社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－HPC－118（12）
2008／12／17

両側ハウスホルダ変換に対するWilkinsonの著書AEP中の「技巧」について

村上 弘

首都大学東京 数理情報科学専攻

J.H. Wilkinsonは彼の著書AEP中で，通常の両側ハウスホルダ変換を用いた対称行列の三重対角化の方法に行列走査のパターンを変更する簡単なプログラム変形を加えることで，当時の二階層記憶システムの行列を格納する低速大容量外部記憶へのアクセスの回数を半減できる方法を文章で記述している．今回このWilkinsonの技巧による実対称行列の三重対角化の算法を高級言語による記述で再現し，さらに拡張として実非対称行列のヘッセンベルグ化の算法にも同じ概念による技巧を適用して，階層記憶システムを採用している現代の計算機上でプログラムによる記憶アクセスパターン変形の効果を調べる．

Experiments on Wilkinson's "Householder's process on a computer with a two-level store"

Murakami Hiroshi

Department of Mathematics and Information Sciences,
Tokyo Metropolitan University

The modification of the Householder transformation for the matrix eigenproblem which was described in the book "The Algebraic Eigenvalue Problem" by J.H. Wilkinson is experimented on today's microprocessors. Experiments are made not only for both symmetric problems but also for unsymmetric problems. The modification has a property to halve the frequency of memory transfers between the memory hierarchy. The data of experiments are presented.

1 はじめに

J.H. Wilkinsonは著書AEP中の一節[1]で，エルミート対称行列に対する通常のハウスホルダ三重対角化に「修正」を加えて二階層記憶（おそらく当時の計算機の大容量主記憶は磁気ドラム）の記憶階層間の転送を半減する方法を1頁程度で文章により記述している．しかしその方法は（例えば後のALGOL[2]のような）プログラミング言語では明示的に記述されていない．著者の知る範囲ではその後の文献や教科書等[2, 3, 4]にもこの技法は注意されていない．それはおそらく演算回数が変わらないことと，プログラムが複雑化し理解し難くなるからであろう．また，大容量の磁心記憶装置の普及で主記憶の機械的な動作が無くなり，メモリのデータ供給能力が浮動小数点演算能力を上回る時代が続いたため，行列が主記憶に納まる規模の問題に関してはこの方法を用いても経過時間が減らなくなったかとも思われる．そのため今日までこの「修正」は忘れられていたのであろう．今回この修正の原理が，対称行列の三重対角化のほかに非対称行列のヘッセンベルグ化にも適用できることを示す．（さらに最近のブロック行列のブロック・ハウスホルダ変換によるブロック三重対角化やブロック・ヘッセンベルグ化の処理にも同様にこの修正法の原理を適用すればデータの転送量が減らせることも容易に理解できる．）

2 算法について

Wilkinsonの修正は，行列要素に対する記憶参照と演算の実施スケジュールの最外側のループでの大局的な組み換えであり，個別のデータ要素が受ける算術演算は同一に保たれる．そのため（演算器内部にガード・ディジットのような隠れた情報ビットが存在したり，コンパイラの最適化による演算の結合順序等の変更がなければ）計算結果をビットパターンのレベルで元のものと同一に保つことが原理的には可能である．

2.1 修正法の簡単な説明

「修正」の原理を簡単に説明すると以下のようになる．本来の両側ハウスホルダ直交変換を一段ずつ適用する際に各段は三つの操作から成っている．いまこれらのk-段目での変換内部の変形操作をそれぞれOP1(k)，OP2(k)，OP3(k)と名付ける．第k-段

