Sphere

Skeleton for PHysical and Engineering REsearch

Ver 2.0.0

データクラスマニュアル

2010 年 8 月 01 日

目次

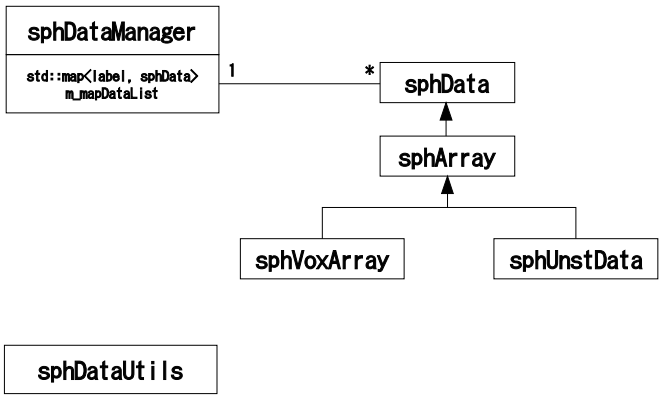
１．概要

データクラスとは内部に配列の実体を持ち、インデックスの管理やデータ通信通信機能を実装したクラス群です。データマネージャはソルバー内でインスタンスされたデータクラスオブジェクトの管理を行います。

データクラスとデータマネージャを用いることで、配列データの管理を容易に行うことが可能となり、ソルバー側ではデータ配列の生成、破棄は行う必要が無くなりメモリ管理を行う必要はなくなります。

１．１　クラス構成

SPHERE でのデータクラス構成は、以下となっています。



クラス名	説明
sphData	データクラスの基底クラスです。 データクラスタイプ、データ型タイプ、パラレルコントローラの管理を行います。
sphArray	データ型のテンプレートクラスです。 1 次元の配列データ（配列データの実体）を持ち、基底クラスから派生した N 次元データクラスはユーザに N 次元のインデックスを用いて 1 次元配列データにアクセスするインターフェイスを提供します。
sphVoxArray	構造格子のデータにて実際に生成されるクラスです。 ノード間のデータ通信、データコピーメソッドを実装しています
sphUnstData	非構造格子のデータにて実際に生成されるクラスです。

１．２　データクラスタイプ

データのスカラ、ベクトル、次元数を定義します。

サポートするデータクラスタイプの種類は以下です。

No	format 属性値	次元数	スカラ ベクトル	データクラス説明	SphDCType 値
1	"scalar1d"	1	スカラ	1D データクラス データ配列{i}	SPH_DC_ARRAY_1D
2	"scalar2d"	2	スカラ	2D データクラス データ配列{i, j}	SPH_DC_ARRAY_2D
3	"scalar3d"	3	スカラ	3D データクラス データ配列{i, j, k}	SPH_DC_ARRAY_3D
4	"vector1d"	1	ベクトル	1D データクラス データ配列{i, l}	SPH_DC_ARRAY_1DN
5	"vector1drv"	1	ベクトル	1D データクラス データ配列{l, i}	SPH_DC_ARRAY_1DNRV
6	"vector2d"	2	ベクトル	2D データクラス データ配列{i, j, l}	SPH_DC_ARRAY_2DN
7	"vector2drv"	2	ベクトル	2D データクラス データ配列{l, i, j}	SPH_DC_ARRAY_2DNRV
8	"vector3d"	3	ベクトル	3D データクラス データ配列{i, j, k,l}	SPH_DC_ARRAY_3DN
9	"vector3drv"	3	ベクトル	3D データクラス データ配列{l,i, j, k}	SPH_DC_ARRAY_3DNRV
10	"unst_node"	3	スカラ	非構造データクラス （節点、スカラ）	SPH_DC_UNST_NODE
11	"unst_node_n"	3	ベクトル	非構造データクラス （節点、ベクトル）	SPH_DC_UNST_NODE_N
12	"unst_node_nrv"	3	ベクトル	非構造データクラス （節点、ベクトル、反転）	SPH_DC_UNST_NODE_NRV
13	"unst_elem"	3	スカラ	非構造データクラス （要素、スカラ）	SPH_DC_UNST_ELEM
14	"unst_elem_n"	3	ベクトル	非構造データクラス （要素、ベクトル）	SPH_DC_UNST_ELEM_N
15	"unst_elem_nrv"	3	ベクトル	非構造データクラス （要素、ベクトル、反転）	SPH_DC_UNST_ELEM_NRV

１．３　データ型タイプ

データのデータ型を定義します。

サポートするデータタイプは以下です。

No	data_type 属性	データタイプ説明	SPL_Datatype
1	"char"	char 型データ	SPL_CHAR
2	"unsigned_char"	unsigned char データ型	SPL_UNSIGNED_CHAR
3	"short"	short 型データ型	SPL_SHORT
4	"unsigned_short"	unsigned short データ型	SPL_UNSIGNED_SHORT
5	"int"	int 型データ型	SPL_INT
6	"unsigned_int"	unsigned int データ型	SPL_UNSIGNED
7	"long"	long 型データ型	SPL_LONG
8	"unsigned_long"	unsigned long データ型	SPL_UNSIGNED_LONG
9	"longlong"	long long 型データ型	SPL_LONG_LONG
10	"float"	float 型データ型	SPL_FLOAT
11	"double"	double 型データ型	SPL_DOUBLE

１．４　データ定義位置

データの定義位置を定義します。

データ定義タイプは以下の４つをサポートします。

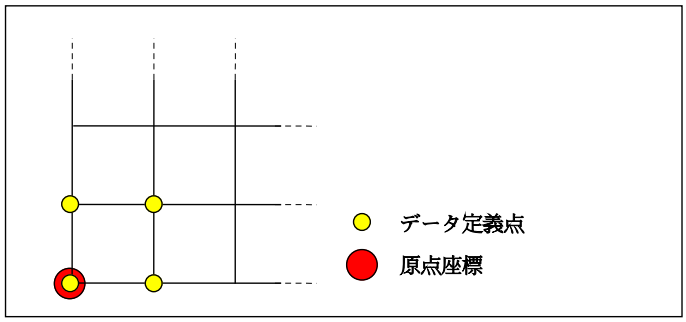
"staggered_1", "staggered_2"については、データクラス（format 属性)、データ数（data_num 属性）はベクトルデータクラスのみサポートします。

No	crddef 属性値	定義説明	サポート対象	
			データクラス	データ数
1	"regular"	格子上データ定義（デフォルト）	すべて	無制限
2	"collocate"	格子中心データ定義	すべて	無制限
3	"staggered_1"	格子中心マイナス方向データ定義	"vector2d" "vector2drv" "vector3d" "vector3drv"	次元数
4	"staggered_2"	格子中心プラスデータ定義	"vector2d" "vector2drv" "vector3d" "vector3drv"	次元数

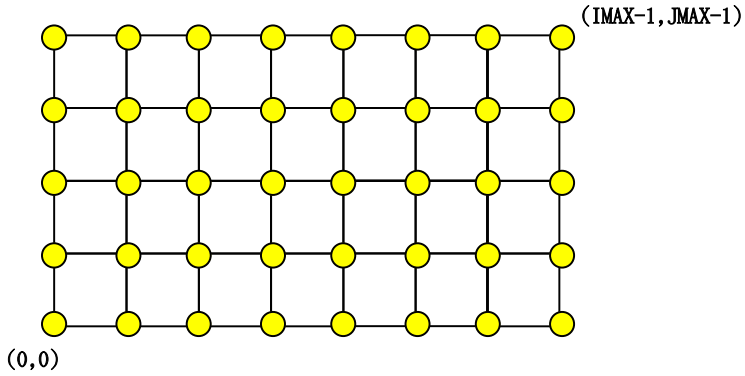
（１）regular

原点座標から格子ピッチのグリッドの交わる点にデータを定義する。

よって、データ座標は原点座標と格子ピッチに等間隔で配置される。



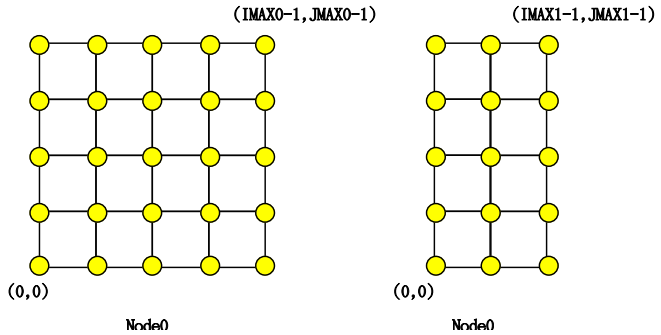
SphSize 要素の{ix, iy, iz}属性の設定値を{IMAX, JMAX, KMAX}のボクセルサイズにて定義した場合、データ領域は以下のサイズの確保を行う。



アロケートサイズ ={IMAX, JMAX, KMAX}

領域内データ範囲 ＝ {0,0,0}～{IMAX-1, JMAX-1, KMAX-1}

並列実行においてもすべてのノードで分割ボクセルサイズとデータサイズは等しい。



計算領域全体{IMAX, JMAX, KMAX}を Node0={IMAX0,JMAX0,KMAX0}, Node1={IMAX1,JMAX1,KMAX1}で分割した場合

Node0

アロケートサイズ ={IMAX0, JMAX0, KMAX0}

領域内データ範囲 ＝ {0,0,0}～{IMAX0-1, JMAX0-1, KMAX0-1}

Node1

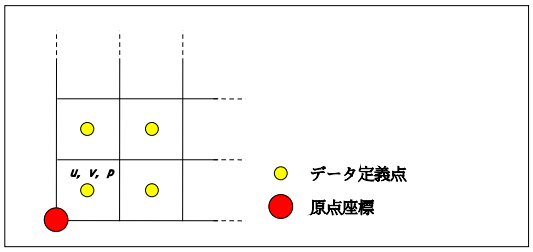
アロケートサイズ ={IMAX1, JMAX1, KMAX1}

領域内データ範囲 ＝ {0,0,0}～{IMAX1-1, JMAX1-1, KMAX1-1}

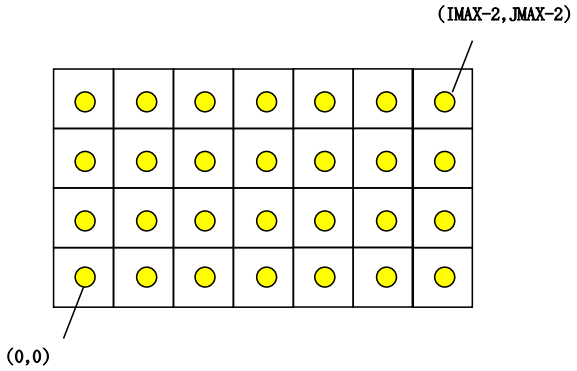
（２）collocate

格子の中央にデータを定義する。

よって、データ座標は**原点座標に対して格子ピッチ／２の位置に配置される。**



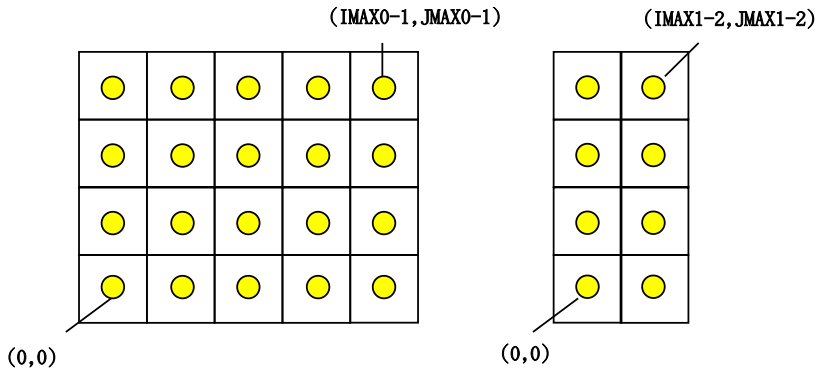
SphSize 要素の{ix, iy, iz}属性の設定値を{IMAX, JMAX, KMAX}のボクセルサイズにて定義した場合、データ領域は以下のサイズの確保を行う。



