Sphere

Skeleton for PHysical and Engineering REsearch

Ver 2.0.0

FORTRAM インターフェイスマニュアル

2010 年 8 月 01 日

目次

１．　概要

SPHERE並列クラスライブラリに実装されているノード間通信メソッド群をFortranサブルーチンから呼出すことができます。

Fortranサブルーチンの名前はSPHERE並列クラスライブラリのメソッド名に1対1に対応しています。引数も基本的に1対1に対応していますので機能詳細は並列ライブラリのマニュアルを参照してください。

Fortranサブルーチンに特化した引数については以下に説明をします。

説明内の定数を用いる為には以下のFortranファイルをインクルードしてください。

インクルードファイル

[SPHERE インストールフォルダ]/include/sphparafort.fin

(1) サブルーチンの引数：integer ierr

サブルーチンの実行が成功したか失敗したかをFortranサブルーチンでは引数integer ierrに返します。

ierr =　　1：成功

-1：失敗

No	定義IERR定数(=定数値)	サブルーチンの成否
1	SPH_SUCCESS = 1	成功
2	SPH_ERROR = -1	失敗

a) 定数を使用する場合は、"sphparafort.fin"をインクルードしてください。

(2) データ型：DATA_TYPE

integer, real(real4 or real8)のデータ型を意味します。

以下のデータ型定数又は対応する定数値を使用してください。

No	定義データ型定数(=定数値)	Frotranデータ型	Cデータ型	サイズ
1	SPH_CHAR = 1	INTEGER (KIND=1)	char	1
2	SPH_UNSIGNED_CHAR = 2	INTEGER (KIND=1)	unsigned char	1
3	SPH_SHORT = 4	INTEGER (KIND=2)	short	2
4	SPH_UNSIGNED_SHORT = 5	INTEGER (KIND=2)	unsigned short	2
5	SPH_INTEGER = 6	INTEGER (KIND=4)	int	4
6	SPH_UNSIGNED = 7	INTEGER (KIND=4)	unsigned int	4

No	定義データ型定数(=定数値)	Frotranデータ型	Cデータ型	サイズ
7	SPH_LONG = 8	INTEGER (KIND=8)	long	8
8	SPH_UNSIGNED_LONG = 9	INTEGER (KIND=8)	unsigned long	8
9	SPH_LONG_LONG = 13	INTEGER (KIND=8)	long long	8
10	SPH_REAL4 = 10	REAL (KIND=4)	float	4
11	SPH_FLOAT = 10	REAL (KIND=4)	float	4
12	SPH_REAL8 = 11	REAL (KIND=8)	double	8
13	SPH_DOUBLE = 11	REAL (KIND=8)	double	8

a) 32ビットマシン環境では"long", "unsigned long"のC言語でのデータサイズは4バイトとなります。

b) 定数を使用する場合は、"sphparafort.fin"をインクルードしてください。

(3) 演算の種類：OP_TYPE

並列演算通信であるレデュース、オールレデュースを行う際の演算の種類をFortran側で指定する定数です。

No	定義演算定数(=定数値)	対応するMPI定数	演算の説明
1	SPH_MAX = 100	MPI_MAX	最大値
2	SPH_MIN = 101	MPI_MIN	最小値
3	SPH_SUM = 102	MPI_SUM	和
4	SPH_PROD = 103	MPI_PROD	積
5	SPH_LAND = 104	MPI_LAND	論理積
6	SPH_BAND = 105	MPI_BAND	ビット演算の積
7	SPH_LOR = 106	MPI_LOR	論理和
8	SPH_BOR = 107	MPI_BOR	ビット演算の和
9	SPH_LXOR = 108	MPI_LXOR	排他的論理輪
10	SPH_BXOR = 109	MPI_BXOR	ビット演算の排他的論理輪

b) 定数を使用する場合は、"sphparafort.fin"をインクルードしてください。

(4) 方向定義定数

"sph_getVoxelHeadIndex"、"sph_getCommIDNum"サブルーチンを使用する際のI,J,K方向、プラス、マイナス方向を指定する定数です。

No	定義方向定数(=定数値)	方向の説明
1	X_DIR = 0	I 方向
2	Y_DIR = 1	J 方向
3	Z_DIR = 2	K 方向
4	MINUS_DIR = -1	マイナス方向
5	PLUS_DIR = 1	プラス方向

a) 定数を使用する場合は、"sphparafort.fin"をインクルードしてください。

(5) ソルバーID

ノード間通信メソッドを使用する為には、複数のソルバーから通信を行うソルバを特定する為にソルバーIDを指定する必要があります。

ソルバーIDは"getSphereId()"関数にて取得します。

（使用例）

```
/*  ソルバークラス側  */
int sphSolverC3D::loopSolver(sphStepTime* steptime) {
      int solverID = getSphereId();              //  ソルバーID の取得を行う。
      int gc = array->getGc();
      int sz[3] = {size[0]-gc-1, size[1]-gc-1, size[2]-gc-1};
      //  ソルバーID を付加して Fortran 関数を呼び出す。
      c3d_jacobi_(& solverID, array->getData() , sz, &gc);
}

/* Fortran 側  */
subroutine c3d_jacobi (solverid, p, sz, g)
      implicit none
      include 'sphparafort.fin'
      integer                    ::    solverid, g
      integer, dimension(3)      ::    sz
      real, dimension(1-g:sz(1)+g, 1-g:sz(2)+g, 1-g:sz(3)+g)      :: p
      real          :: tmp, res
      tmp = 0.5
      //  ソルバーID を付加して通信サブルーチンを呼び出す。
      call sph_allreduce(solverid, tmp, res, 1, SPH_REAL4, SPH_SUM, ierr)
end subroutine c3d_jacobi
```

