技術研究報告（東京大学地震研究所）No. 1，65-71 頁，1996 年.
Technical Research Report (Earthquake Research Institute, University of Tokyo), No. 1, p. 65-71, 1996.

並列計算機 CS 6400 における数値計算ライブラリの現状と問題点

桧山澄子*・長谷川秀彦**

The Present State and the Problems of the Numerical Libraries for CS 6400 Parallel Computer System

Sumiko HIYAMA* and Hidehiko HASEGAWA**

Abstract

We have analyzed the parallel performance of some numerical libraries for the Cray CS 6400 parallel computer system of the computing center of the Earthquake Research Institute, Univ. of Tokyo. We used both the matrix multiplications and the solution of the linear system of equations for comparing the performance of LibSci, IMSL 90 Library, and IMSL 77 Library. The size of matrices was tested from 32 to 1024. As a result, it became clear that LibSci had good performance for this system, but IMSL 90 Library was not parallelized. We had modest parallel performance from IMSL 77 Library with the Cray CF 90 compiler.

Key words : parallel computer system, software library, parallel performance.

はじめに

1995 年 3 月地震研究所では，メモリ共有型の並列計算機 Cray CS 6400 を導入した．このマシンは，従来研究者にサービスしていた科学計算用汎用機に代わるものとして導入した．32 台あるプロセッサには，SUN ワークステーションと同じ Super SPARC+が使われている．OS は SUN Solaris 2.3 であるため，すでにワークステーションに慣れ親しんでいるユーザにとっては，プログラムの移植性，互換性については問題はなかった．しかし，一歩進めて並列化によって高性能を得るという事に関しては，新しい経験であった．

CS 6400 では，並列化はプロセスをサブ・プロセス（スレッド）化して複数の CPU に同時に実行処理させることで並列化される．従って，うまくスレッド化して並列処理ができれば高速処理が期待できるわけであるが，これはそうたやすいことではないため，各ユーザには以下のような並列化を薦めている．

1. 並列化を意識して書かれたライブラリを使う．
2. コンパイラー・オプションを指定することで，自動的に並列化する．（低い並列化，コンパイラーによりこのオプションがあるとは限らない．）
3. 何らかのディレクティブを使って，自動的にまたは明示的に並列化を図る．（経験と知識が必要．簡単でない．）

1. のライブラリとしては，マルチスレッド化を図ったといわれている Cray 社の科学ライブラリ LibSci と，Fortran 90 の構造を活かして並列化向きにプログラムされたという IMSL 社 Fortran 90 MP ライブラリがある．以下それらを **LibSci**，**IMSL 90** と略記する．そのほかに，従来の IMSL 社の Math/Library を，Cray 社の Fortran 90 でコンパイルし直したもの（これは **IMSL 77** と略記する）が入っている．

特に以下の機能のプログラムはどのライブラリにも入っている．

1) 行列・ベクトルの基本操作
2) 連立方程式，固有値・固有ベクトルなど線形代数に関するもの
3) フーリエ変換（FFT）
4) 乱数

1996 年 5 月 18 日受付，1996 年 9 月 17 日受理.
* ~~東京大学地震研究所~~情報センター，（東京大学地震研究所）.
** 図書館情報大学.
* Earthquake Information Center, (Earthquake Research Institute, University of Tokyo).
** University of Library and Information Science

しかも数値解析的には確立したアルゴリズムが使われているので，精度や安定性についてはそう違いはない．違いはこのような並列環境用にチューニングされているか，つまり並列コンピュータ向きのコードになっているかどうかである．しかもそのコードが我々のシステム環境で，どれだけ最適になっているか否かである．

そこで，我々はこれらのライブラリのパフォーマンスはどうか，どのような特長があるのかを調べることにした．

並列計算の環境

1. ハードウェア

当センターのシステム構成は，並列計算機Cray CS 6400を中核に構成されている．このマシンはプロセッサ32台，メモリ8GBである．各プロセッサにはSuper Sparc+が使われている．OSはSUN Solalis 2.3である．このCS 6400の性能と特長を表1に示す．

