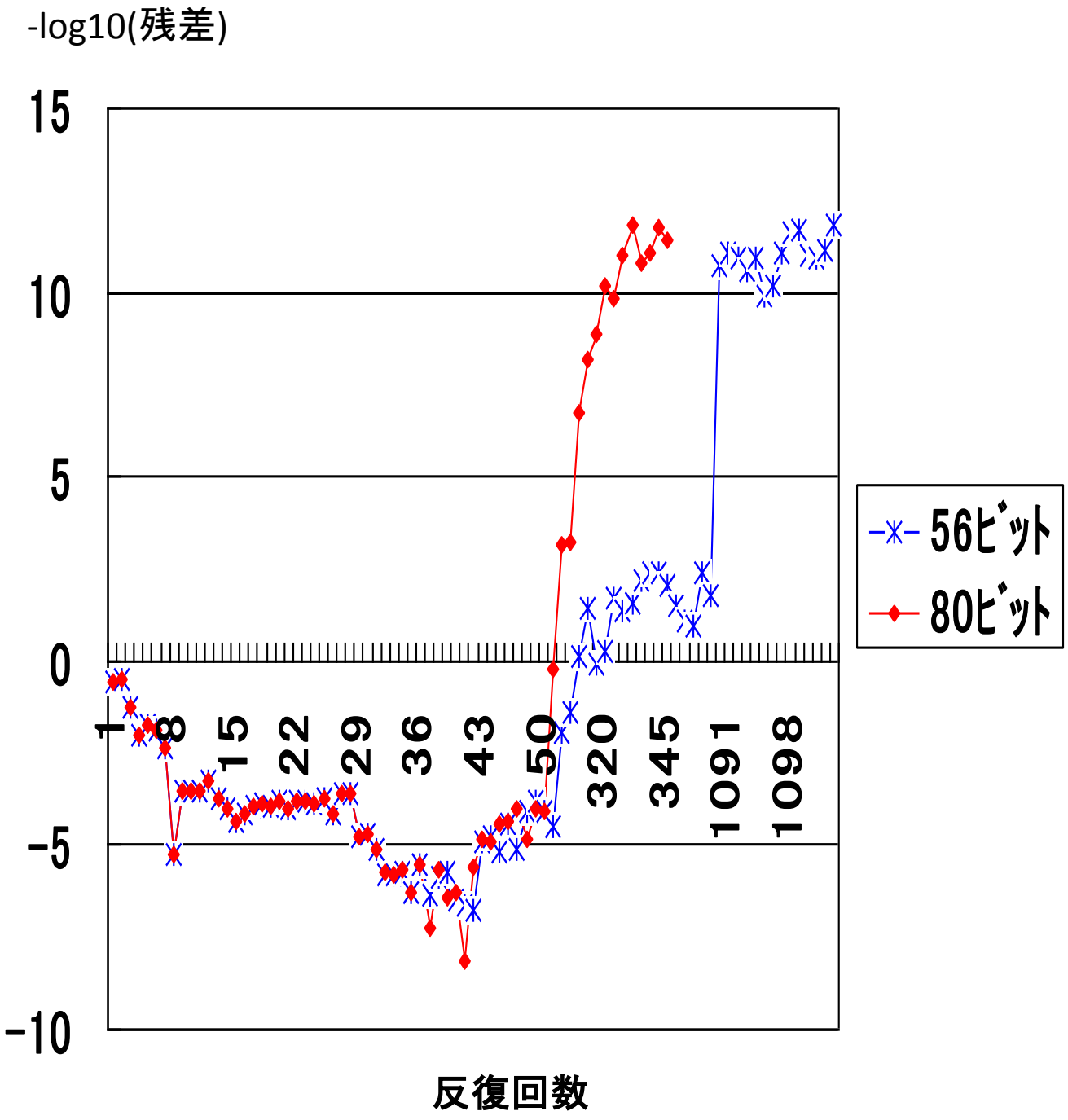
BCG法の収束状況

nx=ny=nz=65 収束判定値=1.0q-12



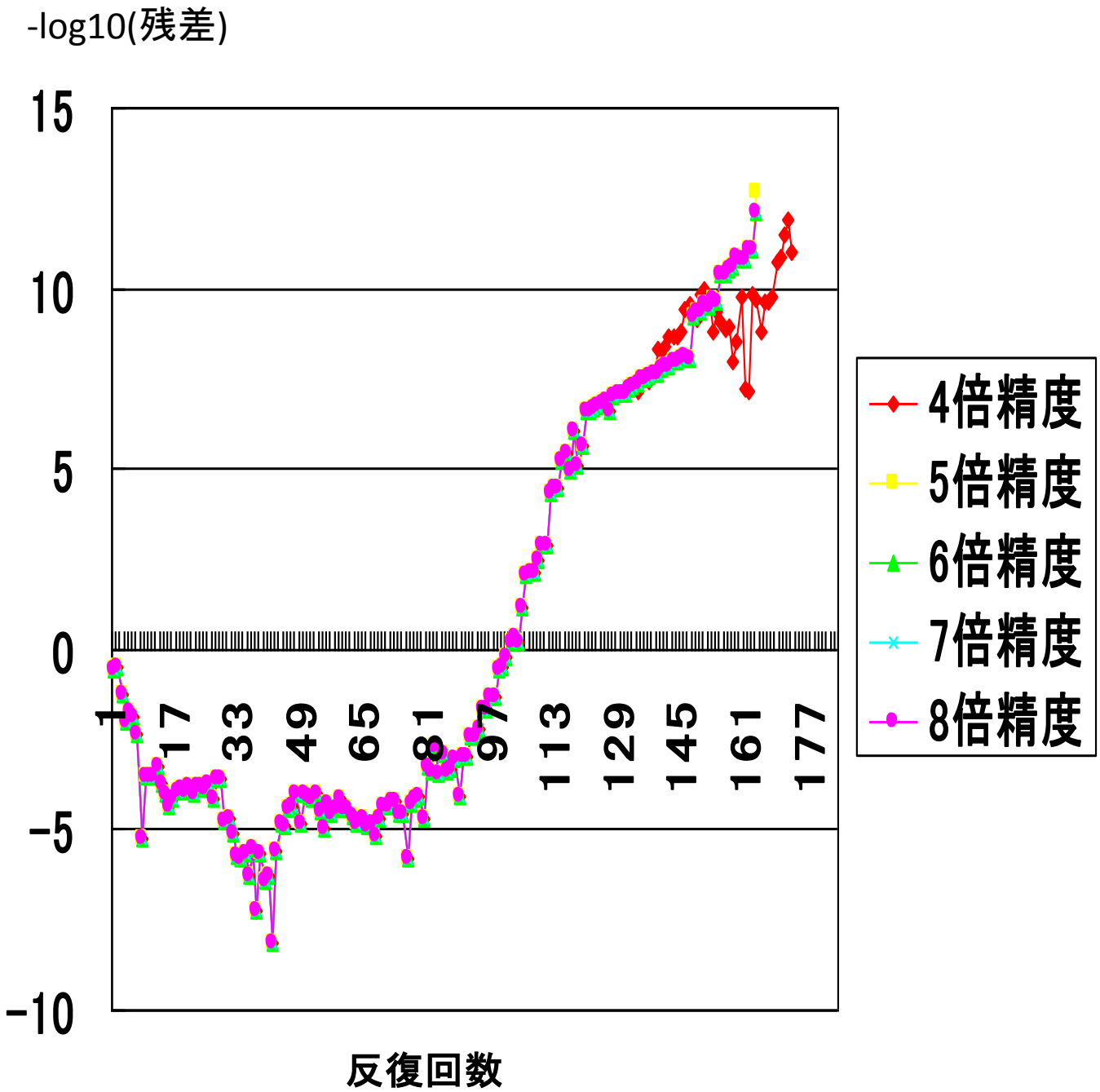
