ハイパフォーマンス
コンピューティング 55－8
（1995．3．9）

# RISCワークステーション上の NUMPAC 行列乗算の性能

山本茂義，小畑繁樹 *，村上明徳 †，片桐秀樹 ‡，秦野やす世 §

中京大学教養部（〒470-03 愛知県豊田市貝津町床立101）
* 愛知技術短期大学電子工学科
† 三菱化学横浜総合研究所計算科学研究所
‡ 電子技術総合研究所材料科学部材料制御研究室
§ 中京大学情報科学部認知科学科

数学ライブラリNUMPACの中に含まれる行列乗算プログラム MULMMW を RISCワークステーション向けに最適化した．いくつかの RISCワークステーションでその性能を評価した．LINPACK性能 12.8 MFLOPS の HP Apollo 9000 シリーズ 700 モデル 712/60 上で 34.7 MFLOPS の速度を得た．中間積法に基づくプログラムであるが，内積法との関連についても比較検討する．

# Performance Evaluation of Matrix Multiplication Program of NUMPAC Mathematical Package on RISC-based Workstations

Shigeyoshi Yamamoto, Shigeki Obata *, Akinori Murakami †,
Hideki Katagiri ‡, and Yasuyo Hatano §

*Faculty of Liberal Arts, Chukyo University (Kaizu, Toyota, 470-03 Japan)*
**Aichi Junior College of Technology*
*† Yokohama Research Center, Mitsubishi Chemical Corporation*
*‡ Electrotechnical Laboratory, AIST, MITI*
*§ School of Computer and Cognitive Sciences, Chukyo University*

We have optimized matrix multiplication program MULMMW of NUMPAC mathematical package for the RISC architecture. We evaluated its performance on several RISC-based workstations. The speed on the HP Apollo 9000 series 700 model 712/60 of which LINPACK performance is 12.8 MFLOPS amounts to 34.7 MFLOPS. The program is based on the intermediate product algorithm. We compare it with the inner product algorithm.

## 1．はじめに

数学ライブラリNUMPACベクトル版 [1] はスーパーコンピュータ向けに最適化された数値計算ライブラリである．線型計算ルーチン18種類50個のプログラムが含まれている．

RISCアーキテクチャに基づくワークステーション（略記 WS）が各社から市場に提供されており，ハイエンド機では初期のスーパーコンピュータを凌駕する性能を誇る．しかしながら，メーカー提供の数学ライブラリは必ずしも充実しているとは言えない [2]．RISC-WS 用の汎用数学ライブラリに対する需要は高いと言える．

また，一部の機種以外はコンパイラもインテリジェントではなく，性能を引き出すためにはコンパイラの最適化を補助する目的で技巧的なコーディングをせざるをえないのが現状であり，ユーザの負担は大きい．

本研究では NUMPACライブラリに含まれている行列乗算のプログラム MULMMW をまず取り上げ，RISCアーキテクチャ向けに最適化してみた．

行列乗算の高速化については本研究会においても既に前野・太田 [3] による報告がある．文献 [3] は内積法に基づくアルゴリズムであるが，我々は中間積法に基づいて最適化を試みた．

なお，大次元行列の乗算に関しては V. Strassen [4] によるアルゴリズムがあるが，ここでは扱わない．

## 2．最適化の方法

行列乗算 $C = AB$ を

$$c_{ij} = c_{ij} + a_{ik} * b_{kj}$$

と記述すると，ループを jki の順に回す中間積法（略記 IT）が，ijk の順に回す内積法（略記 IN）よりもスーパーコンピュータにおいては効率が良いことが知られている [5]．元々のNUMPACベクトル版の行列乗算プログラム MULMMW も中間積法に基づいている．

RISC-WS においても，素朴にコーディングした場合，中間積法の方が最内側ループのストライドが1であり，かつ配列 c，a の添字が行方向に走るため効率が良い．実際，Fig.1 のHP Apollo 9000 シリーズ 700 モデル 712/60（略記 HP712），Fig.2 の HP735，Fig.5 の IBM RS/6000 モデル 590（略記 RS590）においても，その傾向が示されている．

