HITACHI SR8000 用 BLAS，LAPACK，ScaLAPACK ライブラリーのご紹介

（株）日立製作所

BLAS（Basic Linear Algebra Subprograms）、LAPACK（Linear Algebra PACKage）、ScaLAPACK（Scalable LAPACK）は、netlib（http://www.netlib.org/）で一般公開されている、行列計算ライブラリーです。

今回弊社では、各ルーチンのインターフェース（ルーチン名、引数、機能）は変えずに HITACHI SR8000 向きにチューニングした、BLAS、LAPACK、ScaLAPACK の各ライブラリーをリリースしました。ここでは本ライブラリーのサブルーチンの概要、種類、使用方法および注意事項などについてご紹介いたします。

1. ライブラリーの概要

SR8000 向きにチューニングした BLAS、LAPACK、ScaLAPACK の各ライブラリーは、利用するアドレッシングモード（64 ビット版／32 ビット版）と、ノード内の各プロセッサーを並列に利用するかどうか（要素並列版／スカラー版）によってそれぞれ合計 4 種類のバージョンがあります。要素並列版ライブラリーはノード内の複数プロセッサーを用いて要素並列実行し、スカラー版ライブラリーはノード内の一台のプロセッサーを用いて実行するものです。性能的には要素並列版の使用をお薦めしますが、スカラー版は上位のプログラムがノード内で並列化されている場合などに利用できるように用意しています。なお、これらの各バージョンには、ライブラリーとしての機能的な差違はありません。ScaLAPACK は、通信ライブラリーMPI（Message Passing Interface）を用いて、ノードをまたいで並列実行します。

1.1 BLAS ライブラリーの概要

BLAS ライブラリーには、行列計算で現れるほとんどの基本演算がサブルーチンとして用意されており、演算の対象が一重ループ、二重ループあるいは三重ループかによって、それぞれ BLAS1、BLAS2、BLAS3 に分類されています。BLAS1 については、一重ループの演算であり改良の余地がほとんどなく、普通に書いた一重ループのプログラムと性能的にあまり変わりませんが、BLAS2、BLAS3 のルーチンについては二重または三重ループのプログラムをそのまま書くより、性能向上が望めます。

BLAS ライブラリーの各ルーチンの名前は、取り扱う変数配列の種類（実数／複素数、単精度／倍精度）により、以下のように最初の文字が決められています。

S　REAL（単精度実数）
D　DOUBLE PRECISION（倍精度実数）
C　COMPLEX（単精度複素数）
Z　DOUBLE COMPLEX（倍精度複素数）

SR8000 版でも単精度と倍精度の両方をサポートしています。特に倍精度ルーチンに対して、

SR8000 用にチューニングしてありますので、倍精度ルーチンの使用をお薦めします。

BLAS ライブラリーの主なルーチンの機能概要を表 1.1.1～表 1.1.3 に示します。表において、「型」はルーチン名の最初の文字（__の個所）を置き換えることのできる文字を表わしています。各ルーチンは主にサブルーチン形式ですが、関数形式のものは、先頭に FUNCTION をつけて表わしています。詳しい説明は、表の後に記載します。

なお、BLAS の機能仕様の詳細につきましては、以下を参照して下さい。

http://www.netlib.org/blas/

表 1.1.1　BLAS1 の主なルーチンの機能概要

ルーチン名と引数	演算	型(_)
_ROTG(A,B,C,S)	GIVENS 面回転マトリックスの計算	S,D
_ROTMG(D1,D2,A,B,PARAM)	修正 GIVENS 面回転マトリックスの計算	S,D
_ROT(N,X,INCX,Y,INCY,C,S)	GIVENS 面回転の適用	S,D,CS,ZD
_ROTM(N,X,INCX,Y,INCY,PARAM)	修正 GIVENS 面回転の適用	S,D
_SWAP(N,X,INCX,Y,INCY)	$\mathbf{x} \Leftrightarrow \mathbf{y}$	S,D,C,Z
_SCAL(N,ALPHA,X,INCX)	$\mathbf{x} \leftarrow \alpha\mathbf{x}$	S,D,C,Z,CS,ZD
_COPY(N,X,INCX,Y,INCY)	$\mathbf{y} \leftarrow \mathbf{x}$	S,D,C,Z
_AXPY(N,ALPHA,X,INCX,Y,INCY)	$\mathbf{y} \leftarrow \alpha\mathbf{x} + \mathbf{y}$	S,D,C,Z
FUNCTION _DOT(N,X,INCX,Y,INCY)	$\_dot \leftarrow \mathbf{x}^T\mathbf{y}$	S,D
FUNCTION _DOTU(N,X,INCX,Y,INCY)	$\_dotu \leftarrow \mathbf{x}^T\mathbf{y}$	C,Z
FUNCTION _DOTC(N,X,INCX,Y,INCY)	$\_dotc \leftarrow \mathbf{x}^H\mathbf{y}$	C,Z
FUNCTION _NRM2(N,X,INCX)	$\_nrm2 \leftarrow \|\|\mathbf{x}\|\|_2$	S,D,SC,DZ
FUNCTION _ASUM(N,X,INCX)	$\_asum \leftarrow \|\|re(\mathbf{x})\|\|_1 + \|\|im(\mathbf{x})\|\|_1$	S,D,SC,DZ
FUNCTION I_AMAX(N,X,INCX)	最大絶対値をもつ要素のインデックス	S,D,C,Z

