Sphere

Skeleton for PHysical and Engineering REsearch

Ver 2.0.0

ファイル入出力マニュアル

2010 年 8 月 01 日

目次

１．概要

SPHERE ではソルバープログラム中でファイル入出力を行うためのメソッドを用意しています。現バージョンでサポートしているファイルフォーマットは以下です。

(1) SPHERE データファイルフォーマット（以下、SPH という）
(2) SBX データファイルフォーマット（以下、SBX という）
(3) SPX データファイルフォーマット（以下、SBX という）
(4) TEXT データファイルフォーマット（以下、TEXT という）

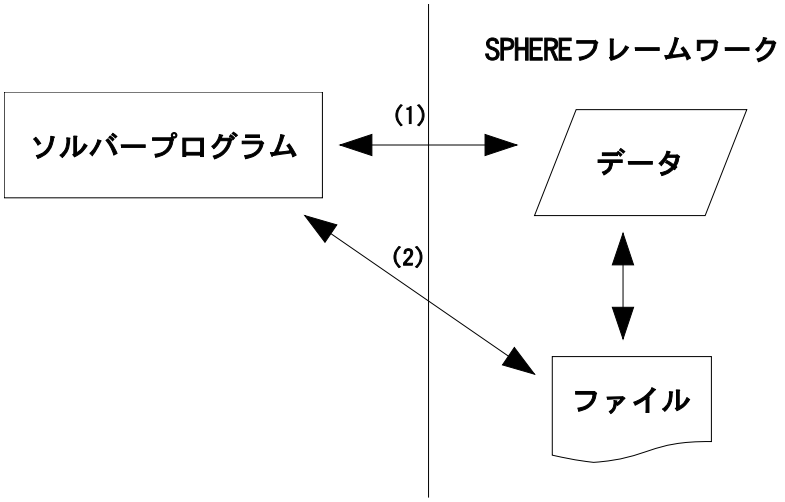
No	フォーマット	データ型	スカラ ベクトル	使用目的
1	SPH	float, double	スカラ ベクトル	計算結果の入出力を行う。 入出力のデータ型は実数(float, double)である。
2	SBX	unsigned char unsigned short unsigned int	スカラ	媒質 ID 等の設定項目の入力に用いる。 １セルデータに上位ビット、下位ビットへデータを設定可能である。
3	SPX	float, double	スカラ ベクトル	計算結果の入出力を行う。 入出力のデータ型は実数(float, double)である。
4	TEXT	char, string	なし	ヒストリ、ログ等のテキストファイルの出力を行う。

ファイル入出力を行うためには、最初にコンフィグレーションファイルに入出力の制御を記述しなければなりません。

入出力機能の基本的パラメータは入出力メソッドで指定可能ですが、コンフィグレーションファイルに記述しなければ動作しない機能があります。

以降では、コンフィグレーションへ定義、及び入出力に及ぼす影響についても説明しますが、これらはコンフィグレーションマニュアルと説明が重複しますのでコンフィグレーションマニュアルも参照してください。

ソルバーからファイルへアクセスするには以下の２つの方法があります。



(1) コンフィグレーションにデータ要素とそれに関連付けられたファイル要素を定義してデータオブジェクトを介してファイルの入出力を行う。
この場合、ソルバープログラムはデータ（データ識別ラベル）のみ管理を行えばよく、ファイルの入出力は SPHERE フレームワークが行います。

(2) コンフィグレーションにファイル要素を定義して直接ファイルの入出力を行う。
この場合、、ソルバープログラムはデータ（データ識別ラベル）とファイル（ファイル識別ラベル）の管理を行う必要があります。
SPHERE フレームワークは、データとファイルを関連付け入出力を行います。

以下のファイル入出力メソッドがソルバープログラムに提供されています。

No	入出力方向	メソッド名	説明
1	入力	bool loadFile(const char* file_label, sphData* data);	コンフィグレーションのファイル識別ラベルを指定してファイルから読込を行います。
2	入力	sphData* loadFile(const char* file_label, const size_t size[3], const size_t start_idx[3], size_t gc);	コンフィグレーションのファイル識別ラベルと読込サイズを指定してファイルから読込を行います。
3	入力	sphData* getDataObj(const char* cfg_label);	コンフィグレーションのデータ識別ラベルを指定してデータ要素に関連付けられたファイルから読込

No	入出力方向	メソッド名	説明
		sphData* allocateDataObj(const char* cfg_label);	関連付けられたファイルから読込を行います。
4	出力	bool writeFile(const char* file_label, sphData* data);	データオブジェクトをコンフィグレーションのファイル識別ラベルのファイル要素定義に従ってファイルの書込みを行います。
5	出力	bool writeFile(const char* file_label, sphData* dataObj, size_t step, double time, bool gc_flag = false, bool force = false);	データオブジェクトをコンフィグレーションのファイル識別ラベルのファイル要素定義に従ってファイルの書込みを行います。 step, time, gc_flag, force は引数で設定された値、設定にて出力します。
6	出力	bool writeFile(const char* file_label, const char* data_reg_label);	データ登録ラベルのデータオブジェクトをコンフィグレーションのファイル識別ラベルのファイル要素定義に従ってファイルの書込みを行います。
7	出力	bool writeFile(const char* data_cfg_label);	コンフィグレーションのデータ識別ラベルにて生成されたデータオブジェクトをデータ要素に関連付けられたファイル要素定義に従ってファイルの書込みを行います。
8	出力	bool writeFile(const char* data_cfg_label, size_t step, double time, bool gc_flag = false, bool force = false);	コンフィグレーションのデータ識別ラベルにて生成されたデータオブジェクトをデータ要素に関連付けられたファイル要素定義に従ってファイルの書込みを行います。 step, time, gc_flag, force は引数で設定された値、設定にて出力します。
10	ヘッダー取得	bool getFileHeader(const char* file_label, sphFileHeader* header);	ファイル識別ラベルに定義されているファイルのヘッダー情報を取得します。

No	入出力方向	メソッド名	説明
11	テキスト出力	bool getTextFile(const char* file_label, sphTextFile* text_file);	コンフィグレーションのファイル識別ラベルのファイル要素定義に従ってテキストファイルクラスを初期化します。 テキスト出力はテキストファイルクラスメソッドを介して行います。

２．ファイル入力

ファイルからの読込を行う為にはコンフィグレーションにファイル要素を定義しなければなりません。
最初に、ファイル要素のファイル入力関係の要素、属性、設定値について説明します。
以下に記述していない属性、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。

２．１　コンフィグレーションの説明

入力ファイルの指定を行うコンフィグレーション要素について説明します。

(1) SphFile 要素（入出力ファイル指定）

要素名	SphFile	
用　途	入力ファイルの指定に使用する	
属　　性		
属性名	値	種別
label (id)	識別ラベル（識別番号）	必須
format	ファイルフォーマット "sbx"：SBX ファイル "spx"：SPX ファイル "sph"：SPH ファイル	必須
file_name	ファイル名（拡張子付き） "SphFileNameFormat"要素記述の場合は必要ない。	任意
type	"in"（入力ファイル）固定	必須
mode	入出力モード multi：各ノードが読込を行う。	任意

	cent：指定ノードが読込を行う。 デフォルト= "multi"	
io_nodeid	ファイルの読み込みを行うノード番号 入出力モード（mode）が"cent"の時のみ有効。 デフォルト＝０	任意

