ハイブリッドMPI-OpenMP並列版自動チューニングライブラリ

pXabclib 0.1.0

詳細仕様書

東京大学情報基盤センター

(株)日立製作所中央研究所

２０１４年１２月１７日

DISCLAIMER

目次

全 35 頁

1 概要

本文書では「自動チューニングライブラリのハイブリッド MPI-OpenMP 並列化」で開発する pXabclib および pOpenATLib の仕様について記述する。

本件では以下の機能を開発する。

・pOpenATI_DAFDC_setup,pOpenATI_DAFDC　各 rank の担当するローカル行列の作成および配布

・pOpenATI_DURMV_presetup,pOpenATI_DURMV_setup　SpMV に必要な通信情報と各 thread 担当行情報の作成

・pOpenATI_DURMV　　MPI-OpenMP でハイブリッド並列化された SpMV

・pXabclib_BICGSTAB　　MPI-OpenMP でハイブリッド並列化された BiCGStab 法ソルバ

【用語定義】

・SpMV：疎行列ベクトル積

・rank：MPI 並列で割り振られるプロセス

・thread：OpenMP 並列で設定する実行単位

・グローバル行列：全体行列

・ローカル行列：グローバル行列を分割し、各 rank で分担して持つ部分行列

・ハイブリッド並列：MPI と OpenMP を併用した並列化方針

・CRS(Compressed Row Storage)形式：要素を行方向に圧縮して保持する行列格納形式

2 pXabclib 使用における前提条件

・rank 番号は 0~NUMPROCS-1 で割り当てる。

・各 rank がローカルな行列およびベクトル(右辺、初期近似解)を保持している。(※1)

・行列分割情報(DECOMP)が設定されている。(※1)

・分割方式は行列、ベクトル共に rank 番号順に連続する行を保持する方式とする。

・分割方法は各 rank が担当する非零要素数が均等に近づくように分割する。

・行列の格納形式は CRS 形式とする。

・パラメーターリスト(IATPARAM,RATPARAM)に初期値が設定されている。

・同じ行列を入力として使用していても、rank 数か thread 数を変更した場合は pOpenATI_DURMV_presetup と pOpenATI_DURMV_setup を再度呼び出す必要がある。

(※1)rank0 のみがグローバルな行列、ベクトルを保持している場合は、pOpenATI_DAFDC_setup,pOpenATI_DAFDC を用いて行列分割情報の作成と各 rank にローカルな行列およびベクトルを送付する。

3 pOpenATI_init

3.1 pOpenATI_init の用途

パラメーターリストである IATPARAM と RATPARAM に初期値を設定する関数。

3.2 引数仕様

変数名	型	I/O	説明
IATPARAM(200)	INT	I	INTEGER 型パラメーターリスト
RATPARAM(200)	DOUBLE	I	DOUBLE 型パラメーターリスト
INFO	INT	O	エラーコード INFO=0:正常値

3.3 IATPARAM 仕様

番号	初期値	I/O	説明
OpenATLib's Information[3~100]			
3	OMP_GET_MAX_THREADS	I	OpenMP の使用 thread 数
9	0	I	SpMV の自動チューニング機能の ON/OFF (自動チューニング機能は未実装)
10	12	I	SpMV の行列の thread 分割方式の選択 (thread 分割は非零要素数分割のみ実装)
Xabclib's Information[101~200]			
102	-1	I	ソルバ内最大反復回数 IATPARAM(102)=-1:次元数 N を設定
103	-	O	ソルバ内反復回数
105	2	I	前処理の選択 IATPARAM(105)=1:前処理なし IATPARAM(105)=2:Jacobi 前処理
111	-	O	SpMV 演算回数
115	0	O	MPI エラーコード IATPARAM(115)=0:正常値 IATPARAM(115)≠0:異常値
200	0	I	デバッグプリントの ON/OFF IATPARAM(200)=0:デバッグプリントなし IATPARAM(200)=1:デバッグプリントあり

3.4 RATPARAM 仕様

番号	初期値	I/O	説明
OpenATLib's Information[3~100]			
7	-	O	pOpenATI_DURMV 内の行列ベクトル積の 1 回の演算時間
8	-	O	pOpenATI_DURMV での 1 回の MPI 並列処理のオーバーヘッド時間
Xabclib's Information[101~200]			
102	-1	I	ソルバ最大演算時間
103	1.0E-8	I	収束判定基準値
108	-	O	右辺ベクトルのノルム
109	-	O	最終残差(||Ax-b||_2/||b||_2)
110	-	O	ソルバの演算量
111	-	O	前処理の演算時間
112	-	O	ソルバの演算時間
115	-	O	pOpenATI_DURMV 内の行列ベクトル積の合計演算時間
116	-	O	pOpenATI_DURMV での MPI 並列処理のオーバーヘッド時間の合計
117	-	O	pOpenATI_DURMV 以外での MPI 通信時間(他 rank とのノルム計算の和や、エラーコードの集計に使用)

4 pOpenATI_DAFDC_setup

4.1 pOpenATI_DAFDC_setup の用途

各 rank の担当するローカル行列の作成および配布を行う関数 pOpenATI_DAFDC(詳細は 5 章)のセットアップ用関数。グローバル行列の分割情報とローカル行列の次元数、非零要素数を計算する。

4.2 引数仕様

CALL pOpenATI_DAFDC_setup(NL,NZL,N,NZ,IRP,MYID,NUMPROCS,COMM,DECOMP,INFO)

変数名	型	I/O	説明
NL	INT*8	O	自 rank の次元数
NZL	INT*8	O	自 rank の非零要素数
N	INT*8	I	グローバル行列の次元数
NZ	INT*8	I	グローバル行列の非零要素数
IRP(N+1)	INT*8	I	グローバル行列の非零要素の各行先頭へのポインタ
MYID	INT	I	自 rank 番号(0~NUMPROCS-1)
NUMPROCS	INT	I	総 rank 数
COMM	INT	I	MPI コミュニケータ
DECOMP(NUMPROCS+1)	INT*8	O	グローバル行列の分割情報。各 rank の先頭行番号(グローバルな番号)
INFO	INT	O	エラーコード INFO=0　：正常値 INFO=-I　：i 番目の引数の値が不正

4.3 処理手順

手順
1. work 領域の設定＆非零要素分割の閾値設定(PNZ,T)
2. 行列分割情報作成と各 rank の非零要素数のカウント(DECOMP,PNZ)
   (rank0 のみ行う)
3. rank0 から他 rank へ行列分割情報を送信(DECOMP を BCAST)
4. rank0 から各 rank へそれぞれの非零要素数を送信(NZL)
5. 自 rank の次元数を求める(NL)

4.3.1 work 領域の設定＆非零要素分割の閾値設定(PNZ,T)

(実装例)
ALLOCATE(PNZ(NUMPROCS))
T=NZ/NUMPROCS

4.3.2 行列分割情報作成と各 rank の非零要素数のカウント(DECOMP,PNZ)

```
(実装例)
IF(MYID.EQ.0)THEN
    K=1
    NZ_CNT=0
    DSUM2=0
    DECOMP(1)=1
    DO I=1,N
        DSUM1=DSUM2
        DSUM2+=IRP(I+1)-IRP(I)
        IF(DSUM2.GE.T)THEN
            IF(K.EQ.NUMPROCS)THEN
                GO TO 100
            END IF
            IF(ABS(DSUM1-T).LT.ABS(DSUM2-T))THEN
                DECOMP(K+1)=I
                PNZ(K)=DSUM1
                DSUM2=IRP(I+1)-IRP(I)
            ELSE
                DECOMP(K+1)=I+1
                PNZ(K)=DSUM2
                DSUM2=0
            END IF
            NZ_CNT+=PNZ(K)
            K++
            DSUM1=0
        END IF
    END DO
    100 CONTINUE
    DECOMP(NUMPROCS+1)=N+1
    PNZ(NUMPROCS)=NZ-NZ_CNT
END IF
```

4.3.3 rank0 から他 rank へ行列分割情報を送信(DECOMP を BCAST)

```
(実装例)
CALL MPI_BCAST(DECOMP,NUMPROCS+1,MPI_INTEGER8,0,COMM,IERR)
```

4.3.4 rank0 から各 rank へそれぞれの非零要素数を送信(NZL)

```
(実装例)
CALL MPI_SCATTER(PNZ,1,MPI_INTEGER8,NZL,1,MPI_INTEGER8,
                                0,COMM,IERR)
```