表 1.1.2　BLAS2 の主なルーチンの機能概要

ルーチン名と引数	演算	行列の型	型(_)
_GEMV(TRANS,M,N,ALPHA,A,LDA,X,INCX, BETA,Y, INCY)	$\mathbf{y} \leftarrow \alpha A\mathbf{x} + \beta\mathbf{y}$ $\mathbf{y} \leftarrow \alpha A^T\mathbf{x} + \beta\mathbf{y}$ $\mathbf{y} \leftarrow \alpha A^H\mathbf{x} + \beta\mathbf{y}$	密	S,D, C,Z
_GBMV(TRANS,M,N,KL,KU, ALPHA,A,LDA,X,INCX,BETA,Y,INCY)	$\mathbf{y} \leftarrow \alpha A\mathbf{x} + \beta\mathbf{y}$ $\mathbf{y} \leftarrow \alpha A^T\mathbf{x} + \beta\mathbf{y}$ $\mathbf{y} \leftarrow \alpha A^H\mathbf{x} + \beta\mathbf{y}$	帯	S,D, C,Z
_HEMV(UPLO,N,ALPHA,A,LDA,X,INCX, BETA,Y,INCY)	$\mathbf{y} \leftarrow \alpha A\mathbf{x} + \beta\mathbf{y}$	ｴﾙﾐｰﾄ	C,Z
_HBMV(UPLO,N,K,ALPHA,A,LDA,X,INCX, BETA,Y,INCY)	$\mathbf{y} \leftarrow \alpha A\mathbf{x} + \beta\mathbf{y}$	ｴﾙﾐｰﾄ帯	C,Z
_HPMV(UPLO,N,ALPHA,AP,X,INCX,BETA,Y,INCY)	$\mathbf{y} \leftarrow \alpha A\mathbf{x} + \beta\mathbf{y}$	圧縮型ｴﾙﾐｰﾄ	C,Z
_SYMV(UPLO,N,ALPHA,A,LDA,X,INCX, BETA,Y,INCY)	$\mathbf{y} \leftarrow \alpha A\mathbf{x} + \beta\mathbf{y}$	対称	S,D

ルーチン名と引数	演算	行列の型	型(_)
_SBMV(UPLO,N,K,ALPHA,A,LDA,X,INCX, BETA,Y,INCY)	$\mathbf{y} \leftarrow \alpha A\mathbf{x} + \beta\mathbf{y}$	対称帯	S,D
_SPMV(UPLO,N,ALPHA,AP,X,INCX,BETA,Y,INCY)	$\mathbf{y} \leftarrow \alpha A\mathbf{x} + \beta\mathbf{y}$	圧縮型対称	S,D
_TRMV(UPLO,TRANS,DIAG,N,A,LDA,X,INCX)	$\mathbf{x} \leftarrow A\mathbf{x}, \mathbf{x} \leftarrow A^T\mathbf{x}, \mathbf{x} \leftarrow A^H\mathbf{x}$	三角	S,D,C,Z
_TBMV(UPLO,TRANS,DIAG,N,K,A,LDA,X,INCX)	$\mathbf{x} \leftarrow A\mathbf{x}, \mathbf{x} \leftarrow A^T\mathbf{x}, \mathbf{x} \leftarrow A^H\mathbf{x}$	三角帯	S,D,C,Z
_TPMV(UPLO,TRANS,DIAG,N,AP,X,INCX)	$\mathbf{x} \leftarrow A\mathbf{x}, \mathbf{x} \leftarrow A^T\mathbf{x}, \mathbf{x} \leftarrow A^H\mathbf{x}$	圧縮型三角	S,D,C,Z
_TRSV(UPLO,TRANS,DIAG,N,A,LDA,X,INCX)	$\mathbf{x} \leftarrow A^{-1}\mathbf{x}, \mathbf{x} \leftarrow A^{-T}\mathbf{x}, \mathbf{x} \leftarrow A^{-H}\mathbf{x}$	三角	S,D,C,Z
_TBSV(UPLO,TRANS,DIAG,N,K,A,LDA,X,INCX)	$\mathbf{x} \leftarrow A^{-1}\mathbf{x}, \mathbf{x} \leftarrow A^{-T}\mathbf{x}, \mathbf{x} \leftarrow A^{-H}\mathbf{x}$	三角帯	S,D,C,Z
_TPSV(UPLO,TRANS,DIAG,N,AP,X,INCX)	$\mathbf{x} \leftarrow A^{-1}\mathbf{x}, \mathbf{x} \leftarrow A^{-T}\mathbf{x}, \mathbf{x} \leftarrow A^{-H}\mathbf{x}$	圧縮型三角	S,D,C,Z
_GER(M,N,ALPHA,X,INCX,Y,INCY,A,LDA)	$A \leftarrow \alpha\mathbf{x}\mathbf{y}^T + A$	密	S,D
_GERU(M,N,ALPHA,X,INCX,Y,INCY,A,LDA)	$A \leftarrow \alpha\mathbf{x}\mathbf{y}^T + A$	密	C,Z
_GERC(M,N,ALPHA,X,INCX,Y,INCY,A,LDA)	$A \leftarrow \alpha\mathbf{x}\mathbf{y}^H + A$	密	C,Z
_HER(UPLO,N,ALPHA,X,INCX,A,LDA)	$A \leftarrow \alpha\mathbf{x}\mathbf{x}^H + A$	ｴﾙﾐｰﾄ	C,Z
_HPR(UPLO,N,ALPHA,X,INCX,AP)	$A \leftarrow \alpha\mathbf{x}\mathbf{x}^H + A$	圧縮型ｴﾙﾐｰﾄ	C,Z
_HER2(UPLO,N,ALPHA,X,INCX,Y,INCY,A,LDA)	$A \leftarrow \alpha\mathbf{x}\mathbf{y}^H + y(\alpha\mathbf{x})^H + A$	ｴﾙﾐｰﾄ	C,Z
_HPR2(UPLO,N,ALPHA,X,INCX,Y,INCY,AP)	$A \leftarrow \alpha\mathbf{x}\mathbf{y}^H + y(\alpha\mathbf{x})^H + A$	圧縮型ｴﾙﾐｰﾄ	C,Z
_SYR(UPLO,N,ALPHA,X,INCX,A,LDA)	$A \leftarrow \alpha\mathbf{x}\mathbf{x}^T + A$	対称	S,D
_SPR(UPLO,N.ALPHA,X,INCX,AP)	$A \leftarrow \alpha\mathbf{x}\mathbf{x}^T + A$	圧縮型対称	S,D
_SYR2(UPLO,N,ALPHA,X,INCX,Y,INCY,A,LDA)	$A \leftarrow \alpha\mathbf{x}\mathbf{y}^T + \alpha\mathbf{y}\mathbf{x}^T + A$	対称	S,D
_SPR2(UPLO,N,ALPHA,X,INCX,Y,INCY,AP)	$A \leftarrow \alpha\mathbf{x}\mathbf{y}^T + \alpha\mathbf{y}\mathbf{x}^T + A$	圧縮型対称	S,D

