多倍長計算手法

平成２５年度第4四半期

目次

1. はじめに
2. 整数演算方式による128倍精度演算の作成
   2.1 概略
   2.2 加減算
   2.3 乗算
   2.4 作成作業に関して
3. 128倍精度演算の精度
   3.1 平方根計算
   3.2 ヒルベルト行列
4. 128倍精度演算の性能
5. 量子モンテカルロ法による物性スペクトル計算
   5.1 大きなβの場合に関して
   5.2 パラメータ領域拡大結果

1.はじめに

量子モンテカルロ法による物性スペクトル計算で正しく
計算出来るパラメータ範囲の限界を求めるために,32倍
精度変数をつなげる方式で,64,96,128倍精度演算を作成
して32倍精度演算でのパラメータ範囲の限界値$\beta = 250$
$U = 10, N = 100, L = 448$の条件で$x5570$で実行した所,
64倍精度演算で2日弱,96倍精度演算で1週間,128倍精度
演算で2週間かかっています。最終的には64倍精度演算で
正しく計算出来るパラメータ範囲は$\beta = 600, U = 10, N = 100$
$L = 448$まで拡大しています。
$ieee754-2008$形式では表現できる数値範囲から理論的には
128倍精度演算まで実行可能ですが,1ケースのテストで2週間
もかかるため,128倍精度演算を整数演算方式で行う様にしました。
ここで問題となるのはソースステップ数の多さです。
原因としては8倍精度演算等の低精度演算では性能などの問題、
32倍精度より低精度演算では繰り返しパターンがほとんどないため
do文を使用せず,演算をすべて展開するという方式を取っていた事
によります。
128倍精度演算となると繰り返しパターンも多くなり,do文を使用せず
演算を展開してもその性能上の効果もなく,あまり展開が多いと性能
上も逆効果となりますので,do文を使用してソースステップ数の削減を
図りました。

整数演算方式では正整数演算$a \pm b, a \times b$が負の値ならない
様にするため, 整数型4倍バイト配列は各要素30ビット使用する,
整数型8倍バイト配列は各要素60ビット使用する様にしています。
また8倍精度変数の仮数部のビット数は240ビットと低精度演算では
もっとも扱いやすくなっています。このため,$8+15\times k$(kは正整数)
精度演算が同じパターンがでる事になります。また$4n$倍精度
のほうが無駄な処理が少なくなるため$8+60\times k$倍精度演算が
適していることがわかります。$k=2$とすれば$8+60\times k=128$
となり,128倍精度演算が適している事がわかります。
最終的には,1つのサブルーチンが350ステップ内で収まり,
加減算, 乗算のルーチンでは共通する部分がありため、デバッグ
作業等も含めても,1人で扱える作業量となりました。
ヒルベルト行列の次元数800(条件数10^{1200})で10進8桁まで一致しました。
これから,128倍精度演算から188倍精度演算,248倍精度演算に
するのには定数を変更するだけで良く、実際デバッグ($N=100$のヒルベルト
行列の誤差を求める)を含め,188倍精度は2時間、これで慣れた事により
248倍精度演算は1時間で作業が終了しました。
この事から$8+60\times k(k=0,1,2,3,4,5,6,7,8)$倍精度演算を作成し,
加減算, 乗算, 除算, 平方根計算, 判定(gt, eq, lt)の合計のステップ数
はコメント, 定義文等を含め, $k=0$で1000ステップ, $k\neq 0$で1500ステップ
となっています。今回は128倍精度演算を中心に必要におおじて他の
精度演算にかんしても記述しました。

2.整数演算方式による128倍精度演算の作成

2.1概略

除算,平方根計算は以下の様に反復法を用いる事で
加減算,乗算の作成に帰着されます。

平方根計算

$$f(x)=x^2-a,\ f'(x)=2x$$

$$x_{n+1}=x_n-\frac{{x_n}^2-a}{2x_n}=\frac{1}{2}(x_n+\frac{a}{x_n})$$

最終段階 $x_{n+1}=\frac{1}{2}(x_n-\frac{a}{x_n})+\frac{a}{x_n}$

初期値$x_0=\sqrt{a}$**(4倍精度)**

.除算

$c=\frac{a}{b}=a\times\frac{1}{b}$**で計算します。**

逆数計算は

$$f(x)=\frac{1}{x}-b,\ f'(x)=-\frac{1}{x^2}$$

$$x_{n+1}=x_n+\frac{\frac{1}{x_n}-b}{\frac{1}{{x_n}^2}}=x_n+x_n(1-bx_n)=x_n(2-bx_n)$$