２．　サブルーチン一覧

No	サブルーチン名	説明
1	sph_isParallel(iret)	並列実行であるかを判断します。
2	sph_getNumberOfGroup(solverid, num, err)	ソルバーのプロセス数を取得します。
3	sph_getNumberOfNode(solverid, num, ierr)	ソルバーのノード数を取得します。
4	sph_getVoxelDivInfo (solverid, idiv, jdiv, kdiv, ierr)	計算領域のノードの分割数を取得します。
5	sph_getWholeVoxelSize (solverid, isz, jsz, ksz,　ierr)	ソルバーの計算領域全体のボクセルサイズを取得します。
6	sph_getVoxelHeadIndex (solverid, id, idim, idx, ierr)	ソルバーノードの計算領域全体に対する始点グローバルインデックスを取得します。
7	sph_getVoxelTailIndex (solverid, id, idim, idx, ierr)	ソルバーノードの計算領域全体に対する終点グローバルインデックスを取得します。
8	sph_getVoxelSize (solverid, isz, jsz, ksz, ierr)	自ノードが担当するボクセルサイズを取得します。
9	sph_getCommIDNum (solverid, i, j, k, id, num, ierr)	隣接ノード数を I,J,K 軸のプラス、マイナス方向別に取得します。
10	sph_getCommIDs(solverid, i, j, k, id_list, ierr)	隣接ノード ID のリストを I,J,K 軸のプラス、マイナス方向別に取得します。
11	sph_isCommID(solverid, i, j, k, num, ierr)	隣接ノードの有無を I,J,K 軸のプラス、マイナス方向別に取得します。
12	sph_getMyID (solverid, myid, ierr)	ソルバ内の自プロセスのローカルランク番号を取得します。
13	sph_broadcast (solverid, data, ic, idatatype, isrc, ierr)	ソルバー内のプロセスに対してデータのブロードキャストを行います。
14	sph_send (solverid, data, ic, idatatype, idst, ierr)	ソルバー内の受信先ノード ID に対してデータの同期送信を行います。
15	sph_recv (solverid, data, ic, idatatype, isrc, ierr)	ソルバー内の送信元ノード ID からデータの同期受信を行います。

No	サブルーチン名	説明
16	sph_iSend (solverid, data, ic, idatatype, idst, ireqkey, ierr)	ソルバー内の受信先ノード ID に対してデータの非同期送信を行います。
17	sph_iRecv(solverid,data, ic, idatatype, isrc, ireqkey, ierr)	ソルバー内の送信元ノード ID からのデータの非同期受信を行います。
18	sph_allreduce(solverid, sendbuf, recvbuf, ic, idatatype, ioptype, ierr)	ソルバー内のノード間にて演算を行います。
19	sph_wait(solverid, ireqkey, ierr)	非同期送受信の完了待ちを行います。
20	sph_waitAll(solverid, ikeynum, ikeylist, ierr)	複数の非同期送受信の完了待ちを行います。
21	sph_waitRequestAll (solverid, ierr)	すべての非同期送受信の完了待ちを行います。
22	sph_abort (ierrcode)	起動している全コミュニケータのプロセスを強制終了します。
23	sph_barrier(solverid, ierr)	ソルバーのすべてのノードが呼び出すまで、ブロッキングを行います。
24	sph_commBndCell(solverid, data, iface, ierr)	ソルバーの隣接ノード間にて仮想セル領域の通信を行います。
25	sph_printText (solverid, label, message)	テキストファイルにメッセージを出力します。
26	function real*8 sph_ getTime ()	現在時間の取得を行います。

３．　サブルーチン詳細

３．１　　並列実行有無の判断

subroutine sph_isParallel(iret)		
並列実行であるかを判断します。		
引数	integer iret [out]	並列実行の有無 iret = 1のとき並列実行 iret = -1のとき逐次実行

（使用例）

```
integer　　:: iret
```

```
      call sph_isParallel(iret)
      if ( iret == 1 ) then
            write(6,"('並列実行')")
      else
            write(6,"('逐次実行')")
      end if
```

３．２　　ソルバープロセス数の取得

subroutine sph_getNumberOfGroup(solverid, num, err)		
ソルバーのプロセス数を取得します。		
引数	integer solverid [in]	ソルバーID
	integer num [out]	プロセス数
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバーのプロセス数を取得します。

sph_getNumberOfNodeと同意の関数ですが、ソルバーのMPIグループ設定からMPI関数を用いてプロセス数を取得します。

（使用例）

```
      integer      ::  solverid
      integer      ::  ierr, iret
!     プロセス数の取得
      call sph_getNumberOfGroup(solverid, iret, ierr)
      write(6, "('group num = ', I4)") iret
```

３．３　　ソルバーノード数の取得

subroutine sph_getNumberOfNode(solverid, num, ierr)		
ソルバーのノード数を取得します。		
引数	integer solverid [in]	ソルバーID
	integer num [out]	ノード数
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバーのノード数を取得します。

sph_getNumberOfGroupと同意の関数ですが、ソルバーのパラレルコントローラから管理

を行っているノード数を取得します。

（使用例）

```
      integer      ::  solverid
      integer      ::  ierr, iret
!     ノード数の取得
      call sph_getNumberOfNode(solverid, iret, ierr)
      write(6, "('node num = ', I4)") iret
```

３．４　　分割情報の取得

subroutine sph_getVoxelDivInfo (solverid, idiv, jdiv, kdiv, ierr)		
計算領域のノードの分割数を取得します。		
引数	integer solverid [in]	ソルバーID
	integer idiv [out]	I方向分割数
	integer jdiv [out]	J方向分割数
	integer kdiv [out]	K方向分割数
	integer ierr [out]	戻り値（成功=1/失敗=-1）

均等分割における計算領域のノードの分割数を取得します。
均等分割ではない場合、戻り値(ierr)は-1を返します。

（使用例）

```
      integer      ::  solverid
      integer      :: idiv, jdiv, kdiv
      integer      ::  ierr, iret
!     ソルバーの分割情報を取得する。
      call sph_getVoxelDivInfo(solverid, idiv, jdiv, kdiv, ierr)
      write(6, "('div info = ', I4, I4, I4)") idiv, jdiv, kdiv
```

３．５　　計算領域のボクセルサイズ取得

subroutine sph_getWholeVoxelSize (solverid, isz, jsz, ksz,　ierr)		
ソルバーの計算領域全体のボクセルサイズを取得します。		
引数	integer solverid [in]	ソルバーID

	integer isz [out]	I方向計算領域ボクセルサイズ
	integer jsz [out]	J方向計算領域ボクセルサイズ
	integer ksz [out]	K方向計算領域ボクセルサイズ
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバーの計算領域全体のボクセルサイズを取得します。
取得ボクセルサイズにはガイドセルは含みません。