2. CS 6400の運用形態

CS 6400は全国からネットワークを通して利用できるようになっており，32台のプロセッサの中，4台をTSS処理用に，残り28台をバッチジョブ処理用に割り当ててある．このバッチジョブはNetwork Queuing System (NQS) で動き，並列化はこのバッチ処理で行う．この種のバッチジョブは，CPU—Time打ち切り（5 days, 20 days, 100 days）によって3つのジョブクラスがある．同時にrunできるのは7ユーザまで，同一クラスには，同一ユーザは1ジョブしか入れないようにしてある．メモリ制限はない．並列化する場合のCPU台数は各ユーザがシェル・スクリプトで，

setenv PARALLEL 台数，または

setenv NCPUS 台数で指定する．

ここで台数を8以下にするように薦めている．

3. ソフトウェア

(1) コンパイラ

コンパイラは，

1) Cray Research Fortran 1.0（ISO，ANSI Fortran 90に準拠）
2) SPARC Fortran 3.0（ISO，ANSI Fortran 77に準拠）
3) Apogee C 3.0（K & R ANSI標準）
4) SPARC C 3.0
5) GNU C

の5種類が導入されており，自動並列化の機能があるのは，前の3つだけである．3）が最も高速のコードを出し，次に1）Cray Fortran（**CF 90**と略記する），2）SPARC Fortran（**f 77**と略記する）の順であるといわれている．従ってより高度の並列化を期待するFortranユーザは，F 77で書いてあるプログラムであってもCF 90でコンパイルした方が，より優れたコードを期待できるはずである．

このCF 90の場合に，並列化する方法には以下の3通りがある．

① プログラムのコンパイル時にコンパイラ・オプションに，-O3を指定する．これは自動並列化と呼ばれているもので，最も低い並列化ができる．具体的には一番外側のDOループだけを並列化するなどが行われる．

② 自動並列化ディレクテブ！MIC$を使う．具体的にはFortranソース・プログラムの中に！MIC$ PARALLELと！MIC$ PARALLELENDの対のステートメントを入れて，その間の部分を並列化する．この他にも，対で囲んでDOループやブロック内を並列化するディレクティブが他にも4つある．

表 1. Cray 6400の性能と特長

項目	性能・特長
並列処理方式	メモリ共有型．複数のプロセスやスレッドを同時に複数のプロセッサで実行させ，特別なハード機構は使用していない．
Mflops値，MIPS値	35.5Mflops/プロセッサ，167.4MIPS/プロセッサ
SPECrate_int92 SPECrate_fp92	54186/32台(1台時の26.5倍) 72177/32 台(1台時の28.3倍)
キャッシュ・サイズ	各CPUごとに2MB(外部),36KB(オンチップ)
メモリへのパス	55MHzのXDBusで接続され，パケット交換方式でデータを転送する．XDBusのスループットは1.76GB/sec
Super Scalarの特長	1クロックで複数命令を同時に実行する方式．各クロックサイクルで3命令を実行し，他にはない命令の組み合わせが可能．

③　ディレクティブ！**DIR**＄をFortranソース・プログラムの中に書いて，並列化したい箇所や内容を明示的に指定する．

たとえ前述のようなディレクティブなど並列化のための道具がそろっていても，一般的に，並列化により性能を向上させるにはかなりの知識と経験が要求される．そこで通常のユーザには，簡便な方法でなおかつ実際的な以下のことを薦めている．

● 前もって**並列化を意識して書かれている**ライブラリを使うこと．

● 自動並列化を使う．つまり，前述の①のコンパイラ・オプション-O3を使って，Fortranソースプログラムをコンパイルする．そして実行のとき，setenv NCPUS 1を行い，次にsetenv NCPUS 2でCPUの数を2倍にしてみる．

もし，並列化の効果があれば，さらにNCPUS 3や4にしてみる．

さて並列化されたライブラリにはどのようなものがあるのかを，以下に説明する．

(2)　ライブラリ

以下3つの科学計算ライブラリがロードモジュールの形で入っている．

1)　IMSL 90

● 並列計算機用にFortran 90で書かれている．

● 行列計算においては，BLAS（Basic Linear Algebra Subprograms）を使わない．代わりにFortran 90のgeneric interfaceや配列dataを使う．これは，並列計算機に対してハイパフォーマンスを得るため，プログラムし易く，メンテナンスしやすいとされている．

● 開発の際，並列計算向きに以下の点をキーポイントにコーデングされている．

common文は使わない，stop文も使わない，エラーメッセージはoptionにする，再起呼び出しの際整合性をもたせるため，固有変数の値はサブルーチンの呼び出し側のプログラムにかならずセーブしておくようにする．