(a) label, id 属性は両方、又はどちらかの定義が可能です。
しかし、ファイルラベル指定"loadFile"メソッドを使用する場合は label 属性が必須ですので、label 属性の定義を推奨します。

(b) 入力ファイル名
入力ファイル名は file_name 属性又は SphFileNameFormat 要素にて行います。
両方記述されている場合は SphFileNameFormat 要素が優先されます。

(2) SphFileNameFormat 要素（入出力ファイル名の書式情報）

要素名	SphFileNameFormat	
用　途	入出力ファイル名をランク番号とソルバ実行ステップ数から生成する書式を設定する。	
属　　性		
属性名	値	種別
value	出力値種別 作成するファイルに出力フォーマットに従い付加する値、 "rank"　　ランク番号 "step"　　ステップ数 "step_rank"　ステップ数＋ランク番号 "rank_step"　ランク番号＋ステップ数	必須
format	入出力ファイル名書式 C 言語"printf"文の書式に準じる。	必須

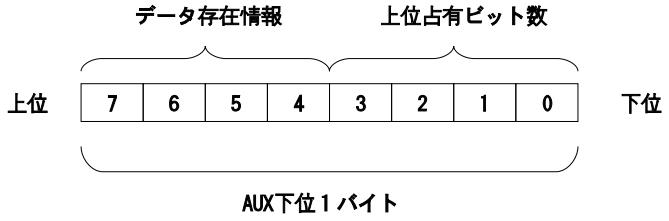
(3) SphFileOption 要素（ファイル入出力オプション情報）

要素名	SphFileOption	
用　途	ファイルの入出力のオプション情報を設定する。	
属　　性		
属性名	値	種別

sbx_bitshandling	SBX ファイル読み込み後のビット操作を行う。 "maskupper":上位データをマスクし、下位ビットのみのデータとする。 "shiftlow":下位ビットをシフトし、上位ビットのみのデータとする。	任意 (入力)
gc_insph	SPH ファイルのファイルに内包するガイドセル値 デフォルト= 0	任意 (入力)

(a) SBX ビット操作

sbx_bitshandling 属性の設定により SBX ファイル読み込み後、ビット操作を行いデータオブジェクトを生成します。
ビット操作は SBX ファイルヘッダー内の AUX レコードのの下位１バイト値によって決定されます。



AUX の下位１バイトの記述仕様

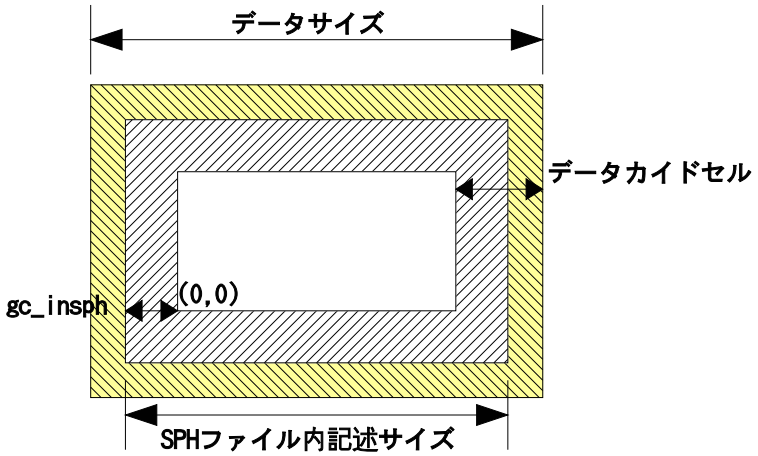
下位４ビット	上位占有ビット数（n）
上位４ビット	データ存在情報（ ０ or １ or ２ ） ０：上位・下位有効データ １：下位有効データ ２：上位有効データ

sbx_bitshandling 属性の設定"maskupper", "shiftlow"によるビット操作と AUX の記述は以下となります。

データ存在情報	"maskupper"	"shiftlow"
０	上位 n ビットをマスク(0 埋め)します	下位ビット(8-n)をシフトします。
１	上位 n ビットをマスク(0 埋め)します	上位下位 0x00 とします。
２	上位下位 0x00 とします。	下位ビット(8-n)をシフトします。

ビット操作を行うことができるのは"unsigned char 型"のみです。

(b) SPH ファイルのガイドセル値

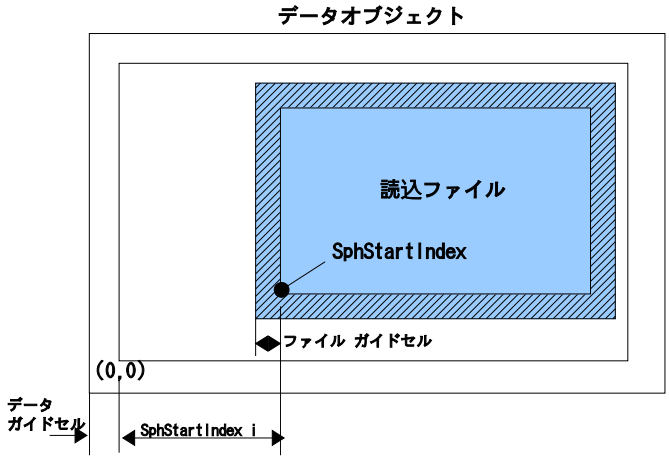


"gc_insph"属性は SPH ファイルフォーマットの読み込み時のみ有効な設定です。SPH ファイルのデータサイズから gc_insph 属性値分内側をガイドセルを除いたデータ領域の(0,0)位置とします。

(4) SphStartIndex 要素（始点インデックス）

要素名	SphStartIndex	
用　途	始点グローバルインデックスを定義する。	
属　　性		
属性名	値	種別
i	I 方向の始点グローバルインデックス	必須
j	J 方向の始点グローバルインデックス	必須(2D)
k	K 方向の始点グローバルインデックス	必須(3D)

ファイルの読み込み位置（始点インデックス）を指定します。



データオブジェクトの原点(0,0)位置に対して SphStartIndex 要素で設定された値だ

け移動した位置に読込データをセットします。

(5) SphMultiFile 要素（入力ファイル複数指定）

入力ファイルを複数設定する為の要素ですが、今後 DFI ファイル（フォーマット）に変更する予定ですので、ここでの説明は割愛します。

(6) SphDir 要素（ソルバの入出力フォルダ）

要素名	SphDir	
用　途	ソルバの入出力フォルダを定義します。	
属　　性		
属性名	値	種別
input	入力フォルダ名　　デフォルト："."	任意
output	出力フォルダ名　　デフォルト："."	任意

入出力ファイルのフォルダを指定します。

フォルダ名は相対パス、絶対パスの記述が可能です。

相対パスの記述を行った場合は、ソルバーの実行パスを基準フォルダとします。

フォルダ属性の記述がない場合は、XML ファイルのフォルダが入出力フォルダとなります。

２．２　コンフィグレーション記述例

```
<SphSolverConfig>
    <SphSolverList>
        <SphSolver>
            <SphDataList>
                <SphDataObj label="prs" format="scalar3d" data_type="double" gc="1">
                    <SphFile label="pressure" />          <!-- (2) -->
                </SphDataObj>
            </SphDataList>
    </SphSolverList>

    <SphFileList>
        <!-- (1) -->
```

```
        <SphFile label="material" type="in" format="sbx" file_name ="material.sbx"/>
        <!-- (2) -->
        <SphFile label="pressure" type="in" format="spx">
            <SphFileNameFormat foramt="prs_%04d.spx" value="rank"/>
        </SphFile>
        <!-- (3) -->
        <SphFile label="temp" type="in" format="spx" file_name="temp.spx"
            mode="cent" io_nodeid="2" />
        <!--(4) -->
        <SphFile label="vel" type="in" format="sph" file_name="velocity.sph">
            <SphFileOption gc_insph="1" />
        </SphFile>
        <!-- (5) -->
        <SphFile label="bnd" type="in" format="sbx" file_name="boundary.sbx">
            <SphFileOption sbx_bitshandling ="shiftlow" />
        </SphFile>
        <!-- (6) -->
        <SphFile label="avr" type="in" format="sph" file_name="average.sph">
            <SphFileOption gc_insph="1" />
            <SphStartIndex   i="4" j="5" k="2" />
        </SphFile>
    </SphFileList>
</SphSolverConfig>
```

