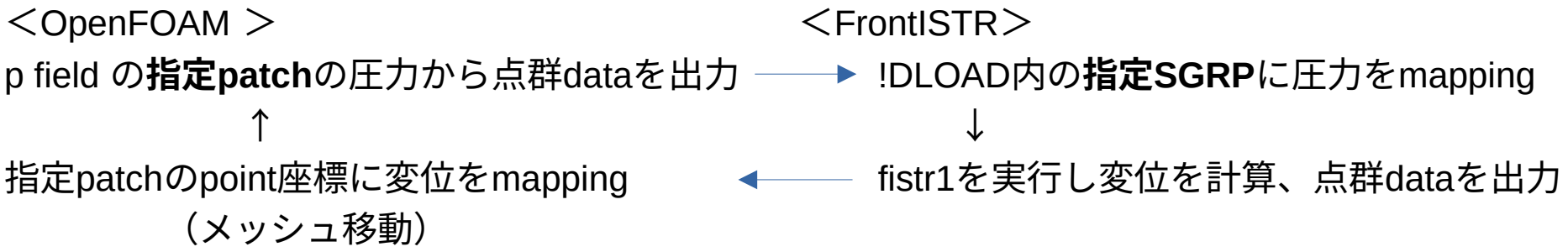
OpenFOAMとFrontISTRによる連成解析
（TreeFoam）（EasyISTR)

TreeFoam-3.17-230716以降、EasyISTR-3.39-230530以降
OF-9、OF-v2206　FrontISTR-5.2以降で作動確認

23/07/29　藤井

1. 連成解析の方法

TreeFoam上から、OpenFOAMとFrontISTRの流体−構造の連成解析を可能にしている。



OpenFOAM　　FrontISTR

「**指定patch**」と「**指定SGRP**」間で圧力、変位をやり取りしながら

流体−構造の連成解析を行っている。

p field

```
dimensions      [0 2 -2 0 0 0 0];

internalField     uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
        value           uniform 0;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    flap
    {
        type            zeroGradient;
        value           uniform 0;
    }
    upperWall
    {
        type            zeroGradient;
        value           uniform 0;
    }
    lowerWall
    {
        type            zeroGradient;
        value           uniform 0;
    }
    frontAndBack
    {
        type            slip;
    }
}
```

OpenFOAM側の p ファイルの内容を
修正して、連成解析を実現している。

```
    flap
    {
        type            codedMixed;
        name            coded_setPress_getDisp;
        value           uniform (0 0 0);
        code
        #{
            pythonスクリプトを起動
              圧力の点群dataをSGRPにmapping
              fistr1を実行
              変位の点群dataをpatch「flap」にmapping
              変位をfileに出力
        #};

    }
```

CのsourceCodeを記述
（任意の内容が記述できる）

2. 連成解析の高速化

従来の連成解析方法は、

　　OpenFOAMの実行中に連成計算結果（変位）が必要になった時、
　　OpenFOAMを一時中断、FrontISTRを実行して変位を計算、
　　その計算結果をOpenFOAMが取得し、OpenFOAMの計算を再開

で、計算している。

FrontISTRは、計算する都度、実行fileをloadして計算開始しており、loadする時間が無駄になっているが、
連成計算するタイミング（deltaTの倍数）が設定できる様にしている為、
時間ループ中で毎回連成計算しなくて済む。