で行います。

初期値$\mathbf{x}_0=\frac{1}{b}$**(4倍精度)**

(注) x_n,$1-bx_n$**が小さい場合**,$x_n(1-bx_n)$**の計算でアンダーフロー発生の可能性があるので**$x_{n+1}=x_n(2-bx_n)$**としています。**

2.2 加減算

加減算は引数を整数型8バイトの60ビットのみ使用する配列
に展開します。準備する配列要素の数は$4080 \div 60 + 1 = 69$
となります。引数を正の整数の演算になる様に引数を組み替えます。
(この処理は加算及び減算に共通です.。.)
この処理は低精度演算のものを定数を変更するだけですので
前処理サブルーチンに分けます。(ソースステップ数約150)
隠れビットと丸めようのビットのために、先頭配列$ix(69)$の値は
加算$2 \le ix(69) \le 3$　　減算$4 \le ix(69) \le 7$となります。
引数の大きい方を配列ix,小さい方を配列iyにいれます。
そして指数部の差分だけ配列iyをシフトして,桁合わせをします。
ここまでは加算及び減算に共通です。ビット位置が1つことなるので
サブルーチンとしての抜出はしていません。
多倍長整数加算または乗算を行い結果を配列izにいれます。
この段階では整数型8バイト配列izは各要素60ビットしか使用
いません.ここで指数部と丸め処理をします。
加算の場合$iz(69) \ge 4$　桁上がりあり,$iz(69) \le 3$　　桁上がりなし
減算の場合$iz(69) \ge 4$　桁下がりなし,$2 \le iz(69) \le 3$　桁下がり1ビット
このケースの場合,ともに丸め処理後,指数部の値と仮数部を配列izの
68要素にいれます。処理がことなるのは,減算で$iz(69) = 1$の場合は
桁下がり2ビットで丸め処理は有りません。$iz(69) = 0$の場合は
$iz(68), iz(67)$.....と調べて最初に1があるビット位置を求める処理が
ある事です。これは$iz(l) \ne 0, iz(l) = 2^m + k(1 \le k \le 2^m - 1)$を実数にすると
mが指数部の値に一致する事を使用します。これで指数部の値と
仮数部を配列izの68要素にいれます。($iz(69) = 0$の場合は前につめ、
下位の配列要素には0をいれます。)。この後は加減算,乗算共通で
60ビット68要素配列を64ビット64要素配列に詰め,128倍精度浮動小数点
数を作成します。とくにここもサブルーチン化して抜き出さなくてむ、コメント
なども含めても約350ステップ数内に収まっています。

2.3 乗算

2つの引数から積の符号と指数部の値を計算して引数を
4080ビットの正整数にします。これを整数型4バイトの30ビット
のみ使用する配列に展開します。準備する配列要素の数は
$4080 \div 30 + 1 = 137$となります。二つの引数を配列ix, iyにいれ
多倍長正整数乗算を行い,結果を整数型4倍バイトの30ビット
使用する配列izにいれます.(izの要素数は274)。
$ix(137), iy(137)$には隠れビット1をいれますので,桁上がりがあるか
どうかは$iz(273)$の値で求まり,これで丸めビットの位置がわかります。
$iz(273) \geq 2$だと桁上がりがあり,$iz(272) = 1$だと桁上がりが有りません。
ここで整数型8バイト配列izz(配列要素数69)を用意して,$iz->izz$
にいれ,加減算と同じ処理をして,128倍精度浮動小数点数を作成
します。これらを含めて300ステップ以下で作成できています。