しかしながら，RISC-WS 上のコンパイラは一部の例外を除いてスーパーコンピュータの自動ベクトル化ほどには強力ではなく，素朴なプログラムのままでは高性能は得られない．RISCアーキテクチャの特質を活かすための技巧的なプログラミングが要求される．

ここで用いる高速化の方法は既に開発され，よく知られているものである．

高速化の第1手法はスーパーコンピュータでも定石となっているループ・アンローリングである．これによりレジスタの効率的使用が達成される．我々が最適化した FORTRAN プログラム（紙面の都合によりコメントおよび引数の整合性チェックの部分は省略してある）を Program 1 に示すが，j および k について，4重にネストしたアンローリングを行っている．コードが長くなっているのはアンローリングの終端処理のためであり，最初に現れる深度4のループまでが主要部分である．

変数名が r で始まる ra*，rb**，rc* が浮動小数点レジスタ（略記 FPR）に割り当てられるべき変数である．合計24個あるが，rc* については機種によってはソースレジスタとデスティネーションレジスタを別個に割り当てる場合もありうる．多くの RISC-WS は32個の64ビットFPRを有するので4×4のアンローリングが最適である．

Fujitsu S-4/10モデル30（SUN SPARC station10 model 30 に相当，略記 SS10）については FPR が16個分しかないため，4×4では FPR を使い切ってしまうので，アンローリングの段数を減らす方が効率が良い．

配列 b は j に関して列方向にアクセスされるが，j に関するアンローリングにより，列方向が自動的にブロッキングされることになり効率が良い．

しかしながら，j と k に関するアンローリングを施しただけでは，配列 c に対するロード・ストアが最内側ループに現れてしまい，

キャッシュミス・TLBミスの問題が残る．これを解決するためにキャッシュコピーイングの方法 [6] を使う．

配列 c の一部をキャッシュ上に確保しておくために作業領域wcを用いる．この配列は 4×128 と宣言しているが，用いる機種のキャッシュのサイズに応じて，NZ に大きな値を設定することにより大きくできる．

我々のアルゴリズムは最内側ループに並列処理可能な演算が多いため演算器・FPRが多数あるほど有利になる．

欠点は最内側に配列wcに対するロード・ストア，大次元配列 a に対するロードが残る点である．特に配列へのストアは機種によっては性能低下を引き起こす可能性がある．性能がキャッシュの属性に強く依存することになる点が不利である．この点，内積法に基づくアルゴリズムは最内側ループに配列へのストアがなく優れている．

## 3．テスト環境

我々が RISC-WS 用に最適化したプログラム（略記 SY）をいくつかの機種でテストした．比較のために，前野・太田 [3] のプログラム（略記 MO），中間積法による素朴なプログラム（IT），内積法による素朴なプログラム（IN）もCPU時間を測定した．

ただし，前野・太田のプログラムは元々C言語で書かれているが，各社の FORTRAN コンパイラの性能を比較するために我々が FORTRAN に書き換えたもの [2] を使用した．ただし，MULMMW プログラムと同じサブルーティン形式に変更し若干の補正を加えてある．C と FORTRAN では多次元配列の添字の走行順が異なることを考慮し，配列 d に対して添字を入れ換えてある．しかし，RS590以外は機種毎の調整（tuning）はしていない．そのためMOプログラムについては性能を過小評価しているおそれがあることをお断りしておく．

性能測定の対象とした機種は HP712, HP735, DEC10000, SS10, RS550, RS580, RS590, RS990, Indy R4600PC, Indigo2 R4400SC である．参考のためメインフレーム Fujitsu M-1800 についてもテストした．各WS の公称性能（SPECfp92, LINPACK）を Table 1 にまとめる．

用いたコンパイラのバージョンは，HPは Ver.9.16，RS590では Ver.3.02 である．SS10では Fujitsu Fortran90 Ver.1.0 を使用している．

