OUKA
The University of Osaka
Institutional Knowledge Archive

Title	スーパーコンピューターの効能 : 原子核物理に於ける散乱問題の例題
Author(s)	佐藤, 透; 田村, 圭介
Citation	大阪大学大型計算機センターニュース. 1988, 69, p. 45-62
Version Type	VoR
URL	https://hdl.handle.net/11094/65777
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

スーパーコンピューターの効能
原子核物理に於ける散乱問題の例題

大阪大学理学部物理

佐藤　透・田村圭介

1 はじめに

計算機の発展にともない原子核理論の分野でも大型の計算が可能になりより精密な議論が可能になってきた。我々の研究においては、新たな研究課題に取り組むごと新たにプログラムを作成し、また修正を加えつつ物理的内容を調べるという計算機の使い方をしている。従ってプログラムの見やすさ、手軽さが重要であり、特に会話形処理は非常に有効になる。逆に、時間を、スーパーコンピューターのため工夫を凝らすために費やすことには敷居が高く感じられる。しかし高速の計算が要求されている最近では、スーパーコンピューターを有効に使うことも必要になってきている。

スーパーコンピューターを有効に使うために、もちろんプログラム技法も重要であろうが、また問題に対するアプローチ、定式化による因子も大きいであろう。ここでは、原子核に於ける散乱問題に関連してスーパーコンピューターに於いて有効であろうと思われる二つの例題を調べてみた。一つは連分数法による Schroedinger 方程式の散乱波の解法である〔1〕。この方法は三核子問題に適応され非常に強力であることが示されている。この方法は、線形の微分方程式をある境界条件で解く問題にかなり一般的に用いることが出来ると思われる。二つめは高エネルギー原子核ー原子核散乱の Glauber 理論による解析である〔2〕。Glauber 理論によれば、原子核ー原子核散乱振幅は原子核の構成粒子である核子の散乱振幅で与えられ、またハドロンー原子核散乱の前方断面積に対して有効であることが知られている。原子核ー原子核散乱では計算時間が入射核と標的核の核子数の増大に従って長くなり高速な計算が必要になる。

ところで我々素人は、ベクトルプロセッサーをいかす計算とは、メモリイサイズを犠牲にして配列を用い判定、分岐を含まないＤＯループにしてしまえば高速になるであろうと大ざっぱに考えている。では実際簡単なプログラムでどの程度高速になりうるのか、計算センターニュース等〔3〕に書かれている高速化技法がどの程度の効果があるのか、まず調べてみる。　定量的にどの程度高速になるかといったデータは、我々の知る範囲ではないので、高速化を計る際の目安として有用であると思われる。２章では簡単なＤＯループで時間測定を行った結果を示す。次に３章で連分数法による散乱問題の解法、４章で原子核ー原子核散乱の Glauber による解析についてベクトルプロセッサーの威力について我々の結果を示す。最後に５章で我々の得た教訓をまとめる。

2　ＳＸ２Ｎ－ＡＰにおけるベクトル化の効果

まず簡単なＤＯループを用いて、ベクトル化率100％の場合、実際ベクトルプロセッサーにより何倍高速になるか調べることから始める。FRT77文のオプションVECTOR (V) とNOVECTOR (NV) による結果を比較しベクトル化による計算時間の短縮率を見る。時間はCALL　CLOCK によりDO文の前後で時間差を測定する。

CALL　CLOCK 自身に要する時間はプログラム－１によりＶ、ＮＶいずれの場合も Ｔ２－Ｔ１＝３．８×１０$^{-6}$ sec であった。以下の結果にはこの時間を含んでいる。

プログラム－１

```
      SUBROUTINE TTIME
      IMPLICIT REAL*8(A-H,O-Z)
      CALL CLOCK(T1)
      CALL CLOCK(T2)
      WRITE(6,1000)T1,T2,T2-T1
 1000 FORMAT(1H ,' SUB CLOCK TIME LOSS ',3E15.8)
      RETURN
      END
```

2.1　ベクトル化による効果の演算子、ループ長依存性

倍精度実数、倍精度複素数について配列要素の四則演算、組み込み関数による演算についてベクトル化による効果を調べた。ループ長、演算子依存性をみる。使用したプログラムはプログラム－２である。各演算子について演算時間(T)のＮＶとＶの場合の比Ｒ

Ｒ＝Ｔ（ＮＶ）／Ｔ（Ｖ）

のループ長依存性をFig－１～３に示す。即ちベクトル化により何倍時間が短縮されたかを示す。また、ほぼ漸近値と思われるループ長がNMAX＝9000　の場合のＲおよび単位ループ当りの演算時間をTable －１に示す。

ベクトル化により倍精度実数で約３０倍、倍精度複素数で約２０倍高速化される。また演算子依存性が大きいことが判る。特に割り算が意外に１２倍と効率が悪く、倍精度複素数ＥＸＰは５３倍と効率がいい。