（使用例）

```
      integer      ::  solverid
      integer      ::  i, j, k, ix, jx, kx
      integer      ::  ierr
!     ソルバー計算領域全体を取得する。（ガイドセルを含まない）
      call sph_getWholeVoxelSize(solverid, ix, jx, kx, ierr)
      write(6, "('whole size = ', I4, I4, I4)") ix, jx, kx
```

３．６　　各ノードの計算領域全体のボクセル始点インデックスを取得

subroutine sph_getVoxelHeadIndex (solverid, id, idim, idx, ierr)		
ソルバーノードの計算領域全体に対する始点グローバルインデックスを取得します。		
引数	integer solverid [in]	ソルバーID
	integer id [in]	ノードID
	integer idim [in]	取得する方向(I=0 or J=1 or K=2)
	integer idx [out]	取得始点インデックス
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバーノードの計算領域全体に対する始点グローバルインデックスを取得します。
取得始点インデックスにはガイドセルは含みません。

（使用例）

```
      include 'sphparafort.fin'
      integer      ::  solverid
      integer      ::  i, j, k, ix, jx, kx
      integer      ::  myid
      integer      ::  ierr
!     自ノードの始点グローバルインデックスを取得する。（ガイドセルを含まない）
      call sph_getMyID(solverid, myid, ierr)          !  自ノードIDの取得
```

```
      call sph_getVoxelHeadIndex(solverid, myid, X_DIR, i, ierr)
      call sph_getVoxelHeadIndex(solverid, myid, Y_DIR, j, ierr)
      call sph_getVoxelHeadIndex(solverid, myid, Z_DIR, k, ierr)
      write(6, "('start index[', I2, ']=', I4, I4, I4)") myid, i, j, k
```

"X_DIR=0", " Y_DIR=1", "Z_DIR=2"は'sphparafort.fin'をincludeして使用してください。

３．７　各ノードの計算領域全体のボクセル終点インデックスを取得

subroutine sph_getVoxelTailIndex (solverid, id, idim, idx, ierr)		
ソルバーノードの計算領域全体に対する終点グローバルインデックスを取得します。		
引数	integer solverid [in]	ソルバーID
	integer id [in]	ノードID
	integer idim [in]	取得する方向(I=0 or J=1 or K=2)
	integer idx [out]	取得終点インデックス
	integer ierr [out]	戻り値（成功=1/失敗=-1)

ソルバーノードの計算領域全体に対する終点グローバルインデックスを取得します。
取得終点インデックスにはガイドセルは含みません。

（使用例）

```
      include 'sphparafort.fin'
      integer      :: solverid
      integer      :: i, j, k, ix, jx, kx
      integer      :: myid
      integer      :: ierr
!     自ノードの終点グローバルインデックスを取得する。（ガイドセルを含まない）
      call sph_getMyID(solverid, myid, ierr)        !  自ノードIDの取得
      call sph_getVoxelTailIndex(solverid, myid, X_DIR, i, ierr)
      call sph_getVoxelTailIndex(solverid, myid, Y_DIR, j, ierr)
      call sph_getVoxelTailIndex(solverid, myid, Z_DIR, k, ierr)
      write(6, "('end index[', I2, ']=', I4, I4, I4)") myid, i, j, k
```

"X_DIR=0", " Y_DIR=1", "Z_DIR=2"は'sphparafort.fin'をincludeして使用してください。

３．８　自ノード担当ボクセル領域のサイズ取得

subroutine sph_getVoxelSize (solverid, isz, jsz, ksz, ierr)		
自ノードが担当するボクセルサイズを取得します。		
引数	integer solverid [in]	ソルバーID
	integer isz [out]	I方向ボクセルサイズ
	integer jsz [out]	J方向ボクセルサイズ
	integer ksz [out]	K方向ボクセルサイズ
	integer ierr [out]	戻り値（成功=1/失敗=-1）

自ノードが担当するボクセルサイズを取得します。
取得ボクセルサイズにはガイドセルは含みません。

（使用例）

```
      include 'sphparafort.fin'
      integer      ::  solverid
      integer      ::  i, j, k, ix, jx, kx
      integer      ::  ierr
!     ノードサイズを取得する。（ガイドセルを含まない）
      call sph_getVoxelSize(solverid, ix, jx, kx, ierr)
      write(6, "('node size = ', I4, I4, I4)") ix, jx, kx
```

３．９　隣接するノード数を取得

subroutine sph_getCommIDNum (solverid, i, j, k, id, num, ierr)		
隣接ノード数を I,J,K 軸のプラス、マイナス方向別に取得します。		
引数	integer solverid [in]	ソルバーID
	integer i [in]	I軸方向に対する位置（minus=-1, plus=1）
	integer j [in]	J軸方向に対する位置（minus=-1, plus=1）
	integer k [in]	K軸方向に対する位置（minus=-1, plus=1）
	integer num [out]	隣接ノード数
	integer ierr [out]	戻り値（成功=1/失敗=-1）

隣接ノード数をI,J,K軸のプラス、マイナス方向別に取得します。

（使用例）

```
      include 'sphparafort.fin'
      integer      ::  solverid
      integer, dimension(6)  :: nums
```

```
      integer       ::   ierr
!     隣接ノード数を取得する。
      call sph_getCommIDNum(solverid, MINUS_DIR, 0, 0, nums(1), ierr)
      call sph_getCommIDNum(solverid, PLUS_DIR, 0, 0, nums(2), ierr)
      call sph_getCommIDNum(solverid, 0, MINUS_DIR, 0, nums(3), ierr)
      call sph_getCommIDNum(solverid, 0, PLUS_DIR, 0, nums(4), ierr)
      call sph_getCommIDNum(solverid, 0, 0, MINUS_DIR, nums(5), ierr)
      call sph_getCommIDNum(solverid, 0, 0, PLUS_DIR, nums(6), ierr)
      write(6, "('comm node size = ', I4, I4, I4, I4, I4, I4)") &
                  nums(1),nums(2),nums(3),nums(4),nums(5),nums(6)
```

