クォークソルバーの計算時間見積りのためのノート

広島大学理学研究科　石川 健一

1　Wilson fermion のホッピング行列の定義

QCD Mult は Wilson フェルミオンを定義するときに、普通のディラックフェルミオンの作用の４次元の１階の偏微分演算子を格子上の対称差分に直したものに、格子上で現れるカイラルダブラーを消す Wilson 項 (４次元の２階差分) を付け加えたものである。格子上の Wilson フェルミオンの作用は以下の通りである。

$$S = \sum_{n,m,\alpha,\beta,a,b} \overline{\psi}^a_\alpha(n) D(n,m)^{a,b}_{\alpha,\beta} \psi^b_\beta(m), \tag{1}$$

ここで、n は４次元格子点 $n=(n_x,n_y,n_z,n_t)$、α,β はスピンの自由度 (長さ 4)、a,b はカラーの自由度 (長さ 3) である。$\psi(n)$ は複素数。$D(n,m)^{a,b}_{\alpha,\beta}$ は

$$D(n,m)^{a,b}_{\alpha,\beta} = \delta^{a,b}\delta_{\alpha,\beta}\delta_{n,m} - \kappa \ \ \mathrm{Mult}(n,m)^{a,b}_{\alpha,\beta}, \tag{2}$$

の様に分解される。κ はクォークの質量に関するパラメータ、Mult がここで問題にしている QCD Mult の演算子である。$\delta_{i,j}$ は $i=j$ の時 1、$i\neq j$ の時 0 となるクロネッカーデルタである。

Mult は以下のようになっている。

$$\mathrm{Mult}(n,m)^{a,b}_{\alpha,\beta} = \sum_{\mu=1}^{4} \left[(1-\gamma_\mu)_{\alpha,\beta} \left(U_\mu(n)\right)^{a,b} \delta_{n+\hat{\mu},m} + (1+\gamma_\mu)_{\alpha,\beta} \left(U^\dagger_\mu(n-\hat{\mu})\right)^{a,b} \delta_{n-\hat{\mu},m} \right], \tag{3}$$

ここで μ は４次元の方向を表し、μ が 1 の時 x 方向、μ が 1 の時 y 方向、μ が 1 の時 z 方向、μ が 1 の時 t 方向とする。$U_\mu(n)$ はグルーオンの自由度を表すリンク変数である。これは 3×3 の複素行列でその値は $SU(3)$ の群上の値を取る。また μ の添字を持つベクトルでもある。$\hat{\mu}$ は μ 方向を向いている単位ベクトルである。すなわち $n+\hat{\mu}$ は格子点 n から μ 方向に $+1$ だけずれた点を表す。γ_μ はディラック行列で以下の式を満たす 4×4 の複素行列である。

$$\gamma_\mu\gamma_\nu + \gamma_\nu\gamma_\mu = \delta_{\mu,\nu}. \tag{4}$$

一般にこの式を満たす γ_μ の組 ($\mu=1,2,3,4$) は一意に決まらない (ユニタリー同値なものがある)。

1.1　Dirac 行列の定義 (ディラック表示)

ここに γ_μ の一つの表現を与える。

$$\gamma_1 = \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & -i & 0 \\ 0 & i & 0 & 0 \\ i & 0 & 0 & 0 \end{pmatrix}, \qquad \gamma_2 = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix},$$

$$\gamma_3 = \begin{pmatrix} 0 & 0 & -i & 0 \\ 0 & 0 & 0 & i \\ i & 0 & 0 & 0 \\ 0 & -i & 0 & 0 \end{pmatrix}, \qquad \gamma_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \tag{5}$$

である。

1.2　計算手順

計算は $t \to z \to y \to x$ の順で差分計算していく。演算量を見積もるため計算の具体的な式を以下に記す。

1. t 方向の差分を取る。

$$
\begin{aligned}
(1-\gamma_4) &= \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}, \\
(1+\gamma_4) &= \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},
\end{aligned}
$$

より、

$$My_3^a(n) := 2 \times \sum_{b=1}^{3} U_4^{a,b}(n) \times y_3^b(n+\hat{4}), \tag{6}$$

$$My_4^a(n) := 2 \times \sum_{b=1}^{3} U_4^{a,b}(n) \times y_4^b(n+\hat{4}), \tag{7}$$

$$My_1^a(n) := 2 \times \sum_{b=1}^{3} (U_4^{b,a}(n-\hat{4}))^* \times y_1^b(n-\hat{4}), \tag{8}$$

$$My_2^a(n) := 2 \times \sum_{b=1}^{3} (U_4^{b,a}(n-\hat{4}))^* \times y_2^b(n-\hat{4}), \tag{9}$$

を $a = 1, 2, 3$ と n 格子点に対して計算。

2. z 方向の差分を取る。

$$
\begin{aligned}
(1-\gamma_3) &= \begin{pmatrix} 1 & 0 & i & 0 \\ 0 & 1 & 0 & -i \\ -i & 0 & 1 & 0 \\ 0 & i & 0 & 1 \end{pmatrix}, \\
(1+\gamma_3) &= \begin{pmatrix} 1 & 0 & -i & 0 \\ 0 & 1 & 0 & i \\ i & 0 & 1 & 0 \\ 0 & -i & 0 & 1 \end{pmatrix},
\end{aligned}
$$

より、

$$
\begin{aligned}
h1p^a &:= \sum_{b=1}^{3} U_3^{a,b}(n) \times (y_1^b(n+\hat{3}) + i\ y_3^b(n+\hat{3})), && (10)\\
h2p^a &:= \sum_{b=1}^{3} U_3^{a,b}(n) \times (y_2^b(n+\hat{3}) - i\ y_4^b(n+\hat{3})), && (11)\\
h1n^a &:= \sum_{b=1}^{3} (U_3^{b,a}(n-\hat{3}))^* \times (y_1^b(n-\hat{3}) - i\ y_3^b(n-\hat{3})), && (12)\\
h2n^a &:= \sum_{b=1}^{3} (U_3^{b,a}(n-\hat{3}))^* \times (y_2^b(n-\hat{3}) + i\ y_4^b(n-\hat{3})), && (13)\\
My_1^a(n) &:= My_1^a(n) + h1p^a + h1n^a, && (14)\\
My_2^a(n) &:= My_2^a(n) + h2p^a + h2n^a, && (15)\\
My_3^a(n) &:= My_3^a(n) - i\ h1p^a + i\ h1n^a, && (16)\\
My_4^a(n) &:= My_4^a(n) + i\ h2p^a - i\ h2n^a, && (17)
\end{aligned}
$$

となる。ここで $h1p^a, h2p^a, h1m^a, h2m^a$ は複素数の作業変数。

3. y 方向の差分を取る。

$$
\begin{aligned}
(1-\gamma_2) &= \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix},\\
(1+\gamma_2) &= \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix},
\end{aligned}
$$

より、

$$
\begin{aligned}
h1p^a &:= \sum_{b=1}^{3} U_2^{a,b}(n) \times (y_1^b(n+\hat{2}) + y_4^b(n+\hat{2})), && (18)\\
h2p^a &:= \sum_{b=1}^{3} U_2^{a,b}(n) \times (y_2^b(n+\hat{2}) - y_3^b(n+\hat{2})), && (19)\\
h1n^a &:= \sum_{b=1}^{3} (U_2^{b,a}(n-\hat{2}))^* \times (y_1^b(n-\hat{2}) - y_4^b(n-\hat{2})), && (20)\\
h2n^a &:= \sum_{b=1}^{3} (U_2^{b,a}(n-\hat{2}))^* \times (y_2^b(n-\hat{2}) + y_3^b(n-\hat{2})), && (21)\\
My_1^a(n) &:= My_1^a(n) + h1p^a + h1n^a, && (22)\\
My_2^a(n) &:= My_2^a(n) + h2p^a + h2n^a, && (23)\\
My_3^a(n) &:= My_3^a(n) - h2p^a + h2n^a, && (24)\\
My_4^a(n) &:= My_4^a(n) + h1p^a - h1n^a, && (25)
\end{aligned}
$$

となる。ここで $h1p^a, h2p^a, h1m^a, h2m^a$ は複素数の作業変数。

4. x 方向の差分を取る。

$$
\begin{aligned}
(1-\gamma_1) &= \begin{pmatrix} 1 & 0 & 0 & i \\ 0 & 1 & i & 0 \\ 0 & -i & 1 & 0 \\ -i & 0 & 0 & 1 \end{pmatrix}, \\
(1+\gamma_1) &= \begin{pmatrix} 1 & 0 & 0 & -i \\ 0 & 1 & -i & 0 \\ 0 & i & 1 & 0 \\ i & 0 & 0 & 1 \end{pmatrix},
\end{aligned}
$$

より、

$$
\begin{aligned}
h1p^a &:= \sum_{b=1}^{3} U_1^{a,b}(n) \times (y_1^b(n+\hat{1}) + i\ y_4^b(n+\hat{1})), && (26)\\
h2p^a &:= \sum_{b=1}^{3} U_1^{a,b}(n) \times (y_2^b(n+\hat{1}) + i\ y_3^b(n+\hat{1})), && (27)\\
h1n^a &:= \sum_{b=1}^{3} (U_1^{b,a}(n-\hat{1}))^* \times (y_1^b(n-\hat{1}) - i\ y_4^b(n-\hat{1})), && (28)\\
h2n^a &:= \sum_{b=1}^{3} (U_1^{b,a}(n-\hat{1}))^* \times (y_2^b(n-\hat{1}) - i\ y_3^b(n-\hat{1})), && (29)\\
My_1^a(n) &:= My_1^a(n) + h1p^a + h1n^a, && (30)\\
My_2^a(n) &:= My_2^a(n) + h2p^a + h2n^a, && (31)\\
My_3^a(n) &:= My_3^a(n) - i\ h2p^a + i\ h2n^a, && (32)\\
My_4^a(n) &:= My_4^a(n) - i\ h1p^a + i\ h1n^a, && (33)
\end{aligned}
$$

となる。ここで $h1p^a, h2p^a, h1m^a, h2m^a$ は複素数の作業変数。

2 $O(a)$ 改良。

格子 QCD で連続極限を取ることは重要だが、実際に連続極限を取るためには複数の有限格子間隔 a で計算を実行し得られたデータを $a \to 0$ へ外挿する必要がある。

ここで式 (2) で表されるフェルミオンには $O(a)$ の有限格子間隔のエラーがある。現在使われているスーパーコンピュータを用いても $1/a = 2-3$ GeV がやっとのことであり、$O(a)$ の誤差は $\Lambda_{\rm QCD} \sim 0.3-0.5$GeV とすると、$O(a\Lambda_{\rm QCD}) \sim 0.1-0.3$ となり 10%から 20%の系統誤差を生じる。この下で連続極限を取る操作は大きな外挿誤差を生じるであろう。

この困難を回避するため、式 (2) にクローバー項と言うものを適切な係数で付け加えること二よりこの $O(a)$ の系統誤差を取り除けることが分かっている。この操作により連続極限での外挿誤差を著しく減らすことができると考えている。