表 1.1.3　BLAS3 の主なルーチンの機能概要

ルーチン名と引数	演算	行列の型	型(_)
_GEMM(TRANSA,TRANSB,M,N,K,ALPHA, A,LDA,B,LDB,BETA,C,LDC)	$C \leftarrow \alpha\ op(A)op(B) + \beta C$ $op(X)=X, X^T, X^H$	密	S,D,C,Z
_SYMM(SIDE,UPLO,M,N,ALPHA, A,LDA,B,LDB,BETA,C,LDC)	$C \leftarrow \alpha AB + \beta C$ $C \leftarrow \alpha BA + \beta C$	対称	S,D,C,Z
_HEMM(SIDE,UPLO,M,N,ALPHA, A,LDA,B,LDB,BETA,C,LDC)	$C \leftarrow \alpha AB + \beta C$ $C \leftarrow \alpha BA + \beta C$	ｴﾙﾐｰﾄ	C,Z
_SYRK(UPLO,TRANS,N,K,ALPHA, A,LDA,BETA,C,LDC)	$C \leftarrow \alpha AA^T + \beta C$ $C \leftarrow \alpha A^TA + \beta C$	対称	S,D,C,Z
_HERK(UPLO,TRANS,N,K,ALPHA, A,LDA,BETA,C,LDC)	$C \leftarrow \alpha AA^H + \beta C$ $C \leftarrow \alpha A^HA + \beta C$	ｴﾙﾐｰﾄ	C,Z
_SYR2K(UPLO,TRANS,N,K,ALPHA, A,LDA,B,LDB,BETA,C,LDC)	$C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$ $C \leftarrow \alpha A^TB + \alpha B^TA + \beta C$	対称	S,D,C,Z
_HER2K(UPLO,TRANS,N,K,ALPHA, A,LDA,B,LDB,BETA,C,LDC)	$C \leftarrow \alpha AB^H + \alpha BA^H + \beta C$ $C \leftarrow \alpha A^HB + \alpha B^HA + \beta C$	ｴﾙﾐｰﾄ	C,Z

ルーチン名と引数	演算	行列の型	型(_)
_TRMM(SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA, A,LDA,B,LDB)	B ← α op(A)B B ← α Bop(A) op(A)=A, A^T, A^H	三角	S,D,C,Z
_TRSM(SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA, A,LDA,B,LDB)	B ← α op(A^{-1})B B ← α Bop(A^{-1}) op(A)=A, A^T, A^H	三角	S,D,C,Z