目に於いて，OP1(k) は行列の k-列の情報を入力として読み取り k-番目のハウスホルダ・ベクトル（鏡映軸ベクトル）を構成して三重対角行列の副対角要素 β_k を決定し，k-列の対角より下側にハウスホルダ・ベクトルを出力する．OP2(k) は行列を全く変更せずに，行列とハウスホルダ・ベクトルの積を計算する．OP3(k) はハウスホルダ・ベクトルと，OP2(k) の出力結果として得られたベクトルを用いて行列を更新する．OP2(k) と OP3(k) は共に行列の第 $(k+1)$-列から最後の列までを走査する．OP3(k) は OP2(k) が完了するまでは開始できない．それゆえ本来の両側ハウスホルダ変換は各段に於いて変換途中の行列全体の走査が二回生じる．

修正法では，隣り合う段の OP3(k) と OP2(k+1) を概ね重ねて実施するインターリーブ手法により行列走査の回数を減らす．OP2(k+1) は OP1(k+1) によって計算されたハウスホルダ・ベクトルだけに依存する．OP1(k+1) は OP3(k) がその処理の先頭部分で生成した第 $(k+1)$-列のみを使用する．OP3(k) の中で第 (k+1)-列が計算されたらその列に対して直ちに OP1(k+1) を実行してハウスホルダ・ベクトルを作ると，それを利用する OP2(k+1) は OP3(k) の完了を待たずに直ちに開始できる．OP3(k) と OP2(k+1) は，それらの処理の内部に於いて OP3(k) が行なう各列の更新結果を行列ベクトル積を作る OP2(k+1) が直ちに利用しながら両者を重ねて実行する．それゆえ本来の両側ハウスホルダ変換の各段での二回の行列の走査が一つに融合できるので，記憶階層間のメモリ転送が減らせることになる．

言葉による説明は愛昧で理解も難しいので，以下では実対称行列の三重対角化と実非対称行列のヘッセンベルグ化の双方について，修正前と修正後の両方の算法を具体的な例で示す．

2.2 プログラム言語による算法の記述例

対称及び非対称の場合の算法を Fortran90 言語で，ベクトル記法を避けて具体的に記述する．(#define で始まる行は Unix スタイルのマクロ処理の指令で，理解を容易とする為にマクロ展開をせずに示している．RK は倍精度実数の KIND 値で，intel コンパイラの場合は 8.) これらのコードは説明用のプロトタイプ（原型）である．実際の実験に用いたものは，対称問題では行列の対角を含む下半分だけを参照することを利用して行列の下半分だけを一次元配列として保持するようにコードを置き換えた．またコンパイラの能力を補うためにループ内の共通部分式や不変式を人手で抽出しソースコード最適化を行い性能を向上させている．また行列の第一次元やローカルのベクトルの配列のサイズを N とせずに NA と変更し，NA に N と異なる（N 以上の）値を与えれば，問題のサイズ N が特別な値で起きる性能低下を抑制できる．

配列には最初は入力の行列を格納し，途中で計算されたハウスホルダ・ベクトルを対角線を含まない下半分に順次格納する．格納されたハウスホルダ・ベクトルは，後で三重対角形やヘッセンベルグ形の固有ベクトルから元の行列の固有ベクトルへ逆変換をするために使用できる．

算法 A.1： 補助の手続き MKHVC は手続き SH1，SH2，UH1，UH2 から呼び出される．行列 A の第 K-列の (K+1) 行以降の値から三重対角形あるいはヘッセンベルグ形の副対角要素と，K 段目のハウスホルダ・ベクトルを計算する．ハウスホルダ・ベクトルは，第 K-列に，第 (K+1)-行以降に上書きして格納される．

引数 N は実正方行列 A の次数である．引数 X は N 次の実ベクトルで行列 A の第 K-列を意味する配列であり，第 (K+1) 行以降だけが参照される．固有ベクトルを後で求めるため際の逆変換に用いるハウスホルダ・ベクトルで上書きされる．引数 K は処理対象とする列の番号．引数 BETK は三重対角あるいはヘッセンベルグ系の第 K-番目の副対角要素である．