アロケートサイズ ={IMAX-1, JMAX-1, KMAX-1}

領域内データ範囲 ＝ {0,0,0}～{IMAX-2, JMAX-2, KMAX-2}

並列実行においては最外郭ノードのみボクセルサイズに対してデータサイズは－１少なくなる。



計算領域全体{IMAX, JMAX, KMAX}を Node0={IMAX0,JMAX0,KMAX0}, Node1={IMAX1,JMAX1,KMAX1}で分割した場合

Node0

アロケートサイズ ={IMAX0, JMAX0, KMAX0}

領域内データ範囲 ＝ {0,0,0}～{IMAX0-1, JMAX0-1, KMAX0-1}

Node1

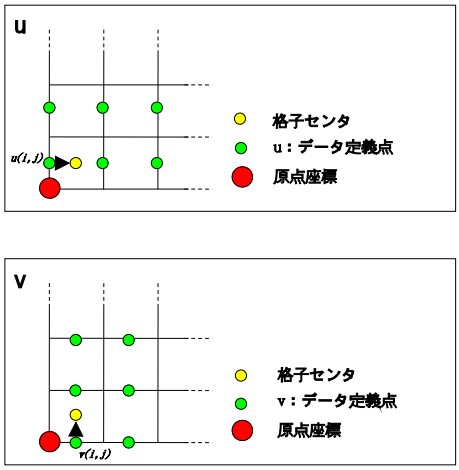
アロケートサイズ ={IMAX1-1, JMAX1-1, KMAX1-1}

領域内データ範囲 ＝ {0,0,0}～{IMAX1-2, JMAX1-2, KMAX1-2}

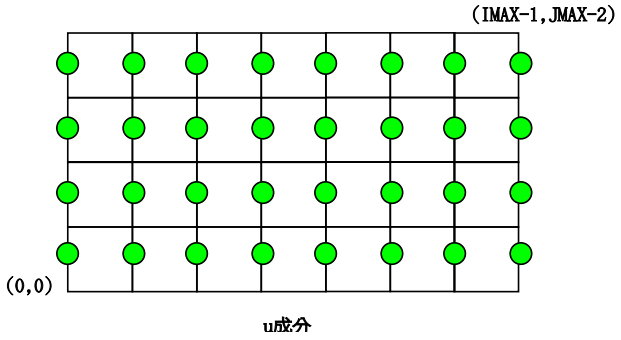
（３）staggered_1

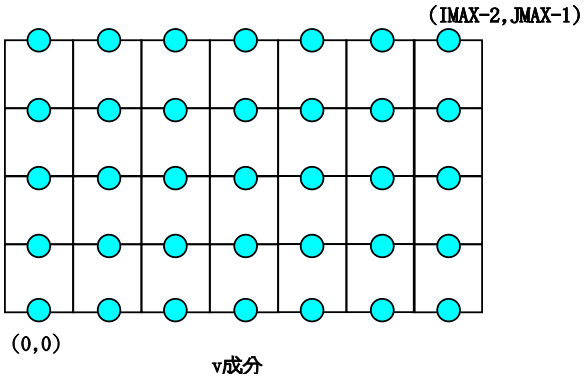
u, v, w のベクトルデータであり、格子センタから u は-I 方向、v は-J 方向、w は-K 方向

に格子ピッチ／２の位置に配置される。



SphSize 要素の{ix, iy, iz}属性の設定値を{IMAX, JMAX, KMAX}のボクセルサイズにて定義した場合、データ領域は以下のサイズの確保を行う。





アロケートサイズ ={IMAX, JMAX, KMAX}

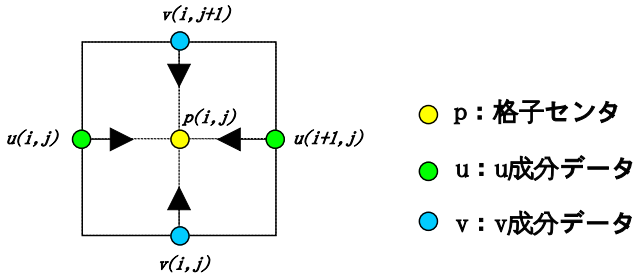
領域内データ範囲 ＝

u 成分:{0,0,0}～{IMAX-1, JMAX-2, KMAX-2}

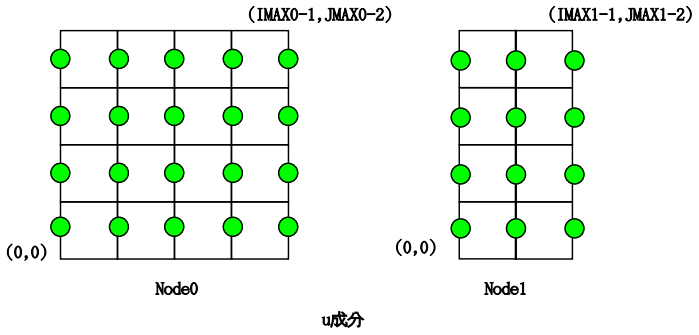
v 成分:{0,0,0}～{IMAX-2, JMAX-1, KMAX-2}

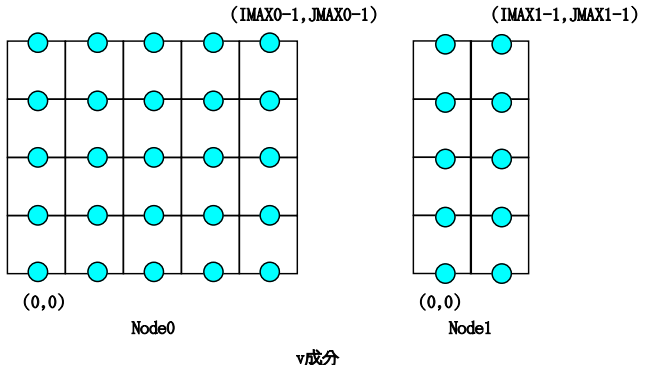
w 成分:{0,0,0}～{IMAX-2, JMAX-2, KMAX-1}

格子センタに p(圧力)データが存在する場合、p と u,v,w 成分の関係は以下となる。



並列実行においてもすべてのノードで分割ボクセルサイズとデータサイズは等しい。





計算領域全体{IMAX, JMAX, KMAX}を Node0={IMAX0,JMAX0,KMAX0}, Node1={IMAX1,JMAX1,KMAX1}で分割した場合

Node0

アロケートサイズ ={IMAX0, JMAX0, KMAX0}

領域内データ範囲 ＝

u 成分:{0,0,0}～{IMAX0-1, JMAX0-2, KMAX0-2}

v 成分:{0,0,0}～{IMAX0-2, JMAX0-1, KMAX0-2}

w 成分:{0,0,0}～{IMAX0-2, JMAX0-2, KMAX0-1}

Node1

アロケートサイズ ={IMAX1, JMAX1, KMAX1}

領域内データ範囲 ＝

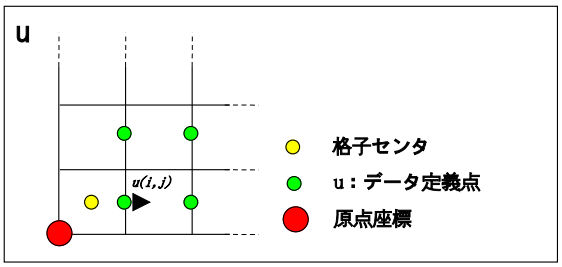
u 成分:{0,0,0}～{IMAX1-1, JMAX1-2, KMAX1-2}

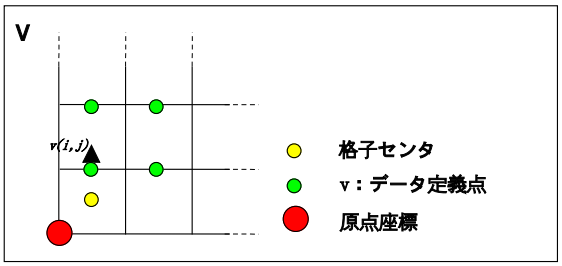
v 成分:{0,0,0}～{IMAX1-2, JMAX1-1, KMAX1-2}

w 成分:{0,0,0}～{IMAX1-2, JMAX1-2, KMAX1-1}

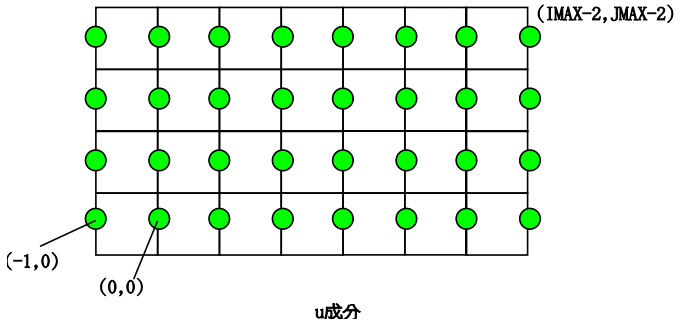
（４）staggered_2

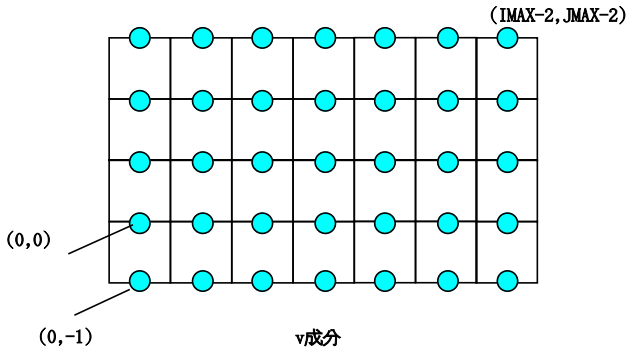
u, v, w のベクトルデータであり、格子センタから u は+I 方向、v は+J 方向、w は+K 方向に格子ピッチ／２の位置に配置される。





SphSize 要素の{ix, iy, iz}属性の設定値を{IMAX, JMAX, KMAX}のボクセルサイズにて定義した場合、データ領域は以下のサイズの確保を行う。