●表 1.1.1～表 1.1.3 で使用されている記号の説明

（1）引数の説明

・TRANS、TRANSA、TRANSB には、'N'、'T'、'C'のいずれかを指定する。

'N'：転置せず（A）　'T'：転置（A^T）　'C'：共役転置（A^H）

・UPLO には、'U'、'L'のいずれかを指定する。

'U'：上三角行列　'L'：下三角行列

・DIAG には、'N'、'U'のいずれかを指定する。

'N'：単位三角行列でない　'U'：単位三角行列（対角要素が 1 の三角行列）

・SIDE には、'L'、'R'のいずれかを指定する。

'L'：左からの行列の乗算　'R'：右からの行列の乗算

・M、N には、それぞれ行列の行数、行列の列数を指定する。

・KL、KU には、それぞれ下三角部分、上三角部分の（対角部分を含まない）帯幅を指定する。

・K には、三角部分あるいは上三角部分の（対角部分を含まない）帯幅を指定する（対称、エルミートあるいは三角行列の場合）。

・LDA、LDB、LDC には、配列 A、B、C の整合寸法を指定する。

・INCX、INCY には、配列 X、Y での計算対象とする要素のアドレス増分値を指定する。連続の場合は 1 を指定する。

（2）演算の説明

α , β はスカラー、**x**, **y** はベクトル、 A, B, C は行列を示す。

A^Tは A の転置、 A^H は A の共役転置、A^{-1} ,A^{-T} ,A^{-H}は、A ,A^T ,A^Hの逆行列を示す。

1.2 LAPACK ライブラリーの概要

LAPACK ライブラリーには、連立一次方程式、固有値問題などの行列の数値解法がサブルーチンとして用意されています。LAPACK ライブラリーには、代表的な問題を解くためのドライバールーチン、個々の計算を実行するための計算ルーチン、および計算ルーチンの機能的補助をする補助ルーチンがあります。

LAPACK ライブラリーの各ルーチンの名前は、BLAS ライブラリーと同様に、取り扱う変数配列の種類（実数／複素数、単精度／倍精度）により、最初の文字が決められています。

なお、SR8000 版では、netlib の LAPACK Version 2.0 で定義されたサブルーチンおよび関数

のうち、倍精度実数と倍精度複素数の二つの型だけのサポートとなっています。

LAPACK ライブラリーの主なルーチンの機能概要を表 1.2.1～表 1.2.3 に示します。表において、「型」はルーチン名の最初の文字（__の個所）を置き換えることのできる文字を表わしています。各ルーチンは主にサブルーチン形式ですが、関数形式のものは、先頭に FUNCTION をつけて表わしています。

なお、LAPACK の機能仕様の詳細につきましては、以下を参照して下さい。

http://www.netlib.org/lapack/

表 1.2.1　LAPACK の主なドライバールーチンの機能概要

ルーチン名	機能概要	行列の型、補足説明	型(_)
_GESV	連立一次方程式の解の計算	非対称密	D,Z
_GBSV		非対称帯	D,Z
_GTSV		非対称三重対角	D,Z
_POSV		対称／エルミート正定値	D,Z
_PBSV		対称／エルミート正定値帯	D,Z
_PTSV		対称／エルミート正定値三重対角	D,Z
_SYSV		対称非正定値	D,Z
_HESV		エルミート非正定値	Z
___EV	標準固有値、固有ベクトルの計算	対称／エルミート	DSY,ZHE
__BEV		対称／エルミート帯	DS,ZH
_STEV		対称三重対角	D
_GEEV		非対称	D,Z
___GV	一般化固有値、固有ベクトルの計算	対称／エルミート	DSY,ZHE
__BGV		対称／エルミート帯	DS,ZH
_GEGV		非対称	D,Z
_GESVD	特異値分解	長方	D,Z
_GGSVD	一般化特異値分解	長方	D,Z
_GELS	線形最小二乗問題	QR/LQ 分解を使用	D,Z
_GELSX		完全直行分解を使用	D,Z
_GELSS		特異値分解を使用	D,Z
_GELS	一般化線形最小二乗問題	線形等式制約問題	D,Z
_GELSX		一般(ｶﾞｳｽ-ﾏﾙｺﾌ)線形モデル問題	D,Z

連立方程式と標準固有値問題用のドライバールーチンには、それぞれの問題について単純ドライバーとエキスパートドライバーの二種類を持つものがあります。エキスパートドライバーは、上記単純ドライバーのルーチン名の末尾に文字'X'を付加したルーチン名を持っており、連立方程式の解の反復改良や、標準固有値問題で指定した範囲の固有値、固有ベクトルを求めることなどができます。

表 1.2.2　LAPACK の主な計算ルーチンの機能概要

ルーチン名	演算	機能概要	型(_)
_GETRF	三角分解	正方行列を部分軸選択付き LU 分解する。	D,Z
_POTRF		対称およびエルミート行列をコレスキー分解する。	D,Z
_GETRS	連立一次方程式の求解	_GETRF で計算した LU 分解を用いて、連立一次方程式の解を求める。	D,Z
_POTRS		_POTRF で計算した結果を用いて、連立一次方程式の解を求める。	D,Z
_GETRI	逆行列の計算	_GETRF で計算した LU 分解を用いて、正方行列の逆行列を計算する。	D,Z
_POTRI		_POTRF で計算した結果を用いて、逆行列を計算する。	D,Z
_TRTRI		三角行列の逆行列を計算する。	D,Z
_GELQF	直交分解	長方行列を LQ 分解する。	D,Z
_GEQLF		長方行列を QL 分解する。	D,Z
_GEQPF		長方行列を部分軸選択つき QR 分解する。	D,Z
_GEQRF		長方行列を QR 分解する。	D,Z
_GERQF		長方行列を RQ 分解する。	D,Z
_GEBRD	直交変換	正規直交変換で長方行列を二重対角形にする。	D,Z
___TRD		正規直交変換で対称およびエルミート行列を三重対角行列にする。	DSY, ZHE
_GEHRD		正規直交変換で長方行列を上ヘッセンベルク形にする。	D,Z
_GECON	行列の条件数計算	_GETRF で計算した LU 分解を用いて、1 ノルムか無限ノルムのもとで正方行列の条件数を計算する。	D,Z
_POCON		_POTRF により計算したコレスキー分解を用いて、対称およびエルミート行列の条件数を計算する。	D,Z
_GERFS	連立一次方程式の計算解の改良	正方行列の連立一次方程式の計算解を反復改良する。解の誤差限界を求める。	D,Z
_PORFS		対称およびエルミート行列の連立一次方程式の計算解を反復改良する。解の誤差限界を求める。	D,Z