(1) ファイル名指定

- SBX フォーマットの"material.spx"ファイルから読込を行います。
- ファイル識別ラベル"material"を指定して"loadFile"メソッドを呼び出します。
- 並列実行においては、すべてのノードが"material.spx"から自ノード領域の読込を行います。

(2) 書式付きファイル名指定、ファイル関連付け

- SPX フォーマットの"prs_[自ノード番号].spx"ファイルから読込を行います。
- ファイル識別ラベル"pressure"はデータ識別ラベル"prs"に関連付けられていますので、データ識別ラベル"prs"を指定して"getDataObj"メソッドを呼び出します。

(3) 並列実行集中入力指定

・ 集中入力モードです。
・ ノード番号=２がファイルの読込を行い、他のノードに読込データを配信します。

(4) SPH ファイルガイドセル指定
・ SPH フォーマットのファイルからファイルのガイドセルを指定して読込を行います。

(5) SBX ファイルビット操作指定
・ SPX フォーマットのファイルから読込後、ビット操作を行います。
・ ビット操作は下位ビットをシフトして上位ビットのみのデータとします。

(6) 始点インデックス指定
・ ファイルの読込位置を SphStartIndex 要素によって指定します。

２．３　ファイル入力メソッド

コンフィグレーションに定義されたファイル識別ラベル、データ識別ラベルを指定してファイルからデータを取得するメソッドについて以下に説明します。

２．３．１　ファイル識別ラベル指定

bool loadFile(const char* file_label, sphData* data);		
ファイル識別ラベルを指定してファイルから読込を行います。		
引数	file_label	ファイル識別ラベル
	data	読込データ格納用データオブジェクト
戻り値	成否	

コンフィグレーションのファイル識別ラベルを指定してファイルから読込を行います。
読込データを格納するデータオブジェクトは、ソルバ側で生成を行う必要があります。

（使用例）

コンフィグレーション例
<SphFileList> <SphFile id="1" label="sbx_file" format="sbx" type="in" file_name='sbx_unsigned_data.sbx' /> </SphFileList>
コード例

```
int sphSolverSolv01::initializeSolver(sphStepTime* steptime)
{
    // データオブジェクトの生成
    sphData* data = allocateDataObj("sbx_data", SPH_DC_ARRAY_3D, SPL_UNSIGNED, 2);

    // ファイルの読込
    if (!this->loadFile("sbx_file", data)) {
      std::cout << "loadFile:error" << std::endl;
      return SPHERE_ERROR;
    }
}
```

２．３．２　読込サイズ指定

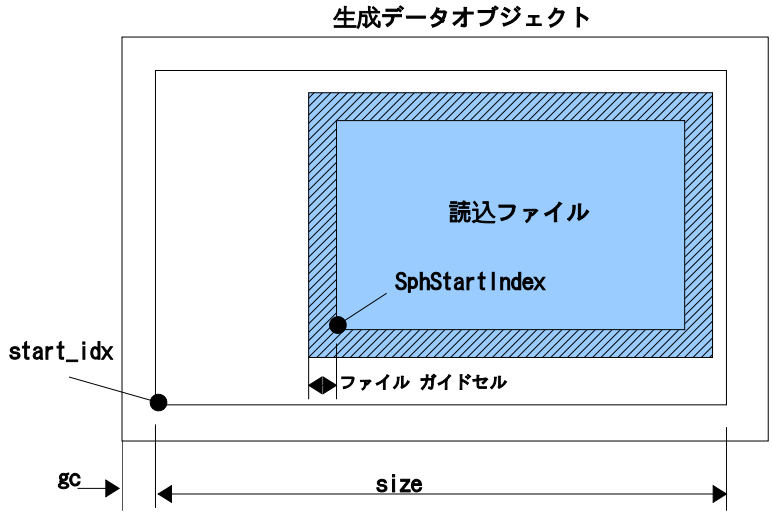
sphData* loadFile (const char* file_label, const size_t size[3], const size_t start_idx[3], size_t gc)		
ファイル識別ラベルのファイルを指定サイズ、始点インデックスにて読み込み、データオブジェクトを生成します。		
引数	file_label	ファイル識別ラベル
	size[3]	読込データサイズ
	start_idx[3]	読込データ始点インデックス
	gc	ガイドセル
戻り値	生成データオブジェクト	

コンフィグレーションのファイル識別ラベルに定義されているファイルからデータを読み込み、データオブジェクトの生成を行います。

生成データオブジェクトは指定読込サイズ、始点インデックス、ガイドセルによって生成しますが、データラベルの設定、データマネージャへの登録は行いません。よって、データオブジェクトは、呼出側にて破棄する必要があります。

size, start_idx, gc は生成されるデータオブジェクトの設定値です。

start_idx は、コンフィグレーションの SphStartIdex 要素に定義された始点インデックスとは異なりますので注意してください。



(1) 引数で指定された size, start_idx, gc にてデータオブジェクトを生成します。

(2) SphStartIndex 要素が定義されていなければ、"<SphStartIndex i='0' j='0' k='0 />と同意ですので、引数で指定された start_idx 分内側の領域を読み込むことになります。

(3) SphStartIndex 要素が定義されていれば、引数で指定された start_idx と SphStartIndex 要素の始点インデックスの相対領域を読み込むことになります。

(4) データオブジェクトの生成領域とファイル読込領域が重ならない部分は 0x00 に初期化されます。

(使用例)

コンフィグレーション例

```
<SphFileList>
  <SphFile id="1" label="sbx_file" format="sbx" type="in" file_name='sbx_unsigned_data.sbx'   />
</SphFileList>
```

コード例

```
// パラレルコントローラの取得
sphParaController* para_ctrl = getParaController();
int myID = para_ctrl->getMyID();
// 読込サイズ、始点インデックスの設定
const size_t data_size[3] = para_ctrl->getVoxelSize();
size_t start_index[3] = {para_ctrl->getVoxelHeadIndex(myID, 0),
                         para_ctrl->getVoxelHeadIndex(myID, 1),
                         para_ctrl->getVoxelHeadIndex(myID, 2)};
```

```
size_t gc = 2;

// ファイルから読込を行う。
sphData* data = this->loadFile("sbx_file", data_size, start_index, gc);
if (data == NULL) {
   std::cout << "loadFile:error" << std::endl;
   return SPHERE_ERROR;
}
```

通常このメソッドを使用する場合は、パラレルコントローラより自ノードサイズ、始点インデックスを取得し、必要に応じて変更を行い使用することになると思います。

２．３．３　データ識別ラベル指定

sphData* getDataObj(const char* cfg_label); sphData* allocateDataObj(const char* cfg_label);	
コンフィグレーションのデータ識別ラベルを指定してデータ要素に関連付けられたファイルから読込を行います。	
引数	cfg_label　　データ識別ラベル
戻り値	データオブジェクト