またこのような単純なループでは最大の効率を得るためには10^3～10^4 回の繰り返しが必要である。

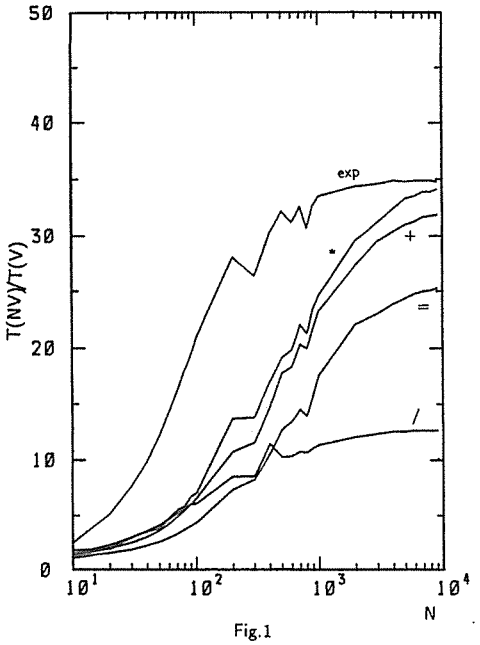

Fig.1

倍精度実数計算におけるベクトル化の効果－1

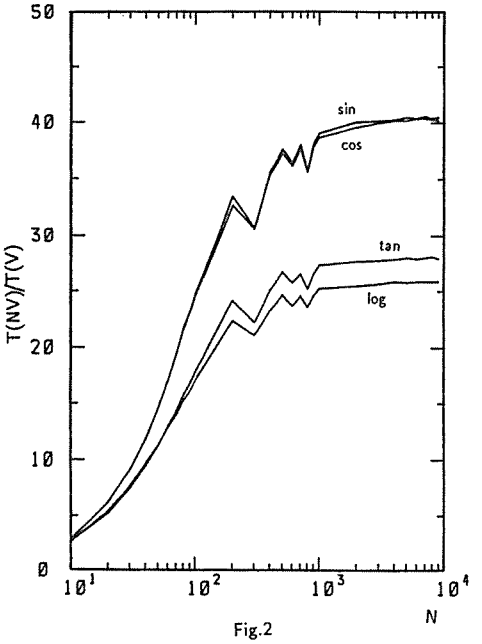

Fig.2

倍精度実数計算におけるベクトル化の効果－2

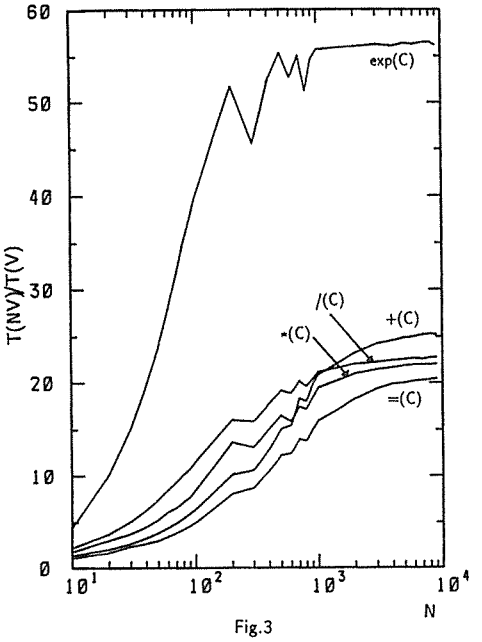

Fig.3

倍精度複素数計算におけるベクトル化の効果

Table 1　ベクトル化による効果と演算時間

	*	/	+	=	exp	sin	tan	cos	log	*(c)	/(c)	+(c)	=(c)	exp(c)
T(NV)/T(V)	34.1	12.7	31.9	25.3	34.8	40.5	27.9	40.2	25.9	22.1	22.8	25.2	20.5	56.2
T(V)/N (nsec)	8.89	39.8	8.83	6.68	98.0	78.0	163.	77.7	123.	31.4	66.9	18.9	14.0	277.

プログラム－2

倍精度実数

```
      SUBROUTINE SUB1(A1,A3,B1,B3,NMAX,TT)
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NDIM=10000)
      DIMENSION A1(NDIM),A3(NDIM),B1(NDIM),B3(NDIM)
      CALL CLOCK(TS)
      DO 410 N=1,NMAX
      [演算子]
  410 CONTINUE
      CALL CLOCK(TE)
      TT = TE-TS
```

```
      RETURN
      END
```

倍精度複素数

```
      SUBROUTINE SUBC1(A1,A3,B1,B3,NMAX,TT)
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NDIM=10000)
      COMPLEX*16 A1(NDIM),A3(NDIM),B1(NDIM),B3(NDIM)
      CALL CLOCK(TS)
      DO 410 N=1,NMAX
      [演算子]
  410 CONTINUE
      CALL CLOCK(TE)
      TT  = TE-TS
      RETURN
      END
```