コンパイルコマンドは HP712では f77 +U77 +O3 +e，HP735では f77 +U77 +O3，RS590 では xlf -qarch=pwr2 -qtune=pwr2 -qhot である．SS10では f90 -O3 -X f7 -KTMS である．

行列 a，b，c はメインプログラムで全て 1001×1001 の大きさで確保している．

行列の次元数を $N$ とすると，総演算数は $2N^3$ であり，これより計算速度をMFLOPSに換算できる．$N$ を100から1000まで50ずつ増やして測定した．

## 4．性能評価

HP712 では Fig.1 に示されるように，我々のプログラム（SY）はMOプログラムと同等の高性能を示し，N=1000 で 34.7 MFLOPS に達している．この機種の LINPACK [7] の性能が 12.8 MFLOPS であることから考えて，成功と言ってよいであろう．

しかしながら，HP735 では Fig.2 に見られるように，N=100 をピークに直線的に性能が降下し，N=250 以降，性能が 53 MFLOPS 前後で停滞してしまう．これは HP735 のキャッシュがダイレクトマップ方式のため，キャッシュの競合（thrashing）[8] が生じているためと考えている．キャッシュの属性に性能が依存するという，SYプログラムの欠点が顕著に現れている例である．MOプログラムでは，このような性能低下は生じない．

HP712もダイレクトマップ方式であるにも関わらず性能低下が生じないのは，PA-RISC7100LC が PA-RISC7100 に比しキャッシュの強化がなされているためであると考えている．

Fig.3 のDEC10000 もダイレクトマップキャッシュであり，この場合も HP735程には極端ではないが，同様の性能低下の傾向を示す．

SS10 は 4 重連想キャッシュを備えており，Fig.4 の SY(mod) に見られるようにアンローリングを 3×3 に変更すると性能が向上する．

Fig.5 の SY(mod) はSYプログラムを RS590 用に調整を行った結果である．NZ を 128 から 1024 に拡張している．N=1000 で 215.9 MFLOPS を与える．ハーバードアーキテクチャの 4 重連想キャッシュが性能向上に寄与していると思われる．

また，POWER2 ファミリには quadword load/store [9] と呼ばれる命令が用意されている．これは連続する 2 個の倍精度データを連接する 2 個の FPR に高速にロード／ストアする命令であり，性能向上に大きく寄与する．この命令を活かすには，配列wcの第一添字を i ではなく j の並びにして，連続的に 2 個ずつアクセスする方が有利である．

同様の調整をMOプログラムに適用し，かつ，ソフトウエアパイプライニングを廃したものが MO(mod) であり，N=1000 で 245.2 MFLOPS を与える．

RS590 では特にコンパイラがインテリジェントな点が注目に値する．最適化レベル 3 のみでは，素朴に書かれた内積法のプログラム（IN）は N が大きくなるにつれ性能が極端に低下し，N = 1000 で 4.3 MFLOPS にまで落ちる．ところが，IN あるいは IT（中間積法）プログラムにコンパイルオプション hot を適用すると，Fig.5 の IT(hot) で表されるグラフとなり，N=1000で 185.7 MFLOPS にまで性能が向上する．

この hot オプションは，ループアンローリング，多重ループにおける外側と内側のループの交換など，ループ・配列に関する最適化を自動的に行う．このため，IN でも IT でも同等の性能向上が得られる．

Fig. 5 で ESSL と記されたグラフはIBM提供の科学技術計算ライブラリESSL [10] の中の行列乗算 サブルーティン DGEMUL を用いた結果である．このルーティンは非常に高速で，N=1000 で 252.7 MFLOPS を与える．この値は RS590 の理論ピーク性能（TPR）266 MFLOPS に迫るものである．以上のように，IBM の FORTRAN環境の生産性は高い．