コンフィグレーションの<SphDataObj>要素配下に<SphFile>要素が記述されている場合、ファイルからデータを読み込み、データオブジェクトの生成までを自動的に行います。

コンフィグレーションにソルバー起動時の自動生成が設定されている場合は、ソルバー起動前に自動的に生成済みですので、ソルバー側は、"getDataObj"メソッドにより取得します。

自動生成が設定されていない場合は、ソルバー側で"allocateDataObj"によって、ファイルからデータを読み込み、データオブジェクトの生成を行います。

No	メソッド	SphDataObj 要素 instance 属性値	自動生成	読込・生成タイミング
1	getDataObj	"instance" 又は「記述なし」	○	ソルバーインスタンス時
2	allocateDataObj	"user"	×	"allocateDataObj"呼出時

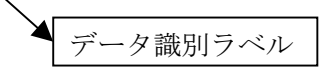
これのメソッドは、コンフィグレーションに定義するだけで、ファイルデータのデータオブジェクトの取得を行うことができるメソッドです。

以下、コンフィグレーションの記述例、メソッドの使用例について記述します。

（コンフィグレーション記述例）

```
<SphereConfig ver="2.0.0" >
  <SphSolverList>
    <SphSolver id="1" label="c3d" class="c3d" dims="3">
      <SphDataList>
        <SphDataObj id="1" label="prs" format="scalar3d" data_type="double" gc="1">
          <SphFile label="prs_file" />
        </SphDataObj>
        <SphDataObj id="2" label="prs_tmp" format="scalar3d" data_type="double" gc="3"
                                instance="user">
          <SphFile id="1" />
        </SphDataObj>
      </SphDataList>
    </SphSolver>
  </SphSolverList>

  <SphFileList>
    <SphFile label="prs_file" format="sph" type="in"' file_name='real8_647_00.sph'/>
  </SphFileList>
</SphereConfig>
```



1) "SphSolver/SphDataList"要素に"SphDataObj"要素を記述します。

<SphDataObj instance='instance">：自動生成する。（デフォルト）

<SphDataObj instance="user">：自動生成しない。

2) "SphDataObj"要素の子要素に"SphFile"要素を記述します。

3) "SphFile"要素 id 属性には、読込ファイル名の記述された"SphFileList"要素配下の"SphFile"要素の"label"(又は"id")属性を記述します。(label 属性の記述の可能です。)

4) "SphFileList"要素配下に読込ファイル名を定義した"SphFile"要素を記述します。（type 属性は"in"である必要があります。）

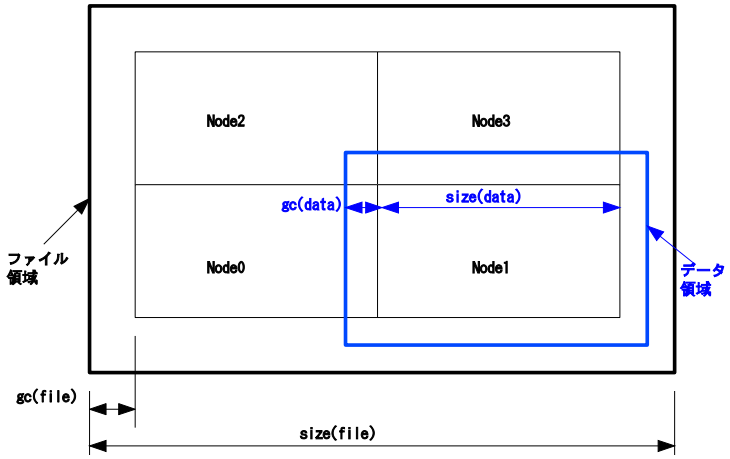
（メソッド使用例）

```
// 生成済みの"prs"を取得する。
sphVoxArray<double>* array = dynamic_cast<sphVoxArray<double>*>(getDataObj("prs"));
if (array == NULL) {
  return SPHERE_TERMINATE;
```

```
}

// 読込、データオブジェクト生成を行って取得する。
sphVoxArray<double>* array_tmp =
                                    dynamic_cast<sphVoxArray<double>*>(allocateDataObj("prs_tmp"));
if (array_tmp == NULL) {
   return SPHERE_TERMINATE;
}
```

ファイルからの読込領域は、読込ノードの分割情報のボクセルサイズ、始点インデックスによって決定されます。



1) データオブジェクトのデータサイズはノード分割情報に従います。
2) ファイルから読み込みを行う始点インデックスは、分割ノードの始点インデックス位置となります。

３．ファイル出力

ファイルへの書込を行う為にはコンフィグレーションにファイル要素を定義しなければなりません。
最初に、ファイル要素のファイル出力関係の要素、属性、設定値について説明します。
以下に記述していない属性、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。

３．１　コンフィグレーションの説明

出力ファイルの指定を行うコンフィグレーション要素について説明します。

(1) SphFile 要素（入出力ファイル指定）

要素名	SphFile	
用　途	入出力ファイルの指定に使用する	
属　　性		
属性名	値	種別
label (id)	識別ラベル（識別番号）	必須
format	ファイルフォーマット "sbx"：SBX ファイル "spx"：SPX ファイル "sph"：SPH ファイル "text"：テキストファイル	必須
file_name	ファイル名（拡張子付き） file_name 属性、prefix 属性、"SphFileNameFormat" 要素のいずれかが必要。	任意
prefix	ファイル名接頭文字 file_name 属性、prefix 属性、"SphFileNameFormat" 要素のいずれかが必要。	任意
type	"out"（出力ファイル）固定	必須
mode	入出力モード multi：各ノードが読込・書込を行う。 cent：指定ノードが読込。書込を行う。 デフォルト= multi	任意
io_nodeid	ファイルの読み書きを行うノード番号 入出力モード（mode）が"cent"の時のみ有効。 デフォルト＝０	任意

(a) label, id 属性は両方、又はどちらかの定義が可能です。
しかし、ファイルラベル指定"writeFile"メソッドを使用する場合は label 属性が必須ですので、label 属性の定義を推奨します。

(b) 出力ファイル名
出力ファイル名の指定は以下のいずれかで行います。
file_name 属性
prefix 属性

SphFileNameFormat 要素

複数記述されている場合は以下の優先順位にて出力されます。

（低）"file_name"属性 ＜ "prefix"属性 ＜ "SphFileNameFormat"要素（高）

"prefix"属性のファイル接頭文字による出力ファイル名は以下の書式になります。

単独実行又は入出力モード（mode 属性）＝"cent"の場合

["prefix"属性]_[ステップ数(８桁)].["format"属性]

並列実行の場合

["prefix"属性]_[ステップ数(８桁)]_id[ランク番号(6 桁)].["format"属性]

(2) SphFileNameFormat 要素（入出力ファイル名の書式情報）

要素名	SphFileNameFormat	
用　途	入出力ファイル名をランク番号とソルバ実行ステップ数から生成する書式を設定する。	
属　　性		
属性名	値	種別
value	出力値種別 作成するファイルに出力フォーマットに従い付加する値、 "rank"　　ランク番号 "step"　　ステップ数 "step_rank"　ステップ数＋ランク番号 "rank_step"　ランク番号＋ステップ数	必須
format	入出力ファイル名書式 C 言語"printf"文の書式に準じる。	必須