算法 A.2： 手続き SH1 は実対称問題の標準的なハウスホルダ法による三重対角化である．三重対角行列の主対角及び副対角が配列 P と R に格納される．

引数 N は実対称行列 A の次数である．引数 A は N 次の実対称行列で，行列の対角を含めた下半分だけが使われ，後の固有ベクトル計算で用いるハウスホルダ・ベクトルで上書きされる．引数 P は次数 N の実ベクトルで，三重対角行列の要素 $T_{K,K}$ が P(K) に格納される．引数 R は次数 N の実ベクトルで，三重対角行列の要素 $T_{K+1,K}$ が R(K) に格納される．

算法 A.3： 手続き SH2 は，SH1 に Wilkinson の修正を施したもので，引数は同一である．

算法 A.4： 手続き UH1（算法 A.4）は実非対称問題の標準的なハウスホルダ法による上ヘッセンベルグ化である．ヘッセンベルグ形の対角線を含めた上半分の部分が元の行列の配列に重ね書きされ，副対角要素は配列 BET に格納される．ローカルに二個の作業用配列 PL と PR をとっている．

引数 N は実正方行列 A の次数．引数 A は次数 N の実非対称行列，固有ベクトルを後で逆変換するためのハウスホルダ・ベクトルが対角を含まない行列の下半分に格納される．対角も含めた上半分に，上ヘッセンベルグ形の上半分が格納される．引数 BET は次数 N の実ベクトルで，ヘッセンベルグ形の副対角要素 $H_{K+1,K}$ が BET(K) に格納される．

算法 A.5： 手続き UH2 は，UH1 に Wilkinson の修正を施したもので引数は同一である．ローカルに作業用の配列 PL，PR，RL，RR をとっている．

算法 A.1

```
01 SUBROUTINE MKHVC(N,X,K,BETK)
02 IMPLICIT NONE
03 INTEGER,INTENT(IN)::N
04 REAL(KIND=RK),INTENT(INOUT)::X(N)
05 INTEGER,INTENT(IN)::K
06 REAL(KIND=RK),INTENT(OUT)::BETK
07 !--------------------------------
08 REAL(KIND=RK) X2,C,S,LAMBDA,DX
09 INTEGER I
10  X2=0.0_RK
11  DO I=K+1, N
12   X2=X2+X(I)**2
13  ENDDO
14  S=SQRT(X2)
15  IF(S==0.0_RK)THEN
16   BETK=0.0_RK
17   DO I=K+1,N
18    X(I)=0.0_RK
19   ENDDO
20  ELSE
21   C=X(K+1)
22   LAMBDA=ABS(C)*S
23   DX=SIGN(S,C)
24   X(K+1)=C+DX
25   BETK=-DX
26   X2=(X2+LAMBDA)*2
27   C=1.0_RK/SQRT(X2)
28   DO I=K+1,N
29    X(I)=X(I)*C
30   ENDDO
31  ENDIF
32 END SUBROUTINE MKHVC
```

算法 A.2

```
01 SUBROUTINE SH1(N,NA,A,P,R)
02 IMPLICIT NONE
03 INTEGER,INTENT(IN) :: N, NA
04 REAL(KIND=RK),INTENT(INOUT)::A(NA,N)
05 REAL(KIND=RK),INTENT(OUT)::P(N),R(N)
06 !-----------------------------------
07 #define Q(I)A(I,K)
08 REAL(KIND=RK) C
09 INTEGER I,J,K
10  LOOP: DO K=1,N-2
11   P(K)=Q(K)
12   CALL MKHVC(N,Q(1),K,R(K))
13   DO I=K+1,N
14    P(I)=0.0_RK
15   ENDDO
16   DO J=K+1,N
17    P(J)=P(J)+A(J,J)*Q(J)
18    DO I=J+1,N
19     P(I)=P(I)+A(I,J)*Q(J)
20     P(J)=P(J)+A(I,J)*Q(I)
21    ENDDO
22   ENDDO
23   C=0.0_RK
24   DO I=K+1,N
25    C=C+Q(I)*P(I)
26   ENDDO
27   DO I=K+1,N
28    P(I)=(Q(I)*C-P(I))*2
29   ENDDO
30   DO J=K+1,N
31    DO I=J,N
32     A(I,J)=A(I,J)+P(I)*Q(J)+Q(I)*P(J)
33    ENDDO
34   ENDDO
35  ENDDO LOOP
36  IF(N>1)THEN
37   P(N-1)=A(N-1,N-1)
38   R(N-1)=A(N,N-1)
39  ENDIF
40  P(N)=A(N,N)
41  R(N)=0.0_RK
42 END SUBROUTINE SH1
```

