Sphere

Skeleton for PHysical and Engineering REsearch

Ver 2.0.0

ソルバクラスマニュアル

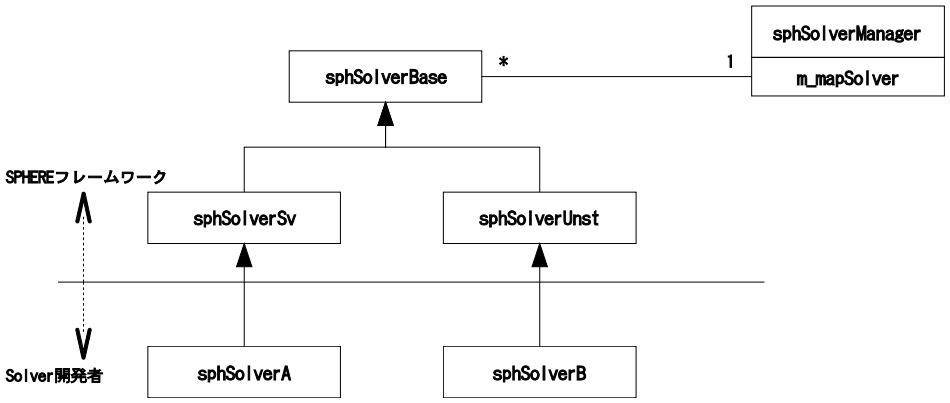
2010 年 8 月 01 日

目次

１．概要

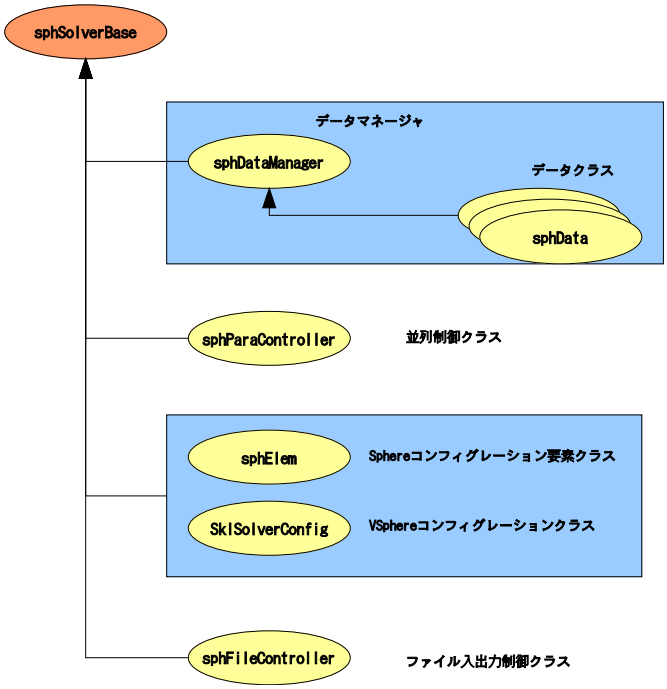
SPHERE フレームワークでは、構造格子ソルバ、非構造格子ソルバをサポートしています。またソルバ間の連成を行うことができるように SPHERE フレームワークでは複数のソルバの管理を行い同時に複数のソルバの実行が可能です。

１．１　クラス構成



クラス名	説明
sphSolverBase	ソルバクラスの基底クラスです。
sphSolverSv	構造格子用のソルバの基底クラスです。
sphSolverUnst	非構造格子用のソルバの基底クラスです。
sphSolverA/sphSolverB	Solver 開発者が実際に構造計算処理を実装するクラスです。 実装ソルバによって構造格子ソルバ(sphSolverSv)、又は非構造格子ソルバ(sphSolverUnst)から派生させます。
sphSolverManager	実行する複数のソルバを管理します。