アロケートサイズ ={IMAX, JMAX, KMAX}

領域内データ範囲 ＝

u 成分:{-1,0,0}～{IMAX-2, JMAX-2, KMAX-2}

v 成分:{0,-1,0}～{IMAX-2, JMAX-2, KMAX-2}

w 成分:{0,0,-1}～{IMAX-2, JMAX-2, KMAX-2}

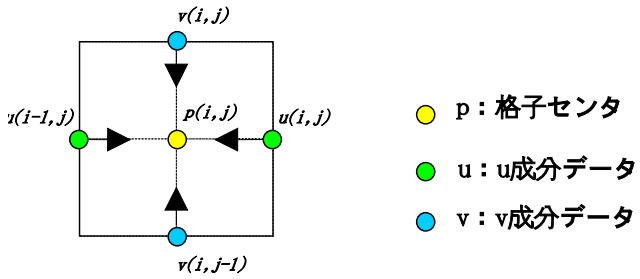
実際のデータ配列上のデータ範囲 ＝

u 成分:{0,0,0}～{IMAX-1, JMAX-2, KMAX-2}

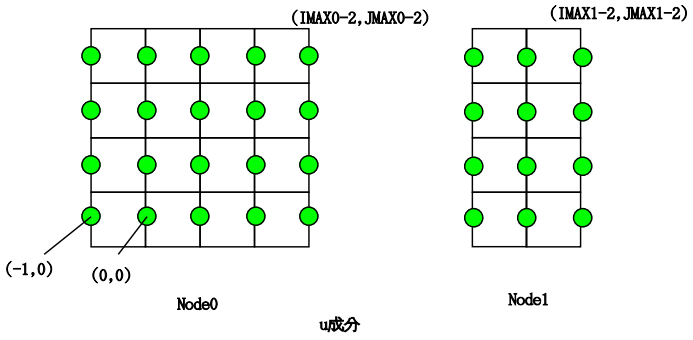
v 成分:{0,0,0}～{IMAX-2, JMAX-1, KMAX-2}

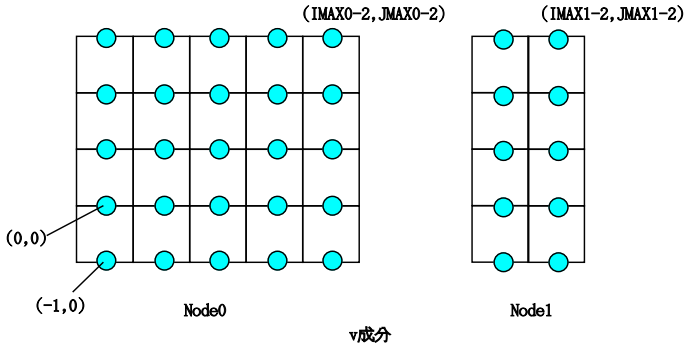
w 成分:{0,0,0}～{IMAX-2, JMAX-2, KMAX-1}

格子センタに p(圧力)データが存在する場合、p と u,v,w 成分の関係は以下となる。



並列実行においてもすべてのノードで分割ボクセルサイズとデータサイズは等しい。





計算領域全体{IMAX, JMAX, KMAX}を Node0={IMAX0,JMAX0,KMAX0}, Node1={IMAX1,JMAX1,KMAX1}で分割した場合

Node0

アロケートサイズ ={IMAX0, JMAX0, KMAX0}

領域内データ範囲 ＝

u 成分:{-1,0,0}～{IMAX0-2, JMAX0-2, KMAX0-2}

v 成分:{0,-1,0}～{IMAX0-2, JMAX0-2, KMAX0-2}

w 成分:{0,0,-1}～{IMAX0-2, JMAX0-2, KMAX0-2}

実際のデータ配列上のデータ範囲 ＝

u 成分:{0,0,0}～{IMAX0-1, JMAX0-2, KMAX0-2}

v 成分:{0,0,0}～{IMAX0-2, JMAX0-1, KMAX0-2}

w 成分:{0,0,0}～{IMAX0-2, JMAX0-2, KMAX0-1}

Node1

アロケートサイズ ={IMAX1, JMAX1, KMAX1}

領域内データ範囲 ＝

u 成分:{-1,0,0}～{IMAX1-2, JMAX1-2, KMAX1-2}

v 成分:{0,-1,0}～{IMAX1-2, JMAX1-2, KMAX1-2}

w 成分:{0,0,-1}～{IMAX1-2, JMAX1-2, KMAX1-2}

実際のデータ配列上のデータ範囲 ＝

u 成分:{0,0,0}～{IMAX1-1, JMAX1-2, KMAX1-2}

v 成分:{0,0,0}～{IMAX1-2, JMAX1-1, KMAX1-2}

w 成分:{0,0,0}～{IMAX1-2, JMAX1-2, KMAX1-1}

２．コンフィグレーション

SPHERE ではコンフィグレーションにデータ要素を定義することにより、データの生成、取得を容易にすることができます。

以下、コンフィグレーションのデータ要素について説明します。

詳細については「コンフィグレーション文法マニュアル」を参照してください。

要素名	SphDataObj	
用　途	アプリケーションで使用するデータクラスオブジェクトを記述する。 フラグによってアプリケーション起動時にフレームワーク側でインスタンスするか、ユーザが明示的にインスタンスするかを決定できる。	
属　　性		
属性名	値	種別
label	データ識別ラベル	必須
format	インスタンスするデータクラスの種類 scalar, scalar2d, scalar3d, vector2d, ...	必須
data_type	データタイプの種類 char, short, int, long, longlong, unsigned_char,　unsigned_short, unsigned_int,　unsigned_long, float, double	必須
instance	インスタンスフラグ "initialize": 起動時にフレームワーク側でインスタンスする（デフォルト） "user":ソルバ側でインスタンスする。	任意
domain	作成エリア 実行 Solver のボクセルサイズから算出 "node":分割された自ノード分の領域サイズ（デフォルト） "solver":ソルバ領域サイズ	任意
crddef	データ定義タイプ "regular"：格子上データ定義（デフォルト） "collocate"：格子中心データ定義 "staggered_1"：格子中心マイナス方向データ定義 "staggered_2"：格子中心プラスデータ定義	任意
gc	ガイドセル数（デフォルト：０）	任意
data_num	１格子当りのデータ数（デフォルト１） データクラスの種類がベクトルデータのみ有効	任意

(a) インスタンスフラグ（"instance"属性）

コンフィグレーションに記述されたデータ定義は、SPHERE のフレームワークにて自動生成されます。

しかし、ソルバー側で生成タイミング、生成の有無を判断したいときは、インスタンスフラグによりソルバー側で生成を行うことができます。

インスタンスフラグを用いた時の生成タイミング、データオブジェクトの生成、取得メソッドは以下となります。

インスタンスフラグ ("instance"属性値)	生成タイミング	生成、取得メソッド
"initialize"	ソルバ実行前	getDataObj
"user"	任意（ソルバー指定）	allocateDataObj

(b) 子要素にファイル要素の記述

子要素にファイル要素を記述することによりファイル入出力を行うことができます。詳細については「ファイル入出力マニュアル」を参照してください。

(c) **１格子当りのデータ数**：data_num 属性

１格子当りのデータ数（data_num 属性）**が定義されていない場合のデフォルト値はデータクラスの種類**（format 属性）**により異なります。**

ベクトルデータは次元数とします。

スカラデータの場合は data_num **属性を定義することはできず、１固定となります。**

No	format 属性値	**次元数**	**スカラ ベクトル**	data_num 属性 **デフォルト値**
1	"scalar1d"	1	**スカラ**	1（固定）
2	"scalar2d"	2	**スカラ**	1（固定）
3	"scalar3d"	3	**スカラ**	1（固定）
4	"vector1d"	1	**ベクトル**	1
5	"vector1drv"	1	**ベクトル**	1
6	"vector2d"	2	**ベクトル**	2
7	"vector2drv"	2	**ベクトル**	2
8	"vector3d"	3	**ベクトル**	3
9	"vector3drv"	3	**ベクトル**	3

３．ソルバーデータメソッド

SPHERE では、ソルバーにてデータの生成、取得を行う為に以下のメソッドを提供します。

No	メソッド名	説明
1	sphData* allocateDataObj(const char* cfg_label);	コンフィグレーションに定義されたデータを生成します。 生成データオブジェクトはデータマネージャに登録されます。
2	sphData* allocateDataObj(const char* reg_label, SphDCType dcType, SPL_Datatype dataType, const size_t gc = 0, const size_t data_num = 1, SphCrdDef crddef = CRDDEF_REGULAR);	指定されたデータクラス、データ型でデータを生成します。 生成データオブジェクトはデータマネージャに登録されます。
3	sphData* allocateDataObj(SphDCType dcType, SPL_Datatype dataType, const size_t gc = 0, const size_t data_num = 1, SphCrdDef crddef = CRDDEF_REGULAR);	指定されたデータクラス、データ型でデータを生成します。 生成データオブジェクトはデータマネージャに登録しません。
4	sphData* allocateDataObjExcept(const char* reg_label, SphDCType dcType, SPL_Datatype dataType, const size_t size[3], const size_t gc = 0, const size_t data_num = 1);	指定されたデータクラス、データ型、サイズでデータを生成します。 生成データオブジェクトはデータマネージャに登録されます。
5	sphData* allocateDataObjExcept(SphDCType dcType, SPL_Datatype dataType, const size_t size[3],	指定されたデータクラス、データ型、サイズでデータを生成します。 生成データオブジェクトはデータマネージャに登録しません。

	const size_t gc = 0, const size_t data_num = 1);	
6	sphData* getDataObj(const char* label);	データマネージャに登録されたデータオブジェクトを取得します。
7	bool registerDataObj(const char* label, sphData* dataObj);	データオブジェクトをデータマネージャに登録します。
8	bool deleteDataObj(const char* label);	登録ラベルを指定して、データマネージャに登録されたデータオブジェクトを削除します。
9	bool deleteDataObj(sphData* dataObj);	データオブジェクトを指定して、データマネージャに登録されたデータオブジェクトを削除します。

２．１　定義済みデータオブジェクトの生成

sphData* allocateDataObj(const char* cfg_label);	
コンフィグレーションに定義されたデータを生成します。	
引数	cfg _label　　データ識別ラベル
戻り値	生成データオブジェクト

生成データオブジェクトのデータクラスタイプ、データ型タイプ、定義位置、ガイドセル数はコンフィグレーションに定義されたパラメータにて生成を行います。
また、サイズは自ノードの分割サイズにて生成されます。

データオブジェクトの生成後、データマネージャに登録されます。この時の登録ラベルはデータ識別ラベルと同じ文字列となります。

コンフィグレーションのデータ要素のインスタンスフラグは"user"でなければなりません。

生成データオブジェクトのデータは 0x00 にて初期化されます。しかしデータ要素の配下にファイル要素が定義されている場合は、ファイルからガイドセルを含むデータ領域を読み込んでデータを設定します。(ファイルからの読込については"ファイル入出力マニュアル"を参照してください。

（使用例）

（コンフィグレーション記述例）

```
    <SphSolver id="1" label="solv01" class="solv01">
      <SphDataList>
        <SphDataObj id="1" label="pressure" format="scalar3d" data_type="double" gc="2"
                    instance="user"/>
      </SphDataList>
    </SphSolver>
```

```
// データオブジェクトの生成を行う。
sphData* array = allocateDataObj ("pressure ");
// スカラー3D, double データ型, ガイドセル数=2, 定義位置=regular にてデータ生成を行う。
```

２．２　クラスタイプ指定データオブジェクトの生成

```
sphData*
allocateDataObj(    const char* reg_label,
                    SphDCType dcType,
                    SPL_Datatype dataType,
                    const size_t gc = 0,
                    const size_t data_num = 1,
                    SphCrdDef crddef = CRDDEF_REGULAR);
```