Indy，Indigo2 では Table 1 に示すように，SYプログラムはMOプログラムとほぼ同等の性能であった．

## 5．おわりに

中間積法に基づいて，RISCアーキテクチャ向けに行列乗算プログラムを最適化した．HP712 では高性能が得られている．しかし，キャッシュの属性に性能がかなり依存し，一部の機種では性能が伸びない場合がある．

中間積法，内積法のいづれに基づくアルゴリズムを採択するかは，キャッシュの性能，FPRの個数，演算器の並列性等に依存するが，現行の RISC-WS では内積法が若干有利であると言えよう．

HPの次期CPUである PA-RISC7200 [11] では 2KB の完全連想データキャッシュ，MIPS の次期CPUの T5 [12] では 32KB の 2 重連想データキャッシュと64個の64ビットFPRが計画されており，性能のキャッシュ依存性は解消される方向にある．

## 6．謝辞

性能測定の一部で，分子科学研究所電子計算機センター，名古屋大学大型計算機センターを利用した．

## 7．参考文献

[1] 二宮，秦野：中部大学経営情報学部論集，Vol.4，No.2，p.61 (1988)．秦野，二宮：情報処理学会研究報告（数値解析18-1），Vol.86，No.68，pp.1-8 (1986)．

[2] 山本，秦野：*SCCS TECHNICAL REPORT*，No.94-2-03，pp.1-31 (1994)．

[3] 前野，太田：情報処理学会研究報告（93-HPC-49），Vol.93，No.89，pp.1-8 (1993)．

[4] V. Strassen: *Numer. Math.*, 13, 354-356 (1969)．

[5] 島崎眞昭：スーパーコンピュータとプログラミング，計算機科学／ソフトウエア技術講座 9（共立出版，1989）．

[6] M. S. Lam, E. E. Rothberg, M. E. Wolf: The Cache Performance and Optimizations of Blocked Algorithms, *ASPLOS IV, ACM*, April 63-74 (1991)．

[7] J. Dongarra: The Performance of Computers on the LINPACK, Colloquium

at IRS, May 12-13 (1988).
[8] J. Hennessy, D. Patterson: Computer Architecture: A Quantitative Approach (Morgan Kaufmann, 1990). D. F. Bacon: *BYTE*, Aug 79-86 (1994).
[9] S. W. White, S.Dhawan: *IBM J. Res. Develop.*, **38**, 493-502 (1994).
[10] R. C. Agarwal, F. G. Gustavson, M. Zubair: *IBM J. Res. Develop.*, **38**, 563-575 (1994).
[11] D. Pountain: *BYTE*, Aug 185-186 (1994).
[12] T. R. Halfhill: *BYTE*, Nov 123-128 (1994).

**Table 1.** Performance of RISC-based workstations.

| System | Clock (MHz) | SPECfp92 | MFLOPS | | | | |
|---|---|---|---|---|---|---|---|
| | | | LINPACK | TPP [a] | TPR [b] | SY [c] | MO [d] |
| HP712 | 60 | 79 | 12.8 | | | 34.7 | 28.9 |
| HP735 | 99 | 149.8 | 41 | 107 | 198 | 53.9 | 105.7 |
| DEC10000 | 200 | 193.6 | 43 | 155 | 200 | 58.1 | 82.9 |
| SS10 | 36 | 54 | 9.3 | | | 10.3 | 11.8 |
| | | | | | | 11.3 * | |
| RS550 | 41 | 71.3 | 25.2 | | | 55.7 | 58.3 |
| RS580 | 62.5 | 134.6 | 38.1 | 104 | | 81.5 | 85.2 |
| RS590 | 66.5 | 242.4 | 130.4 | 236 | 266 | 196.6 | 186.8 |
| | | | | | | 215.9 * | 245.2 * |
| RS990 | 71.5 | 260.4 | 140.3 | | 286 | 208.0 | 196.6 |
| Indy | 50/100 | 49.9 | 11 | | | 14.1 | 13.5 |
| Indigo2 | 75/150 | 86 | 24 | | | 29.9 | 32.9 |
| M-1800 | | | 33.5 | | | 40.0 | 36.2 |

a) Toward Peak Performance. LINPACK 1000×1000.
b) Theoretical peak rate.
c) Matrix multiplication (N=1000) by SY program. The asterisk denotes SY(mod).
d) Matrix multiplication (N=1000) by MO program. The asterisk denotes MO(mod).

