QCD, HMC プログラム説明書 (version 4.01)

石川健一

平成 14 年 12 月 25 日

目 次

1 even-odd 配列分割について **3**

1.1 はじめに . . . 3

1.2 even-odd 配列分割 . . . 3

1.3 plaquette の計算例 . . . 5

2 HMC アルゴリズムの説明 **9**

2.1 全体の流れ . . . 9

2.1.1 PHMC/hmonte_phmc.F . . . 11

2.2 $O(a)$ 改良された Wilson クォーク作用と even-odd プレコンディション . . . 15

2.3 全ハミルトニアンの計算 . . . 16

2.3.1 PHMC/hamil_phmc_all.F . . . 16

2.3.2 HMCLIB/hamil_clv.F . . . 20

2.3.3 HMCLIB/updatey_hmc.F . . . 21

2.3.4 HMCLIB/HMCLIB/hamil_hmc.F . . . 24

2.4 クローバー項の計算 . . . 26

2.4.1 CloverLIB/clvinv.F . . . 27

2.4.2 CloverLIB/clover_f1f2.F . . . 29

2.4.3 CloverLIB/clvinv_lu.F . . . 32

2.4.4 CloverLIB/chiral2dirac_clv.F . . . 42

2.5 分子動力学 . . . 45

2.5.1 PHMC/molecdyn_phmc.F . . . 45

2.5.2 HMCLIB/updateu_hmc.F . . . 48

2.5.3 HMCLIB/updatep_phmc.F . . . 55

2.5.4 HMCLIB/prep_force_g.F . . . 65

2.5.5 HMCLIB/force_g.F . . . 68

2.5.6 HMCLIB/force_hmc.F . . . 87

2.5.7 HMCLIB/force_clv.F . . . 95

2.5.8 HMCLIB/force_clover.F . . . 97

3 PHMC について **109**

1 even-odd 配列分割について

1.1 はじめに

我々の QCD のプログラムは even-odd 配列分割を採用している。even site 上のデータ、 odd site 上のデータ、 それぞれを受け持つ配列を別々に用意している。以下、1×1 Wilson loop (plaquette) を計算するプログラムを例に簡単に説明する。

1.2 even-odd 配列分割

我々の QCD のプログラムでは X, Y, Z 方向に比べて T 方向を長くとるため、配列参照が T 方向の変化に対して連続になるようにした。また even-odd 配列分割を行うのはできるだけ配列の参照が連続で長くなるように T 方向で二つに分けた。

図 1 にその分け方を示す。図 1 では 2 次元に投影して XYZ の長さ 8 、T の長さ 8 の格子を考える。横方向に XYZ 方向、縦方向に T 方向をとる。丸印を even、十字を odd とする。XYZ 方向にはそれぞれ外側に 1 つずつ拡張されたゴーストサイトを持っている。T 方向に対しては even-odd 分割されたためゴーストサイトは even,odd それぞれの配列に対しては 1 サイトだけ付け加わっている。演算対象は点線で囲まれた部分になる。例えば、この上に用意される配列として even サイトに ue(0:8/2,0:8+1)、odd サイトに uo(0:8/2,0:8+1) を用意したとする。ue,uo の添字は、はじめの方が T 方向に関してのもので、後ろのものが XYZ 方向のものになる。

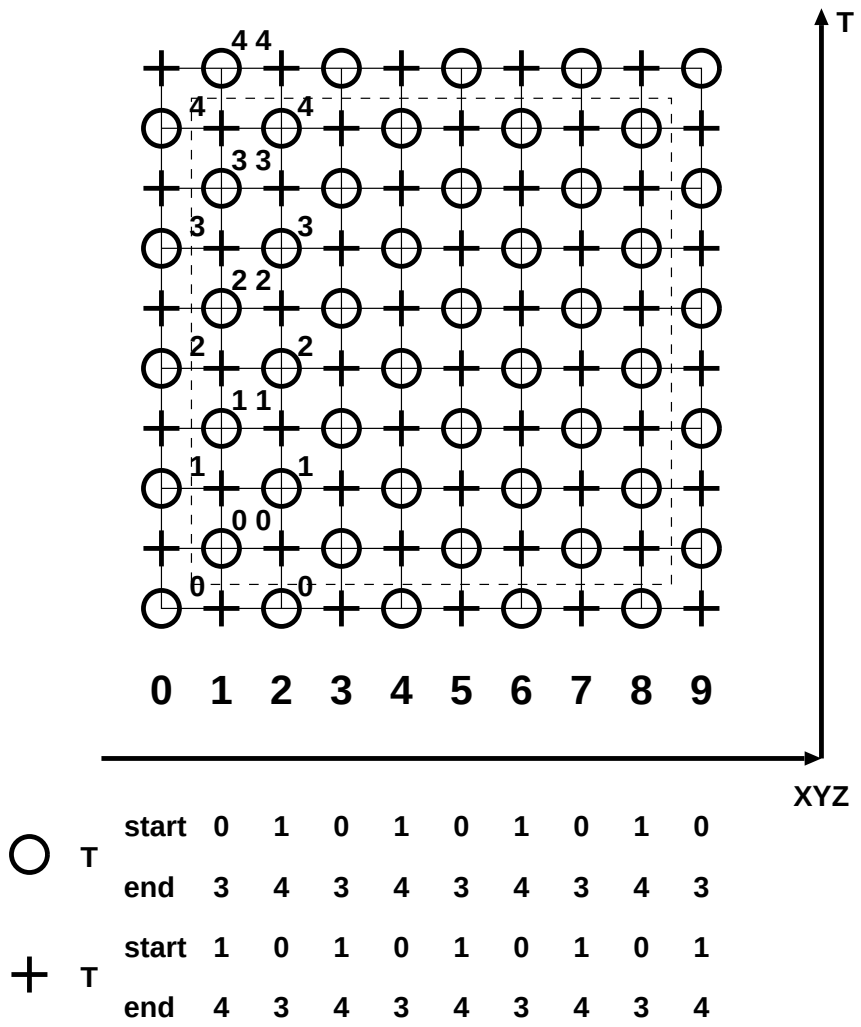


図 1: 分割

配列の参照を行うには、T と XYZ を指定しなければならないが、実際には even, odd に分割したので、T と XYZ から分割された配列の添字を計算するのではなく、直接添字を演算対象となるサイトをすべて訪れるように動かすことにしている。実際には T 方向の添字は XYZ 方向の添字によって、開始位置と終了位置が変化するように even、odd それぞれで配列参照を行う。図 1 では T 方向の添字の動き方は、$XYZ \bmod 2$ が 1 の場合 even ベクトル (丸) は 0 から 3、odd ベクトル (十字) は 1 から 4 を走る。$XYZ \bmod 2$ が 0 の場合は逆に odd ベクトル (十字) が 0 から 3 を走り、even ベクトル (丸) は 1 から 4 を走る。例えば ue,uo を 3 倍する場合のプログラムは以下のようになる。

even-odd 分割の例

```
integer, parameter :: NXYZ=8,NTT=8,NTH=NTT/2
complex(8) :: ue(0:NTH,0:NXYZ+1),uo(0:NTH,0:NXYZ+1)
integer :: ixyz,ieoxyz,itb

do ixyz=1,NXYZ
  ieoxyz=mod(ixyz,2)
  do itb=1-ieoxyz,NTH-ieoxyz
    ue(itb,ixyz)=3*ue(itb,ixyz)
  end do
```

```
        do itb=ieoxyz,NTH+ieoxyz-1
          uo(itb,ixyz)=3*uo(itb,ixyz)
        end do
      end do
```

1.3 plaquette の計算例

以下の例は plaquette の計算を行っているプログラムです。

plaquette の計算例

```
      integer, parameter :: COL=3,NDIM=4,NTH=NTT/2
      integer, parameter :: NY=NTY/NDIMY,NY=NTZ/NDIMZ,NY=NTX/NDIMX
      complex(8) :: ue(COL,COL,0:NTH,0:NZ+1,0:NY+1,0:NX+1,NDIM)
      complex(8) :: uo(COL,COL,0:NTH,0:NZ+1,0:NY+1,0:NX+1,NDIM)
      real(8) :: plqe,plqo,plq

      call plaquette(plqe,ue,uo,0)
      call plaquette(plqo,uo,ue,1)
      plq=plqe+plqo
```

により、plaquette の値が計算されます。ue,uo はそれぞれ even, odd サイトのゲージリンク変数です。サブルーチン plaquette は even サイトもしくは odd サイトにある plaquette を計算します (体積、自由度平均されている)。サブルーチン plaquette は以下のようになっています。

Prog/plaquette.F

```
1  #include "config.h"
2        subroutine plaquette(plq,ue,uo,ieo)
3  ************************************************************************
4  *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
5  * Calculate averaged plaquette
6  *
7  *  plq : averaged plaquett on even/odd site (ieo=0/1)
8  *   ue : even/odd site gauge link (ieo=0/1)
9  *   uo : odd/even site gauge link (ieo=0/1)
10 *  ieo : ieo=0 for even site, =1 for odd site
11 *
12 ************************************************************************
13 #include "param-size.h"
14 #include "com-nnb.h"
15       real(8), intent(out) :: plq
16       complex(8), intent(in) :: ue(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
17       complex(8), intent(in) :: uo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
18       integer, intent(in) :: ieo
19       integer :: ic,jc,mu,nu
20       integer :: ix,iy,iz,itb,ieoxyz
21       integer :: ix2,iy2,iz2,itb2,itb2d
22       integer :: ix4,iy4,iz4,itb4,itb4d
23       real(8) :: rtmp0
24       complex(8) ::   w23(COL,COL,0:NTH)
25       complex(8) ::   w41(COL,COL,0:NTH)
26       complex(8) :: w2341(COL,COL,0:NTH)
27
```

```
      rtmp0 = 0.0d0

*POPTION PARALLEL, PSUM (rtmp0)
#ifdef _COALESCING
*POPTION COALESCING (_COALESCING)
#endif
      do ix=1,NX
      do iy=1,NY
      do iz=1,NZ
        ieoxyz=mod(ipeo+ix+iy+iz+ieo,2)
        do mu = 1,NDIM
          ix2=ix+isx(mu)
          iy2=iy+isy(mu)
          iz2=iz+isz(mu)
          itb2d=+ieoxyz*ist(mu)
          do nu = 1,NDIM
          if (mu.EQ.nu) goto 1000
            ix4=ix+isx(nu)
            iy4=iy+isy(nu)
            iz4=iz+isz(nu)
            itb4d=+ieoxyz*ist(nu)
*VOPTION VEC
            do itb=1-ieoxyz,NTH-ieoxyz
              itb2=itb+itb2d
*SOPTION UNROLL (3)
              do jc=1,COL
*SOPTION UNROLL (3)
              do ic=1,COL
                w23(ic,jc,itb)=      ue(ic, 1,itb ,iz ,iy ,ix ,mu)*
     &                               uo( 1,jc,itb2,iz2,iy2,ix2,nu)
     &                              +ue(ic, 2,itb ,iz ,iy ,ix ,mu)*
     &                               uo( 2,jc,itb2,iz2,iy2,ix2,nu)
     &                              +ue(ic, 3,itb ,iz ,iy ,ix ,mu)*
     &                               uo( 3,jc,itb2,iz2,iy2,ix2,nu)
              enddo
              enddo
            enddo
*VOPTION VEC
            do itb=1-ieoxyz,NTH-ieoxyz
              itb4=itb+itb4d
*SOPTION UNROLL (3)
              do jc=1,COL
*SOPTION UNROLL (3)
              do ic=1,COL
                w41(ic,jc,itb)=conjg(uo( 1,ic,itb4,iz4,iy4,ix4,mu))*
     &                          conjg(ue(jc, 1,itb ,iz ,iy ,ix ,nu))
     &                         +conjg(uo( 2,ic,itb4,iz4,iy4,ix4,mu))*
     &                          conjg(ue(jc, 2,itb ,iz ,iy ,ix ,nu))
     &                         +conjg(uo( 3,ic,itb4,iz4,iy4,ix4,mu))*
     &                          conjg(ue(jc, 3,itb ,iz ,iy ,ix ,nu))
              enddo
              enddo
            enddo

*VOPTION VEC
            do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
```

```
85                do jc=1,COL
86  *SOPTION UNROLL (3)
87                do ic=1,COL
88                  w2341(ic,jc,itb)=w23(ic,1,itb)*w41(1,jc,itb)
89       &                          +w23(ic,2,itb)*w41(2,jc,itb)
90       &                          +w23(ic,3,itb)*w41(3,jc,itb)
91                enddo
92                enddo
93              enddo
94
95  *VOPTION VEC
96              do itb=1-ieoxyz,NTH-ieoxyz
97  *SOPTION UNROLL (3)
98                do ic=1,COL
99                  rtmp0 = rtmp0 + DBLE(w2341(ic,ic,itb))
100               enddo
101             enddo
102
103 1000      continue ! skip from if (mu.EQ.nu)
104           enddo      ! end of do nu
105         enddo      ! end of do mu
106       enddo
107       enddo
108       enddo ! end of do ix,iy,iz
109
110 #ifndef _singlePU
111       call comlib_sumcast_r8(rtmp0)
112 #endif
113
114       plq = rtmp0/(NTX*NTY*NTZ*NTT)/(NDIM-1)/NDIM/COL
115
116       return
117       end
118
```

引数 (出力) **plq :** even または odd サイトの plaquette の自由度平均値を返す。

引数 (入力) **ue :** `ieo = 0` の時は even サイトのリンク変数を渡す。`ieo = 1` の時は odd サイトのリンク変数を渡す。

引数 (入力) **uo :** `ieo = 0` の時は odd サイトのリンク変数を渡す。`ieo = 1` の時は even サイトのリンク変数を渡す。

引数 (入力) **ieo :** `ieo = 0` の時は even サイトの plaquette を計算する。`ieo = 1` の時は odd サイトの plaquette を計算する。

30-36 行目 : x, y, z 方向のループの開始。要素並列化の開始。

37 行目 : x, y, z 方向の位置からサイトの偶奇 (even-odd) 性を判断するインデックスの計算。`ipeo` は並列計算の時にこのサブルーチンを実行している計算ノードが受け持っている分割された格子の始まりが even サイトから始まっているのか odd サイトから始まっているのかを示すインデックス (even の時 0、odd の時 1 である)。

38 行目： plaquette $P_{\mu\nu}(n) = u_\mu(n)u_\nu(n+\hat{\mu})u_\mu^\dagger(n+\hat{\nu})u_\nu^\dagger(n)$ 計算のための μ 方向のループの開始。

39-42 行目： $n+\hat{\mu}$ の計算。

43 行目： ν 方向のループの開始。

44 行目： $\mu = \nu$ は計算しない。

45-48 行目： $n+\hat{\nu}$ の計算。

49-64 行目： $u_\mu(n)u_\nu(n+\hat{\mu})$ を計算する。

65-80 行目： $u_\mu^\dagger(n+\hat{\nu})u_\nu^\dagger(n)$ を計算する。

82-93 行目： $u_\mu(n)u_\nu(n+\hat{\mu})u_\mu^\dagger(n+\hat{\nu})u_\nu^\dagger(n)$ を計算する。

95-101 行目： トレース $\mathrm{Re}[\mathrm{Tr}[P_{\mu\nu}(n)]$ を計算して総和に加える。

110-112 行目： 並列計算の時はさらに各ノードのデータを総和する。

114 行目： 自由度平均をとる。

2 HMC アルゴリズムの説明

2.1 全体の流れ

全体の流れとしては、

1. 初期運動量の計算、擬フェルミオン場の計算、初期配位の全ハミルトニアンの計算
2. 分子動力学の計算
3. 新しい配位での全ハミルトニアンの計算
4. HMC メトロポリステスト

の繰り返しである。

以下の図は実際のプログラムの call tree です。

Call Tree

```
0001: AA0001
0002: AA0002
0003: `--SAVE_PATH
0004: AA0003
0005: AA0004
0006: AA0005
0007: PHMC_MAIN                          メインプログラム
0008: |--FNAME_PLAQ                      プラケットヒストリーをセーブするためのファイル名の生成
0009: |--HMONTE_PHMC                     ハイブリッドモンテカルロ
0010: |  |--HAMIL_PHMC_ALL               全ハミルトニアンの計算
0011: |  |  |--CLOVER                    クローバー項 (F_{mu,nu}) の計算
0012: |  |  |--HAMIL_CLV                 クローバー項から来るハミルトニアン (-Nf log[det[1+T]]) の計算
0013: |  |  |  |--CLVINV                 クローバー項の逆 ((1+T)^(-1)) と固有値の計算
0014: |  |  |  |  |--CHIRAL2DIRAC_CLV    カイラル表示からディラック表示への変換
0015: |  |  |  |  |--CLOVER_F1F2         カイラル表示で F_{mu,nu} から (1+T) の計算
0016: |  |  |  |  `--CLVINV_LU           LU 分解でカイラル表示の (1+T) の逆と固有値の計算
0017: |  |  |  |     |--LUBKSB           LU 分解から逆を求める (see.Numerical Recipes in FORTRAN, p38-)
0018: |  |  |  |     `--LUDCMP           LU 分解
0019: |  |  |  `--COPY_Y                 フェルミオンベクトルのゴーストサイトのコピー
0020: |  |  |--HAMIL_G                   ゲージ作用のハミルトニアンの計算
0021: |  |  |  `--PREP_FORCE_G           力の計算で必要で使い回しのできる部分の計算 (2-link)
0022: |  |  |     `--COPYXX              リンク変数に似たベクトルのゴーストサイトのコピー
0023: |  |  |--HAMIL_HMC           擬フェルミオンからのハミルトニアン (|D^-1psi|^2) の計算 (usual HMC)
0024: |  |  |  `--BICGSTAB_HMC  BiGCStab ソルバー，対称プレコンディションしたディラック演算子の逆を求める
0025: |  |  |     `--MULT_MB_PRE  対称プレコンディションしたディラック演算子をフェルミオンベクトルに掛ける
0026: |  |  |        `--MULT    フェルミオンベクトルにホッピング行列を掛けてさらにクローバー項の逆を掛ける
0027: |  |  |           |--COPY_Y_T      フェルミオンベクトルのゴーストサイトの時間 (T) 方向のコピー
0028: |  |  |           |--COPY_Y_X      フェルミオンベクトルのゴーストサイトの空間 (X) 方向のコピー
0029: |  |  |           |--COPY_Y_Y      フェルミオンベクトルのゴーストサイトの空間 (Y) 方向のコピー
0030: |  |  |           |--COPY_Y_Z      フェルミオンベクトルのゴーストサイトの空間 (Z) 方向のコピー
0031: |  |  |           |--MULT_EO_T     フェルミオンベクトルに時間 (T) 方向のホッピング行列を掛ける
0032: |  |  |           |--MULT_EO_X     フェルミオンベクトルに空間 (X) 方向のホッピング行列を掛ける
0033: |  |  |           |--MULT_EO_Y     フェルミオンベクトルに空間 (Y) 方向のホッピング行列を掛ける
0034: |  |  |           |--MULT_EO_Z     フェルミオンベクトルに空間 (Z) 方向のホッピング行列を掛ける
0035: |  |  |           `--MULT_FCLINV   フェルミオンベクトルにクローバー項の逆を掛ける
0036: |  |  |--HAMIL_PHMC          擬フェルミオンからのハミルトニアン (|P[D]psi|^2) の計算 (polynomial HMC)
0037: |  |  |  `--MULT          > 0026
0038: |  |  |--HAMIL_UV                  UV フィルタリングした場合にでてくるハミルトニアンの計算 (使わない)
0039: |  |  |--INITP                     ゲージリンク変数に対する虚運動量ベクトルのガウス分布からの生成
0040: |  |  |  `--RAND_RR                乱数生成
0041: |  |  |--UPDATEY_HMC               擬フェルミオンからのハミルトニアン (|D^-1psi|^2) の計算と
      |  |  |  |                         擬フェルミオンベクトルの生成 (usual HMC)
0042: |  |  |  |--GAUSS_Y                フェルミオンベクトルにガウス分布を生成
```

```
0043: |  |  |  |  `--RAND_RR
0044: |  |  |  `--MULT_MB_PRE > 0025
0045: |  |  `--UPDATEY_PHMC             擬フェルミオンからのハミルトニアン (|P[D]psi|^2) の計算と
      |  |     |                        擬フェルミオンベクトルの生成 (polynomial HMC)
0046: |  |     |--BICGSTAB_MB_POLY_PRE  BiCGStab ソルバー、P[D]D の逆を求める
0047: |  |     |  `--MULT_MB_POLY_PRE       P[D]D をフェルミオンベクトルに掛ける
0048: |  |     |     |--MULT_EXP_TAYLOR2    UV フィルタリングを掛ける (使わない)
0049: |  |     |     |  `--MULT_UV_PRE      (ホッピング行列+クローバー項の逆) を 2 回掛ける
0050: |  |     |     |     `--MULT       > 0026
0051: |  |     |     |--MULT_MB_PRE > 0025
0052: |  |     |     `--MULT_UV_PRE > 0049
0053: |  |     |--GAUSS_Y    > 0042
0054: |  |     |--MULT       > 0026
0055: |  |     |--MULT_EXP_TAYLOR2 > 0048
0056: |  |     `--MULT_MB_PRE > 0025
0057: |  |--METROPOLIS_HMC              HMC のメトロポリステスト
0058: |  |  `--RAND_RR
0059: |  `--MOLECDYN_PHMC               分子動力学
0060: |     |--REUNIT                   ゲージリンクの再ユニタリー化
0061: |     |--UPDATEP_PHMC             ゲージリンク変数に対する虚運動量ベクトルを時間発展させる
0062: |     |  |--CLOVER
0063: |     |  |--CLVINV     > 0013
0064: |     |  |--COPYXX
0065: |     |  |--COPY_Y
0066: |     |  |--FORCE_CLOVER          クローバー項から来る力の計算
0067: |     |  |--FORCE_CLV             -Nf log[det[1+T]] から来る力の途中結果の計算
0068: |     |  |--FORCE_G               ゲージ作用からの力の計算
0069: |     |  |  `--COPYU_RG           改良されたゲージ作用の場合に必要となるリンクのコピー
0070: |     |  |     `--COPYU           リンクのコピー
0071: |     |  |--FORCE_HMC             擬フェルミオン作用からの力の計算 (usual HMC)
0072: |     |  |  |--BICGSTAB_HMC > 0024
0073: |     |  |  |--COPY_Y
0074: |     |  |  |--MULT       > 0026
0075: |     |  |  |--MULT_FCLINV
0076: |     |  |  `--MULT_GAMMA5        gamma_5 をフェルミオンベクトルに掛ける
0077: |     |  |--FORCE_PHMC            擬フェルミオン作用からの力の計算 (polynomial HMC)
0078: |     |  |  |--COPY_Y
0079: |     |  |  |--MULT       > 0026
0080: |     |  |  |--MULT_FCLINV
0081: |     |  |  `--MULT_GAMMA5
0082: |     |  |--FORCE_UV              UV フィルタリング作用からの力の計算 (使っていない)
0083: |     |  `--PREP_FORCE_G > 0021
0084: |     `--UPDATEU_HMC              ゲージリンクを時間発展させる
0085: |        `--COPYU                 ゲージリンク場のゴーストサイトのコピー
0086: |--INITSET                        gamma 行列、並列化の初期化
0087: |--INITU                          ゲージリンク変数の初期化
0088: |--INIT_RAND                      乱数の初期化
0089: |--METROPOLIS_MB                  多項式 HMC のためのメトロポリステスト
0090: |  |--METROPOLIS_MB_EVENFLV            偶数フレーバーの場合のメトロポリステスト
0091: |  |  |--BICGSTAB_MB_POLY_PRE > 0046
0092: |  |  |--CLOVER
0093: |  |  |--CLVINV     > 0013
0094: |  |  |--GAUSS_Y    > 0042
0095: |  |  |--MULT_MB_POLY_PRE > 0047
0096: |  |  `--RAND_RR
0097: |  `--METROPOLIS_MB_ODDFLV           奇数フレーバーの場合のメトロポリステスト
0098: |     |--CLOVER
0099: |     |--CLVINV     > 0013
0100: |     |--GAUSS_Y    > 0042
0101: |     |--MULT_MB_POLY_PRE > 0047
0102: |     |--MULT_SQRT_MB_POLY_PRE_TAYLOR
0103: |     |  `--MULT_MB_POLY_PRE > 0047
```

```
0104: |     `--RAND_RR
0105: |--PRINT_PARAM_PHMC              プログラムのインプットパラメータなどを表示
0106: |  |--SQRTPOLY                  偶数次の多項式を二つに分ける
0107: |  |  |--INDEXX                 ソートプログラム
0108: |  |  `--ZROOTS                 多項式の根を求める
0109: |  |     `--LAGUER
0110: |  `--SQRTPOLY_CHEV           Chebyshev 多項式 (Taylor 展開) の場合の偶数次の多項式を二つに分ける
0111: |     `--INDEXX
0112: |--READ_CONTG_HMC                継続のためのゲージ場の読み込み
0113: |  |--COMP_JOBINFO               引き継ぎパラメータのチェック
0114: |  |--FNAME_CONT                 継続のためのゲージ場のファイル名生成
0115: |  `--PLAQUETTE                  チェックサムとしてプラケットを計算
0116: |--READ_JOBFILE_HMC              メインプログラムのインプットパラメータを読み込む
0117: |--READ_QUARKPARAM               各クォークのパラメータを読み込む
0118: |  `--INDEXX
0119: |--SAVE_CONF                     物理量を計算するためのゲージ場の保存
0120: |  |--FNAME_SAVE                 物理量を計算するためのゲージ場のファイル名の生成
0121: |  |  `--SAVE_PATH               物理量を計算するためのゲージ場の保存ディレクトリの生成
0122: |  `--SAVE_PATH
0123: |--WRITE_CONTG_HMC               継続のためのゲージ場の保存
0124: |  |--FNAME_CONT
0125: |  `--PLAQUETTE
0126: `--WRITE_JOBFILE_HMC            メインプログラムのインプットパラメータの保存
```

2.1.1 PHMC/hmonte_phmc.F

HMC の心臓部は PHMC/hmonte_phmc.F です。以下にプログラムを示します。

PHMC/hmonte_phmc.F

```
1  #include "config.h"
2        subroutine hmonte_phmc(jobs,ue,uo,quark_all,ye,yo,plq0,plq1,ihit)
3  ************************************************************************
4  *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
5  * Hybrid Monte Carlo
6  *
7  *      jobs : job parameters
8  *        ue : even site gauge links
9  *        uo : odd site gauge links
10 * quark_all : quark parameters
11 *        ye : even site boson filelds
12 *        yo : odd site boson filelds
13 *      plq0 : initial averaged plaquette
14 *      plq1 : final   averaged plaquette
15 *      ihit : result of HMC Metropolis test, 0 rejected, 1 accepted.
16 *
17 ************************************************************************
18       use multiboson
19 #include "param-size.h"
20       type(jobinfo_def), intent(inout) :: jobs
21       complex(8), intent(inout) ::
22      &                       ue(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM),
23      &                       uo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
24       type(quarks_def), intent(inout) :: quark_all
25       complex(8), intent(inout) ::
26      &             ye(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1,quark_all%nmb),
27      &             yo(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1,quark_all%nmb)
28       real(8), intent(out) :: plq0,plq1
29       integer, intent(out) :: ihit
30 ***************
```

```
      complex(8) :: pe(COL,COL,0:NTH,NZ,NY,NX,NDIM)
      complex(8) :: po(COL,COL,0:NTH,NZ,NY,NX,NDIM)
      real(8) :: H0,H1,dH
#ifdef _REVCHECK
      real(8) :: du
      complex(8) :: u0e(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
      complex(8) :: u0o(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)

      u0e=ue
      u0o=uo
#endif

      ihit=0

************************************
* update external fields (y,p)
* calc. initial Hamiltonian
************************************
      call hamil_phmc_all(H0,plq0,0,jobs,pe,po,ue,uo,quark_all,ye,yo)

************************************
* Molecular Dynamics sweep
************************************
      call molecdyn_phmc(jobs,pe,po,ue,uo,quark_all,ye,yo)

************************************
* calc. final Hamiltonian, and plaquette
************************************
      call hamil_phmc_all(H1,plq1,1,jobs,pe,po,ue,uo,quark_all,ye,yo)

************************************
* Metropolis tesf for HMC
************************************
      dH=H1-H0
      call metropolis_hmc(dH,jobs,quark_all,ihit)

#ifdef _REVCHECK
****** reversi
      jobs%dt=-jobs%dt
      call molecdyn_phmc(jobs,pe,po,ue,uo,quark_all,ye,yo)
      call hamil_phmc_all(H1,plq,1,jobs,pe,po,ue,uo,quark_all,ye,yo)
      jobs%dt=-jobs%dt
      call delta_u(du,u0e,u0o,ue,uo)
      stop
#endif

      return
      end
```

引数 (入出力) **jobs** : プログラムパラメータを含む構造体。

引数 (入出力) **ue, uo** : even サイト、odd サイトのゲージリンク変数。入力は初期配位、出力は新しい配位。

引数 (入出力) **quark_all** : 全クォークに関するパラメータを含む構造体

引数 (入出力) **ye, yo :** even サイト、odd サイトの擬フェルミオン場と作業配列

引数 (出力) **plq0 :** 初期配位での plaquette の値

引数 (出力) **plq1 :** 新しい配位での plaquette の値

引数 (出力) **ihit :** HMC メトロポリステストの結果。1 のときアタリ、0 のときハズレ。

49 行目**:** 運動量、擬フェルミオン場の初期化と、初期配位での全ハミルトニアンの計算。

54 行目**:** 分子動力学による積分

59 行目**:** 新しい配位での全ハミルトニアンの計算

64-65 行目**:** HMC メトロポリステスト

ここで、良く使う構造体は INCLUDE/multiboson.F で定義されている。

INCLUDE/multiboson.F

```
1  ************************************************************************
2  *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
3  * Definition of quark parameters
4  ************************************************************************
5        module multiboson
6        implicit none
7        integer , parameter :: MAXNMB=500    ! maximum number of multi boson for each quark
8        integer , parameter :: MAXNQUARK=5   ! maximum number of individual quark
9
10 ****************************************
11 * An independent quark parameters (Nf=1 or 2)
12 ****************************************
13       type quark_def
14         sequence
15         real(8) :: kappa             ! hopping parameter
16         real(8) :: csw               ! clover coefficient
17         real(8) :: pcoef(0:MAXNMB)   ! polynomial coefficients pcoef(0:nmb)
18         real(8) :: acoef             ! UV filtering coefficient
19         real(8) :: amp_bicgstab_err  ! BiCGStab stopping condition for Global Metropolis
20                                      ! (Corr for MB, Hamil for HMC)
21         real(8) :: aqhb_bicgstab_err ! BiCGStab stopping condition
22                                      ! (QHB for MB, Force for HMC)
23         real(8) :: anorm             ! approximate value of a norm of
24                                      ! hopping matrix ||M^2||
25         complex(8) :: zcoef(0:MAXNMB) ! polynomial coefficients zcoef(0:nmb/2)
26                                       ! for single flavor
27         complex(8) :: zmulti(MAXNMB) ! polynomial roots
28         complex(8) :: zmultie(MAXNMB)! polynomial roots (even odd preconditioned)
29         integer :: nmethod           ! update method ( 0 for MB , 1 for HMC )
30         integer :: npoly             ! order of polynomial
31         integer :: nmb               ! # of multi bosons = npoly/(3-nf)
32         integer :: nflavor           ! number of flavor of this quark, 1 or 2
33         integer :: ptype             ! polynomial type, ptype=0 for Chebyshev,
34                                      ! 1 for Adopted, 2 for UV
35         integer :: nexp              ! order of the Taylor expansion of UV
36                                      ! filtering exponential factor
37         integer :: nsqrt             ! order of the Taylor expansion of the square root
38                                      ! of correction factor
39         integer :: iqnum             ! quark number 1,2,...
```

```
        integer :: index             ! field start index, y(1,1,0,0,0,0,index:index+nmb-1)
        integer :: nmp_bicgstab_iter ! # of iteration for Global Metropolis
        integer :: nmp_try,nmp_hit   ! # of try, # of hit for Global Metropolis
        integer, dimension(MAXNMB)
     &          :: nqhb_bicgstab_iter,
     &             nqhb_try,nqhb_hit ! # of BiCGStab iteration, # for try, # of hit
      end type

*****************************************
* All quark parameter used in local updation/QHB updation
*****************************************
      type quarks_def
        type(quark_def) ::  quark(MAXNQUARK)   ! quark parameters
        integer :: nquark                      ! # of indep quark
        integer :: nmb                         ! # of all boson field
        integer :: nhmc_try,nhmc_hit           ! HMC statistics
      end type

*****************************************
* JOB info
*****************************************
      type jobinfo_def
        real(8) :: beta                ! beta
        real(8) :: cRG(0:3)            ! RG-gauge improvement coefficients
        real(8) :: u0                  ! tadpole improvement coefficients
        real(8) :: dt                  ! MD time step
        integer :: niter               ! # of MD step
        integer :: ibsn                ! MB switch, 0 off, 1 on
        integer :: nquark              ! # of dynamical quark (indep mass)
        integer :: igmp                ! GMP switch for  MB, 0 off, 1 on
        integer :: igmp_hmc            ! GMP switch for HMC, 0 off, 1 on
        integer :: iseed               ! random number seed
        integer :: icont               ! total # of continuation of job
        integer :: itraj               ! total # of trajectory (sweep)
        integer :: nsweep              ! # of sweep in this job
        integer :: nskip               ! # of skip of sweep to store conf.
                                       ! ( if itraj mod nskip == 0 then store)
        character(200) :: fnamequark(MAXNQUARK)
                                       ! file name in which quark parameters
                                       ! are described.
        character(200) :: jobname      ! job name
        character(200) :: prefix       ! directory name where configuration files are
                                       ! stored
        character(200) :: fhead        ! configuration file name head
      end type

      end module