指定されたデータクラス、データ型でデータを生成します。

引数		
	reg_label	データ登録ラベル
	dcType	データクラスタイプ
	dtType	データ型タイプ
	gc	ガイドセルサイズ
	data_num	データ数（デフォルト=1）
	crddef	データ定義位置（デフォルト= CRDDEF_REGULAR）
戻り値	生成データオブジェクト	

指定されたデータクラスタイプ、データ型タイプ、ガイドセル数、データ数、定義位置にてデータオブジェクトの生成を行います。
また、サイズは自ノードの分割サイズにて生成されます。

データオブジェクトの生成後、指定された登録ラベルにてデータマネージャに登録されます。

登録ラベルはソルバー内で一意でなければなければなりません。複数のソルバーを連成にて実行する場合は、個々のソルバー内で一意である必要があります。

（使用例）

```
// データオブジェクトの生成を行う。
sphData* array = allocateDataObj ("pressure ", SPH_DC_ARRAY_3D, SPL_FLOAT,
                                                    3, 1, CRDDEF_COLLOCATE);
// スカラー3D, float データ型, ガイドセル数=3, 定義位置=collocate にてデータ生成を行う。
```

２．３　クラスタイプ指定データオブジェクトの生成（登録なし）

sphData* allocateDataObj(SphDCType dcType, SPL_Datatype dataType, const size_t gc = 0, const size_t data_num = 1, SphCrdDef crddef = CRDDEF_REGULAR);		
指定されたデータクラス、データ型でデータを生成します。		
引数	dcType	データクラスタイプ
	dtType	データ型タイプ
	gc	ガイドセルサイズ
	data_num	データ数（デフォルト=1）
	crddef	データ定義位置（デフォルト= CRDDEF_REGULAR）
戻り値	生成データオブジェクト	

指定されたデータクラスタイプ、データ型タイプ、ガイドセル数、データ数、定義位置にてデータオブジェクトの生成を行います。
また、サイズは自ノードの分割サイズにて生成されます。

データオブジェクトの生成後、データマネージャに登録しませんので、データの管理、破棄はソルバー側で行う必要があります。

２．４　サイズ指定データオブジェクトの生成

```
sphData* allocateDataObjExcept(     const char* reg_label,
                                    SphDCType dcType,
                                    SPL_Datatype dataType,
                                    const size_t size[3],
```

		const size_t gc = 0, const size_t data_num = 1);
指定されたデータクラス、データ型、サイズでデータを生成します。		
引数	reg_label dcType dtType size gc data_num	データ登録ラベル データクラスタイプ データ型タイプ データサイズ ガイドセルサイズ データ数（デフォルト=1）
戻り値	生成データオブジェクト	

指定されたデータクラスタイプ、データ型タイプ、サイズ、ガイドセル数、データ数にてデータオブジェクトの生成を行います。

データオブジェクトの生成後、データマネージャに登録します。

生成データオブジェクトに対しては、ガイドセルの通信(comm)は行うことはできません。

また、定義位置を設定する場合は生成されたデータオブジェクトに対して設定を行ってください。

（使用例）

```
// データオブジェクトの生成を行う。
size_t size[3] = {64, 32, 32};
sphData* array = allocateDataObjExcept (
                                "pressure ", SPH_DC_ARRAY_3D, SPL_FLOAT, size, 3);
// スカラー3D, float データ型, サイズ=(64, 32, 32), ガイドセル数=3, にてデータ生成を行う。
```

２．５　サイズ指定データオブジェクトの生成（登録なし）

sphData* allocateDataObjExcept(		SphDCType dcType, SPL_Datatype dataType, const size_t size[3], const size_t gc = 0, const size_t data_num = 1);
指定されたデータクラス、データ型、サイズでデータを生成します。（登録なし）		
引数	reg_label dcType	データ登録ラベル データクラスタイプ

	dtType	データ型タイプ
	size	データサイズ
	gc	ガイドセルサイズ
	data_num	データ数（デフォルト=1）
戻り値	生成データオブジェクト	

指定されたデータクラスタイプ、データ型タイプ、サイズ、ガイドセル数、データ数にてデータオブジェクトの生成を行います。

データオブジェクトの生成後、データマネージャに登録しませんので、データの管理、破棄はソルバー側で行う必要があります。

生成データオブジェクトに対しては、ガイドセルの通信(comm)は行うことはできません。

２．６　登録済みデータオブジェクトの取得

sphData* getDataObj(const char* label);		
データマネージャに登録されたデータオブジェクトを取得します。		
引数	label	データ登録ラベル
戻り値	生成データオブジェクト	

データマネージャに登録されたデータオブジェクトを取得します。

コンフィグレーションのデータ要素のインスタンスフラグが"initialize"の場合はSPHERE フレームワーク側でソルバー実行前に自動生成を行い、データマネージャにデータ識別ラベルにて登録しますので、"getDataObj"にコンフィグレーションのデータ識別ラベルを指定することによりデータオブジェクトを取得できます。

または、ソルバー実行後、"allocateDataObj", "registerDataObj"にてデータマネージャに登録したデータオブジェクトを取得することも可能です。

（使用例）

（コンフィグレーション記述例）

```
<SphSolver id="1" label="solv01" class="solv01">
  <SphDataList>
    <SphDataObj id="1" label="pressure" format="scalar3d" data_type="double" gc="2" />
  </SphDataList>
</SphSolver>
```

```
// データオブジェクトの取得を行う。
sphData* array = getDataObj ("pressure ");
// スカラー3D, double データ型, ガイドセル数=2, 定義位置=regular にてデータ取得を行う。
```


２．７　データマネージャへの登録

bool registerDataObj(const char* label, sphData* dataObj)	
データオブジェクトをデータマネージャに登録します。	
引数	label　　データ登録ラベル dataObj　　登録データ
戻り値	生成済みデータオブジェクト

　データマネージャにデータオブジェクトを登録します。
データマネージャにデータを登録することによりデータ管理、破棄は SPHERE フレームワークが行いますので、ソルバーにてデータの破棄を行う必要はありません。
　登録ラベルはソルバー内で一意でなければなければなりません。複数のソルバーを連成にて実行する場合は、個々のソルバー内で一意である必要があります。

２．８　データマネージャからの削除（登録ラベル指定）

bool deleteDataObj(const char* label);	
データマネージャに登録されたデータオブジェクトを削除します。（登録ラベル指定）	
引数	label　　データ登録ラベル
戻り値	成否

　データマネージャ登録されているデータオブジェクトを削除します。
データオブジェクトによって確保されていたメモリ領域は開放されます。
ソルバー側で明示的にデータの開放を行いたい場合に使用します。

２．９　データマネージャからの削除（データ指定）

bool deleteDataObj(sphData* dataObj);	
データマネージャに登録されたデータオブジェクトを削除します。（データ指定）	
引数	dataObj　　データオブジェクト
戻り値	成否

　データマネージャ登録されているデータオブジェクトを削除します。
データオブジェクトによって確保されていたメモリ領域は開放されます。

ソルバー側で明示的にデータの開放を行いたい場合に使用します。

４．データクラスメソッド

生成済みデータオブジェクトに対してデータの取得、設定、生成情報の取得の為に以下のメソッドを用意しています。
メソッドを使用する場合は、データオブジェクトを呼出クラスにキャストを行う必要があります。

No	呼出クラス名	メソッド名	説明
1	sphData	std::string getLabel() const;	データラベルを取得します。
2		SphDCType getDcType() const;	データクラスタイプを取得します。
3		SPL_Datatype getDataType() const;	データ型タイプを取得します。
4		unsigned getDims() const	次元数を取得します。
5		const size_t* getArraySize() const	ガイドセルは含まないデータサイズを取得します。
6		const size_t* _getArraySize() const	ガイドセルを含んだデータサイズを取得します。
7		size_t getArrayLength() const	ガイドセルを含んだデータ長を取得します。
8		size_t getGc() const	ガイドセル数を取得します。
9		size_t getNumOfDtElem() const	１セルのデータ数を取得します。
10		const void* getDataPointer() const void* getDataPointer()	データポインタを取得します。
11		SphCrdDef getCrdDef() const;	データ定義位置を取得します。
12		bool setCrdDef(SphCrdDef crddef)	データ定義位置を設定します。
13		bool comm(); bool comm(size_t gc, int paraKey = -1)	隣接ノードとガイドセル領域の通信を行います。
14		bool commPeriodic(unsigned int face_dir, int axis); bool commPeriodic(size_t face, unsigned int face_dir, int axis, int paraKey = -1)	周期境界の通信を行います。

No	呼出クラス名	メソッド名	説明
15		const sphParaController* getParaCtrlr(int paraKey = -1) const;	データが属するパラレルコントローラを取得します。
16		bool getDataOrigin(double org[3]) const;	データの原点座標を取得します。
17		bool getDataPitch(double org[3]) const;	データのピッチを取得します。
18		bool getDataStartIdx(size_t start_idx[3]) const;	データの始点グローバルインデックスを取得します。
19	sphArray	const DATATYPE* getData() const; DATATYPE* getData();	データ型にてデータポインタを取得します。
20		const DATATYPE& val(...) const; DATATYPE& val(...);	ガイドセルを含まない領域の指定インデックス位置のデータを取得・代入します。 指定インデックスの並びはデータ配列の並びと同じです。
21		const DATATYPE& _val(...) const; DATATYPE& _val(...);	ガイドセルを含む指定インデックス位置のデータを取得・代入します。 指定インデックスの並びはデータ配列の並びと同じです。

22		const DATATYPE& getValue(
		size_t i,
		size_t j=0,
		size_t k=0,
		size_t l=0) const;
		DATATYPE& getValue(
		size_t i,
		size_t j=0,
		size_t k=0,
		size_t l=0);
			ガイドセルを含まない領域の指定インデックス位置のデータを取得・代入します。
			指定インデックスの並びは I 方向、J 方向、K 方向、データ数の並びです。

No	呼出クラス名	メソッド名	説明

23		const DATATYPE& _getValue(
		size_t i,
		size_t j=0,
		size_t k=0,
		size_t l=0) const;
		DATATYPE& _getValue(
		size_t i,
		size_t j=0,
		size_t k=0,
		size_t l=0);
			ガイドセルを含む指定インデックス位置のデータを取得・代入します。
			指定インデックスの並びはI方向、J方向、K方向、データ数の並びです。

24		size_t getIndex(size_t i, size_t j=0, size_t k=0, size_t l=0) const;	ガイドセルを含まない領域の指定インデックス位置の一次配列上のインデックスを取得します。 指定インデックスの並びはI方向、J方向、K方向、データ数の並びです。
25		size_t _getIndex(size_t i, size_t j=0, size_t k=0, size_t l=0) const;	ガイドセルを含む領域の指定インデックス位置の一次配列上のインデックスを取得します。 指定インデックスの並びはI方向、J方向、K方向、データ数の並びです。
26	sphVoxArray	size_t getWNumberOfID(DATATYPE id, int paraKey = -1) const;	ガイドセルを含まない計算領域全体から指定値の存在するデータ数を取得します。
27		size_t getNumberOfID(DATATYPE id) const;	ガイドセルを含まない自領域から指定値の存在するデータ数を取得します。
28		size_t getWNumberOfIDExtGc(DATATYPE id, int paraKey = -1) const;	ガイドセルを含む計算領域全体から指定値の存在するデータ数を取得します。
29		size_t getNumberOfIDExtGc(DATATYPE id) const;	ガイドセルを含む自領域から指定値の存在するデータ数を取得します。

No	呼出クラス名	メソッド名	説明
30		bool getBndWIndex(DATATYPE id, size_t* bnd_sta, size_t* bnd_len, const size_t ext_gc = 0, int paraKey = -1) const;	ガイドセルを含まない計算領域全体から指定値の存在する境界領域を取得します。
31		bool getBndIndex(DATATYPE id, size_t* bnd_sta, size_t* bnd_len, const size_t ext_gc = 0, int paraKey = -1) const;	ガイドセルを含まない自領域から指定値の存在する境界領域を取得します。
32		bool getBndWIndexExtGc(DATATYPE id, size_t* bnd_sta, size_t* bnd_len, const size_t ext_gc = 0, int paraKey = -1) const;	ガイドセルを含む計算領域全体から指定値の存在する境界領域を取得します。
33		bool getBndIndexExtGc(DATATYPE id, size_t* bnd_sta, size_t* bnd_len, const size_t ext_gc = 0, int paraKey = -1) const;	ガイドセルを含む自領域から指定値の存在する境界領域を取得します。
34		bool copyData(const sphVoxArray<DATATYPE>* src_dc, size_t src_vc = 0, size_t dest_vc = 0, int src_paraKey = -1, int dest_paraKey = -1);	データをコピーします。

DATATYPE：生成データオブジェクトのデータ型(char, int, unsigned, float, ...)