● 1995年3月にリリースされたもので，線形計算，乱数，フーリエ変換，数値積分，常・偏微分方程式，非線形方程式などのプログラムがある．

その他，IMSL 90には演算子や関数を定義して，更に簡易化・簡略化したOperation and Function Modules in Fortran 90というものもある．以下これを**IMSL 90（OF）**と略記する．これはFortran 90のプログラム内であたかもFortranのステートメントのように書くことができる．たとえば連立一次方程式，

$Ax=b$

を解きその解をcに入れるときのプログラムIMSL 90とIMSL 90（OF）では以下のようになる．（プログラム）

```
use lin_sol_gen_int
    use  rand_gen_int
    integer,parameter :: n=32
    real A(n,n),b(n),c(n),y(n**2)

      call rand_gen(y)
      A=rshape(y,(/n,n/))
      call rand_gen(y)
      b=reshape(y,n)
! Computethesolution mtrix of Ax=b
      call  lin_sol_gen(A,b,c)
   end
```

IMSL 90をcallするプログラム

```
USE defined_operations
    integer :: n=32

    real A(n,n),b(n),c(n),y(n**2)

      A=rand(A);b=rand(b)
  ! Computethesolution mtrix of Ax=b
       c=A.ix.b
      end
```

IMSL 90(OF)を使ったプログラム

表 2.　ライブラリ一覧

ライブラリ	開発提供元	略記
Fortran 90 MP Library 1.0	Visual Numerics	IMSL90
Cray Soft LibSci 1.0	Cray Research	LibSci
MATH/STAT Library 3.0	Visual Numerics	IMSL 77

2) LibSci

● 並列計算向きにマルチスレッド化されている.

● BLAS (レベル 1, 2, 3), LAPACK, FFT と信号処理, 一般 MATRIX, 乱数のプログラムがある.

3) IMSL 77

● 従来 F 77 用でサポートされていた IMSL ライブラリを, CF 90 を使ってコンパイルし直してある.

● 並列計算を意識してない.

● 行列演算では BLAS をベースにしている.

● 線形計算, 補間・近似, 微積分, 偏・常微分方程式, Fourier 変換, 特殊関数など約 450 種のプログラムがある.

評価方法とその結果

まず以下の 3 つの場合にどのような性能が得られるかを調べる.

Case 1. 2 つの行列の積 AB を計算する.

Case 2. 連立一次方程式 $Ax=b$ を解く.

1. Case 1.

乱数を発生させて $n\times n$ の正方行列を作り, 各種ライブラリプログラムを利用して積を求める. 元数は n=32 から 1024 まで 2 倍ごとに変化させ, CPU 台数を 1, 2, 3, 6 台で並列化させる. 計算はすべて, 倍精度で行う. コンパイラ・オプションは−O3 を使う.

ライブラリの呼び出しの直前と直後で時刻 $T1$, $T2$ を計測する. 性能は, μsec である.

$$Mflops=op/(T2-T1)*.001$$

でもとめた. op は総浮動小数点演算回数で, 行列積では積演算が n^3, 和演算が $(n-1)n^2$ なので,

合計はおよそ

$$op=2n^3$$

と見積もれる.

各ライブラリプログラムを表 3 にしめす.

2 節で述べたように IMSL 90 では行列の基本演算のプログラムはなく, すべて Fortran 90 が持っている array operation か Generic function で行うことになっている. ここでは参考のため, CF 90 の総称関数 matmul を使う.

以下図 1-図 4 に測定結果を示す.

これから次のようなことが分かる.

● **.x.** の性能は貧弱である.

● matmul と .x. は元数が大きくなると性能が落ちる.

このことは大きな問題で, 本来より高速を要求されるところで性能が出ていない. .x. の場合 n=128 のとき, 約 8.9 Mflops が n=1024 では 1.4 Mflops に落ちる. matmul の場合は n = 128 のとき 54.4 Mflops が n = 1024 では 7.0 Mflops に急減している.

表 3. 行列の積を求めるライブラリプログラム一覧

ライブラリ名	プログラム名	備考
1. IMSL90		ライブラリとしては無い.
2. IMSL90 (F0)	. x.	演算子. x. を使う.
3. IMSL77	DMRRRR	Fortran77 用を Cray Fortran 90 で recompile したもの
4. LIbSci	DGEMM	BLAS レベル 3 のもの