```

注意としては、すでに使われていない定義も含んでいる。
構造体のメンバーの参照は、

(構造体)%(構造体メンバー)

の形で行う。INCLUDE/multiboson.F は構造体の定義を含むモジュール副プログラムとして各サブルーチンから引用される。例えば、PHMC/hmonte_phmc.F では 18 行目に use multiboson として上記のモ

ジュール副プログラムが引用されている。これによりモジュール副プログラム内で定義された構造体がこのサブルーチン内で使えるようになる (20 行目 `type(jobinfo_def), intent(inout) :: jobs`)。

2.2 $O(a)$ 改良された Wilson クォーク作用と even-odd プレコンディション

QCD 分配関数でフェルミオンに関する部分は、

$$\det[D] = \det \begin{pmatrix} 1+T_{ee} & M_{eo} \\ M_{oe} & 1+T_{oo} \end{pmatrix} \tag{1}$$

で書けるが、ここでは対称な even-odd プレコンディションを行い、

$$\det[D] = \det[1+T_{ee}] \det[1+T_{oo}] \det[\hat{D}_{oo}], \tag{2}$$

$$\hat{D}_{oo} = 1 - (1+T_{oo})^{-1} M_{oe} (1+T_{ee})^{-1} M_{eo}. \tag{3}$$

を用いた。プログラムでは `MultLIB/mult.F` が $(1+T_{ee})^{-1}M_{eo}$(または $(1+T_{oo})^{-1}M_{oe}$) を計算している。

ホッピングマトリックスの定義は、

$$M(n,m) = -\kappa \sum_{\mu} \left[(1-\gamma_\mu) U_\mu(n) \delta_{n+\hat{\mu},m} + (1+\gamma_\mu) U_\mu^\dagger(n-\hat{\mu}) \delta_{n-\hat{\mu},m} \right] \tag{4}$$

です。クローバー項の定義は、

$$\begin{aligned}
(1+T)(n,m) &= \delta_{n,m}(1 - \frac{c_{\rm sw}\kappa}{2} \sigma_{\mu\nu} F_{\mu\nu}(n)), \\
F_{\mu\nu}(n) &= -\frac{i}{8}(C_{\mu\nu}(n) - C_{\mu\nu}(n)^\dagger), \\
C_{\mu\nu}(n) &= U_\mu(n) U_\nu(n+\hat{\mu}) U_\mu^\dagger(n+\hat{\nu}) U_\nu^\dagger(n) \\
&+ U_\nu(n) U_\mu^\dagger(n-\hat{\mu}+\hat{\nu}) U_\nu^\dagger(n-\hat{\mu}) U_\mu(n-\hat{\mu}) \\
&+ U_\mu^\dagger(n-\hat{\mu}) U_\nu^\dagger(n-\hat{\mu}-\hat{\nu}) U_\mu(n-\hat{\mu}-\hat{\nu}) U_\nu(n-\hat{\nu}) \\
&+ U_\nu^\dagger(n-\hat{\nu}) U_\mu(n-\hat{\nu}) U_\nu(n+\hat{\mu}-\hat{\nu}) U_\mu^\dagger(n), \\
\sigma_{\mu\nu} &= \frac{i}{2}[\gamma_\mu, \gamma_\nu].
\end{aligned} \tag{5}$$

です。

$N_f = 2$ の HMC QCD 作用は、

$$\begin{aligned}
H[P,U,\psi_o^\dagger,\psi_o] &= T[P] + S_g[U] + S_q[U,\psi_o^\dagger,\psi_o] + S_{\rm det}[U], \\
T[P] &= \frac{1}{2} \sum_{n,\mu} {\rm Tr}[P_\mu(n)^2], \\
S_g[U] &= \frac{\beta}{6} \sum_{n,\mu,\nu} {\rm Tr}[1 - P_{\mu\nu}(n)], \\
S_q[U,\psi_o^\dagger,\psi_o] &= \left| \hat{D}_{oo}^{-1} \psi_o \right|^2, \\
S_{\rm det}[U] &= -N_f \sum_n \log[\det[1+T(n)]],
\end{aligned} \tag{6}$$

である。

2.3 全ハミルトニアンの計算

全ハミルトニアンの計算は式 (6) に従い以下のプログラムで行っている。

PHMC/hamil_phmc_all.F	ハミルトニアンの計算のメイン
HMCLIB/hamil_clv.F	$S_{\rm det}[U]$ の計算
HMCLIB/hamil_g.F	$T[P] + S_g[U]$ の計算
HMCLIB/hamil_hmc.F	$S_q[U, \psi_o^\dagger, \psi_o]$ の計算
HMCLIB/initp.F	運動量 $P_\mu(n)$ の生成
HMCLIB/updatey_hmc.F	擬フェルミオン場 ψ_o の生成と $S_q[U, \psi_o^\dagger, \psi_o]$ の計算
PHMC/hamil_UV.F	(PHMC で UV フィルタリングした場合の余分な作用の計算。使ってない)
PHMC/hamil_phmc.F	(PHMC の $S_q[U, \psi_o^\dagger, \psi_o]$ の計算)
PHMC/updatey_phmc.F	(PHMC の擬フェルミオン場 ψ_o の生成と $S_q[U, \psi_o^\dagger, \psi_o]$ の計算)

2.3.1 PHMC/hamil_phmc_all.F

全ハミルトニアンの計算のメイン PHMC/hamil_phmc_all.F を以下に示す。

PHMC/hamil_phmc_all.F

```
1  #include "config.h"
2        subroutine hamil_phmc_all(H,plq,ifi,
3       &                          jobs,pe,po,ue,uo,quark_all,ye,yo)
4  ************************************************************************
5  *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
6  * Calc. Hamiltonian
7  *
8  *       ifi : 0 for initial (update (y,p)), 1 for final, 2 initial (update p only)
9  *         H : Hamiltonian
10 *       plq : averaged plaquett of final conf.
11 *      jobs : job parameters
12 *        pe : even site canonical momentum (if ifi=1, updated)
13 *        po : odd site canonical momentum (if ifi=1, updated)
14 *        ue : even site gauge link
15 *        uo : odd site gauge link
16 * quark_all : all quark parameters
17 *        ye : even site gauge multi boson (if ifi=1, updated)
18 *        yo : odd site gauge multi boson (if ifi=1, updated)
19 *
20 ************************************************************************
21       use multiboson
22 #include "param-size.h"
23       type(jobinfo_def), intent(inout) :: jobs
24       real(8), intent(out) :: H,plq
25       integer, intent(in) :: ifi
26       complex(8), intent(inout) :: pe(COL,COL,0:NTH,NZ,NY,NX,NDIM)
27       complex(8), intent(inout) :: po(COL,COL,0:NTH,NZ,NY,NX,NDIM)
28       complex(8), intent(in) :: ue(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
29       complex(8), intent(in) :: uo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
30       type(quarks_def), intent(inout) :: quark_all
31       complex(8), dimension(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1,
32      &                      quark_all%nmb),
33      &            intent(in) :: ye     ! work
34       complex(8), dimension(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1,
35      &                      quark_all%nmb),
36      &            intent(inout) :: yo
37 #ifdef _CLOVER
38       complex(8) ::
```

```
     &    ucle(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &    uclo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2)
      common /WORKCLOVER/ ucle,uclo
      complex(8), dimension(CLSP,0:NTH,0:NZ1,0:NY1,0:NX1,CLSP)
     &           :: fclinve,fclinvo
      common /WORKFCLINV/ fclinve,fclinvo
#endif
      real(8) :: SU,SP
      real(8) ::  SQ(quark_all%nquark)
      real(8) ::  SF(quark_all%nquark)
      real(8) :: SUV(quark_all%nquark)
      real(8) :: SF_all,SUV_all,SQ_all
      integer :: iout,imb,iqk

      iout=22

****************************************
* update canonical momentum
****************************************
      if (ifi.EQ.0) call initp(pe,po)

****************************************
* calc. Hamiltonian of gauge contrib.
****************************************
#ifdef _RGGAUGE
      call hamil_g(jobs%beta,jobs%cRG,SU,SP,plq,pe,po,ue,uo)
#else
      call hamil_g(jobs%beta,SU,SP,plq,pe,po,ue,uo)
#endif

      select case (jobs%ibsn)
      case (0)
        do iqk=1,quark_all%nquark
          SQ(iqk) =0.0d0
          SF(iqk) =0.0d0
          SUV(iqk)=0.0d0
        enddo
      case (1)
#ifdef _CLOVER
****************************************
* calc. clover leaf F_{mu,nu}
****************************************
        call clover(ucle,uclo,ue,uo)
#endif
        do iqk=1,quark_all%nquark
          imb=quark_all%quark(iqk)%index

#ifdef _CLOVER
****************************************
* calc. F^-1 and Hamil. of clover contrib.
****************************************
          call hamil_clv(SF(iqk),quark_all%quark(iqk),
     &                   ucle,uclo,fclinve,fclinvo)
#else
          SF(iqk)=0.0d0
#endif

```

```
          select case (quark_all%quark(iqk)%nmethod)
          case (0)  ! PHMC
****************************************
* PHMC
* calc. Hamil. pseudo fermion (lives on odd site) contrib.
****************************************
            select case (ifi)
            case (0)
              call updatey_phmc(SQ(iqk),quark_all%quark(iqk),
     &                        yo(1,1,0,0,0,0,imb),
#ifdef _CLOVER
     &                        uo,ue,1,fclinvo,fclinve)
#else
     &                        uo,ue,1)
#endif
            case (1)
              call hamil_phmc(SQ(iqk),quark_all%quark(iqk),
     &                        yo(1,1,0,0,0,0,imb),
#ifdef _CLOVER
     &                        uo,ue,1,fclinvo,fclinve)
#else
     &                        uo,ue,1)
#endif
            end select
#ifdef _CLOVER
****************************************
* calc. Hamil. UV filter contrib.
****************************************
            call hamil_UV(SUV(iqk),quark_all%quark(iqk),
     &                    ue,uo,fclinve,fclinvo)
#else
            SUV(iqk)=0.0d0
#endif
          case (1) ! HMC
****************************************
* HMC
* calc. Hamil. pseudo fermion (lives on odd site) contrib.
****************************************
            select case (ifi)
            case (0)
              call updatey_hmc(SQ(iqk),quark_all%quark(iqk),
     &                        yo(1,1,0,0,0,0,imb),
#ifdef _CLOVER
     &                        uo,ue,1,fclinvo,fclinve)
#else
     &                        uo,ue,1)
#endif
            case (1)
              call hamil_hmc(SQ(iqk),quark_all%quark(iqk),
     &                        yo(1,1,0,0,0,0,imb),
#ifdef _CLOVER
     &                        uo,ue,1,fclinvo,fclinve)
#else
     &                        uo,ue,1)
#endif
            end select
            SUV(iqk)=0.0d0
```

```
153           end select
154         enddo
155       end select
156
157       SQ_all =0.0d0
158       SF_all =0.0d0
159       SUV_all=0.0d0
160       do iqk=1,quark_all%nquark
161         SQ_all =SQ_all + SQ(iqk)
162         SF_all =SF_all + SF(iqk)
163         SUV_all=SUV_all+SUV(iqk)
164       enddo
165
166       H=SU+SP+SQ_all+SF_all+SUV_all
167
168       if (nodeid.EQ.0) then
169         write(iout,'("@",I8,I2,
170      &               "   H:",E24.16,"  SU:",E24.16,"  SP:",E24.16,
171      &               "  SQ:",E24.16,"  SF:",E24.16," SUV:",E24.16)')
172      &  jobs%itraj,mod(ifi,2),H,SU,SP,SQ_all,SF_all,SUV_all
173         write(iout,'("@",I8,I2," PLQ:",F20.16)')
174      &  jobs%itraj,mod(ifi,2),plq
175         do iqk=1,quark_all%nquark
176           write(iout,'("  Quark:",I3,"  SQ:",E24.16,"  SF:",E24.16,
177      &                                " SUV:",E24.16)')
178      &    iqk,SQ(iqk),SF(iqk),SUV(iqk)
179         enddo
180         write(iout,*)
181       endif
182
183       return
184       end
185
```

引数 (出力) **H :** 全ハミルトニアンの値

引数 (出力) **plq :** plaquette の値

引数 (入力) **ifi :** 0 の時、運動量と擬フェルミオン場を初期化してから、全ハミルトニアンを計算する。1 の時は全ハミルトニアンを計算するだけ。(注) プログラムのコメントは間違っている!

引数 (入出力) **jobs :** プログラムパラメータを含む構造体

引数 (入出力) **pe, po :** even サイト、odd サイトのゲージ場の運動量、`ifi=0` の時は初期化される、`ifi=1` の時は変化なし。

引数 (入力) **ue, uo :** even サイト、odd サイトのゲージ場

引数 (入出力) **quark_all :** 全クォークのパラメータを含む構造体

引数 (入力) **ye :** even サイトの擬フェルミオン場と同じ形の作業配列。

引数 (入出力) **yo :** odd サイトの擬フェルミオン場と作業配列。`ifi=0` の時は各クォークの擬フェルミオン場は初期化される。`ifi=1` の時は変化なし。擬フェルミオンは odd サイトにいる。

58 行目 : ifi=0 の時、運動量を初期化

63-67 行目 : ゲージ作用と運動量からのハミルトニアンの寄与、$T[P] + S_g[U]$ の計算。

69-行目 : クォークの効果を取り入れている場合 (jobs%ibsn=1 の時)、各クォークからのハミルトニアンへの寄与を計算する

81 行目 : クローバー項 $(F_{\mu\nu})$ を計算

83 行目 : 各クォークをめぐるループの開始

84 行目 : 各クォークの擬フェルミオン場の配列へのインデックスを取得

90-91 行目 : クローバー項 $(1 + T)$ の逆を求めつつハミルトニアンへの寄与 $S_{\rm det}[U]$ を計算

96 行目 : クォークを HMC で評価するか PHMC で評価するのかの判断。quark_all%quark(iqk)%nmethod=0 の時は PHMC、quark_all%quark(iqk)%nmethod=1 の時は HMC。

102-128 行目 : PHMC の時の $S_q[U, \psi_o^\dagger, \psi_o]$ の計算と ψ_o の初期化

129-153 行目 : HMC の時の $S_q[U, \psi_o^\dagger, \psi_o]$ の計算と ψ_o の初期化

157-166 行目 : 全ハミルトニアンを計算

168-181 行目 : 結果をログに出力

2.3.2 HMCLIB/hamil_clv.F

ゲージ作用に関するハミルトニアンは HMCLIB/hamil_g.F で行っているが自明なので省略する。クローバー項の行列式から来るハミルトニアンは HMCLIB/hamil_clv.F で行っている。

HMCLIB/hamil_clv.F

```
1  #include "config.h"
2  #ifdef _CLOVER
3        subroutine hamil_clv(action,quark,
4       &                     ucle,uclo,fclinve,fclinvo)
5  ************************************************************************
6  *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
7  * Calculate Hamiltonian of clover term
8  *
9  *  Action : - Nf log[det[F]]
10 *
11 *  action : action value
12 *   quark : quark parameters
13 *    ucle : even site clover leaf
14 *    uclo : odd site clover leaf
15 * fclinve : even site inverse clover term (transposed)
16 * fclinvo : odd site inverse clover term (transposed)
17 *
18 ************************************************************************
19       use multiboson
20 #include "param-size.h"
21       real(8), intent(out) :: action
22       type(quark_def), intent(in) :: quark
23       complex(8), intent(in) ::
24      &    ucle(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
```

```
25      &    uclo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2)
26       complex(8), dimension(CLSP,0:NTH,0:NZ1,0:NY1,0:NX1,CLSP),
27      &            intent(out) :: fclinve,fclinvo
28  **********************
29       real(8) :: logdetfcl
30       integer :: ic
31
32  ************************************
33  * calc. inverse clover term and log[det[Fcl]]
34  ************************************
35       call clvinv(quark%kappa,
36      &            quark%csw,
37      &            fclinve,fclinvo,ucle,uclo,logdetfcl,1)
38       do ic=1,CLSP
39         call copy_y(fclinve(1,0,0,0,0,ic),0)
40         call copy_y(fclinvo(1,0,0,0,0,ic),1)
41       enddo
42
43       action=-logdetfcl*quark%nflavor
44
45       return
46       end
47  #else
48       subroutine hamil_clv
49       return
50       end
51  #endif
52  ********** end of ifdef _CLOVER
53
```

引数 (出力) **action :** $S_{\rm det}[U]$ の値が返る

引数 (入力) **quark :** ある 1 つのクォークのパラメータが入っている構造体

引数 (入力) **ucle, uclo:** even サイト、odd サイトのクローバー項 $(F_{\mu\nu})$ の入った配列

引数 (出力) **fclinve, fclinvo :** even サイト、 odd サイトのクローバー項の逆 $((1+T)^{-1})$ が配列に返る

35-37 行目: $F_{\mu\nu}$ から $(1+T)^{-1}$ と $\log\det[1+T]$ を求める

38-41 行目: $(1+T)^{-1}$ のゴーストサイトのコピー

43 行目: 作用 $S_{\rm det}[U]$ の計算

2.3.3 HMCLIB/updatey_hmc.F

擬フェルミオン場を初期化して作用 $S_q[U,\psi_o^\dagger,\psi_o]$ を計算するのが HMCLIB/updatey_hmc.F です。

HMCLIB/updatey_hmc.F

```
1   #include "config.h"
2   #ifdef _CLOVER
3         subroutine updatey_hmc(action,quark,ye,ue,uo,ieo,fclinve,fclinvo)
4   #else
5         subroutine updatey_hmc(action,quark,ye,ue,uo,ieo)
6   #endif
7   ************************************************************************
```

```
*$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
* update HMC pseudo fermion :
*
* Shmc = |Dee^-1 ye|^2
*
*  v
*  Dee^-1 ye = eta, where  eta : Gaussian noise,
*
*  v
*  Dee = 1 - kappa^2 F^-1ee Meo F^-1oo Moe
*
* action : Hamintonian (initial)
*  quark : quark parameters, statistics.
*     ye : even site pseudo fermion
*     ue : even site gauge link
*     uo : odd  site geuge link
*    ieo : operate even/odd site
*          ieo=0 for even site link
*          ieo=1 for odd  site link
* fclinve : even site inverse clover term
* fclinvo : odd site inverse clover term
*
************************************************************************
      use multiboson
#include "param-size.h"
#include "com-rand_g.h"
      real(8), intent(out) :: action
      type(quark_def), intent(inout) :: quark
      complex(8), intent(inout) :: ye(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1)
      complex(8), intent(in) :: ue(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
      complex(8), intent(in) :: uo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
#ifdef _CLOVER
      complex(8), dimension(CLSP,0:NTH,0:NZ1,0:NY1,0:NX1,CLSP)
     &           :: fclinve,fclinvo
#endif
      integer, intent(in) :: ieo
*************
      integer :: ix,iy,iz,itb,ieoxyz,ic,is
      complex(8) :: ynewe(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1)  ! for heat bath
      common /WORKQHB1/ynewe
      real(8) :: rtmp0

*********************************
*
* Generate gaussian vector y
*
*********************************
      call gauss_y(ynewe,ieo)

*********************************
* calc. Hamiltonian
*********************************
      rtmp0=0.0d0
*POPTION PARALLEL, PSUM (rtmp0)
#ifdef _COALESCING
*POPTION COALESCING (_COALESCING)
#endif
```

```
65        do ix=1,NX
66        do iy=1,NY
67        do iz=1,NZ
68          ieoxyz=mod(ipeo+ix+iy+iz+ieo,2)
69  *VOPTION VEC
70          do itb=1-ieoxyz,NTH-ieoxyz
71  *SOPTION UNROLL (4)
72            do is=1,SPIN
73  *SOPTION UNROLL (3)
74            do ic=1,COL
75              rtmp0=rtmp0+ real(ynewe(ic,is,itb,iz,iy,ix))**2
76       &                 +aimag(ynewe(ic,is,itb,iz,iy,ix))**2
77            enddo
78            enddo
79          enddo
80        enddo
81        enddo
82        enddo
83  #ifndef _singlePU
84        call comlib_sumcast_r8(rtmp0)
85  #endif
86        action=rtmp0
87
88  *********************************
89  *       v
90  *  ye = Dee ynewe
91  *
92  *********************************
93  #ifdef _CLOVER
94        call mult_mb_pre(quark%kappa,ynewe,ye,ue,uo,ieo,fclinve,fclinvo)
95  #else
96        call mult_mb_pre(quark%kappa,ynewe,ye,ue,uo,ieo)
97  #endif
98
99        return
100       end
101
```

引数 (出力) **action :** 作用 $S_q[U, \psi_o^\dagger, \psi_o]$ の値を返す

引数 (入出力) **quark :** ある一つのクォークに対するパラメータを含む構造体

引数 (入出力) **ye :** `ieo=0` の時 even サイトの擬フェルミオン場、`ieo=1` の時 odd サイトの擬フェルミオン場。

引数 (入力) **ue, uo:** `ieo=0` の時 even サイト、odd サイトのゲージ場、`ieo=1` の時 odd サイト、even サイトのゲージ場。

引数 (入力) **ieo :** `ieo=0` の時 even サイトに擬フェルミオン場が存在、`ieo=1` の時 odd サイトに擬フェルミオン場が存在。

引数 **fclinve, fclinvo :** `ieo=0` の時 even サイト、odd サイトのクローバー項の逆 $((1+T)^{-1})$、`ieo=0` の時 even サイト、odd サイトのクローバー項の逆。

55 行目: ガウス分布する場を生成

60-86 行目: 作用を計算

93-97 行目: 擬フェルミオン場を生成。`mult_mb_pre` **は** $\hat{D}_{ee}$ **または** $\hat{D}_{oo}$ **を掛ける。**

2.3.4 HMCLIB/HMCLIB/hamil_hmc.F

作用 $S_q[U, \psi_o^\dagger, \psi_o]$ だけを計算するのが HMCLIB/hamil_hmc.F です。

HMCLIB/hamil_hmc.F

```
1  #include "config.h"
2  #ifdef _CLOVER
3        subroutine hamil_hmc(action,quark,ye,ue,uo,ieo,fclinve,fclinvo)
4  #else
5        subroutine hamil_hmc(action,quark,ye,ue,uo,ieo)
6  #endif
7  ************************************************************************
8  *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
9  * Calculate Hamiltonian of PHMC pseudo fermion :
10 * Action :
11 *
12 *  Shmc = |Dee^-1 ye|
13 *
14 *  action : action
15 *   quark : quark_parameters
16 *      ye : even site pseudo fermion
17 *      ue : even site gauge link
18 *      uo : odd site gauge link
19 *     ieo : operate even/odd site
20 *           ieo=0 for even site link
21 *           ieo=1 for odd  site link
22 * fclinve : even site inverse clover matrix
23 * fclinvo : odd site inverse clover matrix
24 *
25 ************************************************************************
26       use multiboson
27 #include "param-size.h"
28       real(8), intent(out) :: action
29       type(quark_def), intent(inout) :: quark
30       complex(8), intent(in) :: ye(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1)
31       complex(8), intent(in) :: ue(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
32       complex(8), intent(in) :: uo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
33       integer, intent(in) :: ieo
34 #ifdef _CLOVER
35       complex(8), dimension(CLSP,0:NTH,0:NZ1,0:NY1,0:NX1,CLSP)
36      &          , intent(in) :: fclinve,fclinvo
37 #endif
38 **************
39       complex(8) :: we(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1)
40       complex(8) :: re(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1)
41       common /WORKQHB1/ we,re
42       integer :: ieoxyz
43       integer :: ix,iy,iz,itb,is,ic
44       real(8) :: rtmp0,err
45       integer :: iiter
46
47 ***********************
48 * field copy
```

```
**********************
*POPTION PARALLEL
*POPTION TLOCAL (ix,iy,iz,itb,ieoxyz,is,ic)
#ifdef _COALESCING
*POPTION COALESCING (_COALESCING)
#endif
      do ix=1,NX
      do iy=1,NY
      do iz=1,NZ
        ieoxyz=mod(ipeo+ix+iy+iz+ieo,2)
*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (4)
          do is=1,SPIN
*SOPTION UNROLL (3)
          do ic=1,COL
            we(ic,is,itb,iz,iy,ix)=ye(ic,is,itb,iz,iy,ix)
          enddo
          enddo
        enddo
      enddo
      enddo
      enddo ! end of do ix,iy,iz

**********************
*              v
* solve : we = Dee^-1 ye
**********************
      err = quark%amp_bicgstab_err
#ifdef _CLOVER
      call bicgstab_hmc(err,iiter,quark%kappa,re,we,ue,uo,ieo,
     &                  fclinve,fclinvo)
#else
      call bicgstab_hmc(err,iiter,quark%kappa,re,we,ue,uo,ieo)
#endif
      quark%nmp_bicgstab_iter=quark%nmp_bicgstab_iter+iiter
      quark%nmp_hit=quark%nmp_hit+1

**********************
* action = |Dee^-1 ye|^2
**********************
      rtmp0=0.0d0
*POPTION PARALLEL, PSUM (rtmp0)
*POPTION TLOCAL (ix,iy,iz,itb,ieoxyz,is,ic)
#ifdef _COALESCING
*POPTION COALESCING (_COALESCING)
#endif
      do ix=1,NX
      do iy=1,NY
      do iz=1,NZ
        ieoxyz=mod(ipeo+ix+iy+iz+ieo,2)
*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (4)
          do is=1,SPIN
*SOPTION UNROLL (3)
          do ic=1,COL
```

```
106               rtmp0=rtmp0+ real(we(ic,is,itb,iz,iy,ix))**2
107      &                   +aimag(we(ic,is,itb,iz,iy,ix))**2
108            enddo
109            enddo
110          enddo
111        enddo
112        enddo
113        enddo ! end of do ix,iy,iz
114
115  #ifndef _singlePU
116        call comlib_sumcast_r8(rtmp0)
117  #endif
118
119        action=rtmp0
120
121        return
122        end
123
```

引数 : 引数の構造は HMCLIB/updatey_hmc.F と同じ

50-71 行目: 擬フェルミオン場をコピー

77-85 行目: 擬フェルミオン場に $\hat{D}_{ee}$ または $\hat{D}_{oo}$ を掛ける

90-119 行目: 作用 $S_q[U, \psi^\dagger_{e(o)}, \psi_{e(o)}]$ の計算。

2.4 クローバー項の計算

クローバー項の逆の計算はをカイラル表示で行ってそのあとディラック表示に直している。カイラル表示ではガンマ行列が、

$$\begin{aligned}
\gamma_i &= \begin{pmatrix} 0 & -i\sigma_i \\ i\sigma_i & 0 \end{pmatrix}, \\
\gamma_4 &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \\
\gamma_5 &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},
\end{aligned} \tag{7}$$

となり、$\sigma_{\mu\nu}$ は以下のようにブロック対角化される。

$$\begin{aligned}
\sigma_{ij} &= -\varepsilon_{ijk} \begin{pmatrix} \sigma_k & 0 \\ 0 & \sigma_k \end{pmatrix}, \\
\sigma_{i4} &= \begin{pmatrix} \sigma_i & 0 \\ 0 & -\sigma_i \end{pmatrix}.
\end{aligned} \tag{8}$$

これを用いるとクローバー項 $((1+T))$ は、

$$^{\text{chiral}}(1+T)(n,m) = \delta_{n,m} \begin{pmatrix} F_1(n) & 0 \\ 0 & F_2(n) \end{pmatrix}, \tag{9}$$

と表される。ここで、F_1, F_2 は 6×6 行列

$$F_1(n) = 1 - c_{\rm sw}\kappa\left[(-F_{23}(n) + F_{14}(n))\sigma_1 + (F_{13}(n) + F_{24}(n))\sigma_2 + (-F_{12}(n) + F_{34}(n))\sigma_3\right] \quad (10)$$

$$F_2(n) = 1 - c_{\rm sw}\kappa\left[(-F_{23}(n) - F_{14}(n))\sigma_1 + (F_{13}(n) - F_{24}(n))\sigma_2 + (-F_{12}(n) - F_{34}(n))\sigma_3\right], \quad (11)$$

である。

従ってクローバー項の逆は F_1, F_2 の逆を用いて、

$$^{\rm chiral}(1+T)^{-1}(n,m) = \delta_{n,m}\begin{pmatrix} F_1^{-1}(n) & 0 \\ 0 & F_2^{-1}(n) \end{pmatrix}, \quad (12)$$

と表される。

カイラル表示からディラック表示への変換はユニタリー変換により、

$$^{\rm dirac}(1+T)^{-1}(n,m) = \delta_{n,m}\frac{1}{2}\begin{pmatrix} F_1^{-1}(n) + F_2^{-1}(n) & -F_1^{-1}(n) + F_2^{-1}(n) \\ -F_1^{-1}(n) + F_2^{-1}(n) & F_1^{-1}(n) + F_2^{-1}(n) \end{pmatrix}, \quad (13)$$

と表される。

$F_{\mu\nu}(n)$ の計算は、CloverLIB/clover.F で行っている。計算は、式 (6) に従っている。説明は省略する。

$(1+T)$ の逆の計算は、CloverLIB/clvinv.F で行っている。計算は上でのべたようにカイラル表示で逆をとりディラック表示に直している。CloverLIB/clover_f1f2.F が式 (10) に従いカイラル表示での $(1+T)$ の 6×6 ブロック F_1, F_2 を計算している。CloverLIB/clvinv_lu.F が 6×6 ブロック行列の F_1 または F_2 の逆を LU 分解により計算している。このとき同時に固有値も求められている (作用 $S_{\rm det}[U]$ の計算に必要)。CloverLIB/chiral2dirac_clv.F はカイラル表示からディラック表示への変換を式 (13) に従い行っている。

クローバー項の計算を行うプログラムは以下のものです。

CloverLIB/clover.F	クローバー項 $F_{\mu\nu}$ の計算
CloverLIB/clvinv.F	$(1+T)^{-1}$ の計算
CloverLIB/clover_f1f2.F	カイラル表示での $1+T$ の計算
CloverLIB/clvinv_lu.F	カイラル表示での $(1+T)^{-1}$ の計算
CloverLIB/chiral2dirac_clv.F	カイラル表示からディラック表示への変換