"MINUS_DIR=-1", "PLUS_DIR =1"は'sphparafort.fin'をincludeして使用してください。

３．１０　隣接するノードの ID リストを取得

subroutine sph_getCommIDs(solverid, i, j, k, id_list, ierr)		
隣接ノード ID のリストを I,J,K 軸のプラス、マイナス方向別に取得します。		
引数	integer solverid [in]	ソルバーID
	integer i [in]	I軸方向に対する位置（minus=-1, plus=1）
	integer j [in]	J軸方向に対する位置（minus=-1, plus=1）
	integer k [in]	K軸方向に対する位置（minus=-1, plus=1）
	integer id_list [out]	隣接ノードIDリスト
	integer ierr [out]	戻り値（成功=1/失敗=-1）

隣接ノードIDのリストをI,J,K軸のプラス、マイナス方向別に取得します。

（使用例）

```
      include 'sphparafort.fin'
      integer       ::   solverid
      integer, dimension(6)   :: nums
      integer,dimension(:),allocatable :: id_list
      integer       ::   ierr
!     隣接ノード数を取得する。（I-MINUS方向）
      call sph_getCommIDNum(solverid, MINUS_DIR, 0, 0, nums(1), ierr)
!     隣接ノードIDリストを取得する。（I-MINUS方向）
      if ( nums(1) > 0 ) then
           allocate( id_list(nums(1)) )              ! id_listのアロケート
```

```
            call sph_getCommIDs(solverid, MINUS_DIR, 0, 0, id_list, ierr)
!           取得ノードリストの表示
            write(6, "('comm node ids[minus i_dir:', I2, '] = ', $)") nums(1)  ! ノード数
            do i=1, nums(1)          ! ノードリスト
                write(6, "(I2, $)") id_list(i)
            end do
            WRITE( *, '( 1H )' )
            deallocate(id_list)
        end if
```

"MINUS_DIR=-1", "PLUS_DIR =1"は'sphparafort.fin'をincludeして使用してください。

３．１１　隣接するノードの有無を取得

subroutine sph_isCommID(solverid, i, j, k, num, ierr)		
隣接ノードの有無を I,J,K 軸のプラス、マイナス方向別に取得します。		
引数	integer solverid [in]	ソルバーID
	integer i [in]	I軸方向に対する位置（minus=-1, plus=1）
	integer j [in]	J軸方向に対する位置（minus=-1, plus=1）
	integer k [in]	K軸方向に対する位置（minus=-1, plus=1）
	integer num [out]	隣接ノードの有無情報 ＞０：隣接ノード数 ＝０：隣接ノードなし ＝－１：最外郭面
	integer ierr [out]	戻り値（成功=1/失敗=-1）

隣接ノードの有無をI,J,K軸のプラス、マイナス方向別に取得します。
計算領域全体の最外郭面の場合は、"-1"を返し、最外郭面の有無の取得を行うことができる点が"sph_getCommIDNum"と異なります。

（使用例）

```
      include 'sphparafort.fin'
      integer       :: solverid
      integer, dimension(6)  :: nums
      integer       :: ierr
!     隣接ノードの有無を取得する。
```

```
      call sph_isCommID(solverid, MINUS_DIR, 0, 0, nums(1), ierr)
      call sph_isCommID(solverid, PLUS_DIR, 0, 0, nums(2), ierr)
      call sph_isCommID(solverid, 0, MINUS_DIR, 0, nums(3), ierr)
      call sph_isCommID(solverid, 0, PLUS_DIR, 0, nums(4), ierr)
      call sph_isCommID(solverid, 0, 0, MINUS_DIR, nums(5), ierr)
      call sph_isCommID(solverid, 0, 0, PLUS_DIR, nums(6), ierr)
      write(6, "('isCommID = ', I4, I4, I4, I4, I4, I4)") &
                        nums(1),nums(2),nums(3),nums(4),nums(5),nums(6)
```

"MINUS_DIR=-1", "PLUS_DIR =1"は'sphparafort.fin'をincludeして使用してください。

３．１２　ノード ID の取得

subroutine sph_getMyID (solverid, myid, ierr)		
ソルバ内の自プロセスのローカルランク番号を取得します。		
引数	integer solverid [in]	ソルバーID
	integer myid [out]	プロセスグループID
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバ内の自プロセスのローカルランク番号を取得します。。

使用例：

```
      include 'sphparafort.fin'
      integer      :: solverid
      integer      :: myid
      integer      :: ierr
!     ノードIDの取得
      call sph_getMyID(solverid, myid, ierr)
      write(6, "('myid = ', I4)") myid
```

３．１３　ブロードキャスト

subroutine sph_broadcast (solverid, data, ic, idatatype, isrc, ierr)		
ソルバー内のプロセスに対してデータのブロードキャストを行います。		
引数	integer solverid [in]	ソルバーID
	DATA_TYPE data [in/out]	データ

	integer ic [in]	データの数
	integer idatatype [in]	データ型：DATA_TYPE
	integer isrc [in]	送信元ノードID
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバー内のプロセスに対してデータのブロードキャストを行います。
データ型(DATA_TYPE)については、"１．概要（2）データ型：DATA_TYPE"を参照してください。

（使用例）

```
      include 'sphparafort.fin'
      integer       ::  solverid            ! ソルバID
      integer       ::  myid                ! 自ノードID
      integer       ::  g                   ! ガイドセル
      integer, dimension(3)         ::  sz        ! ボクセルサイズ
      real, dimension(1-g:sz(1)+g, 1-g:sz(2)+g, 1-g:sz(3)+g)  ::  p  ! データ
      integer       ::  ierr
      call sph_getMyID(solverid, myid, ierr)            ! 自ノードIDの取得
!       ダミーデータの作成を行う
      if (myid == 0) then
        do k=1-g,sz(3)+g
          do j=1-g,sz(2)+g
              do i=1-g,sz(1)+g
                p(i,j,k)=10*i+1*j+0.1*k
              end do
            end do
        end do
      end if
!     ブロードキャストを行う
      call sph_broadcast(solverid, p,     &
                                (sz(1)+2*g)*(sz(2)+2*g)*(sz(3)+2*g),     &
                                SPH_FLOAT, 0, ierr)
!     ブロードキャストデータのファイル出力を行う。（デバッグ用）
      do k=1-g,sz(3)+g
        write(msg, "('k=',I4)") k
        call sph_printText(solverid, "history", trim(msg)//char(10)//char(0));
        do j=1-g,sz(2)+g
```

```
        do i=1-g,sz(1)+g
          write(msg, "(f6.3)") p(i, j, k)
            call sph_printText(solverid, "history", trim(msg)//"    ");
        end do
        call sph_printText(solverid, "history", char(10)//char(0));
      end do
    end do
```

