多倍長計算手法

平成２５年度第1四半期

目次

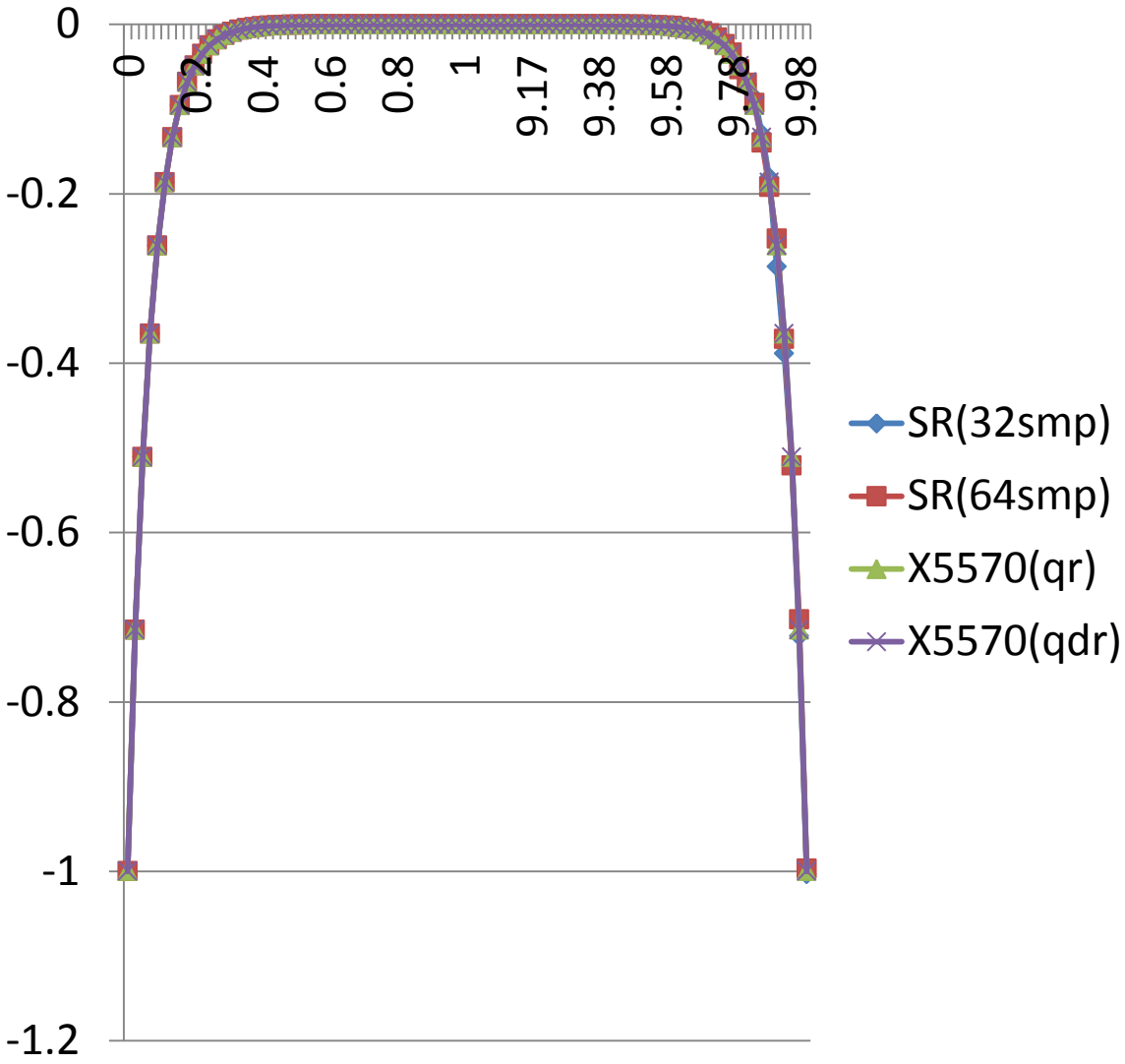
１．はじめに

平成２５年３月８日に行われました
第３回多倍長計算フォーラムで
　　量子モンテカルロ法による物性
　　スペクトル計算と4倍精度
　（KEK物構研　　岩野氏）
において,紹介されました4倍精度演算でも
正しい結果が得られない場合がある件の精度
を調査しました。