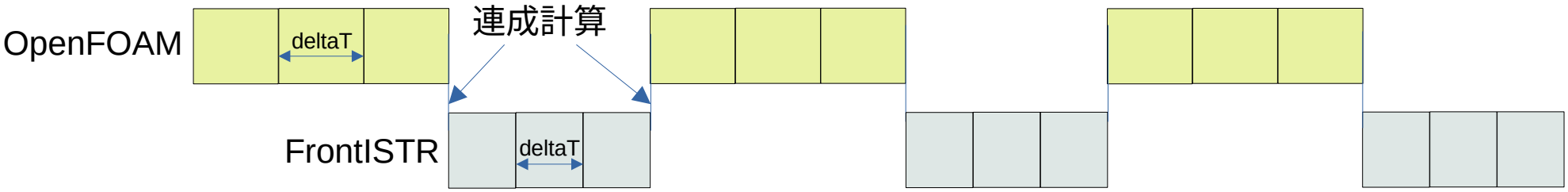
　　↓

高速化の為に、
OpenFOAMとFrontISTRを同時に走らせて、連成計算を進めていく様に修正。

　構造計算時間 ＜ 流体計算時間 の場合、OpenFOAM単独の計算時間で連成計算が終了できる。

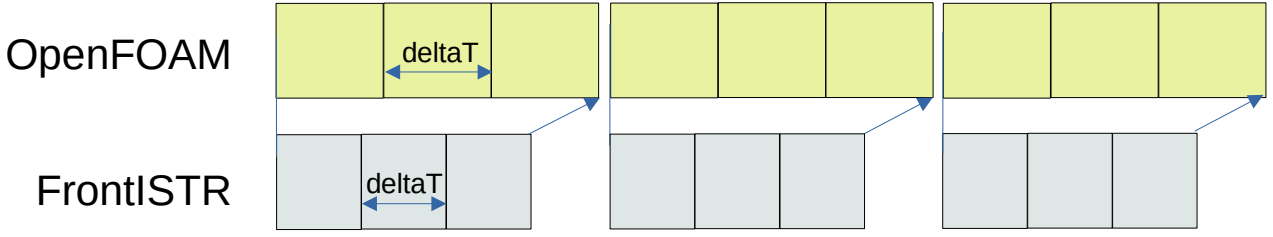
連成計算のイメージ（連成計算のタイミングをdeltaT✕3で行った場合）

従来の計算



合計の計算時間は、OpenFOAM＋FrontISTRの処理時間になる。

今回の計算（OpenFOAMとFrontISTRを同時に走らせる）



上記例の場合、FrontISTRの処理時間が無視でき、OpenFOAM単独の処理時間で済む。
（FrontISTRの処理時間 < OpenFOAMの処理時間の場合）

実現方法

	define patch at constant (boundary)	U (1)	p (2)	phi (3)	pointDisplacement (4)
				0.015378 0.0155268 0.0157116...	

flap	type wall;
	inGroups List<word> 1(wall);
		type movingWallVelocity;
		value uniform (0 0 0);
			type codedMixed;
			refValue uniform 0;
			refGradient uniform 0;
			valueFraction uniform 0;
			value uniform 0;
			name codedMixed_pressCalcDisp;
			code
			#{
			//face中心座標を取得
			// file名を設定...
				type calculated;
				value nonuniform List<scalar>
				66
				(
				0.15444
				0.1548
				0.155534
				0.156674
				0.158273
				0.160411...
					type codedFixedValue;
					name coded_readDisp;
					value uniform (0 0 0);
					code
					#{
					//座標を取得
					// file名を設定
					int pNo = Pstream::myProcNo();
					char procNo[10];
					sprintf(procNo, "%d", pNo); //charのprocNo...