" SPH_FLOAT =-10", 及びその他のデータ型定数を使用する場合は'sphparafort.fin'をincludeして使用してください。

３．１４　同期データ送信

subroutine sph_send (solverid, data, ic, idatatype, idst, ierr)		
ソルバー内の受信先ノード ID に対してデータの送信を行います。		
引数	integer solverid [in]	ソルバーID
	DATA_TYPE data [in]	データ
	integer ic [in]	データの数
	integer idatatype [in]	データ型：DATA_TYPE
	integer idst [in]	受信先ノードID
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバー内の受信先ノードIDに対してデータの送信を行います。
送信はブロッキング送信となります。

（使用例）

```
    include 'sphparafort.fin'
    integer      ::  solverid            ! ソルバID
    integer      ::  myid                ! 自ノードID
    integer      ::  g                   ! ガイドセル
    integer, dimension(3)        ::  sz       ! ボクセルサイズ
    real, dimension(1-g:sz(1)+g, 1-g:sz(2)+g, 1-g:sz(3)+g)  ::  p  ! データ
    integer      ::  ierr
    call sph_getMyID(solverid, myid, ierr)          ! 自ノードIDの取得
!   ダミーデータの作成を行う
    if (myid == 0) then
      do k=1-g,sz(3)+g
```

```
            do j=1-g,sz(2)+g
              do i=1-g,sz(1)+g
                p(i,j,k)=1*i+0.1*j+0.01*k
              end do
            end do
        end do
      end if
!   送受信を行う
      if (myid == 0) then                    ！送信元ノード＝０
        call sph_send(solverid, p,     &
                      (sz(1)+2*g)*(sz(2)+2*g)*(sz(3)+2*g), SPH_FLOAT, 1, ierr)
      else if (myid == 1) then               ！受信先ノード＝１
        call sph_recv(solverid, p,    &
                       (sz(1)+2*g)*(sz(2)+2*g)*(sz(3)+2*g), SPH_FLOAT, 0, ierr)
      end if
!     データのファイル出力を行う。（デバッグ用）
      do k=1-g,sz(3)+g
        write(msg, "('k=',I4)") k
        call sph_printText(solverid, "history", trim(msg)//char(10)//char(0));
        do j=1-g,sz(2)+g
          do i=1-g,sz(1)+g
            write(msg, "(f6.3)") p(i, j, k)
              call sph_printText(solverid, "history", trim(msg)//"    ");
          end do
          call sph_printText(solverid, "history", char(10)//char(0));
        end do
      end do
```

" SPH_FLOAT =-10", 及びその他のデータ型定数を使用する場合は'sphparafort.fin'をincludeして使用してください。

３．１５　同期データ受信

subroutine sph_recv (solverid, data, ic, idatatype, isrc, ierr)	
ソルバー内の受信先ノード ID に対してデータの送信を行います。	
引数	integer solverid [in]　　ソルバーID

	DATA_TYPE data[out]	データ
	integer ic[in]	データの数
	integer idatatype [in]	データ型：DATA_TYPE
	integer isrc [in]	送信元ノードID
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバー内の送信元ノードIDからデータを受信します。
送信はブロッキング受信となり、データの送信が完了するまで完了待ちの状態となります。

（使用例：同期データ送信参照）

３．１６　非同期データ送信

subroutine sph_iSend (solverid, data, ic, idatatype, idst, ireqkey, ierr)		
ソルバー内の受信先ノード ID に対してデータの送信を行います。		
引数	integer solverid [in]	ソルバーID
	DATA_TYPE data[out]	データ
	integer ic[in]	データの数
	integer idatatype [in]	データ型：DATA_TYPE
	integer idst [in]	受信先ノードID
	integer ireqkey [out]	送信リクエストキー sph_wait又はsph_waitAllメソッドで使用
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバー内の受信先ノードIDに対してデータの送信を行います。
送受信は非ブロッキング通信となり、通信の完了待ちとなることはありません。
しかし、送信リクエストキー(ireqkey)にてsph_wait, sph_waitAll又はsph_waitRequestAll メソッドを呼び出し、完了待ちを行わなければなりません。

（使用例）

```
include 'sphparafort.fin'
integer      ::  solverid            ! ソルバID
integer      ::  myid                ! 自ノードID
integer      ::  g                   ! ガイドセル
integer, dimension(3)        ::  sz       ! ボクセルサイズ
real, dimension(1-g:sz(1)+g, 1-g:sz(2)+g, 1-g:sz(3)+g)  ::  p  ! データ
integer      ::  ireqkey,  ierr
```

```
      call sph_getMyID(solverid, myid, ierr)              ！自ノードIDの取得
!     ダミーデータの作成を行う
      if (myid == 0) then
        do k=1-g,sz(3)+g
          do j=1-g,sz(2)+g
            do i=1-g,sz(1)+g
              p(i,j,k)=0.1*i+0.01*j+0.001*k
            end do
          end do
        end do
      end if
!     非同期送受信を行う
      if (myid == 0) then                    ！送信元ノード＝０
        call sph_iSend(solverid, p,     &
                    (sz(1)+2*g)*(sz(2)+2*g)*(sz(3)+2*g), SPH_FLOAT, 1, ireqkey , ierr)
      else if (myid == 1) then                    ！受信先ノード＝１
        call sph_iRecv(solverid, p,    &
                    (sz(1)+2*g)*(sz(2)+2*g)*(sz(3)+2*g), SPH_FLOAT, 0, ireqkey , ierr)
      end if
!       （他の処理を行う）
!      通信の完了待ちを行う。
      call sph_wait (solverid, ireqkey, ierr)
!      データのファイル出力を行う。（デバッグ用）
      do k=1-g,sz(3)+g
        write(msg, "('k=',I4)") k
        call sph_printText(solverid, "history", trim(msg)//char(10)//char(0));
        do j=1-g,sz(2)+g
          do i=1-g,sz(1)+g
            write(msg, "(f6.3)") p(i, j, k)
              call sph_printText(solverid, "history", trim(msg)//"    ");
          end do
          call sph_printText(solverid, "history", char(10)//char(0));
        end do
      end do
```