算法 A.3

```
01 SUBROUTINE SH2(N,NA,A,P,R)
02 IMPLICIT NONE
03 INTEGER,INTENT(IN)::N,NA
04 REAL(KIND=RK),INTENT(INOUT)::A(NA,N)
05 REAL(KIND=RK),INTENT(OUT)::P(N),R(N)
06 !-----------------------------------
07 #define QL(I)A(I,K-1)
08 #define Q(I)A(I,K)
09 REAL(KIND=RK) C
10 INTEGER I,J,K
11  IF(N<3)GOTO 999
12  K=1
13  P(K)=Q(K)
14  CALL MKHVC(N,Q(1),K,R(K))
15  DO I=K+1,N
16   P(I)=0.0_RK
17  ENDDO
18  DO J=K+1,N
19   P(J)=P(J)+A(J,J)*Q(J)
20   DO I=J+1,N
21    P(I)=P(I)+A(I,J)*Q(J)
22    P(J)=P(J)+A(I,J)*Q(I)
23   ENDDO
24  ENDDO
25  C=0.0_RK
26  DO I=K+1,N
27   C=C+Q(I)*P(I)
28  ENDDO
29  DO I=K+1,N
30   P(I)=(Q(I)*C-P(I))*2
31  ENDDO
32  LOOP: DO K=2,N-2
33   J=K
34   DO I=K,N
35    A(I,J)=A(I,J)+P(I)*QL(J)+QL(I)*P(J)
36   ENDDO
37   P(K)=Q(K)
38   CALL MKHVC(N,Q(1),K,R(K))
39   DO I=K+1,N
40    R(I)=0.0_RK
41   ENDDO
42   DO J=K+1,N
43    A(J,J)=A(J,J)+P(J)*QL(J)+QL(J)*P(J)
44    R(J)=R(J)+A(J,J)*Q(J)
45    DO I=J+1,N
46     A(I,J)=A(I,J)+P(I)*QL(J)+QL(I)*P(J)
47     R(I)=R(I)+A(I,J)*Q(J)
48     R(J)=R(J)+A(I,J)*Q(I)
49    ENDDO
50   ENDDO
51   C=0.0_RK
52   DO I=K+1,N
53    C=C+Q(I)*R(I)
54   ENDDO
55   DO I=K+1,N
56    P(I)=(Q(I)*C-R(I))*2
57   ENDDO
58  ENDDO LOOP
59  K=N-2
60  DO J=K+1,N
61   DO I=J,N
62    A(I,J)=A(I,J)+P(I)*Q(J)+Q(I)*P(J)
63   ENDDO
64  ENDDO
65  999 IF(N>1)THEN
66   P(N-1)=A(N-1,N-1)
67   R(N-1)=A(N,N-1)
68  ENDIF
69  P(N)=A(N,N)
70  R(N)=0.0_RK
71 END SUBROUTINE SH2
```