[演算子] = [＊]

```
      A3(N)   = A3(N) * A1(N)
      B3(N)   = B3(N) * B1(N)
```

[＋]

```
      A3(N)   = A3(N) + A1(N)
      B3(N)   = B3(N) + B1(N)
```

[ＥＸＰ]

```
      A3(N)   = EXP(A1(N))
      B3(N)   = EXP(B1(N))
```

[ＴＡＮ]

```
      A3(N)   = TAN(A1(N))
      B3(N)   = TAN(B1(N))
```

[ＬＯＧ]

[／]

```
      A3(N)   = A3(N) / A1(N)
      B3(N)   = B3(N) / B1(N)
```

[＝]

```
      A3(N)   = A1(N)
      B3(N)   = B1(N)
```

[ＳＩＮ]

```
      A3(N)   = SIN(A1(N))
      B3(N)   = SIN(B1(N))
```

[ＣＯＳ]

```
      A3(N)   = COS(A1(N))
      B3(N)   = COS(B1(N))
```

```
A3(N)    = LOG(A1(N))
B3(N)    = LOG(B1(N))
```

2.2　高速化技法の効果

(i)　“ループ展開（アンローリング）を避けた方がいい”

プログラム－３を用いてループ内の演算をＮ倍にしループの繰り返しを１／ＮとするＮ次のループ展開について、ＮＶおよびＶの場合の演算時間の比Ｒ

$$R = T(NV) / T(V)$$

を調べた。ループ展開の次数Ｎ＝１は素直にＤＯループを回したことに相当する。Fig－４には実線でＲをループ展開の次数Ｎの関数として示した。またＶの場合の演算時間を破線で示してある。このときループの長さは10^4とした。

まず、すでに示唆されているように、ループ展開によってベクトル化の効果は悪くなる。しかし実際の演算時間は（これが重要な因子だが）Ｖの場合、Ｎ＝２、３、５、６で２０％程増加するだけである。

またＮ＝９、１０とループ内の演算回数が多くなると効率が悪くなり演算時間も増大する。

ところでＮ＝４、８のゆらぎは何回プログラムを走らせても正確に生じる。

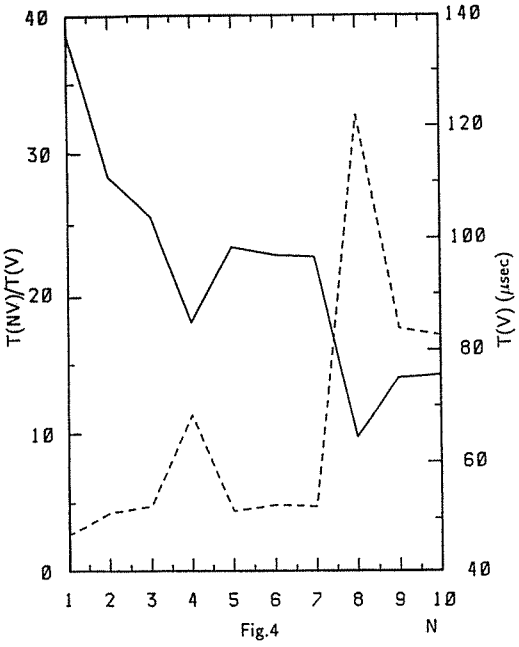

Fig.4
ループ展開におけるベクトル化の効果

プログラム－３

```
SUBROUTINE SUB0(A1,A2,A3,B1,B2,B3)
IMPLICIT REAL*8(A-H,O-Z)
PARAMETER(NDIM=10000)
```

```
      DIMENSION A1(NDIM),A2(NDIM),A3(NDIM)
     &         ,B1(NDIM),B2(NDIM),B3(NDIM)
      CALL CLOCK(T1)
      DO 302 N=1,NDIM                          N = 1
  302 A3(N)   = A1(N)+A2(N)
      CALL CLOCK(T2)
      WRITE(6,1302)T2-T1
 1302 FORMAT(1H ,'LOOP ,1 ',E15.7)
C
      CALL CLOCK(T1)
      DO 301 N=1,NDIM-1,2
      A3(N)   = A1(N) + A2(N)                  N = 2
  301 A3(N+1) = A1(N+1)+A2(N+1)
      CALL CLOCK(T2)
      WRITE(6,1301)T2-T1
 1301 FORMAT(1H ,'LOOP ,2 ',E15.7)
C
   .   .   .   .
     .   .   .
C
      CALL CLOCK(T1)
      DO 310 N=1,NDIM-9,10
      A3(N)   = A1(N) + A2(N)
      A3(N+1) = A1(N+1)+A2(N+1)
      A3(N+2) = A1(N+2)+A2(N+2)
      A3(N+3) = A1(N+3)+A2(N+3)
      A3(N+4) = A1(N+4)+A2(N+4)                N = 1 0
      A3(N+5) = A1(N+5)+A2(N+5)
      A3(N+6) = A1(N+6)+A2(N+6)
      A3(N+7) = A1(N+7)+A2(N+7)
```

```
      A3(N+8) = A1(N+8)+A2(N+8)
  310 A3(N+9) = A1(N+9)+A2(N+9)
      CALL CLOCK(T2)
      WRITE(6,1310)T2-T1
 1310 FORMAT(1H ,'LOOP ,10',E15.7)
C
      RETURN
      END
```

