TreeFoamを使って chtMultiRegionFoamを動かす

chtMultiRegionFoam（流体・個体の熱連成解析）
を動かす為に、TreeFoamに機能追加
→　region（領域）が増えても処理ができる様に改造

1. TreeFoamの概要
2. 演習概要
   1) メッシュ作成
   2) データセット（setFields）
   3) 領域分割
   4) 各regionの設定
   5) 境界条件設定
   6) 計算開始、結果の確認
3. まとめ

regionが増えると設定が煩雑

アンデン（株）
藤井
14/08/16

1. TreeFoamの概要

OpenFOAMは、端末とテキストエディタによるCUIが基本。
・他人の操作を後ろから覗いても、何をしているのか解らない。
・しばらく使わないとコマンドやオプションを忘れ、効率ガタ落ち。
→ 初心者には、敷居が高い。

↓

少しでも操作性を改善し、直感的に操作が理解できるGUIを作成。
・後ろから覗けば、何をしているか、何となく解る

11/4月より作成し始め、約3年間、試行錯誤しながら作り上げ、現在も進行中。
OpenFOAM-1.6より作り始め、旧のバージョンが使える状態で、TreeFoam側のバージョンアップを繰り返してきたので、
OpenFOAM-1.6, 1.7, 2.0, 2.1, 2.2, 2.3
まで対応するはず。

(ver-1.6, 1.7 は、topoSetは、未対応。
ver-1.6, 1.7, 2.0 は、HelyxOS関連は未対応。)

1） TreeFoam本体の外観

視認性、操作性向上のため
folder（case）を**Tree表示**させ、case概要（solver名、結果有無など）を表示。

TreeFoam_2.11-140119 (0)

ファイル(F) case作成変更(M) 編集(E) 計算(C) ツール(T) ヘルプ(H)

TreeFoam上で
startFrom、stopAtが設定できる

case directory: ...aeuser/CAE/CAE-FOAM/fluidSolid-test/fluidSolid-mesh_copy0 OpenFoam環境: bashrc-FOAM-2.2.2

現在の解析case名: regCase-cht

solver: chtMultiRegionFoam

startFrom: latestTime stopAt: endTime:2 controlDict 編集

Tree表示

formatや並列数を表示
anP ：ascii、非圧縮、シングル処理
BCP2 ：binary、圧縮、2並列

Tree		BCPn	nR	st	ed
CAE-FOAM					
fluidManySolid					
fluidSolid-test					
fluidSolid-mesh	icoFoam	anP	1	0.0	
fluidSolid-mesh_copy0	chtMultiRegionFoam	BnP	1	0.0	
regCase-cht	chtMultiRegionFoam	BnP2	2	0.0	0.1
regCase-chtQv	chtQvMultiRegionFoam	BnP2	1	0.0	