４．１　データラベルの取得

std::string sphData::getLabel() const;	
データラベルを取得します。	
引数	なし
戻り値	データラベル

４．２　データクラスタイプの取得

SphDCType sphData::getDcType() const;	
データクラスタイプを取得します。	
引数	なし
戻り値	データクラスタイプ

４．３　データ型タイプの取得

SPL_Datatype sphData::getDataType() const;	
データ型タイプを取得します。	
引数	なし
戻り値	データ型タイプ

４．４　次元数取得

unsigned sphData::getDims () const;	
次元数を取得します。	
引数	なし
戻り値	次元数（1, 2 or 3）

SPHERE フレームワークがサポートしている次元数は 1, 2 又は 3 です。

４．５　データサイズの取得（ガイドセルは含まない）

const size_t* sphData::getArraySize() const	
ガイドセルは含まないデータサイズを取得します。	
引数	なし
戻り値	データサイズ（ガイドセルを含まない）

このメソッドで取得するデータサイズは I,J,K 方向のガイドセルを含まないデータサイズとなります。

戻り値のデータサイズのポインタは I,J,K 方向の順番でデータサイズを返します。

戻り値[0] ＝ I 方向サイズ（ガイドセルを含まない）

戻り値[1] ＝ J 方向サイズ（ガイドセルを含まない）

戻り値[2] ＝ K 方向サイズ（ガイドセルを含まない）

この戻り値には 1 セルのデータ数は含まれていません。

４．６　データサイズの取得（ガイドセルは含む）

const size_t* sphData::_getArraySize() const	
ガイドセルを含んだデータサイズを取得します。	
引数	なし
戻り値	データサイズ（ガイドセルを含む）

このメソッドで取得するデータサイズは I,J,K 方向のガイドセルを含むデータサイズと 1 セルのデータ数となります。

戻り値のデータサイズのポインタにセットされる値の順番は、データクラスタイプによって異なります。

データクラスタイプ	戻り値インデックス（順番）			
	0	1	2	3
SPH_DC_ARRAY_2D	I 方向サイズ	J 方向サイズ	（不定）	（不定）
SPH_DC_ARRAY_2DN	I 方向サイズ	J 方向サイズ	データ数	（不定）
SPH_DC_ARRAY_2DNRV	データ数	I 方向サイズ	J 方向サイズ	（不定）
SPH_DC_ARRAY_3D	I 方向サイズ	J 方向サイズ	K 方向サイズ	（不定）
SPH_DC_ARRAY_3DN	I 方向サイズ	J 方向サイズ	K 方向サイズ	データ数
SPH_DC_ARRAY_3DNRV	データ数	I 方向サイズ	J 方向サイズ	K 方向サイズ

I,J,K 方向のサイズはガイドセルを含んだサイズです。

４．７　データ長の取得（ガイドセルは含む）

size_t sphData::getArrayLength() const	
ガイドセルを含んだデータ長を取得します。	
引数	なし
戻り値	データ長（ガイドセルを含む）

データが実際に確保しているデータ長を取得します。

このデータ長は前述の"_getArraySize()"によって取得したデータサイズを乗じた値となります。

データ長 = I 方向サイズ(GC 込み)
* J 方向サイズ(GC 込み)
* K 方向サイズ(GC 込み)
* データ数

４．８　ガイドセル数の取得

size_t sphData::getGc() const	
ガイドセル数を取得します。	
引数	なし
戻り値	ガイドセル数

４．９　データ数の取得

size_t sphData::getNumOfDtElem() const	
１セルのデータ数を取得します。	
引数	なし
戻り値	データ数

４．１０　データポインタの取得

const void* sphData::getDataPointer() const void* sphData::getDataPointer()	
データポインタを取得します。	
引数	なし
戻り値	データポインタ

４．１１　データ定義位置の取得

SphCrdDef sphData::getCrdDef() const;	
データポインタを取得します。	
引数	なし
戻り値	データポインタ

４．１２　データ定義位置の設定

bool sphData::setCrdDef(SphCrdDef crddef)	
データポインタを取得します。	
引数	crddef　　データ定義位置
戻り値	成否

４．１３　ガイドセル領域の通信

bool sphData::comm(); bool sphData::comm(size_t face, int paraKey = -1)	
ガイドセル領域の通信を行います。	
引数	face　　通信ガイドセル数 paraKey　通信ノードパラレル識別キー番号
戻り値	成否

ガイドセル領域を隣接ノードと通信を行います。
SPHERE では均等分割とマルチボックス分割の２つの分割方法をサポートしています。
分割方法によりガイドセル領域の通信方法が異なりますが、"comm"メソッドでは分割方法に合わせたガイドセル領域の通信を行います。
引数を省略した場合は、データオブジェクトに設定されているガイドセル数にて通信を行います。

４．１４　周期境界の通信

bool sphData::commPeriodic(unsigned int face_dir, int axis);

bool sphData::commPeriodic(size_t face, unsigned int face_dir, int axis, int paraKey = -1)	
周期境界の通信を行います。	
引数	face　　通信ガイドセル数 face_dir　　通信面（X 面=0, Y 面=1, Z 面=2） axis　　通信方向（マイナス方向=-1, プラス方向=+1） paraKey　　通信ノードパラレル識別キー番号
戻り値	成否

周期境界の指定通信面、方向にてガイドセルの通信を行います。
SPHERE では均等分割とマルチボックス分割の２つの分割方法をサポートしていますが、このメソッドは均等分割であるときのみサポートしています。
引数を省略した場合は、データオブジェクトに設定されているガイドセル数にて通信を行います。

４．１５　パラレルコントローラの取得

const sphData::sphParaController* getParaCtrlr(int paraKey = -1) const;	
データが属するパラレルコントローラを取得します。	
引数	paraKey　　通信ノードパラレル識別キー番号
戻り値	パラレルコントローラ

４．１６　原点座標の取得

bool sphData::getDataOrigin(double org[3]) const;	
データの原点座標を取得します。	
引数	org[out]　　原点座標
戻り値	成否

引数の原点座標は呼び出し側で作成済みである必要があります。

４．１７　ピッチの取得

bool sphData::getDataPitch(double pitch[3]) const;	
データのピッチを取得します。	
引数	pitch[out]　　ピッチ

戻り値	成否

引数のピッチは呼び出し側で作成済みである必要があります。

４．１８　始点グローバルインデックスの取得

bool sphData::getDataStartIdx(size_t start_idx[3]) const;	
データの始点グローバルインデックスを取得します。	
引数	start_idx[out]　　始点グローバルインデックス
戻り値	成否

引数の始点グローバルインデックスは呼び出し側で作成済みである必要があります。

４．１９　データ型ポインタの取得

const DATATYPE* sphArray::getData() const; DATATYPE* sphArray::getData();	
データ型にてデータポインタを取得します。	
引数	なし
戻り値	データポインタ

４．２０　データの取得・設定（ガイドセルは含まないデータ配列インデックス）

const DATATYPE& sphArray::val(size_t i) const;
DATATYPE& sphArray::val(size_t i);
const DATATYPE& sphArray::val(size_t i, size_t j) const;
DATATYPE& sphArray::val(size_t i, size_t j);
const DATATYPE& sphArray::val(size_t i, size_t j, size_t k) const;
DATATYPE& sphArray::val(size_t i, size_t j, size_t k);
const DATATYPE& sphArray::val(size_t i, size_t j, size_t k, size_t l) const;
DATATYPE& sphArray::val(size_t i, size_t j, size_t k, size_t l);	

ガイドセルを含まない領域の指定インデックス位置のデータを取得・代入します。	
引数	i　　I 方向データ配列インデックス j　　J 方向データ配列インデックス k　　k 方向データ配列インデックス l　　データ数インデックス

戻り値	データ値

指定インデックス位置のデータ値の取得、又は設定を行います。
指定インデックスはデータ配列の並び順となりますので生成データクラスタイプによって異なります。

データクラスタイプ	引数（インデックス）			
	i	j	k	l
SPH_DC_ARRAY_2D	I 方向	J 方向	（なし）	（なし）
SPH_DC_ARRAY_2DN	I 方向	J 方向	データ数 (0～n-1)	（なし）
SPH_DC_ARRAY_2DNRV	データ数 (0～n-1)	I 方向	J 方向サイズ	（なし）
SPH_DC_ARRAY_3D	I 方向	J 方向	K 方向サイズ	（なし）
SPH_DC_ARRAY_3DN	I 方向	J 方向	K 方向サイズ	データ数 (0～n-1)
SPH_DC_ARRAY_3DNRV	データ数 (0～n-1)	I 方向	J 方向	K 方向

I,J,K 方向のインデックスはガイドセルを含まないインデックスです。

４．２１　データの取得・設定（ガイドセルを含むデータ配列インデックス）

const DATATYPE& sphArray::_val(size_t i) const;
DATATYPE& sphArray::_val(size_t i);
const DATATYPE& sphArray::_val(size_t i, size_t j) const;
DATATYPE& sphArray::_val(size_t i, size_t j);
const DATATYPE& sphArray::_val(size_t i, size_t j, size_t k) const;
DATATYPE& sphArray::_val(size_t i, size_t j, size_t k);
const DATATYPE& sphArray::_val(size_t i, size_t j, size_t k, size_t l) const;
DATATYPE& sphArray::_val(size_t i, size_t j, size_t k, size_t l);	

ガイドセルを含む領域の指定インデックス位置のデータを取得・代入します。	
引数	i　I 方向データ配列インデックス j　J 方向データ配列インデックス k　k 方向データ配列インデックス l　データ数インデックス
戻り値	データ値

指定インデックス位置のデータ値の取得、又は設定を行います。
指定インデックスはデータ配列の並び順となりますので生成データクラスタイプによって異なります。

データクラスタイプ	引数（インデックス）			
	i	j	k	l
SPH_DC_ARRAY_2D	I 方向	J 方向	（なし）	（なし）
SPH_DC_ARRAY_2DN	I 方向	J 方向	データ数 (0～n-1)	（なし）
SPH_DC_ARRAY_2DNRV	データ数 (0～n-1)	I 方向	J 方向サイズ	（なし）
SPH_DC_ARRAY_3D	I 方向	J 方向	K 方向サイズ	（なし）
SPH_DC_ARRAY_3DN	I 方向	J 方向	K 方向サイズ	データ数 (0～n-1)
SPH_DC_ARRAY_3DNRV	データ数 (0～n-1)	I 方向	J 方向	K 方向