(ii) “指標変数は最初の次元にすべし”

プログラム－４を用いて、ループを配列の“前”の足で回した場合に比べて何倍早くなるか見える。演算時間の比Ｒ

Ｒ＝Ｔ(指標変数が後)／Ｔ(指標変数が前)をＶ、ＮＶの場合それぞれについて調べた。結果はFig－５の線－１で示す。以下(ii)－(v)の結果ＲはFig－５に示し、Ｖ、ＮＶの場合それぞれ実線、破線で示す。ループは正方形でN＝NMAX１＝NMAX２である。Ｒの値は高速化技法の効果によりＲ＞＞１が期待される。

結果ＲはＮ＝ 10～290 にわたってほぼ１である。従って指標変数を最初の次元にしても御利益はない。

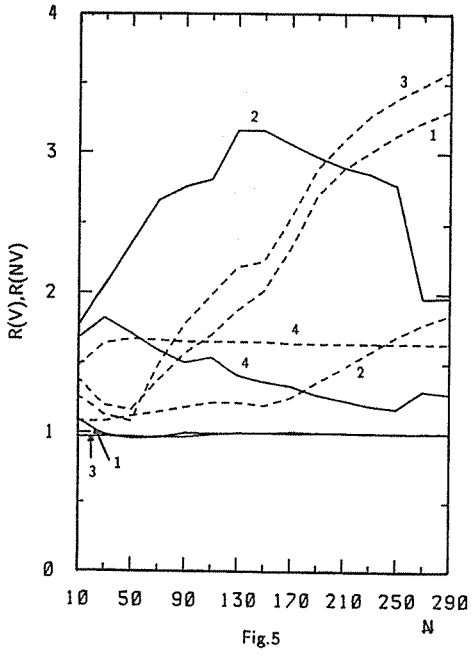

Fig.5
高速化技法の効果

プログラム－４

```
      S3       = SQRT(3.D0)
      CALL CLOCK(T1)
      DO 200 N1=1,NMAX1
      DO 200 N2=1,NMAX2
  200 A2(N1,N2) = A1(N1,N2) + S3*B1(N1,N2)     <－ ”後”で回す
      CALL CLOCK(T2)
```

```
      CALL CLOCK(T1)
      DO 210 N2=1,NMAX2
      DO 210 N1=1,NMAX1
210   A2(N1,N2) = A1(N1,N2) + S3*B1(N1,N2)      <－  "前"で回す
      CALL CLOCK(T2)
```

(iii) "積和型は内積型よりループが短いときには高速である"

プログラム－5を用い(ii)と同様に

R＝T（内積型）／T（積和型）

のV、NVに於ける結果をそれぞれFig－5の線－2で示す。

確かに積和型の方が3倍近く高速になりうる。今の場合ループ長が250を越えると効率が悪くなる。

プログラム－5

```
      CALL CLOCK(T1)
      DO 220 N1=1,NMAX1
      DO 220 N2=1,NMAX2
220   C2(N1)    = C2(N1) + A1(N1,N2)*D1(N2)      <－  内積型
      CALL CLOCK(T2)

      CALL CLOCK(T1)
      DO 230 N2=1,NMAX2
      DO 230 N1=1,NMAX1
230   C2(N1)    = C2(N1) + A1(N1,N2)*D1(N2)      <－  積和型
      CALL CLOCK(T2)                          (ループN1, N2の入れ替え)
```

(iv) "漸化式演算は避けた方がいい"

漸化式の演算を外側、内側ループにいれた場合についてプログラム－6を用いて

R＝T（漸化式が内側ループ）／T（漸化式が外側ループ）

のV、NVに於ける結果をFig－5、線－3で示す。

結果はなぜか漸化式を外側にするとかえって遅くなるぐらいである。

プログラム－6

```
      CALL CLOCK(T1)
      DO 240 N1=1,NMAX1
      DO 240 N2=1,NMAX2
240   A2(N1,N2+1)=A1(N1,N2)+S3*A2(N1,N2)      <－ 漸化式が内側
      CALL CLOCK(T2)

      CALL CLOCK(T1)
      DO 250 N2=1,NMAX2
      DO 250 N1=1,NMAX1
250   A2(N1,N2+1)=A1(N1,N2)+S3*A2(N1,N2)      <－ 漸化式が外側
      CALL CLOCK(T2)                          （Ｎ１、Ｎ２の入れ替え）
```

(V) “外側のループに対するループ展開で、記憶領域の参照回数が減少する場合には大きな効果が得られる場合がある”