4.3.5 自 rank の次元数を求める(NL)

```
(実装例)
NL=DECOMP(MYID+2)-DECOMP(MYID+1)
```

5 pOpenATI_DAFDC

5.1 pOpenATI_DAFDC の用途

pOpenATI_DAFDC_setup の出力を利用し、各 rank の担当するローカル行列の作成および配布を行う関数。関数を呼び出す前にローカル行列の領域確保を行う必要がある。

5.2 引数仕様

CALL pOpenATI_DAFDC(NL,NZL,N,NZ,IRP,ICOL,VAL,X,B,IRP_L,ICOL_L,VAL_L,X_L,B_L,
MYID,NUMPROCS,COMM,DECOMP,INFO)

変数名	型	I/O	説明
NL	INT*8	I	自 rank の次元数
NZL	INT*8	I	自 rank の非零要素数
N	INT*8	I	グローバル行列の次元数
NZ	INT*8	I	グローバル行列の非零要素数
IRP(N+1)	INT*8	I	グローバル行列の非零要素の各行先頭へのポインタ
ICOL(NZ)	INT*8	I	グローバル行列の非零要素の列番号
VAL(NZ)	DOUBLE	I	グローバル行列の非零要素の値
X(N)	DOUBLE	I	グローバル行列のベクトルの値
B(N)	DOUBLE	I	グローバル行列の右辺ベクトル
IRP_L(NL+1)	INT*8	O	自 rank の非零要素の各行先頭へのポインタ
ICOL_L(NZL)	INT*8	O	自 rank の非零要素の列番号(グローバルな番号)
VAL_L(NZL)	DOUBLE	O	自 rank の非零要素の値
X_L(NL)	DOUBLE	O	自 rank のベクトルの値
B_L(NL)	DOUBLE	O	自 rank の右辺ベクトル
MYID	INT	I	自 rank 番号(0~NUMPROCS-1)
NUMPROCS	INT	I	総 rank 数
COMM	INT	I	MPI コミュニケータ
DECOMP(NUMPROCS+1)	INT*8	I	グローバル行列の分割情報。各 rank の先頭行番号(グローバルな番号)
INFO	INT	O	エラーコード INFO=0　：正常値 INFO=-I　：i 番目の引数の値が不正

5.3 処理手順

手順
1. pOpenATI_DAFDC 前にローカルな行列情報のリストの領域確保
   (IRP_L,ICOL_L,VAL_L,X_L,B_L)
2. mpi 通信用配列の設定(SSTAT,SREQ,RSTAT,RREQ)
3. エラーチェック
4. rank0 から各 rank へローカル行列を送信する(IRP_L,ICOL_L,VAL_L,X_L,B_L)
5. 各 rank は rank0 からローカル行列を受信する(IRP_L,ICOL_L,VAL_L,X_L,B_L)

5.3.1 pOpenATI_DAFDC 前にローカルな行列情報のリストの領域確保 (IRP_L,ICOL_L,VAL_L,X_L,B_L)

(実装例)
```
ALLOCATE(IRP_L(NL+1),ICOL_L(NZL),VAL_L(NZL),X_L(NL),B_L(NL))
```

5.3.2 mpi 通信用配列の設定(SSTAT,SREQ,RSTAT,RREQ)

(実装例)
```
ALLOCATE(SSTAT(MPI_STATUS_SIZE,(NUMPROCS-1)*5),SREQ((NUMPROCS-1)*5),
         RSTAT(MPI_STATUS_SIZE,5),RREQ(5))
```

5.3.3 エラーチェック

(実装例)
```
INFO_G = 0
//NL が 2G 以下か
IF (NL .GT. 2000000000) THEN
    INFO = -1
END IF
CALL MPI_ALLREDUCE(INFO,INFO_G,1,MPI_INTEGER,MPI_SUM,COMM,IERR)
//NZL が 2G 以下か
IF (NZL .GT. 2000000000) THEN
    INFO = -2
END IF
CALL MPI_ALLREDUCE(INFO,INFO_G,1,MPI_INTEGER,MPI_SUM,COMM,IERR)
```

5.3.4 rank0 から各 rank へローカル行列を送信する

(IRP_L,ICOL_L,VAL_L,X_L,B_L)

```
(実装例)
CNT = 0
IF (MYID .EQ. 0) THEN
    DO IP = 2, NUMPROCS
        ST = DECOMP(IP)
        NL_TMP = INT(DECOMP(IP+1) - ST,4)
        NZL_TMP = INT(IRP(DECOMP(IP+1)) - IRP(ST),4)
        CALL MPI_ISEND(IRP(ST),NL_TMP+1,MPI_INTEGER8,IP-1,
                                5*CNT+1,COMM,SREQ(5*CNT+1),IERR)
        CALL MPI_ISEND(X(ST),NL_TMP,MPI_DOUBLE_PRECISION,IP-1,
                                5*CNT+2,COMM,SREQ(5*CNT+2),IERR)
        CALL MPI_ISEND(B(ST),NL_TMP,MPI_DOUBLE_PRECISION,IP-1,
                                5*CNT+3,COMM,SREQ(5*CNT+3),IERR)
        CALL MPI_ISEND(ICOL(IRP(ST)),NZL_TMP,MPI_INTEGER8,IP-1,
                                5*CNT+4,COMM,SREQ(5*CNT+4),IERR)
        CALL MPI_ISEND(VAL(IRP(ST)),NZL_TMP,MPI_DOUBLE_PRECISION,
                                IP-1,5*CNT+5,COMM,SREQ(5*CNT+5),IERR)
        CNT++
    END DO
    IF (NUMPROCS .GT. 1) THEN
        CALL MPI_WAITALL((NUMPROCS-1)*5,SREQ,SSTAT,IERR)
    END IF
    IRP_L(1) = 1
    DO I = 1, NL
        IRP_L(I+1) += IRP(I+1) - IRP(I)
        X_L(I) = X(I)
        B_L(I) = B(I)
        DO J = IRP_L(I), IRP_L(I+1)-1
            ICOL_L(J) = ICOL(J)
            VAL_L(J) = VAL(J)
        END DO
    END DO
END IF
```

※MPI_ISEND，MPI_IRECV で使用する通信バッファの個数(第 2 引数)を変数型 INTEGER8 とした場合、バッファに正常な値が受信されないため、INTEGER4 にキャストしてから通信を行う。

5.3.5 各 rank は rank0 からローカル行列を受信する

(IRP_L,ICOL_L,VAL_L,X_L,B_L)

```
(実装例)
IF (MYID .NE. 0) THEN
    NL_TMP  = INT(NL,4)
    NZL_TMP = INT(NZL,4)
    CALL MPI_IRECV(IRP_L,NL_TMP+1,MPI_INTEGER8,0,(MYID-1)*5+1,
                          COMM,RREQ(1),IERR)
    CALL MPI_IRECV(X_L,NL_TMP,MPI_DOUBLE_PRECISION,0,(MYID-1)*5+2,
                          COMM,RREQ(2),IERR)
    CALL MPI_IRECV(B_L,NL_TMP,MPI_DOUBLE_PRECISION,0,(MYID-1)*5+3,
                          COMM,RREQ(3),IERR)
    CALL MPI_IRECV(ICOL_L,NZL_TMP,MPI_INTEGER8,0,(MYID-1)*5+4,
                          COMM,RREQ(4),IERR)
    CALL MPI_IRECV(VAL_L,NZL_TMP,MPI_DOUBLE_PRECISION,0,
                          (MYID-1)*5+5,COMM,RREQ(5),IERR)
    CALL MPI_WAITALL(5,RREQ,RSTAT,IERR)
    ST = IRP_L(1)
    IRP_L(1) = 1
    DO I = 1, NL
       DIF = IRP_L(I+1) - ST
       ST = IRP_L(I+1)
       IRP_L(I+1) = IRP_L(I) + DIF
    END DO
END IF
```

※MPI_ISEND，MPI_IRECV で使用する通信バッファの個数(第 2 引数)を変数型 INTEGER8 とした場合、バッファに正常な値が受信されないため、INTEGER4 にキャストしてから通信を行う。

6 pOpenATI_DURMV_presetup

6.1 pOpenATI_DURMV_presetup の用途

SpMV に必要な通信情報と各 thread 担当行情報を作成する関数 pOpenATI_DURMV_setup(詳細は 7 章)の前準備を行う関数。MPI 通信情報のセットアップ用配列の作成と pOpenATI_DURMV_setup の出力となっている MPI 通信情報格納配列のサイズの計算を行う。

6.2 引数仕様

CALL pOpenATI_DURMV_presetup(NL,NZL,N,IRP_L,ICOL_L,IATPARAM,RATPARAM,MYID,
NUMPROCS,COMM,DECOMP,Xabclib_COMM_SETUP,LCS,
LCI,INFO)