１．２　ソルバクラス構造



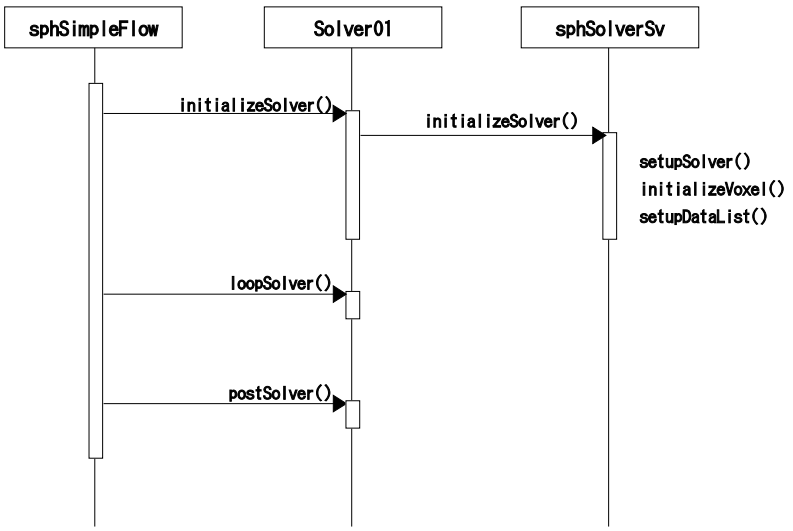
メンバクラス名	説明
sphDataManager	ソルバで使用するデータの管理を行います。
sphParaController	ソルバの計算領域、並列実行の制御を行います。
sphElem	Sphere 形式のコンフィグレーション要素です。 インスタンスソルバのソルバ要素を持ちます。 SklSolverConfig と同時に持つことはできません。
SklSolverConfig	Vsphere 形式のコンフィグレーションクラスです。 sphElem と同時に持つことはできません。
sphFileController	ファイル入出力の制御を行います。

１．３　ソルバ実行フロー

ソルバの実行における実行メソッドのフローについて説明します。

１．３．１　構造格子ソルバ(sphSolverSv クラス)

構造格子ソルバ(sphSolverSv クラス)における実行メソッドのフローについて説明します。



No	メソッド名	説明
1	virtual int initializeSolver(sphStepTime* steptime);	フロー制御クラスから呼び出され、ソルバの初期化を行います。 デフォルトでは、sphSolverSv クラスの initializeSolver メソッドを呼び出します。
2	virtual int loopSolver(sphStepTime* steptime);	フロー制御クラスから呼び出され、ソルバのステップ実行を行います。
3	virtual int postSolver(sphStepTime* steptime);	フロー制御クラスから呼び出され、ソルバの破棄処理を行います。
4	virtual int sphSolverSv::initializeSolver(sphStepTime* steptime);	構造格子用ソルバの基底クラスの初期化処理です。 以下の３つのメソッドを呼び出します。 1) setupSolver() 2) initializeVoxel(); 3) setupDataList();
5	virtual int sphSolverSv::setupSolver();	コンフィグレーションからソルバの実行モード、次元数を取得します。
6	virtual int sphSolverSv::initializeVoxel();	ソルバの計算領域を決定し、並列制御クラスのインスタンスを行います。
7	virtual int sphSolverSv::setupDataList();	データオブジェクトの生成を行います。

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ソルバ要素を定義することにより、ソルバクラスをインスタンスします。

以下、コンフィグレーションのソルバ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。

２．１　SphSolver 要素（ソルバ）

記述ルール

要素名	SphSolver	
用　途	実行ソルバを定義する。	
属　　性		
属性名	値	種別
id	識別 ID（整数）	必須
label	識別名（文字列）	必須
class	インスタンスソルバクラス名	必須
mode	実行モードを設定する。 "check":チェックモード （MainLoop は実行しない。）	任意
dims	次元数を定義する。 "1":１次元 "2":２次元 "3":３次元（デフォルト）	任意
enable	実行可能／不可フラグ "true":実行可能　（デフォルト） "false":実行不可	任意
内包可能な要素（子要素として取り得る要素）		
要素名	意味	種別
SphFlow	ソルバ独自のプログラムフローを定義する。	任意
SphDomainInfo	ソルバのボクセル定義情報の記述	必須
SphSteer	ソルバの実行パラメータ	必須
SphDataList	データリスト	必須
SphUserDefine	ユーザパラメータ	必須

ソルバ要素(SphSolver)は、ソルバリスト要素(SphSolverList)配下に定義します。
複数記述可能ですが、識別 ID(id)、識別名(label)は一意とする必要があります。
ソルバクラス名(class)は、同一ソルバをインスタンスを行うのであれば、同一のソルバクラス名を用いることができます。