プログラム－7を用いて調べる。

Ｒ＝ＴＴ（ループ展開無し）／Ｔ（ループ展開）

のＶ、ＮＶに於ける結果をFig－5、線－4で示す。

確かにループ展開をした方が有利である。ループ長が２００以上では２０％程度の効果である。

プログラム－7

```
      K          = 1
      CALL CLOCK(T1)
      DO 280 N2=1,NMAX2
      DO 280 N1=1,NMAX1
280   C1(N1)     = C1(N1) + A1(N1,K)*B1(K,N2)
      CALL CLOCK(T2)

      K          = 1
      CALL CLOCK(T1)
      DO 290 N2=1,NMAX2,2                    ループ展開
      DO 290 N1=1,NMAX1
290   C1(N1)     = C1(N1) + A1(N1,K)*B1(K,N2) + A1(N1,K)*B1(K,N2+1)
      CALL CLOCK(T2)
```

3　連分数法による散乱問題の解法

Sasakawa らに依って提唱された、連分数法による散乱問題の解法を使ってみる。 定常状態に対するSchroedinger 方程式は一般に線型の微積分方程式で

$$(T+V-E)\,|\Psi\rangle = 0$$

で与えられる。Tは運動エネルギー、Vがポテンシャルエネルギー。$|\Psi\rangle$は適当な境界条件をつけたグリーン関数G_0(＝1／(E－T))に依って

$$|\Psi\rangle = |\phi_0\rangle + G_0 V |\Psi\rangle$$

と書ける。$|\phi_0\rangle$は入射波。 つぎにｔ－行列を連分数の形に次のように定義する。

$$t_i = \langle\phi_{i-1}|V_{i-1}|\phi_i\rangle + \langle\phi_i|V_i|\phi_i\rangle^2 \Big/ \{\langle\phi_i|V_i|\phi_i\rangle - t_{i+1}\}$$

ここで

$$|\phi_{i+1}\rangle = G_0 V_i |\phi_i\rangle$$

$$V_{i+1} = V_i - V_i|\phi_i\rangle\langle\phi_i|V_i / \langle\phi_i|V_i|\phi_i\rangle$$

また　$|\phi_{-1}\rangle = V_{-1} = 0$、$V_0 = V$とする。ｔ－行列は、$t_0$

$$t_0 = \langle\phi_0|V|\Psi\rangle$$

で与えられる。

この方法は非局所的なポテンシャル、チャンネル結合がある場合にも有効であり、詳しくは文献を参照されたい〔1〕。ポテンシャルの行列要素を iterative に積分することにより解がもとまり、ベクトル化しやすい計算方法である。

原子核物理への適用として、Reid による核力のもとで、1チャンネルの散乱問題を解くことを考える。今回の例題は $^1P_1$ の散乱状態であり、プログラム－8がその主要部分である。プログラムはおおまかに3つの部分から構成されている。まず、SUBROUTINE SETPOT によってポテンシャル、グリーン関数および散乱波の初期値が、座標の分割に対応して配列に貯められる。次に、連分数展開において必要とされる積分（INTEGRAL）が実行され、最後に連分数展開によって散乱行列及び波動関数が求められる。プログラム－8から分かるように、主なループはすべてベクトル化され、ベクトル化率は 98.87 %におよんでいる。