2.4.1 CloverLIB/clvinv.F

CloverLIB/clvinv.F

```
1  #include "config.h"
2  #ifdef _CLOVER
3        subroutine clvinv(kappa,csw,fclinve,fclinvo,ucle,uclo,
4       &                  logdetfcl,idet)
5  ************************************************************************
6  *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
7  * Calculate inverse clover term matrix and determiant:
8  *
9  *  fclinv = [ 1 - csw kappa/2 sigma_{mu,nu} F_{mu,nu}]^-1
10 *
11 *  log[det[fcl]]
12 *
13 *   kappa : hopping parameter
14 *     csw : clover coeffcient
```

```
* fclinve : even site inverse clover term matrix
* fclinvo : odd site inverse clover term matrix
*    ucle : even site clover leaf
*    uclo : odd site clover leaf
* logdetfcl : log of determinat of fcl
*    idet : if idet=1 then calc logdetfcl
*
************************************************************************
      use multiboson
#include "param-size.h"
      real(8), intent(in) :: kappa,csw
      complex(8), dimension(CLSP,0:NTH,0:NZ1,0:NY1,0:NX1,CLSP),
     &            intent(out) :: fclinve,fclinvo
      complex(8), intent(in) ::
     &                     ucle(COL,COL,NTH,NZ,NY,NX,NDIM*(NDIM-1)/2),
     &                     uclo(COL,COL,NTH,NZ,NY,NX,NDIM*(NDIM-1)/2)
      real(8), intent(out) :: logdetfcl
      integer, intent(in) :: idet
**** work vector
      complex(8), dimension(CLSP/2,CLSP/2,NTH,NZ,NY,NX) ::
     &            f1cl,f2cl
      complex(8) :: f1clinv(CLSP/2,CLSP/2,NTH,NZ,NY,NX)
      complex(8) :: f2clinv(CLSP/2,CLSP/2,NTH,NZ,NY,NX)
      common /WORKCLVINV/ f1clinv,f2clinv,f1cl,f2cl
      real(8) :: logdetf1cle,logdetf1clo
      real(8) :: logdetf2cle,logdetf2clo

*****************************
* calc clover term matrix in chiral rep. (even part)
*
*  fcl = diag[f1cl,f2cl] = 1 - csw kappa/2 sigma_{mu,nu}F_{mu,nu}
*
* solve inverse matrix (even part)
*
*****************************
      call clover_f1f2(kappa,csw,f1cl,f2cl,ucle)
      call clvinv_lu(f1clinv,f1cl,logdetf1cle,idet)
      call clvinv_lu(f2clinv,f2cl,logdetf2cle,idet)
*****************************
* transform  chiral -> Dirac rep.
*****************************
      call chiral2dirac_clv(f1clinv,f2clinv,fclinve,0)

*****************************
* calc clover term matrix in chiral rep. (odd part)
*
*  fcl = diag[f1cl,f2cl] = 1 - csw kappa/2 sigma_{mu,nu}F_{mu,nu}
*
* solve inverse matrix
*
*****************************
      call clover_f1f2(kappa,csw,f1cl,f2cl,uclo)
      call clvinv_lu(f1clinv,f1cl,logdetf1clo,idet)
      call clvinv_lu(f2clinv,f2cl,logdetf2clo,idet)
*****************************
* transform  chiral -> Dirac rep.
*****************************
```

```
72        call chiral2dirac_clv(f1clinv,f2clinv,fclinvo,1)
73
74        if (idet.EQ.1) then
75          logdetfcl=logdetf1cle+
76       &            logdetf2cle+
77       &            logdetf1clo+
78       &            logdetf2clo
79  #ifndef _singlePU
80          if (NPU.NE.1) call comlib_sumcast_r8(logdetfcl)
81  #endif
82        endif
83
84        return
85        end
86  #else
87        subroutine clvinv
88        return
89        end
90  #endif
91
```

引数 (入力) **kappa, csw :** ホッピングパラメータ、クローバー係数

引数 (出力) **fclinve, fclinvo :** even サイト、odd サイトのクローバー項の逆 $((1+T)^{-1})$ ディラック表示で転置されている

引数 (入力) **ucle, uclo :** even サイト、odd サイトのクローバーリーフ $F_{\mu\nu}$

引数 (出力) **logdetfcl :** `idet=1` の時 $\log\det[1+T]$ を返す

引数 (入力) **idet :** `idet=1` の時 `logdetfcl` に $\log\det[1+T]$ を返す

42-56 行目: even サイトの $(1+T)^{-1}$ を求めている

50 行目: $F_{\mu\nu}$ から F_1, F_2 を求めている

51 行目: F_1 の逆を求めている

52 行目: F_2 の逆を求めている

56 行目: F_1, F_2 の逆から $(1+T)^{-1}$ を求めている

58-72 行目: odd サイトの $(1+T)^{-1}$ を求めている

74-82 行目: `idet=1` の時に $\log\det(1+T)$ を求めている

2.4.2 CloverLIB/clover_f1f2.F

CloverLIB/clover_f1f2.F

```
1  #include "config.h"
2  #ifdef _CLOVER
3        subroutine clover_f1f2(kappa,csw,f1cle,f2cle,ucle)
4  ************************************************************************
5  *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
6  * Calculate clover term matrix in chiral rep.:
7  *
8  *  kappa : hopping parameter
```

```
*    csw : clover coefficient
*  f1cle : even/odd site upper-left component of clover term matrix in chiral rep.
*  f2cle : even/odd site lower-right component of clover term matrix in chiral rep.
*   ucle : even/odd site clover leaf field strength
*
************************************************************************
#include "param-size.h"
      real(8), intent(in) :: kappa,csw
      complex(8), intent(out) :: f1cle(CLSP/2,CLSP/2,NTH*NZ*NY*NX)
      complex(8), intent(out) :: f2cle(CLSP/2,CLSP/2,NTH*NZ*NY*NX)
      complex(8), intent(in)::ucle(COL,COL,NTH*NZ*NY*NX,NDIM*(NDIM-1)/2)
*****
      complex(8) :: zunit(COL,COL),zi
      parameter (zi=(0.0d0,1.0d0))
      real(8) :: cswkappa
      integer :: ixyzt,ic,jc

      cswkappa=csw*kappa

      do jc=1,COL
      do ic=1,COL
        zunit(ic,jc)=(0.0d0,0.0d0)
      enddo
      enddo
      do ic=1,COL
        zunit(ic,ic)=(1.0d0,0.0d0)
      enddo

************************************************************************
*
* f1cl=1-csw*kappa*[(-F23+F14)sig1+(+F13+F24)sig2+(-F12+F34)sig3]
* f2cl=1-csw*kappa*[(-F23-F14)sig1+(+F13-F24)sig2+(-F12-F34)sig3]
*
*  F12 = ucl(1),F13 = ucl(2),F14 = ucl(3)
*  F23 = ucl(4),F24 = ucl(5),F34 = ucl(6)
*
************************************************************************
*POPTION PARALLEL
*VOPTION VEC, PREFETCH (ucle)
      do ixyzt=1,(NTH*NZ*NY*NX)
*SOPTION UNROLL (3)
        do jc=1,COL
*SOPTION UNROLL (3)
        do ic=1,COL
          f1cle(ic    ,jc    ,ixyzt)=-cswkappa*
     &                               (-ucle(ic,jc,ixyzt,1)
     &                                +ucle(ic,jc,ixyzt,6))+zunit(ic,jc)
          f1cle(ic+COL,jc+COL,ixyzt)=-cswkappa*
     &                               (+ucle(ic,jc,ixyzt,1)
     &                                -ucle(ic,jc,ixyzt,6))+zunit(ic,jc)
        enddo
        enddo
      enddo ! end of do ixyzt
*POPTION PARALLEL
*VOPTION VEC, PREFETCH (ucle)
      do ixyzt=1,(NTH*NZ*NY*NX)
*SOPTION UNROLL (3)
```

```
        do jc=1,COL
*SOPTION UNROLL (3)
        do ic=1,COL
          f1cle(ic    ,jc+COL,ixyzt)=-cswkappa*
     &                               (-ucle(ic,jc,ixyzt,4)
     &                                +ucle(ic,jc,ixyzt,3)
     &                              -zi*ucle(ic,jc,ixyzt,2)
     &                              -zi*ucle(ic,jc,ixyzt,5))
          f1cle(ic+COL,jc    ,ixyzt)=-cswkappa*
     &                               (-ucle(ic,jc,ixyzt,4)
     &                                +ucle(ic,jc,ixyzt,3)
     &                              +zi*ucle(ic,jc,ixyzt,2)
     &                              +zi*ucle(ic,jc,ixyzt,5))
        enddo
        enddo
      enddo ! end of do ixyzt

*POPTION PARALLEL
*VOPTION VEC, PREFETCH (ucle)
      do ixyzt=1,(NTH*NZ*NY*NX)
*SOPTION UNROLL (3)
        do jc=1,COL
*SOPTION UNROLL (3)
        do ic=1,COL
          f2cle(ic    ,jc    ,ixyzt)=-cswkappa*
     &                               (-ucle(ic,jc,ixyzt,1)
     &                                -ucle(ic,jc,ixyzt,6))+zunit(ic,jc)
          f2cle(ic+COL,jc+COL,ixyzt)=-cswkappa*
     &                               (+ucle(ic,jc,ixyzt,1)
     &                                +ucle(ic,jc,ixyzt,6))+zunit(ic,jc)
        enddo
        enddo
      enddo ! end of do ixyzt
*POPTION PARALLEL
*VOPTION VEC, PREFETCH (ucle)
      do ixyzt=1,(NTH*NZ*NY*NX)
*SOPTION UNROLL (3)
        do jc=1,COL
*SOPTION UNROLL (3)
        do ic=1,COL
          f2cle(ic    ,jc+COL,ixyzt)=-cswkappa*
     &                               (-ucle(ic,jc,ixyzt,4)
     &                                -ucle(ic,jc,ixyzt,3)
     &                              -zi*ucle(ic,jc,ixyzt,2)
     &                              +zi*ucle(ic,jc,ixyzt,5))
          f2cle(ic+COL,jc    ,ixyzt)=-cswkappa*
     &                               (-ucle(ic,jc,ixyzt,4)
     &                                -ucle(ic,jc,ixyzt,3)
     &                              +zi*ucle(ic,jc,ixyzt,2)
     &                              -zi*ucle(ic,jc,ixyzt,5))
        enddo
        enddo
      enddo ! end of do ixyzt

      return
      end
#else
```

```
123       subroutine clover_f1f2
124       return
125       end
126 #endif
127
```

引数 (入力) **kappa, csw** : ホッピングパラメータ、クローバー係数

引数 (出力) **f1cle, f2cle** : カイラル表示での $(1+T)$ の上成分と下成分 (F_1, F_2)

引数 (入力) **ucle:** クローバーリーフ $F_{\mu\nu}$

46-81 行目: F_1 の計算。スピン (is=1,2) とカラー (ic=1,2,3) は ic+(is-1)*3 の形でパックしてある。

46-61 行目: F_1 のスピン (1,1) と (2,2) 成分の計算

62-81 行目: F_1 のスピン (1,2) と (2,1) 成分の計算

83-118 行目: F_2 の計算。

2.4.3 CloverLIB/clvinv_lu.F

CloverLIB/clvinv_lu.F

```
1  #include "config.h"
2  #ifdef _CLOVER
3        subroutine clvinv_lu(be,Ae,logdetAe,idet)
4  ************************************************************************
5  * $Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
6  * LU decomposition
7  *  Solve:  F X = 1 -> X = F^-1
8  *
9  *      be : even/odd site inverse clover term matrix (output)
10 *      Ae : even/odd site clover term matrix (input)
11 * logdetAe : log determinant of Ae
12 *    idet : if idet = 1 then calc. logdetAe.
13 *
14 ************************************************************************
15 #include "param-size.h"
16       integer, parameter :: NSITE=NTH*NZ*NY*NX
17       complex(8), intent(out) :: be(CLSP/2,CLSP/2,NSITE)
18       complex(8), intent(in) :: Ae(CLSP/2,CLSP/2,NSITE)
19       real(8), intent(out) :: logdetAe
20       integer, intent(in) :: idet
21 ************
22       complex(8) :: AAe(CLSP/2,CLSP/2,NSITE)
23       complex(8) :: zunit(CLSP/2,CLSP/2)
24       complex(8) :: ctmp(CLSP/2,CLSP/2,NSITE)
25       real(8) :: d(NSITE)
26       integer :: iout,icount
27       integer :: indx(CLSP/2,NSITE)
28       integer :: ixyzt,ics,jcs
29
30       iout = 23
31
32       do jcs=1,CLSP/2
33       do ics=1,CLSP/2
```

```
        zunit(ics,jcs)=(0.0d0,0.0d0)
      enddo
      enddo
      do ics=1,CLSP/2
        zunit(ics,ics)=(1.0d0,0.0d0)
      enddo

*****************
* set unit matrix for source b
*****************
*POPTION NOPARALLEL
      do jcs=1,CLSP/2
      do ics=1,CLSP/2
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
        do ixyzt=1,NSITE
           be(ics,jcs,ixyzt)=(0.0d0,0.0d0)
          AAe(ics,jcs,ixyzt)=Ae(ics,jcs,ixyzt)
        enddo
      enddo
      enddo
*POPTION NOPARALLEL
      do ics=1,CLSP/2
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
        do ixyzt=1,NSITE
           be(ics,ics,ixyzt)=(1.0d0,0.0d0)
        enddo
      enddo

      call ludcmp(Ae,indx,d)

************************************
* calculate det[Ae] when idet = 1.
************************************
      if (idet.EQ.1) then
        logdetAe=0.0d0
*POPTION PARALLEL
*VOPTION VEC, PREFETCH (Ae)
*SOPTION NOLOOPEXPAND
        do ixyzt=1,NSITE
*SOPTION UNROLL (6)
          do ics=1,CLSP/2
            d(ixyzt)=d(ixyzt)*dble(Ae(ics,ics,ixyzt))
          enddo
        enddo
*POPTION PARALLEL, PSUM (logdetAe)
*VOPTION VEC, PVFUNC (2)
*SOPTION NOLOOPEXPAND
        do ixyzt=1,NSITE
          logdetAe=logdetAe+log(d(ixyzt))
        enddo
      endif

************************************
```

```
* calculate b = A^-1
************************************
      call lubksb(Ae,indx,be)

************************************
* check  b * A = A^-1 * A = I ?
************************************
*POPTION NOPARALLEL
      do jcs=1,CLSP/2
*POPTION NOPARALLEL
      do ics=1,CLSP/2
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
        do ixyzt=1,NSITE
          ctmp(ics,jcs,ixyzt)=
     &                 AAe(ics,1,ixyzt)*be(1,jcs,ixyzt)
     &                +AAe(ics,2,ixyzt)*be(2,jcs,ixyzt)
        enddo
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
        do ixyzt=1,NSITE
          ctmp(ics,jcs,ixyzt)=ctmp(ics,jcs,ixyzt)
     &                +AAe(ics,3,ixyzt)*be(3,jcs,ixyzt)
     &                +AAe(ics,4,ixyzt)*be(4,jcs,ixyzt)
        enddo
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
        do ixyzt=1,NSITE
          ctmp(ics,jcs,ixyzt)=ctmp(ics,jcs,ixyzt)
     &                +AAe(ics,5,ixyzt)*be(5,jcs,ixyzt)
     &                +AAe(ics,6,ixyzt)*be(6,jcs,ixyzt)
        enddo
       enddo
       enddo
***************
*  ctmp =  A^-1*A - I
***************
*POPTION NOPARALLEL
      do jcs=1,CLSP/2
*POPTION NOPARALLEL
      do ics=1,CLSP/2
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
        do ixyzt=1,NSITE
          ctmp(ics,jcs,ixyzt)=ctmp(ics,jcs,ixyzt)-zunit(ics,jcs)
        enddo
       enddo
       enddo

*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
      do ixyzt=1,NSITE
```

```
148         d(ixyzt)=0.0d0
149       enddo
150 
151 *POPTION NOPARALLEL
152       do jcs=1,CLSP/2
153 *POPTION NOPARALLEL
154       do ics=1,CLSP/2
155 *POPTION PARALLEL
156 *VOPTION VEC
157 *SOPTION NOLOOPEXPAND
158         do ixyzt=1,NSITE
159           d(ixyzt)=d(ixyzt)
160      &            + real(ctmp(ics,jcs,ixyzt))**2
161      &            +aimag(ctmp(ics,jcs,ixyzt))**2
162         enddo
163       enddo
164       enddo
165 
166       icount=0
167 *POPTION PARALLEL, PSUM (icount)
168 *VOPTION VEC
169 *SOPTION NOLOOPEXPAND
170       do ixyzt=1,NSITE
171         if (d(ixyzt).GT.(1.0d-15*(CLSP/2))**2) icount=icount+1
172       enddo
173 
174       if (icount.NE.0) then
175         write(iout,'(" LU decomp. inversion : ",I3)')nodeid
176         write(iout,'("   # of site, (residual > 1.0d-15) :",I4)')icount
177       endif
178 
179       return
180       end
181 #else
182       subroutine clvinv_lu
183       return
184       end
185 #endif
186 
187 #ifdef _CLOVER
188 ************************************************************************
189       subroutine lubksb(a,indx,b)
190 ************************************************************************
191 *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
192 *  Solve A X = B with LU decomposed matrix A using forward(backward)
193 *  substitution. (originally from Numerical Recipes in FORTRAN p.38)
194 *
195 *    a : LU decomposed matrix
196 * indx : row parmutation by partial pivotting for LU decomposed matrix 'a'
197 *    b : source matrix for input, solution matrix for output
198 *
199 ************************************************************************
200 #include "param-size.h"
201       integer, parameter :: NSITE=NTH*NZ*NY*NX
202       integer, intent(in) :: indx(CLSP/2,NSITE)
203       complex(8), intent(in) :: a(CLSP/2,CLSP/2,NSITE)
204       complex(8), intent(inout) :: b(CLSP/2,CLSP/2,NSITE)
```

```
      complex(8) :: ctmp1(NSITE),sum(NSITE)
      integer :: ics,jcs,kcs,ll,ixyzt

*POPTION NOPARALLEL
      do jcs=1,CLSP/2
******************
*POPTION NOPARALLEL
        do ics=1,CLSP/2

*POPTION PARALLEL, TLOCAL (ll)
*VOPTION VEC
*SOPTION NOLOOPEXPAND
          do ixyzt=1,NSITE
            ll=indx(ics,ixyzt)
            sum(ixyzt)=b(ll,jcs,ixyzt)
          enddo
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
          do ixyzt=1,NSITE
            ctmp1(ixyzt)=b(ics,jcs,ixyzt)
          enddo
*POPTION PARALLEL, TLOCAL (ll)
*VOPTION VEC
*SOPTION NOLOOPEXPAND
          do ixyzt=1,NSITE
            ll=indx(ics,ixyzt)
            b(ll,jcs,ixyzt)=ctmp1(ixyzt)
          enddo

*POPTION NOPARALLEL
*VOPTION NOVEC
*SOPTION NOLOOPEXPAND
          do kcs=1,ics-1
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
            do ixyzt=1,NSITE
              sum(ixyzt)=sum(ixyzt)
     &                   -a(ics,kcs,ixyzt)*b(kcs,jcs,ixyzt)
            enddo
          enddo ! end of do kcs

*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
          do ixyzt=1,NSITE
            b(ics,jcs,ixyzt)=sum(ixyzt)
          enddo

******************
        enddo ! end of do ics

*POPTION NOPARALLEL
        do ics=CLSP/2,1,-1

*POPTION PARALLEL
```

```
*VOPTION VEC
*SOPTION NOLOOPEXPAND
          do ixyzt=1,NSITE
            sum(ixyzt)=b(ics,jcs,ixyzt)
          enddo

*VOPTION NOVEC
*SOPTION NOLOOPEXPAND
          do kcs=ics+1,CLSP/2
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
            do ixyzt=1,NSITE
              sum(ixyzt)=sum(ixyzt)
     &                   -a(ics,kcs,ixyzt)*b(kcs,jcs,ixyzt)
            enddo
          enddo ! end of do kcs

*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
          do ixyzt=1,NSITE
            b(ics,jcs,ixyzt)=sum(ixyzt)/a(ics,ics,ixyzt)
          enddo

        enddo ! end of do ics
      enddo ! end of do jcs

      return
      end

************************************************************************
      subroutine ludcmp(a,indx,d)
************************************************************************
*$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
* LU decomposition (originally from Numerical Recipes in FORTRAN p.38)
*
*    a : input matrix to be LU decomposed
* indx : row parmutation by partial pivotting
*    d : row parmutation even-odd parity
*
************************************************************************
#include "param-size.h"
      integer, parameter :: NSITE=NTH*NZ*NY*NX
      complex(8), intent(inout) :: a(CLSP/2,CLSP/2,NSITE)
      integer, intent(out) :: indx(CLSP/2,NSITE)
      real(8), intent(out) :: d(NSITE)
*******************
      complex(8) :: vv(CLSP/2,NSITE),sum(NSITE),dum(NSITE),ctmp1(NSITE)
      real(8) :: aamax(NSITE),rtmp1(NSITE),rtmp
      integer :: imax(NSITE)
      integer :: ixyzt,ics,jcs,kcs

*POPTION PARALLEL
*VOPTION VEC
      do ixyzt=1,NSITE
        d(ixyzt)=1.0d0
```

```
      enddo

*POPTION NOPARALLEL
      do ics=1,CLSP/2

*POPTION PARALLEL
*VOPTION VEC
        do ixyzt=1,NSITE
          aamax(ixyzt)=0.0d0
        enddo

*POPTION NOPARALLEL
        do jcs=1,CLSP/2

*POPTION PARALLEL, TLOCAL (rtmp)
*VOPTION VEC, PVFUNC (2)
*SOPTION NOLOOPEXPAND
          do ixyzt=1,NSITE
            rtmp=conjg(a(ics,jcs,ixyzt))*a(ics,jcs,ixyzt)
            rtmp1(ixyzt)=dsqrt(rtmp)
          enddo
*POPTION PARALLEL
*VOPTION VEC, PVFUNC (2)
*SOPTION NOLOOPEXPAND, PREDICATE
          do ixyzt=1,NSITE
            if (rtmp1(ixyzt).GT.aamax(ixyzt))
     &        aamax(ixyzt)=rtmp1(ixyzt)
          enddo
        enddo ! end of do jcs

*POPTION PARALLEL
*VOPTION VEC, PVFUNC(2)
*SOPTION NOLOOPEXPAND
        do ixyzt=1,NSITE
          vv(ics,ixyzt)=1.0d0/aamax(ixyzt)
        enddo

      enddo ! end of do ics

*******
*POPTION NOPARALLEL
      do jcs=1,CLSP/2

*POPTION NOPARALLEL
        do ics=1,jcs-1

*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
          do ixyzt=1,NSITE
            sum(ixyzt)=a(ics,jcs,ixyzt)
          enddo

*VOPTION NOVEC
*SOPTION NOLOOPEXPAND
          do kcs=1,ics-1
*POPTION PARALLEL
```

```
376 *VOPTION VEC
377 *SOPTION NOLOOPEXPAND
378             do ixyzt=1,NSITE
379               sum(ixyzt)=sum(ixyzt)
380      &                  -a(ics,kcs,ixyzt)*a(kcs,jcs,ixyzt)
381             enddo
382           enddo ! end of do kcs
383
384 *POPTION PARALLEL
385 *VOPTION VEC
386 *SOPTION NOLOOPEXPAND
387           do ixyzt=1,NSITE
388             a(ics,jcs,ixyzt)=sum(ixyzt)
389           enddo
390
391         enddo ! end of do ics
392
393 *POPTION PARALLEL
394 *VOPTION VEC
395 *SOPTION NOLOOPEXPAND
396         do ixyzt=1,NSITE
397           aamax(ixyzt)=0.0d0
398         enddo
399
400 *POPTION NOPARALLEL
401         do ics=jcs,CLSP/2
402 *POPTION PARALLEL
403 *VOPTION VEC
404 *SOPTION NOLOOPEXPAND
405           do ixyzt=1,NSITE
406             sum(ixyzt)=a(ics,jcs,ixyzt)
407           enddo
408
409 *VOPTION NOVEC
410 *SOPTION NOLOOPEXPAND
411           do kcs=1,jcs-1
412 *POPTION PARALLEL
413 *VOPTION VEC
414 *SOPTION NOLOOPEXPAND
415             do ixyzt=1,NSITE
416               sum(ixyzt)=sum(ixyzt)
417      &                  -a(ics,kcs,ixyzt)*a(kcs,jcs,ixyzt)
418             enddo
419           enddo ! end of do kcs
420
421 *POPTION PARALLEL, TLOCAL (rtmp)
422 *VOPTION VEC, PVFUNC (2)
423 *SOPTION NOLOOPEXPAND
424           do ixyzt=1,NSITE
425             a(ics,jcs,ixyzt)=sum(ixyzt)
426             rtmp=conjg(sum(ixyzt))*sum(ixyzt)
427             dum(ixyzt)=vv(ics,ixyzt)*sqrt(rtmp)
428           enddo
429
430 *POPTION PARALLEL, TLOCAL (rtmp)
431 *VOPTION VEC, PVFUNC (2)
432 *SOPTION NOLOOPEXPAND
```

```
          do ixyzt=1,NSITE
            rtmp=conjg(dum(ixyzt))*dum(ixyzt)
            rtmp1(ixyzt)=sqrt(rtmp)
          enddo
*POPTION PARALLEL
*VOPTION VEC, PVFUNC (2)
*SOPTION NOLOOPEXPAND, PREDICATE
          do ixyzt=1,NSITE
            if (rtmp1(ixyzt).GE.aamax(ixyzt)) then
              imax(ixyzt)=ics
              aamax(ixyzt)=dum(ixyzt)
            endif
          enddo

        enddo ! end of do ics

*POPTION NOPARALLEL
*VOPTION NOVEC
*SOPTION NOLOOPEXPAND
        do kcs=1,CLSP/2

*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
          do ixyzt=1,NSITE
            dum(ixyzt)=a(imax(ixyzt),kcs,ixyzt)
          enddo
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
          do ixyzt=1,NSITE
            ctmp1(ixyzt)=a(jcs,kcs,ixyzt)
          enddo
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
          do ixyzt=1,NSITE
            a(imax(ixyzt),kcs,ixyzt)=ctmp1(ixyzt)
          enddo
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
          do ixyzt=1,NSITE
            a(jcs,kcs,ixyzt)=dum(ixyzt)
          enddo
        enddo ! end of do kcs

*POPTION PARALLEL
*VOPTION VEC, PVFUNC (2)
*SOPTION NOLOOPEXPAND, PREDICATE
        do ixyzt=1,NSITE
          if (jcs.NE.imax(ixyzt))d(ixyzt)=-d(ixyzt)
        enddo
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
        do ixyzt=1,NSITE
```

```
          ctmp1(ixyzt)=vv(jcs,ixyzt)
        enddo
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
        do ixyzt=1,NSITE
          vv(imax(ixyzt),ixyzt)=ctmp1(ixyzt)
        enddo

*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
        do ixyzt=1,NSITE
          indx(jcs,ixyzt)=imax(ixyzt)
        enddo

        if(jcs.NE.CLSP/2)then
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
          do ixyzt=1,NSITE
            dum(ixyzt)=1.0d0/a(jcs,jcs,ixyzt)
          enddo
*POPTION NOPARALLEL
*VOPTION NOVEC
*SOPTION NOLOOPEXPAND
          do ics=jcs+1,CLSP/2
*POPTION PARALLEL
*VOPTION VEC
*SOPTION NOLOOPEXPAND
            do ixyzt=1,NSITE
              a(ics,jcs,ixyzt)=a(ics,jcs,ixyzt)*dum(ixyzt)
            enddo
          enddo ! end of do ics
        endif

      enddo ! end of do jcs
*******

      return
      end
#endif

```

引数 (出力) **be :** be(6,6,NSITE) の配列に Ae の 6×6 の行列としての逆を返す。

引数 (入力) **Ae :** Ae(6,6,NSITE) の配列に $F_1(n), F_2(n)$ がセットされる。

引数 (出力) **logdetAe :** `idet=1` のとき Ae(6,6,NSITE) の $\log\det Ae$ が返る。

引数 (入力) **idet :** `idet=1` のとき Ae(6,6,NSITE) の $\log\det Ae$ を返す。

中身 : は LU 分解を用いて 6×6 行列の逆と行列式を求めている。

2.4.4 CloverLIB/chiral2dirac_clv.F

CloverLIB/chiral2dirac_clv.F

```
#include "config.h"
#ifdef _CLOVER
      subroutine chiral2dirac_clv(f1cl,f2cl,fcl,ieo)
************************************************************************
*$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
* Transform clover term matrix from chiral representation
* to Dirac representation
*
*  f1cl : even/odd site inverse clover term matrix in chiral rep. (upper)
*  f2cl : even/odd site inverse clover term matrix in chiral rep. (lower)
*   fcl : even/odd site inverse clover term matrix in Dirac rep. (transposed !x!)
*   ieo : oparete even/odd site vector
*           ieo=0 for even site
*           ieo=1 for odd site
*
*  !x! fcl = [1-csw/2 kappa sigma_{mu,nu} F_{mu,nu}]^T
*  Reason: SR8000 pseudo vectorization works well, when multiply fcl on
*          bosonic vector.
*
************************************************************************
#include "param-size.h"
      complex(8), intent(out) ::  fcl(CLSP,0:NTH,0:NZ1,0:NY1,0:NX1,CLSP)
      complex(8), intent(in)  :: f1cl(CLSP/2,CLSP/2,NTH,NZ,NY,NX)
      complex(8), intent(in)  :: f2cl(CLSP/2,CLSP/2,NTH,NZ,NY,NX)
      integer, intent(in) :: ieo
**** work vector
      complex(8) :: f1111(COL,COL,0:NTH),f1112(COL,COL,0:NTH)
      complex(8) :: f1121(COL,COL,0:NTH),f1122(COL,COL,0:NTH)
      complex(8) :: f1211(COL,COL,0:NTH),f1212(COL,COL,0:NTH)
      complex(8) :: f1221(COL,COL,0:NTH),f1222(COL,COL,0:NTH)
      integer :: ix,iy,iz,itb,itb0,ieoxyz,ic,jc


*****************************************
* transform  chiral -> Dirac rep.
*
*           / f1cl   0  \           1 / 1  1 \
*   fcl = S |           | S^+, S = ---|      |
*           \  0   f2cl /         \/2 \-1  1 /
*
*          1 / f1cl+f2cl  f2cl-f1cl \
*   fcl = ---|                       |
*          2 \ f2cl-f1cl  f1cl+f2cl /
*
*****************************************
*POPTION PARALLEL
*POPTION TLOCAL (ix,iy,iz,itb,ieoxyz,ic,jc,itb0)
*POPTION TLOCAL (f1111,f1112,f1121,f1122)
*POPTION TLOCAL (f1211,f1212,f1221,f1222)
#ifdef _COALESCING
*POPTION COALESCING (_COALESCING)
#endif
      do ix=1,NX
      do iy=1,NY
      do iz=1,NZ
```

```
        ieoxyz=mod(ipeo+ix+iy+iz+ieo,2)

*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
          itb0=itb+ieoxyz
*SOPTION UNROLL (3)
          do jc=1,COL
*SOPTION UNROLL (3)
          do ic=1,COL
            f1111(ic,jc,itb)=                                   ! (f1+f2)_11
     &                      ( f1cl(ic    ,jc    ,itb0,iz,iy,ix)
     &                       +f2cl(ic    ,jc    ,itb0,iz,iy,ix))
            f1112(ic,jc,itb)=                                   ! (f1+f2)_12
     &                      ( f1cl(ic    ,jc+COL,itb0,iz,iy,ix)
     &                       +f2cl(ic    ,jc+COL,itb0,iz,iy,ix))
            f1211(ic,jc,itb)=                                   ! (f2-f1)_11
     &                      ( f2cl(ic    ,jc    ,itb0,iz,iy,ix)
     &                       -f1cl(ic    ,jc    ,itb0,iz,iy,ix))
            f1212(ic,jc,itb)=                                   ! (f2-f1)_12
     &                      ( f2cl(ic    ,jc+COL,itb0,iz,iy,ix)
     &                       -f1cl(ic    ,jc+COL,itb0,iz,iy,ix))
          enddo
          enddo
        enddo

*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
          itb0=itb+ieoxyz
*SOPTION UNROLL (3)
          do jc=1,COL
*SOPTION UNROLL (3)
          do ic=1,COL
            f1121(ic,jc,itb)=                                   ! (f1+f2)_21
     &                      ( f1cl(ic+COL,jc    ,itb0,iz,iy,ix)
     &                       +f2cl(ic+COL,jc    ,itb0,iz,iy,ix))
            f1122(ic,jc,itb)=                                   ! (f1+f2)_22
     &                      ( f1cl(ic+COL,jc+COL,itb0,iz,iy,ix)
     &                       +f2cl(ic+COL,jc+COL,itb0,iz,iy,ix))
            f1221(ic,jc,itb)=                                   ! (f2-f1)_21
     &                      ( f2cl(ic+COL,jc    ,itb0,iz,iy,ix)
     &                       -f1cl(ic+COL,jc    ,itb0,iz,iy,ix))
            f1222(ic,jc,itb)=                                   ! (f2-f1)_22
     &                      ( f2cl(ic+COL,jc+COL,itb0,iz,iy,ix)
     &                       -f1cl(ic+COL,jc+COL,itb0,iz,iy,ix))
          enddo
          enddo
        enddo

*VOPTION VEC
*SOPTION NOLOOPINTERCHANGE
        do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
          do ic=1,COL
*SOPTION UNROLL (3)
          do jc=1,COL
            fcl(jc      ,itb,iz,iy,ix,ic      )=f1111(ic,jc,itb)*0.5d0
            fcl(jc+  COL,itb,iz,iy,ix,ic      )=f1112(ic,jc,itb)*0.5d0
```

```
            fcl(jc+2*COL,itb,iz,iy,ix,ic      )=f1211(ic,jc,itb)*0.5d0
            fcl(jc+3*COL,itb,iz,iy,ix,ic      )=f1212(ic,jc,itb)*0.5d0

            fcl(jc      ,itb,iz,iy,ix,ic+2*COL)=f1211(ic,jc,itb)*0.5d0
            fcl(jc+  COL,itb,iz,iy,ix,ic+2*COL)=f1212(ic,jc,itb)*0.5d0
            fcl(jc+2*COL,itb,iz,iy,ix,ic+2*COL)=f1111(ic,jc,itb)*0.5d0
            fcl(jc+3*COL,itb,iz,iy,ix,ic+2*COL)=f1112(ic,jc,itb)*0.5d0
          enddo
          enddo
        enddo

*VOPTION VEC
*SOPTION NOLOOPINTERCHANGE
        do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
          do ic=1,COL
*SOPTION UNROLL (3)
          do jc=1,COL
            fcl(jc      ,itb,iz,iy,ix,ic+  COL)=f1121(ic,jc,itb)*0.5d0
            fcl(jc+  COL,itb,iz,iy,ix,ic+  COL)=f1122(ic,jc,itb)*0.5d0
            fcl(jc+2*COL,itb,iz,iy,ix,ic+  COL)=f1221(ic,jc,itb)*0.5d0
            fcl(jc+3*COL,itb,iz,iy,ix,ic+  COL)=f1222(ic,jc,itb)*0.5d0

            fcl(jc      ,itb,iz,iy,ix,ic+3*COL)=f1221(ic,jc,itb)*0.5d0
            fcl(jc+  COL,itb,iz,iy,ix,ic+3*COL)=f1222(ic,jc,itb)*0.5d0
            fcl(jc+2*COL,itb,iz,iy,ix,ic+3*COL)=f1121(ic,jc,itb)*0.5d0
            fcl(jc+3*COL,itb,iz,iy,ix,ic+3*COL)=f1122(ic,jc,itb)*0.5d0
          enddo
          enddo
        enddo
      enddo
      enddo
      enddo ! end of do ix,iy,iz

      return
      end
#else
      subroutine chiral2dirac_clv
      return
      end
#endif
```