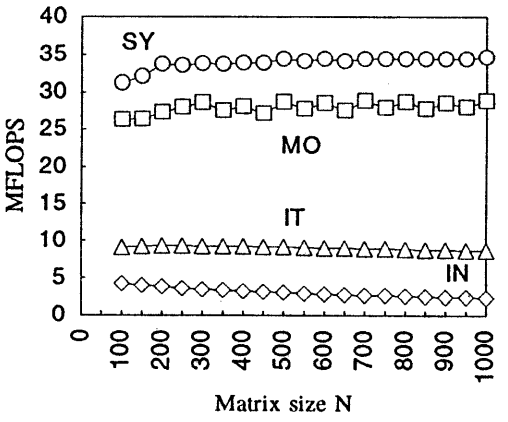


**Figure 1.** HP9000-712/60.

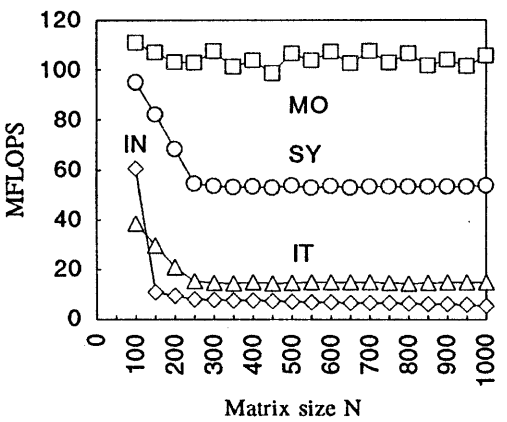


**Figure 2.** HP9000-735

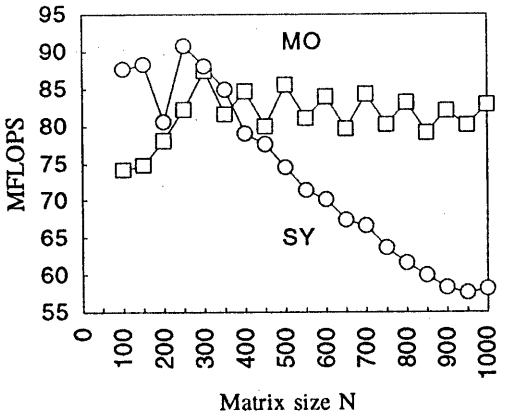


**Figure 3.** DEC10000.

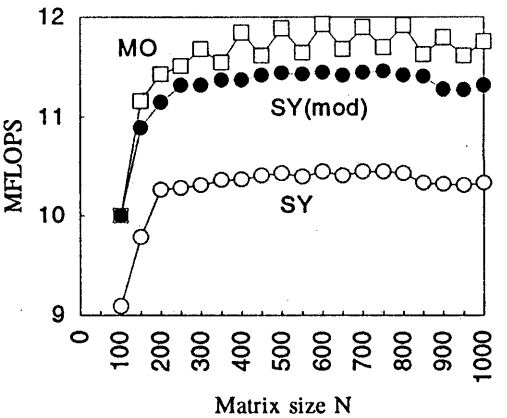


**Figure 4.** S-4/10 model 30.

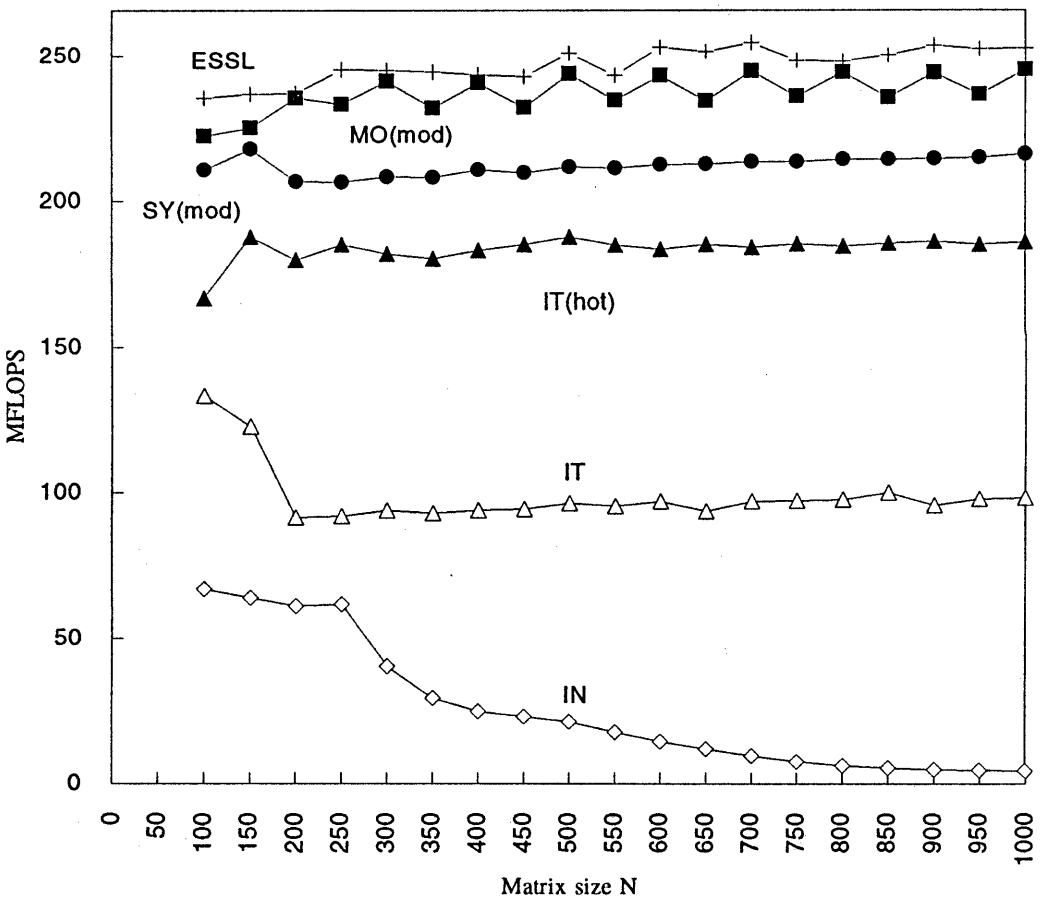


**Figure 5.** IBM RS/6000 model 590.

**Program 1.** MULMMW program optimized for the RISC architecture.

```
      subroutine mulmmw
     + (a,b,c,KA,KB,KC,L,M,N,ILL)
      implicit real*8 (a-h,o-z)
      parameter (NZ = 128)
      parameter (NJ = 4, NK = 4)
      parameter (NJ1 = NJ - 1)
      parameter (NZ1 = NZ - 1)
      real*8 wc(0:NJ1,0:NZ1)
      real*8 a(KA,M), b(KB,N)
      real*8 c(KC,N)
      LL = (L - 1) / NZ + 1
      MM = (M / NK) * NK
      NN = (N / NJ) * NJ
      do j = 1, NN, NJ
        do ii = 1, LL
          is = (ii - 1) * NZ + 1
          ie = MIN(ii * NZ, L)
          do i = is, ie
            it = i - is
            wc(0,it) = 0.d0
            wc(1,it) = 0.d0
            wc(2,it) = 0.d0
            wc(3,it) = 0.d0
          end do
          do k = 1, MM, NK
            rb00 = b(k,  j)
            rb10 = b(k+1,j)
            rb20 = b(k+2,j)
            rb30 = b(k+3,j)
            rb01 = b(k,  j+1)
            rb11 = b(k+1,j+1)
            rb21 = b(k+2,j+1)
            rb31 = b(k+3,j+1)
            rb02 = b(k,  j+2)
            rb12 = b(k+1,j+2)
            rb22 = b(k+2,j+2)
            rb32 = b(k+3,j+2)