ANALYZERのレポートは以下のようになっている。

```
                      "-----------------------------------------------------------------------
                      "      TEST OF MVFG METHOD   STATIONALLY GREEN FUNCTION
                      "-----------------------------------------------------------------------
                      IMPLICIT REAL*8(A-H,O-Z)
                      PARAMETER(NDIM=100,NITE=10,FMN=938.903D0,HBC=0.506769D-2)
                      DIMENSION CWF(0:NITE,NDIM),CGR(NDIM,NDIM),CWB(NDIM),
                     &          CVNL(NDIM,NDIM),CV(0:NITE,2),CKB(0:NITE,2),CVLC(NDIM),
                     &          XR(NDIM),XRRW(NDIM),VLC(NDIM),VNL(NDIM,NDIM),
                     &          CWFI(NDIM),CGRI(NDIM,NDIM),WF(NDIM),A(0:NITE),
                     &          TIME2(0:NITE),TIME3(0:NITE)
                      "
                      CZE      = 0.D0
                      "
                      ELAB = 24.D0
                      Q = SQRT(FMN*ELAB*0.5D0)*HBC
                      "
                      ITMAX = 5
                      NMAX  = 100
                      RMAX  = 20.D0
                      "
                      "  SET POTENTIAL AND GREEN FUNCTION
                      "
                      DMESH = RMAX/DBLE(NMAX)
                      DO 10 K = 1 , NMAX
V-------->            !  XR(K) = DMESH*DBLE(K)
V---------    10      CONTINUE
                      CALL CLOCK(TIME0)
                      CALL SETPOT(Q,XR,NMAX,VLC,VNL,CWFI,CGRI)
                      CALL CLOCK(TIME1)
                      "
                      "  LOOP MESH = NMAX,NMAX/2
                      "
                      DO 1000 JJ = 2 , 1 , -1
2-------->            !  IIMAX = NMAX/JJ
!                     !  DO 100 KI = 1 , IIMAX
! V------>            !  !  KIJ       = KI*JJ
! !                   !  !  CWF(0,KI) = CWFI(KIJ)
! !                   !  !  XRRW(KI)  = XR(KIJ)**2*DMESH*DBLE(JJ)
! !                   !  !  CVLC(KI)  = VLC(KIJ)
! V-------    100     !  CONTINUE
!                     !  DO 110 KI = 1 , IIMAX
! 4------>            !  !  KIJ = KI*JJ
! !                   !  !  DO 120 KF = 1 , IIMAX
! ! V---->            !  !  !  KFJ       = KF*JJ
! ! !                 !  !  !  CVNL(KI,KF)= VNL(KIJ,KFJ)
! ! !                 !  !  !  CGR(KI,KF) = CGRI(KIJ,KFJ)
! ! V-----    120     !  !  CONTINUE
! 4-------    110     !  CONTINUE
!                     !  "
```

```
!                    !  "     LOOP ITERATION
!                    !  "
!                    !  CKB(0,JJ) = 0.D0
!                    !  "
!                    !  DO 200 ITEL=0 , ITMAX+1
!                    !  "
! 6------>           !  !  DO 210 K1 = 1 , IIMAX
! ! 7---->           !  !  !  CSUM2 = CZE
! ! !                !  !  !  DO 220 K2 = 1 , IIMAX
! ! ! V-->           !  !  !  !  CSUM2 = CSUM2 + XRRW(K2)*CVNL(K1,K2)*CWF(ITEL,K2)
! ! ! V---    220    !  !  !  CONTINUE
! ! !                !  !  !  CWB(K1) = CSUM2 + CVLC(K1)*CWF(ITEL,K1)
! ! 7-----    210    !  !  CONTINUE
! ! !                !  !  CSUM1 = CZE
! ! !                !  !  DO 230 K1 = 1 , IIMAX
! ! V---->           !  !  !  CSUM1 = CSUM1 + XRRW(K1)*CWB(K1)*CWF(ITEL,K1)
! ! V-----    230    !  !  CONTINUE
! ! !                !  !  CV(ITEL,JJ) = CSUM1
! ! !                !  !  "
! ! !                !  !  IF(ITEL.EQ.ITMAX+1) GO TO 200
! ! !                !  !  "
! ! !                !  !  DO 300 K1 = 1 , IIMAX
! ! 10--->           !  !  !  CSUM2 = CZE
! ! ! !              !  !  !  DO 310 K2 = 1 , IIMAX
! ! ! ! V-->         !  !  !  !  CSUM2        = CSUM2 + XRRW(K2)*CGR(K1,K2)*CWB(K2)
! ! ! ! !            !  !  !  !  CVNL(K1,K2) = CVNL(K1,K2) - CWB(K1)*CWB(K2)/CV(ITEL,JJ)
! ! ! V---    310    !  !  !  CONTINUE
! ! !                !  !  !  CWF(ITEL+1,K1) = CSUM2
! ! 10----    300    !  !  CONTINUE
! ! !                !  !  CSUM1 = CZE
! ! !                !  !  DO 320 K1 = 1 , IIMAX
! ! V---->           !  !  !  CSUM1 = CSUM1 + XRRW(K1)*CWF(ITEL+1,K1)*CWB(K1)
! ! V-----    320    !  !  CONTINUE
! ! !                !  !  CKB(ITEL+1,JJ) =  CSUM1
! ! !                !  !  CALL CLOCK(TIME2(ITEL))
! 6-------    200    !  CONTINUE
2---------   1000    CONTINUE
                     "
                     "  OUTPUT INTERMEDIATE STEP
                     "
                     "  CAL OF K-MATRIX
                     "
                     DO 1000 ITTMAX = 1 , ITMAX+1
13------->           !  CKK1 = CZE
!                    !  CKK2 = CZE
!                    !  DO 1010 ITEL = ITTMAX , 0 , -1
! V------>           !  !  CV1      = CV(ITEL,1)
! !                  !  !  CV2      = CV(ITEL,2)
! !                  !  !  CV3      = (4.D0*CV1-CV2)/3.D0
! !        S         !  !  CKK3     = (4.D0*CKK1-CKK2)/3.D0
! !                  !  !  A(ITEL) = CV3/(CV3-CKK3)
! !        S         !  !  CKK1     = CKB(ITEL,1) + CV1**2/(CV1-CKK1)
! !        S         !  !  CKK2     = CKB(ITEL,2) + CV2**2/(CV2-CKK2)
! V-------   1010    !  CONTINUE
!                    !  COEF = 1.D0
!                    !  DO 1020 K = 1 , NMAX
! V------>           !  !  WF(K) = 0.D0
! V-------   1020    !  CONTINUE
!                    !  DO 1030 ITEL = 0 , ITTMAX+1
! 16----->           !  !  DO 1040 K = 1 , NMAX
! ! V---->           !  !  !  WF(K) = WF(K) + COEF*CWF(ITEL,K)
! ! V-----   1040    !  !  CONTINUE
! !                  !  !  COEF = COEF * A(ITEL)
! 16------   1030    !  CONTINUE
!                    !  DEL = ATAN(-Q*(4.D0*CKK1-CKK2)/3.D0)
!                    !  CALL CLOCK(TIME3(ITEL))
13--------   1000    CONTINUE
                     "
                     WRITE(*,5000) TIME0,TIME1
                     WRITE(*,5010) (TIME2(ITEL),ITEL=0,ITMAX+1)
                     WRITE(*,5010) (TIME3(ITEL),ITEL=0,ITMAX+1)
                5000 FORMAT(1H ,2D17.6)
                5010 FORMAT(1H ,10D17.6)
                     STOP
                     END
```