引数 (入力) **f1cl**： カイラル表示での上成分 (6×6) 行列 (F_1 や F_1^{-1})

引数 (入力) **f2cl**： カイラル表示での下成分 (6×6) 行列 (F_2 や F_2^{-1})

引数 (出力) **fcl**： ディラック表示でのクローバー項 (12×12) 行列。転置されている。カラー (ic=1,2,3) とスピン (is=1,2,3,4) は ic+(is-1)*3 としてパックされている。

引数 (入力) **ieo**： ieo=0 の時 even サイトとして計算する。ieo=1 の時 odd サイトとして計算する。

中身： は式 (13) に従う。

2.5 分子動力学

HMC の分子動力学部分は、式 (6) に従っている。運動方程式は、

$$\dot{U}_\mu(n) = iP_\mu(n)U_\mu(n), \tag{14}$$
$$\dot{P}_\mu(n) = -i\left[(U_\mu(n)F_\mu(n)) - (U_\mu(n)F_\mu(n))^\dagger\right], \tag{15}$$

を使っている (CP-PACS code は式 (14) や (15) で P_μ と U_μ が後ろに来ているようだ)。ここで、$\dot{}$ は分子動力学時間 t に関する全微分、$F_\mu(n)$ は、

$$\dot{S}[U] = \sum_\mu \mathrm{Tr}\left[iP_\mu(n)\left(U_\mu(n)F_\mu(n) - (U_\mu(n)F_\mu(n))^\dagger\right)\right] \tag{16}$$

で定義される。式 (14)、(15) の積分は一番簡単な 1 段の $O(dt^3)$ の誤差のある Leapfrog 積分法で行っている。

一番簡単な 1 段の $O(dt^3)$ の誤差のある Leapfrog 積分法の計算ステップは時間を N_{MD} ステップに分て 1 ステップを dt 進むとすると、

1. $U_\mu(n)_{t=0}$ から $U_\mu(n)_{t=\frac{1}{2}dt}$ を得る。
2. $P_\mu(n)_{t=0}$ と $U_\mu(n)_{t=\frac{1}{2}dt}$ から $P_\mu(n)_{t=dt}$ を得る。
3. $P_\mu(n)_{t=dt}$ と $U_\mu(n)_{t=\frac{1}{2}dt}$ から $U_\mu(n)_{t=\frac{3}{2}dt}$ を得る。
4. $P_\mu(n)_{t=mdt}$ と $U_\mu(n)_{t=(m+\frac{1}{2})dt}$ から $P_\mu(n)_{t=(m+1)dt}$ を得る。
5. $P_\mu(n)_{t=(m+1)dt}$ と $U_\mu(n)_{t=(m+\frac{1}{2})dt}$ から $U_\mu(n)_{t=(m+1+\frac{1}{2})dt}$ を得る。
6. 4. と 5. を適当な回数繰り返す
7. 最後は、$P_\mu(n)_{t=N_{\mathrm{MD}}dt}$ と $U_\mu(n)_{t=(N_{\mathrm{MD}}-1+\frac{1}{2})dt}$ から $U_\mu(n)_{t=N_{\mathrm{MD}}dt}$ を得る。

2.5.1 PHMC/molecdyn_phmc.F

以下に、Leapfrog 積分を行っている PHMC/molecdyn_phmc.F を示す。

PHMC/molecdyn_phmc.F

```
1  #include "config.h"
2  #define _SimpleInteg
3        subroutine molecdyn_phmc(jobs,pe,po,ue,uo,quark_all,ye,yo)
4  ************************************************************************
5  *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
6  * Molecular dynamics, U(1/2)P(1)U(1/2) type (HMC/PHMC)
7  *
8  *   jobs : job parameters
9  *     pe : even site canonical momentum of gauge link
10 *     po : odd site canonical momentum of gauge link
11 *     ue : even site gauge link
12 *     uo : odd site gauge link
13 * quark_all : all quark parameters
14 *     ye : even site boson fields
15 *     yo : odd site boson fields
16 *
17 ************************************************************************
18       use multiboson
```

```
#include "param-size.h"
      type(jobinfo_def), intent(inout) :: jobs
      complex(8), intent(inout) :: pe(COL,COL,0:NTH,NZ,NY,NX,NDIM)
      complex(8), intent(inout) :: po(COL,COL,0:NTH,NZ,NY,NX,NDIM)
      complex(8), intent(inout) ::
     &                       ue(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM),
     &                       uo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
      type(quarks_def), intent(in) :: quark_all
      complex(8), intent(in) ::
     &             ye(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1,quark_all%nmb),
     &             yo(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1,quark_all%nmb)
***********
      real(8) :: dt,dt2,beta
#ifdef _RGGAUGE
      real(8) :: cRG(0:3)
#endif
      integer :: iiter,niter,ibsn
#ifndef _SimpleInteg
      real(8) :: sigma,alpha
#endif

#ifdef _RGGAUGE
      cRG=jobs%cRG
  #define _BETA_ beta,cRG
#else
  #define _BETA_ beta
#endif
      beta=jobs%beta
      dt=jobs%dt
      niter=jobs%niter
      ibsn=jobs%ibsn

      dt2=dt/2.0d0

*****************************************************************
#ifdef _SimpleInteg
#ifdef _PUP
      call updatep_phmc(_BETA_,dt2,pe,po,ue,uo,
     &                  quark_all,ye,yo,ibsn)
      do iiter = 1,niter-1
        call updateu_hmc(dt,pe,ue,0)
        call updateu_hmc(dt,po,uo,1)
        call updatep_phmc(_BETA_,dt,pe,po,ue,uo,
     &                    quark_all,ye,yo,ibsn)
      enddo
      call updateu_hmc(dt,pe,ue,0)
      call updateu_hmc(dt,po,uo,1)
      call updatep_phmc(_BETA_,dt2,pe,po,ue,uo,
     &                  quark_all,ye,yo,ibsn)
#else
      call updateu_hmc(dt2,pe,ue,0)
      call updateu_hmc(dt2,po,uo,1)
      do iiter = 1,niter-1
        call updatep_phmc(_BETA_,dt,pe,po,ue,uo,
     &                    quark_all,ye,yo,ibsn)
        call updateu_hmc(dt,pe,ue,0)
        call updateu_hmc(dt,po,uo,1)
```

```
76        enddo
77        call updatep_phmc(_BETA_,dt,pe,po,ue,uo,
78       &                  quark_all,ye,yo,ibsn)
79        call updateu_hmc(dt2,pe,ue,0)
80        call updateu_hmc(dt2,po,uo,1)
81  #endif
82  ******************************************
83  #else
84        sigma=2.0d0**(1.0d0/3.0d0)
85        alpha=2.0d0-sigma
86        do iiter = 1,niter
87          call updatep_phmc(_BETA_,dt2/alpha,pe,po,ue,uo,
88       &                      quark_all,ye,yo,ibsn)
89          call updateu_hmc(dt/alpha,pe,ue,0)
90          call updateu_hmc(dt/alpha,po,uo,1)
91          call updatep_phmc(_BETA_,(1.0d0-sigma)*dt2/alpha,
92       &                      pe,po,ue,uo,
93       &                    quark_all,ye,yo,ibsn)
94          call updateu_hmc(-sigma*dt/alpha,pe,ue,0)
95          call updateu_hmc(-sigma*dt/alpha,po,uo,1)
96          call updatep_phmc(_BETA_,(1.0d0-sigma)*dt2/alpha,
97       &                      pe,po,ue,uo,
98       &                    quark_all,ye,yo,ibsn)
99          call updateu_hmc(dt/alpha,pe,ue,0)
100         call updateu_hmc(dt/alpha,po,uo,1)
101         call updatep_phmc(_BETA_,dt2/alpha,pe,po,ue,uo,
102      &                      quark_all,ye,yo,ibsn)
103       enddo
104 #endif
105
106       call reunit(ue)
107       call reunit(uo)
108
109       return
110       end
111
```

引数 (入出力) **jobs** : プログラムパラメータを含む構造体

引数 (入出力) **pe, po** : even サイト、 odd サイトのリンク変数に対する運動量 $P_\mu(n)$

引数 (入出力) **ue, uo** : even サイト、 odd サイトのリンク変数 $U_\mu(n)$

引数 (入力) **quark_all** : 全クォークに対するパラメータを含む構造体

引数 (入力) **ye, yo** : even サイト、 odd サイトの擬フェルミオン場を含む作業配列

40-49 行目 : パラメータのコピー

53-52 行目 : 運動方程式 (14)、(15) の数値積分

56-57 行目 : $P_\mu(n)$ から先に発展させる $O(dt^3)$ の誤差のある Leapfrog 積分

69-80 行目 : $U_\mu(n)$ から先に発展させる $O(dt^3)$ の誤差のある Leapfrog 積分

84-103 行目 : $P_\mu(n)$ から先に発展させる $O(dt^4)$ の誤差のある Leapfrog 積分

106-107 行目 : $U_\mu(n)$ の再ユニタリー行列化

ここで、updateu_hmc、updatep_phmc は以下のプログラムで実行されています。

HMCLIB/updateu_hmc.F	式 (14) に従い $U_\mu(n)$ を発展させる
PHMC/updatep_phmc.F	式 (15) に従い $P_\mu(n)$ を発展させる

です。

2.5.2 HMCLIB/updateu_hmc.F

$U_\mu(n)$ の時間発展は式 (14) に従い、

$$U_\mu(n)_{t+dt} = e^{idtP_\mu(n)_{t'}} U_\mu(n)_t, \tag{17}$$

の形で行っている。ここで、入力となる $U_\mu(n), P_\mu(n)$ の分子動力学時刻は Leapfrog 積分の方法によりそれぞれ異なっている (いまは、形式的に t, t' とした)。

su(3)(リー代数) の元である $P_\mu(n)$ の指数関数 $e^{idtP_\mu(n)}$ は以下のようにして求める。いま su(3) の 3×3 行列 A の指数関数 e^{iA} を考える。A は $A^\dagger = A$、$\mathrm{Tr}[A] = 0$ を満たし、さらに Cayley-Hamilton の定理により

$$A^3 = \left(\frac{1}{2}\mathrm{Tr}[A^2]\right) A + \left(\frac{1}{3}\mathrm{Tr}[A^3]\right) 1, \tag{18}$$

を満たす。したがって、e^{iA} は

$$e^{iA} = \sum_{k=0}^{\infty} \frac{1}{k!} (iA)^k = a_0 1 + a_1 A + a_2 A^2, \tag{19}$$

と表される。ここで、a_0, a_1, a_2 は $\mathrm{Tr}[A^2]/2, \mathrm{Tr}[A^3]/3$ の関数である。実際のプログラムでは、12 次のテイラー展開から a_0, a_1, a_2 を求めて使っている。また、12 次のテイラー展開ではユニタリー性が悪い場合には $P_\mu(n)$ の固有値、固有ベクトルを求めて指数関数を評価している。詳細は文献 [1] を見てください。

以下に HMCLIB/updateu_hmc.F を示す。

HMCLIB/updateu_hmc.F

```
1  #include "config.h"
2        subroutine updateu_hmc(dt,pe,ue,ieo)
3  ************************************************************************
4  *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
5  * Update gauge link with canonical momentum :
6  *
7  *    u_mu(dt)=exp(i p_mu(dt')dt) u_mu(0)
8  *
9  *    dt : delta t time step for MD
10 *    pe : even/odd site cononical momentum of gauge link
11 *    ue : even/odd site gauge link
12 *   ieo : operate even/odd site
13 *           ieo=0 for even site
14 *           ieo=1 for odd site
15 *
16 ************************************************************************
17 #include "param-size.h"
18       real(8), intent(in) :: dt
19       complex(8), intent(inout) ::
20      &                          ue(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
```

```
      complex(8), intent(in) :: pe(COL,COL,0:NTH,NZ,NY,NX,NDIM)
      integer, intent(in) :: ieo
******
      complex(8) :: W(COL,COL,0:NTH),W2(COL,COL,0:NTH),coef(COL,0:NTH)
      complex(8) :: expP(COL,COL,0:NTH),expfac(COL,0:NTH)
      real(8) :: delta(COL,0:NTH),det(0:NTH)
      real(8) :: t2(0:NTH),t3(0:NTH),fail(0:NTH),res(0:NTH)
      real(8) :: lm(COL,0:NTH),th(0:NTH),uf(0:NTH),gm(0:NTH)
      real(8) :: runit(COL,COL)
      complex(8) :: zunit(COL,COL,0:NTH)
      complex(8), parameter :: zi=(0.0d0,1.0d0)
      integer :: ix,iy,iz,itb,ieoxyz,mu,ic,jc,iflag

      runit(1,1)=1.0d0
      runit(2,1)=0.0d0
      runit(3,1)=0.0d0
      runit(1,2)=0.0d0
      runit(2,2)=1.0d0
      runit(3,2)=0.0d0
      runit(1,3)=0.0d0
      runit(2,3)=0.0d0
      runit(3,3)=1.0d0

************************************************************
* Solve eigen values and vectors of canonical conjugate
* momentum p (SU(3)).
************************************************************
*POPTION NOPARALLEL
      do mu=1,NDIM
*POPTION PARALLEL
*POPTION TLOCAL (ix,iy,iz,itb,ieoxyz,ic,jc,iflag)
*POPTION TLOCAL (W,W2,expP,expfac,coef,delta,det)
*POPTION TLOCAL (lm,th,uf,gm,t2,t3,zunit,fail)
#ifdef _COALESCING
*POPTION COALESCING (_COALESCING)
#endif
      do ix=1,NX
      do iy=1,NY
      do iz=1,NZ
        ieoxyz=mod(ipeo+ix+iy+iz+ieo,2)

*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
          do jc = 1,COL
*SOPTION UNROLL (3)
          do ic = 1,COL
            W(ic,jc,itb)=dt*pe(ic,jc,itb,iz,iy,ix,mu)
          enddo
          enddo
        enddo
*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
          fail(itb)=1.0d0
        enddo
        iflag=1

```

```
78  ****************************
79  * W2 = W^2
80  ****************************
81  *VOPTION VEC
82          do itb=1-ieoxyz,NTH-ieoxyz
83  *SOPTION UNROLL (3)
84            do jc = 1,COL
85  *SOPTION UNROLL (3)
86            do ic = 1,COL
87              W2(ic,jc,itb)= W(ic,1,itb)*W(1,jc,itb)
88       &                     +W(ic,2,itb)*W(2,jc,itb)
89       &                     +W(ic,3,itb)*W(3,jc,itb)
90            enddo
91            enddo
92          enddo
93
94  ****************************
95  * tr[W^2]/2
96  ****************************
97  *VOPTION VEC, PREFETCH (W2)
98  *SOPTION NOLOOPEXPAND
99          do itb=1-ieoxyz,NTH-ieoxyz
100           t2(itb) = real(W2(1,1,itb)
101      &                  +W2(2,2,itb)
102      &                  +W2(3,3,itb))/2.0d0
103         enddo
104 ****************************
105 * tr[W^3]/3
106 ****************************
107 *VOPTION VEC, PREFETCH (W)
108 *SOPTION NOLOOPEXPAND
109         do itb=1-ieoxyz,NTH-ieoxyz
110           t3(itb) = ( real(W(1,1,itb))**3
111      &               +real(W(2,2,itb))**3
112      &               +real(W(3,3,itb))**3 )/3.0d0
113      &    +( real(W(1,1,itb))*( real(W(1,2,itb))**2+aimag(W(1,2,itb))**2
114      &                         +real(W(1,3,itb))**2+aimag(W(1,3,itb))**2 )
115      &      +real(W(2,2,itb))*( real(W(1,2,itb))**2+aimag(W(1,2,itb))**2
116      &                         +real(W(2,3,itb))**2+aimag(W(2,3,itb))**2 )
117      &      +real(W(3,3,itb))*( real(W(1,3,itb))**2+aimag(W(1,3,itb))**2
118      &                         +real(W(2,3,itb))**2+aimag(W(2,3,itb))**2 )
119      &      +real(W(1,2,itb)*W(2,3,itb)*W(3,1,itb))*2.0d0 )
120         enddo
121
122 ************************************************
123 * calc. exp(i W) with 12th order Taylor expansion
124 ************************************************
125 *VOPTION VEC
126 *SOPTION NOLOOPEXPAND
127         do itb=1-ieoxyz,NTH-ieoxyz
128         coef(1,itb)=2.0876756987868099d-9*(-1.32d2*t2(itb)**4+t2(itb)**5
129      &  +2.4d1*t2(itb)**3*(4.95d2-(0,2.d0)*t3(itb))-3.96d2*t2(itb)*(-5.0
130      &  4d4+(0,4.8d2)*t3(itb)+t3(itb)**2)+6.d0*t2(itb)**2*(-1.1088d5+(0,
131      &  6.6d2)*t3(itb)+t3(itb)**2)+1.2d1*(-1.99584d7+(0,3.3264d5)*t3(itb
132      &  )+9.9d2*t3(itb)**2-(0,1.d0)*t3(itb)**3))
133         coef(2,itb)=(0,-2.0876756987868099d-9)*(1.2d1*t2(itb)**5+5.28d2*
134      &  t2(itb)**3*(1.8d2-(0,1.d0)*t3(itb))+(0,5.d0)*t2(itb)**4*((0,2.64
```

```
     &  d2)+t3(itb))+7.2d1*t2(itb)**2*(-5.544d4+(0,4.95d2)*t3(itb)+t3(it
     &  b)**2)+t2(itb)*(7.98336d7-(0,1.33056d6)*t3(itb)-3.96d3*t3(itb)**
     &  2+(0,4.d0)*t3(itb)**3)-(0,1.32d2)*((0,-3.6288d6)-1.512d5*t3(itb)
     &  +(0,7.2d2)*t3(itb)**2+t3(itb)**3))
        coef(3,itb)=2.0876756987868099d-9*(4.790016d8-(0,1.2d1)*(6.6528d
     &  6-3.3264d5*t2(itb)+7.92d3*t2(itb)**2-1.1d2*t2(itb)**3+t2(itb)**4
     &  )*t3(itb)+4.d0*(-1.6632d5+5.94d3*t2(itb)-9.9d1*t2(itb)**2+t2(itb
     &  )**3)*t3(itb)**2-(0,1.2d1)*(-1.1d2+3.d0*t2(itb))*t3(itb)**3+t3(i
     &  tb)**4)
        enddo

****************************
* calc exp(i W) with Cayley-Hamilton theorem
****************************
*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
          expP(1,1,itb)=coef(1,itb)*W2(1,1,itb)
     &                 +coef(2,itb)* W(1,1,itb)+coef(3,itb)
          expP(2,1,itb)=coef(1,itb)*W2(2,1,itb)
     &                 +coef(2,itb)* W(2,1,itb)
          expP(3,1,itb)=coef(1,itb)*W2(3,1,itb)
     &                 +coef(2,itb)* W(3,1,itb)
          expP(1,2,itb)=coef(1,itb)*W2(1,2,itb)
     &                 +coef(2,itb)* W(1,2,itb)
          expP(2,2,itb)=coef(1,itb)*W2(2,2,itb)
     &                 +coef(2,itb)* W(2,2,itb)+coef(3,itb)
          expP(3,2,itb)=coef(1,itb)*W2(3,2,itb)
     &                 +coef(2,itb)* W(3,2,itb)
          expP(1,3,itb)=coef(1,itb)*W2(1,3,itb)
     &                 +coef(2,itb)* W(1,3,itb)
          expP(2,3,itb)=coef(1,itb)*W2(2,3,itb)
     &                 +coef(2,itb)* W(2,3,itb)
          expP(3,3,itb)=coef(1,itb)*W2(3,3,itb)
     &                 +coef(2,itb)* W(3,3,itb)+coef(3,itb)
        enddo

****************************
* check unitarity
****************************
*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
          do jc=1,COL
*SOPTION UNROLL (3)
          do ic=1,COL
            zunit(ic,jc,itb)=conjg(expP(1,ic,itb))*expP(1,jc,itb)
     &                      +conjg(expP(2,ic,itb))*expP(2,jc,itb)
     &                      +conjg(expP(3,ic,itb))*expP(3,jc,itb)
     &                      -runit(ic,jc)
          enddo
          enddo
        enddo
*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
          res(itb)= real(zunit(1,1,itb))**2
     &            +aimag(zunit(1,1,itb))**2
     &            + real(zunit(2,1,itb))**2
```

```
     &            +aimag(zunit(2,1,itb))**2
     &            + real(zunit(3,1,itb))**2
     &            +aimag(zunit(3,1,itb))**2
     &            + real(zunit(1,2,itb))**2
     &            +aimag(zunit(1,2,itb))**2
     &            + real(zunit(2,2,itb))**2
     &            +aimag(zunit(2,2,itb))**2
     &            + real(zunit(3,2,itb))**2
     &            +aimag(zunit(3,2,itb))**2
     &            + real(zunit(1,3,itb))**2
     &            +aimag(zunit(1,3,itb))**2
     &            + real(zunit(2,3,itb))**2
     &            +aimag(zunit(2,3,itb))**2
     &            + real(zunit(3,3,itb))**2
     &            +aimag(zunit(3,3,itb))**2
        enddo
*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
        if (res(itb).GT.(1.0d-14*COL**2)**2) then
          fail(itb)=-1.0d0
          iflag = -1
        endif
        enddo

        if (iflag.EQ.1) goto 3000   ! Good condition

************************************************
* calc. exp(i W) with eigenvalue formula
************************************************
****************************
* uf = ((tr[W^2]/2)/3)^(1/2)
* gm =  (tr[W^3]/3)/(2 uf^3)
* th = (cos^-1(gm))/3
****************************
*VOPTION VEC, PVFUNC (3)
*SOPTION NOLOOPEXPAND
        do itb=1-ieoxyz,NTH-ieoxyz
        if (fail(itb).LT.0.0d0) then
          uf(itb) = dsqrt(t2(itb)/3.0d0)
          gm(itb) = t3(itb)/uf(itb)**3/2.0d0
          th(itb) = dacos(gm(itb))/3.0d0
        endif
        enddo

****************************
* lm1 =   uf(-cos(th)+3^1/2 sin(th))
* lm2 =   uf(-cos(th)-3^1/2 sin(th))
* lm3 = 2 uf cos(th) = -(lm1+lm2)
****************************
*VOPTION VEC, PVFUNC (3)
*SOPTION NOLOOPEXPAND
        do itb=1-ieoxyz,NTH-ieoxyz
        if (fail(itb).LT.0.0d0) then
          lm(1,itb)=uf(itb)*(-dcos(th(itb))+sqrt(3.0d0)*dsin(th(itb)))
          lm(2,itb)=uf(itb)*(-dcos(th(itb))-sqrt(3.0d0)*dsin(th(itb)))
          lm(3,itb)=-(lm(1,itb)+lm(2,itb))   ! traceless condition
        endif
```

```
249         enddo
250
251 ****************************
252 * calc exp(i eigen)
253 * calc coef ugins eigenvalue eq.
254 ****************************
255 *VOPTION VEC, PVFUNC (3)
256         do itb=1-ieoxyz,NTH-ieoxyz
257         if (fail(itb).LT.0.0d0) then
258           expfac(1,itb)=exp(zi*lm(1,itb))
259           expfac(2,itb)=exp(zi*lm(2,itb))
260           expfac(3,itb)=exp(zi*lm(3,itb))
261         endif
262         enddo
263 *VOPTION VEC, PVFUNC (3)
264         do itb=1-ieoxyz,NTH-ieoxyz
265         if (fail(itb).LT.0.0d0) then
266           delta(1,itb)=   lm(1,itb)+2*lm(2,itb)
267           delta(2,itb)=-2*lm(1,itb)  -lm(2,itb)
268           delta(3,itb)=   lm(1,itb)  -lm(2,itb)
269           det(itb)=-delta(1,itb)*delta(2,itb)*delta(3,itb)
270         endif
271         enddo
272 *VOPTION VEC, PVFUNC (3)
273         do itb=1-ieoxyz,NTH-ieoxyz
274         if (fail(itb).LT.0.0d0) then
275           coef(1,itb)=delta(1,itb)*expfac(1,itb)
276      &               +delta(2,itb)*expfac(2,itb)
277      &               +delta(3,itb)*expfac(3,itb)
278           coef(2,itb)=lm(1,itb)*delta(1,itb)*expfac(1,itb)
279      &               +lm(2,itb)*delta(2,itb)*expfac(2,itb)
280      &               +lm(3,itb)*delta(3,itb)*expfac(3,itb)
281           coef(3,itb)=lm(2,itb)*lm(3,itb)*delta(1,itb)*expfac(1,itb)
282      &               +lm(3,itb)*lm(1,itb)*delta(2,itb)*expfac(2,itb)
283      &               +lm(1,itb)*lm(2,itb)*delta(3,itb)*expfac(3,itb)
284         endif
285         enddo
286 *VOPTION VEC, PVFUNC (3)
287         do itb=1-ieoxyz,NTH-ieoxyz
288         if (fail(itb).LT.0.0d0) then
289           coef(1,itb)=coef(1,itb)/det(itb)
290           coef(2,itb)=coef(2,itb)/det(itb)
291           coef(3,itb)=coef(3,itb)/det(itb)
292         endif
293         enddo
294
295 ****************************
296 * calc exp(i W) with Cayley-Hamilton theorem
297 ****************************
298 *VOPTION VEC
299         do itb=1-ieoxyz,NTH-ieoxyz
300         if (fail(itb).LT.0.0d0) then
301           expP(1,1,itb)=coef(1,itb)*W2(1,1,itb)
302      &                 +coef(2,itb)* W(1,1,itb)+coef(3,itb)
303           expP(2,1,itb)=coef(1,itb)*W2(2,1,itb)
304      &                 +coef(2,itb)* W(2,1,itb)
305           expP(3,1,itb)=coef(1,itb)*W2(3,1,itb)
```

```
     &                +coef(2,itb)* W(3,1,itb)
          expP(1,2,itb)=coef(1,itb)*W2(1,2,itb)
     &                +coef(2,itb)* W(1,2,itb)
          expP(2,2,itb)=coef(1,itb)*W2(2,2,itb)
     &                +coef(2,itb)* W(2,2,itb)+coef(3,itb)
          expP(3,2,itb)=coef(1,itb)*W2(3,2,itb)
     &                +coef(2,itb)* W(3,2,itb)
          expP(1,3,itb)=coef(1,itb)*W2(1,3,itb)
     &                +coef(2,itb)* W(1,3,itb)
          expP(2,3,itb)=coef(1,itb)*W2(2,3,itb)
     &                +coef(2,itb)* W(2,3,itb)
          expP(3,3,itb)=coef(1,itb)*W2(3,3,itb)
     &                +coef(2,itb)* W(3,3,itb)+coef(3,itb)
        endif
        enddo

3000    continue
***************************
* mul  exp(i W) on u
***************************
*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
          do jc = 1,COL
*SOPTION UNROLL (3)
          do ic = 1,COL
            W(ic,jc,itb)=
     &       expP(ic,1,itb)*ue(1,jc,itb,iz,iy,ix,mu)
     &      +expP(ic,2,itb)*ue(2,jc,itb,iz,iy,ix,mu)
     &      +expP(ic,3,itb)*ue(3,jc,itb,iz,iy,ix,mu)
          enddo
          enddo
        enddo

*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
          do jc = 1,COL
*SOPTION UNROLL (3)
          do ic = 1,COL
            ue(ic,jc,itb,iz,iy,ix,mu)=W(ic,jc,itb)
          enddo
          enddo
        enddo

      enddo
      enddo
      enddo ! end of do ix,iy,iz

      call copyu(ue,mu,ieo)

      enddo ! end of do mu

c      call reunit(ue)

      return
      end
```