結果の検証は静的かつ規則的な
パスに対して$G(\beta) = -1$でチェックします。
問題の発端は$\beta = 10, u = 5, l = 448$の
4倍精度演算の結果です。

この問題を異なるアーキテクチャーの
SR16000/M1とX5570で実行した結果、
若干結果の精度に問題がみられました。
（次ページに結果を掲載）

Beta=10,u=5,l=448　4倍精度精度比較



DT	SR(32smp)	SR(64smp)	X5570(qr)	X5570(qdr)
0	-1.000042	-0.999995	-1.000004	-0.999991
10	-1.00375	-0.996719	-0.999997	-1.000001

DT=0,10　での厳密解は-1.0

2.グリーン関数計算

$R_\sigma(\tau,x)=e^{-\Delta H(\tau)}e^{-\Delta H(\tau-\Delta)}\cdots\cdots e^{-\Delta H(\Delta)}=B_mB_{m-1}\cdots\cdots B_1$

$\tau=m\Delta,\beta=L\Delta \quad (0\le\tau\le\beta)$

$R_\sigma(\beta,x)=e^{-\Delta H(\beta)}e^{-\Delta H(\beta-\Delta)}\cdots\cdots e^{-\Delta H(\Delta)}=B_LB_{L-1}\cdots\cdots B_1$

グリーン関数 $\equiv R_\sigma(\tau,x)\ [I+R_\sigma(\beta,x)\ ]^{-1}$

行列表現だと $G_m=(B_mB_{m-1}\cdots\cdots B_1B_0)\ [I+B_LB_{L-1}\cdots\cdots B_1]^{-1}$

$m=0,B_0=I,B_mB_{m-1}\cdots\cdots B_1$の行列積をそのまま計算すると精度が良くありません。

そこで,精度良く計算する方法には以下の2つの方式があります。

(1) QDR方式

$B_1=Q_1D_1R_1$

$B_2Q_1=Q_2D_2R_2 \qquad B_2B_1=Q_2D_2R_2D_1R_1=Q_2D_2D_1'R_2'R_1=Q_2D_2''R_2''$

:

:

$B_mQ_{m-1}=Q_mD_m''R_m''$

$B_mB_{m-1}\cdots\cdots B_1=QDR$

Q,直交行列,D対角行列,R上三角行列 (対角要素はすべて1)