2.4 作成作業に関して

整数演算方式の作成の作業量は連続して使用
できるビット数によります。例えば8192ビット使用
できるとすれば、ソースステップ数は128倍精度
演算は20-30程度で行う事が可能と言えます。
また丸め処理は行わない(例えば切り捨て)と
すればやはりソースステップ数が大きく削減
されます。FPGA等で作成する場合も
ほとんど同じ様になります。

3.128倍精度演算の精度

量子モンテカルロ法による物性スペクトル計算で演算精度に影響があるのは、逆行列を求める計算とグラムシュミット法の計算です。
そこでヒルベルト行列を使用した連立一次方程式の求解と平方根計算の精度を求めました。

また128倍精度演算では,量子モンテカルロ法による物性スペクトル計算では表現可能な数値範囲の制限にかかる可能性があり,問題発生時にその原因が有効ビット数不足なのか、表現可能な数値範囲の制限によるものかの切り分けのため、ヒルベルト行列では188倍精度演算,248倍精度演算,308倍精度演算,368倍精度演算,428倍精度演算,488倍精度演算でも実行しています。

3.1 平方根計算

テスト内容は以下のものです。

$$a = 10^n \ (n = 4931, 4932)$$

$$d = \frac{1}{a}$$

$$b = \sqrt{a}$$

$$c = \sqrt{d}$$

```
sqrt div limit test
 a=  1.0000000000000000000000000000000E+4931
 b=  3.1622776601683793319988935444327 18E-2466
 c=  9.9999999999999999999999999999999E-4932
 d=  3.1622776601683793319988935444327 18E+2465
```

$a = 10^{4931}$ では、除算, 平方根は正しく計算できています。

```
sqrt div limit test
 a=  9.99999999999999999999999999999999E+4931
 b=  0.00000000000000000000000000000000E+0000
 c=  0.00000000000000000000000000000000E+0000
 d=  9.99999999999999999999999999999999E+2465
```

$a = 10^{4932}$ の場合,除算が正しく実行されていません。

これは指数部の値が0の場合には0としている事によります。

加減算,乗算などのアンダーフロー処理を容易にするために

行っています。このため最小値の10の指数は

$-16382 \times \log_{10} 2 = -4931.473389.....$となります

3.2 ヒルベルト行列

ヒルベルト行列のサイズNと条件数には近似的には以下の様になっています。

ヒルベルト行列の条件数

n	条件数 (ビット数表示)
400	2022
450	2276
500	2530
550	2785
600	3039
650	3293
700	3547
750	3801
800	4056
850	4310
900	4564
950	4818
1000	5072
1050	5327
1100	5581
1150	5835
1200	6089
1250	6344
1300	6598
1350	6852
1400	7106
1450	7361
1500	7615

128倍精度
有効ビット数4081

188 倍精度
有効ビット数　6001

248 倍精度
有効ビット数　　7921

実測結果は以下の様になっています。

ヒルベルト行列の精度		
N	最大誤差	精度
400	1.014E-621	128倍精度
500	1.681E-467	128倍精度
600	1.321E-314	128倍精度
700	1.349E-161	128倍精度
800	7.603E-009	128倍精度
1000	1.013E-280	188倍精度
1100	4.211E-127	188倍精度
1400	1.013E-245	248倍精度

最大誤差のビット数 = 有効ビット数 − 条件数のビット数となっています。

248倍精度演算では$N = 1400$が使用できるメモリ容量の上限となっています。

より高精度演算での精度

N=100でのヒルベルト行列計算				
	条件数0.127E+152		ビット数501	
精度	有効ビット数	誤差		
		実測	理論値	
308	9841	4.73E-2814	2.80E-2812	
368	11761	3.38E-3392	2.52E-3390	
428	13681	1.61E-3971	2.66E-3968	
488	15601	5.36E-4198	2.80E-4546	(注)
488	15601	1.00E-4547	2.80E-4546	
(注)逆数計算で4倍精度の初期値から7回反復した場合の値				
308,368,428倍精度は7回反復,488倍精度は8回反復が				
必要				