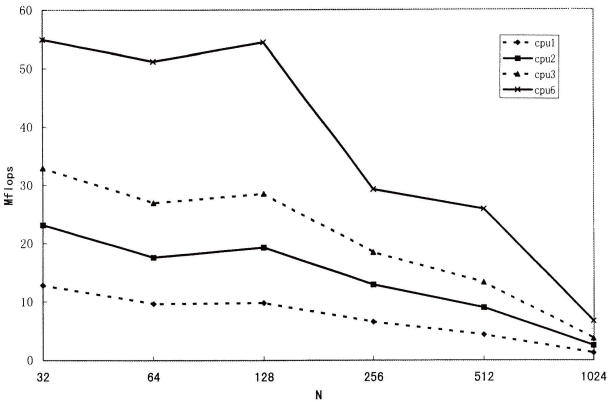

図 1. CF 90 matmul の場合

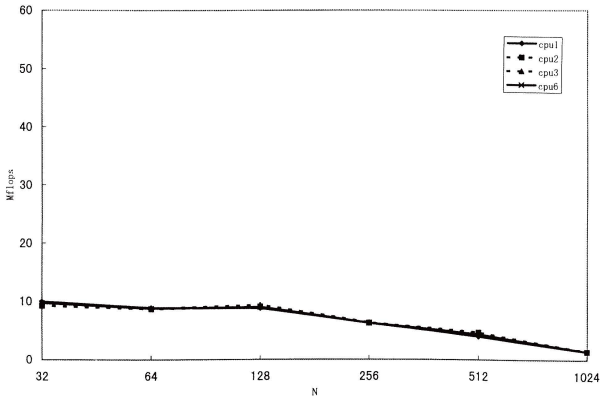

図 2. IMSL 90 (OF) .x. の場合

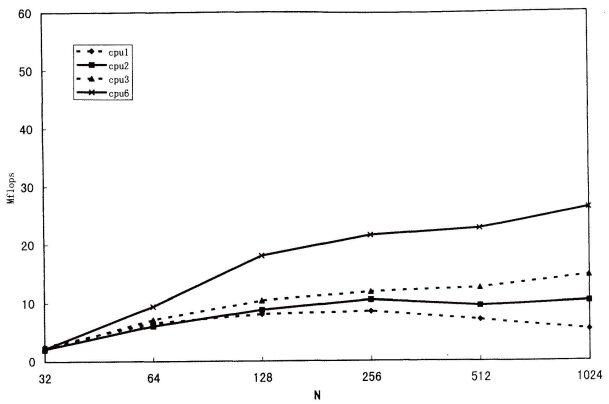

図 3.　IMSL 77 DMRRR の場合

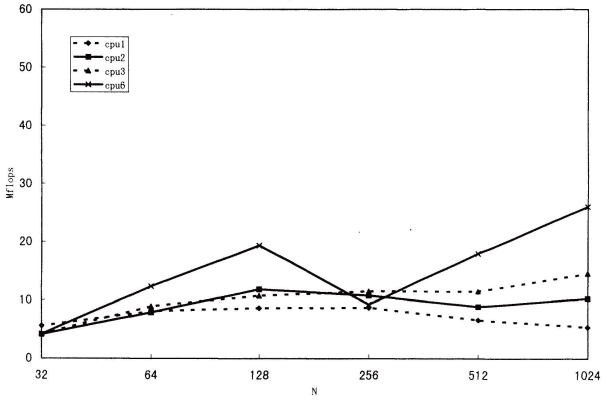

図 4.　LibSci DGEMM の場合

表 4.　転置行列と転置しない行列の積の比較

n=1024	CPU=1	CPU=6
matmul(A,B)	1.4Mflops	7.0Mflops
matmul(transepose(A),B)	8.4Mflops	48Mflops

表 5.　連立一次方程式を求めるプログラム一覧

ライブラリ	プログラム名	備考
IMSL90	lin_sol_gen	
IMSL90(OF)	.ix.	演算子
IMSL77	DLSORG	
LibSci	DGETRF ,DGETRS	DGETRF で，三角行列にし DGETRS で解く

● .x. は並列化の効果がまったくない．

● DMRRRR と DGEMM は IMSL 77 と共に並列向きになっている．

単一 CPU のときキャッシュがオンチップが 36 KB，外部 2 MB なので n＝128 で性能が落ちるのは，妥当である．CPU をふやすと，各 CPU のキャッシュでデータを分割して持つので，性能は落ちにくくなる．

以上の他に

1)　matmul の計測プログラムで，C＝matmul（A，B）と書いて，その前後で時間を計測するだけで，C はその後で参照しないプログラムを書いた．すると，n を増加させても経過時間はほとんど 0 になってしまった．このことより，

● CF 90 では，後で計算結果を参照していないと，コードが生成されない．オプティマイズ・オプションを－O1 に変えても結果は同じである．このことを知らないとひどい目にあうこともある．