クローバー項を付け加えた場合、式 (2) は以下のようになる。

$$
\begin{aligned}
D_{\rm CL}(n,m)^{a,b}_{\alpha,\beta} &= \delta^{a,b}\delta_{\alpha,\beta}\delta_{n,m} + c_{\rm SW} F^{a,b}_{\alpha,\beta}(n)\delta_{n,m} - \kappa\ \ {\rm M}(n,m)^{a,b}_{\alpha,\beta}, \\
&= F(n)^{a,b}_{\alpha,\beta}\delta_{n,m} - \kappa\ \ {\rm M}(n,m)^{a,b}_{\alpha,\beta} && (34)
\end{aligned}
$$

ここで、

$$
F^{a,b}_{\alpha,\beta}(n) = \delta^{a,b}\delta_{\alpha,\beta} + c_{\rm SW} F^{a,b}_{\alpha,\beta}(n)、 \qquad (35)
$$

である。$c_{\rm SW}$ が $O(a)$ の誤差を取り除くように調整される係数、$F^{a,b}_{\alpha,\beta}(n)$ はリンク変数より作られる 12×12 の格子点に依存した行列である。F はエルミート行列である。

実際の数値計算では以下のような変形された式を使う。

$$D_{\rm CL}(n,m)^{a,b}_{\alpha,\beta}=\delta^{a,b}\delta_{\alpha,\beta}\delta_{n,m}-\kappa\ \ (F^{-1})^{a,b}_{\alpha,\beta}(n){\rm M}(n,m)^{a,b}_{\alpha,\beta}, \tag{36}$$

このばあい、M の演算の後、F^{-1} を各格子点で掛ける。

ディラック表示では F^{-1} は 2 つの 6×6 のエルミート行列の形で保存できる (ブロック対角な形)。

2 つの 6×6 のエルミート行列はそれぞれ以下のようなベクトルの形で保存できる。

$$F_i^{-1}=\begin{pmatrix} {\rm fclinv}(1,i) & {\rm fclinv}(2,i) & {\rm fclinv}(3,i) & {\rm fclinv}(4,i) & {\rm fclinv}(5,i) & {\rm fclinv}(6,i)\\ & {\rm fclinv}(7,i) & {\rm fclinv}(8,i) & {\rm fclinv}(9,i) & {\rm fclinv}(10,i) & {\rm fclinv}(11,i)\\ & & {\rm fclinv}(12,i) & {\rm fclinv}(13,i) & {\rm fclinv}(14,i) & {\rm fclinv}(15,i)\\ & & & {\rm fclinv}(16,i) & {\rm fclinv}(17,i) & {\rm fclinv}(18,i)\\ & & & & {\rm fclinv}(19,i) & {\rm fclinv}(20,i)\\ & & & & & {\rm fclinv}(21,i) \end{pmatrix}, \tag{37}$$

ここで $i=1,2$、fclinv は長さ 21 の複素ベクトル。エルミート行列なので独立な上三角成分のみを保持する。これを用いて F^{-1} を表すと

$$F^{-1}=\begin{pmatrix} F_1^{-1}+F_2^{-1} & -F_1^{-1}+F_2^{-1}\\ -F_1^{-1}+F_2^{-1} & F_1^{-1}+F_2^{-1}\end{pmatrix} \tag{38}$$

3　5 次元有効理論 (オーバーラップフェルミオン)

オーバーラップフェルミオンの演算子 $D_{\rm ov}$ は

$$D_{\rm ov}=\frac{1+\mu}{2}+\frac{1-\mu}{2}\gamma_5{\rm sign}(H_w(M_{\rm dwh}), \tag{39}$$

となる。ここで、

$$H_W(M_{\rm dwh})=\gamma_5\left((4-M_{\rm dwh})-\frac{1}{2}M\right), \tag{40}$$

である。行列の符号関数を含んでいる。連立方程式

$$D_{\rm ov}x=b, \tag{41}$$

を解くことはこのままでは符号関数の計算を毎回行列ベクトル積のたびに計算するので困難である。また、係数行列が密行列となるので有効な前処理が見付かっていない。現在では一度、符号関数を近似した後 5 次元時空の演算子に変形し 4 次元時空で疎行列の係数行列をもつ連立方程式に変形し、その解から欲しい 4 次元時空の解を抜き出す方法の方が効率がよいことが分かっている。効率がよいのは 5 次元時空の演算子が 4 次元時空で疎行列の構造を持つため格子点に基づく前処理を行なうことができることも関係している。

符号関数の近似方法には大まかに 3 種類ある。一つはケーリー変換 (Euclidean Cayley transformation)/多重極展開に基づく方法、一つは部分分数展開に基づく方法、一つは連分数展開に基づく方法である。

変形された連立方程式は、以下の様になる。

$$\begin{aligned} B &= Pb, &(42)\\ D_{\rm 5Deff}X &= B, &(43)\\ x &= QX. &(44)\end{aligned}$$

P,Q の具体的形はどの近似を用いたかによって異なる。$D_{5\mathrm{Deff}}$ も具体的形はどの近似を用いたかによって異なるが以下の共通の形を取る。

$$D_{5\mathrm{Deff}} = K - \frac{1}{2}M, \quad \text{or} \quad D_{5\mathrm{Deff}} = 1 - \frac{1}{2}LM. \tag{45}$$

ここで、K, L は

$$\begin{aligned} K_{a,\alpha;b,\beta}(\bar{n},\bar{m}) &= K_{\alpha;\beta}(n_5,m_5)\delta(n,m)\delta_{a,b} && (46)\\ K_{\alpha;\beta}(n_5,m_5) &= K^{-}(n_5,m_5)(1-\gamma_5)_{\alpha,\beta} + K^{+}(n_5,m_5)(1+\gamma_5)_{\alpha,\beta} && (47)\\ K^{-}(n_5,m_5) &= K^{+}(m_5,n_5) && (48) \end{aligned}$$

を満たす。また、M の部分は 4 次元のホッピング行列を 5 次元方向に繰り返し掛ける演算を行なう。LM の演算が支配的になるが，L,K は定数行列なので再利用しやすい。またリンク変数は 4 次元座標のみに依存し 5 次元方向に繰り返しホッピング行列を演算する時には再利用出来る (5 次元後方の添字をメモリ連続アクセスするようにデータ構造を定義)。

格子カイラル対称性の近似による破れを抑えるにはホッピング行列 H_W の固有値の絶対値の下限と上限の間の幅を抑える必要がある。固有値シフト法では H_W の上側下側の固有値、固有ベクトルを複数求めておき、H_W 上側下側の固有値を符号を変えないように $+1$ や -1 にシフトして近似がよくなるようにする。H_W のシフトしたい固有値固有ベクトルを

$$\begin{aligned} H_W V_k &= V_k \Lambda_k, && (49)\\ V_k &= (v_1, v_2, \cdots, v_k), && (50)\\ \Lambda_k &= \mathrm{diag}\,(\lambda_1, \lambda_2, \cdots, \lambda_k), && (51) \end{aligned}$$

と k 本、あらかじめ求めてあるとする。固有値をシフトしたものを $\tilde{H}_W$ とすれば

$$\begin{aligned} \tilde{H}_W &= H_W + V_k X_k V_k^{\dagger}, && (52)\\ X_k &= \mathrm{Shift}(\Lambda_k) - \Lambda_k, && (53) \end{aligned}$$

である。この変化はホッピング行列や K, L の変更に反映されて有効 5 次元係数行列は、

$$\tilde{D}_{5\mathrm{Deff}} = \tilde{K} - \frac{1}{2}(M + W_k Y_k Z_k^{\dagger}), \quad \text{or} \quad \tilde{D}_{5\mathrm{Deff}} = 1 - \frac{1}{2}\tilde{L}(M + W_k Y_k Z_k^{\dagger}). \tag{54}$$

の様になる。$\tilde{K}, \tilde{L}$, Y_k ($k \times k$ 行列), $W_k = (w_1, w_2, \cdots, w_k)$, $Z_k = (z_1, z_2, \cdots, z_k)$, ($w_j, z_j$:4d-fermion vector) の具体的計算式は符号関数をどのような近似で計算しているかに依存する。この係数行列を用いて解けば格子カイラル対称性を計算精度でコントロールしたクォーク伝搬関数を求めることが出来る。

計算コストとしてあらかじめ、H_W の固有値固有ベクトルを求めておき W_k, Z_k を作っておく必要がある。有効 5 次元疎行列ベクトル積の計算のたびに射影計算 $W_k Y_k Z_k^{\dagger}$ が必要になるシフトした $\tilde{H}_W$ の固有値の範囲を、必要な格子カイラル対称性の度合に応じて決めるが、シフトする必要のある固有値の数はゲージ配位に依存していて固定ではないので、たくさんの固有値を求める必要がある場合が出てくる可能性がある。射影部分の具体的な計算式は

$$\begin{aligned} a_j(s) &= \sum_n z_j(n)^{\dagger} y(s,n), && (55)\\ c_i(s) &= \sum_{j=1}^{k} (Y_k)_{i,j} a_j(s), && (56)\\ h(s,n) &= h(s,n) + \sum_{i=1}^{k} w_i(n) c_i(s), && (57) \end{aligned}$$

である。4 次元格子でのフェルミオンベクトルの内積を $k \times N_5$ 回行ない $a_j(s)$ を求める。これに Y_k の $k \times k$ を掛ける演算を N_5 回行ない、$c_i(s)$ を求める。ホッピング行列の計算の中で $h(s,n)$ に $w_i(n)c_i(s)$ を足し込んでむ計算を k 回行なう。

4　演算数など。

複素数の掛け算と足し算の演算数

$$z = u * v \quad \leftrightarrow \quad z_r = u_r * v_r - u_i * v_i, z_i = u_r * v_i + u_i * v_r, \tag{58}$$

より、 6 flop。

$$z = u + v \quad \leftrightarrow \quad z_r = u_r + v_i, z_i = u_i + v_i, \tag{59}$$

より、 2 flop。
　実部と虚部の入れ替えや i を掛ける演算、符号の付け替えは 0 flop とする。

SU(3) 行列ベクトル積の演算数

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} u_{11} * y_1 + u_{12} * y_2 + u_{13} * y_3 \\ u_{21} * y_1 + u_{22} * y_2 + u_{23} * y_3 \\ u_{31} * y_1 + u_{32} * y_2 + u_{33} * y_3 \end{pmatrix}, \tag{60}$$

は複素数の掛け算が 9, 足し算が 6 である。従って $9 * 6 + 6 * 2 = 66$ flop。積和算 (FMA) で書くと 6(1 flop+5 FMA) = 6 flop+30 FMA。

スピノールのプロジェクションの演算数　スピノールのプロジェクション演算は符号の付け替えと実部虚部の入れ替えと足し算のみを使う。x, y, z 方向に関してはスピノール 2 成分を計算し、残りの成分は実部と虚部の入れ替え符号の付け替えで実現出来る。t 方向に関してはスピノール 2 成分のみに 2 を掛ける。
　t 方向のスピノールのプロジェクションの演算数はそれぞれ 2 成分のスピノールに対して 2 を掛ける計算より、 $6 * 2 = 12$ flop。この結果は一時変数に代入される。x, y, z 方向のスピノールのプロジェクションの演算数はそれぞれ 2 成分のスピノールに対しての足し算より、 $6 * 2 = 12$ flop。これらの結果から残りの成分を実部と虚部の入れ替え符号の付け替えを行ない生成する。これらの 4 成分を一時変数に足し込むのでさらに 24 flop 必要。結局 36 flop 必要。