363

引数 (入力) **dt** : 時間ステップ、$e^{idtP_\mu(n)}U_\mu(n)$ を計算。

引数 (入力) **pe** : ゲージ場の運動量

引数 (入出力) **ue** : ゲージ場、入力を dt だけ時間を進めて返す

引数 (入力) **ieo** : `ieo=0` の時 even サイトとして計算する、`ieo=1` の時 odd サイトとして計算する

34-42 行目 : 単位行列の設定

48-59 行目 : μ, x, y, z ループの開始

62-71 行目 : $dtP_\mu(n)$ を W にコピー

73-76 行目 : フラグの設定

78-120 行目 : $\mathrm{Tr}[W^2]/2, \mathrm{Tr}[W^3]/3$ の計算

122-144 行目 : テイラー展開の係数 a_0, a_1, a_2 の計算 (プログラムでは `coef(i,itb)`)

146-169 行目 : e^{iW} の計算

171-216 行目 : e^{iW} のユニタリティーの評価

218-320 行目 : W の固有値、固有ベクトルをもとめて e^{iW} の計算

323-349 行目 : $U_\mu(n)$ に e^{iW} を左から掛けて $U_\mu(n)$ に上書き

351-353 行目 : x, y, z ループの終り

355 行目 : $U_\mu(n)$ のゴーストサイトのコピー

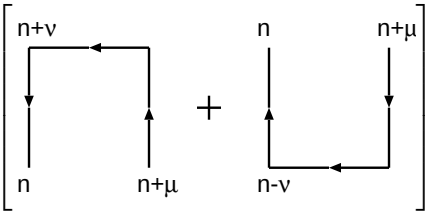
357 行目 : μ ループの終り

2.5.3 `HMCLIB/updatep_phmc.F`

$P_\mu(n)$ の時間発展は式 (15) に従っています。力 $F_\mu(n)$ はハミルトニアン (6) にあるように、ゲージ部分、擬フェルミオン部分、クローバー項の行列式部分に分られています。さらに、擬フェルミオン部分でも、クローバー項の逆からくる力の部分とホッピング行列部分からくる力に分れています。

まずゲージ作用からの力は、

$$F_{(\mathrm{gauge})_\mu}(n) = -\frac{\beta}{6}\sum_{\nu\neq\mu,\nu=1}^{4}\left[\quad\right], \tag{20}$$



です。

擬フェルミオン作用からの力は、以下のように求められる。まず、作用 S_q を微分すると、

$$\dot{S}_q = \left\{ -X^\dagger \dot{M} Y - X^\dagger \dot{T} Z \right\} + \{\text{h.c.}\}, \tag{21}$$

のように、ホッピング行列とクローバー項の微分に関するものの 2 つに分られる。ここで、微分 $\dot{M}, \dot{T}$、外場 X, Y, Z は

$$\dot{M} = \begin{pmatrix} 0 & \dot{M}_{eo} \\ \dot{M}_{oe} & 0 \end{pmatrix}, \tag{22}$$

$$\dot{T} = \begin{pmatrix} \dot{T}_{ee} & 0 \\ 0 & \dot{T}_{oo} \end{pmatrix}, \tag{23}$$

$$X = \begin{pmatrix} -(1+T)^{-1}_{ee} {M_{oe}}^\dagger \hat{X}_o \\ \hat{X}_o \end{pmatrix}, \tag{24}$$

$$Y = \begin{pmatrix} -(1+T)^{-1}_{ee} M_{eo} \hat{Y}_o \\ \hat{Y}_o \end{pmatrix}, \tag{25}$$

$$Z = \begin{pmatrix} -(1+T)^{-1}_{ee} M_{eo} \hat{Y}_o \\ (1+T)^{-1}_{oo} M_{oe} (1+T)^{-1}_{ee} M_{eo} \hat{Y}_o \end{pmatrix}, \tag{26}$$

$$\hat{X}_o = (1+T)^{-1}_{oo} \left({\hat{D}_{oo}}^\dagger \right)^{-1} \left(\hat{D}_{oo} \right)^{-1} \psi_o, \tag{27}$$

$$\hat{Y}_o = \left(\hat{D}_{oo} \right)^{-1} \psi_o, \tag{28}$$

で定義される。ディラック演算子 $\hat{D}_{oo}$ の γ_5 対称性は ${\hat{D}_{oo}}^\dagger = \gamma_5 (1+T)_{oo} \hat{D}_{oo} (1+T)^{-1}_{oo} \gamma_5$ となります。したがって、ホッピング行列の微分からの力は、

$$F_{(\text{hopping})_\mu}(n) = \kappa \text{tr}_{\text{spin}} \left[(1-\gamma_\mu) Y(n+\hat{\mu}) X^\dagger(n) + (1+\gamma_\mu) X(n+\hat{\mu}) Y^\dagger(n) \right], \tag{29}$$

となります。ここで tr_{spin} はディラックスピンの添字に対するトレースです。

次に、クローバー項に関する微分からの力ですが、これはクローバー項の行列式からの作用の微分からの力とまとめることができます。そこで先にクローバー項の行列式の作用 $S_{\text{det}}[U]$ の微分を見てみます。

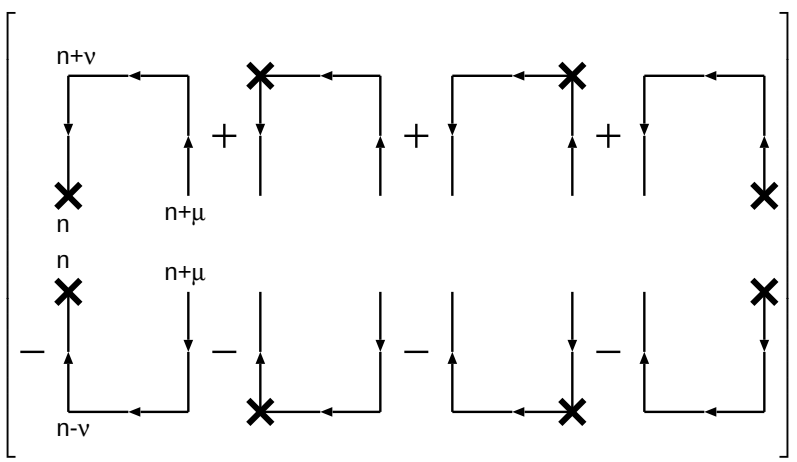
$$\dot{S}_{\text{det}} = -N_f \text{Tr} \left[\dot{T} (1+T)^{-1} \right], \tag{30}$$

となりますので、擬フェルミオン項のクローバー項からの寄与とまとめると、

$$\dot{S}_{\text{det}} + \dot{S}_{q(\text{clover})} = \text{Tr} \left[\dot{T} \left(-\frac{N_f}{2} (1+T)^{-1} - Z X^\dagger \right) \right] + \{\text{h.c.}\}, \tag{31}$$

となります。$\dot{T}$ に関する計算の後、クローバー項からの力の寄与は、

$$F_{(\text{clover})_\mu}(n) = \sum_{\nu \neq \mu, \nu=1}^{4}$$



, (32)

と表される。ここで $\times$ は、

$$(\times)_{\mu\nu}(n) = \left\{ XX_{(\mathrm{q})\,\mu\nu}(n) + XX_{(\mathrm{det})\,\mu\nu}(n) \right\} - \{\mathrm{h.c.}\}, \tag{33}$$

$$XX_{(\mathrm{q})\mu\nu}(n) = -\frac{ic_{sw}\kappa}{8}\mathrm{tr}_{\mathrm{spin}}\left[\sigma_{\mu\nu}Z(n)X(n)^{\dagger}\right], \tag{34}$$

$$XX_{(\mathrm{det})\mu\nu}(n)\} = -\frac{ic_{sw}\kappa}{8}\frac{N_f}{2}\mathrm{tr}_{\mathrm{spin}}\left[\sigma_{\mu\nu}(1+T)^{-1}(n)\right], \tag{35}$$

をリンク変数の間に挟むようにして計算します。

以上の力の寄与をそれぞれ合計して、式 (15) により、$P_\mu(n)$ の時間発展は、

$$P_\mu(n)_{t+dt} = P_\mu(n)_t - idt\left[\{U_\mu(n)_{t'}F_\mu(n)_{t'}\} - \{\mathrm{h.c.}\}\right], \tag{36}$$

で行われます。ここで、$F_\mu(n)_{t'}$ は

$$F_\mu(n)_{t'} = F_{(\mathrm{gauge})_\mu}(n)_{t'} + F_{(\mathrm{hopping})_\mu}(n)_{t'} + F_{(\mathrm{clover})_\mu}(n)_{t'}, \tag{37}$$

で、$U_\mu(n)_{t'}$ の上で計算されます。

以下に、HMCLIB/updatep_phmc.F を示します。この中ではさらに各部分からの力の計算に対応して以下のサブルーチンを呼んでいます。これらのサブルーチンの中身についてはさらにサブセクションを割いて説明します。

HMCLIB/prep_force_g.F	力の計算に必要な再利用可能な 2 つのリンクがつながったものを計算する
HMCLIB/force_g.F	$S_g[U]$ から来る力の計算 (式 (20))
PHMC/force_phmc.F	PHMC の擬フェルミオン作用から来る力の計算、クローバー項の微分は部分計算だけ行う
HMCLIB/force_hmc.F	HMC の擬フェルミオン作用から来る力の計算 (式 (29))、クローバー項の微分は部分計算だけ行う (式 (34))
HMCLIB/force_clv.F	$S_{\mathrm{det}}[U]$ から来る力の部分計算 (式 (35))
HMCLIB/force_clover.F	擬フェルミオン作用と $S_{\mathrm{det}}[U]$ のクローバー項の微分から来る力を部分計算したものをまとめて力にする計算を行う (32)

HMCLIB/updatep_phmc.F

```
1  #include "config.h"
2  #ifdef _RGGAUGE
3        subroutine updatep_phmc(beta,cRG,dt,pe,po,ue,uo,
4       &                        quark_all,ye,yo,ibsn)
5  #else
6        subroutine updatep_phmc(beta,dt,pe,po,ue,uo,
7       &                        quark_all,ye,yo,ibsn)
8  #endif
9  ************************************************************************
10 *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
11 * Gauge momentum updation with pseudo fermion fields
12 * Action : SG+SQ+SF+SUV
13 *     SG : Gauge (Plaquett) action
14 *     SQ : pseudo fermion action
15 *           v
16 *          |Doo^-1 yo|^2 for Nf=2 HMC,
17 *            v
18 *          P[Doo] yo|^2  for Nf=2 PHMC,
```

```
*            v
*          T[Doo] yo|^2  for Nf=1 PHMC,
*     SF : clover term determinant, -Nf*Tr[Log[F]]
*    SUV : UV filtering term, (not used, test version).
*
*   beta : gauge coupling 6/g^2
*    cRG : RG-gauge improvement coefficients (RG-only)
*     dt : delta t in MD
*     pe : even site conjugate momentum of gauge link
*     po : odd site conjugate momentum of gauge link
*     ue : even site gauge link
*     uo : odd site gauge link
* quark_all : all quark parameters
*     ye : even site boson fields
*     yo : odd site boson fields
*   ibsn : boson switch,
*           ibsn=0 OFF, ibsn=1 ON
*
************************************************************************
      use multiboson
#include "param-size.h"
#include "com-nnb.h"
#include "com-dirac.h"
***
      real(8), intent(in) :: beta,dt
#ifdef _RGGAUGE
      real(8), intent(in) :: cRG(0:3)
#endif
      complex(8), intent(inout) :: pe(COL,COL,0:NTH,NZ,NY,NX,NDIM)
      complex(8), intent(inout) :: po(COL,COL,0:NTH,NZ,NY,NX,NDIM)
      complex(8), intent(in) :: ue(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
      complex(8), intent(in) :: uo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
      type(quarks_def), intent(in) :: quark_all
      complex(8), intent(inout) ::
     &              ye(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1,quark_all%nmb),
     &              yo(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1,quark_all%nmb)
      integer, intent(in) :: ibsn
************************************************
      complex(8) ::
     &     BBe(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM),
     &     BBo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
      common /WORKFORCEWL/ BBe,BBo
      complex(8) ::
     &   wl34e(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &   wl34o(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &   wl98e(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &   wl98o(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2)
      common /WORKFORCE2LINKS/ wl34e,wl34o,wl98e,wl98o
*************
      integer :: iqk,imb
      integer :: ic,jc,mu
      integer :: ix,iy,iz,itb,ieoxyz
      complex(8) :: zidt
      complex(8), parameter :: zi=(0.0d0,1.0d0)
      complex(8) :: uve(COL,COL,0:NTH),uvo(COL,COL,0:NTH)
      complex(8) :: tracee(0:NTH),traceo(0:NTH)
*************
```

```
#ifdef _CLOVER
      integer :: ipl
      real(8) :: dummy
      complex(8) :: fclinve(CLSP,0:NTH,0:NZ1,0:NY1,0:NX1,CLSP)
      complex(8) :: fclinvo(CLSP,0:NTH,0:NZ1,0:NY1,0:NX1,CLSP)
      complex(8) ::
     &    ucle(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &    uclo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2)
      common /WORKCLOVER/ ucle,uclo
      complex(8) ::
     &     XXe(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &     XXo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2)
      common /WORKFORCECL/ XXe,XXo
#endif

      zidt = zi*dt

************************************************************************
* clear force vector ( dS_action )
************************************************************************
*POPTION PARALLEL
#ifdef _COALESCING
*POPTION COALESCING (_COALESCING)
#endif
*POPTION TLOCAL (ix,iy,ix,itb,ic,jc,mu)
      do ix=1,NX
      do iy=1,NY
      do iz=1,NZ
        ieoxyz=mod(ipeo+ix+iy+iz,2)
        do mu=1,NDIM
          do itb=1-ieoxyz,NTH-ieoxyz
            do jc=1,COL
            do ic=1,COL
              BBe(ic,jc,itb,iz,iy,ix,mu)=(0.0d0,0.0d0)
            enddo
            enddo
          enddo
          do itb=ieoxyz,NTH+ieoxyz-1
            do jc=1,COL
            do ic=1,COL
              BBo(ic,jc,itb,iz,iy,ix,mu)=(0.0d0,0.0d0)
            enddo
            enddo
          enddo
        enddo ! end of do mu
      enddo
      enddo
      enddo ! end of do ix,iy,iz

************************************************************************
* Prepare 2-links for force contribution
************************************************************************
      call prep_force_g(wl34e,wl34o,wl98e,wl98o,ue,uo)

************************************************************************
* Prepare force contributions from RG-gauge (rectangle) action
************************************************************************
```

```
#ifdef _RGGAUGE
      call force_g(beta,cRG,BBe,BBo,ue,uo,wl34e,wl34o,wl98e,wl98o)
#else
      call force_g(beta,BBe,BBo,ue,uo,wl34e,wl34o,wl98e,wl98o)
#endif

************************************************************************
* Prepare force contributions from multi bosons
************************************************************************
      if (ibsn.EQ.1) then
#ifdef _CLOVER
******************************
* Clear clover term contribution
******************************
*POPTION PARALLEL
#ifdef _COALESCING
*POPTION COALESCING (_COALESCING)
#endif
*POPTION TLOCAL (ix,iy,ix,itb,ic,jc,mu,ipl)
        do ix=1,NX
        do iy=1,NY
        do iz=1,NZ
          ieoxyz=mod(ipeo+ix+iy+iz,2)
          do ipl=1,NDIM*(NDIM-1)/2
            do itb=1-ieoxyz,NTH-ieoxyz
              do jc=1,COL
              do ic=1,COL
                XXe(ic,jc,itb,iz,iy,ix,ipl)=(0.0d0,0.0d0)
              enddo
              enddo
            enddo
            do itb=ieoxyz,NTH+ieoxyz-1
              do jc=1,COL
              do ic=1,COL
                XXo(ic,jc,itb,iz,iy,ix,ipl)=(0.0d0,0.0d0)
              enddo
              enddo
            enddo
          enddo ! end of do ipl
        enddo
        enddo
        enddo ! end of do ix,iy,iz

******************************
* calc clover leaf  F_{mu,nu}
******************************
        call clover(ucle,uclo,ue,uo)
#endif

        do iqk=1,quark_all%nquark    ! loop for quark with independent mass
#ifdef _CLOVER
******************************
* solve inverce clover term matrix
* fclinv = [1 - csw kappa/2 sigma_{mu,nu}F_{mu,nu}]^-1
******************************
          call clvinv(quark_all%quark(iqk)%kappa,
     &                quark_all%quark(iqk)%csw,
```

```
     &                fclinve,fclinvo,ucle,uclo,dummy,0)
          do ic=1,CLSP
            call copy_y(fclinve(1,0,0,0,0,ic),0)
            call copy_y(fclinvo(1,0,0,0,0,ic),1)
          enddo
#endif
          imb=quark_all%quark(iqk)%index

          select case (quark_all%quark(iqk)%nmethod)
          case (0)  ! PHMC
*******************************
* calc force from PHMC
*******************************
#ifdef _CLOVER
          call force_phmc(BBe,BBo,XXe,XXo,
     &                    quark_all%quark(iqk),
     &                    ye(1,1,0,0,0,0,imb),
     &                    yo(1,1,0,0,0,0,imb),
     &                    ue,uo,fclinve,fclinvo)
          if (quark_all%quark(iqk)%ptype.EQ.2) then
            call force_UV(BBe,BBo,XXe,XXo,
     &                    quark_all%quark(iqk),
     &                    ue,uo,fclinve,fclinvo)
          endif
#else
          call force_phmc(BBe,BBo,
     &                    quark_all%quark(iqk),
     &                    ye(1,1,0,0,0,0,imb),
     &                    yo(1,1,0,0,0,0,imb),
     &                    ue,uo)
#endif
          case (1) ! HMC
#ifdef _CLOVER
          call force_hmc(BBe,BBo,XXe,XXo,
     &                   quark_all%quark(iqk),
     &                   ye(1,1,0,0,0,0,imb),
     &                   yo(1,1,0,0,0,0,imb),
     &                   ue,uo,fclinve,fclinvo)
#else
          call force_hmc(BBe,BBo,
     &                   quark_all%quark(iqk),
     &                   ye(1,1,0,0,0,0,imb),
     &                   yo(1,1,0,0,0,0,imb),
     &                   ue,uo)
#endif
          end select

#ifdef _CLOVER
*******************************
* calc force from det[F]
*******************************
          call force_clv(XXe,XXo,
     &                   quark_all%quark(iqk),
     &                   fclinve,fclinvo)
#endif
        enddo ! end of do iqk

```

```
#ifdef _CLOVER
        do mu=1,NDIM*(NDIM-1)/2
          call copyXX(XXe,mu,0)
          call copyXX(XXo,mu,1)
        enddo

        call force_clover(BBe,BBo,XXe,XXo,ue,uo,wl34e,wl34o,wl98e,wl98o)
#endif
      endif ! end of if ibsn

************************************************************************
*POPTION NOPARALLEL
      do mu=1,NDIM
*POPTION PARALLEL
#ifdef _COALESCING
*POPTION COALESCING (_COALESCING)
#endif
*POPTION TLOCAL (ix,iy,iz,itb,ic,jc,ieoxyz)
*POPTION TLOCAL (uve,uvo,tracee,traceo)
      do ix=1,NX
      do iy=1,NY
      do iz=1,NZ
        ieoxyz=mod(ipeo+ix+iy+iz,2)

************************************************************************
*  uv = - u BB
************************************************************************
*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
          do jc=1,COL
*SOPTION UNROLL (3)
          do ic=1,COL
            uve(ic,jc,itb)= - ue(ic, 1,itb,iz,iy,ix,mu)*
     &                        BBe( 1,jc,itb,iz,iy,ix,mu)
     &                      - ue(ic, 2,itb,iz,iy,ix,mu)*
     &                        BBe( 2,jc,itb,iz,iy,ix,mu)
     &                      - ue(ic, 3,itb,iz,iy,ix,mu)*
     &                        BBe( 3,jc,itb,iz,iy,ix,mu)
          enddo
          enddo
        enddo
*VOPTION VEC
        do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
          do jc=1,COL
*SOPTION UNROLL (3)
          do ic=1,COL
            uvo(ic,jc,itb)= - uo(ic, 1,itb,iz,iy,ix,mu)*
     &                        BBo( 1,jc,itb,iz,iy,ix,mu)
     &                      - uo(ic, 2,itb,iz,iy,ix,mu)*
     &                        BBo( 2,jc,itb,iz,iy,ix,mu)
     &                      - uo(ic, 3,itb,iz,iy,ix,mu)*
     &                        BBo( 3,jc,itb,iz,iy,ix,mu)
          enddo
          enddo
        enddo
```

```
304
305 ************************************************************************
306 * update p
307 *   p(dt)=p(0) + i dt [ uv - uv^+]
308 ************************************************************************
309 ****************************
310 * traceless condition on uv
311 ****************************
312 *VOPTION VEC
313 *SOPTION NOLOOPEXPAND
314         do itb=1-ieoxyz,NTH-ieoxyz
315           tracee(itb)=(uve(1,1,itb)+uve(2,2,itb)+uve(3,3,itb))/COL
316         enddo
317 *VOPTION VEC
318 *SOPTION NOLOOPEXPAND
319         do itb=ieoxyz,NTH+ieoxyz-1
320           traceo(itb)=(uvo(1,1,itb)+uvo(2,2,itb)+uvo(3,3,itb))/COL
321         enddo
322 *VOPTION VEC
323         do itb=1-ieoxyz,NTH-ieoxyz
324 *SOPTION UNROLL (3)
325           do ic=1,COL
326             uve(ic,ic,itb)=uve(ic,ic,itb)-tracee(itb)
327           enddo
328         enddo
329 *VOPTION VEC
330         do itb=ieoxyz,NTH+ieoxyz-1
331 *SOPTION UNROLL (3)
332           do ic=1,COL
333             uvo(ic,ic,itb)=uvo(ic,ic,itb)-traceo(itb)
334           enddo
335         enddo
336 ************************
337 * update p
338 ************************
339 *VOPTION VEC
340         do itb=1-ieoxyz,NTH-ieoxyz
341 *SOPTION UNROLL (3)
342           do jc=1,COL
343 *SOPTION UNROLL (3)
344           do ic=1,COL
345             pe(ic,jc,itb,iz,iy,ix,mu)=pe(ic,jc,itb,iz,iy,ix,mu)
346      &                 +zidt*(       uve(ic,jc,itb)
347      &                        -conjg(uve(jc,ic,itb)))
348           enddo
349           enddo
350         enddo
351 *VOPTION VEC
352         do itb=ieoxyz,NTH+ieoxyz-1
353 *SOPTION UNROLL (3)
354           do jc=1,COL
355 *SOPTION UNROLL (3)
356           do ic=1,COL
357             po(ic,jc,itb,iz,iy,ix,mu)=po(ic,jc,itb,iz,iy,ix,mu)
358      &                 +zidt*(       uvo(ic,jc,itb)
359      &                        -conjg(uvo(jc,ic,itb)))
360           enddo
```

```
          enddo
        enddo
      enddo
      enddo
      enddo ! end of do ix,iy,iz
      enddo ! end of do mu

      return
      end

```

引数 (入力) **beta, (cRG)** : ゲージカップリング $6/g^2$,(c_i)

引数 (入力) **dt** : 分子動力学ステップサイズ

引数 (入出力) **pe, po** : even サイト、odd サイトのゲージ運動量

引数 (入力) **ue, uo:** even サイト、odd サイトのゲージリンク

引数 (入力) **quark_all** : 全クォークパラメータを含む構造体

引数 (入出力) **ye, yo** : even サイト、odd サイトの擬フェルミオン場を含む作業配列

引数 (入力) **ibsn** : `ibsn=0` の時クォークの効果を入れない (クエンチ近似)、`ibsn=1` の時クォークの効果を入れる (full QCD)

93-123 行目: $F_\mu(n)$ をクリアー (プログラム上では `BBe,BBo`)

125-128 行目: 2 リンク変数の計算

130-137 行目: ゲージ場からの力 $F_{(\mathrm{gauge})_\mu}(n)$ の計算 (`BBe,BBo` に足し込む)

139-255 行目: `ibsn=1` の時クォークからの力の寄与を計算する

144-174 行目: クローバー項からの力の寄与 $(\times)_{\mu\nu}(n)$ のクリアー (プログラム上では `XXe,XXo`)

176-179 行目: クローバーリーフ $F_{\mu\nu}(n)$ の計算

182 行目: クォークの種類 (異なる質量とか) のループの開始

184-194 行目: クローバー項の逆 $(1+T)^{-1}$ の計算

196 行目: 作業配列の中の擬フェルミオン場を指すインデックスを取り出す

198-235 行目: 擬フェルミオン作用 S_q からの力の寄与の計算

199-220 行目: PHMC の場合の力の計算 (`BBe,BBo,XXe,XXo` に足し込む)

221-234 行目: HMC の場合の力の計算 (`BBe,BBo,XXe,XXo` に足し込む)

238-243 行目: クローバー項の行列式 S_{det} からの力の寄与の計算 (`XXe,XXo` に足し込む)

245 行目: クォークの種類 (異なる質量とか) のループの終り

248-251 行目: $(\times)_{\mu\nu}(n)$ のゴーストサイトのコピー

253 行目: $(\times)_{\mu\nu}(n)$ から力 $F_{(\mathrm{clover})_\mu}(n)$ の計算

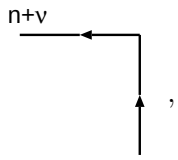
258-366 行目: 式 (36) に従って $P_\mu(n)$ を時間発展させる

271-303 行目: $-U_\mu(n)F_\mu(n)$ の計算

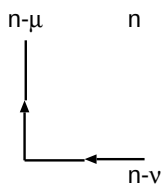
305-362 行目: $P_\mu(n)$ の時間発展。トレースレス条件も考慮にいれる

2.5.4 HMCLIB/prep_force_g.F

HMCLIB/prep_force_g.F では、$F_{(\mathrm{gauge})_\mu}(n)$ や $F_{(\mathrm{clover})_\mu}(n)$ の計算で再利用できる 2 つのリンクがつながった以下のものを計算しています。

$\mathtt{wl34}_{\mu\nu}(n) =$ 

, (38)

$\mathtt{wl98}_{\mu\nu}(n) =$ 

. (39)

また、$\mathtt{wl34}_{\nu\mu}(n) = \mathtt{wl34}^\dagger_{\mu\nu}(n)$、$\mathtt{wl98}_{\nu\mu}(n) = \mathtt{wl98}^\dagger_{\mu\nu}(n)$ なので、実際には $\mu < \nu$ なもの (もしくはその反対) だけ必要なことがわかります。ここでは、2 次元の添字 $(\mu, \nu) = (1,2), (1,3), (1,4), (2,3), (2,4), (3,4)$ を 1 次元の添字 $\mathtt{ipl} = 1,2,3,4,5,6$ に割り当てています。

HMCLIB/prep_force_g.F

```
#include "config.h"
      subroutine prep_force_g(wl34e,wl34o,wl98e,wl98o,ue,uo)
************************************************************************
*$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
* Prepare 2-links for force contribution
*
*    4---<---3     8       n
*            |     |
*      wl34  ^     ^  wl98
*            |     |
*    n       2     9---<---6    for mu < nu
*
*  for mu < nu, use wl34^+, wl98^+.
*
* wl34e : even site upper right 2-links
* wl34o : odd site upper right 2-links
* wl98e : even site lower left 2-links
* wl98o : odd site lower left 2-links
*    ue : even site gauge link with 1-ghost site
*    uo : odd site gauge link with 1-ghost site
*
************************************************************************
#include "param-size.h"
#include "com-nnb.h"
#include "com-dirac.h"
******
      complex(8), intent(out) ::
     &       wl34e(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &       wl34o(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &       wl98e(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &       wl98o(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2)
      complex(8), intent(in) :: ue(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
      complex(8), intent(in) :: uo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
```

```
******
      integer :: ic,jc,mu,nu,ipl
      integer :: ix ,iy ,iz ,itb ,ieoxyz
      integer :: ix2,iy2,iz2,itb2
      integer :: ix4,iy4,iz4,itb4
      integer :: ix9,iy9,iz9,itb9

********* prepare force
*POPTION NOPARALLEL
      do mu=1,NDIM-1
      do nu=mu+1,NDIM
      ipl=imunu(mu,nu)
*POPTION PARALLEL
*POPTION TLOCAL (ic,jc,ieoxyz)
*POPTION TLOCAL (ix ,iy ,iz ,itb )
*POPTION TLOCAL (ix2,iy2,iz2,itb2)
*POPTION TLOCAL (ix4,iy4,iz4,itb4)
*POPTION TLOCAL (ix9,iy9,iz9,itb9)
#ifdef _COALESCING
*POPTION COALESCING (_COALESCING)
#endif
        do ix=1,NX
        do iy=1,NY
        do iz=1,NZ
          ieoxyz=mod(ipeo+ix+iy+iz,2)

          ix2=ix+isx(mu)   ! n + mu
          iy2=iy+isy(mu)
          iz2=iz+isz(mu)
          ix4=ix+isx(nu)   ! n + nu
          iy4=iy+isy(nu)
          iz4=iz+isz(nu)
          ix9=ix-isx(mu)-isx(nu)    ! n - mu - nu
          iy9=iy-isy(mu)-isy(nu)
          iz9=iz-isz(mu)-isz(nu)

**********************************
*
*   4---<---3      8       n
*           |      |
*     wl34  ^      ^  wl98
*           |      |
*   n       2      9---<---6
*
**********************************
*VOPTION VEC, PREFETCH (uo)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb2=itb    +ieoxyz*ist(mu)
            itb4=itb    +ieoxyz*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              wl34e(ic,jc,itb,iz,iy,ix,ipl)=
     &                               uo(ic, 1,itb2,iz2,iy2,ix2,nu)*
     &                         conjg(uo(jc, 1,itb4,iz4,iy4,ix4,mu))
     &                       +      uo(ic, 2,itb2,iz2,iy2,ix2,nu)*
```

```
     &                          conjg(uo(jc, 2,itb4,iz4,iy4,ix4,mu))
     &                         +      uo(ic, 3,itb2,iz2,iy2,ix2,nu)*
     &                          conjg(uo(jc, 3,itb4,iz4,iy4,ix4,mu))
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (ue)
          do itb=ieoxyz,NTH+ieoxyz-1
            itb2=itb+(1-ieoxyz)*ist(mu)
            itb4=itb+(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              wl34o(ic,jc,itb,iz,iy,ix,ipl)=
     &                               ue(ic, 1,itb2,iz2,iy2,ix2,nu)*
     &                          conjg(ue(jc, 1,itb4,iz4,iy4,ix4,mu))
     &                         +      ue(ic, 2,itb2,iz2,iy2,ix2,nu)*
     &                          conjg(ue(jc, 2,itb4,iz4,iy4,ix4,mu))
     &                         +      ue(ic, 3,itb2,iz2,iy2,ix2,nu)*
     &                          conjg(ue(jc, 3,itb4,iz4,iy4,ix4,mu))
            enddo
            enddo
          enddo

*VOPTION VEC, PREFETCH (ue)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb9=itb-(1-ieoxyz)*ist(mu)-(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              wl98e(ic,jc,itb,iz,iy,ix,ipl)=
     &                          conjg(ue( 1,ic,itb9,iz9,iy9,ix9,mu))*
     &                                ue( 1,jc,itb9,iz9,iy9,ix9,nu)
     &                         +conjg(ue( 2,ic,itb9,iz9,iy9,ix9,mu))*
     &                                ue( 2,jc,itb9,iz9,iy9,ix9,nu)
     &                         +conjg(ue( 3,ic,itb9,iz9,iy9,ix9,mu))*
     &                                ue( 3,jc,itb9,iz9,iy9,ix9,nu)
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (uo)
          do itb=ieoxyz,NTH+ieoxyz-1
            itb9=itb    -ieoxyz*ist(mu)    -ieoxyz*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              wl98o(ic,jc,itb,iz,iy,ix,ipl)=
     &                          conjg(uo( 1,ic,itb9,iz9,iy9,ix9,mu))*
     &                                uo( 1,jc,itb9,iz9,iy9,ix9,nu)
     &                         +conjg(uo( 2,ic,itb9,iz9,iy9,ix9,mu))*
     &                                uo( 2,jc,itb9,iz9,iy9,ix9,nu)
     &                         +conjg(uo( 3,ic,itb9,iz9,iy9,ix9,mu))*
     &                                uo( 3,jc,itb9,iz9,iy9,ix9,nu)
            enddo
```

```
148             enddo
149           enddo
150         enddo
151         enddo
152         enddo ! end of do ix,iy,iz
153       enddo ! end of do nu
154       enddo ! end of do mu
155
156       do mu=1,NDIM*(NDIM-1)/2
157         call copyXX(wl34e,mu,0)
158         call copyXX(wl34o,mu,1)
159         call copyXX(wl98e,mu,0)
160         call copyXX(wl98o,mu,1)
161       enddo
162
163       return
164       end
165
```