log open /home/caeuser/TreeFoam/temp/0_logTreeFoam

TreeFoam ver 2.11-140119 (0) を起動しました。
OpenFOAM - 2.2.2

windowを分割してlogを表示
logを表示するかしないかは、configTreeFoamで設定

TreeFoamのアイコンも準備

2) TreeFoam本体のメニュー

メニューバー、メニューボタン操作、ダブルクリック操作がある
これらの操作は、**解析case(✅ マーク)に対する操作**

TreeFoam画面

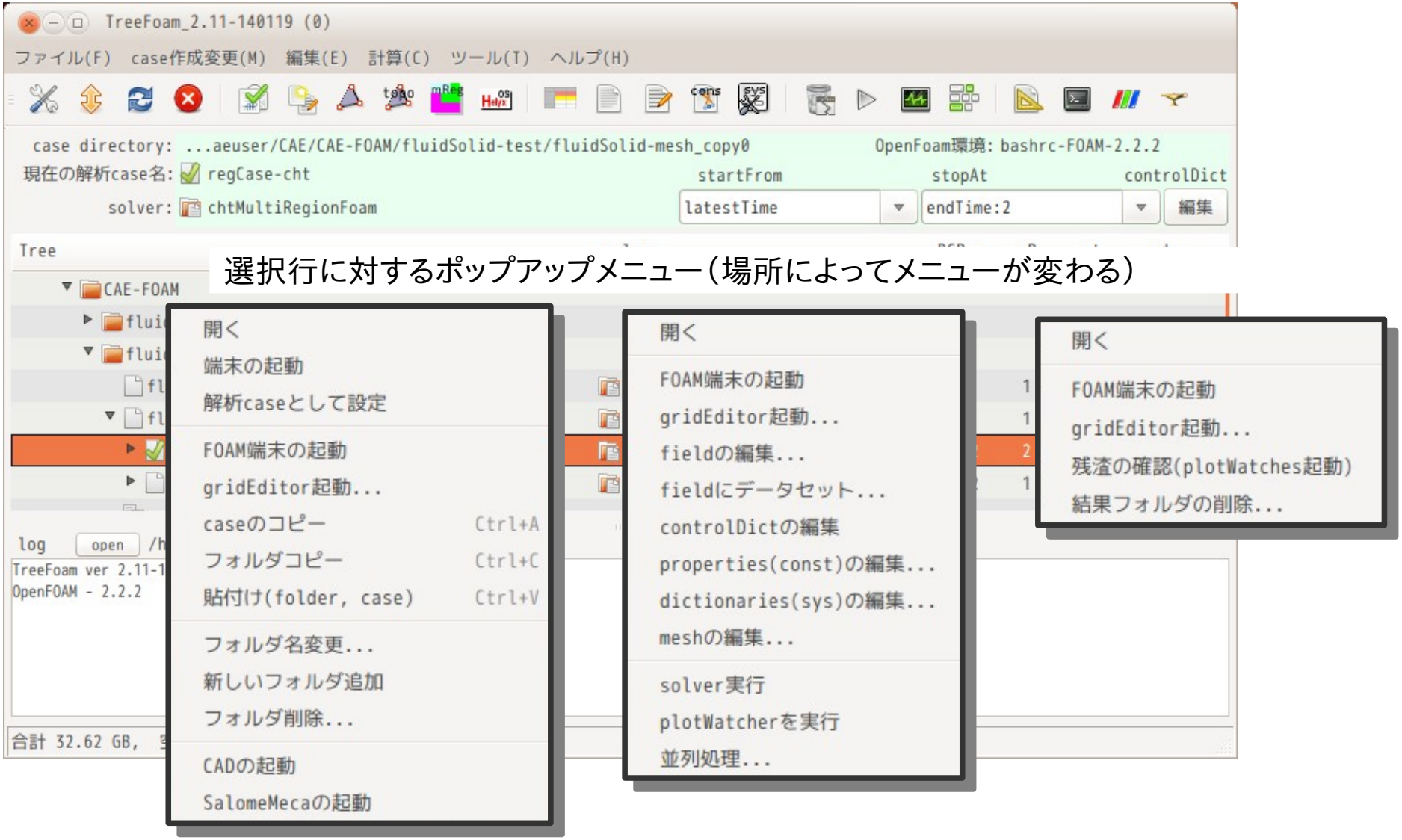
（マウスを合わせるとtoolTipを表示）

（マウスを合わせるとtoolTipを表示）