変数名	型	I/O	説明
NL	INT*8	I	自 rank の次元数
NZL	INT*8	I	自 rank の非零要素数
N	INT*8	I	グローバル行列の次元数
IRP_L(NL+1)	INT*8	I	自 rank の非零要素の各行先頭へのポインタ
ICOL_L(NZL)	INT*8	I	自 rank の非零要素の列番号(グローバルな番号)
IATPARAM(200)	INT	I	INTEGER 型パラメーターリスト
RATPARAM(200)	DOUBLE	I	DOUBLE 型パラメーターリスト
MYID	INT	I	自 rank 番号(0~NUMPROCS-1)
NUMPROCS	INT	I	総 rank 数
COMM	INT	I	MPI コミュニケータ
DECOMP(NUMPROCS+1)	INT*8	I	グローバル行列の分割情報。各 rank の先頭行番号(グローバルな番号)
Xabclib_COMM_SETUP(LCS)	INT*8	O	MPI 通信情報セットアップ用配列(詳細は図 1)
LCS	INT*8	I	Xabclib_COMM_SETUP のサイズ LCS≧6+4*NUMPROCS
LCI	INT*8	O	pOpenATI_DURMV_setup で作成する Xabclib_COMM_INFO のサイズ LCI≧ 7+2*NS_RANKS+2*NR_RANKS+NS_TOT+NZL+NUMTHREADS
INFO	INT	O	エラーコード INFO=0　：正常値 INFO=-I　：i 番目の引数の値が不正

6.3 Xabclib_COMM_SETUP 仕様

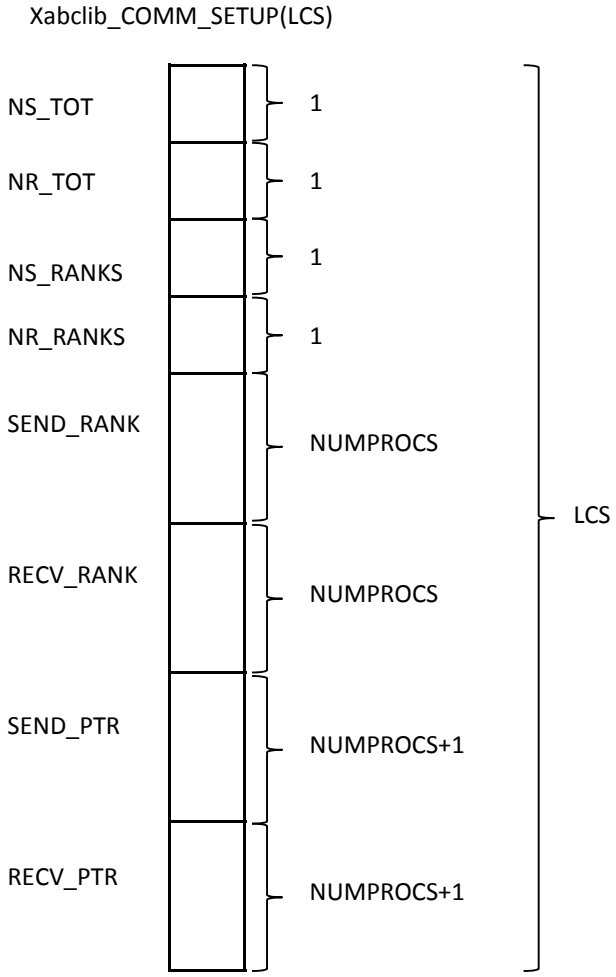


図 1　MPI 通信情報セットアップ用配列 Xabclib_COMM_SETUP の仕様

6.4 処理手順

手順
1. エラーチェック
2. work 領域の設定(MARK,R_COUNT,S_COUNT)
3. MPI 通信情報セットアップ用配列の分割ポインタの設定(Xabclib_COMM_SETUP 用ポインタ)
4. 通信相手の rank 番号取得と通信する要素数のカウント(R_COUNT)
5. 他 rank に通信相手と通信要素数を送信(R_COUNT を ALLTOALL)
6. MPI 通信情報セットアップ用配列の受信側要素作成 (NR_TOT,NR_RANKS,RECV_RANK,RECV_PTR)
7. MPI 通信情報セットアップ用配列の送信側要素作成 (NS_TOT,NS_RANKS,SEND_RANK,SEND_PTR)
8. OpenMP 並列の thread 数を取得する(NUMTHREADS)
9. pOpenATI_DURMV_setup で作成する MPI 通信情報格納配列のサイズを計算する(LCI)

6.4.1 エラーチェック

```
(実装例)
//rank 番号が 0~NUMPROCS-1 になっているか
IF(MYID.LT.0 .OR. MYID.GT.NUMPROCS-1)THEN
    INFO=-1
END IF
CALL MPI_ALLREDUCE(INFO,INFO_G,1,MPI_INTEGER,MPI_SUM,COMM,IERR)
//NL が N よりも大きくないか
IF(NL.GT.N)THEN
    INFO=-5
END IF
CALL MPI_ALLREDUCE(INFO,INFO_G,1,MPI_INTEGER,MPI_SUM,COMM,IERR)
//一つ前の rank が担当している DECOMP が小さくないか
DO I=1,NUMPROCS
    IF(DECOMP(I+1).LT.DECOMP(I))THEN
        INFO=-7
    END IF
END DO
//DECOMP，NL が正しい値か
IF(DECOMP(MYID+2)-DECOMP(MYID+1).NE.NL)THEN
    INFO=-7
END IF
//NL の合計が N となっているか
SIZE=0
DO I=1,NUMPROCS
    SIZE+=DECOMP(I+1)-DECOMP(I)
END DO
IF(SIZE.NE.N)THEN
    INFO=-7
ENDIF
//DECOMP の第一要素が 1 になっているか
IF(DECOMP(1).NE.1)THEN
    INFO=-7
END IF
//DECOMP の最終要素が N+1 になっているか
IF(DECOMP(NUMPROCS+1)-1.NE.N)THEN
    INFO=-7
END IF
CALL MPI_ALLREDUCE(INFO,INFO_G,1,MPI_INTEGER,MPI_SUM,COMM,IERR)
//LCS は十分大きいか
IF(LCS.LT.6+4*NUMPROCS)THEN
    INFO=-13
END IF
CALL MPI_ALLREDUCE(INFO,INFO_G,1,MPI_INTEGER,MPI_SUM,COMM,IERR)
```

6.4.2 work 領域の設定(MARK,R_COUNT,S_COUNT)

```
(実装例)
ALLOCATE(MARK(N),R_COUNT(NUMPROCS),S_COUNT(NUMPROCS))
R_COUNT(1:NUMPROCS)=0
S_COUNT(1:NUMPROCS)=0
MARK(1:N)=0
DO I=DECOMP(MYID+1),DECOMP(MYID+2)-1
   MARK(I)=1
END DO
```

6.4.3 MPI 通信情報セットアップ用配列の分割ポインタの設定

(Xabclib_COMM_SETUP 用ポインタ)

```
(実装例)
IP_NS_TOT＝1
IP_NR_TOT= IP_NS_TOT+1
IP_NS_RANKS=IP_NR_TOT+1
IP_NR_RANKS=IP_NS_RANKS+1
IP_SEND_RANK=IP_NR_RANKS+1
IP_RECV_RANK=IP_SEND_RANK+NUMPROCS
IP_SEND_PTR=IP_RECV_RANK+NUMPROCS
IP_RECV_PTR=IP_SEND_PTR+NUMPROCS+1
※以降、設定したポインタにより分割された Xabclib_COMM_SETUP は以下のように表す
  NS_TOT=Xabclib_COMM_SETUP(IP_NS_TOT)
  NR_TOT=Xabclib_COMM_SETUP(IP_NR_TOT)
  NS_RANKS=Xabclib_COMM_SETUP(IP_NS_RANKS)
  NR_RANKS=Xabclib_COMM_SETUP(IP_NR_RANKS)
  SEND_RANK=Xabclib_COMM_SETUP(IP_SEND_RANK)
  RECV_RANK=Xabclib_COMM_SETUP(IP_RECV_RANK)
  SEND_PTR=Xabclib_COMM_SETUP(IP_SEND_PTR)
  RECV_PTR= Xabclib_COMM_SETUP(IP_RECV_PTR)
```

6.4.4 通信相手の rank 番号取得と通信する要素数のカウント(R_COUNT)

```
(実装例)
DO I=1,NL
   DO J=IRP_L(I),IRP_L(I+1)-1
      JP=ICOL_L(J)
      IF (MARK(JP).EQ.0) THEN
         MARK(JP)=1
      END IF
   END DO
END DO
DO I=1,NUMPROCS
   DO J=DECOMP(I),DECOMP(I+1)-1
      R_COUNT(I)+=MARK(J)
   END DO
END DO
R_COUNT(MYID+1)=0
```