表 1.2.3　LAPACK の主な補助ルーチンの機能概要

ルーチン名	機能	機能概要	型(_)
_LASET	行列の要素の初期化	行列の非対角要素をα、対角要素をβで初期化する。	D,Z
FUNCTION _LANGE	ノルム値、最大要素絶対値を求める	長方行列の各種ノルムまたは最大要素絶対値を求める。	D,Z
FUNCTION _LANTR		三角行列の各種ノルムまたは最大要素絶対値を求める。	D,Z
FUNCTION _LANSY		実対称および複素対称行列の各種ノルムまたは最大要素絶対値を求める。	D,Z

ルーチン名	機能	機能概要	型(_)
FUNCTION _LANHE		エルミート行列の各種ノルムまたは最大要素絶対値を求める。	Z
_GETF2	三角分解	正方行列の部分軸選択付き LU 分解をする。	D,Z
_POTF2		対称およびエルミート正定値行列をコレスキー分解する。	D,Z
_GELQ2	直交分解	長方行列の LQ 分解をする。	D,Z
_GEQL2		長方行列の QL 分解をする。	D,Z
_GEQR2		長方行列の QR 分解をする。	D,Z
_GERQ2		長方行列の RQ 分解をする。	D,Z
_GEBD2	直交変換	正規直交変換により長方行列を二重対角形へ変換する。	D,Z
_SYTD2		正規直交変換により対称およびエルミート行列を実対称三重対角形へ変換する。	D,Z
_GEHD2		正規直交変換により長方行列を上ヘッセンベルク形へ変換する。	D,Z
___GS2	固有値問題の標準形への変換	対称およびエルミートの一般固有値問題を標準固有値問題へ変換する。	DSY, ZHE

1.3 ScaLAPACK ライブラリーの概要

ScaLAPACK ライブラリーでは、連立一次方程式、固有値問題などの行列の数値解法がサブルーチンとして用意されています。ScaLAPACK ライブラリーは、LAPACK ライブラリーの分散メモリー型並列計算機用ライブラリーです。本ライブラリーは、BLACS（Basic Linear Algebra Communication Subprograms）、PBLAS（Parallel Basic Linear Algebra Subprograms）などを含み、通信関数に MPI を使用しています。

ScaLAPACK ライブラリーには、代表的な問題を解くためのドライバールーチン、個々の計算を実行するための計算ルーチン、計算ルーチンの機能的補助をする補助ルーチン、各プロセッサーに分散された行列を管理するテーブルの初期設定やインデックス計算などを行う共通ルーチンおよび MPI を使った通信ルーチンがあります。

ScaLAPACK ライブラリーの各ルーチンの名前は文字'P'で始まり、BLAS ライブラリー等と同様に、取り扱う変数配列の種類（実数／複素数、単精度／倍精度）によって二番目の文字が決められています。

なお、SR8000 版では、netlib の ScaLAPACK Version 1.6 で定義されたサブルーチンおよび関数のうち、倍精度実数と倍精度複素数の二つの型だけのサポートとなっています。

ScaLAPACK ライブラリーの主要なルーチンの機能概要を表 1.3.1～表 1.3.3 に示します。表において、「型」はルーチン名の二番目の文字（__の個所）を置き換えることのできる文字を表わしています。各ルーチンは主にサブルーチン形式ですが、関数形式のものは、先頭に FUNCTION をつけて表わしています。

なお、ScaLAPACK の機能仕様の詳細につきましては、以下を参照して下さい。

http://www.netlib.org/scalapack/

表 1.3.1　ScaLAPACK の主なドライバールーチンの機能概要

ルーチン名	機能概要	行列の型、補足説明	型(_)
P_GESV	連立一次方程式の解の計算	非対称密	D,Z
P_GBSV		非対称帯	D,Z
P_POSV		対称／エルミート正定値	D,Z
P_PBSV		対称／エルミート正定値帯	D,Z
P_PTSV		対称／エルミート正定値三重対角	D,Z
P___EV	標準固有値、固有ベクトルの計算	対称／エルミート	DSY,ZHE
P___GVX	一般化固有値、固有ベクトルの計算	対称／エルミート	DSY,ZHE
P_GESVD	特異値分解	長方	D,Z
P_GELS	線形最小二乗問題	QR/LQ 分解を使用	D,Z

連立方程式と標準固有値問題用のドライバールーチンには、それぞれの問題について単純ドライバーとエキスパートドライバーの二種類を持つものがあります。エキスパートドライバーは、上記単純ドライバーのルーチン名の末尾に文字'X'を付加したルーチン名を持っており、連立方程式の解の反復改良や、標準固有値問題で指定した範囲の固有値、固有ベクトルを求めることなどができます。ScaLAPACK ライブラリーには LAPACK ライブラリー程多くのドライバールーチンは用意されていませんが、利用者は個々の計算ルーチンを組み合わせることにより、上記以外の問題も解くことができます。