BCG法の収束状況

nx=ny=nz=65 収束判定値=1.0q-12



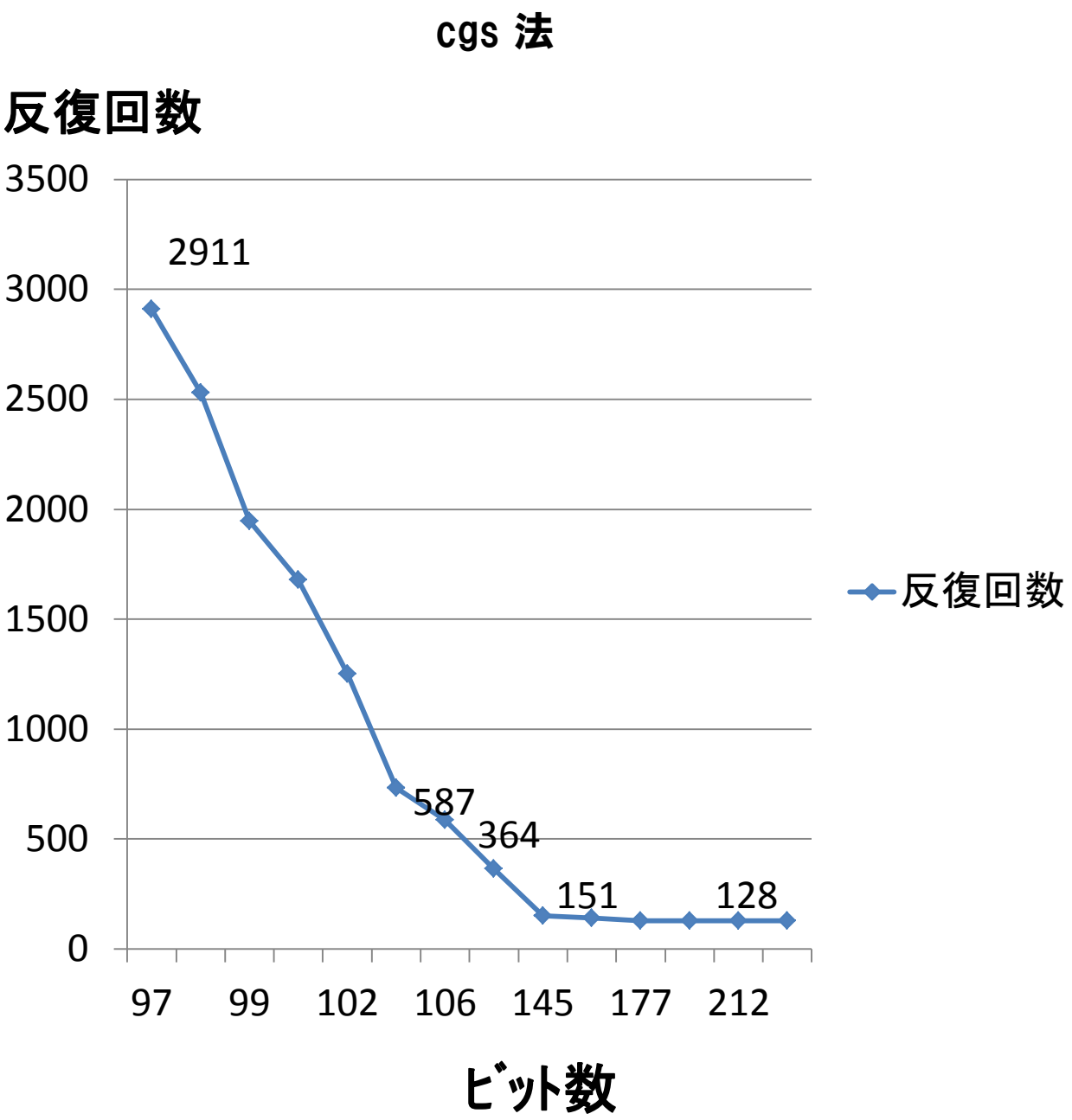
BCG法の収束状況

nx=ny=nz=65 収束判定値=1.0q-12