クローバー項の掛け算の演算数　クローバー項の掛け算は 6×6 のエルミート行列を掛ける演算を 2 回と、その前後にカイラル表示とディラック表示の間のスピノールの変換を行なう。スピノールの変換は足し算で行なわれる。$12 * 2$ 回の複素数の足し算は 48 flop。6×6 のエルミート行列を複素 6 成分ベクトルに掛ける演算は複素数の掛け算が 6 回と足し算が 5 回を 6 回行なうので $(6 * 6 + 5 * 2) * 6 = 276$ flop。これを 2 回行なうので 552 flop。全体で 600 flop 必要。

Wilson 型の hopping 行列の演算コスト　格子点 1 点当たりに必要な演算数、データロード数、データストア数は以下の通り。

方向	演算数 (flop)[FP レジスタ数]	データロード数 (cmplx)	データストア数 (cmplx)
$t+$	$66*2+12=144$[18+12+(12)]	(6+9) (mem)	12 (reg)
$t-$	$66*2+12=144$[18+12+(12)]	(6+9) (mem)	12 (reg)
$z+$	$66*2+36=168$[18+24+(24)]	(12+9) (mem) 12 (reg)	12 (reg)
$z-$	$66*2+36=168$[18+24+(24)]	(12+9) (mem) 12 (reg)	12 (reg)
$y+$	$66*2+36=168$[18+24+(24)]	(12+9) (mem) 12 (reg)	12 (reg)
$y-$	$66*2+36=168$[18+24+(24)]	(12+9) (mem) 12 (reg)	12 (reg)
$x+$	$66*2+36=168$[18+24+(24)]	(12+9) (mem) 12 (reg)	12 (reg)
$x-$	$66*2+36=168$[18+24+(24)]	(12+9) (mem) 12 (reg)	12 (reg/mem)
clover	600[42*2+24+24+(24)]	(21*2) (mem) 12 (reg)	12 (mem)
total(wilson)	1296	156 (mem) 72 (reg)	12 (mem) 84 (reg)
total(clover)	1896	198 (mem) 84 (reg)	12 (mem) 96 (reg)

表 1: 方向毎の演算、データロード、データストア数。(mem) はメモリ/キャッシュからのロード。(reg) は一時変数で全体に渡って共通の領域なので register に割り当てるのが妥当なロード ストア先。

reg からのロード ストアを無視すると格子点あたりの演算とロード ストアの比は表 2 になる。

Wilson hopping	7.714 flop/complex	0.9643 flop/byte(S.P.)	0.4821 flop/byte(D.P.)
(1296 flop):(168 complex)	0.1296 complex/flop	1.037 byte/flop(S.P.)	2.074 byte/flop(D.P.)
Clover hopping	9.029 flop/complex	1.129 flop/byte(S.P.)	0.5643 flop/byte(D.P.)
(1896 flop):(210 complex)	0.1108 complex/flop	0.8861 byte/flop(S.P.)	1.772 byte/flop(D.P.)

表 2: 格子点あたりの演算とロード ストアの比

キャッシュによるメモリロード の削減　理想的な状況を考える。ある方向を向いたリンク変数は一度ロード すると前方差分部分と後方差分部分で同じデータが 2 回使用される。入力のスピノールはある格子点のデータが 4 次元の前方差分部分と後方差分部分で使用されるので同じデータが 8 回使用される。従って一度ロードしたデータをキャッシュに置くならば次の使用ではメモリアクセスすることなくデータを利用できる。格子点あたりのリンク変数のメモリロード 数は半分になり、スピノールのメモリロード 数は 1/8 になる。

キャッシュが理想的に働いた場合の演算数、データロード 数、データストア数は以下の通り。

方向	演算数 (flop)	データロード数 (cmplx)	データストア数 (cmplx)
$t+$	144	$(12/8+9/2)$ (mem)$(12*7/8+9/2)$ (cache)	12 (reg)
$t-$	144	$(12/8+9/2)$ (mem)$(12*7/8+9/2)$ (cache)	12 (reg)
$z+$	168	$(12/8+9/2)$ (mem)$(12*7/8+9/2)$ (cache) 12 (reg)	12 (reg)
$z-$	168	$(12/8+9/2)$ (mem)$(12*7/8+9/2)$ (cache) 12 (reg)	12 (reg)
$y+$	168	$(12/8+9/2)$ (mem)$(12*7/8+9/2)$ (cache) 12 (reg)	12 (reg)
$y-$	168	$(12/8+9/2)$ (mem)$(12*7/8+9/2)$ (cache) 12 (reg)	12 (reg)
$x+$	168	$(12/8+9/2)$ (mem)$(12*7/8+9/2)$ (cache) 12 (reg)	12 (reg)
$x-$	168	$(12/8+9/2)$ (mem)$(12*7/8+9/2)$ (cache) 12 (reg)	12 (reg/mem)
clover	600	$(21*2)$ (mem) 12 (reg)	12 (mem)
total(wilson)	1296	48 (mem) 120(cache) 72 (reg)	12 (mem) 84 (reg)
total(clover)	1896	90 (mem) 120(cache) 84 (reg)	12 (mem) 96 (reg)

表 3: 方向毎の演算、データロード、データストア数。(mem) はメモリ/キャッシュからのロード。(reg) は一時変数で全体に渡って共通の領域なので register に割り当てるのが妥当なロード ストア先。

キャッシュからのロード ストアを無視すると、格子点あたりの演算とロード ストアの比は表 6 の様になる。

Wilson hopping	21.600 flop/complex	2.7 flop/byte(S.P.)	1.35 flop/byte(D.P.)
(1296 flop):(60 complex)	0.04630 complex/flop	0.3704 byte/flop(S.P.)	0.7407 byte/flop(D.P.)
Clover hopping	18.588 flop/complex	2.3235 flop/byte(S.P.)	1.1618 flop/byte(D.P.)
(1896 flop):(102 complex)	0.05330 complex/flop	0.4304 byte/flop(S.P.)	0.8608 byte/flop(D.P.)

表 4: 格子点あたりの演算とロード ストアの比

全てのデータがキャッシュに乗る場合はキャッシュからのロード ストアと演算の比が再び問題となるがこの時の比は表 2 と同じ値になる。

SU(3) 行列再構築によるメモリロード の削減　リンク変数は SU(3) 行列なので 9 つの全ての成分が独立ではない。理想的には 8 つの実数で表現できるが 8 つの実数から元の SU(3) 行列を再構築するには $\sqrt{\ }$ や $\sin, \cos$ などの初等関数の演算が必要で再構築のコストが大きい。そのほかの方法として SU(3) 行列のひとつの行 (又は列) を他の二つの行 (又は列) から特殊ユニタリ性を用いて計算することを利用することを考える。

$$U = \begin{pmatrix} (u_1)_1 & (u_2)_1 & (u_3)_1 \\ (u_1)_2 & (u_2)_2 & (u_3)_2 \\ (u_1)_3 & (u_2)_3 & (u_3)_3 \end{pmatrix} = (u_1, u_2, u_3)\,, \tag{61}$$

と置く。u_i は SU(3) 行列 U の i 列目の縦ベクトルとする。この時 SU(3) の特殊ユニタリ性から、

$$u_3 = (u_1 \times u_2)^*, \tag{62}$$

が導かれる。$\times$ は外積、* は複素共役である。メモリ上のリンク変数は u_1, u_2 のみをロードし Eq. (62) を用いて u_3 を計算すれば U を再構築できる。

この場合メモリからのロード量は 2/3 になる。Eq. (62) の演算量は複素数の積算が 6 回加減算が 3 回必要なので $6 * 6 + 3 * 2 = 42$ flop 増加する。積和算が出来る場合は $(2 + 6$ FMA$) * 3 = 6$ flop $+18$ FMA である。

もともとのリンク変数のデータ構造から u_3 を除いたデータ構造にしておかないと u_1, u_2 のロード時にいっしょにメモリ上の u_3 もキャッシュにロード されてメモリバンド幅を圧迫してしまうので注意が必要である。クォーク伝搬関数の反復解法を行なう直前にデータ構造の変換圧縮を行ない反復解法中は、圧縮されたリンク変数を使うようにプログラムを作成する必要があることに注意する。

このデータ圧縮法を使用した場合の演算数、データロード ストアの表は 5 になる。

方向	演算数 (flop)	データロード数 (cmplx)	データストア数 (cmplx)
$t+$	$42+66*2+12=186$	$(6+9*2/3)$ (mem)	12 (reg)
$t-$	$42+66*2+12=186$	$(6+9*2/3)$ (mem)	12 (reg)
$z+$	$42+66*2+36=210$	$(12+9*2/3)$ (mem) 12 (reg)	12 (reg)
$z-$	$42+66*2+36=210$	$(12+9*2/3)$ (mem) 12 (reg)	12 (reg)
$y+$	$42+66*2+36=210$	$(12+9*2/3)$ (mem) 12 (reg)	12 (reg)
$y-$	$42+66*2+36=210$	$(12+9*2/3)$ (mem) 12 (reg)	12 (reg)
$x+$	$42+66*2+36=210$	$(12+9*2/3)$ (mem) 12 (reg)	12 (reg)
$x-$	$42+66*2+36=210$	$(12+9*2/3)$ (mem) 12 (reg)	12 (reg/mem)
clover	600	$(21*2)$ (mem) 12 (reg)	12 (mem)
total(wilson)	1632	132 (mem) 72 (reg)	12 (mem) 84 (reg)
total(clover)	2232	174 (mem) 84 (reg)	12 (mem) 96 (reg)

表 5: 方向毎の演算、データロード、データストア数。SU(3) 圧縮版。(mem) はメモリ/キャッシュからのロード。(reg) は一時変数で全体に渡って共通の領域なので register に割り当てるのが妥当なロードストア先。

Wilson hopping	11.33 flop/complex	1.417 flop/byte(S.P.)	0.7083 flop/byte(D.P.)
(1632 flop):(144 complex)	0.08824 complex/flop	0.7059 byte/flop(S.P.)	1.412 byte/flop(D.P.)
Clover hopping	12 flop/complex	1.5 flop/byte(S.P.)	0.75 flop/byte(D.P.)
(2232 flop):(186 complex)	0.08333 complex/flop	0.6667 byte/flop(S.P.)	1.333 byte/flop(D.P.)