算法 A.4

```
01 SUBROUTINE UH1(N,NA,A,BET)
02 IMPLICIT NONE
03 INTEGER,INTENT(IN)::N,NA
04 REAL(KIND=RK),INTENT(INOUT)::A(NA,N)
05 REAL(KIND=RK),INTENT(OUT)::BET(N)
06 !-----------------------------------
07 #define Q(I)A(I,K)
08 REAL(KIND=RK) PL(N),PR(N)
09 REAL(KIND=RK) C
10 INTEGER I,J,K
11  LOOP: DO K=1,N-2
12   CALL MKHVC(N,Q(1),K,BET(K))
13   DO I=1,N
14    PR(I)=0.0_RK
15   ENDDO
16   DO J=K+1,N
17    DO I=1,K
18     PR(I)=PR(I)+A(I,J)*Q(J)
19    ENDDO
20    PL(J)=0.0_RK
21    DO I=K+1,N
22     PR(I)=PR(I)+A(I,J)*Q(J)
23     PL(J)=PL(J)+A(I,J)*Q(I)
24    ENDDO
25   ENDDO
26   C=0.0_RK
27   DO I=K+1,N
28    C=C+Q(I)*PR(I)
29   ENDDO
30   DO I=1,K
31    PR(I)=(-PR(I))*2
32   ENDDO
33   DO I=K+1,N
34    PR(I)=(Q(I)*C-PR(I))*2
35    PL(I)=(Q(I)*C-PL(I))*2
36   ENDDO
37   DO J=K+1,N
38    DO I=1,K
39     A(I,J)=A(I,J)+PR(I)*Q(J)
40    ENDDO
41    DO I=K+1,N
42     A(I,J)=A(I,J)+PR(I)*Q(J)+Q(I)*PL(J)
43    ENDDO
44   ENDDO
45  ENDDO LOOP
46  IF(N>1)THEN
47    BET(N-1)=A(N,N-1)
48  ENDIF
49  BET(N)=0.0_RK
50 END SUBROUTINE UH1
```

算法 A.5

```
01 SUBROUTINE UH2(N,NA,A,BET)
02 IMPLICIT NONE
03 INTEGER,INTENT(IN)::N,NA
04 REAL(KIND=RK),INTENT(INOUT)::A(NA,N)
05 REAL(KIND=RK),INTENT(OUT)::BET(N)
06 !-----------------------------------
07 #define QL(I)A(I,K-1)
08 #define Q(I)A(I,K)
09 REAL(KIND=RK) PL(N),PR(N),RL(N),RR(N)
10 REAL(KIND=RK) C
11 INTEGER I,J,K
12  IF(N<3)GOTO 999
13  K=1
14  CALL MKHVC(N,Q(1),K,BET(K))
15  DO I=1,N
16   PR(I)=0.0_RK
17  ENDDO
18  DO J=K+1,N
19   DO I=1,K
20    PR(I)=PR(I)+A(I,J)*Q(J)
21   ENDDO
22   PL(J)=0.0_RK
23   DO I=K+1,N
24    PR(I)=PR(I)+A(I,J)*Q(J)
25    PL(J)=PL(J)+A(I,J)*Q(I)
26   ENDDO
27  ENDDO
28  C=0.0_RK
29  DO I=K+1,N
30   C=C+Q(I)*PR(I)
31  ENDDO
32  DO I=1,K
33   PR(I)=(-PR(I))*2
34  ENDDO
35  DO I=K+1,N
36   PR(I)=(Q(I)*C-PR(I))*2
37   PL(I)=(Q(I)*C-PL(I))*2
38  ENDDO
39  LOOP: DO K=2,N-2
40   J=K
41   DO I=1,K-1
42    A(I,J)=A(I,J)+PR(I)*QL(J)
43   ENDDO
44   DO I=K,N
45    A(I,J)=A(I,J)+PR(I)*QL(J)+QL(I)*PL(J)
46   ENDDO
47   CALL MKHVC(N,Q(1),K,BET(K))
48   DO I=1,N
49    RR(I)=0.0_RK
50   ENDDO
51   DO J=K+1,N
52    DO I=1,K-1
53     A(I,J)=A(I,J)+PR(I)*QL(J)
54     RR(I)=RR(I)+A(I,J)*Q(J)
55    ENDDO
56    I=K
57     A(I,J)=A(I,J)+PR(I)*QL(J)+QL(I)*PL(J)
58     RR(I)=RR(I)+A(I,J)*Q(J)
59    RL(J)=0.0_RK
60    DO I=K+1,N
61     A(I,J)=A(I,J)+PR(I)*QL(J)+QL(I)*PL(J)
62     RR(I)=RR(I)+A(I,J)*Q(J)
63     RL(J)=RL(J)+A(I,J)*Q(I)
64    ENDDO
65   ENDDO
66   C=0.0_RK
67   DO I=K+1,N
68    C=C+Q(I)*RR(I)
69   ENDDO
70   DO I=1,K
71    PR(I)=(-RR(I))*2
72   ENDDO
73   DO I=K+1,N
74    PR(I)=(Q(I)*C-RR(I))*2
75    PL(I)=(Q(I)*C-RL(I))*2
76   ENDDO
77  ENDDO LOOP
78  K=N-2
79  DO J=K+1,N
80   DO I=1,K
81    A(I,J)=A(I,J)+PR(I)*Q(J)
82   ENDDO
83   DO I=K+1,N
84    A(I,J)=A(I,J)+PR(I)*Q(J)+Q(I)*PL(J)
85   ENDDO
86  ENDDO
87  999 IF(N>1)THEN
88   BET(N-1)=A(N,N-1)
89  ENDIF
90  BET(N)=0.0_RK
91 END SUBROUTINE UH2
```