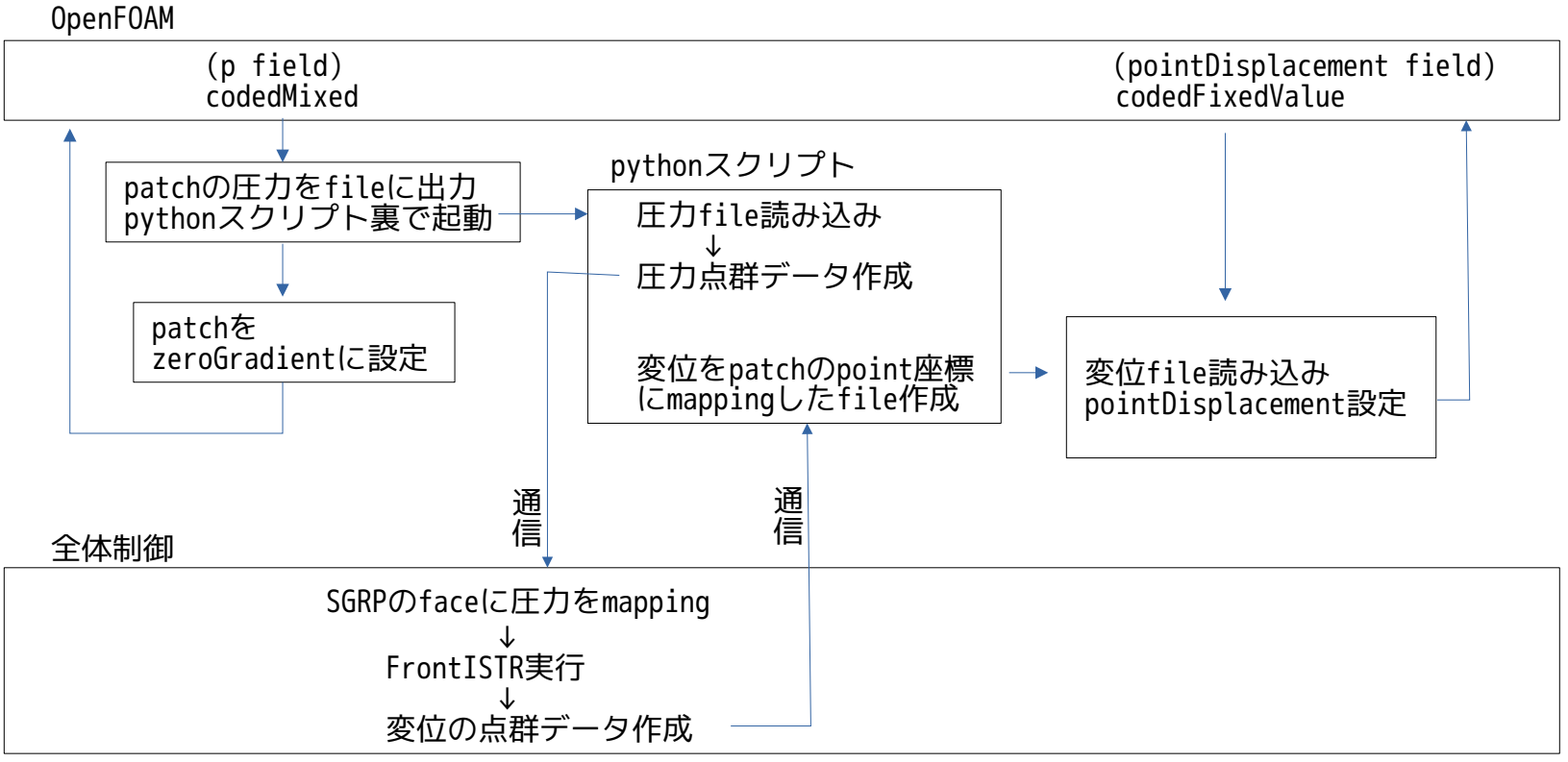
upperWall	type wall; inGroups List<word> 1(wall);	type noSlip;	type zeroGradient;	type calculated; value uniform 0;	type slip;
lowerWall	type wall; inGroups List<word> 1(wall);	type noSlip;	type zeroGradient;	type calculated; value uniform 0;	type fixedValue; value uniform (0 0 0);
frontAndBack	type wall; inGroups List<word> 1(wall);	type slip;	type slip;	type calculated; value uniform 0;	type slip;

```
    :
//変位を計算
char timeIndex[16];
sprintf(timeIndex, "%d", timeIdx);
word comm = "python3 coupling_FrontISTR/python/" +
            "patchRun_setPress_calcDisp.py";
comm += " " + word(tVal);        //現在のtime
comm += " " + word(btVal);       //前のtime
comm += " " + word(patchName);   //patch名
comm += " " + word(procNo);      //procNo
comm += " " + word(nProcs);      //全proc数
comm += " " + word(timeIndex);   //timeIndex(step数)
comm += " &";                    //裏で起動
char ccomm[comm.size() + 1];
strcpy(ccomm, comm.c_str());
int err = system(ccomm);         //シェル起動
    :
```