引数 (出力) **wl34e, wl34o :** even サイト、odd サイトの式 (38)

引数 (出力) **wl98e, wl98o :** even サイト、odd サイトの式 (39)

引数 (入力) **ue, uo :** even サイト、odd サイトのリンク変数

42-45 行目: μ, ν ループの開始

46-58 行目: x, y, z ループの開始

60-68 行目: $n + \hat{\mu}, n + \hat{\nu}, n - \hat{\mu} - \hat{\nu}$ の計算 (x, y, z 座標の)

70-114 行目: `wl34` の計算

116-149 行目: `wl98` の計算

150-154 行目: x, y, z ループ、μ, ν ループの終り

156-161 行目: `wl34`、`wl98` のゴーストサイトのコピー

2.5.5 HMCLIB/force_g.F

HMCLIB/force_g.F では $F_{(\text{gauge})_\mu}(n)$ 式 (20) を計算しています。

HMCLIB/force_g.F

```
1  #include "config.h"
2  #ifdef _RGGAUGE
3        subroutine force_g(beta,cRG,BBe,BBo,ue,uo,
4       &                   wl34e,wl34o,wl98e,wl98o)
5  #else
6        subroutine force_g(beta,BBe,BBo,ue,uo,
7       &                   wl34e,wl34o,wl98e,wl98o)
8  #endif
9  ************************************************************************
10 *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
11 * Force from gauge action
```

```
*
*  BB_{mu}(n) = -beta/6 cRT(0) Sum[VPL_{mu,nu}(n),nu=1,NDIM,nu=!=mu]
*               -beta/6 cRT(1) Sum[VRT_{mu,nu}(n),nu=1,NDIM,nu=!=mu],
*  force for U(n)_mu.
*
*  VPL_{mu,nu}(n)=
*
*    4---<---3       n       2
*    |       |       |       |
*    v       ^   +   ^       v
*    |       |       |       |
*    n       2       6---<---5
*
*  VRT_{mu,nu}(n)=
*
*    e---<---d
*    |       |
*    v       ^
*    |       |
*    4       3       4---<---3---<---b       7---<---4---<---3
*    |       |       |               |       |               |
*    v       ^   +   v               ^   +   v               ^
*    |       |       |               |       |               |
*    n       2       n       2--->---a       8--->---n       2
*
*    n       2       n       2--->---a       8--->---n       2
*    |       |       |               |       |               |
* +  ^       v   +   ^               v   +   ^               v
*    |       |       |               |       |               |
*    6       5       6---<---5---<---f       9---<---6---<---5
*    |       |
*    ^       v
*    |       |
*    i---<---h
*
*  beta : gauge coupling
*   cRG : RG-improvement coefficients (RG-only)
*   BBe : even site force contribution
*   BBo : odd site force contribution
*    ue : even site gauge link with 1-ghost site
*    uo : odd site gauge link with 1-ghost site
*
************************************************************************
      use multiboson
#include "param-size.h"
#include "com-nnb.h"
#include "com-dirac.h"
******
      real(8), intent(in) :: beta
#ifdef _RGGAUGE
      real(8), intent(in) :: cRG(0:3)
#endif
      complex(8), intent(inout) ::
     &           BBe(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM),
     &           BBo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
      complex(8), intent(in) :: ue(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
      complex(8), intent(in) :: uo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
```

```
      complex(8), intent(in) ::
     &         wl34e(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &         wl34o(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &         wl98e(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &         wl98o(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2)
************************
      real(8) :: beta_c0
      complex(8) ::   vge(COL,COL,0:NTH),  vgo(COL,COL,0:NTH)
      complex(8) ::  w34e(COL,COL,0:NTH), w34o(COL,COL,0:NTH)
      complex(8) ::  w61e(COL,COL,0:NTH), w61o(COL,COL,0:NTH)
      complex(8) :: w561e(COL,COL,0:NTH),w561o(COL,COL,0:NTH)
      complex(8) :: w341e(COL,COL,0:NTH),w341o(COL,COL,0:NTH)
#ifdef _RGGAUGE
      real(8) :: beta_c1
      complex(8) :: wl8e(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM) ! wl8_{mu}(n)=U^+_{mu}(n-mu)
      complex(8) :: wl8o(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
      common /RGLINK/ wl8e,wl8o
      complex(8) :: tmp0e(COL,COL,0:NTH),tmp0o(COL,COL,0:NTH)
      complex(8) :: vg_RGe(COL,COL,0:NTH),vg_RGo(COL,COL,0:NTH)
      complex(8) :: wdee(COL,COL,0:NTH),wdeo(COL,COL,0:NTH)
      complex(8) :: wb3e(COL,COL,0:NTH),wb3o(COL,COL,0:NTH)
      complex(8) :: wafe(COL,COL,0:NTH),wafo(COL,COL,0:NTH)
      complex(8) :: wi6e(COL,COL,0:NTH),wi6o(COL,COL,0:NTH)
      complex(8) :: w98e(COL,COL,0:NTH),w98o(COL,COL,0:NTH)
      complex(8) :: w81e(COL,COL,0:NTH),w81o(COL,COL,0:NTH)
      complex(8) ::  w3de4e(COL,COL,0:NTH), w3de4o(COL,COL,0:NTH)
      complex(8) ::  wab34e(COL,COL,0:NTH), wab34o(COL,COL,0:NTH)
      complex(8) ::  whi61e(COL,COL,0:NTH), whi61o(COL,COL,0:NTH)
      complex(8) ::  w6981e(COL,COL,0:NTH), w6981o(COL,COL,0:NTH)
      complex(8) :: w34781e(COL,COL,0:NTH),w34781o(COL,COL,0:NTH)
      complex(8) :: waf561e(COL,COL,0:NTH),waf561o(COL,COL,0:NTH)
      complex(8) :: w3de41_wab341e(COL,COL,0:NTH)
      complex(8) :: w3de41_wab341o(COL,COL,0:NTH)
      complex(8) :: w5hi61_w56981e(COL,COL,0:NTH)
      complex(8) :: w5hi61_w56981o(COL,COL,0:NTH)
#endif
****
      integer :: ic,jc,mu,nu,ipl
      integer :: ix ,iy ,iz ,itb ,ieoxyz
      integer :: ix2,iy2,iz2,itb2
      integer :: ix5,iy5,iz5,itb5
#ifdef _RGGAUGE
      integer :: ix4,iy4,iz4,itb4
      integer :: ix6,iy6,iz6,itb6
      integer :: ix7,iy7,iz7,itb7
      integer :: ix8,iy8,iz8,itb8
#endif
****

#ifdef _RGGAUGE
      call copyu_RG(wl8e,wl8o,ue,uo)
      beta_c0 = (beta/6.0d0)*cRG(0)
      beta_c1 = (beta/6.0d0)*cRG(1)
#else
      beta_c0 = (beta/6.0d0)
#endif

```

```
126
127 ************************************************************************
128 *POPTION NOPARALLEL
129       do mu=1,NDIM
130 *POPTION PARALLEL
131 #ifdef _COALESCING
132 *POPTION COALESCING (_COALESCING)
133 #endif
134 *POPTION TLOCAL (vge,vgo)
135 *POPTION TLOCAL (w34e,w34o,w341e,w341o)
136 *POPTION TLOCAL (w61e,w61o,w561e,w561o)
137 #ifdef _RGGAUGE
138 *POPTION TLOCAL (vg_RGe,vg_RGo)
139 *POPTION TLOCAL (tmp0e,tmp0o)
140 *POPTION TLOCAL (wdee,wdeo,wb3e,wb3o,w34e,w34o,wafe,wafo)
141 *POPTION TLOCAL (wi6e,wi6o,w61e,w61o,w98e,w98o,w81e,w81o)
142 *POPTION TLOCAL (w3de4e,w3de4o,wab34e,wab34o,w34781e,w34781o)
143 *POPTION TLOCAL (whi61e,whi61o,w6981e,w6981o,waf561e,waf561o)
144 *POPTION TLOCAL (w3de41_wab341e,w3de41_wab341o)
145 *POPTION TLOCAL (w5hi61_w56981e,w5hi61_w56981o)
146 #endif
147 *POPTION TLOCAL (ic,jc,nu,ipl,ieoxyz)
148 *POPTION TLOCAL (ix ,iy ,iz ,itb )
149 *POPTION TLOCAL (ix2,iy2,iz2,itb2)
150 *POPTION TLOCAL (ix5,iy5,iz5,itb5)
151 #ifdef _RGGAUGE
152 *POPTION TLOCAL (ix4,iy4,iz4,itb4)
153 *POPTION TLOCAL (ix6,iy6,iz6,itb6)
154 *POPTION TLOCAL (ix7,iy7,iz7,itb7)
155 *POPTION TLOCAL (ix8,iy8,iz8,itb8)
156 #endif
157       do ix=1,NX
158       do iy=1,NY
159       do iz=1,NZ
160         ieoxyz=mod(ipeo+ix+iy+iz,2)
161         ix2=ix+isx(mu)   ! n + mu
162         iy2=iy+isy(mu)
163         iz2=iz+isz(mu)
164 #ifdef _RGGAUGE
165         ix8=ix-isx(mu)   ! n - mu
166         iy8=iy-isy(mu)
167         iz8=iz-isz(mu)
168 #endif
169
170         do itb=1-ieoxyz,NTH-ieoxyz
171           do jc=1,COL
172           do ic=1,COL
173             vge(ic,jc,itb)=(0.0d0,0.0d0)
174 #ifdef _RGGAUGE
175             vg_RGe(ic,jc,itb)=(0.0d0,0.0d0)
176 #endif
177           enddo
178           enddo
179         enddo
180         do itb=ieoxyz,NTH+ieoxyz-1
181           do jc=1,COL
182           do ic=1,COL
```

```
            vgo(ic,jc,itb)=(0.0d0,0.0d0)
#ifdef _RGGAUGE
            vg_RGo(ic,jc,itb)=(0.0d0,0.0d0)
#endif
          enddo
          enddo
        enddo

        do nu=1,NDIM
        if (mu.EQ.nu) goto 1000
          ix5=ix2-isx(nu)    ! n + mu - nu
          iy5=iy2-isy(nu)
          iz5=iz2-isz(nu)
#ifdef _RGGAUGE
          ix4=ix +isx(nu)    ! n + nu
          iy4=iy +isy(nu)
          iz4=iz +isz(nu)
          ix6=ix -isx(nu)    ! n - nu
          iy6=iy -isy(nu)
          iz6=iz -isz(nu)
          ix7=ix8+isx(nu)    ! n - mu + nu
          iy7=iy8+isy(nu)
          iz7=iz8+isz(nu)
#endif
          ipl=imunu(mu,nu)

          if (ipl.GT.0) then ! mu < nu
*VOPTION VEC
            do itb=1-ieoxyz,NTH-ieoxyz
              itb2=itb    +ieoxyz *ist(mu)
#ifdef _RGGAUGE
              itb4=itb    +ieoxyz *ist(nu)
              itb5=itb2-(1-ieoxyz)*ist(nu)
#endif
*SOPTION UNROLL (3)
              do jc=1,COL
*SOPTION UNROLL (3)
              do ic=1,COL
                w34e(ic,jc,itb)=      wl34e(ic,jc,itb ,iz ,iy ,ix ,ipl)
                w61e(ic,jc,itb)=      wl98o(ic,jc,itb2,iz2,iy2,ix2,ipl)
#ifdef _RGGAUGE
                wdee(ic,jc,itb)=      wl34o(ic,jc,itb4,iz4,iy4,ix4,ipl)
                wb3e(ic,jc,itb)=      wl34o(ic,jc,itb2,iz2,iy2,ix2,ipl)
                wafe(ic,jc,itb)=conjg(wl34e(jc,ic,itb5,iz5,iy5,ix5,ipl))
                wi6e(ic,jc,itb)=      wl98e(ic,jc,itb5,iz5,iy5,ix5,ipl)
                w98e(ic,jc,itb)=      wl98e(ic,jc,itb ,iz ,iy ,ix ,ipl)
                w81e(ic,jc,itb)=conjg(wl98o(jc,ic,itb4,iz4,iy4,ix4,ipl))
#endif
              enddo
              enddo
            enddo
*VOPTION VEC
            do itb=ieoxyz,NTH+ieoxyz-1
              itb2=itb +(1-ieoxyz)*ist(mu)
#ifdef _RGGAUGE
              itb4=itb +(1-ieoxyz)*ist(nu)
              itb5=itb2-   ieoxyz *ist(nu)
```

```
#endif
*SOPTION UNROLL (3)
              do jc=1,COL
*SOPTION UNROLL (3)
              do ic=1,COL
                w34o(ic,jc,itb)=      wl34o(ic,jc,itb ,iz ,iy ,ix ,ipl)
                w61o(ic,jc,itb)=      wl98e(ic,jc,itb2,iz2,iy2,ix2,ipl)
#ifdef _RGGAUGE
                wdeo(ic,jc,itb)=      wl34e(ic,jc,itb4,iz4,iy4,ix4,ipl)
                wb3o(ic,jc,itb)=      wl34e(ic,jc,itb2,iz2,iy2,ix2,ipl)
                wafo(ic,jc,itb)=conjg(wl34o(jc,ic,itb5,iz5,iy5,ix5,ipl))
                wi6o(ic,jc,itb)=      wl98o(ic,jc,itb5,iz5,iy5,ix5,ipl)
                w98o(ic,jc,itb)=      wl98o(ic,jc,itb ,iz ,iy ,ix ,ipl)
                w81o(ic,jc,itb)=conjg(wl98e(jc,ic,itb4,iz4,iy4,ix4,ipl))
#endif
              enddo
              enddo
            enddo
          else               ! mu > nu
            ipl=-ipl
*VOPTION VEC
            do itb=1-ieoxyz,NTH-ieoxyz
              itb2=itb    +ieoxyz *ist(mu)
#ifdef _RGGAUGE
              itb4=itb    +ieoxyz *ist(nu)
              itb5=itb2-(1-ieoxyz)*ist(nu)
#endif
*SOPTION UNROLL (3)
              do jc=1,COL
*SOPTION UNROLL (3)
              do ic=1,COL
                w34e(ic,jc,itb)=conjg(wl34e(jc,ic,itb ,iz ,iy ,ix ,ipl))
                w61e(ic,jc,itb)=conjg(wl98o(jc,ic,itb2,iz2,iy2,ix2,ipl))
#ifdef _RGGAUGE
                wdee(ic,jc,itb)=conjg(wl34o(jc,ic,itb4,iz4,iy4,ix4,ipl))
                wb3e(ic,jc,itb)=conjg(wl34o(jc,ic,itb2,iz2,iy2,ix2,ipl))
                wafe(ic,jc,itb)=      wl34e(ic,jc,itb5,iz5,iy5,ix5,ipl)
                wi6e(ic,jc,itb)=conjg(wl98e(jc,ic,itb5,iz5,iy5,ix5,ipl))
                w98e(ic,jc,itb)=conjg(wl98e(jc,ic,itb ,iz ,iy ,ix ,ipl))
                w81e(ic,jc,itb)=      wl98o(ic,jc,itb4,iz4,iy4,ix4,ipl)
#endif
              enddo
              enddo
            enddo
*VOPTION VEC
            do itb=ieoxyz,NTH+ieoxyz-1
              itb2=itb +(1-ieoxyz)*ist(mu)
#ifdef _RGGAUGE
              itb4=itb +(1-ieoxyz)*ist(nu)
              itb5=itb2-   ieoxyz *ist(nu)
#endif
*SOPTION UNROLL (3)
              do jc=1,COL
*SOPTION UNROLL (3)
              do ic=1,COL
                w34o(ic,jc,itb)=conjg(wl34o(jc,ic,itb ,iz ,iy ,ix ,ipl))
                w61o(ic,jc,itb)=conjg(wl98e(jc,ic,itb2,iz2,iy2,ix2,ipl))
```

```
#ifdef _RGGAUGE
                wdeo(ic,jc,itb)=conjg(wl34e(jc,ic,itb4,iz4,iy4,ix4,ipl))
                wb3o(ic,jc,itb)=conjg(wl34e(jc,ic,itb2,iz2,iy2,ix2,ipl))
                wafo(ic,jc,itb)=      wl34o(ic,jc,itb5,iz5,iy5,ix5,ipl)
                wi6o(ic,jc,itb)=conjg(wl98o(jc,ic,itb5,iz5,iy5,ix5,ipl))
                w98o(ic,jc,itb)=conjg(wl98o(jc,ic,itb ,iz ,iy ,ix ,ipl))
                w81o(ic,jc,itb)=      wl98e(ic,jc,itb4,iz4,iy4,ix4,ipl)
#endif
              enddo
              enddo
            enddo
          endif

************************************************************
* Force from Plaquette part of gauge action
************************************************************
*********************************
*
*   4---<---3
*   |       |
*   v       ^
*   |       |
*   n       2   w341
*
*********************************
*VOPTION VEC, PREFETCH (uo)
          do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w341e(ic,jc,itb)=    w34e(ic, 1,itb)*
     &                          conjg(ue(jc, 1,itb ,iz ,iy ,ix ,nu))
     &                         +    w34e(ic, 2,itb)*
     &                          conjg(ue(jc, 2,itb ,iz ,iy ,ix ,nu))
     &                         +    w34e(ic, 3,itb)*
     &                          conjg(ue(jc, 3,itb ,iz ,iy ,ix ,nu))
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (ue)
          do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w341o(ic,jc,itb)=    w34o(ic, 1,itb)*
     &                          conjg(uo(jc, 1,itb ,iz ,iy ,ix ,nu))
     &                         +    w34o(ic, 2,itb)*
     &                          conjg(uo(jc, 2,itb ,iz ,iy ,ix ,nu))
     &                         +    w34o(ic, 3,itb)*
     &                          conjg(uo(jc, 3,itb ,iz ,iy ,ix ,nu))
            enddo
            enddo
          enddo
*********************************
*
```

```
*   n       2
*   |       |
*   ^       v
*   |       |
*   6---<---5   w561
*
*********************************
*VOPTION VEC, PREFETCH (uo)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb5=itb+   ieoxyz *ist(mu)-(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w561e(ic,jc,itb)=conjg(ue( 1,ic,itb5,iz5,iy5,ix5,nu))*
     &                              w61e( 1,jc,itb)
     &                         +conjg(ue( 2,ic,itb5,iz5,iy5,ix5,nu))*
     &                              w61e( 2,jc,itb)
     &                         +conjg(ue( 3,ic,itb5,iz5,iy5,ix5,nu))*
     &                              w61e( 3,jc,itb)
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (ue)
          do itb=ieoxyz,NTH+ieoxyz-1
            itb5=itb+(1-ieoxyz)*ist(mu)-   ieoxyz *ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w561o(ic,jc,itb)=conjg(uo( 1,ic,itb5,iz5,iy5,ix5,nu))*
     &                              w61o( 1,jc,itb)
     &                         +conjg(uo( 2,ic,itb5,iz5,iy5,ix5,nu))*
     &                              w61o( 2,jc,itb)
     &                         +conjg(uo( 3,ic,itb5,iz5,iy5,ix5,nu))*
     &                              w61o( 3,jc,itb)
            enddo
            enddo
          enddo
************************
*VOPTION VEC
          do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              vge(ic,jc,itb)=vge(ic,jc,itb)
     &                    +w341e(ic,jc,itb)
     &                    +w561e(ic,jc,itb)
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
```

```
            do ic=1,COL
              vgo(ic,jc,itb)=vgo(ic,jc,itb)
     &                     +w341o(ic,jc,itb)
     &                     +w561o(ic,jc,itb)
            enddo
            enddo
          enddo

#ifdef _RGGAUGE
*****************************************************************
* Force from Rectanluar part of gauge action
*****************************************************************
*********************************
*
*   e---<---d
*   |       |
*   v       ^
*   |       |
*   4       3   w3de4
*           |
*           ^
*           |
*           2
*
*********************************
*VOPTION VEC, PREFETCH (uo)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb2=itb    +ieoxyz*ist(mu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              tmp0e(ic,jc,itb)=     uo(ic, 1,itb2,iz2,iy2,ix2,nu)*
     &                             wdee( 1,jc,itb)
     &                        +     uo(ic, 2,itb2,iz2,iy2,ix2,nu)*
     &                             wdee( 2,jc,itb)
     &                        +     uo(ic, 3,itb2,iz2,iy2,ix2,nu)*
     &                             wdee( 3,jc,itb)
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (ue)
          do itb=ieoxyz,NTH+ieoxyz-1
            itb2=itb+(1-ieoxyz)*ist(mu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              tmp0o(ic,jc,itb)=     ue(ic, 1,itb2,iz2,iy2,ix2,nu)*
     &                             wdeo( 1,jc,itb)
     &                        +     ue(ic, 2,itb2,iz2,iy2,ix2,nu)*
     &                             wdeo( 2,jc,itb)
     &                        +     ue(ic, 3,itb2,iz2,iy2,ix2,nu)*
     &                             wdeo( 3,jc,itb)
            enddo
            enddo
          enddo
```

```
*VOPTION VEC, PREFETCH (uo)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb4=itb    +ieoxyz*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w3de4e(ic,jc,itb)=  tmp0e(ic, 1,itb)*
     &                           conjg(uo(jc, 1,itb4,iz4,iy4,ix4,nu))
     &                         +   tmp0e(ic, 2,itb)*
     &                           conjg(uo(jc, 2,itb4,iz4,iy4,ix4,nu))
     &                         +   tmp0e(ic, 3,itb)*
     &                           conjg(uo(jc, 3,itb4,iz4,iy4,ix4,nu))
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (ue)
          do itb=ieoxyz,NTH+ieoxyz-1
            itb4=itb+(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w3de4o(ic,jc,itb)=  tmp0o(ic, 1,itb)*
     &                           conjg(ue(jc, 1,itb4,iz4,iy4,ix4,nu))
     &                         +   tmp0o(ic, 2,itb)*
     &                           conjg(ue(jc, 2,itb4,iz4,iy4,ix4,nu))
     &                         +   tmp0o(ic, 3,itb)*
     &                           conjg(ue(jc, 3,itb4,iz4,iy4,ix4,nu))
            enddo
            enddo
          enddo

**********************************
*
*         wab34
*
*   4---<---3---<---b
*                   |
*                   ^
*                   |
*   n       2--->---a
*
**********************************
*VOPTION VEC, PREFETCH (uo)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb2=itb    +ieoxyz*ist(mu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              tmp0e(ic,jc,itb)=      uo(ic, 1,itb2,iz2,iy2,ix2,mu)*
     &                              wb3e( 1,jc,itb)
     &                         +      uo(ic, 2,itb2,iz2,iy2,ix2,mu)*
     &                              wb3e( 2,jc,itb)
     &                         +      uo(ic, 3,itb2,iz2,iy2,ix2,mu)*
     &                              wb3e( 3,jc,itb)
```

```
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (ue)
          do itb=ieoxyz,NTH+ieoxyz-1
            itb2=itb+(1-ieoxyz)*ist(mu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              tmp0o(ic,jc,itb)=      ue(ic, 1,itb2,iz2,iy2,ix2,mu)*
     &                              wb3o( 1,jc,itb)
     &                         +      ue(ic, 2,itb2,iz2,iy2,ix2,mu)*
     &                              wb3o( 2,jc,itb)
     &                         +      ue(ic, 3,itb2,iz2,iy2,ix2,mu)*
     &                              wb3o( 3,jc,itb)
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (uo)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb4=itb    +ieoxyz*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              wab34e(ic,jc,itb)=  tmp0e(ic, 1,itb)*
     &                         conjg(uo(jc, 1,itb4,iz4,iy4,ix4,mu))
     &                        +   tmp0e(ic, 2,itb)*
     &                         conjg(uo(jc, 2,itb4,iz4,iy4,ix4,mu))
     &                        +   tmp0e(ic, 3,itb)*
     &                         conjg(uo(jc, 3,itb4,iz4,iy4,ix4,mu))
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (ue)
          do itb=ieoxyz,NTH+ieoxyz-1
            itb4=itb+(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              wab34o(ic,jc,itb)=  tmp0o(ic, 1,itb)*
     &                         conjg(ue(jc, 1,itb4,iz4,iy4,ix4,mu))
     &                        +   tmp0o(ic, 2,itb)*
     &                         conjg(ue(jc, 2,itb4,iz4,iy4,ix4,mu))
     &                        +   tmp0o(ic, 3,itb)*
     &                         conjg(ue(jc, 3,itb4,iz4,iy4,ix4,mu))
            enddo
            enddo
          enddo

*********************************
*
*   e---<---d
*   |       |
*   v       ^
```

```
*   |       |
*   4       3       4---<---3---<---b       4
*           |                       |       |
* (   w3de4 ^   +        wab34      ^  )*   v   =  w3de41_wab341
*           |                       |       |
*   n       2       n       2--->---a       n
*
*********************************
*VOPTION VEC
          do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w3de41_wab341e(ic,jc,itb)=
     &                          (w3de4e(ic, 1,itb)+wab34e(ic, 1,itb))*
     &                         conjg(ue(jc, 1,itb ,iz ,iy ,ix ,nu))
     &                        + (w3de4e(ic, 2,itb)+wab34e(ic, 2,itb))*
     &                         conjg(ue(jc, 2,itb ,iz ,iy ,ix ,nu))
     &                        + (w3de4e(ic, 3,itb)+wab34e(ic, 3,itb))*
     &                         conjg(ue(jc, 3,itb ,iz ,iy ,ix ,nu))
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (uo)
          do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w3de41_wab341o(ic,jc,itb)=
     &                          (w3de4o(ic, 1,itb)+wab34o(ic, 1,itb))*
     &                         conjg(uo(jc, 1,itb ,iz ,iy ,ix ,nu))
     &                        + (w3de4o(ic, 2,itb)+wab34o(ic, 2,itb))*
     &                         conjg(uo(jc, 2,itb ,iz ,iy ,ix ,nu))
     &                        + (w3de4o(ic, 3,itb)+wab34o(ic, 3,itb))*
     &                         conjg(uo(jc, 3,itb ,iz ,iy ,ix ,nu))
            enddo
            enddo
          enddo

*********************************
*
*         w34781
*
*   7---<---4---<---3
*   |               |
*   v               ^
*   |               |
*   8--->---n       2
*
*********************************
*VOPTION VEC, PREFETCH (ue)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb7=itb-(1-ieoxyz)*ist(mu)    +ieoxyz*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
```

```
*SOPTION UNROLL (3)
            do ic=1,COL
              tmp0e(ic,jc,itb)=    w34e(ic, 1,itb)*
     &                          conjg(ue(jc, 1,itb7,iz7,iy7,ix7,mu))
     &                         +    w34e(ic, 2,itb)*
     &                          conjg(ue(jc, 2,itb7,iz7,iy7,ix7,mu))
     &                         +    w34e(ic, 3,itb)*
     &                          conjg(ue(jc, 3,itb7,iz7,iy7,ix7,mu))
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (uo)
          do itb=ieoxyz,NTH+ieoxyz-1
            itb7=itb-    ieoxyz*ist(mu)+(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              tmp0o(ic,jc,itb)=    w34o(ic, 1,itb)*
     &                          conjg(uo(jc, 1,itb7,iz7,iy7,ix7,mu))
     &                         +    w34o(ic, 2,itb)*
     &                          conjg(uo(jc, 2,itb7,iz7,iy7,ix7,mu))
     &                         +    w34o(ic, 3,itb)*
     &                          conjg(uo(jc, 3,itb7,iz7,iy7,ix7,mu))
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w34781e(ic,jc,itb)=tmp0e(ic, 1,itb)*w81e( 1,jc,itb)
     &                           +tmp0e(ic, 2,itb)*w81e( 2,jc,itb)
     &                           +tmp0e(ic, 3,itb)*w81e( 3,jc,itb)
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w34781o(ic,jc,itb)=tmp0o(ic, 1,itb)*w81o( 1,jc,itb)
     &                           +tmp0o(ic, 2,itb)*w81o( 2,jc,itb)
     &                           +tmp0o(ic, 3,itb)*w81o( 3,jc,itb)
            enddo
            enddo
          enddo

*********************************
*
*   n       2
*   |
*   ^
```

```
*   |
*   6       5   whi61
*   |       |
*   ^       v
*   |       |
*   i---<---h
*
*********************************
*VOPTION VEC, PREFETCH (uo)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb5=itb    +ieoxyz*ist(mu)-(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              tmp0e(ic,jc,itb)=    wl8e(ic, 1,itb5,iz5,iy5,ix5,nu)*
     &                             wi6e( 1,jc,itb)
     &                         +   wl8e(ic, 2,itb5,iz5,iy5,ix5,nu)*
     &                             wi6e( 2,jc,itb)
     &                         +   wl8e(ic, 3,itb5,iz5,iy5,ix5,nu)*
     &                             wi6e( 3,jc,itb)
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (ue)
          do itb=ieoxyz,NTH+ieoxyz-1
            itb5=itb+(1-ieoxyz)*ist(mu)-   ieoxyz *ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              tmp0o(ic,jc,itb)=    wl8o(ic, 1,itb5,iz5,iy5,ix5,nu)*
     &                             wi6o( 1,jc,itb)
     &                         +   wl8o(ic, 2,itb5,iz5,iy5,ix5,nu)*
     &                             wi6o( 2,jc,itb)
     &                         +   wl8o(ic, 3,itb5,iz5,iy5,ix5,nu)*
     &                             wi6o( 3,jc,itb)
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (uo)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb6=itb-(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
               whi61e(ic,jc,itb)= tmp0e(ic, 1,itb)*
     &                               uo( 1,jc,itb6,iz6,iy6,ix6,nu)
     &                         +   tmp0e(ic, 2,itb)*
     &                               uo( 2,jc,itb6,iz6,iy6,ix6,nu)
     &                         +   tmp0e(ic, 3,itb)*
     &                               uo( 3,jc,itb6,iz6,iy6,ix6,nu)
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (ue)
```

```
          do itb=ieoxyz,NTH+ieoxyz-1
            itb6=itb-    ieoxyz*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
               whi61o(ic,jc,itb)= tmp0o(ic, 1,itb)*
     &                                ue( 1,jc,itb6,iz6,iy6,ix6,nu)
     &                         +   tmp0o(ic, 2,itb)*
     &                                ue( 2,jc,itb6,iz6,iy6,ix6,nu)
     &                         +   tmp0o(ic, 3,itb)*
     &                                ue( 3,jc,itb6,iz6,iy6,ix6,nu)
            enddo
            enddo
          enddo

*********************************
*
*   8--->---n       2
*   |
*   ^
*   |
*   9---<---6---<---5
*
*         w6981
*
*********************************
*VOPTION VEC, PREFETCH (ue,uo)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb6=itb                   -(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              tmp0e(ic,jc,itb)=conjg(uo( 1,ic,itb6,iz6,iy6,ix6,mu))*
     &                              w98e( 1,jc,itb)
     &                         +conjg(uo( 2,ic,itb6,iz6,iy6,ix6,mu))*
     &                              w98e( 2,jc,itb)
     &                         +conjg(uo( 3,ic,itb6,iz6,iy6,ix6,mu))*
     &                              w98e( 3,jc,itb)
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (uo,ue)
          do itb=ieoxyz,NTH+ieoxyz-1
            itb6=itb                   -    ieoxyz*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              tmp0o(ic,jc,itb)=conjg(ue( 1,ic,itb6,iz6,iy6,ix6,mu))*
     &                              w98o( 1,jc,itb)
     &                         +conjg(ue( 2,ic,itb6,iz6,iy6,ix6,mu))*
     &                              w98o( 2,jc,itb)
     &                         +conjg(ue( 3,ic,itb6,iz6,iy6,ix6,mu))*
     &                              w98o( 3,jc,itb)
            enddo
```

```
            enddo
          enddo
*VOPTION VEC, PREFETCH (uo)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb8=itb-(1-ieoxyz)*ist(mu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
               w6981e(ic,jc,itb)= tmp0e(ic, 1,itb)*
     &                                uo( 1,jc,itb8,iz8,iy8,ix8,mu)
     &                         +   tmp0e(ic, 2,itb)*
     &                                uo( 2,jc,itb8,iz8,iy8,ix8,mu)
     &                         +   tmp0e(ic, 3,itb)*
     &                                uo( 3,jc,itb8,iz8,iy8,ix8,mu)
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (ue)
          do itb=ieoxyz,NTH+ieoxyz-1
            itb8=itb-   ieoxyz *ist(mu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
               w6981o(ic,jc,itb)= tmp0o(ic, 1,itb)*
     &                                ue( 1,jc,itb8,iz8,iy8,ix8,mu)
     &                         +   tmp0o(ic, 2,itb)*
     &                                ue( 2,jc,itb8,iz8,iy8,ix8,mu)
     &                         +   tmp0o(ic, 3,itb)*
     &                                ue( 3,jc,itb8,iz8,iy8,ix8,mu)
            enddo
            enddo
          enddo

*********************************
*
*         n       2
*         |
*         v
*         |
*  2      6       5       8--->---n       2
*  |      |       |       |
*  v   *( ^ whi61 v   +   ^    w6981         )   =  w5hi61_w56981
*  |      |       |       |
*  5      i---<---h       9---<---6---<---5
*
*********************************
*VOPTION VEC
          do itb=1-ieoxyz,NTH-ieoxyz
            itb5=itb    +ieoxyz*ist(mu)-(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w5hi61_w56981e(ic,jc,itb)=
     &                         conjg(ue( 1,ic,itb5,iz5,iy5,ix5,nu))*
```

```
     &                           (whi61e( 1,jc,itb)+w6981e( 1,jc,itb))
     &                         +conjg(ue( 2,ic,itb5,iz5,iy5,ix5,nu))*
     &                           (whi61e( 2,jc,itb)+w6981e( 2,jc,itb))
     &                         +conjg(ue( 3,ic,itb5,iz5,iy5,ix5,nu))*
     &                           (whi61e( 3,jc,itb)+w6981e( 3,jc,itb))
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=ieoxyz,NTH+ieoxyz-1
            itb5=itb+(1-ieoxyz)*ist(mu)-    ieoxyz*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w5hi61_w56981o(ic,jc,itb)=
     &                          conjg(uo( 1,ic,itb5,iz5,iy5,ix5,nu))*
     &                           (whi61o( 1,jc,itb)+w6981o( 1,jc,itb))
     &                         +conjg(uo( 2,ic,itb5,iz5,iy5,ix5,nu))*
     &                           (whi61o( 2,jc,itb)+w6981o( 2,jc,itb))
     &                         +conjg(uo( 3,ic,itb5,iz5,iy5,ix5,nu))*
     &                           (whi61o( 3,jc,itb)+w6981o( 3,jc,itb))
            enddo
            enddo
          enddo

**********************************
*
*   n       2--->---a
*   |               |
*   ^               v
*   |               |
*   6---<---5---<---f
*
*         waf561
*
**********************************
*VOPTION VEC, PREFETCH (ue)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb5=itb    +ieoxyz*ist(mu)-(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              tmp0e(ic,jc,itb)=    wafe(ic, 1,itb)*
     &                          conjg(ue(jc, 1,itb5,iz5,iy5,ix5,mu))
     &                         +    wafe(ic, 2,itb)*
     &                          conjg(ue(jc, 2,itb5,iz5,iy5,ix5,mu))
     &                         +    wafe(ic, 3,itb)*
     &                          conjg(ue(jc, 3,itb5,iz5,iy5,ix5,mu))
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (uo)
          do itb=ieoxyz,NTH+ieoxyz-1
            itb5=itb+(1-ieoxyz)*ist(mu)-    ieoxyz*ist(nu)
*SOPTION UNROLL (3)
```

```
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              tmp0o(ic,jc,itb)=    wafo(ic, 1,itb)*
     &                          conjg(uo(jc, 1,itb5,iz5,iy5,ix5,mu))
     &                         +    wafo(ic, 2,itb)*
     &                          conjg(uo(jc, 2,itb5,iz5,iy5,ix5,mu))
     &                         +    wafo(ic, 3,itb)*
     &                          conjg(uo(jc, 3,itb5,iz5,iy5,ix5,mu))
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              waf561e(ic,jc,itb)=tmp0e(ic, 1,itb)*w61e( 1,jc,itb)
     &                           +tmp0e(ic, 2,itb)*w61e( 2,jc,itb)
     &                           +tmp0e(ic, 3,itb)*w61e( 3,jc,itb)
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              waf561o(ic,jc,itb)=tmp0o(ic, 1,itb)*w61o( 1,jc,itb)
     &                           +tmp0o(ic, 2,itb)*w61o( 2,jc,itb)
     &                           +tmp0o(ic, 3,itb)*w61o( 3,jc,itb)
            enddo
            enddo
          enddo

**************
*VOPTION VEC
          do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              vg_RGe(ic,jc,itb)=vg_RGe(ic,jc,itb)
     &                  +w34781e(ic,jc,itb)
     &                  +waf561e(ic,jc,itb)
     &             +w3de41_wab341e(ic,jc,itb)
     &             +w5hi61_w56981e(ic,jc,itb)
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
```

```
            do ic=1,COL
              vg_RGo(ic,jc,itb)=vg_RGo(ic,jc,itb)
     &                  +w34781o(ic,jc,itb)
     &                  +waf561o(ic,jc,itb)
     &           +w3de41_wab341o(ic,jc,itb)
     &           +w5hi61_w56981o(ic,jc,itb)
            enddo
            enddo
          enddo
********** end of ifdef _RGGAUGE
#endif
***********************************************************************
1000    continue
        enddo ! end of do nu

*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
          do jc=1,COL
*SOPTION UNROLL (3)
          do ic=1,COL
            BBe(ic,jc,itb,iz,iy,ix,mu)=BBe(ic,jc,itb,iz,iy,ix,mu)
     &                        -beta_c0*vge(ic,jc,itb)
#ifdef _RGGAUGE
     &                     -beta_c1*vg_RGe(ic,jc,itb)
#endif
          enddo
          enddo
        enddo
*VOPTION VEC
        do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
          do jc=1,COL
*SOPTION UNROLL (3)
          do ic=1,COL
            BBo(ic,jc,itb,iz,iy,ix,mu)=BBo(ic,jc,itb,iz,iy,ix,mu)
     &                        -beta_c0*vgo(ic,jc,itb)
#ifdef _RGGAUGE
     &                     -beta_c1*vg_RGo(ic,jc,itb)
#endif
          enddo
          enddo
        enddo

      enddo
      enddo
      enddo ! end of do ix,iy,iz
      enddo ! end of do mu

      return
      end