(3) SphFileOption 要素（ファイル入出力オプション情報）

要素名	SphFileOption	
用　途	ファイルの入出力のオプション情報を設定する。	
属　　性		
属性名	値	種別
block_size	圧縮ブロックサイズ デフォルト＝I 方向サイズｘJ 方向サイズをブロックサイズとする。	任意

	block_size="0"の場合、圧縮を行わない。	
extrec_flag	拡張レコードの出力フラグ デフォルト＝"false" "true"：拡張レコードを出力する。 "false"：拡張レコードを出力しない。	任意
rlen	実数データタイプ デフォルト＝"4"（float） "4"：float（４バイト） "8"：double（８バイト） SBX ファイルの書き込み時のみ有効	任意
sbx_validnum	SBX ファイルデータの占有ビット数 占有ビット数: 上位データのビット数 AUX レコードの以下のビット値として出力する。 AUX：下位４ビット　0000XXXX デフォルト＝0	任意
sbx_validflag	SBX ファイルデータのデータ存在情報 "0":上位、下位ビット共に有効データ "1":下位のみに有効データ "2":上位のみに有効データ AUX レコードの以下のビット値として出力する。 AUX：上位４ビット　YYYY0000 デフォルト＝0	任意
guide_out	ファイル出力時にガイドセルを出力するかどうかのフラグ "with"：出力するデータオブジェクトのガイドセル値に従い ガイドセルを出力する。 "without"：ガイドセルを出力しない。 デフォルト＝"without"	任意
interval	ファイル出力ステップ間隔 デフォルト＝１	任意

３．２　コンフィグレーション記述例

```
<SphSolverConfig>
    <SphSolverList>
```

```
        <SphSolver>
            <SphDataList>
                <SphDataObj label="prs" format="scalar3d" data_type="double" gc="1">
                    <SphFile label="prs_file" />          <!-- (2) -->
                </SphDataObj>
            </SphDataList>
    </SphSolverList>

    <SphFileList>
        <!-- (1) -->
        <SphFile label="pressure" type="out" format="spx"    prefix="prs"/>
        <!-- (2) -->
        <SphFile label="prs_file" type="out" format="spx">
            <SphFileNameFormat foramt="prs_%04_id0x04d.spx" value="step_rank"/>
            <SphFileOption interval='100' />
        </SphFile>
        <!-- (3) -->
        <SphFile label="temp" type="out" format="spx" prefix="temp" mode="cent"    />
        <!--(4) -->
        <SphFile label="vel" type="out" format="sph"    prefix="vel">
            <SphFileOption    guide_out ="with" />
        </SphFile>
        <!-- (5) -->
        <SphFile label="bnd" type="in" format="sbx" file_name="boundary.sbx">
            <SphFileOption    sbx_validflag ="2" />
        </SphFile>
    </SphFileList>
</SphSolverConfig>
```

(1) ファイル名接頭文字指定

- SPX フォーマットにて"prs"をファイル名先頭に付け出力を行います。
- 並列実行においては、すべてのノードが出力を行います。

(2) 書式付きファイル名指定、ファイル関連付け

- SPX フォーマットにてファイル名書式に従いファイル出力を行います。
- ファイル識別ラベル"prs_file"はデータ識別ラベル"prs"に関連付けられていますの

で、データ識別ラベル"prs"を指定して"writeFile"メソッドを呼び出します。

・ ファイルへの出力間隔は 100 ステップ毎に出力されます。

(3) 並列実行集中入力指定

・ 集中出力モードです。

・ ノード番号=0（デフォルト）が他のノードからデータ収集を行い、ファイルに出力を行います。

(4) ガイドセル出力指定

・ SPH フォーマットのファイルへガイドセルを出力しないで書込を行います。

(5) AUX レコード指定

・ SPX フォーマットの AUX レコードへの出力値を指定してファイルに出力を行います。

３．３　ファイル出力メソッド

コンフィグレーションに定義されたファイル識別ラベル、データ識別ラベル、出力設定を指定してファイルへデータを出力するメソッドについて以下に説明します。

３．３．１　ファイル識別ラベル指定

bool writeFile (const char *　file_label,　sphData *　dataObj)		
データオブジェクトをコンフィグレーションのファイル識別ラベルのファイル要素定義に従ってファイルの書込みを行います。		
引数	file_label	ファイル識別ラベル
	dataObj	出力データオブジェクト
戻り値	成否	

コンフィグレーションの<SphFile>要素のファイル識別ラベルとデータオブジェクトを指定します。

出力データの生成済みのデータオブジェクトをファイル識別ラベルの<SphFile>要素の設定に従ってファイルに書き込みを行います。

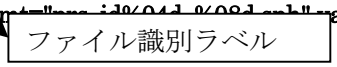
出力ステップ数、時間は現在実行中のステップ数、時間が出力されます。

また、<SphFileOption>要素の"interval"属性が設定されている場合は、実行ステップが設定間隔になったときのみ出力されます。

以下、コンフィグレーションの記述例、メソッドの使用例について記述します。

（コンフィグレーション記述例）

```
<SphereConfig ver="2.0.0" >
  <SphFileList>
    <SphFile id="1" label="prs_file" format="sph" type="out" >
      < SphFileNameFormat format="prs_id%04d_%08d.sph" value="step_rank"/>
    </SphFile>
  </SphFileList>
</SphereConfig>
```



1) "SphFileList"要素配下に"SphFile"要素を記述します。(type 属性は"out"である必要があります。)

2) "SphFileNameFormat"要素に出力ファイル名を定義します。

（メソッド使用例）

```
// データオブジェクトの取得を行う。
sphData* array = getDataObj("prs");
if (array == NULL) {
  return SPHERE_TERMINATE;
}
// データをファイル出力する。
if (!writeFile("prs_out", array)) {
  return SPHERE_TERMINATE;
}
```

３．３．２　出力ステップ・時間指定（ファイル識別ラベル）

```
bool writeFile(
        const char* file_label,
        sphData* dataObj,
        size_t step, double time,
        bool gc_flag = false, bool force = false);
```

データオブジェクトをファイル識別ラベルの定義に従ってファイルの書込みを行います。

step, time, gc_flag, force は引数で設定された値、設定にて出力します。

引数	file_label	ファイル識別ラベル
	dataObj	出力データオブジェクト

	step	出力ステップ数
	time	出力時間
	gc_flag= false	ガイドセルの出力フラグ デフォルト = false：ガイドセルを出力しない。
	force = false	強制出力フラグ デフォルト = false："interval"に従って出力を行う。
戻り値	成否	

コンフィグレーションの<SphFile>要素のファイル識別ラベルとデータオブジェクトを指定します。

出力データの生成済みのデータオブジェクトをファイル識別ラベルの<SphFile>要素の設定に従ってファイルに書き込みを行います。

以下の出力設定を行うことができます。

No	出力設定引数	デフォルト	説明
1	step	なし	出力を行うステップ数を設定します。 実行中のステップ数は無視されます。
2	time	なし	出力を行う時間を設定します。 実行中の時間は無視されます。
3	gc_flag	false	ガイドセルの出力の有無を設定します。 <SphFileOption>要素の"guide_out"属性の設定は無視されます。 true: ガイドセルを出力する。 false: ガイドセルを出力しない。
4	force	false	<SphFileOption>要素の"interval"属性の設定値に従って強制出力を行うかのフラグです。 true："interval"に関係なく出力する。 false："interval"に従って出力を行う。