6.4.5 他 rank に通信相手と通信要素数を送信(R_COUNT を ALLTOALL)

```
(実装例)
CALL MPI_ALLTOALL(R_COUNT,1,MPI_INTEGER8,S_COUNT,1,MPI_INTEGER8,
                  COMM,IERR)
```

6.4.6 MPI 通信情報セットアップ用配列の受信側要素作成

(NR_TOT,NR_RANKS,RECV_RANK,RECV_PTR)

```
(実装例)
RECV_PTR(1)=NL+1
NR_RANKS=0
NR_TOT=0
DO I=1,NUMPROCS
   IF (R_COUNT(I).NE.0) THEN
      NR_RANKS++
      RECV_PTR(NR_RANKS+1)=RECV_PTR(NR_RANKS)+R_COUNT(I)
      RECV_RANK(NR_RANKS)=I-1
      NR_TOT+=R_COUNT(I)
   END IF
END DO
```

6.4.7 MPI 通信情報セットアップ用配列の送信側要素作成

(NS_TOT,NS_RANKS,SEND_RANK,SEND_PTR)

```
(実装例)
SEND_PTR(1)=1
NS_RANKS=0
NS_TOT=0
DO I=1,NUMPROCS
   IF (S_COUNT(I).NE.0) THEN
      NS_RANKS++
      SEND_PTR(NS_RANKS+1)=SEND_PTR(NS_RANKS)+S_COUNT(I)
      SEND_RANK(NS_RANKS)=I-1
      NS_TOT+=S_COUNT(I)
   END IF
END DO
```

6.4.8 OpenMP 並列の thread 数を取得する(NUMTHREADS)

```
(実装例)
NUMTHREADS=IATPARAM(3)
```

6.4.9 pOpenATI_DURMV_setup で作成する MPI 通信情報格納配のサイズを計算する(LCI)

(実装例)
LCI=7+2*NS_RANKS+2*NR_RANKS+NS_TOT+NZL+NUMTHREADS

7 pOpenATI_DURMV_setup

7.1 pOpenATI_DURMV_setup の用途

pOpenATI_DURMV_setup で作成した MPI 通信情報セットアップ用配列を利用し、SpMV に必要な通信情報と各 thread 担当行情報を作成する関数。関数を呼び出す前に MPI 通信情報格納配列の領域確保を行う必要がある。

7.2 引数仕様

CALL pOpenATI_DURMV_setup(NL,NZL,N,IRP_L,ICOL_L,IATPARAM,RATPARAM,MYID,
NUMPROCS,COMM,DECOMP,Xabclib_COMM_SETUP,LCS,
Xabclib_COMM_INFO,LCI,INFO)

変数名	型	I/O	説明
NL	INT*8	I	自 rank の次元数
NZL	INT*8	I	自 rank の非零要素数
N	INT*8	I	グローバル行列の次元数
IRP_L(NL+1)	INT*8	I	自 rank の非零要素の各行先頭へのポインタ
ICOL_L(NZL)	INT*8	I	自 rank の非零要素の列番号(グローバルな番号)
IATPARAM(200)	INT	I	INTEGER 型パラメーターリスト
RATPARAM(200)	DOUBLE	I	DOUBLE 型パラメーターリスト
MYID	INT	I	自 rank 番号(0~NUMPROCS-1)
NUMPROCS	INT	I	総 rank 数
COMM	INT	I	MPI コミュニケータ
DECOMP(NUMPROCS+1)	INT*8	I	グローバル行列の分割情報。各 rank の先頭行番号(グローバルな番号)
Xabclib_COMM_SETUP(LCS)	INT*8	I	MPI 通信情報セットアップ用配列 (pOpenATI_DURMV_presetup の出力をそのまま使用)
LCS	INT*8	I	Xabclib_COMM_SETUP のサイズ LCS≧6+4*NUMPROCS
Xabclib_COMM_INFO(LCI)	INT*8	O	MPI 通信情報格納配列(詳細は図 2)
LCI	INT*8	I	Xabclib_COMM_INFO のサイズ LCI≧ 7+2*NS_RANKS+2*NR_RANKS+NS_TOT+NZL+NUMTHREADS (pOpenATI_DURMV_presetup の出力をそのまま使用)
INFO	INT	O	エラーコード INFO=0　：正常値 INFO=-I　：i 番目の引数の値が不正

7.3 Xabclib_COMM_INFO 仕様

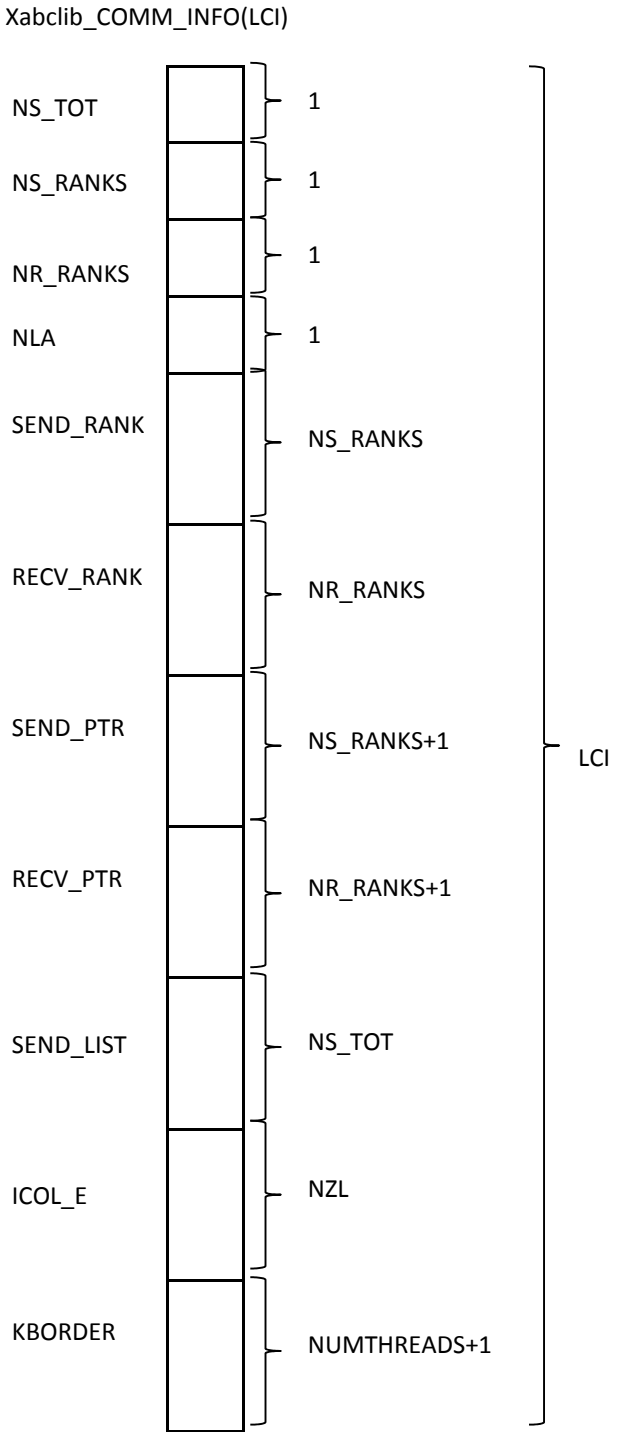


図 2　MPI 通信情報格納配列 Xabclib_COMM_INFO の仕様

7.4 処理手順

手順
1. pOpenATI_DURMV_setup 前に MPI 通信情報格納配列の領域確保 (Xabclib_COMM_INFO)
2. エラーチェック
3. MPI 通信情報格納配列の分割情報の設定(Xabclib_COMM_INFO 用ポインタ)
4. work 領域&MPI 通信用配列の設定 (MARK,ROW_NUM,IORD,RECV_LIST,SSTAT.SREQ,RSTAT,RREQ)
5. ICOL_E の作成と要求する行番号のパック(ICOL_E,RECV_LIST)
6. 要求する行番号の LIST を他 rank に送信(RECV_LIST を ISEND)
7. 他 rank から要求行番号の受信(SEND_LIST に IRECV)
8. 受信した行番号をグローバルな番号からローカルなものへ変更する
9. OpenMP を利用したローカル行列の分割情報作成(KBORDER)

7.4.1 pOpenATI_DURMV_setup 前に MPI 通信情報格納配列の領域確保 (Xabclib_COMM_INFO)