pythonスクリプトを裏で起動する。
起動後直ぐに戻ってくる為、
OpenFOAMは、直ぐに処理が継続
できる。

pythonスクリプトと全体制御で、
FrontISTRを起動して、変位を
計算し、変位のfileを作成する

連成するタイミングで出力されている
変位fileを読み込み、値をセットする。


OpenFOAM
(p field)
codedMixed
(pointDisplacement field)
codedFixedValue
patchの圧力をfileに出力
pythonスクリプト裏で起動
pythonスクリプト
圧力file読み込み
↓
圧力点群データ作成
patchを
zeroGradientに設定
変位をpatchのpoint座標
にmappingしたfile作成
変位file読み込み
pointDisplacement設定
通信
通信
全体制御
SGRPのfaceに圧力をmapping
↓
FrontISTR実行
↓
変位の点群データ作成

tutorialsに準備しているcase（solver）と連成内容

連成内容	solver（tutorials内のsolver）				備考
	pimpleFoam	interFoam	buoyant PimpleFoam	reactingFoam	
fsi 流体構造連成	○	○	○	×	dynamicMeshが扱えるsolver
cht 流体固体熱連成	×	×	○	○	温度T fieldを持つsolver
chtss 固体の熱ひずみ	×	×	○	×	温度T fieldを持ちdynamicMeshが扱える

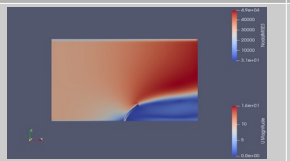
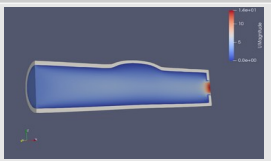
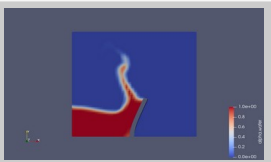
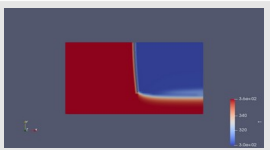
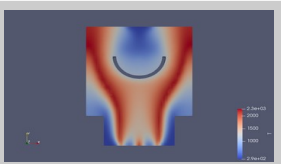
cht：流体固体の熱連成のみであれば、温度T field が扱えるsolverであれば連成できるので、
対象のsolverが多い。

他バージョンへの適用

作成した連成解析のtutorials内のcaseは、OpenFOAM-9で作成している。
これらtutorialsのcaseをOpenFOAM-v2206上での作動を確認し、作動するように改造している。

tutorials中では、pimpleFoam、interFoam、buoyantPimpleFoam、reactingFoamの例題を準備しているがpimpleFoam以外は、solverの仕様が微妙に異なっており、OF-9で作成したcaseがそのまま、OF-v2206上では動かず、修正を加えて準備している。
（pimpleFoamについては、同じcaseがOF-9、OF-v2206上で動く。）

tutorialsに準備しているcase

solver	case名	圧縮file名
pimpleFoam	flag_perp 	OF9用:flap_perp_OF9v2206-FrontISTR.zip v2204用: ↑
	softPipe 	OF9用:softPipe_OF9v2206-FrontISTR.zip v2204用: ↑
interFoam	waterToWall 	OF9用:waterToWall_OF9-FrontISTR.zip v2204用:waterToWall_OFv2206-FrontISTR.zip
buoyantPimpleFoam	biMetal 	OF9用:biMetal_OF9-FrontISTR.zip v2204用:biMetal_OFv2206-FrontISTR.zip
reactingFoam	fire 	OF9用:fire_OF9-FrontISTR.zip v2204用:fire_OFv2206-FrontISTR.zip