(2) QR方式

$B_1=Q_1R_1$

$B_mQ_{m-1}=Q_mR_m$

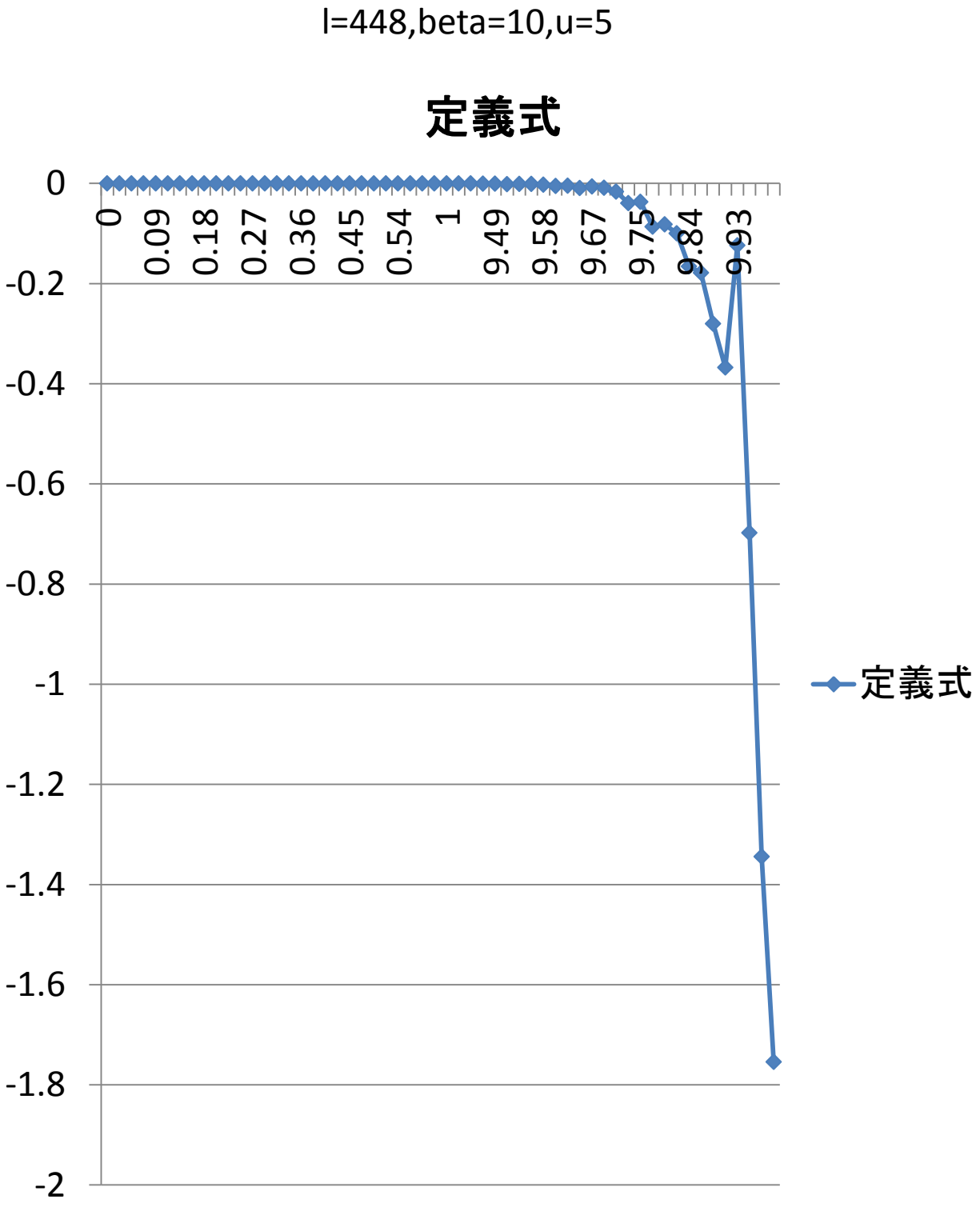
$B_mB_{m-1}\cdots\cdots B_1=Q_m(R_m\cdots\cdots R_1)=QR$