            rb03 = b(k,  j+3)
            rb13 = b(k+1,j+3)
            rb23 = b(k+2,j+3)
            rb33 = b(k+3,j+3)
            do i = is, ie
              it = i - is
              rc0 = wc(0,it)
              rc1 = wc(1,it)
              rc2 = wc(2,it)
              rc3 = wc(3,it)
              ra0 = a(i,k)
              ra1 = a(i,k+1)
              ra2 = a(i,k+2)
              ra3 = a(i,k+3)
              rc0 = rc0
                  + ra0 * rb00
     +            + ra1 * rb10
     +            + ra2 * rb20
     +            + ra3 * rb30
              rc1 = rc1
                  + ra0 * rb01
     +            + ra1 * rb11
     +            + ra2 * rb21
     +            + ra3 * rb31
              rc2 = rc2
                  + ra0 * rb02
     +            + ra1 * rb12
     +            + ra2 * rb22
     +            + ra3 * rb32
              rc3 = rc3
                  + ra0 * rb03
     +            + ra1 * rb13
     +            + ra2 * rb23
     +            + ra3 * rb33
              wc(0,it) = rc0
              wc(1,it) = rc1
              wc(2,it) = rc2
              wc(3,it) = rc3
            end do
          end do
          do k = MM+1, M
            rb00 = b(k,j)
            rb01 = b(k,j+1)
            rb02 = b(k,j+2)
            rb03 = b(k,j+3)
            do i = is, ie
              it = i - is
              rc0 = wc(0,it)
              rc1 = wc(1,it)
```

```
          rc2 = wc(2,it)
          rc3 = wc(3,it)
          ra0 = a(i,k)