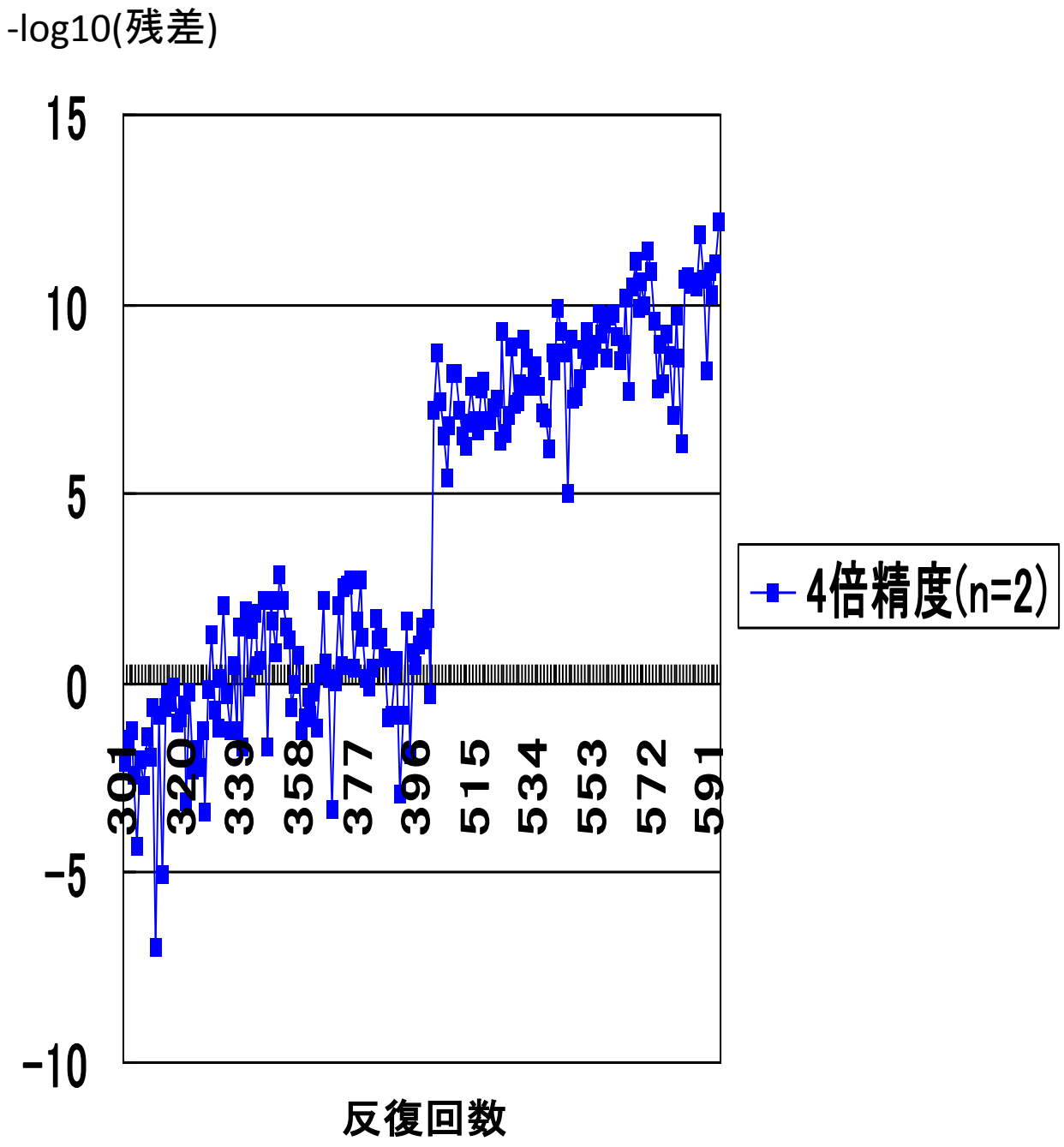


6.2　cgs　法

　　演算精度と反復回数は以下の様になっています。これより,6倍精度演算が良い事がわかります。
またそれぞれの演算精度の誤差状況を次ページ以降に記載しました。



CGS法の収束状況
nx=ny=nz=65 収束判定値=1.0q-12
4倍精度



CGS法の収束状況

nx=ny=nz=65 収束判定値=1.0q-12

6倍精度、8倍精度