Q,直交行列,R上三角行列

どちらの場合も精度上 $[I+B_LB_{L-1}\cdots\cdots B_1]^{-1}$の計算が問題となります。

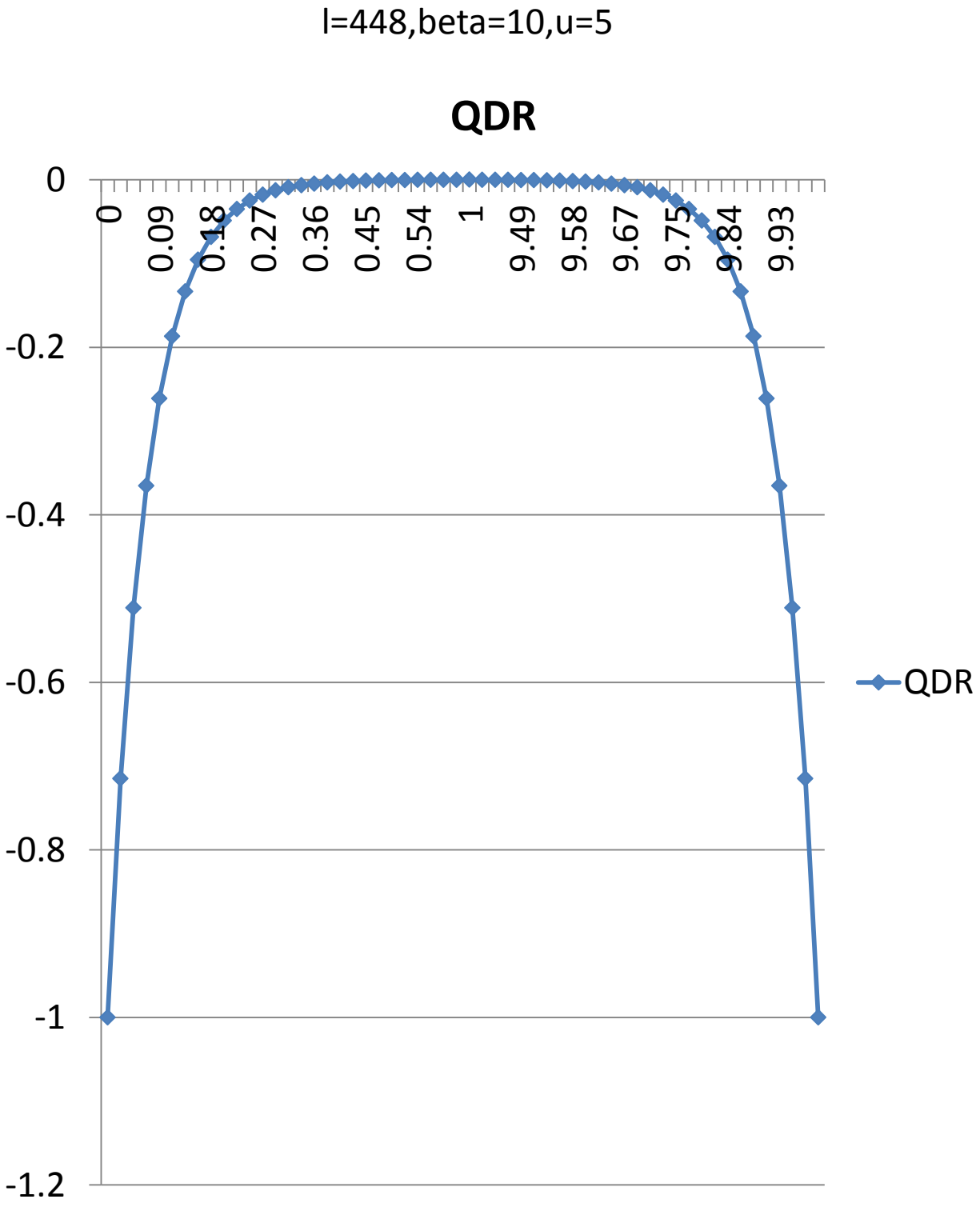
3つの計算方法による結果を次ページ以降に記しました。

Green 関数



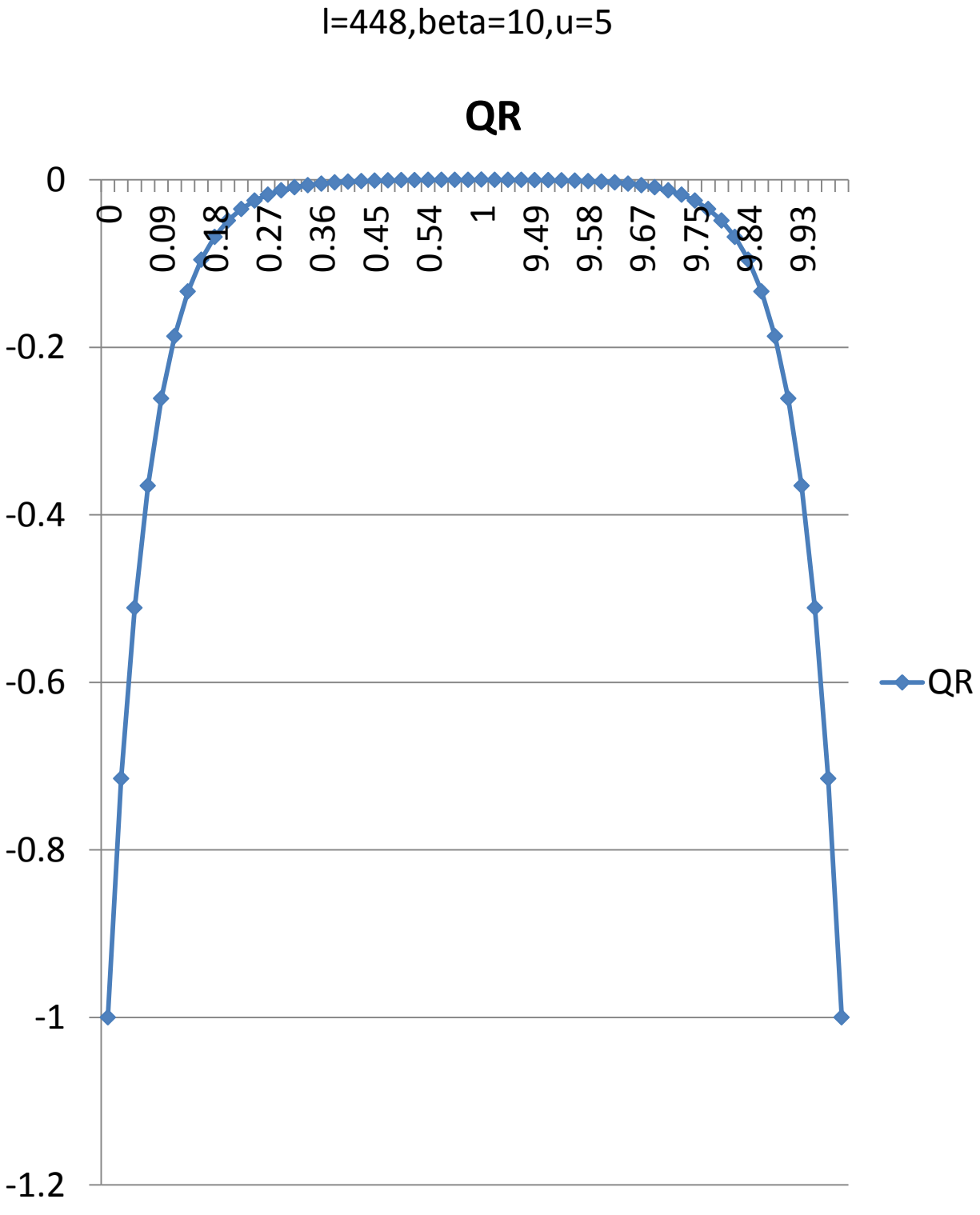
演算精度　4倍精度

Green 関数



演算精度　4倍精度

Green 関数



演算精度　4倍精度

3.条件数に関して

今回扱ったケースでは,設定したパラメータから結果の
精度が見積もり易くなっています。
パラメータ　n,L,β,uのうち$n=100,L=448$と固定しています。
G_m：グリーン関数。
$P=e^{\sqrt{dt\times u}\times L}=e^{\sqrt{\beta\times u\times L}}$,$E(m)=|Tr(G_m)|$とすると,
プログラム実行中に現れる数値の最大値P^2,
0以外の絶対値最小値$1/P^2$。
行列やベクトル要素に現れる数値の最大値P,
0以外の絶対値最小値1/P。
$E(0)=1,E(L)=1,m=\frac{L}{2}$で$E(\frac{L}{2})=\frac{1}{\sqrt{P}}$で最小となるので,
条件数は$\sqrt{P}$となり、演算に必要な最小ビット数は
$\log_2\sqrt{P}=\frac{1}{2}\log_2 P$となります。

理論式と実際の実行結果との対応は
次ページ以降に示しました。

演算に必要な最小ビット数(L＝448)

β	u=5	u=6	u=7	u=8	u=9	u=10
10	108	118	128	137	145	153
20	153	167	181	193	205	216

演算精度の有効ビット数

精度	ieee2008	dd形式
	ビット数	ビット数
4倍精度	113	106
5倍精度	145	
6倍精度	177	159
7倍精度	209	
8倍精度	241	212

DD形式,ieee754－2008形式の8倍精度までの実行結果は演算に必要な最小ビット数と各演算精度の有効ビット数の関係とよく一致しています。

dd形式の多倍長計算での実行結果

ここでOKは10桁以上一致,NGは最初の2桁以下しか一致していない
(全桁不一致も含む) 場合を示しています.