記述例

```
<SphSolverList>
  <SphSolver id="1" label="c3d" class="c3d">
    <SphFlow></ SphFlow >
    <SphDomainInfo></SphDomainInfo>
    <SphSteer></SphSteer>
    <SphDataList></SphDataList>
    <SphUserDefine></SphUserDefine>
  </SphSolver>
</SphSolverList>
```

３．ソルバクラス実装

３．１　ソルバクラスの生成クラス・メソッド

ソルバクラスはソルバ側に実装された sphSolverFactory クラスによって生成されます。

sphSolverBase* sphSolverFactory::factorySolver(const char* class_name, const char* label, int id) const		
ソルバクラスの生成を行う		
引数	class_name	ソルバクラス名
	label	ソルバ識別ラベル
	id	ソルバ識別 ID
戻り値	生成カプラ	

ソルバクラス名、ソルバ識別ラベル、ソルバ識別 ID はコンフィグレーションに定義されたインスタンスフロークラス名、識別名（文字列)、識別 ID（整数）です。
ソルバクラス名によってソルバ側に実装しているソルバクラスを判断し、ソルバクラスの生成を行います。

（使用例）

```
sphSolverBase*
sphSolverFactory::factorySolver(const char* class_name, const char* label, int id) const
{
  sphSolverBase* solverObj = NULL;
  if (class_name != NULL) {
    if (strcasecmp(class_name, "SOLVER01") == 0) solverObj = new sphSolver01(label, id);
    if (strcasecmp(class_name, "SOLVER02") == 0) solverObj = new sphSolver02(label, id);
  }
  return solverObj;
}
```

３．２　ソルバクラスのテンプレート

ソルバクラスは以下のテンプレートをソルバ側に実装する必要があります。

(1) ソルバクラスヘッダーファイル（ファイル名：sphSolver01.h）

```
#ifndef _SPHSOLVER01_H_
#define _SPHSOLVER01_H_
#include "solverbase/sphSolverBase.h"
#include "sphSolverSvDefine.h"
class sphSolverFluid : public sphSolverSv {
private:
protected:
  sphSolver01();
public:
  sphSolver01                    (const char* label);
  sphSolver01                    (const char* label, int id);
  virtual ~sphSolver01           (void);
  // -1 : error
  //  0 : forced terminated
  //  1 : Normality
  virtual int initializeSolver(sphStepTime* steptime);
  virtual int loopSolver(sphStepTime* steptime);
  virtual int postSolver(sphStepTime* steptime);
```

```
  virtual int printSolverUsage(const char* cmd);
public:


};
#endif // _SPHSOLVER01_H_
```

(2) ソルバクラスソースファイル（ファイル名：sphSolver01.C）

```
#include "sphSolver01.h"
sphSolver01::sphSolver01()
{
}
sphSolver01::sphSolver01(const char* label) : sphSolverSv(label)
{
}
sphSolver01::sphSolver01(const char* label, int id) : sphSolverSv(label, id)
{
}
sphSolver01::~sphSolver01()
{
}


int
sphSolver01::initializeSolver(sphStepTime* steptime)
{
  if (sphSolverSv::initializeSolver(steptime) != SPHERE_SUCCESS) return SPHERE_ERROR;
  return SPHERE_SUCCESS;
}
int
sphSolver01::loopSolver(sphStepTime* steptime)
{
  return SPHERE_SUCCESS;
}
int
sphSolver01::postSolver(sphStepTime* steptime)
{
```

```
  return SPHERE_SUCCESS;
}
int
sphSolver01::printSolverUsage(const char* cmd) {
  if( !cmd ) return false;
  return SPHERE_SUCCESS;
}
```

以下、テンプレートに実装されたメソッドについて説明します。

３．２．１　コンストラクタ

sphSolver01 (const char* label); sphSolver01 (const char* label, int id);		
ソルバクラスの初期化を行う		
引数	label	ソルバ識別ラベル
	id	ソルバ識別 ID
戻り値	なし	