プログラム－8

```
SXOS              VER. R1.21  ANALYZER/SX REV. 026

 TOTAL EXECUTION ( CPU ) TIME =  0 :  0' 0" 11  ( 11 MSEC)
 TOTAL EXECUTION FREQUENCY    =           424171
 TOTAL VECTORIZATION RATIO    = 96.41%
```

ATR	PROGRAM	FREQUENCY	EXEC COST(%)	V.RATIO	LOOP	V.LOOP	V.LOOP RATIO
-->	MAIN	1	89.71	98.87	10	10	99.97
SUB	SETPOT	1	6.49	97.47	1	1	100.00
FNC	VPOT	90	0.17	0.00	0	0	0.00
FNC	SBESL	90	3.63	38.51	1	1	40.74

VPOT は SETPOT の中で CALL されるポテンシャルを与える FUNCTION で、SBESL は球ベッセル関数である。Table 2 に SETPOT 及び MAIN 中の積分（INTEGRAL）、連分数（C．F）の計算について1回の iteration あたりに消費される時間（ベクトル・プロセッサーを用いた場合）と NOVECTOR 指定をした場合の比(NV／V)、さらに6回の iteration にかかる時間と NOVECTOR VECTOR の比を与える。MESH は、座標の分割数である。MESH の増加に従って、ベクトル・プロセッサーの効果が顕著になり、500 ポイントの場合には、積分（INTEGRAL）は63倍！？もの利得が認められる。さて、iteration と MESH についての収束性を見るために、散乱の位相差について調べる。Table 3 に、MESH と iteration の回数に対して、得られる散乱の位相差をまとめる。iteration の回数については、3～4回で3桁程度の収束が得られている。MESH については、200ポイント程度で十分あると考えられる。波動関数については、具体的な結果は示さないが、位相差と同様な収束が得られる。今回示した結果は、実験室系のエネルギーで24 MeV（原子核物理においては低エネルギー）についてのもので、収束性はエネルギーの増加に従って悪くなっていくものと考えられる。この問題とあわせて、チャンネル結合がある場合、非局所的ポテンシャル扱い等については、別の機会に議論したい。

Table 2　ベクトル化による効果と演算時間（連分数法）

MESH	SETPOT		INTEGRAL		C.F.		TOTAL	
	T(sec)	NV/V	T(sec)	NV/V	T(sec)	NV/V	T(sec)	NV/V
100	4.60D-3	(5.0)	5.46D-4	(21.3)	1.81D-5	(7.4)	1.00D-02	(11.5)
200	1.01D-2	(8.5)	1.74D-3	(30.1)	1.95D-5	(15.2)	2.62D-02	(19.7)
300	1.70D-2	(10.9)	3.17D-3	(43.9)	2.47D-5	(19.9)	4.60D-02	(28.6)
400	2.55D-2	(12.8)	7.61D-3	(33.5)	2.60D-5	(33.7)	9.51D-02	(26.0)
500	3.46D-2	(14.6)	6.42D-3	(63.0)	2.78D-5	(39.6)	9.46D-02	(41.7)

Table 3　核子-核子散乱 $^1P_1$ チャンネル位相差の収束性

	100	200	300	400	500
1	-0.4272887D-01	-0.4244949D-01	-0.4251735D-01	-0.4253189D-01	-0.4253619D-01
2	-0.3414182D-01	-0.3352741D-01	-0.3351588D-01	-0.3352099D-01	-0.3352325D-01
3	-0.3315012D-01	-0.3314364D-01	-0.3313523D-01	-0.3313455D-01	-0.3313472D-01
4	-0.3312012D-01	-0.3312477D-01	-0.3312485D-01	-0.3312484D-01	-0.3312485D-01
5	-0.3311945D-01	-0.3312512D-01	-0.3312465D-01	-0.3312438D-01	-0.3312434D-01
6	-0.3311936D-01	-0.3312397D-01	-0.3312423D-01	-0.3312427D-01	-0.3312429D-01