2)　Fortran 90 では 2 次元配列は行方向に割り振られる．転置行列の積つまり $C=A^tB$ を行うと，matmul のときでも格段に性能がよくなる（表 4）．

● このことより，一般の行列積ライブラリにはブロック化などの高速化が不可欠である．

2.　Case 2

連立一次方程式 $Ax=b$ で，左辺 A と右辺ベクトル b に乱数を入れ，Case 1 同様に，n＝32，64，1024 の各場合を CPU＝1，2，3，6 で並列化したときのパフォーマンスを調べた．いずれの場合も A をピボット選択付き LU 分解（行列の上三角，下三角分解）をし，その後，前進・後退代入法で解いている．この場合の浮動小数演算は LU 分解におよそ $2n^3/3$，前進消去と後退代入の各々におよそ n^2 ずつである．なお，LU 分解に前進・後退代入を合わせた時間を，$op=2n^3/3$ と概算して Mflops を出している．

この種の連立一次方程式のプログラムは表 5 に示す．

以下に測定結果を示す（図 5-図 8）．

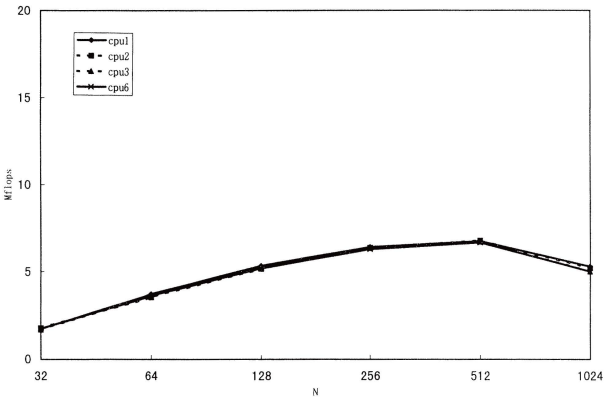

図 5. IMSL 90 lin_sol_gen の場合

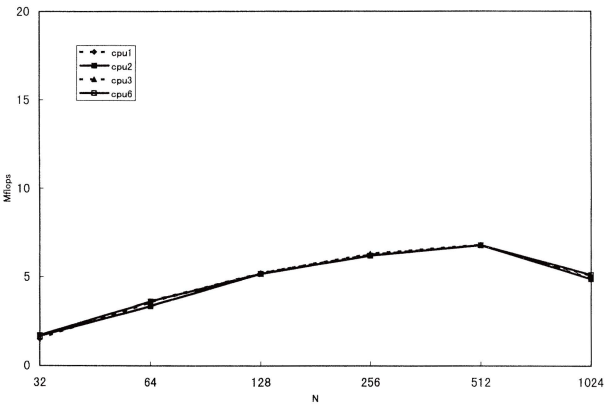

図 6. IMSL 90（OF）の .ix. の場合

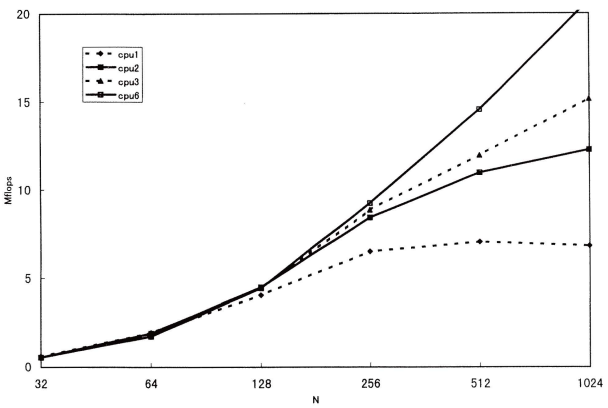

図 7. IMSL 77 DLSORG の場合

以上の結果から以下のことがいえる.

● lin_sol_gen, .ix. の性能が貧弱である. また 2 つは同一コードをだしていると思われる.

● lin_sol_gen, .ix. は全く並列化しない.

● lin_sol_gen, .ix. は $n=460$ からパフォーマンスが落ちる, キャッシュのあふれを起こし易い.