ソルバクラスの初期化を行います。
コンストラクタにて必ず基底ソルバクラスのコンストラクタを呼び出してください。

３．２．２　デストラクタ

virtual ~sphSolverFluid (void);	
ソルバクラスの破棄処理を行う	
引数	なし
戻り値	なし

ソルバクラスの破棄処理を行います。

３．２．３　ソルバ初期化

```
  virtual int initializeSolver(sphStepTime* steptime);
```

ソルバの初期化を行う	
引数	steptime　　ステップ・時間クラス
戻り値	-1:error, 0:forced terminated, 1:normality

ソルバの初期化を行います。
並列実行環境の初期化、データの生成を行います。

３．２．４　ソルバ実行

virtual int loopSolver(sphStepTime* steptime);	
ソルバのステップ実行を行う	
引数	steptime　　実行ステップ・時間
戻り値	-1:error, 0:forced terminated, 1:normality

ソルバのステップ実行を行います。
現在の実行ステップ数、時間は引数の"steptime"にセットされています。
この steptime にセットされているステップ、時間のインクリメントは SPHERE フレームワーク側で行いますのでソルバ側では通常行う必要はありません。

３．２．５　ソルバ破棄

virtual int postSolver(sphStepTime* steptime);	
ソルバの破棄処理を行います。	
引数	なし
戻り値	-1:error, 0:forced terminated, 1:normality

３．２．６　使用方法出力

virtual int printSolverUsage(const char* cmd);	
ソルバの使用方法、バージョン等を標準出力に出力します。	
引数	cmd　　起動コマンド
戻り値	-1:error, 0:forced terminated, 1:normality

４．メソッド一覧

４．１　基本情報・クラス取得メソッド

ソルバのコンフィグレーションに定義された情報の取得、ソルバの制御クラスの取得メソッドです。

No	メソッド名	説明
1	int getDims() const; void setDims(int dims);	ソルバ次元数の取得・設定を行います。
2	const std::string getSolverMode() const; bool isCheckMode() const;	ソルバの実行モードを取得します。
3	const sphDataManager* getDataManager() const; sphDataManager* getDataManager();	データマネージャの取得を行います。
4	const sphParaController* getParaController() const; sphParaController* getParaController();	並列制御クラスの取得を行います。
5	const sphElem* getConfigElem() const;	Sphere 形式のコンフィグレーションのソルバ要素の取得を行います。
6	const SklCfg::SklSolverConfig* getSklConfig();	Vsphere 形式のコンフィグレーションの取得を行います。
7	bool isClassicConfig();	コンフィグレーションが Vsphere 形式であるかチェックを行います。
8	virtual int sphSolverSv::setupSolver();	コンフィグレーションからソルバの実行モード、次元数を取得します。
9	virtual int sphSolverSv::initializeVoxel();	ソルバの計算領域を決定し、並列制御クラスのインスタンスを行います。
10	virtual int sphSolverSv::setupDataList();	データオブジェクトの生成を行います。

４．２　データメソッド：構造格子ソルバ(sphSolverSv クラス)

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], 　　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しません。
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(	データオブジェクトをデータマネージャに登録します。

	const char* label, sphData* dataObj);	
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。

メソッドの詳細については、「データクラスマニュアル」を参照してください。

４．３　ファイル IO メソッド：構造格子ソルバ(sphSolverSv クラス)

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); 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。
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。

No	入出力方向	メソッド名	説明
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。
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。

メソッドの詳細については、「ファイル入出力マニュアル」を参照してください。

４．４　ボクセル初期化メソッド：構造格子ソルバ(sphSolverSv クラス)

ソルバの計算領域を初期化を行い、並列制御クラス(sphParaController)のインスタンスを行うメソッドです。

No	メソッド名	説明
1	int initializeVoxel(const size_t voxelsize[3], const unsigned int division[3], const double voxel_origin[3] = NULL, const double voxel_pitch[3] = NULL);	I,J,K 方向の分割数からパラレルコントローラの生成を行う。
2	int initializeVoxel(const size_t voxelsize[3], int proc_num = 0, const double voxel_origin[3] = NULL, const double voxel_pitch[3] = NULL);	分割プロセス数からパラレルコントローラの生成を行う。
3	int initializeVoxel(const sphElem* nodelist_elem, const double voxel_origin[3] = NULL, const double voxel_pitch[3] = NULL);	SphNodeList 要素からパラレルコントローラの生成を行う。