```

引数 (入力) **beta, (cRG):** ゲージカップリング、 $6/g^2, c_i$

引数 (入出力) **BBe, BBo :** even サイト、odd サイトの力 $F_\mu(n)$、これに $F_{(\text{gauge})_\mu}(n)$ を足し込む

引数 (入力) **ue, uo :** even サイト、odd サイトのゲージリンク

引数 (入力) **wl34e, wl34o :** even サイト、odd サイトの `wl34` 式 (38)

引数 (入力) **wl98e, wl98o :** even サイト、odd サイトの `wl98` 式 (39)

118-124 行目: カップリングの設定

129 行目: μ ループの開始

130-160 行目: x, y, z ループの開始

161-168 行目: x, y, z 座標に関して $n + \hat{\mu}(n - \hat{\mu})$ の計算

170-189 行目: 作業配列のクリアー (ν ループに関する足し上げのため)

191-192 行目: ν ループの開始、$\nu = \mu$ はスキップ

193-206 行目: x, y, z 座標に関して $n + \hat{\mu} - \hat{\nu}$ $(n + \hat{\nu}, n - \hat{\nu}, n - \hat{\mu} + \hat{\nu})$ の計算

207 行目: 2 次元座量 (μ, ν) から 1 次元座標 `ipl` の指定

209-308 行目: `wl34`、`wl98` の作業配列へのコピー

210-257 行目: $\mu < \nu$ の場合のコピー

259-307 行目: $\mu > \nu$ の場合のコピー

310-417 行目: プラケット作用からの力 (ステイプル) の計算

420-991 行目: 作用に長方形を含む場合の力 (ステイプル) の計算

993-994 行目: ν ループの終り

996-1023 行目: $F_\mu(n)$ に力 (ステイプルにゲージカップリングを掛けて) を足し込む

1025-1028 行目: x, y, z ループ、μ ループの終り

2.5.6 HMCLIB/force_hmc.F

HMCLIB/force_hmc.F では擬フェルミオン作用 S_q からの力を計算しています。外場 $\hat{X}, \hat{Y}, X, Y, Z$ を式 (27)、(28)、(24)、(25)、(26) に従いながら、ホッピング行列からの力 $F_{(\mathrm{hopping})_\mu}(n)$ を式 (29)、クローバー項からの寄与 $XX_{(\mathrm{q})_{\mu\nu}}(n)$ を式 (34) で、計算しています。

HMCLIB/force_hmc.F

```
1  #include "config.h"
2  #ifdef _CLOVER
3        subroutine force_hmc(BBe,BBo,XXe,XXo,
4       &                             quark,ye,yo,ue,uo,fclinve,fclinvo)
5  #else
6        subroutine force_hmc(BBe,BBo,quark,ye,yo,ue,uo)
7  #endif
8  ************************************************************************
9  *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
10 * Calc force from a multi boson
11 *  Action : Sphmc        v
```

```
*    Shmc : HMC action, |Doo^-1 yo|^2  (odd site only)
*                 v
*           where D = 1 - kappa^2 (F^-1 M)^2
*
*     BBe : even site force contribution (\dot{u})
*     BBo : odd site force contribution (\dot{u})
*     XXe : even site force contribution (\dot{\sigma_{mu,nu}F_{mu,nu}})
*     XXo : odd site force contribution (\dot{\sigma_{mu,nu}F_{mu,nu}})
*   quark : quark parameters
*      ye : even site pseudo fermion (work vector, not used)
*      yo : odd site pseudo fermion and work vector (nmb = npoly (Nf=2) or npoly/2 (Nf=1))
*      ue : even site gauge link
*      uo : odd site gauge link
* fclinve : even site inverse clover term matrix (transposed)
* fclinvo : odd site inverse clover term matrix (transposed)
*
************************************************************************
      use multiboson
#include "param-size.h"
#include "com-nnb.h"
#include "com-dirac.h"
*********
      complex(8), intent(inout) ::
     &           BBe(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM),
     &           BBo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
#ifdef _CLOVER
      complex(8), intent(inout) ::
     &           XXe(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &           XXo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2)
#endif
      type(quark_def), intent(inout) :: quark
      complex(8), dimension(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1),
     &            intent(in) :: ye,yo
      complex(8), intent(in) :: ue(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
      complex(8), intent(in) :: uo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
#ifdef _CLOVER
      complex(8), dimension(CLSP,0:NTH,0:NZ1,0:NY1,0:NX1,CLSP),
     &            intent(in) :: fclinve,fclinvo
#endif
****************** for mb
      real(8) :: kappa2
      complex(8) :: FDye(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1)
      complex(8) :: FDyo(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1)
      complex(8) ::  xe(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1)
      complex(8) ::  xo(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1)
      complex(8) ::  we(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1)
      complex(8) ::  wo(COL,SPIN,0:NTH,0:NZ1,0:NY1,0:NX1)
      common /WORKODD/ FDye,FDyo,xe,xo,we,wo
      complex(8) :: b1e(COL,SPIN,0:NTH),b2e(COL,SPIN,0:NTH)
      complex(8) :: b1o(COL,SPIN,0:NTH),b2o(COL,SPIN,0:NTH)
      complex(8), parameter :: zi=(0.0d0,1.0d0),z1=(1.0d0,0.0d0)
      complex(8) :: zgm
#ifdef _CLOVER
      complex(8) :: zsg1,zsg2,zsg3,zsg4
      complex(8) :: zickd8
      integer :: isg1,isg2,isg3,isg4
#endif
```

```
      integer :: igm
      integer :: ix,iy,iz,itb,ieoxyz,ic,jc,is,nu,mu,ipl
      integer :: ix2,iy2,iz2,itb2
      real(8) :: err
      integer :: iiter

      kappa2=quark%kappa**2
#ifdef _CLOVER
      zickd8=zi*quark%csw*quark%kappa**3/8.0d0
#endif

************************************************************************
* Prepare gauge force contributions from polynomial pseudo ferimon
************************************************************************
******************************
*        v
*   xo = Doo^-1 yo
******************************
*POPTION PARALLEL
#ifdef _COALESCING
*POPTION COALESCING (_COALESCING)
#endif
      do ix=1,NX
      do iy=1,NY
      do iz=1,NZ
        ieoxyz=mod(ipeo+ix+iy+iz,2)
*VOPTION VEC
        do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (4)
          do is=1,SPIN
*SOPTION UNROLL (3)
          do ic=1,COL
            xo(ic,is,itb,iz,iy,ix)=yo(ic,is,itb,iz,iy,ix)
          enddo
          enddo
        enddo
      enddo
      enddo
      enddo ! end of do ix,iy,iz
      err = quark%aqhb_bicgstab_err
#ifdef _CLOVER
      call bicgstab_hmc(err,iiter,quark%kappa,wo,xo,uo,ue,1,
     &                                        fclinvo,fclinve)
#else
      call bicgstab_hmc(err,iiter,quark%kappa,wo,xo,uo,ue,1)
#endif
      quark%nqhb_bicgstab_iter(1)=quark%nqhb_bicgstab_iter(1)+iiter
      quark%nqhb_hit(1)=quark%nqhb_hit(1)+1

******************************
*               v+     v
* FDyo = Foo^-1 Doo^-1 Doo^-1 yo
*                           v
*      = Foo^-1 (gamma_5 Foo Doo^-1 Foo^-1 gamma_5) xo
*                v
*      = gamma_5 Doo^-1 Foo^-1 gamma5 xo
*
```

```
126 ******************************
127 #ifdef _CLOVER
128       call mult_gamma5(xo,wo,1)
129       call mult_fclinv(wo,xe,fclinvo,1)
130       call bicgstab_hmc(err,iiter,quark%kappa,wo,xe,uo,ue,1,
131      &                                       fclinvo,fclinve)
132 #else
133       call mult_gamma5(xo,xe,1)
134       call bicgstab_hmc(err,iiter,quark%kappa,wo,xe,uo,ue,1)
135 #endif
136       call mult_gamma5(xe,FDyo,1)
137       quark%nqhb_bicgstab_iter(1)=quark%nqhb_bicgstab_iter(1)+iiter
138       quark%nqhb_hit(1)=quark%nqhb_hit(1)+1
139 
140 ******************************
141 *   xe = Fee^-1 Meo xo
142 ******************************
143 #ifdef _CLOVER
144       call mult(xo,xe,ue,uo,0,fclinve)
145 #else
146       call mult(xo,xe,ue,uo,0)
147 #endif
148 
149 ******************************
150 * FDye = gamma_5 (F^-1ee Meo) gamma_5 FDyo
151 ******************************
152       call mult_gamma5(FDyo,wo,1)
153 #ifdef _CLOVER
154       call mult(wo,we,ue,uo,0,fclinve)
155 #else
156       call mult(wo,we,ue,uo,0)
157 #endif
158       call mult_gamma5(we,FDye,0)
159 
160       call copy_y(xe,0)
161       call copy_y(xo,1)
162       call copy_y(FDye,0)
163       call copy_y(FDyo,1)
164 
165 ************************************************************************
166 * Calc force from pseudo fermion
167 ************************************************************************
168       do mu=1,NDIM
169 *POPTION PARALLEL
170 *POPTION TLOCAL (ix,iy,iz,itb,ieoxyz,ic,jc,is)
171 *POPTION TLOCAL (ix2,iy2,iz2,itb2)
172 *POPTION TLOCAL (zgm,igm,b1e,b1o,b2e,b2o)
173 #ifdef _COALESCING
174 *POPTION COALESCING (_COALESCING)
175 #endif
176         do ix=1,NX
177         do iy=1,NY
178         do iz=1,NZ
179           ieoxyz=mod(ipeo+ix+iy+iz,2)
180 
181           ix2=ix+isx(mu)
182           iy2=iy+isy(mu)
```

```
          iz2=iz+isz(mu)
*********************************
*
*  BB =-kappa^2*tr[ (1-gamma_mu) x(n+mu)[FDy](n)^{+}
*
*                 + [FDy](n+mu) x(n)^{+}(1+gamma_mu) ]
*
*     =-kappa^2*tr[ b1(n) FDy(n)^{+} + FDy(n+mu) b2(n)^{+} ]
*
* where
*
*   b1(n)=(1-gamma_mu)x(n+mu)
*   b2(n)=(1+gamma_mu)x(n)
*
*********************************
*VOPTION VEC, PREFETCH (xe,xo)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb2=itb    +ieoxyz*ist(mu)
*SOPTION UNROLL (4)
            do is=1,SPIN
              zgm=zgamma(is,mu)
              igm=igamma(is,mu)
*SOPTION UNROLL (3)
            do ic=1,COL
              b1e(ic,is,itb)=xo(ic, is,itb2,iz2,iy2,ix2)
     &                   -zgm*xo(ic,igm,itb2,iz2,iy2,ix2)
              b2e(ic,is,itb)=xe(ic, is,itb ,iz ,iy ,ix )
     &                   +zgm*xe(ic,igm,itb ,iz ,iy ,ix )
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (xe,xo)
*SOPTION NOLOOPINTERCHANGE
          do itb=ieoxyz,NTH+ieoxyz-1
            itb2=itb+(1-ieoxyz)*ist(mu)
*SOPTION UNROLL (4), NOLOOPINTERCHANGE
            do is=1,SPIN
              zgm=zgamma(is,mu)
              igm=igamma(is,mu)
*SOPTION UNROLL (3)
            do ic=1,COL
              b1o(ic,is,itb)=xe(ic, is,itb2,iz2,iy2,ix2)
     &                   -zgm*xe(ic,igm,itb2,iz2,iy2,ix2)
              b2o(ic,is,itb)=xo(ic, is,itb ,iz ,iy ,ix )
     &                   +zgm*xo(ic,igm,itb ,iz ,iy ,ix )
            enddo
            enddo
          enddo
*********************************************
*VOPTION VEC
          do itb=1-ieoxyz,NTH-ieoxyz
            itb2=itb    +ieoxyz*ist(mu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              BBe(ic,jc,itb,iz,iy,ix,mu)=BBe(ic,jc,itb,iz,iy,ix,mu)
```

```
     &       +kappa2*(
     &        +conjg(FDye(jc,1,itb ,iz ,iy ,ix ))*      b1e(ic,1,itb)
     &        +conjg(FDye(jc,2,itb ,iz ,iy ,ix ))*      b1e(ic,2,itb)
     &        +conjg(FDye(jc,3,itb ,iz ,iy ,ix ))*      b1e(ic,3,itb)
     &        +conjg(FDye(jc,4,itb ,iz ,iy ,ix ))*      b1e(ic,4,itb)
     &        +      FDyo(ic,1,itb2,iz2,iy2,ix2) *conjg(b2e(jc,1,itb))
     &        +      FDyo(ic,2,itb2,iz2,iy2,ix2) *conjg(b2e(jc,2,itb))
     &        +      FDyo(ic,3,itb2,iz2,iy2,ix2) *conjg(b2e(jc,3,itb))
     &        +      FDyo(ic,4,itb2,iz2,iy2,ix2) *conjg(b2e(jc,4,itb)) )
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=ieoxyz,NTH+ieoxyz-1
            itb2=itb+(1-ieoxyz)*ist(mu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              BBo(ic,jc,itb,iz,iy,ix,mu)=BBo(ic,jc,itb,iz,iy,ix,mu)
     &       +kappa2*(
     &        +conjg(FDyo(jc,1,itb ,iz ,iy ,ix ))*      b1o(ic,1,itb)
     &        +conjg(FDyo(jc,2,itb ,iz ,iy ,ix ))*      b1o(ic,2,itb)
     &        +conjg(FDyo(jc,3,itb ,iz ,iy ,ix ))*      b1o(ic,3,itb)
     &        +conjg(FDyo(jc,4,itb ,iz ,iy ,ix ))*      b1o(ic,4,itb)
     &        +      FDye(ic,1,itb2,iz2,iy2,ix2) *conjg(b2o(jc,1,itb))
     &        +      FDye(ic,2,itb2,iz2,iy2,ix2) *conjg(b2o(jc,2,itb))
     &        +      FDye(ic,3,itb2,iz2,iy2,ix2) *conjg(b2o(jc,3,itb))
     &        +      FDye(ic,4,itb2,iz2,iy2,ix2) *conjg(b2o(jc,4,itb)) )
            enddo
            enddo
          enddo

        enddo
        enddo
        enddo ! end of do ix,iy,iz
      enddo ! end of do mu

#ifdef _CLOVER
**************************
*  xo = Foo^-1 Moe xe
**************************
      call mult(xe,xo,uo,ue,1,fclinvo)
************************************************************
*
*  XX(n) =
*
*   -kappa^2 tr[ sigma_{mu,nu} [y(n)] [F^-1 D x](n)^{+}]
*
*   -kappa^2 tr[ sigma_{mu,nu} [y(n)] [FDy(n)]^{+}]
*
************************************************************
*POPTION PARALLEL
*POPTION TLOCAL (ix,iy,iz,itb,ieoxyz,ic,mu,nu,ipl)
*POPTION TLOCAL (zsg1,zsg2,zsg3,zsg4)
*POPTION TLOCAL (isg1,isg2,isg3,isg4)
#ifdef _COALESCING
```

```
*POPTION COALESCING (_COALESCING)
#endif
      do ix=1,NX
      do iy=1,NY
      do iz=1,NZ
        ieoxyz=mod(ipeo+ix+iy+iz,2)
        do ipl=1,NDIM*(NDIM-1)/2
          zsg1=zsigma(1,ipl)
          zsg2=zsigma(2,ipl)
          zsg3=zsigma(3,ipl)
          zsg4=zsigma(4,ipl)
          isg1=isigma(1,ipl)
          isg2=isigma(2,ipl)
          isg3=isigma(3,ipl)
          isg4=isigma(4,ipl)
*VOPTION VEC, PREFETCH (xe,FDye)
          do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
            XXe(ic,jc,itb,iz,iy,ix,ipl)=XXe(ic,jc,itb,iz,iy,ix,ipl)
     &        +zickd8*( +  zsg1*  xe(ic,isg1,itb ,iz ,iy ,ix )
     &                   *conjg(FDye(jc,   1,itb ,iz ,iy ,ix ))
     &                  +  zsg2*  xe(ic,isg2,itb ,iz ,iy ,ix )
     &                   *conjg(FDye(jc,   2,itb ,iz ,iy ,ix ))
     &                  +  zsg3*  xe(ic,isg3,itb ,iz ,iy ,ix )
     &                   *conjg(FDye(jc,   3,itb ,iz ,iy ,ix ))
     &                  +  zsg4*  xe(ic,isg4,itb ,iz ,iy ,ix )
     &                   *conjg(FDye(jc,   4,itb ,iz ,iy ,ix )) )
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (xo,FDyo)
          do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
            XXo(ic,jc,itb,iz,iy,ix,ipl)=XXo(ic,jc,itb,iz,iy,ix,ipl)
     &        +zickd8*( +  zsg1*  xo(ic,isg1,itb ,iz ,iy ,ix )
     &                   *conjg(FDyo(jc,   1,itb ,iz ,iy ,ix ))
     &                  +  zsg2*  xo(ic,isg2,itb ,iz ,iy ,ix )
     &                   *conjg(FDyo(jc,   2,itb ,iz ,iy ,ix ))
     &                  +  zsg3*  xo(ic,isg3,itb ,iz ,iy ,ix )
     &                   *conjg(FDyo(jc,   3,itb ,iz ,iy ,ix ))
     &                  +  zsg4*  xo(ic,isg4,itb ,iz ,iy ,ix )
     &                   *conjg(FDyo(jc,   4,itb ,iz ,iy ,ix )) )
            enddo
            enddo
          enddo
        enddo ! end of do ipl
      enddo
      enddo
      enddo ! end of do ix,iy,iz
#endif

```

```
354       return
355       end
356
357
```

引数 (入出力) **BBe, BBo :** even サイト、odd サイトの $F_\mu(n)$。これに $F_{(\mathrm{hopping})_\mu}(n)$ が足し込まれる

引数 (入出力) **XXe, XXo :** even サイト、odd サイトの $(\times)_{\mu\nu}(n)$。これに S_q からの $(\times)_{\mu\nu}(n)$ が足し込まれる

引数 (入出力) **quark:** 1 個のクォークのパラメータを含む構造体

引数 (入力) **ye, yo :** even サイト、odd サイトの擬フェルミオン場 (ye は使ってない)

引数 (入力) **ue, uo :** even サイト、odd サイトのゲージリンク

引数 (入力) **fclinve, fclinvo :** even サイト、odd サイトクローバー項の逆

75-98 行目: パラメータの設定 (κ^2, $\kappa^2 i c_{\mathrm{sw}}\kappa/8$)

83-116 行目: $\hat{Y}_o = \hat{D}_{oo}^{-1}\psi_o$ の計算 (xo)、式 (28)

118-138 行目: $\hat{X}_o$ の計算 (FDyo)、式 (27)

140-147 行目: $(1+T)_{ee}^{-1}M_{eo}\hat{Y}_o$ の計算 (xe)、式 (25) の even 成分の負号なし

149-158 行目: $(1+T)_{ee}^{-1}M_{oe}^{\dagger}\hat{X}_o$ の計算 (FDye)、式 (24) の even 成分の負号なし

160-163 行目: X, Y のゴーストサイトのコピー

165-276 行目: ホッピング行列からの力の寄与 $F_{(\mathrm{hopping})_\mu}(n)$ の計算

168 行目: μ ループの開始

169-178 行目: x, y, z ループの開始

181-183 行目: x, y, z 座標の $n+\hat{\mu}$ の計算

198-230 行目: $(1-\gamma_\mu)Y(n+\hat{\mu})$、$(1+\gamma_\mu)X(n+\hat{\mu})$ の計算

232-271 行目: $F_\mu(n)$ への足し込み

279-351 行目: $(\times)_{\mu\nu}(n)$ への寄与の計算

280-282 行目: 式 (26) の Z の odd 成分の計算

292-302 行目: x, y, z ループの開始

303 行目: ipl($\mu < \nu$) ループの開始

232-271 行目: 式 (34) の計算 (負号なし)

348-351 行目: ipl, x, y, z ループの終り

2.5.7 HMCLIB/force_clv.F

クローバー項の行列式からの作用 $S_{\rm det}$ の寄与は、HMCLIB/force_clv.F で計算されています。ここでは、式 (35) を計算しています。

HMCLIB/force_clv.F

```
#include "config.h"
#ifdef _CLOVER
      subroutine force_clv(XXe,XXo,quark,fclinve,fclinvo)
************************************************************************
*$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
* Calc force contiribution from clover term log determinant derived
* by even-odd preconditioning.
*
*     SF = - Nf log[det[F]], Action
*
*     XXe : even site force contribution (\dot{\sigma_{mu,nu}F_{mu,nu}})
*     XXo : odd site force contribution (\dot{\sigma_{mu,nu}F_{mu,nu}})
*   quark : quark parameters
* fclinve : even site inverse clover term matrix (transposed)
* fclinvo : odd site inverse clover term matrix (transposed)
*
************************************************************************
      use multiboson
#include "param-size.h"
#include "com-nnb.h"
#include "com-dirac.h"
*********
      complex(8), intent(inout) ::
     &           XXe(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
     &           XXo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2)
      type(quark_def), intent(in) :: quark
      complex(8), dimension(CLSP,0:NTH,0:NZ1,0:NY1,0:NX1,CLSP),
     &            intent(in) :: fclinve,fclinvo
*********
      complex(8), parameter :: zi=(0.0d0,1.0d0)
      complex(8) :: zickd8
      complex(8) :: zsg1,zsg2,zsg3,zsg4
      integer :: isg1,isg2,isg3,isg4
      integer :: ix,iy,iz,itb,ieoxyz,ic,jc,nu,mu,ipl

*****************************
* Nf/2 (i csw kappa/8)
*****************************
      zickd8=+quark%nflavor*zi*quark%csw*quark%kappa/16.0d0

************************************************************
*
*  XX(n) = (Nf/2)(i csw kappa /8) tr[sigma_{mu,nu}F^-1]
*
************************************************************
*POPTION PARALLEL
*POPTION TLOCAL (ix,iy,iz,itb,ieoxyz,ic,mu,nu,ipl)
*POPTION TLOCAL (zsg1,zsg2,zsg3,zsg4)
*POPTION TLOCAL (isg1,isg2,isg3,isg4)
#ifdef _COALESCING
*POPTION COALESCING (_COALESCING)
#endif
```

```
      do ix=1,NX
      do iy=1,NY
      do iz=1,NZ
        ieoxyz=mod(ipeo+ix+iy+iz,2)
        do mu=1,NDIM-1
        do nu=mu+1,NDIM
          ipl=imunu(mu,nu)
          zsg1=zsigma(1,ipl)
          zsg2=zsigma(2,ipl)
          zsg3=zsigma(3,ipl)
          zsg4=zsigma(4,ipl)
          isg1=(isigma(1,ipl)-1)*COL
          isg2=(isigma(2,ipl)-1)*COL
          isg3=(isigma(3,ipl)-1)*COL
          isg4=(isigma(4,ipl)-1)*COL
*VOPTION VEC, PREFETCH (fclinve)
          do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              XXe(ic,jc,itb,iz,iy,ix,ipl)=XXe(ic,jc,itb,iz,iy,ix,ipl)
     &        +zickd8*( zsg1*fclinve(jc      ,itb,iz,iy,ix,ic+isg1)
     &                +zsg2*fclinve(jc+  COL,itb,iz,iy,ix,ic+isg2)
     &                +zsg3*fclinve(jc+2*COL,itb,iz,iy,ix,ic+isg3)
     &                +zsg4*fclinve(jc+3*COL,itb,iz,iy,ix,ic+isg4))
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (fclinvo)
          do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              XXo(ic,jc,itb,iz,iy,ix,ipl)=XXo(ic,jc,itb,iz,iy,ix,ipl)
     &        +zickd8*( zsg1*fclinvo(jc      ,itb,iz,iy,ix,ic+isg1)
     &                +zsg2*fclinvo(jc+  COL,itb,iz,iy,ix,ic+isg2)
     &                +zsg3*fclinvo(jc+2*COL,itb,iz,iy,ix,ic+isg3)
     &                +zsg4*fclinvo(jc+3*COL,itb,iz,iy,ix,ic+isg4))
            enddo
            enddo
          enddo
        enddo
        enddo ! end of do mu,nu
      enddo
      enddo
      enddo ! end of do ix,iy,iz

      return
      end
#else
      subroutine force_clv
      return
      end
#endif
******* end of ifdef _CLOVER
```

```
110
```

引数 (入出力) **XXe, XXo :** even サイト、odd サイトの $(\times)_{\mu\nu}(n)$、ここに足し込まれる。

引数 (入力) **quark :** 1 個のクォークのパラメータを含む構造体

引数 (入力) **fclinve, fclinvo :** even サイト、odd サイトのクローバー項の逆

36-39 行目: パラメータの設定 $(N_f/2ic_{\rm sw}\kappa/8)$

46-56 行目: x, y, z ループの開始

57-59 行目: μ, ν ループの開始

60-67 行目: $\sigma_{\mu\nu}$ のインデックスの計算

68-95 行目: 式 (35) の計算 (負号なし) を `XXe`、`XXo` に足し込む

96-100 行目: μ, ν, x, y, z ループの終り

2.5.8 HMCLIB/force_clover.F

`HMCLIB/force_clover.F` でようやく $(\times)_{\mu\nu}(n)$ を力に組み込みます。計算は、式 (32) に従っています。