4　高エネルギー原子核－原子核散乱の Glauber 理論による解析

我々は、高エネルギー原子核－原子核散乱の例として$\alpha-\alpha$散乱を取りあげる。α粒子は2個の中性子と2個の陽子からなる原子核であるが、4核子系ぐらいであるとGlauber の多重散乱過程を全て取り入れることが可能であると思われる。$\alpha-\alpha$ INCLUSIVE 非弾性散乱断面積の表式を模式的に表すと

$$d\sigma/d\triangle^2 = \Sigma_{ij} C_i C_j A_{ij} EXP(-B_{ij}{}^{*}\triangle^2)$$

とかける。ここで△は散乱角によって決まる運動量移行でC，A，Bは△に依らない定数である。i，jは多重散乱の過程を表す index でそれぞれ3875通りある。係数A，Bは6＊6の行列の逆行列を用いて求まる。

この計算で最も時間のかかる部分は10^7個の6＊6の行列要素を求め、逆行列を計算する部分と、EXPの計算である。そこで、ベクトルプロセッサーをいかすために、6＊6の逆行列の計算を陽に書き下し行列要素の計算と共に1重のDOループの中に入れてしまうことにした。ところで、全てのi，jの場合を配列にしまうのは大変なので、SUB2である程度貯めてから計算することにした。SUB3でA，B，Cの係数を計算しSUB4で断面積の計算をする。SUB3は、1つのDOループの中に150行ほど入っている。

```
SXOS              VER. R1.21  ANALYZER/SX REV. 026


TOTAL EXECUTION ( CPU ) TIME =   0 :  0 ' 29 " 930    (  29930 MSEC)
TOTAL EXECUTION FREQUENCY    =           1299734765
TOTAL VECTORIZATION RATIO    =  96.88%

ATR PROGRAM FREQUENCY EXEC COST(%) V.RATIO LOOP V.LOOP  V.LOOP RATIO
--> MAIN            1         0.01   83.17    3      2         99.80
SUB SUB1            1         0.01   99.96    2      1         99.97
SUB SUB2            1         3.11    0.00    1      0          0.00
SUB SUB3         2504        69.46  100.00    1      1        100.00
SUB SUB4         2504        27.41   99.97    1      1        100.00
```

さて、ANALIZERのレポートを見れば分かるように、96.8％の時間がSUB3（69.5％）、SUB4（27.4％）費やされている。また、おのおののサブルーチンのベクトル化率は、ほぼ100％である。例によって全体の演算時間のNV、Vの比　R=T（NV）／T（V）はほぼ13である。これは一応の成果ではあるが、2章の経験から20倍ぐらいは期待していたので少々期待はずれでもある。SUB3、SUB4各々別々にRを測定してめると、Table 4 のようになる。

Table 4　ベクトル化による効果（α-α散乱）

	full	sub3	sub4
R	13	5	37

これから分かるように、苦労して6＊6の逆行列を書き下し、SUB3をベクトル化率100％近くにしたにも関わらず、5倍の効果しかない。そこでSUB3の中を細かく分けて調べると、次のプログラム－9の関接番地指定の部分がベクトル化によって少しも早くならず（R＜1）かつ時間を消費していた。

プログラム－9

```
A11(IL)  = A110(I1(IL))+A110(I2(IL))
A12(IL)  = A120(I1(IL))+A120(I2(IL))
A13(IL)  = A130(I1(IL))+A130(I2(IL))
・・・
```

現在の演算時間は一応問題のない範囲まで短縮された。更に間接番地指定を避ける工夫をする可能性が残されてはいるが、今後の問題としたい。

5 ま と め

２章で行ったような基本的なアルゴリズムにおける、ベクトル化の効果のデータは、我々のような初心者にとってこれからプログラムを作っていこうとする際に、有用な情報になると思われる。今回の報告はまだまだ不完全で、高速化技法等について系統的に調べでいく必要がある。ここの例題の範囲で少なくとも分かったことは、

ベクトル化による効果は、非常に演算依存性が強い。割り算は特に効率が悪い。

ループ展開はやめた方がいい。特に４、８次は遅い。４次までならしかし演算時間は２０％程度の遅れである。

指標変Ｎは前でも後ろでもかまわない。

積和型は和積型より、最大３倍早くなる。

漸化式演算を気にしなくてもよい。

外側のループ展開……は、９０－１０％の効果がある。

また、実際の応用計算の中で、ベクトルプロセッサーは威力を発揮し１０～４０倍以上の計算時間の短縮を行えた。この因子は非常に大きく、可能な計算の範囲がかなり広がってくる。ところで、ベクトル化率は一応の目安にしかならない。４章の例題ではベクトル化率１００％でも、間接番地指定が高速化に対する一つの阻害要因であった。この点については改良する必要がある。

最後に有益な議論をしてくださった大坪助教授に感謝する。

参 考 文 献

〔1〕 J. Horacek and T. Sasakawa, Phys. Rev. C32 (1985), 70; Phys. Rev. A28 (1983), 2151.

〔2〕 W. CZYZ and L. Maximon, Ann. Phys. 52 (1969), 59.
H. Yoshiki, to be published.

〔3〕 後藤、大中　大阪大学大型計算機センターニュース Vol. 16 No. 1 1986, 37.