β＝10
　6倍精度　u＝6,7,8,9　OK,u＝10　　7桁一致と微妙
　8倍精度　u＝6,7,8,9,10　OK

β＝20
　6倍精度　u＝5　　5桁一致と微妙　,u＝6,7,8,9,10　NG
　8倍精度　u＝6,7,8　OK,u＝9　　6桁一致と微妙,u＝10　NG
　10倍精度　　u＝9,10　　OK

6倍精度はieee754－2008形式の5倍精度と6倍精度のほぼ中間の有効ビット数
8倍精度はieee754－2008形式の7倍精度と有効ビット数は等しい

ieee754－2008形式の多倍長計算での実行結果

β＝10
　5倍精度　u＝6,7,8　OK,　u＝9　　4桁一致と微妙,u＝10　NG
　6倍精度　u＝6,7,8,9,10 OK
　7倍精度　u＝6,7,8,9,10 OK
　8倍精度　u＝6,7,8,9,10 OK

β＝20
　5倍精度　u＝5,6,7,8,9,10　NG
　　6倍精度　　u＝5, 6　　OK,u＝7, 8, 9, 10　　NG
　　7倍精度　　u＝5, 6, 7, 8　　OK,u＝9　　5桁一致と微妙,u＝10　　NG
　　8倍精度　　u＝6, 7, 8, 9, 10　　OK

4.表現可能な数値範囲の影響とデータ形式

多倍長計算では,以下の２種類のデータ形式があります。

(1) DD形式

SR16000/M1およびBG/Qシステムなどで使用されている多倍長変数を複数個の倍精度変数の和であらわしたもの.

2個で4倍精度変数,3個で6倍精度変数,
4個で8倍精度変数,5個で10倍精度変数など

(2) ieee形式

ieee754－2008の4倍精度データ形式の仮数部部分を拡張したもの.

P倍精度変数：符号部1ビット,指数部15ビット,
仮数部32×P－16ビット

数値表現可能な範囲の制限でDD形式の演算は,オーバーフローや,絶対値の小さい数の精度の低下の影響があります。

u=10,L=448

Q:プログラム実行中に現れる最大数の10進桁数
P:行列,ベクトル要素の最大数の10進桁数
R:演算に必要な最小ビット数

beta	理論式			16倍精度の実測			SR16000
	Q	P	R	Q	P	R	dd形式
30	318.4	159.2	264.5	322.8	161.4	264.9	overfow
29	313.1	156.5	260	317.2	158.6	260.4	overflow
28	307.6	153.8	255.5	311.5	155.8	255.8	overflow
27	302.1	151	250.9	305.7	152.8	251.2	NG
26	296.4	148.2	246.2	299.8	149.9	246.4	NG
25	290.7	145.3	241.4	293.8	146.9	241.6	NG
24	284.8	142.4	236.5	287.7	143.8	236.7	NG
23	278.8	139.4	231.6	281.5	140.8	231.6	NG
22	272.7	136.3	226.5	275.1	137.6	226.5	NG
21	266.4	133.2	221.3	268.6	134.3	221.2	5桁一致

ieee754-2008　8倍精度

β	一致10進桁数	
21	10	OK
22	9	微妙
23	7	微妙
24	3	微妙

1/Qの計算において,dd形式の0以外の絶対値最小数は

$2^{-1074} \fallingdotseq 10^{-323.3} \fallingdotseq 0.494 \times 10^{-323}$ のため,

$1074-4\times R$　ビットの精度しか持ち得ません。

aの平方根計算には以下の２つの方法があります。

(1) 逆数平方根$1/\sqrt{a}$を求め$\sqrt{a} = a \times (1/\sqrt{a})$より計算する。

$x_0 = 1/\sqrt{a}$(倍精度) から$x_{n+1} = x_n + \dfrac{x_n(1 - a x_n^{\,2})}{2}$で計算。

$x_n -> 1/\sqrt{a}$.

除算がないという利点はあるが,aがおおきな数値の場合,$x_n^{\,2}$の計算結果の精度が数値表現の制限の影響をうけ悪くなる.