" SPH_FLOAT =-10", 及びその他のデータ型定数を使用する場合は'sphparafort.fin'をincludeして使用してください。

３．１７　非同期データ受信

subroutine sph_iRecv(data, ic, idatatype, isrc, ireqkey, ierr)		
ソルバー内の送信元ノード ID からのデータの受信を行います。		
引数	integer solverid [in]	ソルバーID
	DATA_TYPE data[out]	データ
	integer ic[in]	データの数
	integer idatatype [in]	データ型：DATA_TYPE
	integer isrc [in]	送信元ノードID
	integer ireqkey [out]	受信リクエストキー
		sph_wait又はsph_waitAllメソッドで使用
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバー内の送信元ノードIDからのデータの受信を行います。
送受信は非ブロッキング通信となり、通信の完了待ちとなることはありません。
しかし、受信リクエストキー(ireqkey)にてsph_wait, sph_waitAll又はsph_waitRequestAll メソッドを呼び出し、完了待ちを行わなければなりません。

（使用例：非同期データ送信参照）

３．１８　ノード間演算処理

subroutine sph_allreduce(solverid, sendbuf, recvbuf, ic, idatatype, ioptype, ierr)		
ソルバー内の送信元ノード ID からのデータの受信を行います。		
引数	integer solverid [in]	ソルバーID
	DATA_TYPE sendbuf [in]	送信バッファ
	DATA_TYPE recvbuf [out]	受信バッファ
	integer ic [in]	データの数
	integer idatatype [in]	データ型：DATA_TYPE
	integer ioptype [in]	演算の種類：OP_TYPE
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバー内のノード間にて演算を行います。

（使用例）

```
      integer       ::   solverid
      real          :: tmp, sum_val
      integer       ::   ierr, iret
!     ソルバーノード間で合計演算を行う。
        tmp = 1.5
        sum_val = 0.0
        call sph_allreduce(solverid, tmp, sum_val, 1, SPH_REAL4, SPH_SUM, ierr)
        write(6, "('sum value = ', f6.3)") sum_val
```

" SPH_REAL4 =-10", 及びその他のデータ型定数、SPH_SUM=102, 及びその他の演算定数を使用する場合は'sphparafort.fin'をincludeして使用してください。

３．１９　非同期通信の完了待ち

subroutine sph_wait(solverid, ireqkey, ierr)		
非同期送受信の完了待ちを行います。		
引数	integer solverid [in]	ソルバーID
	integer ireqkey [out]	送信(受信)リクエストキー
	integer ierr [out]	戻り値（成功=1/失敗=-1)

"非同期データ送信(sph_iSend)", "非同期データ受信(sph_iRecv)"の送受信完了待ちを行います。

sph_iSend/ sph_iRecvにて受け取った送信/受信リクエストキーの送受信完了待ちを行います。

sph_iSend/ sph_iRecvによって取得したリクエストキーはsph_wait, sph_waitAll又はsph_waitRequestAllメソッドによって、すべての送受信の完了を行わなければなりません。

（使用例：非同期データ送信参照）

３．２０　複数の非同期通信の完了待ち

subroutine sph_waitAll(solverid, ikeynum, ikeylist, ierr)		
複数の非同期送受信の完了待ちを行います。		
引数	integer solverid [in]	ソルバーID
	integer ikeynum [in]	送信(受信) リクエストキーリストの数 （ikeylistの要素数）

	integer ikeylist [in]	送信(受信)リクエストキーリスト
	integer ierr [out]	戻り値（成功=1/失敗=-1)

"非同期データ送信(sph_iSend)", "非同期データ受信(sph_iRecv)"の送受信完了待ちを行います。

sph_iSend/ sph_iRecvにて受け取った送信/受信リクエストキーの配列であるリクエストキーリストのすべての送受信完了待ちを行います。

sph_iSend/ sph_iRecvによって取得したリクエストキーはsph_wait, sph_waitAll又はsph_waitRequestAllメソッドによって、すべての送受信の完了を行わなければなりません。

（使用例）

```
      include 'sphparafort.fin'
      integer       ::  solverid              ！ソルバID
      integer       ::  myid                  ！自ノードID
      integer       ::  g                     ！ガイドセル
      integer, dimension(3)         ::  sz        ！ボクセルサイズ
      real, dimension(1-g:sz(1)+g, 1-g:sz(2)+g, 1-g:sz(3)+g)  ::  p  ！データ
      integer, dimension(2)       ::  ireqkey_list
      integer       ::  ierr
!     （省略：非同期データ送信参照）

!     非同期送受信を行う
      if (myid == 0) then                   ！送信元ノード＝０
        call sph_iSend(solverid, p,     &
                    (sz(1)+2*g)*(sz(2)+2*g), SPH_FLOAT, 1, ireqkey_list(1), ierr)
        call sph_iSend(solverid, p,     &
                    (sz(1)+2*g)*(sz(2)+2*g), SPH_FLOAT, 2, ireqkey_list(1), ierr)
      else if (myid /= 1) then                  ！受信先ノード！＝０
        call sph_iRecv(solverid, p,    &
                   (sz(1)+2*g)*(sz(2)+2*g), SPH_FLOAT, 0, ireqkey_list(0), ierr)
      end if
!       （他の処理を行う）
!      通信の完了待ちを行う。
      call sph_waitAll (solverid, 2, ireqkey_list, ierr)

!     （省略：非同期データ送信参照）
```

３．２１　すべての非同期通信の完了待ち

subroutine sph_waitRequestAll (solverid, ierr)		
すべての非同期送受信の完了待ちを行います。		
引数	integer solverid [in]	ソルバーID
	integer ierr [out]	戻り値（成功=1/失敗=-1）