表 1.3.2　ScaLAPACK の主な計算ルーチンの機能概要

ルーチン名	演算	機能概要	型(_)
P_GETRF	三角分解	正方行列を部分軸選択付き LU 分解する。	D,Z
P_POTRF		対称およびエルミート行列をコレスキー分解する。	D,Z
P_GETRS	連立一次方程式の求解	P_GETRF で計算した LU 分解を用いて、連立一次方程式の解を求める。	D,Z
P_POTRS		P_POTRF で計算した結果を用いて、連立一次方程式の解を求める。	D,Z
P_GETRI	逆行列の計算	P_GETRF で計算した LU 分解を用いて、正方行列の逆行列を計算する。	D,Z
P_POTRI		P_POTRF で計算した結果を用いて、逆行列を計算する。	D,Z
P_TRTRI		三角行列の逆行列を計算する。	D,Z
P_GELQF	直交分解	長方行列を LQ 分解する。	D,Z
P_GEQLF		長方行列を QL 分解する。	D,Z
P_GEQPF		長方行列を部分軸選択つき QR 分解する。	D,Z
P_GEQRF		長方行列を QR 分解する。	D,Z
P_GERQF		長方行列を RQ 分解する。	D,Z
P_GEBRD	直交変換	正規直交変換で長方行列を二重対角形にする。	D,Z

ルーチン名	演算	機能概要	型(_)
P___TRD		正規直交変換で対称およびエルミート行列を三重対角行列にする。	DSY, ZHE
P_GEHRD		正規直交変換で長方行列を上ヘッセンベルク形にする。	D,Z
P_GECON	行列の条件数計算	P_GETRF で計算した LU 分解を用いて、1 ノルムか無限ノルムのもとで正方行列の条件数を計算する。	D,Z
P_POCON		P_POTRF により計算したコレスキー分解を用いて、対称およびエルミート行列の条件数を計算する。	D,Z
P_GERFS	連立一次方程式の計算解の改良	正方行列の連立一次方程式の計算解を反復改良する。解の誤差限界を求める。	D,Z
P_PORFS		対称およびエルミート行列の連立一次方程式の計算解を反復改良解の誤差限界を求める。	D,Z

表 1.3.3　ScaLAPACK の主な補助ルーチンの機能概要

ルーチン名	機能	機能概要	型(_)
P_LASET	行列の要素の初期化	行列の非対角要素をα、対角要素をβで初期化する。	D,Z
FUNCTION P_LANGE	ノルム値、最大要素絶対値を求める	長方行列の各種ノルムまたは最大要素絶対値を求める。	D,Z
FUNCTION P_LANTR		三角行列の各種ノルムまたは最大要素絶対値を求める。	D,Z
FUNCTION P_LANSY		実対称および複素対称行列の各種ノルムまたは最大要素絶対値を求める。	D,Z
FUNCTION P_LANHE		エルミート行列の各種ノルムまたは最大要素絶対値を求める。	Z
P_GETF2	三角分解	正方行列の部分軸選択付き LU 分解をする。	D,Z
P_POTF2		対称およびエルミート正定値行列をコレスキー分解する。	D,Z
P_GELQ2	直交分解	長方行列の LQ 分解をする。	D,Z
P_GEQL2		長方行列の QL 分解をする。	D,Z
P_GEQR2		長方行列の QR 分解をする。	D,Z
P_GERQ2		長方行列の RQ 分解をする。	D,Z
P_GEBD2	直交変換	正規直交変換により長方行列を二重対角形へ変換する。	D,Z
P_SYTD2		正規直交変換により対称およびエルミート行列を実対称三重対角形へ変換する。	D,Z
P_GEHD2		正規直交変換により長方行列を上ヘッセンベルク形へ変換する。	D,Z
P___GS2	固有値問題の標準形への変換	対称およびエルミートの一般固有値問題を標準固有値問題へ変換する。	DSY, ZHE

2. 各ライブラリーの使用方法と注意事項

2.1　64 ビット版ライブラリーと 32 ビット版ライブラリーについて

通常は、64 ビット版ライブラリーをご使用ください。アドレッシングモードが 64 ビットであるため、2GB を超える大きな配列を使用することができます。しかし、他の 32 ビット版ライブラリーと併せて使用する時は、32 ビット版をご使用ください。64 ビットオブジェクトと 32 ビットオブジェクトの混在はできません。

2.2　要素並列版ライブラリーとスカラー版ライブラリーについて

通常は、要素並列版ライブラリーをご使用ください。要素並列版ライブラリーはノード内の複数プロセッサーを用いて要素並列実行されますので、利用者はプログラムの並列化を意識することなく、高い性能を得るプログラムを作成することができます。

ただし、次のような場合はスカラー版ライブラリーを使用する必要があります。

（1）プログラム全体が MPI などを用いてプロセッサー単位で並列化されている場合

（2）プログラムが BLAS、あるいは LAPACK のサブルーチンまたは関数をその外側のループで並列化して実行する場合

（3）問題が小規模で、配列のサイズが小さい場合（本来の処理に比べ、要素並列化によるオーバーヘッドが無視できなくなるため、要素並列版ライブラリーを使用して実行するほうが却って遅くなる場合があります。目安として、BLAS1 ライブラリーの場合、配列サイズが約 1000 以下ではスカラー版ライブラリーの方が速くなります。BLAS2 ライブラリーの場合でも、配列サイズが約 100 以下ではスカラー版の方が速くなります。）

ScaLAPACK ライブラリーでは、ノード内を要素並列化により並列実行する場合は要素並列版を、ノード内外の各プロセッサーを MPI などを用いて並列実行する場合はスカラー版を使用します。