I,J,K 方向のインデックスはガイドセルを含むインデックスです。

４．２２　データの取得・設定（ガイドセルは含まないボクセルインデックス）

const DATATYPE& sphArray::getValue(size_t i, size_t j=0, size_t k=0, size_t l=0) const; DATATYPE& sphArray::getValue(size_t i, size_t j=0, size_t k=0, size_t l=0);	
ガイドセルを含まない領域の指定インデックス位置のデータを取得・代入します。	
引数	i　I 方向ボクセルインデックス j　J 方向ボクセルインデックス k　k 方向ボクセルインデックス l　データ数インデックス
戻り値	データ値

指定インデックス位置のデータ値の取得、又は設定を行います。
指定インデックス値はガイドセルを含まない領域のインデックスとなります。
引数のインデックスの指定方法はすべてのデータクラスタイプにても同じで、I 方向, J 方向, K 方向, データ数の順番となります。

４．２３　データの取得・設定（ガイドセルを含むボクセルインデックス）

const DATATYPE& sphArray::_getValue(size_t i, size_t j=0, size_t k=0, size_t l=0) const; DATATYPE& sphArray::_getValue(size_t i, size_t j=0, size_t k=0, size_t l=0);	
ガイドセルを含む領域の指定インデックス位置のデータを取得・代入します。	
引数	i　I 方向ボクセルインデックス j　J 方向ボクセルインデックス k　k 方向ボクセルインデックス l　データ数インデックス
戻り値	データ値

指定インデックス位置のデータ値の取得、又は設定を行います。
指定インデックス値はガイドセルを含む領域のインデックスとなります。
引数のインデックスの指定方法はすべてのデータクラスタイプにても同じで、I 方向, J 方向, K 方向, データ数の順番となります。

４．２４　データ配列インデックスの取得（ガイドセルは含まない）

size_t sphArray::getIndex(size_t i, size_t j=0, size_t k=0, size_t l=0) const;	
ガイドセルを含まない領域の指定インデックス位置の一次配列上のインデックスを取得します。	
引数	i　I 方向データ配列インデックス j　J 方向データ配列インデックス k　k 方向データ配列インデックス l　データ数インデックス
戻り値	一次配列インデックス

指定ボクセルインデックス位置の一次配列上のインデックスを取得します。
指定インデックス値はガイドセルを含まない領域のインデックスとなります。
引数のインデックスの指定方法はすべてのデータクラスタイプにても同じで、I 方向, J 方向, K 方向, データ数の順番となります。

４．２５　データ配列インデックスの取得（ガイドセルを含む）

size_t sphArray::_getIndex(size_t i, size_t j=0, size_t k=0, size_t l=0) const;	
ガイドセルを含む領域の指定インデックス位置の一次配列上のインデックスを取得します。	
引数	i　I 方向データ配列インデックス

	j　　J 方向データ配列インデックス k　　k 方向データ配列インデックス l　　データ数インデックス
戻り値	一次配列インデックス

指定ボクセルインデックス位置の一次配列上のインデックスを取得します。
指定インデックス値はガイドセルを含む領域のインデックスとなります。
引数のインデックスの指定方法はすべてのデータクラスタイプにても同じで、I 方向, J 方向, K 方向, データ数の順番となります。

４．２６　計算領域全体の格子点数の取得（ガイドセルは含まない）

size_t sphVoxArray::getWNumberOfID(DATATYPE id, int paraKey = -1) const;	
ガイドセルを含まない計算領域全体から指定値の存在するデータ数を取得します。	
引数	id　　データ値 paraKey　　ノードパラレル識別キー番号
戻り値	格子点数

すべてのノードのガイドセルを含まない領域から指定データ値の格子点数を集計します。

４．２７　データ領域の格子点数の取得（ガイドセルは含まない）

size_t sphVoxArray::getNumberOfID(DATATYPE id) const;	
ガイドセルを含まない自領域から指定値の存在するデータ数を取得します。	
引数	id　　データ値
戻り値	格子点数

このデータクラスオブジェクト内のガイドセルを含まない領域から指定データ値の格子点数を取得します。

４．２８　計算領域全体の格子点数の取得（ガイドセルを含む）

size_t sphVoxArray::getWNumberOfIDExtGc(DATATYPE id, int paraKey = -1) const;	
ガイドセルを含む計算領域全体から指定値の存在するデータ数を取得します。	
引数	id　　データ値 paraKey　　ノードパラレル識別キー番号

戻り値	格子点数

すべてのノードのガイドセルを含む領域から指定データ値の格子点数を集計します。

４．２９　データ領域の格子点数の取得（ガイドセルを含む）

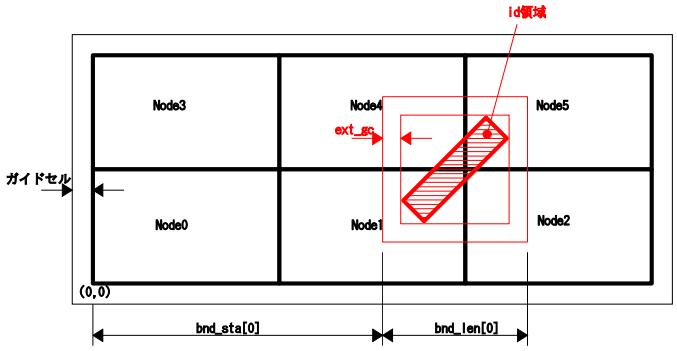
size_t sphVoxArray::getNumberOfID(DATATYPE id) const;	
ガイドセルを含む自領域から指定値の存在するデータ数を取得します。	
引数	id　　データ値
戻り値	格子点数

このデータクラスオブジェクト内のガイドセルを含む領域から指定データ値の格子点数を取得します。

４．３０　計算領域全体の境界領域の取得（ガイドセルは含まない）

bool sphVoxArray::getBndWIndex(DATATYPE id, size_t* bnd_sta, size_t* bnd_len, const size_t ext_gc = 0, int paraKey = -1) const	
ガイドセルを含まない計算領域全体から指定値の存在する境界領域を取得します。	
引数	id　　データ値 bnd_sta[out]　　始点グローバルインデックス bnd_len[out]　　境界領域サイズ ext_gc　　拡張ガイドセル paraKey　　ノードパラレル識別キー番号
戻り値	成否

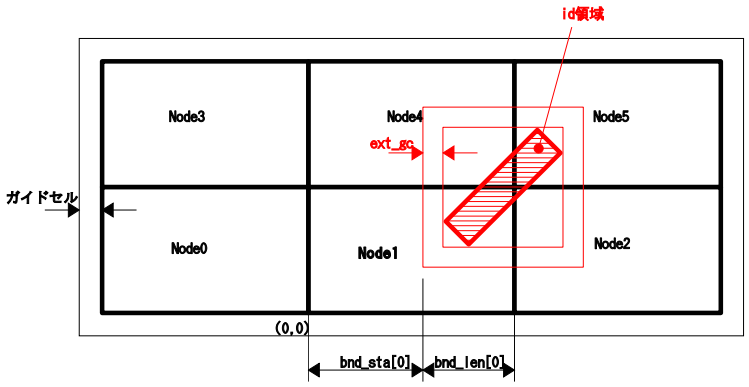
すべてのノードのガイドセルを含まない領域から指定データ値の境界領域を取得します。



４．３１　データ領域の境界領域の取得（ガイドセルは含まない）

bool sphVoxArray::getBndIndex(DATATYPE id, size_t* bnd_sta, size_t* bnd_len, const size_t ext_gc = 0, int paraKey = -1) const		
ガイドセルを含まない自領域から指定値の存在する境界領域を取得します。		
引数	id	データ値
	bnd_sta[out]	始点ローカルインデックス
	bnd_len[out]	境界領域サイズ
	ext_gc	拡張ガイドセル
	paraKey	ノードパラレル識別キー番号
戻り値	成否	

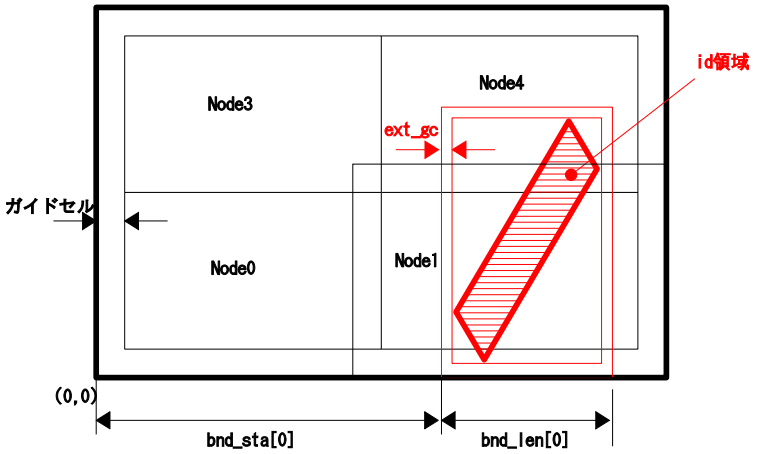
自ノードのガイドセルを含まない領域から指定データ値の境界領域を取得します。



４．３２　計算領域全体の境界領域の取得（ガイドセルを含む）

bool sphVoxArray::getBndWIndexExtGc(DATATYPE id, size_t* bnd_sta, size_t* bnd_len, const size_t ext_gc = 0, int paraKey = -1) const		
ガイドセルを含む計算領域全体から指定値の存在する境界領域を取得します。		
引数	id	データ値
	bnd_sta[out]	始点グローバルインデックス
	bnd_len[out]	境界領域サイズ
	ext_gc	拡張ガイドセル
	paraKey	ノードパラレル識別キー番号
戻り値	成否	

すべてのノードのガイドセルを含む領域から指定データ値の境界領域を取得します。

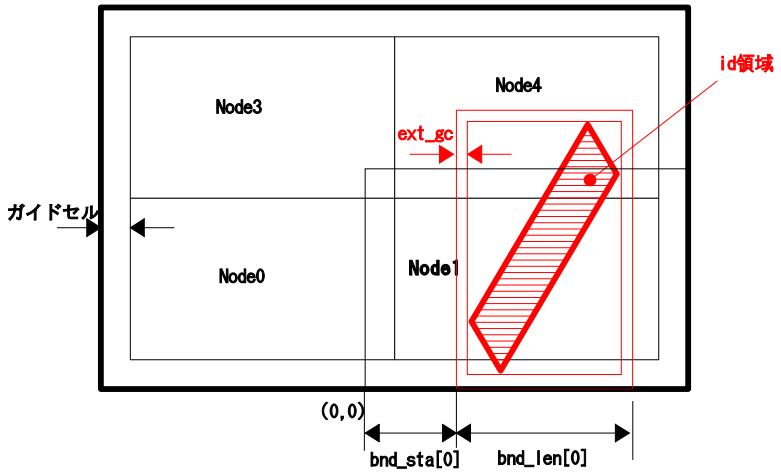


４．３３　データ領域の境界領域の取得（ガイドセルを含む）

bool sphVoxArray::getBndIndexExtGc(DATATYPE id, size_t* bnd_sta, size_t* bnd_len, const size_t ext_gc = 0, int paraKey = -1) const
ガイドセルを含む自領域から指定値の存在する境界領域を取得します。