          rc0 = rc0
     +        + ra0 * rb00
          rc1 = rc1
     +        + ra0 * rb01
          rc2 = rc2
     +        + ra0 * rb02
          rc3 = rc3
     +        + ra0 * rb03
          wc(0,it) = rc0
          wc(1,it) = rc1
          wc(2,it) = rc2
          wc(3,it) = rc3
        end do
      end do
      do i = is, ie
        it = i - is
        c(i,j)   = wc(0,it)
        c(i,j+1) = wc(1,it)
        c(i,j+2) = wc(2,it)
        c(i,j+3) = wc(3,it)
      end do
    end do
  end do
  do j = NN+1, N
    do ii = 1, LL
      is = (ii - 1) * NZ + 1
      ie = MIN(ii * NZ, L)
      do i = is, ie
        it = i - is
        wc(0,it) = 0.d0
      end do
      do k = 1, MM, NK
        rb00 = b(k,  j)
        rb10 = b(k+1,j)
        rb20 = b(k+2,j)
        rb30 = b(k+3,j)
        do i = is, ie
          it = i - is
          rc0 = wc(0,it)
          ra0 = a(i,k)
          ra1 = a(i,k+1)
          ra2 = a(i,k+2)
          ra3 = a(i,k+3)
          rc0 = rc0
              + ra0 * rb00
     +        + ra1 * rb10
     +        + ra2 * rb20
     +        + ra3 * rb30
          wc(0,it) = rc0
        end do
      end do
      do k = MM+1, M
        rb00 = b(k,j)
        do i = is, ie
          it = i - is
          rc0 = wc(0,it)
          ra0 = a(i,k)
          rc0 = rc0
     +        + ra0 * rb00
          wc(0,it) = rc0
        end do
      end do
      do i = is, ie
        it = i - is
        c(i,j) = wc(0,it)
      end do
    end do
  end do
  return
  end
```