表 6: 格子点あたりの演算とロードストアの比。SU(3) 圧縮版。

5 次元有効型のホッピング行列の演算コスト　5 方向の格子点数を N_5 とする。L を掛ける演算はカイラル射影とそれに引き続く L の掛け算。L は実数行列。ディラック表示でのカイラル射影は 2 成分スピノールに対する加減算を行ない求め、残りの 2 成分は前者の結果の符号の付け替えで求めることが出来る。n_5 一点あたり $+$ 射影の演算量は複素数の加減算 6 回 $-$ 射影の演算量も複素数の加減算 6 回で、24 flop。従って、$24*N_5$ flop。L の掛け算は実数と複素数の掛け算が N_5^2 回複素数の足し算が $N_5(N_5-1)$ 回。カラーとスピン毎 ($+$ 射影 2 成分と $-$ 射影 2 成分) にこれを行なうので 12 倍すると $(N_5^2*2+N_5(N_5-1)*2)*12=24(2N_5^2-N_5)$ flop. 結局 $24(2N_5^2-N_5)+24*N_5=48N_5^2$ flop。

データの再利用性を考慮して 5 次元有効型のホッピング行列の演算コストを見積もると表 7、8 になる。N_5 が大きくなり、レジスタに一時変数を保持し続けることが出来なくなると効率が大幅に悪化することが予想される。この場合、メモリアクセスの量はは N_5 比例分が支配的となり SU(3) 再構築の効果はあまり無い。

$N_5<27$ では主にホッピング行列を N_5 回掛ける演算が全体の計算量を支配しているが、$N_5>27$ になると密行列 L の掛け算が支配的になる。

格子カイラル対称性を制御するために k 次元のシフト射影を行なう場合は 4 次元内積をあらかじめ $k\times N_5$ 回行なう。内積のための縮約通信は個々の要素毎に行なうと遅いのでまとめて $k\times N_5$ の行列の形で縮訳する。縮訳のための通信時間は 1 回の縮訳時間のほぼ等しいとする。この内積の計算量は $8*12*k*N_5*V$ であり、ロードストア量は $2*12*k*N_5*V$(cmplx) である。これから $k\times N_5$ の係数 $c_j(s)$ を求める。ホッピング行列の掛け算時に $w_j(n)*c_j(s)$ を足し込む。この計算量は格子点当たり $8*12*k*N_5$、ロードストア量は $12*k$ である。従って格子点当たりで計算量は $2*8*12*k*N_5$ 増え、ロードストア量は $(2N_5+1)12k$ 増える。

N_5 の大きさの見積りは物理対象による。$N_5\sim 16$ – 32 で、十分格子カイラル対称性を保持できるかの検討が必要。格子カイラル対称性の破れの計量と 4 次元時空体積と N_5 の大きさの関係を有効 5 次元演算子の型毎に見積もる必要がある。Zorotalev 型の近似で N_5 の大きさはある程度低く出来る。

方向	演算数 (flop)	データロード数 (cmplx)	データストア数 (cmplx)
$t+$	$(66*2+12)*N_5$	$(6*N_5+9)$ (mem)	$12*N_5$ (reg)
$t-$	$(66*2+12)*N_5$	$(6*N_5+9)$ (mem)	$12*N_5$ (reg)
$z+$	$(66*2+36)*N_5$	$(12*N_5+9)$ (mem) $12*N_5$ (reg)	$12*N_5$ (reg)
$z-$	$(66*2+36)*N_5$	$(12*N_5+9)$ (mem) $12*N_5$ (reg)	$12*N_5$ (reg)
$y+$	$(66*2+36)*N_5$	$(12*N_5+9)$ (mem) $12*N_5$ (reg)	$12*N_5$ (reg)
$y-$	$(66*2+36)*N_5$	$(12*N_5+9)$ (mem) $12*N_5$ (reg)	$12*N_5$ (reg)
$x+$	$(66*2+36)*N_5$	$(12*N_5+9)$ (mem) $12*N_5$ (reg)	$12*N_5$ (reg)
$x-$	$(66*2+36)*N_5$	$(12*N_5+9)$ (mem) $12*N_5$ (reg)	$12*N_5$ (reg)
proj (縮約 kN_5 回)	$192kN_5$	$(2N_5+1)12k$(mem) kN_5+12N_5(reg)	$12*N_5$ (reg)
L	$48*N_5^2$	$(N_5^2)/2$ (reg) $12*N_5$ (reg)	$12*N_5$ (mem)
total	$1296N_5+48N_5^2$	$84N_5+72$ (mem) $N_5^2/2+84N_5$ (reg)	$12N_5$ (mem) $96N_5$ (reg)
total(proj)	$(1296+192k)N_5+48N_5^2$	$(84+24k)N_5+72+12k$ (mem) $N_5^2/2+(96+k)N_5$ (reg)	$12N_5$ (mem) $108N_5$ (reg)

表 7: 5 次元有効演算子の方向毎の演算、データロード、データストア数。

方向	演算数 (flop)	データロード数 (cmplx)	データストア数 (cmplx)
$t+$	$42+(66*2+12)*N_5$	$(6*N_5+9*2/3)$ (mem)	$12*N_5$ (reg)
$t-$	$42+(66*2+12)*N_5$	$(6*N_5+9*2/3)$ (mem)	$12*N_5$ (reg)
$z+$	$42+(66*2+36)*N_5$	$(12*N_5+9*2/3)$ (mem) $12*N_5$ (reg)	$12*N_5$ (reg)
$z-$	$42+(66*2+36)*N_5$	$(12*N_5+9*2/3)$ (mem) $12*N_5$ (reg)	$12*N_5$ (reg)
$y+$	$42+(66*2+36)*N_5$	$(12*N_5+9*2/3)$ (mem) $12*N_5$ (reg)	$12*N_5$ (reg)
$y-$	$42+(66*2+36)*N_5$	$(12*N_5+9*2/3)$ (mem) $12*N_5$ (reg)	$12*N_5$ (reg)
$x+$	$42+(66*2+36)*N_5$	$(12*N_5+9*2/3)$ (mem) $12*N_5$ (reg)	$12*N_5$ (reg)
$x-$	$42+(66*2+36)*N_5$	$(12*N_5+9*2/3)$ (mem) $12*N_5$ (reg)	$12*N_5$ (reg)
L	$48*N_5^2$	$(N_5^2)/2$ (reg) $12*N_5$ (reg)	$12*N_5$ (mem)
total	$1296N_5+48N_5^2+336$	$84N_5+48$ (mem) $N_5^2/2+84N_5$ (reg)	$12N_5$ (mem) $96N_5$ (reg)

表 8: 5 次元有効演算子の方向毎の演算、データロード、データストア数。SU(3) 再構築版。

バンド幅律速の時の演算効率

データロード ストアのレイテンシーを無視もしくは完全に隠蔽でき、かつ演算のレンテンシーやスループットも理想的であった場合の演算効率を以下のようにして見積もる。F: 演算数 (Flop)、S : CPU 速度 (GFlop/sec)、N:データロード ストア量 (Byte)、B:メモリバンド幅 (GByte/sec) とする。格子あたりの演算時間 T は計算時間とデータロード ストアの時間の和として表せば、

$$T = F/S + N/B, \tag{63}$$

である。ここから有効演算速度 S_{eff} は

$$S_{\text{eff}} = F/T = \frac{BF}{(B/S)F+N}, \tag{64}$$

となる。効率は

$$S_{\text{eff}}/S = \frac{(B/S)F}{(B/S)F+N}, \tag{65}$$

である。F, N は固定だから、B/S の値により効率は決まる。当然理想から外れるとこれより効率は悪くなる。

図 1 は Wilson 型のホッピング行列の行列ベクトル積の演算効率を示している。SU(3) 再構築が効率は良い。いくつかの Byte/Flop 場合の効率を表 9 に示す。効率は単精度が良く、SU(3) 再構築をしたほうが良い。マシン Byte/Flop が 0.1 での状況がメモリアクセスのみで計算を行なっている状況であるが、倍精度ではいずれも効率は 1 割未満で効率は悪い。単精度で漸く効率は 1 割に近くになる。一方データが全てキャッシュに乗っている状況に対応しそうなマシン Byte/Flop が 1 付近では、倍精度 3 割〜4 割、単精度で 4 割〜5 割の性能が予想される。

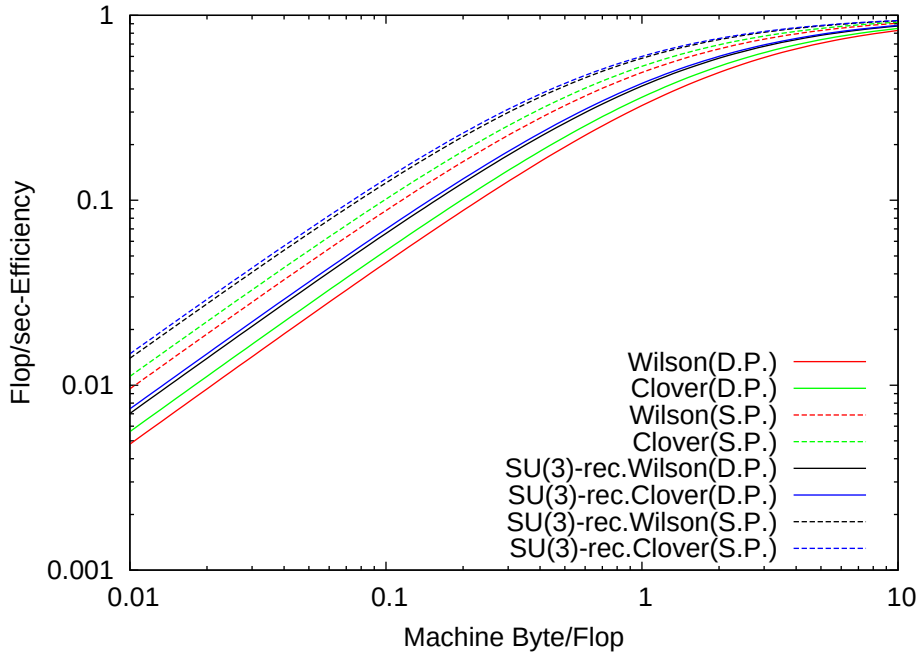

図 1: Wilson 型ホッピング行列の効率と Byte/Flop の関係

SU(3) 再構築	Machine Byte/Flop	精度	型	効率
無	0.1	D.P	Wilson	0.046
無	0.1	D.P	Clover	0.053
無	0.1	S.P	Wilson	0.088
無	0.1	S.P	Clover	0.101
無	1	D.P	Wilson	0.33
無	1	D.P	Clover	0.36
無	1	S.P	Wilson	0.49
無	1	S.P	Clover	0.53
無	2	D.P	Wilson	0.49
無	2	D.P	Clover	0.53
無	2	S.P	Wilson	0.66
無	2	S.P	Clover	0.69
有	0.1	D.P	Wilson	0.066
有	0.1	D.P	Clover	0.070
有	0.1	S.P	Wilson	0.12
有	0.1	S.P	Clover	0.13
有	1	D.P	Wilson	0.41
有	1	D.P	Clover	0.43
有	1	S.P	Wilson	0.59
有	1	S.P	Clover	0.60
有	2	D.P	Wilson	0.59
有	2	D.P	Clover	0.60
有	2	S.P	Wilson	0.74
有	2	S.P	Clover	0.75