(実装例)
```
ALLOCATE(Xabclib_COMM_INFO(LCI))
```

7.4.2 エラーチェック

(実装例)
```
//LCI は十分大きいか
IF(LCI.LT.7+2*NS_RANKS+2*NR_RANKS+NS_TOT+NZL+NUMTHREADS)THEN
    INFO=-15
END IF
INFO_G = 0
//NL が 2G 以下か
IF (NL .GT. 2000000000) THEN
    INFO = -1
END IF
CALL MPI_ALLREDUCE(INFO,INFO_G,1,MPI_INTEGER,MPI_SUM,COMM,IERR)
//NZL が 2G 以下か
IF (NZL .GT. 2000000000) THEN
    INFO = -2
END IF
CALL MPI_ALLREDUCE(INFO,INFO_G,1,MPI_INTEGER,MPI_SUM,COMM,IERR)
```

7.4.3 MPI 通信情報格納配列の分割情報の設定(Xabclib_COMM_INFO 用ポインタ)

(実装例)
IP_NS_TOT＝1
IP_NS_RANKS=IP_NS_TOT+1
IP_NR_RANKS=IP_NS_RANKS+1
IP_NLA=IP_NR_RANKS+1
IP_SEND_RANK=IP_NLA+1
IP_RECV_RANK=IP_SEND_RANK+NS_RANKS
IP_SEND_PTR=IP_RECV_RANK+NR_RANKS
IP_RECV_PTR=IP_SEND_PTR+NS_RANKS+1
IP_SEND_LIST=IP_RECV_PTR+NR_RANKS+1
IP_ICOL_E=IP_SEND_LIST+NS_TOT
IP_KBORDER=IP_ICOL_E+NZL
※以降、設定したポインタにより分割された Xabclib_COMM_INFO は以下のように表す
NS_TOT=Xabclib_COMM_INFO(NS_TOT)
NS_RANKS=Xabclib_COMM_INFO(IP_NS_RANKS)
NR_RANKS=Xabclib_COMM_INFO(IP_NR_RANKS)
NLA=Xabclib_COMM_INFO(IP_NLA)
SEND_RANK=Xabclib_COMM_INFO(IP_SEND_RANK)
RECV_RANK=Xabclib_COMM_INFO(IP_RECV_RANK)
SEND_PTR=Xabclib_COMM_INFO(IP_SEND_PTR)
RECV_PTR=Xabclib_COMM_INFO(IP_RECV_PTR)
SEND_LIST=Xabclib_COMM_INFO(IP_SEND_LIST)
ICOL_E=Xabclib_COMM_INFO(IP_ICOL_E)
KBODER=Xabclib_COMM_INFO(IP_KBODER)

7.4.4 work 領域&MPI 通信用配列の設定

(MARK,ROW_NUM,IORD,RECV_LIST,SSTAT,SREQ,RSTAT,RREQ)

(実装例)
```
ALLOCATE(MARK(N),ROW_NUM(NR_TOT),IORD(NR_TOT),RECV_LIST(NR_TOT),
              SSTAT(MPI_STATUS_SIZE,NR_RANKS),SREQ(NR_RANKS),
              RSTAT(MPI_STATUS_SIZE,NS_RANKS),RREQ(NS_RANKS))
MARK(1;N)=0
IP=DECOMP(MYID+1)-1
DO I=DECOMP(MYID+1),DECOMP(MYID+2)-1
   MARK(I)=I-IP
END DO
```

7.4.5 ICOL_E の作成と要求する行番号のパック(ICOL_E,RECV_LIST)

```
(実装例)
K=1
DO I=1,NL
    DO J=IRP_L(I),IRP_L(I+1)-1
        JP=ICOL_L(J)
        IF (MARK(JP).EQ.0) THEN
            MARK(JP)=1
            ROW_NUM(K)=JP
            K++
        END IF
    END DO
END DO
CALL pOPENATI_QSORT(IORD,NR_TOT,ROW_NUM)
K=NL+1
DO I=1,NR_TOT
    RECV_LIST(I)=ROW_NUM(IORD(I))
    MARK(RECV_LIST(I))=K
    K++
END DO
DO I=1,NL
    DO J=IRP_L(I),IRP_L(I+1)-1
        JP=ICOL_L(J)
        ICOL_E(J)=MARK(JP)
    END DO
END DO
```

7.4.6 要求する行番号のリストを他 rank に送信(RECV_LIST を ISEND)

```
(実装例)
DO I=1,NR_RANKS
    RANK=RECV_RANK(I)
    SIZE=INT(RECV_PTR(I+1)-RECV_PTR(I),4)
    CALL MPI_ISEND(RECV_LIST(RECV_PTR(I)-NL),SIZE,MPI_INTEGER8,
                                RANK,RANK+1,COMM,SREQ(I),IERR)
END DO
```

※MPI_ISEND，MPI_IRECV で使用する通信バッファの個数(第 2 引数)を変数型 INTEGER8 とした場合、バッファに正常な値が受信されないため、INTEGER4 にキャストしてから通信を行う。

7.4.7 他 rank から要求行番号の受信(SEND_LIST に IRECV)

```
(実装例)
DO I=1,NS_RANKS
   RANK=SEND_RANK(I)
   SIZE=INT(SEND_PTR(I+1)-SEND_PTR(I),4)
   CALL MPI_IRECV(SEND_LIST(SEND_PTR(I)),SIZE,MPI_INTEGER8,
                          RANK,MYID+1,COMM,RREQ(I),IERR)
END DO

IF (NR_RANKS.NE.0) THEN
   RANK=INT(NR_RANKS,4)
   CALL MPI_WAITALL(RANK,SREQ,SSTAT,IERR)
END IF
IF (NS_RANKS.NE,0) THEN
   RANK=INT(NS_RANKS,4)
   CALL MPI_WAITALL(RANK,RREQ,RSTAT,IERR)
END IF
```

※MPI_ISEND，MPI_IRECV で使用する通信バッファの個数(第 2 引数)を変数型 INTEGER8 とした場合、バッファに正常な値が受信されないため、INTEGER4 にキャストしてから通信を行う。MPI_WAITALL で使用する同期の必要な通信回数(第 1 引数)を変数型 INTEGER8 とした場合、INTEGER4 にキャストしてから通信を行う。

7.4.8 受信した行番号をグローバルな番号からローカルなものへ変更する

```
(実装例)
DO I=1,NS_TOT
   SEND_LIST(I)=SEND_LIST(I)-DECOMP(MYID+1)+1
END DO
```

7.4.9 OpenMP を利用したローカル行列の分割情報作成(KBORDER)

```
(実装例)
CALL pOpenATI_DURMV_SetupOMP(NL,NZL,IRP_L,NUMTHREADS,KBORDER)
```

8 pOpenATI_DURMV

8.1 pOpenATI_DURMV の用途

OpenMP と MPI でハイブリット並列化された SpMV を実行する関数。計算を行う前に MPI 通信情報格納配列を利用し，入力ベクトルの送受信をする。

8.2 引数仕様

CALL pOpenATI_DURMV(NL,NLA,NZL,IRP_L,VAL_L,XM_L,Y_L,IATPARAM,RATPARAM,
MYID,NUMPROCS,COMM,Xabclib_COMM_INFO,LCI,INFO)

変数名	型	I/O	説明
NL	INT*8	I	自 rank の次元数
NLA	INT*8	I	自 rank の行数と他 rank から受信する X_L の要素数の和
NZL	INT*8	I	自 rank の非零要素数
IRP_L(NL+1)	INT*8	I	自 rank の非零要素の各行先頭へのポインタ
VAL_L(NZL)	DOUBLE	I	自 rank の非零要素の値
XM_L(NLA)	DOUBLE	I/O	1～NL に自 rank のベクトルの値を格納して入力 関数内で NL+1～NLA に” pOpenATI_DURMV”で他 rank から受信した値が格納される
Y_L(NL)	DOUBLE	O	自 rank の解ベクトルの値
IATPARAM(200)	INT	I	INTEGER 型パラメーターリスト
RATPARAM(200)	DOUBLE	I	DOUBLE 型パラメーターリスト
MYID	INT	I	自 rank 番号(0~NUMPROCS-1)
NUMPROCS	INT	I	総 rank 数
COMM	INT	I	MPI コミュニケータ
Xabclib_COMM_INFO(LCI)	INT*8	I	MPI 通信情報格納配列
LCI	INT*8	I	Xabclib_COMM_INFO のサイズ LCI≧ 7+2*NS_RANKS+2*NR_RANKS+NS_TOT+NZL+NUMTHREADS
INFO	INT	O	エラーコード INFO=0 ：正常値 INFO=-I ：i 番目の引数の値が不正 INFO=1100 ：MPI エラー