3 実験

Intel Core2Quad： 計算システムは Intel Core2Quad Q6600 2.4GHz で，CPU パッケージはダイを 2 組含み，各ダイごとにコアが 2 個と共有 L2 キャッシュ4M バイトを持つ．本実験はコア 1 個

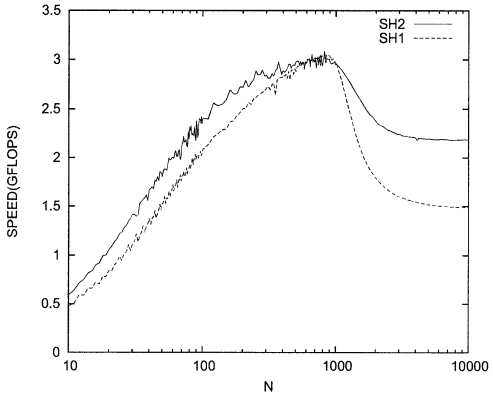

図 1: 実効演算速度，対称行列，Core2Quad.

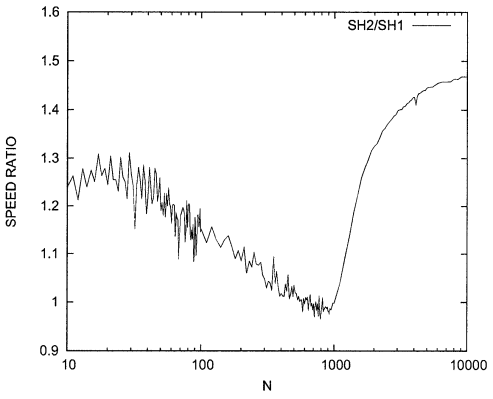

図 2: 速度向上比，対称行列，Core2Quad.

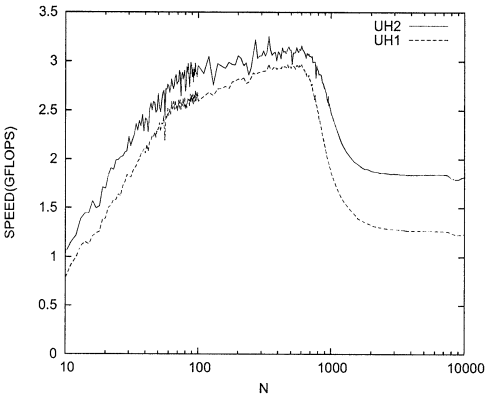

図 3: 実効演算速度，非対称行列，Core2Quad.