引数	id	データ値
	bnd_sta[out]	始点ローカルインデックス
	bnd_len[out]	境界領域サイズ
	ext_gc	拡張ガイドセル
	paraKey	ノードパラレル識別キー番号
戻り値	成否	

自ノードのガイドセルを含む領域から指定データ値の境界領域を取得します。

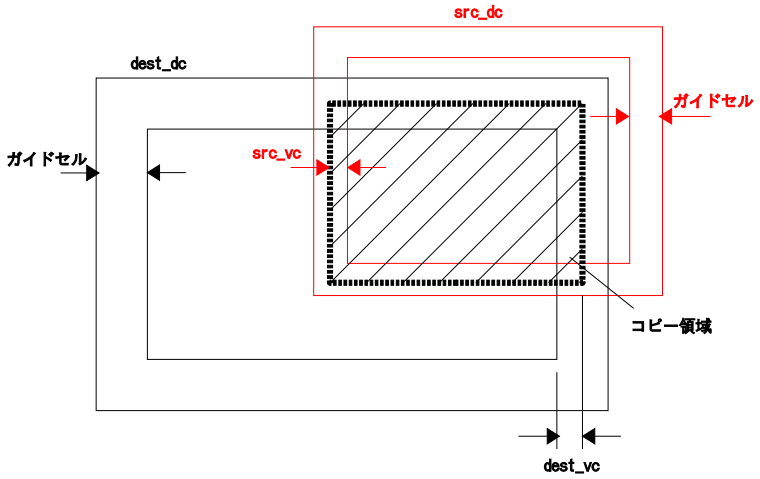


４．３４　データコピー

bool copyData(const sphVoxArray<DATATYPE>* src_dc, size_t src_vc = 0, size_t dest_vc = 0, int src_paraKey = -1, int dest_paraKey = -1);		
データをコピーします。		
引数	src_dc	コピー元データ
	src_vc	コピー元ガイドセル
	dest_vc	コピー先ガイドセル
	src_paraKey	コピー元ノードパラレル識別キー番号
	dest_paraKey	コピー先ノードパラレル識別キー番号
戻り値	成否	

コピー元データオブジェクトから重なっているインデックス位置のデータをコピーします。
コピー元ガイドセルはコピー元データのガイドセルからコピーするガイドセル範囲を指定

します。よってコピー元ガイドセルはコピー元データのガイドセルの値より大きな値を指定することはできません。
コピー先ガイドセルも同様で自データのガイドセルより大きな値を指定することはできません。
コピー元データの始点グローバルインデックスとサイズとコピー先（自データ）の始点グローバルインデックスとサイズを比較して重なっている領域のコピーを行います。
重なっていない領域はコピーしません。また別ノードからデータを取得することもしません。コピー元データからのみデータをコピーします。



５．データユーティリティー

sphDataUtils クラスはデータ操作に必要なインデックスの計算、データコピー用のメソッドを用意しています。

No	機能	メソッド名	説明
1	インデックス計算	size_t getCindexS2D(sz, gc, i, j);	ガイドセルを含むi,j,k,lインデックスからデータ配列のインデックスを算出する。
		size_t getCindexS3D(sz, gc, i, j, k);	
		size_t getCindexV2D(sz, i, j, l);	
		size_t getCindexV2DEx(sz, gc, l, i, j);	
		size_t getCindexV3D(sz, gc, i, j, k, l);	
		size_t getCindexV3DEx(sz, gc, i, j, k, l);	
2		size_t getFindexS2D(sz, gc, i, j);	ガイドセルを含まないi,j,k,lインデックスからデータ配列のインデックスを算
		size_t getFindexS3D(sz, gc, i, j, k);	

No	機能	メソッド名	説明
		size_t getFindexV2D(sz, gc, i, j, l);	出する。
		size_t getFindexV2DEx(sz, gc, l, i, j);	
		size_t getFindexV3D(sz, gc, i, j, k, l);	
		size_t getFindexV3DEx(sz, gc, l, i, j, k);	
3	積算	bool avrS2D(dst, src)	積算元データを積算先データに積算する。
		bool avrV2D(dst, src)	
		bool avrS3D(dst, src)	
		bool avrV3D(dst, src)	
4	乗算・商算	bool scaleAvrS2D(src, stepAvr, mode);	stepAvr 値の乗算/商算を行う。
		bool scaleAvrV2D(src, stepAvr, mode);	
		bool scaleAvrS3D(src, stepAvr, mode);	
		bool scaleAvrV3D(src, stepAvr, mode);	
5	乗加算	bool shiftVin2D(dst, src, v00[2], stepAvr);	データに stepAvr 値の乗算,v00 の加算を行う。
		bool shiftVin3D(dst, src, v00[2], stepAvr);	
6	商減算	bool shiftVout2D(dst, src, v00[2], stepAvr);	データに stepAvr 値の商算,v00 の減算を行う。
		bool shiftVout3D(dst, src, v00[2], stepAvr);	
7	データコピー	bool cpyS2D(dst, src)	データのコピーを行う。
		bool cpyS3D(dst, src)	
		bool cpyS4D(dst, src)	
		bool cpyS4DEx(dst, src)	
8	成分データコピー	bool cpyS4D(dst, dst_pos, src, src_pos)	成分データのコピーを行う。
		bool cpyS4DEx(dst, dst_pos, src, src_pos)	
9	スカラ・ベクトル成分コピー	bool cpyS4D(dst, src, pos)	スカラ・ベクトルデータの成分データコピーを行う。
		bool cpyS4DEx(dst, src, pos)	

No	機能	メソッド名	説明
10	中央コピー	bool cpyS2DCenter(dst, src)	データを中央部分にコピーする。
		bool cpyS3DCenter(dst, src)	
11	Ex 変換	bool convS4D(dst, src)	Scalar4D データと Scalar4DEx の相互変換

５．１　インデックス計算（ガイドセルを含む）

size_t getCindexS2D　(const size_t* sz, size_t gc, size_t i, size_t j);
size_t getCindexS3D　(const size_t* sz, size_t gc, size_t i, size_t j, size_t k);
size_t getCindexV2D　(const size_t* sz, size_t gc, size_t i, size_t j, size_t l);
size_t getCindexV2DEx(const size_t* sz, size_t gc, size_t l, size_t i, size_t j);
size_t getCindexV3D　(const size_t* sz, size_t gc, size_t i, size_t j, size_t k, size_t l);
size_t getCindexV3DEx(const size_t* sz, size_t gc, size_t l, size_t i, size_t j, size_t k);		

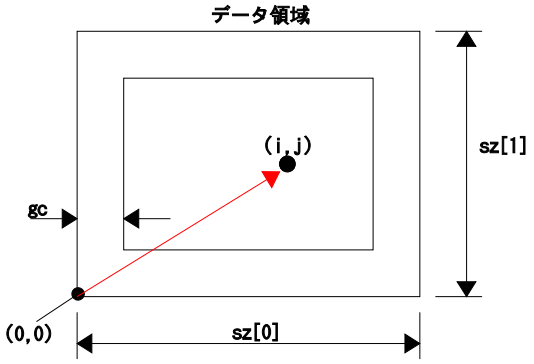
ガイドセルを含む i,j,k,l インデックスからデータ配列のインデックスを算出する。		
引数	sz	データサイズ
	gc	ガイドセル
	i	I 方向インデックス（ガイドセルを含む）
	j	J 方向インデックス（ガイドセルを含む）
	k	K 方向インデックス（ガイドセルを含む）
	l	データ数インデックス
戻り値	一次配列インデックス	

ボクセルのインデックス(i, j, k)から一次配列のインデックス値を計算します。

計算式は以下となります。

戻り値 = (sz[0]+gc*2)*(sz[1]+gc*2)*k + (sz[0]+gc*2)*j + i

I,J,K 方向インデックスはガイドセルを含んだインデックスです。



５．２　インデックス計算（ガイドセルを含まない）

size_t getFindexS2D　(const size_t* sz, size_t gc, size_t i, size_t j);
size_t getFindexS3D　(const size_t* sz, size_t gc, size_t i, size_t j, size_t k);
size_t getFindexV2D　(const size_t* sz, size_t gc, size_t i, size_t j, size_t l);
size_t getFindexV2DEx(const size_t* sz, size_t gc, size_t l, size_t i, size_t j);
size_t getFindexV3D　(const size_t* sz, size_t gc, size_t i, size_t j, size_t k, size_t l);
size_t getFindexV3DEx(const size_t* sz, size_t gc, size_t l, size_t i, size_t j, size_t k);		

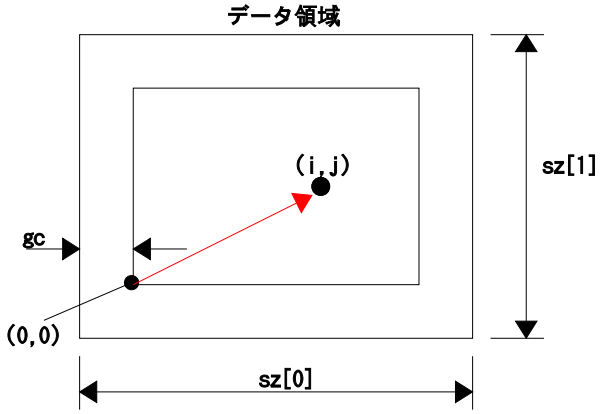
ガイドセルを含まない i,j,k,l インデックスからデータ配列のインデックスを算出する。		
引数	sz	データサイズ
	gc	ガイドセル
	i	I 方向インデックス（ガイドセルを含まない）
	j	J 方向インデックス（ガイドセルを含まない）
	k	K 方向インデックス（ガイドセルを含まない）
	l	データ数インデックス
戻り値	一次配列インデックス	

ボクセルのインデックス(i, j, k)から一次配列のインデックス値を計算します。

計算式は以下となります。

戻り値 = (sz[0]+gc*2)*(sz[1]+gc*2)*(k+gc-1) + (sz[0]+gc*2)*(j+gc-1) + i+gc-1

I,J,K 方向インデックスはガイドセルを含まないインデックスです。



５．３　積算

bool avrS2D(sphVoxArray<float>* dst, const sphVoxArray<float>* src); bool avrV2D(sphVoxArray<float>* dst, const sphVoxArray<float>* src);

bool avrS2D(sphVoxArray<double>* dst, const sphVoxArray<double>* src);
bool avrV2D(sphVoxArray<double>* dst, const sphVoxArray<double>* src);
bool avrS3D(sphVoxArray<float>* dst, const sphVoxArray<float>* src);
bool avrV3D(sphVoxArray<float>* dst, const sphVoxArray<float>* src);
bool avrS3D (sphVoxArray<double>* dst, const sphVoxArray<double>* src);
bool avrV3D(sphVoxArray<double>* dst, const sphVoxArray<double>* src);		