以下、コンフィグレーションの記述例、メソッドの使用例について記述します。

（コンフィグレーション記述例）

```
<SphereConfig ver="2.0.0" >
  <SphFileList>
    <SphFile id="1" label="prs_file" format="sph" type="out" >
      < SphFileNameFormat foramt="prs_id%04d_%08d.sph" value="step_rank"/>
      <SphFileOption    interval ="100"    guide_out="with" />
                                    <!--    guide_out の記述は無視される  -->
```

```
    </SphFile>
  </SphFileList>
</SphereConfig>
```

1) "SphFileList"要素配下に"SphFile"要素を記述します。(type 属性は"out"である必要があります。)

2) "SphFileNameFormat"要素に出力ファイル名を定義します。

3) <SphFileOption>要素の"interval"属性を設定します。(任意)

4) <SphFileOption>要素の"guide_out"属性の記述は無視されます。引数"gc_flag"の設定が優先されます。(引数"gc_flag"を省略した場合は、gc_flag=false の設定となります。)

(メソッド使用例)

```
int sphSolverFS3D::loopSolver(sphStepTime* steptime) {
  // データラベル"prs"のデータを取得します。
  sphData* array = getDataObj("prs");

  // 現在ステップ、時間を"interval"属性の間隔に従わないで、毎回出力します。
  // ガイドセルを出力します。
  if (!writeFile("prs_out", array, steptime->getCurrentStep(), steptime->getCurrentTime(), true, true))
  {
    return SPHERE_TERMINATE;
  }
  return SPHERE_SUCCESS;
}
```

３．３．３　データ登録ラベル指定

bool writeFile(const char* file_label, const char* data_reg_label);		
データ登録ラベルのデータオブジェクトをコンフィグレーションのファイル識別ラベルのファイル要素定義に従ってファイルの書込みを行います。		
引数	file_label	ファイル識別ラベル
	data_reg_label	出力データ登録ラベル（データ識別ラベル）
戻り値	成否	

コンフィグレーションの<SphFile>要素のファイル識別ラベルと生成済みのデータオブジェクトのデータ登録ラベルを指定します。

データ登録ラベルは、<SphDataObj>要素に定義を行ったデータオブジェクトの場合はデータ識別ラベルと同じとなります。

データ登録ラベル（データ識別ラベル）によって生成済みのデータオブジェクトをファイル識別ラベルの<SphFile>要素の設定に従ってファイルに書き込みを行います。

データ登録ラベルは、コンフィグレーションの<SphDataObj>要素に定義されていなくても、ソルバによってデータマネージャに登録されたデータオブジェクトのデータ登録ラベルも指定できます。

出力ステップ数、時間は現在実行中のステップ数、時間が出力されます。
また、<SphFileOption>要素の"interval"属性が設定されている場合は、実行ステップが設定間隔になったときのみ出力されます。

以下、コンフィグレーションの記述例、メソッドの使用例について記述します。

（コンフィグレーション記述例）

```
<SphereConfig ver="2.0.0" >
  <SphSolverList>
    <SphSolver id="1" label="c3d" class="c3d" dims="3">
      <SphDataList>
        <SphDataObj id="1" label="prs_data" format="scalar3d" data_type="double" gc="1" />
      </SphDataList>
    </SphSolver>
  </SphSolverList>

  <SphFileList>
    <SphFile id="1" label="prs_file" format="sph" type="out" >
      < SphFileNameFormat format="prs_id%04d_%08d.sph" value="step_rank"/>
    </SphFile>
  </SphFileList>
</SphereConfig>
```


1) "SphSolver/SphDataList"要素に"SphDataObj"要素を記述します。
（"SphDataObj"要素を記述は必須ではありません。）

2) "SphFileList"要素配下に"SphFile"要素を記述します。(type 属性は"out"である必要があります。)

3) "SphFileNameFormat"要素に出力ファイル名を定義します。

（メソッド使用例）

```
// データ識別ラベル"prs_data"のデータを出力する。
if (!writeFile("prs_file", "prs_data")) {
  return SPHERE_TERMINATE;
}
```

３．３．４　データ識別ラベル指定

bool writeFile(const char* data_cfg_label);	
コンフィグレーションのデータ識別ラベルにて生成されたデータオブジェクトをデータ要素に関連付けられたファイル要素定義に従ってファイルの書込みを行います。	
引数	data_cfg_label　　出力データ識別ラベル
戻り値	成否

コンフィグレーションの<SphDataObj>要素配下に<SphFile>要素が記述されている場合、"writeFile"メソッドによって<SphFile>要素の定義に従ってファイルに書き込みます。

出力ステップ数、時間は現在実行中のステップ数、時間が出力されます。

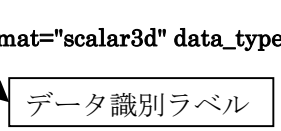
また、<SphFileOption>要素の"interval"属性が設定されている場合は、実行ステップが設定間隔になったときのみ出力されます。

以下、コンフィグレーションの記述例、メソッドの使用例について記述します。

（コンフィグレーション記述例）

```
<SphereConfig ver="2.0.0" >
  <SphSolverList>
    <SphSolver id="1" label="c3d" class="c3d" dims="3">
      <SphDataList>
        <SphDataObj id="1" label="prs" format="scalar3d" data_type="double" gc="1">
          <SphFile label="prs_file" />
        </SphDataObj>
      </SphDataList>
    </SphSolver>
  </SphSolverList>

  <SphFileList>
    <SphFile label="prs_file" format="sph" type="out" >
      < SphFileNameFormat foramt="prs_id%04d_%08d.sph" value="step_rank"/>
```



```
    </SphFile>
  </SphFileList>
</SphereConfig>
```

1) "SphSolver/SphDataList"要素に"SphDataObj"要素を記述します。
2) "SphDataObj"要素の子要素に"SphFile"要素を記述します。
3) "SphFile"要素 id 属性には、書込ファイル名の記述された"SphFileList"要素配下の"SphFile"要素の"id"を記述します。(label 属性の記述の可能です。)
4) "SphFileList"要素配下に"SphFile"要素を記述します。(type 属性は"out"である必要があります。)
5) "SphFileNameFormat"要素に出力ファイル名を定義します。

(メソッド使用例)

```
// データ識別ラベル"prs"のデータを出力する。
if (!writeFile("prs")) {
  return SPHERE_TERMINATE;
}
```

３．３．５　出力ステップ・時間指定（データ識別ラベル）

```
bool writeFile(
        const char* data_cfg_label,
        size_t step,
        double time,
        bool gc_flag = false,
        bool force = false);
```

コンフィグレーションのデータ識別ラベルにて生成されたデータオブジェクトをデータ要素に関連付けられたファイル要素定義に従ってファイルの書込みを行います。
step, time, gc_flag, force は引数で設定された値、設定にて出力します。

引数		
	data_cfg_label	出力データ識別ラベル
	step	出力ステップ数
	time	出力時間
	gc_flag= false	ガイドセルの出力フラグ デフォルト = false： ガイドセルを出力しない。
	force = false	強制出力フラグ デフォルト = false："interval"に従って出力を行う。

戻り値	成否

コンフィグレーションの<SphDataObj>要素配下に<SphFile>要素が記述されている場合、"writeFile"メソッドによって<SphFile>要素の定義に従ってファイルを書き込みます。