なお、一つのプログラムの中で、要素並列版とスカラー版のライブラリーを併用することはできません。

2.3 ブロックサイズの指定とプロセッサーの割り当てについて

ScaLAPACK ライブラリーでは、ブロック化アルゴリズムを採用して計算の高速化を図っています。適切なブロックサイズを指定することにより、プログラムの高速化ができます。適切なブロックサイズの値は、使用するルーチンにより多少変化しますが、200 程度を目安としてください。

さらに、ScaLAPACK ライブラリーでは、ノードの割り当て方によっても性能が変わってきます。一般に、P 台のノードで実行する場合、プロセッサーグリッド（P＝Pr×Pc）として Pr と Pc を指定しますが、このとき Pr と Pc を同じ値か近い値にするのが効果的です。ただし、一部の処理については、表 2.2.1 の指定が効果的です。

表 2.2.1　プロセッサーグリッドの指針（P＝Pr×Pc）

処理	指針	P=8 の例	P=16 の例	P=32 の例
LU 分解、直交分解（QR,QL 分解）	Pr＜Pc	1×8	2×8	2×16
直交分解（LQ,RQ 分解）	Pr＞Pc	8×1	8×2	8×4

2.4　利用者プログラムのコンパイルおよび各ライブラリーとのリンク方法

ここでは、BLAS、LAPACK、ScaLAPACK を使用するプログラムのコンパイル、リンク方法をご紹介します。

手順において、誌面の都合で一行に入りきらない入力行は、「¥」で折り返して記述しています。また、sample.f、sample.c は、ソースプログラムの FORTRAN 版、C 版とし、sample.out はロードモジュールファイル名とします。

2.4.1　BLAS の使用方法

（1）FORTRAN の場合

・64 ビットアドレッシングモードの要素並列版ライブラリーを使用する場合

```
% f90 -64 -parallel -o sample.out sample.f -lblas
```

・64 ビットアドレッシングモードのスカラー版ライブラリーを使用する場合

```
% f90 -64 -noparallel -o sample.out sample.f -lblas_sc
```

・32 ビットアドレッシングモードの要素並列版ライブラリーを使用する場合

```
% f90 -32 -parallel -o sample.out sample.f -lblas
```

・32 ビットアドレッシングモードのスカラー版ライブラリーを使用する場合

```
% f90 -32 -noparallel -o sample.out sample.f -lblas_sc
```

（2）C の場合

・64 ビットアドレッシングモードの要素並列版ライブラリーを使用する場合

```
% cc -64 +03 -parallel -c sample.c
% f90 -64 -parallel -o sample.out sample.o -lblas
```

・64 ビットアドレッシングモードのスカラー版ライブラリーを使用する場合

```
% cc -64 -noparallel -c sample.c
% f90 -64 -noparallel -o sample.out sample.o -lblas_sc
```

・32 ビットアドレッシングモードの要素並列版ライブラリーを使用する場合

```
% cc -32 +03 -parallel -c sample.c
% f90 -32 -parallel -o sample.out sample.o -lblas
```

・32 ビットアドレッシングモードのスカラー版ライブラリーを使用する場合

```
% cc -32 -noparallel -c sample.c
% f90 -32 -noparallel -o sample.out sample.o -lblas_sc
```

2.4.2　LAPACK の使用方法

LAPACK ライブラリーを使用する際には、プログラムのリンク時に BLAS ライブラリーもリンクする必要があります。

（1）FORTRAN の場合

・64 ビットアドレッシングモードの要素並列版ライブラリーを使用する場合

```
% f90 -64 -parallel -o sample.out sample.f -llapack -lblas
```

・64 ビットアドレッシングモードのスカラー版ライブラリーを使用する場合

```
% f90 -64 -noparallel -o sample.out sample.f -llapack_sc -lblas_sc
```

・32 ビットアドレッシングモードの要素並列版ライブラリーを使用する場合

```
% f90 -32 -parallel -o sample.out sample.f -llapack -lblas
```

・32 ビットアドレッシングモードのスカラー版ライブラリーを使用する場合

```
% f90 -32 -noparallel -o sample.out sample.f -llapack_sc -lblas_sc
```

（2）C の場合

・64 ビットアドレッシングモードの要素並列版ライブラリーを使用する場合

```
% cc -64 +03 -parallel -c sample.c
% f90 -64 -parallel -o sample.out sample.o -llapack -lblas
```

・64 ビットアドレッシングモードのスカラー版ライブラリーを使用する場合

```
% cc -64 -noparallel -c sample.c
% f90 -64 -noparallel -o sample.out sample.o -llapack_sc -lblas_sc
```

・32 ビットアドレッシングモードの要素並列版ライブラリーを使用する場合

```
% cc -32 +03 -parallel -c sample.c
% f90 -32 -parallel -o sample.out sample.o -llapack -lblas
```

・32 ビットアドレッシングモードのスカラー版ライブラリーを使用する場合

```
% cc -32 -noparallel -c sample.c
% f90 -32 -noparallel -o sample.out sample.o -llapack_sc -lblas_sc
```