488倍精度以上の精度を作成してもこれ以上精度向上はないと言えます。

4.128倍精度演算の性能

比較のため行列サイズN=400のヒルベルト行列による連立一次方程式の求解で実施しました。

ヒルベルト行列のN=400の実行時間一覧表

単位: 秒

方式	精度	x5570	e5430
浮動型	64倍精度	540.603	743.938
浮動型	96倍精度	1551.696	2165.918
浮動型	128倍精度	3467.115	4280.500
整数型	68倍精度	179.059	277.818
整数型	128倍精度	652.896	918.142
整数型	188倍精度	1235.567	1945.560
整数型	248倍精度	2125.342	3337.811

32倍精度変数をつなげる方式に比べ,
整数演算方式の性能が非常に良くなっています。

他の精度での性能

N=100でのヒルベルト行列計算

実行時間（秒）

精度	x5570	e5430
308	50.298	81.123
368	70.468	115.572
428	95.133	156.090
488	123.216	205.238
488	123.253	205.404

反復回数7回

N=400 での8倍精度でのヒルベルト行列計算

Ax=b でAの要素は4倍精度演算で作成

実行時間（秒）

ソース	x5570	e5430
従来	4.686	5.926
新規	4.885	6.354

1.反復回数の増加の影響はほとんどない.

2.8倍精度演算ではソースステップ数の大幅削減をしても性能低下は5%程度に収まっています。

5.量子モンテカルロ法による物性スペクトル計算

5.1　大きなβの場合に関して

量子モンテカルロ法による物性スペクトル計算

では$B_L \ldots\ldots B_1 = QDR$とし$(I + QDR)^{-1}$を求めるのに,

$D + Q^T R^{-1} = Q'D'R'$とQDR分解し,

$Q(D + Q^T R^{-1})R = QQ'D'R'R = Q''D''R'' = I + QDR$から

逆行列を求めます。

$P = e^Q (Q = \sqrt{\frac{\beta U}{L}} (\frac{L+1}{2}))$とすると,

$\beta U \le L$なら結果の絶対値の最小値は$\frac{1}{P}$,

Dの最大数はP^2となり,グラムシュミット法で

平方数の和の平方根を求めるため,

プログラムに現れる最大数はP^4となります。

$\beta U \geq L$, では条件数はβ / Lの大きさ依存する度合いが大きくなる。

これは, $B_L .....B_1 = QDR$とし$(I + QDR)^{-1}$を求める際に発生します。

$D + Q^T R^{-1} = Q'D'R'$とQDR分解し,

$Q(D + Q^T R^{-1})R = QQ'D'R'R = Q''D''R'' = I + QDR$から逆行列を求めます.

この$D + Q^T R^{-1}$のQDR分解の際オーバーフローが発生します.

正確には, $b = \sqrt{a_1{}^2 + a_2{}^2 + ... + a_n{}^2}$の計算の際

$a_i \times a_i$の計算で発生します。

オーバーフローを防ぐには,

a_{max} = 絶対値最大値$(a_i, i = 1,2,...., n)$

として, $b = | a_{max} | \times \sqrt{c_1{}^2 + c_2{}^2 + ... + c_n{}^2}$ $(c_i = \frac{a_i}{a_{max}})$

とします。ここで

正しい答えのでるβ最大数における, D''の最大数S,

$S = e^{\sqrt{\beta UL} \times \alpha}$としてその影響を調査しました。

調査では$n = 20, U = 10, L = 448$と固定しています。

α調査結果

条件　$N = 20, U = 10, L = 448$

精度	β の最大数	最大値	α
68倍精度	732	2.518D+1287	1.64
128倍精度	1620	7.778D+2441	2.09
188倍精度	2590	6.198D+3593	2.43
248倍精度	3609	6.580D+4745	2.72
308倍精度	3776	6.591D+4930	2.76

368倍精度では308倍精度と同じく、
β =3776では正しく計算出来ていて,
β =3777では結果が不正となっている事から
308倍精度の結果が限界を示している
と言えます。