表 9: Wilson 型の効率

図 2 は有効 5 次元型のホッピング行列の行列ベクトル積の演算効率を示している。SU(3) 再構築の効果は無い。表 10 は有効 5 次元型のホッピング行列の演算効率の代表例である。単精度の方が効率は良い。マシン Byte/Flop 0.1 での効率は、倍精度は 10%から 15%、単精度は 20%〜27% で効率は Wilson 型より良く見える。これは定数行列 L の演算がメモリバンド幅を圧迫せずに計算を繰り返すことによるものである。$N_5 > 27$ では定数行列 L の演算支配的となってくる。N_5 が大きくなると計算時間は延びていくことには変わらないことに注意。

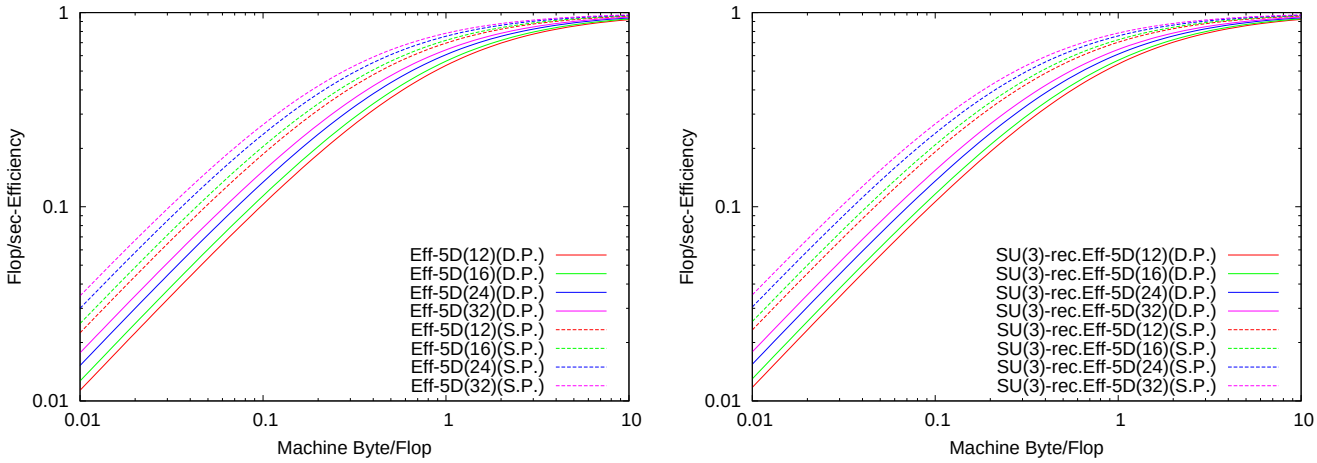

図 2: 有効 5 次元ホッピング行列の効率と Byte/Flop の関係

Machine Byte/Flop	精度	N_5	効率
0.1	D.P	12	0.103
0.1	D.P	16	0.114
0.1	D.P	24	0.134
0.1	D.P	32	0.153
0.1	S.P	12	0.187
0.1	S.P	16	0.204
0.1	S.P	24	0.236
0.1	S.P	32	0.265
1	D.P	12	0.534
1	D.P	16	0.562
1	D.P	24	0.607
1	D.P	32	0.643
1	S.P	12	0.696
1	S.P	16	0.720
1	S.P	24	0.756
1	S.P	32	0.783
2	D.P	12	0.696
2	D.P	16	0.720
2	D.P	24	0.756
2	D.P	32	0.783
2	S.P	12	0.821
2	S.P	16	0.837
2	S.P	24	0.861
2	S.P	32	0.878

表 10: 5 次元有効型の効率

SU(3) 再構築による効率向上　　Wilson 型では SU(3) 再構築で効率が向上することを見た。しかし再構築のための演算量は増えているので、計算時間は延びている可能性がある。計算時間の変化がどう

なっているかを式 (63) で見積もってみる。SU(3) 再構築しない場合の演算量、データ量、計算時間を

$$F_0, \quad N_0, \quad T_0 = F_0/S + N_0/B, \tag{66}$$

と置く。SU(3) 再構築する場合の演算量、データ量、計算時間を

$$F_r, \quad N_r, \quad T_r = F_r/S + N_r/B, \tag{67}$$

とする。SU(3) 再構築する場合としない場合の演算量と、データ量の変化 (比) を

$$a = F_r/F_0, \quad b = N_r/N_0, \tag{68}$$

と置く。この時計算時間の変化 (比) は

$$T_r/T_0 = \frac{a\frac{B}{S} + b\frac{N_0}{F_0}}{\frac{B}{S} + \frac{N_0}{F_0}}, \tag{69}$$

となる。Wilson の場合

$$a = 1632/1296 = 1.26, \quad b = 144/168 = 0.857, \quad N_0/F_0 = 168/1296 * z = 0.1296 * z, \tag{70}$$

(z は complex 変数のバイト数)。Clover の場合

$$a = 2232/1890 = 1.18, \quad b = 186/210 = 0.886, \quad N_0/F_0 = 210/1890 * z = 0.1111 * z, \tag{71}$$

である。

図 3 に式 (69) をプロットした。マシン Byte/Flop が低い領域 0.1 以下ではいずれも 10%ほどの時間削減が見込める。これはメモリバンド幅が律速時の削減になる。一方マシン Byte/Flop が 0.5 を越えると単精度ではかえって時間が延びる。Byte/Flop が 1 を越えると倍精度ではかえって時間が延びる。ワーキングセットが全てキャッシュに乗り、高い Byte/Flop が出ているときは SU(3) 再構築はかえって計算時間を伸ばす結果となる。一方 SU(3) 再構築はワーキングセットを減らす効果もあるので、キャッシュにのせて高い Byte/Flop による計算時間削減効果と SU(3) 再構築による計算時間増加効果とのトレードオフを考える必要がある。

メモリ律速で計算を進めるのか、ワーキングセット全てキャッシュに乗せて計算を進めるのかの方針をはっきりさせないといけない。

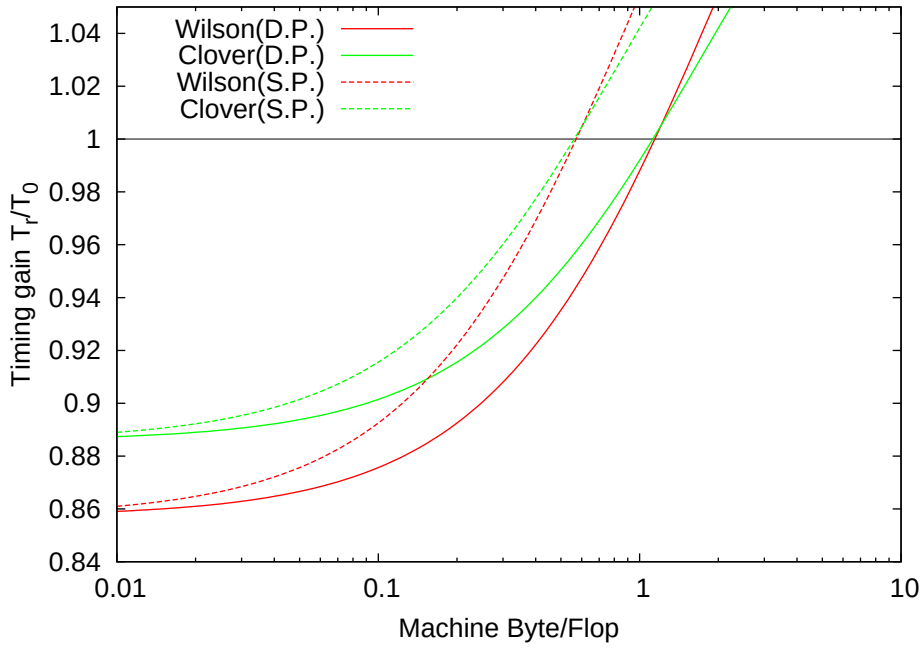

図 3: ウィルソン型の SU(3) 再構築による時間削減割合。

精度の問題　メモリバンド幅の向上が見られない近年、反復解法での倍精度の結果を得るために倍精度計算に単精度計算を組み合わせてみかけのメモリバンド幅を向上させ計算速度を上げる方法が取られてきている。基本的には良く知られたリチャードソン反復補正を利用している。単精度で連立方程式を解き、その結果の倍精度での残差を計算し再び、その残差に対して単精度で連立方程式を解き、補正ベクトルを計算し倍精度の結果を改良し、再び倍精度の残差を計算し……を倍精度の残差ベクトルが収束するまでこの過程を繰り返す方法である。この方法ではほとんどの計算が単精度での反復に費される。

単精度での連立方程式が元の倍精度での連立方程式を良く表現 (近似) できていないと、この方法の収束が遅くなると考えられる。単精度での連立方程式が元の倍精度での連立方程式を良く表現 (近似) できているかは、まずは係数行列の倍精度で表した固有値の範囲が単精度で表現できる範囲に収まっているかで、まずは評価できる。

係数行列の最大固有値と最小固有値の比を条件数というが、条件数はすなわち固有値の分布範囲、もしくは数値での桁数を意味する。倍精度変数の有効桁数は 10 進数表現で約 16 桁であるので 10 進数表現で約 16 桁程度の散らばりに入る数値は同時に扱ってもよかろう。単精度変数の有効桁数は 10 進数表現で約 7 桁であるので 10 進数表現で約 7 桁程度の散らばりに入る数値は同時に扱ってもよかろう。条件数で言うと 7 桁以下の条件数であれば係数行列の性質をあるていど近似できていると考えられる。

格子 QCD でのクォークの係数行列の条件数 K を考えてみる。クォーク係数行列の最低固有値はクォーク質量を m_q、格子間隔を a とすれば、am_q 程度、最大固有値は $O(1)$ である (Free wilson で最大値は $8 + am_q$)。条件数は $1/am_q$ である。クォーク質量は軽い u, d に対して $m_q \sim 2$ MeV とし格子間隔逆数を $1/a \sim 2$ GeV-4 GeV とすると、

$$K \sim 1/am_q \sim 1 \times 10^3 \sim 2 \times 10^3, \tag{72}$$

である。およそ 3 桁-4 桁の範囲であり、7 桁の範囲をとらえられる単精度で十分表現できる。ウィルソン型のクォーク反復解法では既に単精度をもちいた加速が有効であることは実証されている。(ただし Intel SSE の単精度計算が倍精度の 2 倍速いことも効いている。)

反復改良で残差が倍精度の 10^{-15} まで収束するには単精度による収束を 10^{-6} をおよそ 3 回繰り返す必要がある。全体の反復回数は全てを倍精度で行なった場合と、反復改良を行なった場合とで、変化することが知られている。$N_{\mathrm{D.P.}}$ を倍精度で行なった場合の収束までの反復回数 $N_{\mathrm{M.P.}}$ を反復改良での収束までの反復回数とし、

$$N_{\mathrm{M.P.}} = N_{\mathrm{D.P.}}\alpha, \tag{73}$$

と表すことにする。経験的に $\alpha \gtrsim 1$ であり、$\alpha > 2$ になることは無い様である。

単精度演算性能が倍精度演算性能より 2 倍速い場合これはとても効果的である。計算時間の比を以下のように見積もる。反復改良を用いた場合の収束までの計算時間を $T_{\mathrm{M.P.}}$ とし倍精度のみでの収束までの計算時間を $T_{\mathrm{D.P.}}$ とする。計算時間は行列ベクトル積の回数に比例すると近似すれば、

$$T_{\mathrm{M.P.}} = cT_{\mathrm{matvecS.P.}}N_{\mathrm{M.P.}} = cT_{\mathrm{S.P.}}N_{\mathrm{D.P.}}\alpha, \tag{74}$$

$$T_{\mathrm{D.P.}} = cT_{\mathrm{matvecD.P.}}N_{\mathrm{D.P.}}, \tag{75}$$

ここで c は比例係数で共通とした。比は、

$$\frac{T_{\mathrm{M.P.}}}{T_{\mathrm{D.P.}}} = \frac{T_{\mathrm{matvecS.P.}}}{T_{\mathrm{matvecD.P.}}}\alpha, \tag{76}$$