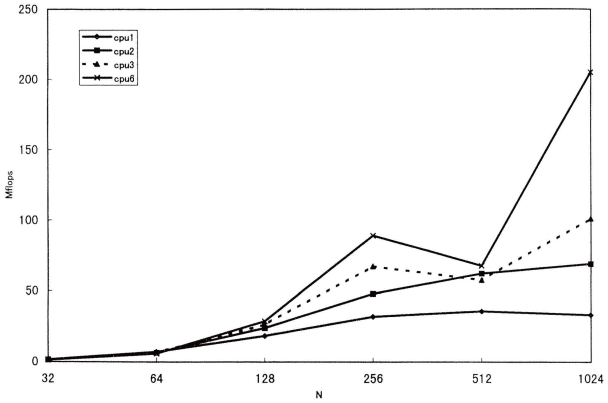

図 8. LibSci DGETRF, DGETRS の場合

● DLSORG と {DGETRF, DGETRS} はともに並列化の効果がある.

● 特に {DGETRF, DGETRS} は 186.8 Mflops（CPU＝6, n＝1024), 33.2 Mflops（CPU＝1, n＝1024）と lin_sol_gen の 37 倍, 6.5 倍, DLSORG の 9 倍, 5 倍であり並列化性能, 実効性能ともすぐれている.

これは LAPACK のルーチンであり, キャッシュアーキテクチャのコンピュータ用に, ブロック化されていることが良い結果を出しているようだ.

ま と め

Case 1 : 行列積, Case 2 : 連立一次方程式の解法に用いられる市販のライブラリを利用して並列化を行った結果から次のことがわかった.

● IMSL 90（OF）は利用方法は簡単でよいが, まだ未熟で性能面, 並列化という点で大きな問題がある.

● 並列化はキャシュにとっても良い効果をもたらすが（メモリ参照が分散し, 局所化される), IMSL 90, IMSL 90（OF）はその点でも問題がある.

● Case 2 : 連立一次方程式の DGETRF, DGETRS のようにブロック化は, 並列化計算機には非常に良い結果をもたらす.

並列化に対する考え方は大きくいって,

(a) ユーザが, ブロック化アルゴリズムなどを使ってプログラムを書く.

(b) コンパイラ, あるいはディレクティブに頑張ってもらう.

(c) 高性能ライブラリを活用する.

に分けられるだろう. 本実験の結果からよくチューニングされたライブラリ, たとえば LibSci の DGETRF, DGETRS なら満足できる結果が得られたが, IMSL 90 の場合のように旧来のものより明らかに性能が落ちているも

のもある．この事は，IMSL 90 に付いている，Benchmarking or Timing Program（15 種類の IMSL 90 と IMSL 77 ライブラリの性能評価を行う事が出来る）からも明らかである．せっかくこうした優れたベンチマーク機能があるのだから，ソフトウェア・ベンダーとしてはこれらを活かし，さらに性能アップに努めて欲しい．

その他，実験を通して感じた点を以下に述べる．

① IMSL 90 のネーミングは目的が分かって良い．また，ルートのライブラリに optional Argument を持たせることで，実際にはそれに関連した階層構造の幾つかのライブラリを呼び出すことと同じことになる．これは，ライブラリの名前が少なくて済みユーザには便利である．

例えば，連立一次方程式 $Ax=b$ を解く際に，計算途中の LU 分解した結果や，ピポット選択に使った P_i が欲しい場合がある．そのようなとき通常は，それぞれ目的のプログラム lin_sol_gen_save_LU や lin_sol_gen_no_pivoting を呼ぶことになる．それを IMSKL 90 では，lin_sol_gen（A，b，x，2）や lin_sol_gen（A，b，x，5）とするだけで済ませる事が出来る．

② IMSL 90（OF）はパフォーマンスを抜きにすれば，簡便であり，ユーザインターフェスとしては良く考えてある．

③ IMSL 90 はソース・プログラムの中で using プログラム名を指定しているにもかかわらず，link のときに，-p オプションで別途モジュール名を指定しないといけない．それを止めると，インタフェースモジュール（43 本）総てを link することになり a. out が巨大になる． この辺も改良して欲しいところである．

謝　辞：本稿をまとめるに当たって，日本ビジュアルニューメリック社　技術サポートグループ飯島昭氏には大変お世話になった．お礼申し上げる．

文　　献

Hanson, R.J., 1992, *A design of High-Performance Fortran 90 Libraries*, IMSL, Inc., IMSL Technical Report Series, No. 9201.

Hanson, R.J., 1993, *Operator and Function Modules with Fortran 90*, Visual Numerics, Inc., VNI Technical Report Series, No. 9305.

Metcalf, M. and Reid J., 1994, *Fortran 90 Explained*, Oxford Science Publications. pp. 306.