8.3 処理手順

手順
1. work 領域&MPI 通信用配列の設定(SEND_X,SSTAT,SREQ,RSTAT,RREQ)
2. 送信する X をパックし他 rank へ送信(SEND_X を作成し ISEND)
3. 参照する X の受信(XM_L に IRECV)
4. ハイブリッド並列版 SpMV

8.3.1 work 領域&MPI 通信用配列の設定(SEND_X,SSTAT,SREQ,RSTAT,RREQ)

```
(実装例)
ALLOCATE(SEND_X(NS_TOT),
                SSTAT(MPI_STATUS_SIZE,NS_RANKS),SREQ(NS_RANKS),
                RSTAT(MPI_STATUS_SIZE,NR_RANKS),RREQ(NR_RANKS))
```

8.3.2 送信する X をパックし他 rank へ送信(SEND_X を作成し ISEND)

```
(実装例)
DO I=1,NS_RANKS
    DO J=SEND_PTR(I),SEND_PTR(I+1)-1
        JP=SEND_LIST(J)
        SEND_X(J)=XM_L(JP)
    END DO
END DO
DO I=1,NS_RANKS
    RANK=SEND_RANK(I)
    SIZE=INT(SEND_PTR(I+1)-SEND_PTR(I),4)
    CALL
MPI_ISEND(SEND_X(SEND_PTR(I)),SIZE,MPI_DOUBLE_PRECISION,RANK,
                            RANK+1,COMM,SREQ(I),IERR)
END DO
```

※MPI_ISEND，MPI_IRECV で使用する通信バッファの個数(第 2 引数)を変数型 INTEGER8 とした場合、バッファに正常な値が受信されないため、INTEGER4 にキャストしてから通信を行う。

8.3.3 参照する X の受信(X_L に IRECV)

```
(実装例)
DO I=1,NR_RANKS
    RANK=RECV_RANK(I)
    SIZE=INT(RECV_PTR(I+1)-RECV_PTR(I),4)
    CALL MPI_IRECV(XM_L(RECV_PTR(I)),SIZE,MPI_DOUBLE_PRECISION,RANK,
                        MYID+1,COMM,RREQ(I),IERR)
END DO
IF (NR_RANKS.NE.0) THEN
    RANK=INT(NR_RANKS,4)
    CALL MPI_WAITALL(RANK,RREQ,RSTAT,IERR)
END IF
IF (NS_RANKS.NE.0) THEN
    RANK=(NS_RANKS,4)
    CALL MPI_WAITALL(RANK,SREQ,SSTAT,IERR)
END IF
```

※MPI_ISEND，MPI_IRECV で使用する通信バッファの個数(第 2 引数)を変数型 INTEGER8 とした場合、バッファに正常な値が受信されないため、INTEGER4 にキャストしてから通信を行う。MPI_WAITALL で使用する同期の必要な通信回数(第 1 引数)を変数型 INTEGER8 とした場合、INTEGER4 にキャストしてから通信を行う。

8.3.4 ハイブリッド並列版 SpMV

```
(実装例)
!$omp parallel do private(S,J,I,JP)
DO K=1,NUMTHREADS
    DO I=KBORDER(K),KBORDER(K+1)-1
        S=0.0D0
        DO J=IRP_L(I),IRP_L(I+1)-1
            JP=ICOL_E(J)
            S+=VAL_L(J)*XM_L(JP)
        END DO
        Y_L(I)=S
    END DO
END DO
!$omp end parallel do
```

9 pXabclib_BICGSTAB

9.1 pXabclib_BICGSTAB の用途

OpenMP と MPI でハイブリット並列化された BiCGStab 法。大規模な非対称疎行列を係数とする連立一次方程式 Ax=b を反復解法で解く方式を取る。

9.2 pXabclib_BICGSTAB アルゴリズム

(1) $x_0 = initial\ guess$
(2) pOpenATI_DURMV($r_0 = Ax_0$)
(3) $r_0 = b - r_0$
(4) $r_0^* = M^{-1}r_0$, solve $M\hat{r}_0 = r_0, \rho_0 = \langle r_0^*, \hat{r}_0 \rangle, \beta_0^G = 0, p_0 = v_0 = 0$
(5) MPI_Allreduce($\rho_0 \rightarrow \rho_0^G$)
(6) $k = 0,1,2,\cdots do$
(7) $p_k = \hat{r}_k + \beta_k^G p_k$
(8) pOpenATI_DURMV($\hat{p}_k = Ap_k$)
(9) solve $Mv_k = \hat{p}_k$
(10) $\gamma_k = \langle r_0^*, v_k \rangle$
(11) MPI_Allreduce($\gamma_k \rightarrow \gamma_k^G$)
(12) $\alpha_k^G = \rho_0^G / \gamma_k^G$
(13) $s_k = r_k - \alpha_k^G \hat{p}_k$
(14) $\hat{s}_k = \hat{r}_k - \alpha_k^G v_k$
(15) check conv.?if "$\|\hat{s}_k\|$" small enough then $x_{k+1} = x_k + \alpha_k^G p_k$; exit
(16) pOpenATI_DURMV($t_k = A\hat{s}_k$)
(17) $td = \langle t_k, s_k \rangle$
(18) MPI_Allreduce($td \rightarrow td^G$)
(19) $tmpt = \langle t_k, t_k \rangle$
(20) MPI_Allreduce($tmpt \rightarrow tmpt^G$)
(21) $\zeta_k^G = td^G / tmpt^G$
(22) $x_{k+1} = x_k + \alpha_k^G p_k + \zeta_k^G \hat{s}_k$
(23) $r_{k+1} = s_k - \zeta_k^G t_k$
(24) $rnorm_k = \|r_{k+1}\|$
(25) MPI_Allreduce($rnorm_k \rightarrow rnorm_k^G$)
(26) check conv.?if "$rnorm_k^G$" small enough exit
(27) solve $M\hat{r}_{k+1} = r_{k+1}$
(28) $\rho_k = \langle r_0^*, \hat{r}_{k+1} \rangle$
(29) MPI_Allreduce($\rho_k \rightarrow \rho_k^G$)
(30) $p_{k+1} = p_k - \zeta_k^G v_k$
(31) $\beta_{k+1}^G = (\rho_k^G / \rho_0^G) * (\alpha_k^G / \zeta_k^G)$
(32) $\rho_0^G = \rho_k^G$
(33) $enddo$

※全 rank で共有している変数、配列は上付き G で表示する

9.3 pXabclib_BICGSTAB を呼び出すまでの流れ

① pXabclib_BICGSTAB のパラメーターをセット(pOpenATI_init)
※既に各 rank がローカル行列を保持している場合は⑤へ
② 行列情報の作成(N,NZ,IRP(N+1),ICOL(NZ),VAL(NZ),X(N),B(N))
③ 分割情報の作成(DECOMP(NUMPROCS+1))(ユーザーが作成)
④ ローカルな行列情報の作成(NL,NZL,IRP_L(NL+1),ICOL_L(NZL),VAL_L(NZL),X_L(NL),B_L(NL))(ユーザーが作成)
⑤ MPI 通信情報セットアップ用配列の確保(Xabclib_COMM_SETUP(LCS))
LCS=6+4*NUMPROCS
⑥ pOpenATI_DURMV_presetup 呼び出し
⑦ MPI 通信情報格納配列の確保(Xabclib_COMM_INFO(LCI))
LCI=7+2*NS_RANKS+2*NR_RANKS+NS_TOT+NZL+NUMTHREADS
⑧ pOpenATI_DURMV_setup 呼び出し
⑨ pXabclib_BICGSTAB 呼び出し
⑩ 結果の収集

9.4 pXabclib_BICGSTAB 引数仕様

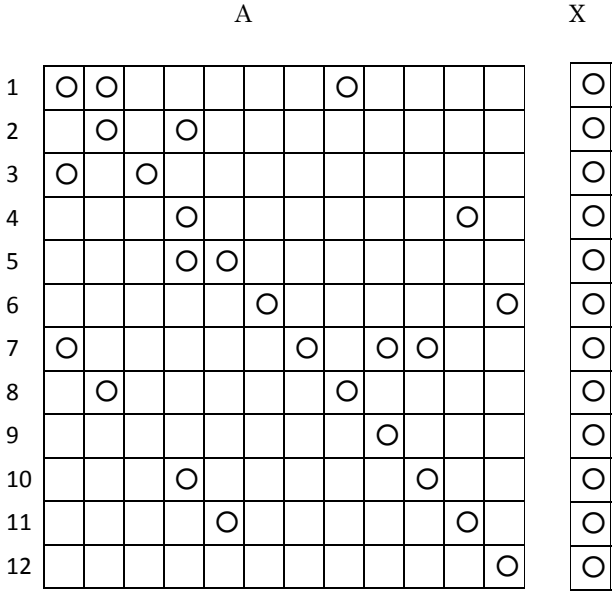
```
CALL pXabclib_BICGSTAB(NL,NZL,N,NZ,IRP_L,ICOL_L,VAL_L,X_L,B_L,PRECOND,IWPC,
                       IATPARAM,RATPARAM,MYID,NUMPROCS,COMM,DECOMP,
                       Xabclib_COMM_INFO,LCI,WORK,LDWORK,INFO)
```