となる。一回あたりの行列ベクトル積の時間を T_{matvec} とした。行列ベクトル積の時間はホッピング行列の積で支配されているので、演算数を F 必要データ量を $N_{\mathrm{D.P.}}, N_{\mathrm{S.P.}}$、演算速度を $S_{\mathrm{D.P.}}, S_{\mathrm{S.P.}}$ バンド幅を B とすると、倍精度、単精度それぞれ

$$T_{\mathrm{matvecD.P.}} = \frac{F}{S_{\mathrm{D.P.}}} + \frac{N_{\mathrm{D.P.}}}{B}, \tag{77}$$

$$T_{\mathrm{matvecS.P.}} = \frac{F}{S_{\mathrm{S.P.}}} + \frac{N_{\mathrm{S.P.}}}{B}, \tag{78}$$

である。これを用いれば

$$\begin{aligned}\frac{T_{\rm M.P.}}{T_{\rm D.P.}} &= \frac{B/S_{\rm S.P.}+N_{\rm S.P.}/F}{B/S_{\rm D.P.}+N_{\rm D.P.}/F}\alpha,\\ &= \frac{B/S_{\rm D.P.}(S_{\rm D.P.}/S_{\rm S.P.})+N_{\rm D.P.}/F/2}{B/S_{\rm D.P.}+N_{\rm D.P.}/F}\alpha,\\ &= \frac{B/S_{\rm D.P.}(2S_{\rm D.P.}/S_{\rm S.P.})+N_{\rm D.P.}/F}{B/S_{\rm D.P.}+N_{\rm D.P.}/F}\frac{\alpha}{2},\end{aligned} \tag{79}$$

となる。単精度が倍精度の 2 倍の性能であれば $(2S_{\rm D.P.}/S_{\rm S.P.})=1$ なので

$$\frac{T_{\rm M.P.}}{T_{\rm D.P.}}=\frac{\alpha}{2}, \tag{80}$$

である。$\alpha<2$ なので必ず速くなると言える。単精度が倍精度と同じ性能であれば $(2S_{\rm D.P.}/S_{\rm S.P.})=2$ なので

$$\frac{T_{\rm M.P.}}{T_{\rm D.P.}}=\frac{2B/S_{\rm D.P.}+N_{\rm D.P.}/F}{B/S_{\rm D.P.}+N_{\rm D.P.}/F}\frac{\alpha}{2}, \tag{81}$$

となる。ウィルソン型ではおよそ $N_{\rm D.P.}/F\sim 2$ であり、この時、$B/S_{\rm D.P.}\ll 1$ では $T_{\rm M.P.}/T_{\rm D.P.}\sim\alpha/2$、となる。$B/S_{\rm D.P.}\sim 1$ では $T_{\rm M.P.}/T_{\rm D.P.}\sim 3\alpha/2$、$B/S_{\rm D.P.}\gg 1$ では $T_{\rm M.P.}/T_{\rm D.P.}\sim\alpha$ となる。まとめる、ウィルソン型では $B/S_{\rm D.P.}<1$ で $\alpha<3/2$ であれば速くなると予想される。

5 次元有効型での単精度加速の有効性について議論するためには、5 次元有効係数行列の固有値分布の性質を見る必要がある。5 次元有効係数行列は、符号関数に対するどの近似を用いても負の実部を持つ固有値を避けることが出来ない様である。連立方程式を反復解法を用いて解くときは正規化して解くために、係数行列は 2 乗 ($A=D_{5d}D_{5d}^{\dagger}$ もしくは $A=D_{5d}^{\dagger}D_{5d}$) され、エルミート行列となる。2 乗された係数行列 A の条件数は元の行列の条件数の 2 乗となる。元の D_{5d} の最低固有値は am_q ではなく、符号関数の近似手法に応じた解析が必要。

5 ノード間通信

並列計算時の通信について吟味する。計算要素当たりの格子体積を $V_{\rm node=}M_xM_yM_zM_t=M_x^3N_t$ とする。係数行列は 1 回差分を含むので計算要素の保持していないデータへのアクセスのために通信が必要。

4 次元格子は 4 次元トーラス (周期境界) とする。係数行列は 4 次元の 1 回差分をノード構成も 4 次元トーラス (周期境界) とする。並列化を 4 次元で行なう場合，3 次元で行なう場合 2 次元で行なう場合と分けて考える。

表面部分の格子点数は空間方向が $M_x^2M_t$、時間方向が M_x^3 である。通信に必要なデータ量はこれをウィルソン型は 12 倍した物、5 次元有効型は $12N_5$ 倍した物となる。このデータをノード間で交換する。非同期の通信を使うと仮定する。データ転送の直前に一度同期を取る必要がある。

通信に要する時間はレイテンシーとバンド幅で決まる。MPI 通信バッファーへのデータの詰め込み展開時間も考える。同期のための時間を $T_{\rm sync}$ バイセクションバンド幅を B_B Byte/sec 、メモリバンド幅を B、通信レイテンシを T_l sec とする。送信時にメモリ上にある表面データを一度通信バッファに詰め込むため memory → CPU → memory のデータ移動を伴う。受信時に受け取ったデータを再び展開し外表面データに置くために memory → CPU → memory のデータ移動を伴う。

3 次元空間分割並列　データ交換に必要な時間 $T_{\rm comm}$ は

$$T_{\rm comm}=\frac{1}{B_B}\left(T_lB_B+12*z\frac{3}{M_x}\left(1+2\frac{B_B}{B}\right)V_{\rm node}\right), \tag{82}$$

と表せる。ここで z はウイルソン型で $z=16$(倍精度), 8 (単精度)Byte、5 次元有効型で $z=N_5*16$(倍精度), N_5*8 (単精度)Byte。

4 次元空間分割並列　データ交換に必要な時間 $T_{\rm comm}$ は

$$T_{\rm comm} = \frac{1}{B_B}\left(T_l B_B + 12 * z\left(\frac{1}{M_t}+\frac{3}{M_x}\right)\left(1+2\frac{B_B}{B}\right)V_{\rm node}\right), \tag{83}$$

と表せる。ここで z はウイルソン型で $z = 16$(倍精度), 8 (単精度)Byte、5 次元有効型で $z = N_5 * 16$(倍精度), $N_5 * 8$ (単精度)Byte。

通信と演算の重ね合わせ　通信と演算を重ね合わせることを考える。演算を内点のデータのみ使い計算する部分と、外点を使う部分に分ける。

分けた場合の演算量とロード ストア数をウィルソン型は表 11,13 に記す。5 次元有効型は表 15,16 に記す。

疎行列ベクトル積の時間 $T_{\rm matvec}$ は、内点計算の時間を $T_{\rm in}$、外点計算の時間を $T_{\rm out}$、同期時間を $T_{\rm sync}$、通信時間を $T_{\rm comm}$ とすれば、

$$T_{\rm matvec} = T_{\rm sync} + \max(T_{\rm in}, T_{\rm comm}) + T_{\rm out}, \tag{84}$$

である。ただし通信のためのデータの詰め込み展開も計算と同時にできるとした。この仮定が成り立たない場合はデータの詰め込み展開の時間を内点計算の前後に加える必要がある。

通信の重ね合わせや通信がないときの元の計算時間を $T_{\rm org}$ とすると、通信による影響は

$$\begin{aligned} T_{\rm matvec} &= T_{\rm org} * \beta, & (85)\\ \beta &= \frac{T_{\rm matvec}}{T_{\rm org}}, \\ &= \frac{\max(T_{\rm in}, T_{\rm comm}) + T_{\rm out}}{T_{\rm org}}, & (86) \end{aligned}$$

とパラメトライズ出来る。

	演算数 (flop)	ロード ストア数 (cmplx)
時間方向	$144 * 2M_x^3 M_t$	$15 * 2M_x^3 M_t$
空間方向 (内点)	$3 * 168 * 2(M_x - 1)M_x^2 M_t$	$3 * 21 * 2(M_x - 1)M_x^2 M_t (+12M_x^2 M_t)$
clover(内点)	$600M_x^3 M_t$	$21 * 2M_x^3 M_t + 12M_x^3 M_t$
total(wilson)	$(1296 - 1008/M_x)V_{\rm node}$	$(168 - 126/M_x)V_{\rm node}$
total(clover)	$(1896 - 1008/M_x)V_{\rm node}$	$(210 - 126/M_x)V_{\rm node}$
空間方向 (外点)	$3 * 168 * 2M_x^2 M_t$	$3 * 21 * 2M_x^2 M_t + 3 * 12 * 2M_x^2 M_t$
clover(外点)	$(600 + 24)(M_x^3 - (M_x - 2)^3)M_t$	$(21 * 2 + 2 * 12)(M_x^3 - (M_x - 2)^3)M_t$
total(wilson)	$(1008/M_x)V_{\rm node}$	$((126 + 72)/M_x)V_{\rm node}$
total(clover)	$(1008/M_x + 624(1 - (1 - 2/M_x)^3))V_{\rm node}$	$(198/M_x + 66(1 - (1 - 2/M_x)^3))V_{\rm node}$

表 11: ウィルソン型、3 次元空間分割の場合の計算内訳 (外点の計算量とロード ストア数が正しいか自信無)

	演算数 (flop)	ロード ストア数 (cmplx)
時間方向	$186 * 2M_x^3M_t$	$12 * 2M_x^3M_t$
空間方向 (内点)	$3 * 210 * 2(M_x - 1)M_x^2M_t$	$3 * 18 * 2(M_x - 1)M_x^2M_t(+12M_x^2M_t)$
clover(内点)	$600M_x^3M_t$	$21 * 2M_x^3M_t + 12M_x^3M_t$
total(wilson)	$(1632 - 1260/M_x)V_{\rm node}$	$(144 - 108/M_x)V_{\rm node}$
total(clover)	$(2232 - 1260/M_x)V_{\rm node}$	$(186 - 108/M_x)V_{\rm node}$
空間方向 (外点)	$3 * 210 * 2M_x^2M_t$	$3 * 18 * 2M_x^2M_t + 3 * 12 * 2M_x^2M_t$
clover(外点)	$(600 + 24)(M_x^3 - (M_x - 2)^3)M_t$	$(21 * 2 + 2 * 12)(M_x^3 - (M_x - 2)^3)M_t$
total(wilson)	$(1260/M_x)V_{\rm node}$	$((108 + 72)/M_x)V_{\rm node}$
total(clover)	$(1260/M_x + 624(1 - (1 - 2/M_x)^3))V_{\rm node}$	$(180/M_x + 66(1 - (1 - 2/M_x)^3))V_{\rm node}$