以下の出力設定を行うことができます。

No	出力設定引数	デフォルト	説明
1	step	なし	出力を行うステップ数を設定します。 実行中のステップ数は無視されます。
2	time	なし	出力を行う時間を設定します。 実行中の時間は無視されます。
3	gc_flag	false	ガイドセルの出力の有無を設定します。 <SphFileOption>要素の"guide_out"属性の設定は無視されます。 true: ガイドセルを出力する。 false: ガイドセルを出力しない。
4	force	false	<SphFileOption>要素の"interval"属性の設定値に従って強制出力を行うかのフラグです。 true： "interval"に関係なく出力する。 false： "interval"に従って出力を行う。

以下、コンフィグレーションの記述例、メソッドの使用例について記述します。

（コンフィグレーション記述例）

```
<SphereConfig ver="2.0.0" >
  <SphSolverList>
    <SphSolver id="1" label="c3d" class="c3d" dims="3">
      <SphDataList>
        <SphDataObj id="1" label="prs" format="scalar3d" data_type="double" gc="1">
          <SphFile  label="prs_file" />
        </SphDataObj>
      </SphDataList>
    </SphSolver>
  </SphSolverList>

  <SphFileList>
    <SphFile label="prs_file" format="sph" type="out" >
      < SphFileNameFormat foramt="prs_id%04d_%08d.sph" value="step_rank"/>
      <SphFileOption    interval ="100"    guide_out="with" />
```

データ識別ラベル

```
                                    <!--  guide_out の記述は無視される  -->
    </SphFile>
  </SphFileList>
</SphereConfig>
```

1) "SphSolver/SphDataList"要素に"SphDataObj"要素を記述します。

2) "SphDataObj"要素の子要素に"SphFile"要素を記述します。

3) "SphFile"要素 id 属性には、書込ファイル名の記述された"SphFileList"要素配下の"SphFile"要素の"id"を記述します。(label 属性の記述の可能です。)

4) "SphFileList"要素配下に"SphFile"要素を記述します。(type 属性は"out"である必要があります。)

5) "SphFileNameFormat"要素に出力ファイル名を定義します。

6) <SphFileOption>要素の"interval"属性を設定します。(任意)

7) <SphFileOption>要素の"guide_out"属性の記述は無視されます。引数"gc_flag"の設定が優先されます。(引数"gc_flag"を省略した場合は、gc_flag=false の設定となります。)

(メソッド使用例)

```
int sphSolverFS3D::loopSolver(sphStepTime* steptime) {
  // データラベル"prs"のデータを出力します。
  // 現在ステップ、時間を"interval"属性の間隔に従わないで、毎回出力します。
  // ガイドセルを出力します。
  if (!writeFile("prs", steptime->getCurrentStep(), steptime->getCurrentTime(), true, true)) {
    return SPHERE_TERMINATE;
  }
  return SPHERE_SUCCESS;
}
```

４．ファイルヘッダー取得

コンフィグレーションの<SphFile>要素のファイル識別ラベルに定義されているファイルのヘッダー情報を取得します。

ファイルフォーマットによって、以下のヘッダーファイルをインクルードする必要があります。

No	ファイルフォーマット	インクルードヘッダーファイル	ヘッダー情報クラス
1	sbx	"fileio/sphSbxHeader.h";	sphSbxHeader

2	spx	"fileio/sphSpxHeader.h";	sphSpxHeader
3	sph	"fileio/sphSphHeader.h";	sphSphHeader

（メソッド）

bool getFileHeader(const char* file_label, sphFileHeader* header);		
ファイルのヘッダー情報を取得します。		
引数	file_label	コンフィグレーション記述のファイル識別ラベル
	heade	ファイルヘッダーオブジェクト 呼出側でインスタンスする必要があります。
戻り値	成否	

以下、コンフィグレーションの記述例、メソッドの使用例について記述します。

（コンフィグレーション記述例）

```
<SphereConfig version="2.0.0" >
  <SphFileList>
    <SphFile id="1" label="prs_in" format="sph" type="in"' file_name='real8_647_00.sph'/>
  </SphFileList>
</SphereConfig>
```

ファイル識別ラベル

1）"SphFileList"要素配下に読込ファイル名を定義した"SphFile"要素を記述します。（type 属性は"in"である必要があります。）

（メソッド使用例）

```
#include "fileio/sphSphHeader.h";            // SPH ファイル用のヘッダーファイル


int sphSolverFS3D::initializeSolver(sphStepTime* steptime) {
  // SPH ファイルのヘッダー情報を取得します。
  sphSphHeader header;
  getFileHeader("prs_in", &header);
}
```

ヘッダー情報は、ヘッダー情報クラス（sphSbxHeader, sphSpxHeader, sphSphHeader）にセットされますので、ヘッダー情報を取得する場合は、以下のメンバー変数から値を取得します。

ファイルフォーマットによってサポートするメンバー変数が異なりますので、以下、ヘッダー情報クラスのメンバー変数について説明します。

No	メンバー変数	ファイルフォーマット			説明
		sbx	spx	sph	
1	unsigned int m_dims;	○	○	△	次元数 sph の場合は 3 固定
2	unsigned int m_vlen;	○	○	○	ベクトル数
3	unsigned int m_dtype;	○	○	○	データタイプ
4	size_t m_gc;	○	○	×	ガイドセル数
5	unsigned int m_rlen;	○	○	○	実数タイプ
6	unsigned int m_crddef;	○	○	×	データ定義位置
7	unsigned int m_aux;	○	○	×	AUX 値
8	size_t m_size[3];	○	○	○	ボクセルサイズ(GC を含む)
9	double m_origin[3];	○	○	○	原点座標
10	double m_pitch[3];	○	○	○	ピッチ
11	size_t m_block_size;	○	○	×	ブロックサイズ
12	size_t m_step;	×	○	○	**ステップ数**
13	double m_time;	×	○	○	時間
14	size_t m_wsize[3];	○	○	×	全体ボクセルサイズ
15	size_t m_start_idx[3];	○	○	×	始点グローバルインデックス

５．テキストファイル出力

テキストファイルへの出力サポートの為に SPHERE では、"sphTextFile"クラスを用意しています。

"sphTextFile"クラスによるテキストファイルの出力を行う前に以下のメソッドにより"sphTextFile"クラスの構築を行う必要があります。

（ソルバクラスメソッド）

No	関数定義	説明
1	bool getTextFile(const char* file_label, sphTextFile* text_file)	コンフィグレーション記述されたファイル識別ラベルの <SphFile> 要素の定義に従って sphTextFile クラスを構築します。

構築された sphTextFile クラスは、以下のメソッドによりテキストファイル出力を行います。

（sphTextFile クラスメソッド）

No	関数定義	説明
1	bool openFile(bool init = false);	ファイルをオープンして書き込みが可能な状態にします。
2	bool print(const char* fmt, ...) const;	ファイルにテキスト出力します。 （改行なし）
3	bool printWithConsole (const char* fmt, ...) const;	ファイルと標準出力にテキスト出力します。 （改行なし）
4	bool printLine(const char* fmt, ...) const;	ファイルにテキスト出力します。 （改行付加）
5	bool printLineWithConsole (const char* fmt, ...) const;	ファイルと標準出力にテキスト出力します。 （改行付加）
6	bool close();	ファイルを明示的にクローズします。 sphTextFile クラスの破棄時に自動的にクローズされますので、明示的にクローズする場合に使用してください、
7	bool flush();	書き込みバッファーのテキストをファイルに出力します。 リアルタイムにテキストファイルを更新したい時に使用します。
8	bool getFileName(char* file_name) const;	出力テキストファイル名を取得します。