2.4.3 ScaLAPACK の使用方法

ScaLAPACK ライブラリーを使用する際には、プログラムのリンク時に BLAS ライブラリー、および MPI ライブラリーもリンクする必要があります。

（1）FORTRAN の場合

・64 ビットアドレッシングモードの要素並列版ライブラリーを使用する場合

```
% f90 -64 -parallel -o sample.out sample.f -lscalapack -lpblas \
      -ltools -lredist -lblacsBASE -lblacsF77 -lblacsBASE -lblas \
      -L/usr/mpi/lib/lib64 -lfmpi -lmpi
```

・64 ビットアドレッシングモードのスカラー版ライブラリーを使用する場合

```
% f90 -64 -noparallel -o sample.out sample.f -lscalapack_sc \
      -lpblas_sc -ltools_sc -lredist_sc -lblacsBASE_sc \
      -lblacsF77_sc -lblacsBASE_sc -lblas_sc \
```

```
        -L/usr/mpi/lib/lib64  -lfmpi  -lmpi
```

・32 ビットアドレッシングモードの要素並列版ライブラリーを使用する場合

```
% f90  -32  -parallel  -o  sample.out  sample.f  -lscalapack  -lpblas \
        -ltools  -lredist  -lblacsBASE  -lblacsF77  -lblacsBASE  -lblas \
        -L/usr/mpi/lib/lib32  -lfmpi  -lmpi
```

・32 ビットアドレッシングモードのスカラー版ライブラリーを使用する場合

```
% f90  -32  -noparallel  -o sample.out  sample.f  -lscalapack_sc \
        -lpblas_sc  -ltools_sc  -lredist_sc  -lblacsBASE_sc \
        -lblacsF77_sc  -lblacsBASE_sc  -lblas_sc \
        -L/usr/mpi/lib/lib32  -lfmpi  -lmpi
```

(2) C の場合

・64 ビットアドレッシングモードの要素並列版ライブラリーを使用する場合

```
% cc  -64  +03  -parallel  -c  sample.c
% f90  -64  -parallel  -o  sample.out  sample.o  -lscalapack  -lpblas \
        -ltools  -lredist  -lblacsBASE  -lblacsC  -lblacsBASE  -lblas \
        -L/usr/mpi/lib/lib64  -lfmpi  -lmpi
```

・64 ビットアドレッシングモードのスカラー版ライブラリーを使用する場合

```
% cc  -64  -noparallel  -c  sample.c
% f90  -64  -noparallel  -o sample.out  sample.o  -lscalapack_sc \
        -lpblas_sc  -ltools_sc  -lredist_sc  -lblacsBASE_sc \
        -lblacsC_sc  -lblacsBASE_sc  -lblas_sc \
        -L/usr/mpi/lib/lib64  -lfmpi  -lmpi
```

・32 ビットアドレッシングモードの要素並列版ライブラリーを使用する場合

```
% cc  -32  +03  -parallel  -c  sample.c
% f90  -32  -parallel  -o  sample.out  sample.o  -lscalapack  -lpblas \
        -ltools  -lredist  -lblacsBASE  -lblacsC  -lblacsBASE  -lblas \
        -L/usr/mpi/lib/lib32  -lfmpi  -lmpi
```

・32 ビットアドレッシングモードのスカラー版ライブラリーを使用する場合

```
% cc  -32  -noparallel  -c  sample.c
% f90  -32  -noparallel  -o sample.out  sample.o  -lscalapack_sc \
        -lpblas_sc  -ltools_sc  -lredist_sc  -lblacsBASE_sc \
        -lblacsC_sc  -lblacsBASE_sc  -lblas_sc \
        -L/usr/mpi/lib/lib32  -lfmpi  -lmpi
```

3. プログラム作成上の注意事項

（1）行列の格納形式は、行列の型により決められています。

実対称行列、エルミート行列、帯行列、上、下三角行列に対して圧縮形式、一次元配列、二次元配列で各々に格納形式があります。アドレス方向にも注意して下さい。

格納形式についての詳細は以下を参照して下さい。

http://www.netlib.org/lapack/lug/node121.html

http://www.netlib.org/scalapack/slug/node68.html

（2）本ライブラリーを C プログラムから使用するときは次のことに注意して下さい。

・配列変数のアドレス方向を FORTRAN のアドレス方向に合わせる。

FORTRAN では A(1,1), A(2,1), A(3,1),・・・の順、

C では A[1][1], A[1][2], A[1][3],・・・の順となります。

・呼び出す関数名称は FORTRAN から呼び出す場合のサブルーチン名称と同様です。関数名称は、大文字でも小文字でも使用できますが、大文字、小文字が混在する名前は使用できません。

例）　ddot, DDOT：使用可　　Ddot, dDOT：使用不可

・引数は、すべてアドレス渡しでなければなりません。

例）X,Y が配列変数として、DDOT 関数を使う場合

FORTRAN での呼び出し

```
S = DDOT(1000, X, 1, Y, 1)
```

は、C では、

```
n = 1000; incx = 1; incy = 1;
s = ddot(&n, x, &incx, y, &incy);
```

とします。

4. サンプルプログラムのご紹介

BLAS ライブラリー、LAPACK ライブラリー、ScaLAPACK ライブラリーを使用したサンプルプログラムをそれぞれ以下のディレクトリーにインストールしてあります。

/usr/examples/BLAS/

/usr/examples/LAPACK/

/usr/examples/ScaLAPACK/

Makefile やプログラム実行用のスクリプトもありますので、コンパイル方法や実行方法の参考資料としてご利用いただければ幸いです。

以上