368倍精度演算で正しく計算されている例

```
n=         20
l=        448
beta, dt=   3776.000000000000000000000000000000
  8.428571428571428571428571428571428571
u, t0=   10.000000000000000000000000000000000
  1.000000000000000000000000000000000000
fac=   9.180725150319787550452016865372020
For denominator: mstep=          1
For green func: mstep2=          1

ispin=          1
QDR  =  6.99319981312499889573274808973757 6E+4922
I+QDR=  6.99319981312499889573274808973757 6E+4922


ispin=          2
QDR  =  6.59101777091279783292513311646599 4E+4930
I+QDR=  6.59101777091279783292513311646599 4E+4930

   0.0000000000D+00   -0.1000000000D+01
   0.1879571429D+04   -0.7381849773-890
   0.1888000000D+04   -0.1520746834-893
   0.1896428571D+04   -0.7381849773-890
   0.3776000000D+04   -0.1000000000D+01
```

368倍精度演算でも正しく計算されてない例

```
n=         20
l=        448
beta, dt=   3777.000000000000000000000000000000
  8.430803571428571428571428571428571428
u, t0=   10.000000000000000000000000000000000
  1.000000000000000000000000000000000
fac=   9.181940737898808410372036425164 43
For denominator: mstep=          1
For green func: mstep2=          1

ispin=          1
QDR  =  8.886250347365217688103022558980503E+4923
I+QDR=  8.886250347365217688103022558980503E+4923


ispin=          2
QDR  =  8.395584469058393694708873673477994E+4931
I+QDR=  3.111402625840683560090283433225110E+4770

   0.0000000000D+00   -0.4999999997D+00
   0.1880069196D+04   -0.4586860821-890
   0.1888500000D+04   -0.7614617438-894
   0.1896930804D+04   -0.2814546226-890
   0.3777000000D+04   -0.5000000000D+00
```

N=20,U=10,L=448と固定すると正しく計算出来るβの最大値は3776でα=2.76であった。ここでN=20,U=10でLを小さくした場合正しく計算できた場合は以下の表の様になっています。

n=20,U=10 での実行結果一覧

β	L	最大値	√β UL	α
3776	448	6.571D+4930	1.730D+1786	2.76
4000	320	2.857D+4931	6.014D+1553	3.18
4250	208	6.161D+4919	1.780D+1291	3.81
4500	132	1.077D+4927	2.934D+1058	4.66

この表からβが3776以上になると,βが250増加するとLは0.64倍となる。$\beta = 3776$**と4000ではβが224の増加なのでLは**$250/224 \times 0.64 = 0.714$**倍となっています。**

$\beta_1, U_1, L_1, \alpha_1$**が既知として**$\beta_1 < \beta_2$**の**$\beta_2$**を選ぶと,**$L_2$**が推定され実行する**$U_2$**を決めれば(通常**$= U_1$**)とすると**

$\alpha_2 = \alpha_1 \sqrt{\dfrac{\beta_1 U_1 L_1}{\beta_2 U_2 L_2}}$**からα₂が求まり実行可能かどうかがわかる。**

$\beta = 4750, L = 84, \alpha = 5.686$　　**最大値**$= 10^{4932.6}$

$\beta = 5000, L = 54, \alpha = 6.912$　　**最大値**$= 10^{4932.5}$

と計算が困難になる事がわかります。

5.2 パラメータ領域拡大結果

測定結果一覧　x5570

測定条件　$N=100, U=10, L=448$

βと絶対温度Kとの関係

絶対温度$(K)=10000/\beta$

β	精度	絶対値最小値		実行時間
		実測値	理論値	(秒)
730	68倍精度	0.824D-384	0.269D-383	57642
1600	128倍精度	0.170D-582	0.214D-582	185097
2500	188倍精度	0.773D-728	0.462D-728	403155
10000/3	248倍精度	0.292D-840	0.978D-841	673631
3770	308倍精度	0.156D-893	0.397D-894	975735

この条件では,32**倍精度**$\beta=250$(**絶対温度**40K)
から,68**倍精度で**13.7K,128**倍精度で**6.26K
188**倍精度で**4K,248**倍精度で**3K,308**倍精度**
で2.65K**までパラメータ領域が拡大されました。**
これ以上の演算精度を用いても改善はされませんの
でこの値が限界と言えます。