変数名	型	I/O	説明
NL	INT*8	I	自 rank の次元数
NZL	INT*8	I	自 rank の非零要素数
N	INT*8	I	グローバル行列の次元数
NZ	INT*8	I	グローバル行列の非零要素数
IRP_L(NL+1)	INT*8	I	自 rank の非零要素の各行先頭へのポインタ
ICOL_L(NZL)	INT*8	I	自 rank の非零要素の列番号(グローバルな番号)
VAL_L(NZL)	DOUBLE	I	自 rank の非零要素の値
X_L(NL)	DOUBLE	I/O	自 rank のベクトルの値
B_L(NL)	DOUBLE	I	自 rank の右辺ベクトル
PRECOND(IWPC)	INT*8	I	前処理行列(対角スケーリング)
IWPC	INT*8	I	PRECOND のサイズ(IWPC≧NL)
IATPARAM(200)	INT	I/O	INTEGER 型パラメーターリスト
RATPARAM(200)	DOUBLE	I/O	DOUBLE 型パラメーターリスト
MYID	INT	I	自 rank 番号(0~NUMPROCS-1)
NUMPROCS	INT	I	総 rank 数
COMM	INT	I	MPI コミュニケータ
DECOMP(NUMPROCS+1)	INT*8	I	グローバル行列の分割情報。各 rank の先頭行番号(グローバルな番号)

Xabclib_COMM_INFO(LCI)	INT*8	I	MPI 通信情報格納配列
LCI	INT*8	I	Xabclib_COMM_INFO のサイズ LCI≧ 7+2*NS_RANKS+2*NR_RANKS+NS_TOT+NZL+NUMTHREADS
WORK(LDWORK)	DOUBLE	WORK	作業領域
LDWORK	INT*8	I	WORK のサイズ LDWORK≧8NL+2NLA
INFO	INT	O	エラーコード INFO=0　：正常値 INFO=-i　：i 番目の引数の値が不正 INFO=100　：前処理行列作成部分での異常検知 INFO=200　：プログラム breakdown INFO=400　：最大演算時間オーバー INFO=500　：最大反復回数オーバー INFO=1100：MPI エラー

付録　行列，ベクトルの分割方式の図

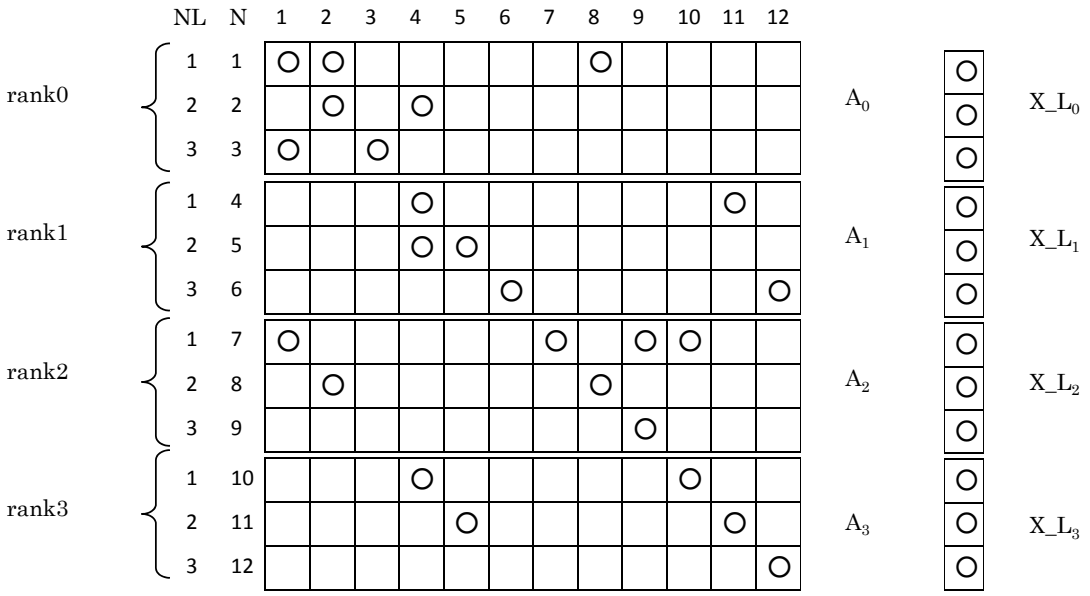
N	：行列の次元数
NZ	：行列の非ゼロ数
IRP(N+1)	：非ゼロ要素の各行先頭へのポインタ
ICOL(NZ)	：非ゼロ要素の列番号
X(N)	：ベクトルの値



※図中の”○”は非ゼロ要素を表す

N=12、NZ=25
IRP={1,4,6,8,10,12,14,18,20,21,23,25,26}
ICOL={1,2,8｜2,4｜1,3｜4,11｜4,5｜6,12｜1,7,9,10｜2,8｜9｜4,10｜5,11｜12}

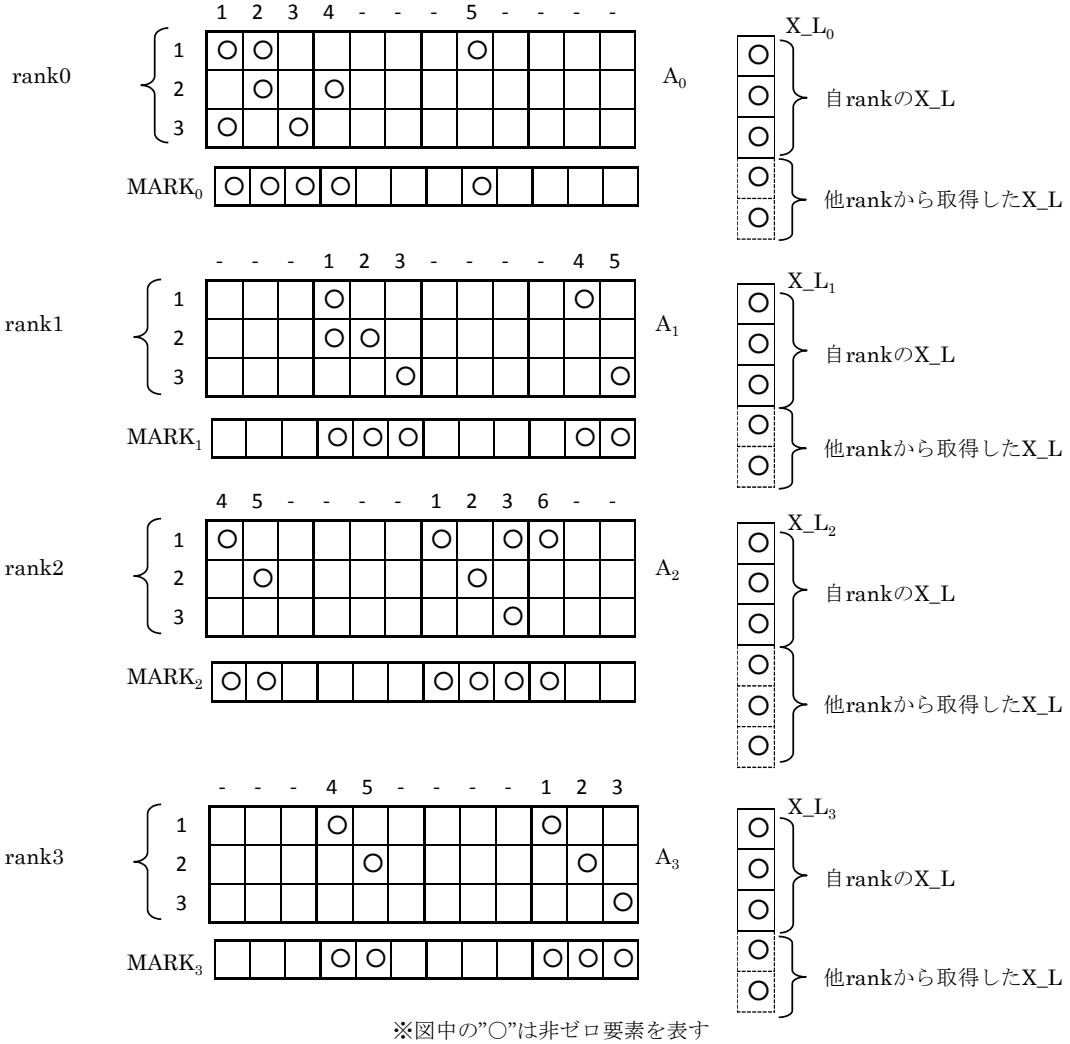
NL　　：自rankの行列の次元数
NZL　　：自rankの行列の非ゼロ数
NLA　　：自rankの行数と他rankから受信するX_Lの和
IRP_L(NL+1)　　：自rankの非ゼロ要素の各行先頭へのポインタ
ICOL_L(NZL)　　：自rankの非ゼロ要素の列番号(列番号はグローバルな値)
X_L(NLA)　　：１～NLに自rankのベクトルの値を格納
DECOMP(NUMPROCS+1)　：分割情報。各rankの先頭ポインタ