"非同期データ送信(sph_iSend)", "非同期データ受信(sph_iRecv)"を行ったすべての通信の完了待ちを行います。

sph_iSend/ sph_iRecvにて受け取った送信/受信リクエストキーを使用せず、通信が完了となっていないすべての非同期送信／受信の完了待ちを行います。

（使用例）

```
      include 'sphparafort.fin'
      integer       ::  solverid              ! ソルバID
      integer       ::  myid                  ! 自ノードID
      integer       ::  g                     ! ガイドセル
      integer, dimension(3)         ::  sz        ! ボクセルサイズ
      real, dimension(1-g:sz(1)+g, 1-g:sz(2)+g, 1-g:sz(3)+g)  ::  p  ! データ
      integer, dimension(2)       ::  ireqkey_list
      integer       ::  ierr
!     （省略：非同期データ送信参照）

!     非同期送受信を行う
      if (myid == 0) then                   ! 送信元ノード＝０
        call sph_iSend(solverid, p,     &
                    (sz(1)+2*g)*(sz(2)+2*g), SPH_FLOAT, 1, ireqkey_list(1), ierr)
        call sph_iSend(solverid, p,     &
                    (sz(1)+2*g)*(sz(2)+2*g), SPH_FLOAT, 2, ireqkey_list(1), ierr)
      else if (myid /= 1) then                  ! 受信先ノード！＝０
        call sph_iRecv(solverid, p,    &
                   (sz(1)+2*g)*(sz(2)+2*g), SPH_FLOAT, 0, ireqkey_list(0), ierr)
      end if
!       （他の処理を行う）
!      すべての通信の完了待ちを行う。
```

```
      call sph_waitRequsetAll (solverid, ierr)


!   (省略：非同期データ送信参照)
```

３．２２　強制終了

subroutine sph_abort (ierrcode)		
起動している全コミュニケータのプロセスを強制終了します。		
引数	integer ierrcode [out]	エラーコード

起動している全コミュニケータのプロセスを強制終了します。

（使用例）

```
      integer      ::  solverid
      integer      ::  myid
      integer      ::  ierr
!     ノードIDの取得
      call sph_getMyID(solverid, myid, ierr)
      if (ierr == -1) then
          ! エラー
          call sph_abort(-1)
      end if
```

３．２３　バリア

subroutine sph_barrier(solverid, ierr)		
ソルバーのすべてのノードが呼び出すまで、ブロッキングを行います。		
引数	integer solverid [in]	ソルバーID
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバーのすべてのノードが呼び出すまで、ブロッキングを行います。

（使用例）

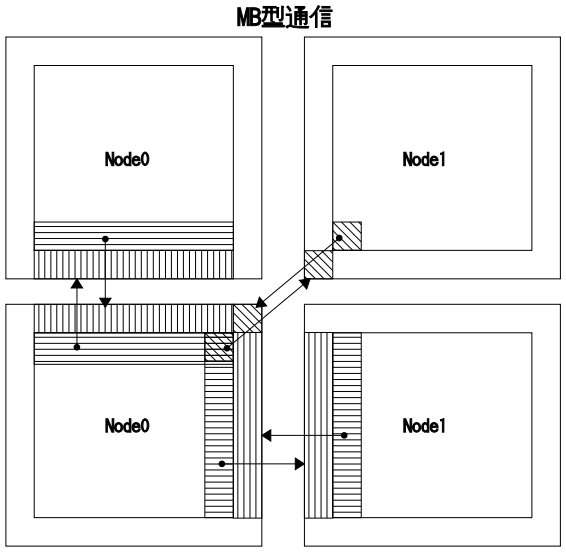
```
      integer      ::  solverid
      integer      ::  ierr
      real*8     :: sph_getTime
```

```
      real*8      :: cur_time00, cur_time01
!   バリア
      call sph_barrier(solverid, ierr)
      cur_time01 = sph_getTime()                    ! 時間測定
```

３．２４　仮想セルの値の更新

subroutine sph_commBndCell(solverid, data, iface, ierr)		
ソルバーの隣接ノード間にて仮想セル領域の通信を行います。		
引数	integer solverid [in]	ソルバーID
	DATA_TYPE data[in/out]	データ
	integer iface [in]	仮想セルの通信面の数
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバーの隣接ノード間にて仮想セル領域の通信を行い、仮想セルの値の更新を行います。
並列実行時の分割方法は均等分割、マルチボックス分割共に使用可能です。



（使用例）

```
      include 'sphparafort.fin'
      integer       ::   solverid            ! ソルバID
      integer       ::   myid                ! 自ノードID
      integer       ::   g                   ! ガイドセル
      integer, dimension(3)        ::   sz       ! ボクセルサイズ
```

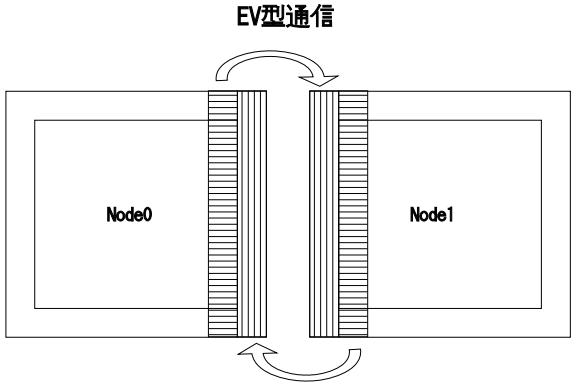
```
      real, dimension(1-g:sz(1)+g, 1-g:sz(2)+g, 1-g:sz(3)+g)  ::  p  ! データ
      integer       ::  ierr
      call sph_getMyID(solverid, myid, ierr)            ! 自ノードIDの取得
!     ダミーデータの作成を行う
      do k=1-g,sz(3)+g
          do j=1-g,sz(2)+g
              do i=1-g,sz(1)+g
                  p(i,j,k)=myid+0.1*i+0.01*j+0.001*k
              end do
          end do
      end do
!     ガイドセルの更新
      call sph_commBndCell(solverid, p, g, ierr)
!     データのファイル出力を行う。（デバッグ用）
      do k=1-g,sz(3)+g
          write(msg, "('k=',I4)") k
          call sph_printText(solverid, "history", trim(msg)//char(10)//char(0));
          do j=1-g,sz(2)+g
              do i=1-g,sz(1)+g
                write(msg, "(f6.3)") p(i, j, k)
                call sph_printText(solverid, "history", trim(msg)//"    ");
              end do
            call sph_printText(solverid, "history", char(10)//char(0));
            end do
       end do
```