HMCLIB/force_clover.F

```
1  #include "config.h"
2  #ifdef _CLOVER
3        subroutine force_clover(BBe,BBo,XXe,XXo,ue,uo,
4       &                        wl34e,wl34o,wl98e,wl98o)
5  ************************************************************************
6  *$Id: phmc_program_note.tex,v 1.8 2001/07/23 09:34:24 ishikawa Exp ishikawa $
7  * Force for clover terms.
8  *
9  *  BB_{mu}(n) = Sum[vcl_{mu,nu}(n),nu=!mu]
10 *
11 *        *4---<---3*       *n       2*
12 *         |       |         |       |
13 *  vcl =  v       ^    -    ^       v
14 *         |       |         |       |
15 *        *n       2*       *6---<---5*
16 *
17 *  * : X_{mu,nu}(n) - X^+_{mu,nu}(n),
18 *  where
19 *   X_{mu,nu}(n) = \sigma_{mu,nu} (force contributions at site n)
20 *
21 *   BBe : even site force contirubution (\dot{u})
22 *   BBe : odd site force contirubution
23 *   XXe : even site force contirubution (\dot{\sigma_{mu,nu}F_{mu,nu})
24 *   XXe : odd site force contirubution
25 *    ue : even site gauge link
26 *    uo : odd site gauge link
27 * wl23e : even site 2-links conecting 2-3-4 sites
28 * wl23o : odd site 2-links conecting 2-3-4 sites
29 * wl98e : even site 2-links conecting 6-9-8 sites
30 * wl98o : odd site 2-links conecting 6-9-8 sites
31 *
```

```
32  ************************************************************************
33        use multiboson
34  #include "param-size.h"
35  #include "com-nnb.h"
36  #include "com-dirac.h"
37  ***
38        complex(8), intent(inout) ::
39       &                          BBe(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM),
40       &                          BBo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
41        complex(8), intent(in) ::
42       &              XXe(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
43       &              XXo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2)
44        complex(8), intent(in) :: ue(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
45        complex(8), intent(in) :: uo(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM)
46        complex(8), intent(in) ::
47       &            wl34e(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
48       &            wl34o(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
49       &            wl98e(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2),
50       &            wl98o(COL,COL,0:NTH,0:NZ1,0:NY1,0:NX1,NDIM*(NDIM-1)/2)
51  *************
52        integer :: ic,jc,mu,nu,ipl
53        integer :: ix,iy,iz,itb,ieoxyz
54        integer :: ix2,iy2,iz2,itb2
55        integer :: ix3,iy3,iz3,itb3
56        integer :: ix4,iy4,iz4,itb4
57        integer :: ix5,iy5,iz5,itb5
58        integer :: ix6,iy6,iz6,itb6
59        complex(8) :: w34e(COL,COL,0:NTH),w34o(COL,COL,0:NTH)
60        complex(8) :: w41e(COL,COL,0:NTH),w41o(COL,COL,0:NTH)
61        complex(8) :: w56e(COL,COL,0:NTH),w56o(COL,COL,0:NTH)
62        complex(8) :: w61e(COL,COL,0:NTH),w61o(COL,COL,0:NTH)
63        complex(8) :: X2w34X4e(COL,COL,0:NTH)
64        complex(8) :: X2w34X4o(COL,COL,0:NTH)
65        complex(8) :: X3w41X1e(COL,COL,0:NTH)
66        complex(8) :: X3w41X1o(COL,COL,0:NTH)
67        complex(8) :: X2w56X6e(COL,COL,0:NTH)
68        complex(8) :: X2w56X6o(COL,COL,0:NTH)
69        complex(8) :: X5w61X1e(COL,COL,0:NTH)
70        complex(8) :: X5w61X1o(COL,COL,0:NTH)
71        complex(8) :: X1e(COL,COL,0:NTH),X2e(COL,COL,0:NTH)
72        complex(8) :: X1o(COL,COL,0:NTH),X2o(COL,COL,0:NTH)
73        complex(8) :: X3e(COL,COL,0:NTH),X4e(COL,COL,0:NTH)
74        complex(8) :: X3o(COL,COL,0:NTH),X4o(COL,COL,0:NTH)
75        complex(8) :: X5e(COL,COL,0:NTH),X6e(COL,COL,0:NTH)
76        complex(8) :: X5o(COL,COL,0:NTH),X6o(COL,COL,0:NTH)
77        complex(8) :: vcle(COL,COL,0:NTH)
78        complex(8) :: vclo(COL,COL,0:NTH)
79
80  ************************************************************************
81  *POPTION NOPARALLEL
82        do mu=1,NDIM
83  *POPTION PARALLEL
84  #ifdef _COALESCING
85  *POPTION COALESCING (_COALESCING)
86  #endif
87  *POPTION TLOCAL (ic,jc,ieoxyz,nu,ipl)
88  *POPTION TLOCAL (ix ,iy ,iz ,itb ,ic,jc,ieoxyz,nu)
```

```
*POPTION TLOCAL (ix2,iy2,iz2,itb2)
*POPTION TLOCAL (ix3,iy3,iz3,itb3)
*POPTION TLOCAL (ix4,iy4,iz4,itb4)
*POPTION TLOCAL (ix5,iy5,iz5,itb5)
*POPTION TLOCAL (ix6,iy6,iz6,itb6)
*POPTION TLOCAL (vcle,vclo)
*POPTION TLOCAL (w34e,w34o,w61e,w61o)
*POPTION TLOCAL (w41e,w41o,w56e,w56o)
*POPTION TLOCAL (X1e,X1o,X2e,X2o,X3e,X3o,X4e,X4o,X5e,X5o,X6e,X6o)
*POPTION TLOCAL (X2w34X4e,X2w34X4o,X3w41X1e,X3w41X1o)
*POPTION TLOCAL (X2w56X6e,X2w56X6o,X5w61X1e,X5w61X1o)
      do ix=1,NX
      do iy=1,NY
      do iz=1,NZ
        ieoxyz=mod(ipeo+ix+iy+iz,2)
************************************************************************
* Calc force from clover term of multi boson and inverse clover term boson
*
*  *4---<---3*       *n       2*
*   |       |         |       |
*   v       ^    -    ^       v  = vcl
*   |       |         |       |
*  *n       2*       *6---<---5*
*
*  * : X_{mu,nu} - X^+_{mu,nu}
*  where
*   X_{mu,nu} = \sigma_{mu,nu} (I csw kappa) (force contrib.)
*
************************************************************************
        ix2=ix +isx(mu) ! n + mu
        iy2=iy +isy(mu)
        iz2=iz +isz(mu)

        do itb=1-ieoxyz,NTH-ieoxyz
          do jc=1,COL
          do ic=1,COL
              vcle(ic,jc,itb)=(0.0d0,0.0d0)
          enddo
          enddo
        enddo
        do itb=ieoxyz,NTH+ieoxyz-1
          do jc=1,COL
          do ic=1,COL
              vclo(ic,jc,itb)=(0.0d0,0.0d0)
          enddo
          enddo
        enddo

        do nu=1,NDIM
        if (mu.EQ.nu) goto 1000
          ix3=ix2+isx(nu) ! n + mu + nu
          iy3=iy2+isy(nu)
          iz3=iz2+isz(nu)
          ix4=ix +isx(nu) ! n + nu
          iy4=iy +isy(nu)
          iz4=iz +isz(nu)
          ix5=ix2-isx(nu) ! n + mu - nu
```

```
          iy5=iy2-isy(nu)
          iz5=iz2-isz(nu)
          ix6=ix -isx(nu) ! n - nu
          iy6=iy -isy(nu)
          iz6=iz -isz(nu)

          ipl=imunu(mu,nu)
*******************************
*  mu > nu case
*******************************
          if (ipl .GT. 0 ) then
*VOPTION VEC
            do itb=1-ieoxyz,NTH-ieoxyz
              itb2=itb+   ieoxyz *ist(mu)
              itb3=itb+   ieoxyz *ist(mu)+   ieoxyz *ist(nu)
              itb4=itb+   ieoxyz *ist(nu)
              itb5=itb+   ieoxyz *ist(mu)-(1-ieoxyz)*ist(nu)
              itb6=itb-(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
              do jc=1,COL
*SOPTION UNROLL (3)
              do ic=1,COL
                w34e(ic,jc,itb)=      wl34e(ic,jc,itb ,iz ,iy ,ix ,ipl)
                w61e(ic,jc,itb)=      wl98o(ic,jc,itb2,iz2,iy2,ix2,ipl)
                 X1e(ic,jc,itb)=XXe(ic,jc,itb ,iz ,iy ,ix ,ipl)
                 X2e(ic,jc,itb)=XXo(ic,jc,itb2,iz2,iy2,ix2,ipl)
                 X3e(ic,jc,itb)=XXe(ic,jc,itb3,iz3,iy3,ix3,ipl)
                 X4e(ic,jc,itb)=XXo(ic,jc,itb4,iz4,iy4,ix4,ipl)
                 X5e(ic,jc,itb)=XXe(ic,jc,itb5,iz5,iy5,ix5,ipl)
                 X6e(ic,jc,itb)=XXo(ic,jc,itb6,iz6,iy6,ix6,ipl)
              enddo
              enddo
            enddo
*VOPTION VEC
            do itb=ieoxyz,NTH+ieoxyz-1
              itb2=itb+(1-ieoxyz)*ist(mu)
              itb3=itb+(1-ieoxyz)*ist(mu)+(1-ieoxyz)*ist(nu)
              itb4=itb+(1-ieoxyz)*ist(nu)
              itb5=itb+(1-ieoxyz)*ist(mu)-   ieoxyz *ist(nu)
              itb6=itb-   ieoxyz *ist(nu)
*SOPTION UNROLL (3)
              do jc=1,COL
*SOPTION UNROLL (3)
              do ic=1,COL
                w34o(ic,jc,itb)=      wl34o(ic,jc,itb ,iz ,iy ,ix ,ipl)
                w61o(ic,jc,itb)=      wl98e(ic,jc,itb2,iz2,iy2,ix2,ipl)
                 X1o(ic,jc,itb)=XXo(ic,jc,itb ,iz ,iy ,ix ,ipl)
                 X2o(ic,jc,itb)=XXe(ic,jc,itb2,iz2,iy2,ix2,ipl)
                 X3o(ic,jc,itb)=XXo(ic,jc,itb3,iz3,iy3,ix3,ipl)
                 X4o(ic,jc,itb)=XXe(ic,jc,itb4,iz4,iy4,ix4,ipl)
                 X5o(ic,jc,itb)=XXo(ic,jc,itb5,iz5,iy5,ix5,ipl)
                 X6o(ic,jc,itb)=XXe(ic,jc,itb6,iz6,iy6,ix6,ipl)
              enddo
              enddo
            enddo
*******************************
*  mu < nu case change the sign of XX(mu,nu)
```

```
*******************************
          else
            ipl=-ipl
*VOPTION VEC
            do itb=1-ieoxyz,NTH-ieoxyz
              itb2=itb+   ieoxyz *ist(mu)
              itb3=itb+   ieoxyz *ist(mu)+   ieoxyz *ist(nu)
              itb4=itb+   ieoxyz *ist(nu)
              itb5=itb+   ieoxyz *ist(mu)-(1-ieoxyz)*ist(nu)
              itb6=itb-(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
              do jc=1,COL
*SOPTION UNROLL (3)
              do ic=1,COL
                w34e(ic,jc,itb)=conjg(wl34e(jc,ic,itb ,iz ,iy ,ix ,ipl))
                w61e(ic,jc,itb)=conjg(wl98o(jc,ic,itb2,iz2,iy2,ix2,ipl))
                 X1e(ic,jc,itb)=-XXe(ic,jc,itb ,iz ,iy ,ix ,ipl)
                 X2e(ic,jc,itb)=-XXo(ic,jc,itb2,iz2,iy2,ix2,ipl)
                 X3e(ic,jc,itb)=-XXe(ic,jc,itb3,iz3,iy3,ix3,ipl)
                 X4e(ic,jc,itb)=-XXo(ic,jc,itb4,iz4,iy4,ix4,ipl)
                 X5e(ic,jc,itb)=-XXe(ic,jc,itb5,iz5,iy5,ix5,ipl)
                 X6e(ic,jc,itb)=-XXo(ic,jc,itb6,iz6,iy6,ix6,ipl)
              enddo
              enddo
            enddo
*VOPTION VEC
            do itb=ieoxyz,NTH+ieoxyz-1
              itb2=itb+(1-ieoxyz)*ist(mu)
              itb3=itb+(1-ieoxyz)*ist(mu)+(1-ieoxyz)*ist(nu)
              itb4=itb+(1-ieoxyz)*ist(nu)
              itb5=itb+(1-ieoxyz)*ist(mu)-   ieoxyz *ist(nu)
              itb6=itb-   ieoxyz *ist(nu)
*SOPTION UNROLL (3)
              do jc=1,COL
*SOPTION UNROLL (3)
              do ic=1,COL
                w34o(ic,jc,itb)=conjg(wl34o(jc,ic,itb ,iz ,iy ,ix ,ipl))
                w61o(ic,jc,itb)=conjg(wl98e(jc,ic,itb2,iz2,iy2,ix2,ipl))
                 X1o(ic,jc,itb)=-XXo(ic,jc,itb ,iz ,iy ,ix ,ipl)
                 X2o(ic,jc,itb)=-XXe(ic,jc,itb2,iz2,iy2,ix2,ipl)
                 X3o(ic,jc,itb)=-XXo(ic,jc,itb3,iz3,iy3,ix3,ipl)
                 X4o(ic,jc,itb)=-XXe(ic,jc,itb4,iz4,iy4,ix4,ipl)
                 X5o(ic,jc,itb)=-XXo(ic,jc,itb5,iz5,iy5,ix5,ipl)
                 X6o(ic,jc,itb)=-XXe(ic,jc,itb6,iz6,iy6,ix6,ipl)
              enddo
              enddo
            enddo
          endif ! end of if ipl

*****************
*
*   n       2
*           |
*      w56  v
*           |
*   6---<---5
*
```

```
******************
*VOPTION VEC, PREFETCH (ue,uo)
          do itb=1-ieoxyz,NTH-ieoxyz
            itb5=itb    +ieoxyz*ist(mu)-(1-ieoxyz)*ist(nu)
            itb6=itb-(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w56e(ic,jc,itb)=conjg(ue( 1,ic,itb5,iz5,iy5,ix5,nu))*
     &                       conjg(uo(jc, 1,itb6,iz6,iy6,ix6,mu))
     &                      +conjg(ue( 2,ic,itb5,iz5,iy5,ix5,nu))*
     &                       conjg(uo(jc, 2,itb6,iz6,iy6,ix6,mu))
     &                      +conjg(ue( 3,ic,itb5,iz5,iy5,ix5,nu))*
     &                       conjg(uo(jc, 3,itb6,iz6,iy6,ix6,mu))
            enddo
            enddo
          enddo
*VOPTION VEC, PREFETCH (ue,uo)
          do itb=ieoxyz,NTH+ieoxyz-1
            itb5=itb+(1-ieoxyz)*ist(mu)    -ieoxyz*ist(nu)
            itb6=itb    -ieoxyz*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              w56o(ic,jc,itb)=conjg(uo( 1,ic,itb5,iz5,iy5,ix5,nu))*
     &                       conjg(ue(jc, 1,itb6,iz6,iy6,ix6,mu))
     &                      +conjg(uo( 2,ic,itb5,iz5,iy5,ix5,nu))*
     &                       conjg(ue(jc, 2,itb6,iz6,iy6,ix6,mu))
     &                      +conjg(uo( 3,ic,itb5,iz5,iy5,ix5,nu))*
     &                       conjg(ue(jc, 3,itb6,iz6,iy6,ix6,mu))
            enddo
            enddo
          enddo
******************
*
*   4---<---3
*   |
*   v  w41
*   |
*   n       2
*
******************
*VOPTION VEC, PREFETCH (ue,uo)
            do itb=1-ieoxyz,NTH-ieoxyz
              itb4=itb    +ieoxyz*ist(nu)
*SOPTION UNROLL (3)
              do jc=1,COL
*SOPTION UNROLL (3)
              do ic=1,COL
                w41e(ic,jc,itb)=conjg(uo( 1,ic,itb4,iz4,iy4,ix4,mu))*
     &                         conjg(ue(jc, 1,itb ,iz ,iy ,ix ,nu))
     &                        +conjg(uo( 2,ic,itb4,iz4,iy4,ix4,mu))*
     &                         conjg(ue(jc, 2,itb ,iz ,iy ,ix ,nu))
     &                        +conjg(uo( 3,ic,itb4,iz4,iy4,ix4,mu))*
     &                         conjg(ue(jc, 3,itb ,iz ,iy ,ix ,nu))
```

```
              enddo
              enddo
            enddo
*VOPTION VEC, PREFETCH (ue,uo)
            do itb=ieoxyz,NTH+ieoxyz-1
              itb4=itb+(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
              do jc=1,COL
*SOPTION UNROLL (3)
              do ic=1,COL
                w41o(ic,jc,itb)=conjg(ue( 1,ic,itb4,iz4,iy4,ix4,mu))*
     &                          conjg(uo(jc, 1,itb ,iz ,iy ,ix ,nu))
     &                         +conjg(ue( 2,ic,itb4,iz4,iy4,ix4,mu))*
     &                          conjg(uo(jc, 2,itb ,iz ,iy ,ix ,nu))
     &                         +conjg(ue( 3,ic,itb4,iz4,iy4,ix4,mu))*
     &                          conjg(uo(jc, 3,itb ,iz ,iy ,ix ,nu))
              enddo
              enddo
            enddo

**********************
*  X2w34X4 = (X2-X2^+)w34 + w34(X4-X4^+)
*  X3w41X1 = (X3-X3^+)w41 + w41(X1-X1^+)
**********************
*VOPTION VEC
          do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              X2w34X4e(ic,jc,itb)=
     &              (X2e(ic,1,itb)-conjg(X2e(1,ic,itb)))*w34e(1,jc,itb)
     &             +(X2e(ic,2,itb)-conjg(X2e(2,ic,itb)))*w34e(2,jc,itb)
     &             +(X2e(ic,3,itb)-conjg(X2e(3,ic,itb)))*w34e(3,jc,itb)
     &             +w34e(ic,1,itb)*(X4e(1,jc,itb)-conjg(X4e(jc,1,itb)))
     &             +w34e(ic,2,itb)*(X4e(2,jc,itb)-conjg(X4e(jc,2,itb)))
     &             +w34e(ic,3,itb)*(X4e(3,jc,itb)-conjg(X4e(jc,3,itb)))
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              X3w41X1e(ic,jc,itb)=
     &              (X3e(ic,1,itb)-conjg(X3e(1,ic,itb)))*w41e(1,jc,itb)
     &             +(X3e(ic,2,itb)-conjg(X3e(2,ic,itb)))*w41e(2,jc,itb)
     &             +(X3e(ic,3,itb)-conjg(X3e(3,ic,itb)))*w41e(3,jc,itb)
     &             +w41e(ic,1,itb)*(X1e(1,jc,itb)-conjg(X1e(jc,1,itb)))
     &             +w41e(ic,2,itb)*(X1e(2,jc,itb)-conjg(X1e(jc,2,itb)))
     &             +w41e(ic,3,itb)*(X1e(3,jc,itb)-conjg(X1e(jc,3,itb)))
            enddo
            enddo
          enddo
*VOPTION VEC
```

```
          do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              X2w34X4o(ic,jc,itb)=
     &              (X2o(ic,1,itb)-conjg(X2o(1,ic,itb)))*w34o(1,jc,itb)
     &             +(X2o(ic,2,itb)-conjg(X2o(2,ic,itb)))*w34o(2,jc,itb)
     &             +(X2o(ic,3,itb)-conjg(X2o(3,ic,itb)))*w34o(3,jc,itb)
     &             +w34o(ic,1,itb)*(X4o(1,jc,itb)-conjg(X4o(jc,1,itb)))
     &             +w34o(ic,2,itb)*(X4o(2,jc,itb)-conjg(X4o(jc,2,itb)))
     &             +w34o(ic,3,itb)*(X4o(3,jc,itb)-conjg(X4o(jc,3,itb)))
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              X3w41X1o(ic,jc,itb)=
     &              (X3o(ic,1,itb)-conjg(X3o(1,ic,itb)))*w41o(1,jc,itb)
     &             +(X3o(ic,2,itb)-conjg(X3o(2,ic,itb)))*w41o(2,jc,itb)
     &             +(X3o(ic,3,itb)-conjg(X3o(3,ic,itb)))*w41o(3,jc,itb)
     &             +w41o(ic,1,itb)*(X1o(1,jc,itb)-conjg(X1o(jc,1,itb)))
     &             +w41o(ic,2,itb)*(X1o(2,jc,itb)-conjg(X1o(jc,2,itb)))
     &             +w41o(ic,3,itb)*(X1o(3,jc,itb)-conjg(X1o(jc,3,itb)))
            enddo
            enddo
          enddo
**********************
*  X2w56X6 = (X2-X2^+)w56 + w56(X6-X6^+)
*  X5w61X1 = (X5-X5^+)w61 + w61(X1-X1^+)
**********************
*VOPTION VEC
          do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              X2w56X6e(ic,jc,itb)=
     &              (X2e(ic,1,itb)-conjg(X2e(1,ic,itb)))*w56e(1,jc,itb)
     &             +(X2e(ic,2,itb)-conjg(X2e(2,ic,itb)))*w56e(2,jc,itb)
     &             +(X2e(ic,3,itb)-conjg(X2e(3,ic,itb)))*w56e(3,jc,itb)
     &             +w56e(ic,1,itb)*(X6e(1,jc,itb)-conjg(X6e(jc,1,itb)))
     &             +w56e(ic,2,itb)*(X6e(2,jc,itb)-conjg(X6e(jc,2,itb)))
     &             +w56e(ic,3,itb)*(X6e(3,jc,itb)-conjg(X6e(jc,3,itb)))
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
```

```
              X5w61X1e(ic,jc,itb)=
     &              (X5e(ic,1,itb)-conjg(X5e(1,ic,itb)))*w61e(1,jc,itb)
     &             +(X5e(ic,2,itb)-conjg(X5e(2,ic,itb)))*w61e(2,jc,itb)
     &             +(X5e(ic,3,itb)-conjg(X5e(3,ic,itb)))*w61e(3,jc,itb)
     &             +w61e(ic,1,itb)*(X1e(1,jc,itb)-conjg(X1e(jc,1,itb)))
     &             +w61e(ic,2,itb)*(X1e(2,jc,itb)-conjg(X1e(jc,2,itb)))
     &             +w61e(ic,3,itb)*(X1e(3,jc,itb)-conjg(X1e(jc,3,itb)))
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              X2w56X6o(ic,jc,itb)=
     &              (X2o(ic,1,itb)-conjg(X2o(1,ic,itb)))*w56o(1,jc,itb)
     &             +(X2o(ic,2,itb)-conjg(X2o(2,ic,itb)))*w56o(2,jc,itb)
     &             +(X2o(ic,3,itb)-conjg(X2o(3,ic,itb)))*w56o(3,jc,itb)
     &             +w56o(ic,1,itb)*(X6o(1,jc,itb)-conjg(X6o(jc,1,itb)))
     &             +w56o(ic,2,itb)*(X6o(2,jc,itb)-conjg(X6o(jc,2,itb)))
     &             +w56o(ic,3,itb)*(X6o(3,jc,itb)-conjg(X6o(jc,3,itb)))
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              X5w61X1o(ic,jc,itb)=
     &              (X5o(ic,1,itb)-conjg(X5o(1,ic,itb)))*w61o(1,jc,itb)
     &             +(X5o(ic,2,itb)-conjg(X5o(2,ic,itb)))*w61o(2,jc,itb)
     &             +(X5o(ic,3,itb)-conjg(X5o(3,ic,itb)))*w61o(3,jc,itb)
     &             +w61o(ic,1,itb)*(X1o(1,jc,itb)-conjg(X1o(jc,1,itb)))
     &             +w61o(ic,2,itb)*(X1o(2,jc,itb)-conjg(X1o(jc,2,itb)))
     &             +w61o(ic,3,itb)*(X1o(3,jc,itb)-conjg(X1o(jc,3,itb)))
            enddo
            enddo
          enddo

*************************************************************************
*  vcl  = (X2w34 + w34X4) u^{+}_nu(n)
*       + u_nu(n+mu) (X3w41 + w41X1)
*       -(X2w56 + w56X6) u_nu(n-nu)
*       - u^{+}_nu(n+mu-nu) (X5w61 + w61X1)
*************************************************************************
*VOPTION VEC
          do itb=1-ieoxyz,NTH-ieoxyz
            itb2=itb+    ieoxyz*ist(mu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              vcle(ic,jc,itb)=vcle(ic,jc,itb)
```

```
     &          +X2w34X4e(ic,1,itb)*conjg(ue(jc,1,itb ,iz ,iy ,ix ,nu))
     &          +X2w34X4e(ic,2,itb)*conjg(ue(jc,2,itb ,iz ,iy ,ix ,nu))
     &          +X2w34X4e(ic,3,itb)*conjg(ue(jc,3,itb ,iz ,iy ,ix ,nu))
     &          +uo(ic,1,itb2,iz2,iy2,ix2,nu)*X3w41X1e(1,jc,itb)
     &          +uo(ic,2,itb2,iz2,iy2,ix2,nu)*X3w41X1e(2,jc,itb)
     &          +uo(ic,3,itb2,iz2,iy2,ix2,nu)*X3w41X1e(3,jc,itb)
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=1-ieoxyz,NTH-ieoxyz
            itb5=itb+    ieoxyz*ist(mu)-(1-ieoxyz)*ist(nu)
            itb6=itb-(1-ieoxyz)*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              vcle(ic,jc,itb)=vcle(ic,jc,itb)
     &          -X2w56X6e(ic,1,itb)*uo(1,jc,itb6,iz6,iy6,ix6,nu)
     &          -X2w56X6e(ic,2,itb)*uo(2,jc,itb6,iz6,iy6,ix6,nu)
     &          -X2w56X6e(ic,3,itb)*uo(3,jc,itb6,iz6,iy6,ix6,nu)
     &          -conjg(ue(1,ic,itb5,iz5,iy5,ix5,nu))*X5w61X1e(1,jc,itb)
     &          -conjg(ue(2,ic,itb5,iz5,iy5,ix5,nu))*X5w61X1e(2,jc,itb)
     &          -conjg(ue(3,ic,itb5,iz5,iy5,ix5,nu))*X5w61X1e(3,jc,itb)
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=ieoxyz,NTH+ieoxyz-1
            itb2=itb+(1-ieoxyz)*ist(mu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              vclo(ic,jc,itb)=vclo(ic,jc,itb)
     &          +X2w34X4o(ic,1,itb)*conjg(uo(jc,1,itb ,iz ,iy ,ix ,nu))
     &          +X2w34X4o(ic,2,itb)*conjg(uo(jc,2,itb ,iz ,iy ,ix ,nu))
     &          +X2w34X4o(ic,3,itb)*conjg(uo(jc,3,itb ,iz ,iy ,ix ,nu))
     &          +ue(ic,1,itb2,iz2,iy2,ix2,nu)*X3w41X1o(1,jc,itb)
     &          +ue(ic,2,itb2,iz2,iy2,ix2,nu)*X3w41X1o(2,jc,itb)
     &          +ue(ic,3,itb2,iz2,iy2,ix2,nu)*X3w41X1o(3,jc,itb)
            enddo
            enddo
          enddo
*VOPTION VEC
          do itb=ieoxyz,NTH+ieoxyz-1
            itb5=itb+(1-ieoxyz)*ist(mu)-    ieoxyz*ist(nu)
            itb6=itb-    ieoxyz*ist(nu)
*SOPTION UNROLL (3)
            do jc=1,COL
*SOPTION UNROLL (3)
            do ic=1,COL
              vclo(ic,jc,itb)=vclo(ic,jc,itb)
     &          -X2w56X6o(ic,1,itb)*ue(1,jc,itb6,iz6,iy6,ix6,nu)
     &          -X2w56X6o(ic,2,itb)*ue(2,jc,itb6,iz6,iy6,ix6,nu)
     &          -X2w56X6o(ic,3,itb)*ue(3,jc,itb6,iz6,iy6,ix6,nu)
     &          -conjg(uo(1,ic,itb5,iz5,iy5,ix5,nu))*X5w61X1o(1,jc,itb)
```

```
     &          -conjg(uo(2,ic,itb5,iz5,iy5,ix5,nu))*X5w61X1o(2,jc,itb)
     &          -conjg(uo(3,ic,itb5,iz5,iy5,ix5,nu))*X5w61X1o(3,jc,itb)
             enddo
             enddo
           enddo
1000    continue ! skip from if (mu.EQ.nu)
        enddo    ! end of do nu

*********************************
*  BB = BB - vcl
*********************************
*VOPTION VEC
        do itb=1-ieoxyz,NTH-ieoxyz
*SOPTION UNROLL (3)
          do jc=1,COL
*SOPTION UNROLL (3)
          do ic=1,COL
            BBe(ic,jc,itb,iz,iy,ix,mu)=BBe(ic,jc,itb,iz,iy,ix,mu)
     &                                -vcle(ic,jc,itb)
          enddo
          enddo
        enddo
*VOPTION VEC
        do itb=ieoxyz,NTH+ieoxyz-1
*SOPTION UNROLL (3)
          do jc=1,COL
*SOPTION UNROLL (3)
          do ic=1,COL
            BBo(ic,jc,itb,iz,iy,ix,mu)=BBo(ic,jc,itb,iz,iy,ix,mu)
     &                                -vclo(ic,jc,itb)
          enddo
          enddo
        enddo

      enddo
      enddo
      enddo ! end of do ix,iy,iz
      enddo ! end of do mu

      return
      end
#else
      subroutine force_clover
      return
      end
#endif
********** end if ifdef _CLOVER

```

引数 (入出力) **BBe, BBo :** even サイト、odd サイトの $F_\mu(n)$

引数 (入力) **XXe, XXo :** even サイト、odd サイトの式 (33) の中括弧の中身の負号が反対のもの $((\times)_{\mu\nu}(n)$ の前の方)

引数 (入力) **ue, uo :** even サイト、odd サイトのゲージリンク

引数 (入力) **wl34e, wl34o :** even サイト、odd サイトの 2 リンク (wl34)

引数 (入力) **wl98e, wl98o :** even サイト、odd サイトの 2 リンク (wl34)

82 行目: μ ループの開始

83-103 行目: x, y, z ループの開始

118-120 行目: x, y, z 座標の $n + \hat{\mu}$ の計算

122-135 行目: 作業配列のクリアー (ν ループに関する総和)

137-138 行目: ν ループの開始、$\mu = \nu$ はスキップ

139-150 行目: x, y, z 座標の $n + \hat{\mu} + \hat{\nu}$、$n + \hat{\nu}$、$n + \hat{\mu} - \hat{\nu}$、$n - \hat{\nu}$ の計算

152 行目: 2 次元座標 (μ, ν) から 1 次元座標 ipl の指定

153-250 行目: wl34、wl98、XX から作業配列へコピー

157-200 行目: $\mu < \nu$ の時のコピー (プログラムのコメントは間違っている!)。

204-249 行目: $\mu > \nu$ の時のコピー (プログラムのコメントは間違っている!)。$\mu > \nu$ は $\mu < \nu$ の時の負号が反対になる ($\sigma_{\mu\nu}$ のため)

152-335 行目: wl34、wl98 でカバーできない 2 リンクの計算

337-549 行目: リンクの間に $(\times)_{\mu\nu}(n)$ を挟んだステイプルの計算 (負号を回復)

550-551 行目: ν ループの終り

553-577 行目: $F_\mu(n)$ への足し込み

579-582 行目: μ, x, y, z ループの終り

3 PHMC について

参考文献

[1] A. D. Kennedy, Parallel Computing 25 (1999) 1311.