※図中の"○"は非ゼロ要素を表す

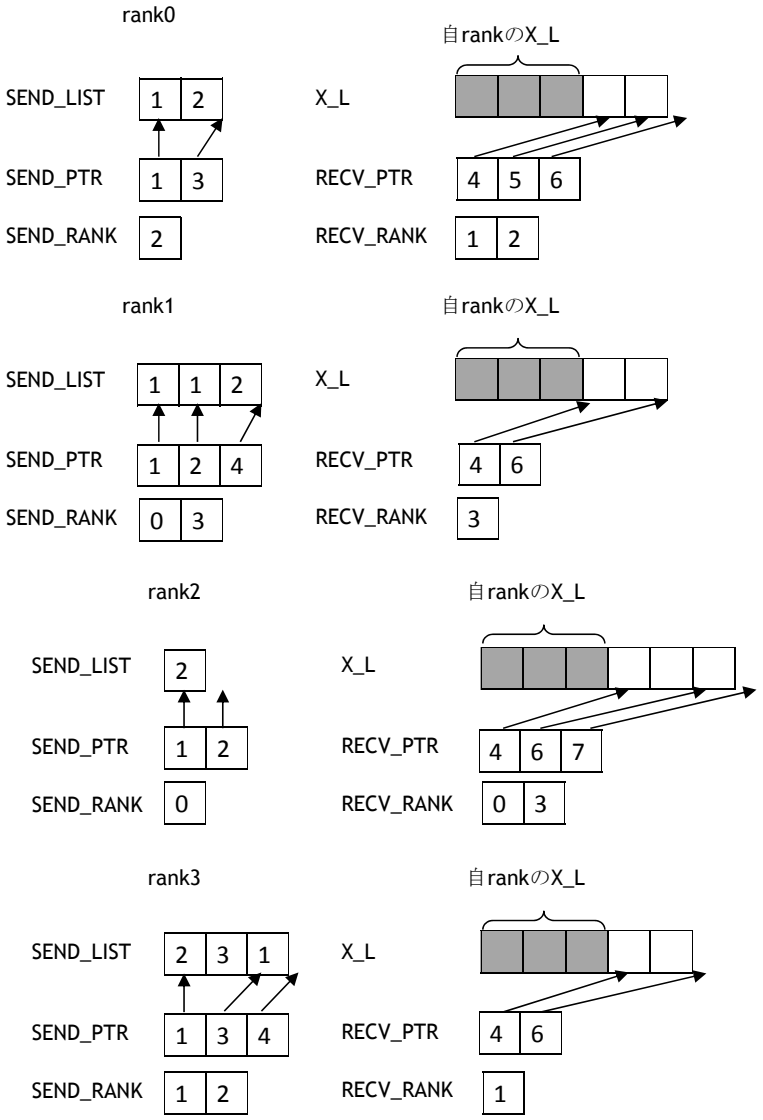
DECOMP={1,4,7,10,13}
NL_0=3、NZL_0=7
IRP_L_0={1,4,6,8}
$ICOL\_L_0$={1,2,8｜2,4｜1,3}
NL_1=3、NZL_1=6
IRP_L_1={1,3,5,7}
$ICOL\_L_1$={4,11｜4,5｜6,12}
NL_2=3、NZL_2=7
IRP_L_2={1,5,7,8}
$ICOL\_L_2$={1,7,9,10｜2,8｜9}
NL_3=3、NZL_3=5
IRP_L_3={1,3,5,6}
$ICOL\_L_3$={4,10｜5,11｜12}

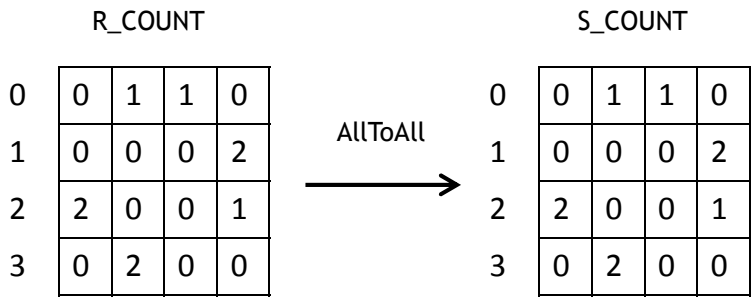
ICOL_E(NZL)：ICOL_Lの列番号をX_L用にリナンバリングしたリスト
MARK(N)：非ゼロ要素に対応した列にマーク

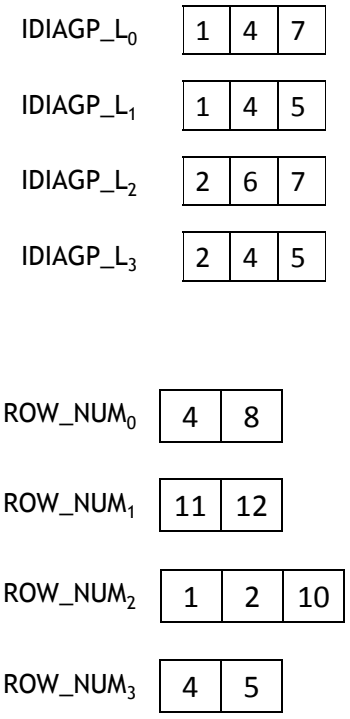


※図中の”○”は非ゼロ要素を表す

$ICOL\_L_0$={1,2,8｜2,4｜1,3}
$MARK_0$={1,1,1,1,0,0,0,1,0,0,0,0}
$ICOL\_E_0$={1,2,5｜2,4｜1,3}
$ICOL\_L_1$={4,11｜4,5｜6,12}
$MARK_1$={0,0,0,1,1,1,0,0,0,0,1,1}
$ICOL\_E_1$={1,4｜1,2｜3,5}
$ICOL\_L_2$={1,7,9,10｜2,8｜9}
$MARK_2$={1,1,0,0,0,0,1,1,1,1,0,0}
$ICOL\_E_2$={4,1,3,6｜5,2｜3}
$ICOL\_L_3$={4,10｜5,11｜12}
$MARK_3$={0,0,0,1,1,0,0,0,0,1,1,1}
$ICOL\_E_3$={4,1｜5,2｜3}


rank0
自rankのX_L
SEND_LIST 1 2
SEND_PTR 1 3
SEND_RANK 2
X_L
RECV_PTR 4 5 6
RECV_RANK 1 2
rank1
自rankのX_L
SEND_LIST 1 1 2
SEND_PTR 1 2 4
SEND_RANK 0 3
X_L
RECV_PTR 4 6
RECV_RANK 3
rank2
自rankのX_L
SEND_LIST 2
SEND_PTR 1 2
SEND_RANK 0
X_L
RECV_PTR 4 6 7
RECV_RANK 0 3
rank3
自rankのX_L
SEND_LIST 2 3 1
SEND_PTR 1 3 4
SEND_RANK 1 2
X_L
RECV_PTR 4 6
RECV_RANK 1


R_COUNT
0 0 1 1 0
1 0 0 0 2
2 2 0 0 1
3 0 2 0 0
AllToAll
S_COUNT
0 0 1 1 0
1 0 0 0 2
2 2 0 0 1
3 0 2 0 0


IDIAGP_L0
1 4 7
IDIAGP_L1
1 4 5
IDIAGP_L2
2 6 7
IDIAGP_L3
2 4 5
ROW_NUM0
4 8
ROW_NUM1
11 12
ROW_NUM2
1 2 10
ROW_NUM3
4 5