３．２５　仮想セルの値の更新（EV 型通信）

subroutine sph_commBndCellEv(solverid, data, iface, ierr)		
ソルバーの隣接ノード間にて仮想セル領域の EV 型の通信にて行います。		
引数	integer solverid [in]	ソルバーID
	DATA_TYPE data[in/out]	データ
	integer iface [in]	仮想セルの通信面の数
	integer ierr [out]	戻り値（成功=1/失敗=-1）

ソルバーの隣接ノード間にて仮想セル領域の通信を行い、仮想セルの値の更新を行います。

通信方式はEV型形式の通信を行います。
EV型の通信は均等分割によるソルバー実行のみ行うことができます。



（使用例：sph_commBndCell参照）

３．２６　仮想セルの値の更新（周期通信）

subroutine sph_commPeriodicBndCell(solvID, data, iface, iface_dir, iaxis, ierr)		
ソルバーの隣接ノード間にて仮想セル領域の EV 型の通信にて行います。		
引数	integer solverid [in]	ソルバーID
	DATA_TYPE data[in/out]	データ
	integer iface [in]	仮想セルの通信面の数
	integer iface_dir [in]	通信面
	integer iaxis [in]	通信方向
	integer ierr [out]	戻り値（成功=1/失敗=-1）

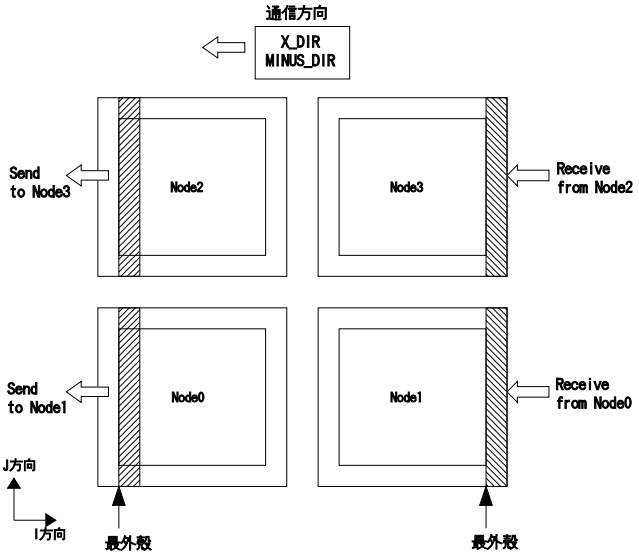
ソルバーの最外郭の周期ノード間にて仮想セル領域の通信を行い、仮想セルの値の更新を行います。
通信方式はEV型形式の通信を行います。
EV型の通信は均等分割によるソルバー実行のみ行うことができます。
通信面、及び通信方向は"sphparafort.fin"をインクルードして"方向定義定数"を使用してください。

（通信面：iface_dir）

X_DIR = 0　　I方向
Y_DIR = 1　　J方向
Z_DIR = 2　　K方向

（通信方向：iaxis）

MINUS_DIR = -1　　マイナス方向
PLUS_DIR = 1　　プラス方向



３．２７　デバッグ用テキストファイル出力

subroutine sph_ printText (solverid, label, message)		
テキストファイルへのメッセージ出力を行います。		
引数	integer solverid [in]	ソルバーID
	character*32 file_label [in]	ファイルラベル（コンフィグレーション）
	character*X message [in]	出力メッセージ

テキストファイルへのメッセージ出力を行います。
コンフィグレーションに定義されたファイルラベルの出力設定に従ってテキストファイルにメッセージを出力します。
出力メッセージに変数(character*X)を使用する際は、"trim()"を使用して、前後のスペースの削除、NULL("char(0)"の付加を行ってください。
また、改行は行いませんので、改行の必要がある場合は改行コード（"char(10)"）を付加してください。
ファイルラベルの設定については、コンフィグレーションマニュアルを参照してください。（出力ファイル名、出力ノード番号等の設定を行う）
このテキストファイル出力は、ソルバ性能に影響を与えますので確認、デバッグ用としてのみお使いください。

（使用例）

```
      include 'sphparafort.fin'
      integer       ::  solverid              ! ソルバID
      integer       ::  myid                  ! 自ノードID
      integer       ::  g                     ! ガイドセル
      integer, dimension(3)          ::  sz        ! ボクセルサイズ
      real, dimension(1-g:sz(1)+g, 1-g:sz(2)+g, 1-g:sz(3)+g)  ::  p  ! データ
       character*64 msg
!     データのファイル出力を"history",ラベルの設定にて行う。（デバッグ用）
      do k=1-g,sz(3)+g
!           k=XXXXの出力
            write(msg, "('k=',I4)") k
!           trim() + 改行コード ＋ NULL
            call sph_printText(solverid, "history", trim(msg)//char(10)//char(0));
            do j=1-g,sz(2)+g
                do i=1-g,sz(1)+g
                  write(msg, "(f6.3)") p(i, j, k)
                  call sph_printText(solverid, "history", trim(msg)//"    ");
                end do
!               改行のみ出力
                call sph_printText(solverid, "history", char(10)//char(0));
              end do
       end do
```

３．２８　現在時間の取得

function real*8 sph_ getTime ()	
現在時間を取得します。	
引数	なし
戻り値	real*8　　現在時間

現在時間を取得します。（ソルバー実行におけるループ時間ではありません。）

現在時間はsplGetTime()関数より取得しています。

splGetTime()関数の時間の取得方法は以下となっています。

```
/****     splGetTime()関数    ****/
          double splGetTime() {
```

```
        struct timeval tv;
        gettimeofday(&tv, NULL);
        double t = 0.0;
        t += (double)tv.tv_sec;
        t += (double)tv.tv_usec * 1.0e-6;
        return t;
    }
/*********************************/
```

（使用例）

```
    integer       ::  solverid
    integer       ::  ierr
    real*8      :: sph_getTime
    real*8      :: cur_time01
!   バリア
    call sph_barrier(solverid, ierr)
    cur_time01 = sph_getTime()                ! 時間取得
```