積算元データを積算先データに積算する。		
引数	dst	積算先データ
	src	積算元データ
戻り値	成否	

積算元データを積算先データに積算します。平均値の計算に利用します。
計算式は以下となります。

dst[index] = dst[index] + src[index]

index：データクラスが異なっても同一 I,J,K 成分

SPH_DC_ARRAY_3DN

index = sz[0]*sz[1]*sz[2]*l + sz[0]*sz[1]*k + sz[0]*j + i

SPH_DC_ARRAY_3DNRV

index = data_num*sz[0]*sz[1]*k + data_num*sz[0]*j + data_num*i + l

２つのデータの始点グローバルインデックスは同じものとして積算を行います。
ガイドセル数は異る場合は、重なっている領域のデータ積算を行います。

５．４　乗算・商算

bool scaleAvrS2D (sphVoxArray<float>* src, unsigned stepAvr = 1, bool mode = false);
bool scaleAvrS2D (sphVoxArray<double>* src, unsigned stepAvr = 1, bool mode = false);
bool scaleAvrV2D (sphVoxArray<float>* vec, unsigned stepAvr = 1, bool mode = false);
bool scaleAvrV2D (sphVoxArray<double>* vec, unsigned stepAvr = 1, bool mode = false);
bool scaleAvrS3D (sphVoxArray<float>* src, unsigned stepAvr = 1, bool mode = false);
bool scaleAvrV3D (sphVoxArray<double>* vec, unsigned stepAvr = 1, bool mode = false);
bool scaleAvrV3D (sphVoxArray<float>* vec, unsigned stepAvr = 1, bool mode = false);
bool scaleAvrV3D (sphVoxArray<double>* vec, unsigned stepAvr = 1, bool mode = false);		

stepAvr 値の乗算/商算を行う。		
引数	src	計算データ

	stepAvr	乗算・商算値
	mode	true:乗算、false:商算
戻り値	成否	

stepAvr を計算データに乗算・商算します。

計算式は以下となります。

mode = true

src[index] = src[index] * stepAvr

mode = false

src[index] = src[index] / stepAvr

index：データクラスが異なっても同一 I,J,K 成分

SPH_DC_ARRAY_3DN

index = sz[0]*sz[1]*sz[2]*l + sz[0]*sz[1]*k + sz[0]*j + i

SPH_DC_ARRAY_3DNRV

index = data_num*sz[0]*sz[1]*k + data_num*sz[0]*j + data_num*i + l

２つのデータの始点グローバルインデックスは同じものとして計算を行います。

ガイドセル数は異る場合は、重なっている領域のデータ計算を行います。

５．５　乗加算

```
bool shiftVin2D(
                sphVoxArray<float>* dst,
                const sphVoxArray<float>* src,
                double v00[2],
                unsigned stepAvr = 1);
bool shiftVin2D (
                sphVoxArray<double>* dst,
                const sphVoxArray<double>* src,
                double v00[2],
                unsigned stepAvr = 1);
bool shiftVin3D (
                sphVoxArray<float>* dst,
                const sphVoxArray<float>* src,
                double v00[3],
                unsigned stepAvr = 1);
```

bool shiftVin3D (sphVoxArray<double>* dst, const sphVoxArray<double>* src, double v00[3], unsigned stepAvr = 1);		
stepAvr 値の乗算/商算を行う。		
引数	dst	計算結果データ
	src	計算元データ
	v00	速度成分
	stepAvr	乗算値
戻り値	成否	

src のデータにある速度成分 v00[3]を加え、stepAvr を乗じ、dst にコピーします。
計算式は以下となります。

I 成分：dst[index] = (src[index] + v00[0]) * stepAvr
J 成分：dst[index] = (src[index] + v00[1]) * stepAvr
K 成分：dst[index] = (src[index] + v00[2]) * stepAvr

index：データクラスが異なっても同一 I,J,K 成分

SPH_DC_ARRAY_3DN

index = sz[0]*sz[1]*sz[2]*l + sz[0]*sz[1]*k + sz[0]*j + i

SPH_DC_ARRAY_3DNRV

index = data_num*sz[0]*sz[1]*k + data_num*sz[0]*j + data_num*i + l

２つのデータの始点グローバルインデックスは同じものとして計算を行います。
ガイドセル数は異る場合は、重なっている領域のデータ計算を行います。

５．６　商減算

bool shiftVout2D(sphVoxArray<float>* dst, const sphVoxArray<float>* src, double v00[2], unsigned stepAvr = 1); bool shiftVout2D (sphVoxArray<double>* dst, const sphVoxArray<double>* src,

double v00[2], unsigned stepAvr = 1); bool shiftVout3D (sphVoxArray<float>* dst, const sphVoxArray<float>* src, double v00[3], unsigned stepAvr = 1); bool shiftVout3D (sphVoxArray<double>* dst, const sphVoxArray<double>* src, double v00[3], unsigned stepAvr = 1);		
stepAvr 値の乗算/商算を行う。		
引数	dst	計算結果データ
	src	計算元データ
	v00	速度成分
	stepAvr	乗算値
戻り値	成否	

src のデータに stepAvr を商じ、速度成分 v00[3]を減じ、dst にコピーします。
計算式は以下となります。

I 成分：dst[index] = src[index] / stepAvr - v00[0]

J 成分：dst[index] = src[index] / stepAvr - v00[1]

K 成分：dst[index] = src[index] / stepAvr - v00[2]

２つのデータの始点グローバルインデックスは同じものとして計算を行います。
ガイドセル数は異る場合は、重なっている領域のデータ計算を行います。

５．７　データコピー

bool cpyS2D (sphVoxArray<int>* dst, const sphVoxArray<int>* src); bool cpyS2D (sphVoxArray<unsigned int>* dst, const sphVoxArray<unsigned int>* src); bool cpyS2D (sphVoxArray<float>* dst, const sphVoxArray<float>* src); bool cpyS2D (sphVoxArray<double>* dst, const sphVoxArray<double>* src); bool cpyS3D (sphVoxArray<int>* dst, const sphVoxArray<int>* src); bool cpyS3D (sphVoxArray<unsigned int>* dst, const sphVoxArray<unsigned int>* src); bool cpyS3D (sphVoxArray<float>* dst, const sphVoxArray<float>* src); bool cpyS3D (sphVoxArray<double>* dst, const sphVoxArray<double>* src);

bool cpyS4D (sphVoxArray<int>* dst, const sphVoxArray<int>* src); bool cpyS4DEx (sphVoxArray<int>* dst, const sphVoxArray<int>* src); bool cpyS4D (sphVoxArray<float>* dst, const sphVoxArray<float>* src); bool cpyS4DEx (sphVoxArray<float>* dst, const sphVoxArray<float>* src); bool cpyS4D (sphVoxArray<double>* dst, const sphVoxArray<double>* src); bool cpyS4DEx (sphVoxArray<double>* dst, const sphVoxArray<double>* src);		
データコピーを行う。		
引数	dst	コピー先データ
	src	コピー元データ
戻り値	成否	

コピー元データからコピー先データにデータコピーを行います。

２つのデータの始点グローバルインデックスは同じものとしてコピーを行います。

ガイドセル数は異る場合は、重なっている領域のデータコピーを行います。

５．８　成分データコピー

```
bool cpyS4D (
        sphVoxArray<int>* dst, size_t dst_pos,
        const sphVoxArray<int>* src, size_t src_pos);
bool cpyS4DEx (
        sphVoxArray<int>* dst, size_t dst_pos,
        const sphVoxArray<int>* src, size_t src_pos);
bool cpyS4D (
        sphVoxArray<float>* dst, size_t dst_pos,
        const sphVoxArray<float>* src, size_t src_pos);
bool cpyS4DEx (
        sphVoxArray<float>* dst, size_t dst_pos,
        const sphVoxArray<float>* src, size_t src_pos);
bool cpyS4D (
        sphVoxArray<double>* dst, size_t dst_pos,
        const sphVoxArray<double>* src, size_t src_pos);
bool cpyS4DEx (
        sphVoxArray<double>* dst, size_t dst_pos,
        const sphVoxArray<double>* src, size_t src_pos);
```

成分データコピーを行う。

引数	dst	コピー先データ
	src	コピー元データ
	dst_pos	コピー先成分インデックス(0～データ数-1)
	src_pos	コピー元成分インデックス(0～データ数-1)
戻り値	成否	

コピー元ベクトルデータからコピー先ベクトルデータに指定データ成分のデータコピーを行います。

５．９　スカラ・ベクトル成分データコピー

```
bool cpyS4D (
        sphVoxArray<int>* dst,
        const sphVoxArray<int>* src, size_t pos);
bool cpyS4DEx (
        sphVoxArray<int>* dst,
        const sphVoxArray<int>* src, size_t pos);
bool cpyS4D (
        sphVoxArray<float>* dst,
        const sphVoxArray<float>* src, size_t pos);
bool cpyS4DEx (
        sphVoxArray<float>* dst,
        const sphVoxArray<float>* src, size_t pos);
bool cpyS4D (
        sphVoxArray<double>* dst,
        const sphVoxArray<double>* src, size_t pos);
bool cpyS4DEx (
        sphVoxArray<double>* dst,
        const sphVoxArray<double>* src, size_t pos);
```

成分データコピーを行う。		
引数	dst	コピー先データ
	src	コピー元データ
	pos	コピーベクトル成分インデックス(0～データ数-1)
戻り値	成否	

コピー元データからコピー先データに指定データ成分のデータコピーを行います。
コピー元データ、コピー先データはスカラーデータとベクトルの組み合わせである必要が

あります。
pos はベクトルデータの成分インデックスを指定します。

dst: ベクトルデータ <- src:スカラーデータ

スカラーデータをベクトルデータの pos 成分位置にコピーします。

dst: スカラデータ <- src:ベクトルデータ

ベクトルデータの pos 成分データをスカラーデータにコピーします。

５．１０中央コピー

bool cpyS2DCenter (sphVoxArray<int>* dst, const sphVoxArray<int>* src); bool cpyS2DCenter (sphVoxArray<unsigned int>* dst, const sphVoxArray<unsigned int>* src); bool cpyS2DCenter (sphVoxArray<float>* dst, const sphVoxArray<float>* src); bool cpyS2DCenter (sphVoxArray<double>* dst, const sphVoxArray<double>* src); bool cpyS3DCenter (sphVoxArray<int>* dst, const sphVoxArray<int>* src); bool cpyS3DCenter (sphVoxArray<unsigned int>* dst, const sphVoxArray<unsigned int>* src); bool cpyS3DCenter (sphVoxArray<float>* dst, const sphVoxArray<float>* src); bool cpyS3DCenter (sphVoxArray<double>* dst, const sphVoxArray<double>* src);		
データの中央コピーを行う。		
引数	dst	コピー先データ
	src	コピー元データ
戻り値	成否	

コピー元データからコピー先データにデータコピーを行います。
データは内部の本来の配列長（ガイドセルを含んだ配列長さ）に対して、データの中心が揃うようコピーされます

５．１１Ex 変換

bool convS4D (sphVoxArray<int>* dst, const sphVoxArray<int>* src); bool convS4D (sphVoxArray<float>* dst, const sphVoxArray<float>* src); bool convS4D (sphVoxArray<double>* dst, const sphVoxArray<double>* src);		
ベクトルデータのデータ並びを変換します。		
引数	dst	コピー先データ
	src	コピー元データ
戻り値	成否	

ベクトルデータのデータクラスタイプを相互に変換します。

コピー元／先データ	変換	コピー元／先データ
SPH_DC_ARRAY_2DN	<-->	SPH_DC_ARRAY_2DNRV
SPH_DC_ARRAY_3DN	<-->	SPH_DC_ARRAY_3DNRV