(2) $x_0 = \sqrt{a}$(倍精度)　から$x_{n+1} = \dfrac{1}{2}(x_n + \dfrac{a}{x_n})$で計算。

$x_n -> \sqrt{a}$.

除算があるが,aがおおきな数値の場合,数値表現の制限の影響はうけない。

平方根計算の方式を(1)から(2)に変更すると,
SR16000/M1　DD 形式10倍精度では,
β=21,22,23,24,25,26,27　u=10で結果は
10進10桁まで一致と精度改善ができました。

β=28,29,30　u=10　はオーバーフローで結果は変わりません。

5.dd形式に関して

a_i : 倍精度変数,ieee754－20084倍精度変数　仮数部,指数部が1つ

これは,2つつなげた場合と3つ以上つなげた場合では
異なる処理が必要になる.

$c_1 = a_1 + a_2$
$p_1 = c_1 - a_1$
$c_2 = (a_1 - (c_1 - p_1)) + (a_2 - p_1)$
なら$|c_1| \geq |c_2|$となる.

$n = 3$

t_1, t_2, t_3が得られると,$(t_1, t_2, t_3) -> (t_4, t_5, t_6) -> (c_1, c_2, c_3)$
で$|c_1| \geq |c_2| \geq |c_3|$となる.

$n = 4$

t_1, t_2, t_3, t_4が得られると,
$(t_1, t_2, t_3, t_4) -> (t_5, t_6, t_7, t_8) -> (t_9, t_{10}, t_{11}, t_{12}) -> (c_1, c_2, c_3, c_4)$
で$|c_1| \geq |c_2| \geq |c_3| \geq |c_4|$となる.

8倍精度の注意事項

DD形式の4倍精度を2つつなげた場合 (DQ形式)

$c_1 = a_1 + a_2, c_2 = a_3 + a_4$　c_i : 4倍精度変数,a_i : 倍精度変数

$|c_1| \geq |c_2|$,$|a_1| \geq |a_3| \geq |a_4|$ではあるが,
$|a_2|$と$|a_3|$,$|a_4|$の大小関係は定まらない.

SR16000/M1での8倍精度乗算に関して

倍精度変数を4つつなげたQD形式
4倍精度変数を2つつなげたDQ形式

$a = a_0 + a_1 + a_2 + a_3, b = b_0 + b_1 + b_2 + b_3$
$a = c_0 + c_1, b = d_0 + d_1$
$c_0 = a_0 + a_1, c_1 = a_2 + a_3, d_0 = b_0 + b_1, d_1 = b_2 + b_3$
とすると,
QD形式

$$\begin{aligned} a \times b = {} & a_0 b_0 \\ & + a_0 b_1 + a_1 b_0 \\ & + a_0 b_2 + a_1 b_1 + a_2 b_0 \\ & + a_0 b_3 + a_1 b_2 + a_2 b_1 + a_3 b_0 \\ & + a_1 b_3 + a_2 b_2 + a_3 b_1 \end{aligned}$$

DQ形式

$$\begin{aligned} a \times b &= c_0 d_0 + c_0 d_1 + c_1 d_0 \\ &= a_0 b_0 + a_0 b_1 + a_1 b_0 + a_1 b_1 \\ &\quad + a_0 b_2 + a_0 b_3 + a_1 b_2 + a_1 b_3 \\ &\quad + a_2 b_0 + a_2 b_1 + a_3 b_0 + a_3 b_1 \end{aligned}$$

DQ形式はQD形式より$a_2 b_2 (O(\varepsilon^4), \varepsilon = 2^{-53})$だけ精度が悪くなります。
$a_3 = b_3 = 0$の場合、もっとも精度が悪くなり,
倍精度変数を3つつなげたTD(6倍精度)の乗算結果

$$\begin{aligned} a \times b = {} & a_0 b_0 \\ & + a_0 b_1 + a_1 b_0 \\ & + a_0 b_2 + a_1 b_1 + a_2 b_0 \\ & + a_1 b_2 + a_2 b_1 \end{aligned}$$

と一致する事になります.