メソッドの詳細については、「並列実行形態・並列制御クラスマニュアル」を参照してください。

５．メソッド一覧

５．１　基本情報・クラス取得メソッド

５．１．１　　ソルバ次元数取得・設定

int getDims() const; void setDims(int dims);	
ソルバ次元数の取得・設定を行います。	
引数	なし
戻り値	ソルバ次元数

コンフィグレーションの SphSolver 要素-dims 属性に定義されているソルバ次元数を取得・設定します。
コンフィグレーションから自動取得するには"int sphSolverSv::setupSolver()"を呼び出す必要があります。

５．１．２　ソルバ実行モードの取得

const std::string getSolverMode() const; bool isCheckMode() const;	
ソルバの実行モードを取得します。	
引数	なし
戻り値	ソルバ実行モード文字列 true:チェックモード

コンフィグレーションの SphSolver 要素-mode 属性に定義されているソルバ実行モードを取得します。
mode 属性が"check"の場合、実行モード＝チェックモードとなります。
コンフィグレーションから自動取得するには"int sphSolverSv::setupSolver()"を呼び出す必要があります。

５．１．３　データマネージャの取得

const sphDataManager* getDataManager() const; sphDataManager* getDataManager();	
データマネージャの取得を行います。	
引数	なし
戻り値	データマネージャ

５．１．４　並列制御クラスの取得

const sphParaController* getParaController() const; sphParaController* getParaController();	
並列制御クラスの取得を行います。	
引数	なし
戻り値	並列制御クラス

５．１．５　Sphere 形式ソルバ要素の取得

const sphElem* getConfigElem() const;	
Sphere 形式のコンフィグレーションのソルバ要素の取得を行います。	
引数	なし
戻り値	ソルバ要素

sphere の起動引数に指定されたコンフィグレーションファイルが Sphere 形式の場合、インスタンスソルバのソルバ要素の取得を行います。

５．１．６　Vsphere 形式コンフィグレーションクラスの取得

const SklCfg::SklSolverConfig* getSklConfig();	
Vsphere 形式のコンフィグレーションの取得を行います。	
引数	なし
戻り値	コンフィグレーションクラス

sphere の起動引数に指定されたコンフィグレーションファイルが Vsphere 形式の場合、コンフィグレーションクラスの取得を行います。

５．１．７　Vsphere 形式コンフィグレーションのチェック

bool isClassicConfig();	
コンフィグレーションが Vsphere 形式であるかチェックを行います。	
引数	なし
戻り値	true: Vsphere 形式

sphere の起動引数に指定されたコンフィグレーションファイルが Vsphere 形式の場合、true を返します。

５．１．８　ソルバ基本実行情報の取得

virtual int sphSolverSv::setupSolver();	
コンフィグレーションからソルバの実行モード、次元数を取得します。	
引数	なし
戻り値	-1:error, 0:forced terminated, 1:normality

コンフィグレーションの SphSolver 要素からソルバの実行モード、次元数の取得を行います。
ソルバ開発者が独自に基本情報を決定する場合は、"setupSolver"メソッドをソルバ側に実装してください。

５．１．９　並列制御クラスの生成

virtual int sphSolverSv::initializeVoxel();	
ソルバの計算領域を決定し、並列制御クラスのインスタンスを行います。	
引数	なし
戻り値	-1:error, 0:forced terminated, 1:normality

コンフィグレーションの SphDomainInfo 要素からソルバの計算領域を決定し、並列制御クラスのインスタンスを行います。
ソルバ開発者が独自にソルバ計算領域を決定する場合は、"initializeVoxel"メソッドをソルバ側に実装してください。

５．１．１０ データの生成

virtual int sphSolverSv::setupDataList();	
データオブジェクトの生成を行います。	
引数	なし
戻り値	-1:error, 0:forced terminated, 1:normality

コンフィグレーションの SphDataList 要素からデータオブジェクトの生成を行います。
ソルバ開発者が独自にデータオブジェクトの生成を行う場合は、"setupDataList"メソッドをソルバ側に実装してください。