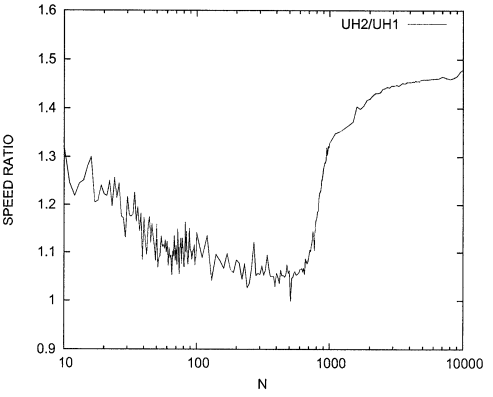

図 4: 速度向上比，非対称行列，Core2Quad.

だけを用いて計算した．コンパイラは intel fortran ver.10.1，オプションは"-xT -ipo -O3 -fast"である．

SH1 は実対称問題のハウスホルダ法の標準的な算法，SH2 は Wilkinson の修正を加えた算法である．SH1 と SH2 について，実対称なランダム行列の次数 N の範囲を 10 から 10000 まで変えて三重対角化の経過時間を測定した．三重対角化の浮動小数点演算量を $(4/3)N^3$ として実効演算速度を求めたグラフが図 1 である．SH1 に対する SH2 の実効速度向上比のグラフが図 2 である．行列の下半分がダイ上の 4MB の L2 キャッシュに納まらなくなるサイズ $N = 1000$ を越えた付近から修正による速度比が上昇し，$N = 10000$ では 1.45 倍程度に達している．比較的小規模の行列で N が 10 から 100 程度の範囲でも 1.1 倍〜1.2 倍程度の向上が得られているのは，修正による L1 と L2 のキャッシュ間の転送回数の削減とレジスタの有効利用に依るものであろう．

UH1 は実非対称問題に対するハウスホルダ法の標準的な算法，UH2 は Wilkinson の修正を加えた算法である．UH1 と UH2 について，実非対称なランダム行列を行列の次数 N の範囲を 10 から 10000 まで変えてヘッセンベルグ化の経過時間を測定し，浮動小数点演算量を $(10/3)N^3$ とした実効演算速度のグラフが図 3 である．UH1 に対する UH2 の速度比のグラフが図 4 である．正方行列全体がダイ上の 4MB の L2 キャッシュに納まらなくなるサイズ $N = 700$ を越えた付近から修正による速度比が上昇し，$N = 10000$ では 1.45 倍程度に達している．比較的小規模の行列で N が 10 から 100 程度の範囲でも 1.1 倍〜1.2 倍程度の向上が得られている．

AMD Opteron: 計算システムは東大 T2K，日立 HA8000（AMD Opteron 8356，2.3GHz）で，CPU パッケージはダイ 1 個でコア 4 個，各コア毎に専有 L2 キャッシュ512K バイト，4 個のコアで共有する L3 キャッシュ2M バイトを持つ．本実験はコア 1 個だけ

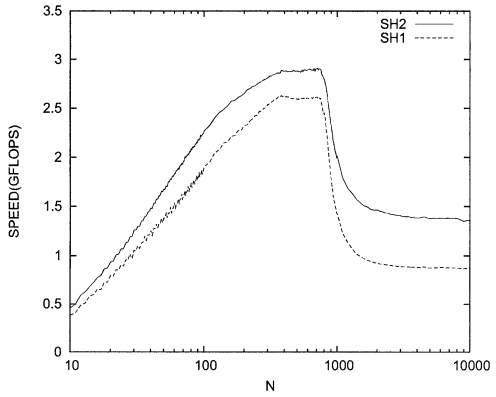

図 5: 実効演算速度，対称行列，Opteron.