表 12: ウィルソン型 SU(3) 再構築版、3 次元空間分割の場合の計算内訳 (外点の計算量とロード ストア数が正しいか自信無)

	演算数 (flop)	ロード ストア数 (cmplx)
時間方向 (内点)	$144 * 2M_x^3(M_t - 1)$	$15 * 2M_x^3(M_t - 1)$
空間方向 (内点)	$3 * 168 * 2(M_x - 1)M_x^2M_t$	$3 * 21 * 2(M_x - 1)M_x^2M_t(+12M_x^2M_t)$
clover(内点)	$600M_x^3M_t$	$21 * 2M_x^3M_t + 12M_x^3M_t$
total(wilson)	$(1296 - 288/M_t - 1008/M_x)V_{\rm node}$	$(168 - 30/M_t - 126/M_x)V_{\rm node}$
total(clover)	$(1896 - 288/M_t - 1008/M_x)V_{\rm node}$	$(210 - 30/M_t - 126/M_x)V_{\rm node}$
時間方向 (外点)	$144 * 2M_x^3$	$15 * 2M_x^3 + 12 * 2M_x^3$
空間方向 (外点)	$3 * 168 * 2M_x^2M_t$	$3 * 21 * 2M_x^2M_t + 3 * 12 * 2M_x^2M_t$
clover(外点)	$(600 + 24)(M_x^3M_t - (M_x - 2)^3(M_t - 2))$	$(21 * 2 + 2 * 12)(M_x^3M_t - (M_x - 2)^3(M_t - 2))$
total(wilson)	$(288/M_t + 1008/M_x)V_{\rm node}$	$(54/M_t + 198/M_x)V_{\rm node}$
total(clover)	$(288/M_t + 1008/M_x + 624(1 - (1 - 2/M_x)^3)(1 - 2/M_t))V_{\rm node}$	$(54/M_t + 198/M_x + 66(1 - (1 - 2/M_x)^3(1 - 2/M_t))V_{\rm node}$

表 13: ウィルソン型、4 次元空間分割の場合の計算内訳 (外点の計算量とロード ストア数が正しいか自信無)

	演算数 (flop)	ロード ストア数 (cmplx)
時間方向 (内点)	$186 * 2M_x^3(M_t - 1)$	$12 * 2M_x^3(M_t - 1)$
空間方向 (内点)	$3 * 210 * 2(M_x - 1)M_x^2M_t$	$3 * 18 * 2(M_x - 1)M_x^2M_t(+12M_x^2M_t)$
clover(内点)	$600M_x^3M_t$	$21 * 2M_x^3M_t + 12M_x^3M_t$
total(wilson)	$(1632 - 372/M_t - 1260/M_x)V_{\rm node}$	$(144 - 24/M_t - 108/M_x)V_{\rm node}$
total(clover)	$(2232 - 372/M_t - 1260/M_x)V_{\rm node}$	$(186 - 24/M_t - 108/M_x)V_{\rm node}$
時間方向 (外点)	$186 * 2M_x^3$	$12 * 2M_x^3 + 12 * 2M_x^3$
空間方向 (外点)	$3 * 210 * 2M_x^2M_t$	$3 * 18 * 2M_x^2M_t + 3 * 12 * 2M_x^2M_t$
clover(外点)	$(600 + 24)(M_x^3M_t - (M_x - 2)^3(M_t - 2))$	$(21 * 2 + 2 * 12)(M_x^3M_t - (M_x - 2)^3(M_t - 2))$
total(wilson)	$(372/M_t + 1260/M_x)V_{\rm node}$	$(48/M_t + 180/M_x)V_{\rm node}$
total(clover)	$(372/M_t + 1260/M_x + 624(1 - (1 - 2/M_x)^3)(1 - 2/M_t))V_{\rm node}$	$(48/M_t + 180/M_x + 66(1 - (1 - 2/M_x)^3(1 - 2/M_t))V_{\rm node}$

表 14: ウィルソン型 SU(3) 再構築版、4 次元空間分割の場合の計算内訳 (外点の計算量とロード ストア数が正しいか自信無)

	演算数 (flop)	ロード ストア数 (cmplx)
時間方向	$144 * 2N_5M_x^3M_t$	$(6N_5 + 9)2M_x^3M_t$
空間方向 (内点)	$3 * 168 * 2(M_x - 1)N_5M_x^2M_t$	$3(12N_5 + 9)2(M_x - 1)M_x^2M_t$
proj	$192kN_5M_x^3M_t$	$(2N_5 + 1)12kM_x^3M_t$
L(内点)	$48N_5^2M_x^3M_t$	$12N_5M_x^3M_t$
total(5deff)	$(1296 - 1008/M_x + 48N_5)N_5V_{\rm node}$	$(96N_5 + 72 - (72N_5 + 54)/M_x)V_{\rm node}$
total(5deff-proj)	$(1296 - 1008/M_x + 48N_5 + 192k)N_5V_{\rm node}$	$(96N_5 + 72 - (72N_5 + 54)/M_x + 12k(2N_5 + 1))V_{\rm node}$
空間方向 (外点)	$3 * 168 * 2N_5M_x^2M_t$	$3(12N_5 + 9)2M_x^2M_t + 3 * 12 * 2N_5M_x^2M_t$
L(外点)	$48N_5^2(M_x^3 - (M_x - 2)^3)M_t$	$12 * 2N_5(M_x^3 - (M_x - 2)^3)M_t$
total(5deff)	$(1008/M_x + 48(1 - (1 - 2/M_x)^3)N_5)N_5V_{\rm node}$	$((144N_5 + 54)/M_x + 24N_5(1 - (1 - 2/M_x)^3))V_{\rm node}$

表 15: 5 次元有効型、3 次元空間分割の場合の計算内訳 (外点の計算量とロード ストア数が正しいか自信無)

	演算数 (flop)	ロード ストア数 (cmplx)
時間方向 (内点)	$144 * 2N_5M_x^3(M_t-1)$	$(6N_5+9)2M_x^3(M_t-1)$
空間方向 (内点)	$3 * 168 * 2(M_x-1)N_5M_x^2M_t$	$3(12N_5+9)2(M_x-1)M_x^2M_t$
proj	$192kN_5M_x^3M_t$	$(2N_5+1)12kM_x^3M_t$
L(内点)	$48N_5^2M_x^3M_t$	$12N_5M_x^3M_t$
total(5deff)	$(1296-288/M_t-1008/M_x+48N_5)N_5V_{\rm node}$	$(96N_5+72-(12N_5+18)/M_t-(72N_5+54)/M_x)V_{\rm node}$
total(5deff-proj)	$(1296-288/M_t-1008/M_x+48N_5+192k)N_5V_{\rm node}$	$(96N_5+72-(12N_5+18)/M_t-(72N_5+54)/M_x+12k(2N_5+1))V_{\rm node}$
時間方向 (外点)	$144 * 2N_5M_x^3$	$(6N_5+9)2M_x^3+12*2N_5M_x^3$
空間方向 (外点)	$3 * 168 * 2N_5M_x^2M_t$	$3(12N_5+9)2M_x^2M_t+3*12*2N_5M_x^2M_t$
L(外点)	$48N_5^2(M_x^3M_t-(M_x-2)^3(M_t-2))$	$12*2N_5(M_x^3M_t-(M_x-2)^3(M_t-2))$
total(5deff)	$(288/M_t+1008/M_x+48(1-(1-2/M_x)^3(1-2/M_t))N_5)N_5V_{\rm node}$	$((78N_5+18)/M_t+(144N_5+54)/M_x+24N_5(1-(1-2/M_x)^3(1-2/M_t)))V_{\rm node}$

表 16: 5 次元有効型、4 次元空間分割の場合の計算内訳 (外点の計算量とロード ストア数が正しいか自信無)

6 反復ソルバーに必要なデータ量、計算時間、計算速度の見積り

これまで格子点 1 点あたりの計算機性能とホッピング行列の演算時間について評価してきた。つぎに、計算機の演算要素 (ノードとかコアとか CPU とかマシン全体) に複数格子点を割り当てたときのデータ量、計算時間、計算速度の見積りを行なう。4 次元格子体積を単純に $V=N^4$ とする。係数行列に対して前処理を行なわない場合の評価をおこなう。通信を無視して固定反復、BiCGStab、CG 法で解く場合をそれぞれ見積もる。

	データ量 (cmplx)	(S.P.) Byte	(D.P.) Byte
リンク変数	$3\times3\times4V=36V$	$288V$	$576V$
リンク変数 (SU(3) 圧縮)	$3\times2\times4V=24V$	$192V$	$384V$
フェルミオン変数	$3\times4V=12V$	$96V$	$192V$
クローバー項	$21\times2V=42V$	$336V$	$672V$
5 次元質量行列 L	$N_5^2/2$	$4N_5^2$	$8N_5^2$

表 17: V のデータ量

	演算量 $/V$	ロード ストア (cmplx)$/V$
Wilson $D = 1 - \kappa M$	$48 + 1296\beta$ $[48 + 1632\beta]$	$168\beta + 12$ $[144\beta + 12]$
Clover $D = 1 - \kappa F^{-1} M$	$48 + 1896\beta$ $[48 + 2232\beta]$	$210\beta + 12$ $[186\beta + 12]$
5D-eff. $A = D_{5d}^{\dagger} D_{5d}$	$(48N_5 + (1296N_5 + 48N_5^2)\beta) * 2$	$((96N_5 + 72)\beta + 12N_5) * 2$
5D-eff.(shift proj)(縮約 $2kN_5$ 回=1 回とする)	$(48N_5 + (1296N_5 + 48N_5^2 + 192kN_5)\beta) * 2$	$(((96 + 24k)N_5 + 72 + 12k)\beta + 12N_5) * 2$

表 18: 疎行列ベクトル積の演算量とロード ストア量。括弧内は SU(3) 圧縮を行なった場合。β は通信によるファクター。

ソルバー	作業ベクトルの数	行列ベクトル積の数	ベクトル和の数 $v = v + \alpha a$	内積の数	ベクトルの長さの数
固定反復	3	1	2 ($\alpha = 1$)	0	(1)
CG	5	1	3 (α:real)	1	1
BiCGStab	7	2	6 (α:cmplx)	3	1+(2)

表 19: $Ax = b$ ソルバーの作業ベクトルの数、線形代数計算数（1 反復あたり。反復前準備計算部分は省いた)。ベクトルの長さの計算で括弧内の数字は残差計算のための計算で反復回数固定時は残差計算を省くことが出来るもの。

演算の種類	演算量	ロード数 (cmplx)	ストア数 (cmplx)	(cmplx/flop)
$v = v + a$	2	2	1	3/2
$v = v + \alpha a$ (α: real)	4	2	1	3/4
$v = v + \alpha a$ (α: cmplx)	8	2	1	3/8
$\langle p \vert q \rangle$	8	2	0	2/8
$\vert r \vert^2$	4	1	0	1/4

表 20: 線形代数計算の演算量、データロード ストア数見積り (要素当)。スカラ量のロード ストアは無視した。演算数ロード ストア数の格子点当たりのウィルソン型へ換算は 12 倍、5 次元有効型への換算は $12 * N_5$ 倍。

反復当たりの演算量　表 21–23 に各種ソルバーの基礎データを示した。

1 反復当たりの演算時間 $T_{\rm iter}$ は

$$T_{\rm iter} = \frac{F}{S} + \frac{N}{B}, \tag{87}$$

である。F は演算数 (Flop)、N はロード ストア量 (Byte)、B はマシンバンド幅 (Byte/sec)、S はマシン演算速度 (Flop/sec) である。収束までの反復回数は CG 法については公式が知られている。

$$\frac{||x_* - x_m||_2}{||x_* - x_0||_2} \leq 2\sqrt{K} \left[\frac{\sqrt{K} - 1}{\sqrt{K} + 1} \right]^m, \tag{88}$$

ここで、A が係数行列、K は A の条件数、x_m は反復回数 m での近似解、x_* は真の解、$||\dots||_2$ は 2–ノルム。反復回数を以下のように仮定する。

- 倍精度

- ウィルソン型: およそ 20000 反復 (CG 条件数 $K \sim (1/am_q)^2 \sim 10^6$)。(BiCGStab だと半分 10000 反復)
- 5 次元有効型: およそ 1000〜10000 反復 と見積もる?