以下、テキストファイルの出力方法について説明します。

５．１　コンフィグレーション

テキストファイル出力の為のファイル名等の設定をコンフィグレーションに記述する必要があります。

(1) SphFile 要素（入出力ファイル指定）

要素名	SphFile	
用　途	入出力ファイルの指定に使用する	
属　　性		
属性名	値	種別
label	識別ラベル	必須

format	ファイルフォーマット "text"：テキストファイル	必須
file_name	ファイル名（拡張子付き） file_name 属性、prefix 属性、"SphFileNameFormat" 要素のいずれかが必要。	任意
prefix	ファイル名接頭文字 file_name 属性、prefix 属性、"SphFileNameFormat" 要素のいずれかが必要。	任意
type	"out"（出力ファイル）固定	必須
mode	入出力モード multi：各ノードが読込・書込を行う。 cent：指定ノードが読込。書込を行う。 デフォルト= cent	任意
io_nodeid	ファイルの読み書きを行うノード番号 入出力モード（mode）が"cent"の時のみ有効。 デフォルト＝０	任意

(a) 入出力モード

テキストファイル出力の場合、"cent"がデフォルトとなります。

よって、mode, io_nodeid を指定しない場合、ノード 0 のテキスト出力のみファイル出力します。

他のフォーマットのファイル出力では入出力モード(mode)は"multi"がデフォルト値ですので注意してください。

(2) SphFileNameFormat 要素（入出力ファイル名の書式情報）

要素名	SphFileNameFormat	
用　途	入出力ファイル名をランク番号とソルバ実行ステップ数から生成する書式を設定する。	
属　　性		
属性名	値	種別
value	出力値種別 作成するファイルに出力フォーマットに従い付加する値、 "rank"　　ランク番号 "step"　　ステップ数 "step_rank"　ステップ数＋ランク番号	必須

	"rank_step"　ランク番号＋ステップ数	
format	入出力ファイル名書式 C 言語"printf"文の書式に準じる。	必須

（コンフィグレーション記述例）

```
<SphereConfig ver="2.0.0" >
  <SphFileList>
      <SphFile id="1" label="history" format="text" type="out" file_name="history.txt"
                  mode="cent" io_nodeid="0" />
  </SphFileList>
</SphereConfig>
```

1) ファイル識別ラベルを"history"に設定します。
2) ファイルフォーマットは"text"、出力用("out")とします。
3) 出力ファイル名は、"history.txt"とします。
4) 出力モードは１つのノードのみ出力("cent")します。
5) 出力はノード０が出力します。（省略可能）

５．２　sphTextFile クラスの構築

コンフィグレーションに定義されたテキストファイル出力設定によって sphTextFile クラスを構築する為に以下のメソッドを使用します。

（メソッド仕様）

bool getTextFile(const char* file_label, sphTextFile* text_file, bool initopen = true)		
説明	コンフィグレーション記述されたファイル識別ラベルの<SphFile>要素の定義に従って sphTextFile クラスを構築します。	
引数	const char* file_label	コンフィグレーション記述のファイル識別ラベル
	sphTextFile* text_file	構築を行う sphTextFile クラス
	bool initopen	追加モードでファイルをオープンします。 デフォルト：true=追加モードでファイルオープン
戻り値	成否	true:構築成功

（メソッド使用例）

```
// sphTextFile クラスを構築します。
sphTextFile history_file;
if (!getTextFile("history", &history_file)) {
   return SPHERE_TERMINATE;
}
history_file->printLine("Solver Start");
```

５．３　テキストファイルのオープン・クローズ

構築した sphTextFile クラスを用いてテキストファイルへの出力を行う前に出力テキストファイルをオープンします。

以下、オープン、クロースメソッドについて説明します。

（sphTextFile クラスメソッド仕様）

bool openFile(bool init = false);		
説明	ファイルをオープンして書き込みが可能な状態にします。	
引数	bool init = false	ファイルのオープンモードを設定します。 true: ファイルを上書きモードでオープンする。 false: ファイルを追加モードでオープンする。 デフォルト = false
戻り値	成否	true:ファイルオープン成功

出力ファイルをオープンします。**既にファイルが存在していた場合、**上書きモード(init=true)**は以前のファイルデータは失われ先頭から上書きします。**追加モード(init=false)はファイル末尾に追加します。

bool close();		
説明	ファイルを明示的にクローズします。	
引数	**なし**	
戻り値	成否	true:ファイルクローズ成功

sphTextFile クラスオブジェクトの破棄時にクローズしますので、出力ファイルを明示的にクローズする時のみ使用します。

bool flush();		
説明	書き込みバッファーのテキストをファイルに出力します。	
引数	**なし**	
戻り値	成否	true:ファイルフラッシュ成功

書き込みバッファーは自動的、又はクローズ時にテキストファイルへ出力されますのでリアルタイムに出力ファイルを閲覧したい時に使用します。

bool getFileName(char* file_name) const;		
説明	出力テキストファイル名を取得します。	
引数	char* file_name	出力テキストファイル名
戻り値	成否	true:取得成功

sphTextFile クラスを使用せず、ソルバ側でファイルオープン、管理を行う場合に設定出力ファイル名を取得します。

（テキストファイル出力例）

```
// sphTextFile クラスを構築します。
sphTextFile history_file;
if (!getTextFile("history", &history_file, false)) {        // ファイルオープンはしない。
   return SPHERE_TERMINATE;
}
// ファイルオープン（上書きモードでオープン：新規作成）
history_file.openFile(true);

// SPH ファイルのヘッダー情報を取得します。
sphSphHeader header;
getFileHeader("prs_in", &header);

// ヘッダーサイズ出力
history_file.print("size = %d, %d, %d ¥n", header.m_size[0], header.m_size[1], header.m_size[3]);
```

５．４　テキストファイル出力

構築､ファイルオープンした sphTextFile クラスを用いてテキストファイルへの出力を行う方法を以下に説明します。

（sphTextFile クラスメソッド仕様）

bool print(const char* fmt, ...) const;		
説明	ファイルにテキスト出力します。 改行は自動付加しません。	
引数	fmt	テキスト出力フォーマット書式 フォーマット書式は、"printf"文書式に従います。
	...	出力パラメータ
戻り値	成否	true:テキスト出力成功

bool printWithConsole(const char* fmt, ...) const;		
説明	ファイルと標準出力（コンソール）にテキスト出力します。 改行は自動付加しません。	
引数	fmt	テキスト出力フォーマット書式 フォーマット書式は、"printf"文書式に従います。
	...	出力パラメータ
戻り値	成否	true:テキスト出力成功

bool printLine(const char* fmt, ...) const;		
説明	ファイルにテキスト出力します。 改行を自動付加します。	
引数	fmt	テキスト出力フォーマット書式 フォーマット書式は、"printf"文書式に従います。
	...	出力パラメータ
戻り値	成否	true:テキスト出力成功

bool printLineWithConsole(const char* fmt, ...) const;		
説明	ファイルと標準出力（コンソール）にテキスト出力します。 改行を自動付加します。	
引数	fmt	テキスト出力フォーマット書式 フォーマット書式は、"printf"文書式に従います。

	...	出力パラメータ
戻り値	成否	true:テキスト出力成功

（テキストファイル出力例）

```
// ファイルサイズ出力
history_file.printLineWithConsole("Start Solver");
history_file. printWithConsole ("size =");
history_file.printWithConsole("%d, %d, %d ¥n",
                                header.m_size[0], header.m_size[1], header.m_size[3]);
---------- 出力例 ----------
Start Solver
size = 64, 64, 64
```