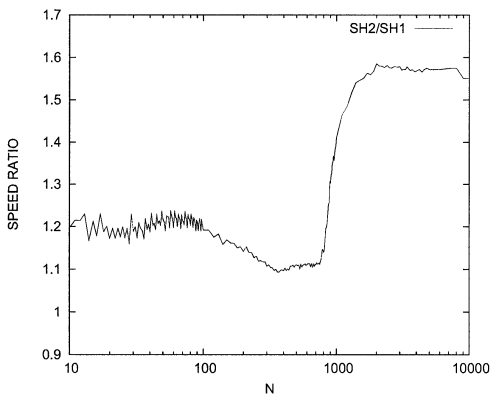

図 6: 速度向上比，対称行列，Opteron.

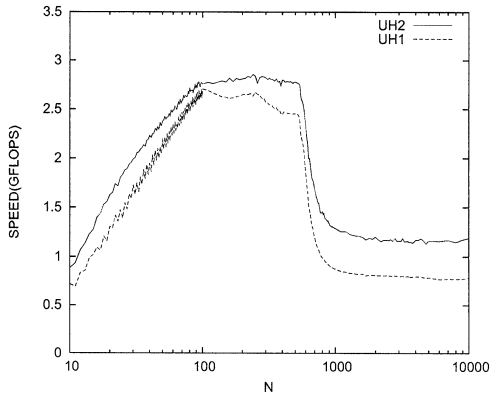

図 7: 実効演算速度，非対称行列，Opteron.

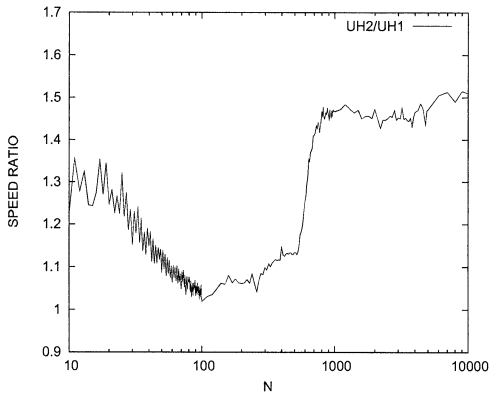

図 8: 速度向上比，非対称行列，Opteron.

で計算した．コンパイラは intel fortran ver.10.1，オプションは"-axT -ipo -O3"である．

SH1 と SH2 の実対称ランダム行列を三重対角化する実効演算速度のグラフが図 5 で，速度の向上比のグラフが図 6 である．行列の下半分が 2MB の L3 キャッシュに納まらなくなるサイズ $N = 700$ を越えた付近から修正による速度比が上昇し，$N = 10000$ では 1.55 倍程度に達している．比較的小規模の行列で N が 10 から 100 程度の範囲でも 1.2 倍程度の速度向上が得られている．

UH1 と UH2 について，実非対称ランダム行列をヘッセンベルグ化する実効演算速度のグラフが図 7 で，実効速度の向上比のグラフが図 8 である．行列全体が 2MB の L3 キャッシュに納まらなくなるサイズ $N = 500$ を越えた付近から修正による速度比が上昇し，$N = 10000$ では 1.5 倍程度に達している．比較的小規模の行列で N が 10 から 30 の範囲でも 1.2 倍程度の向上が得られている．

結論： 両側ハウスホルダ変換に対する Wilkinson の修正は，記憶階層間の転送を減らし，性能向上効果があることを確認できた．

参考文献

[1] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, (Chap.5. Sec.32), Oxford Univ. Press, 1965.

[2] J.H. Wilkinson and C. Reinsch, eds., *Handbook for Automatic Computation, Vol.2, Linear Algebra*, Springer-Verlag, New York, 1971.

[3] G.H. Golub and C.F. Van Loan, *Matrix Computations*, The Johns Hopkins Univ. Press, Baltimore, Maryland, 1983.

[4] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *Numerical Recipes*, Cambridge Univ. Press, New York, 1986.