- 単精度
  - ウィルソン型: およそ 10000 反復。(BiCGStab だと半分 5000 反復) 反復改良版は 20000α
  - 5 次元有効型: およそ 500〜5000 反復 と見積もる? 反復改良版は $(100〜10000)\alpha$.

	メモリ (cmplx)	演算数 (Flop)	ロード ストア (cmplx)	縮約回数
Wilson	$72V$	$(144+1296\beta)V$	$(96+168\beta)V$	1
Clover	$114V$	$(144+1896\beta)V$	$(96+210\beta)V$	1
Wilson(SU(3)rec.)	$60V$	$(144+1632\beta)V$	$(96+144\beta)V$	1
Clover(SU(3)rec.)	$102V$	$(144+2232\beta)V$	$(96+186\beta)V$	1
5D-eff.	$12(4N_5+3)V$	$96N_5(2+(27+N_5)\beta)V$	$12(9N_5+4(3+4N_5)\beta)V$	1
5D-eff.(shift-proj)	$12(4N_5+3+2k)V$	$96N_5(2+(27+N_5+4k)\beta)V$	$12(9N_5+2(6+8N_5+k+2kN_5)\beta)V$	$1+[kN_5]$

表 21: 固定反復ソルバーの 1 反復あたりの演算数ロード ストア数メモリ容量。5D-eff. では正規化のための中間作業ベクトルが 1 本追加されている。

	メモリ (cmplx)	演算数 (Flop)	ロード ストア (cmplx)	縮約回数
Wilson	$108V$	$(384+2592\beta)V$	$(168+336\beta)V$	2
Clover	$150V$	$(384+3792\beta)V$	$(168+420\beta)V$	2
Wilson(SU(3)rec.)	$96V$	$(384+3264\beta)V$	$(168+288\beta)V$	2
Clover(SU(3)rec.)	$138V$	$(384+4464\beta)V$	$(168+372\beta)V$	2
5D-eff.	$12(6N_5+3)V$	$96N_5(4+(27+N_5)\beta)V$	$24(7N_5+(6+8N_5)\beta)V$	2
5D-eff.(shift-proj)	$12(6N_5+3+2k)V$	$96N_5(4+(27+N_5+4k)\beta)V$	$24(7N_5+(6+8N_5+k+2kN_5)\beta)V$	$2+[kN_5]$

表 22: CG ソルバーの 1 反復あたりの演算数ロード ストア数メモリ容量。ウィルソン型、5D-eff 型共に CG で解ける様に正規化しており、Matvec を 2 倍、作業用中間ベクトルを一本追加している。

	メモリ (cmplx)	演算数 (Flop)	ロード ストア (cmplx)	縮約回数
Wilson	$120V$	$(1104+2592\beta)V$	$(348+336\beta)V$	6
Clover	$162V$	$(1104+3792\beta)V$	$(348+420\beta)V$	6
Wilson(SU(3)rec.)	$108V$	$(1104+3264\beta)V$	$(348+288\beta)V$	6
Clover(SU(3)rec.)	$150V$	$(1104+4464\beta)V$	$(348+372\beta)V$	6

表 23: BiCGStab ソルバーの 1 反復あたりの演算数ロード ストア数メモリ容量。5D-eff は解けないので省略した。

計算時間の見積り　以上の表から計算時間を見積もる。以下の条件を用いる。

- 必要なパラメータ (計算機):
  - 全体計算速度 S [Flop/sec] ($S_{\mathrm{S.P.}}$:単精度, $S_{\mathrm{D.P.}}$:倍精度)

- 全体メモリバンド幅 B [Byte/sec]
- 全体キャッシュバンド幅 $B_{\rm cache}$ [Byte/sec]
- ノードの 1 体 1 ノード間通信バンド幅 B_B [Byte/sec]
- 全体同期時間 $T_{\rm sync}$ [sec]
- 全体縮約時間 $T_{\rm red} = T_{\rm sync}\gamma$ [sec]
- 1 体 1 ノード間通信遅延時間 T_l [sec]
- 通信による疎行列ベクトル積遅延ファクター β

- 必要なパラメータ (格子、物理):
  - 全体格子点数 $V = N_x^3 N_t$
  - ノード当たり格子点数 $V_{\rm node} = M_x^3 M_t$
  - 倍精度での反復法収束までの反復回数 $N_{\rm iterD.P.}$
  - 反復改良での反復法収束までの単精度反復回数 $N_{\rm iterM.P.} = \alpha N_{\rm iterD.P.}$

- 使用メモリ容量をノード数でわった後のノード当たりの使用メモリ容量がキャッシュ容量に十分収まるのであればメモリバンド幅はキャッシュバンド幅で置き換える。

- 反復改良ではほとんどの計算時間は単精度ソルバーなので単精度ソルバーを使ったことにする。反復回数は倍精度での反復回数の α 倍とする。

- 縮約計算の時間を $T_{\rm red}$ とする。同期時間 $T_{\rm sync}$ に比例すると思われるので $T_{\rm red} = T_{\rm sync}\gamma$ とする。

計算したい物理量を求めるために必要な時間は以下の手順で求める。

1. 総計算量を計算時間で割った有効計算速度を求める。
2. 最後まで計算したい物理対象の総計算量のうちクォークソルバーに要した部分の割合とクォークソルバーの総計算量を出す。
3. クォークソルバーの総計算量を有効計算速度で割ると総計算時間が分かる。
4. クォークソルバーに要する時間から物理対象を求めるまでの総計算時間がわかる。

7 時間の見積り例

ターゲット格子サイズを 128^4 とする。計算機の構成としていくつかの例を使ってソルバーの計算時間を計算してみた。

同期時間 $T_{\rm sync}$ と縮約時間 $T_{\rm redu}$ は、ノードノード間通信のレイテンシーを T_l、ノード数を $N_{\rm node}$ として、

$$T_{\rm sync} = 2\log_2(N_{\rm node}) \times T_l, \tag{89}$$
$$T_{\rm redu} = 2T_{\rm sync}, \tag{90}$$

とした。

構成	$V_{\rm node}$	N_5	k
P1	4^4	32	24
P2	8^4	32	24
P3	16^4	32	24
P4	$16^3 \times 32$	32	24

表 24: ノード当たりの格子サイズ、N_5, k

構成	cache [Byte],[Byte/sec]	CPU speed D.P.,S.P.[Flop/sec]	mem. bw [byte/sec]	comm. bw [byte/sec]	comm. latency [sec]
A	4 M, 64 G	64 G, 128 G	64 G	10 G	5 μ
B	8 M, 128 G	128 G, 128 G	64 G	5 G	5 μ
C	16 M, 256 G	256 G, 512 G	128 G	5 G	5 μ
D	128K, 128 G	512 G, 1024 G	128 G	5 G	5 μ
E	128K, 512 G	1024 G, 2048 G	256 G	5 G	5 μ

表 25: 計算機構成例 (ノード当たり)。ノードに積めるメモリ量は制限していない。

構成	型	#node	Time/iter [sec]	Mult [%]	Reduction [%]	Niter	Time [sec]
(P1,A)	clover	1048576	$2.8810^{-3}(2.8410^{-3})$	15.9(15.1)	83.2(84.4)	10000	28.8 (28.4)
	clover(su3rec)	1048576	$2.8810^{-3}(2.8410^{-3})$	15.9(15.1)	83.2(84.4)	10000	28.8 (28.4)
	5deff	1048576	$3.3210^{-3}(2.2610^{-3})$	64.1(55.9)	24.1(35.4)	10000	33.2 (22.6)
	5deff(proj)	1048576	$8.2410^{-3}(5.2710^{-3})$	88.7(88.3)	9.7(15.2)	10000	82.4 (52.7)
(P1,B)	clover						
	clover(su3rec)						
	5deff						
	5deff(proj)						
(P1,C)	clover						
	clover(su3rec)						
	5deff						
	5deff(proj)						
(P1,D)	clover						
	clover(su3rec)						
	5deff						
	5deff(proj)						
(P1,E)	clover						
	clover(su3rec)						
	5deff						
	5deff(proj)						

表 26: P1 構成での見積り

Algorithm 1 固定反復

input x,b, Solve $Ax = b$.
$\sigma = |b|$
while $|r|/\sigma > \epsilon$ **do**
    $v = Ax$
    $r = b - v$
    $x = x + r$
end while

Algorithm 2 BiCGStab

input x,b, Solve $Ax = b$.
$q = Ax$
$r = b - Ax$
$\tilde{r} = r$ ($\tilde{r}$ can be arbitrary.)
$p = r$
$\sigma = |b|$
$\rho_0 = \langle \tilde{r} | r \rangle$
for do
    $q = Ap$
    $\alpha = \rho_0 / \langle \tilde{r} | q \rangle$
    $x = x + \alpha p$
    $r = r - \alpha q$
    if $|r|/\sigma < \epsilon$ **then**
        exit
    end if
    $t = Ar$
    $\omega = \langle t | r \rangle / |t|^2$
    $x = x + \omega r$
    $r = r - \omega t$
    if $|r|/\sigma < \epsilon$ **then**
        exit
    end if
    $\rho = \langle \tilde{r} | r \rangle$
    $\beta = (\alpha/\omega)(\rho/\rho_0)$
    $\rho_0 = \rho$
    $p = p - \omega q$
    $p = r + \beta p$
end for

Algorithm 3 CG

input x,b, Solve $Ax = b$.
$q = Ax$
$r = b - Ax$
$p = r$
$\sigma = |b|$
$\rho_0 = |r|^2$
for do
    $q = Ap$
    $\alpha = \rho_0 / \langle p | q \rangle$
    $x = x + \alpha p$
    $r = r - \alpha q$
    $\rho = |r|^2$
    if $|r|/\sigma < \epsilon$ **then**
        exit
    end if
    $\beta = \rho / \rho_0$
    $\rho_0 = \rho$
    $p = r + \beta p$
end for