ポップアップメニュー（右クリック操作）

FOAM端末起動、gridEditor起動、case・folderのコピーなど操作できる
選択行に対する操作（解析caseではない）

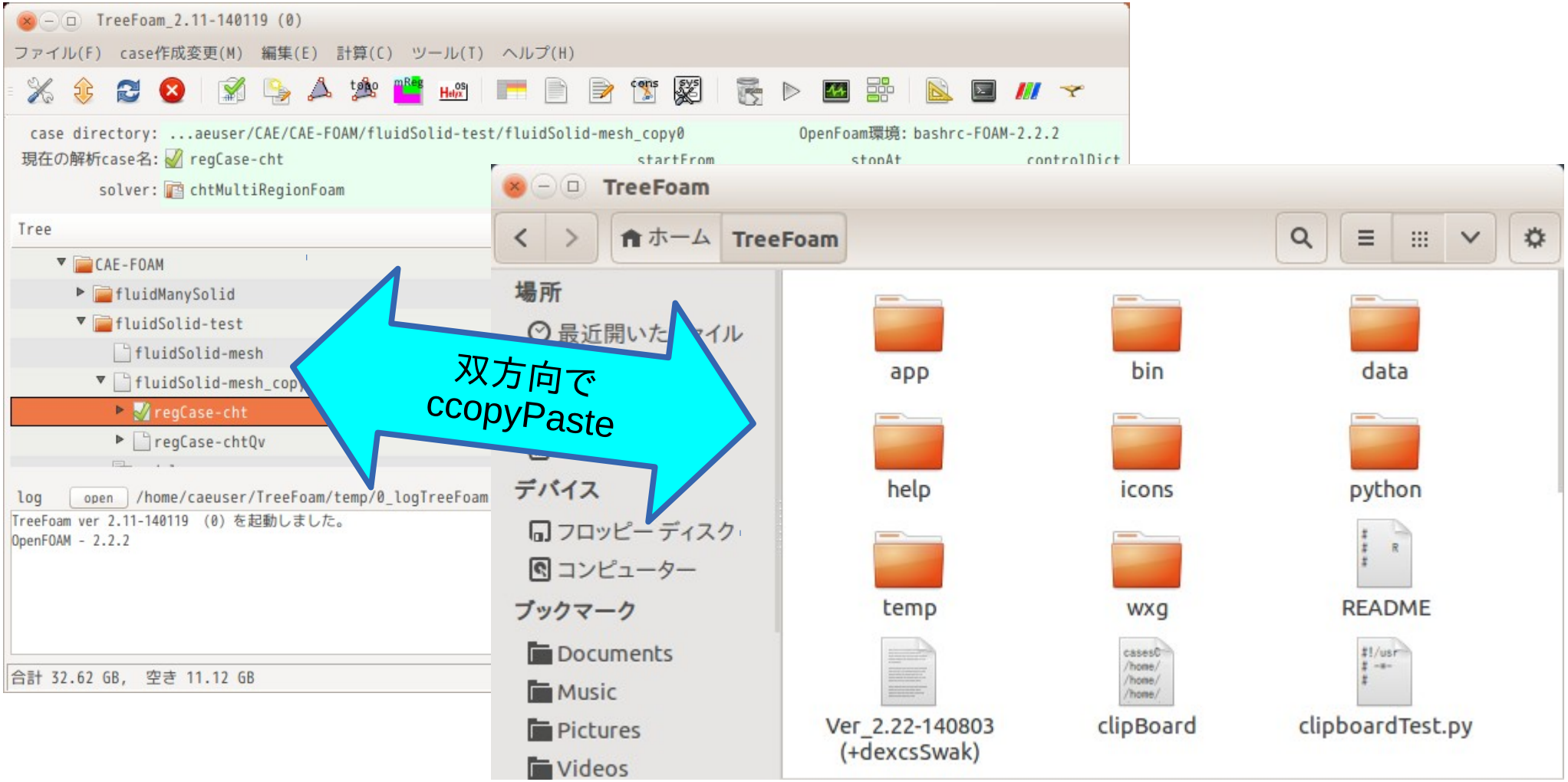
TreeFoam画面



folder、fileコピー

systemのclipBoardを使用

fileManeger（nautilus）とTreeFaom間でcopyPasteが可能



国際化の為、日本語・英語に対応。

3) gridEditor概要（境界条件の設定・確認）

境界条件の全貌が一覧表で確認できる（設定ミスが減る）
patch名の変更、boundaryFieldの確認・修正が表形式で可能になる
圧縮file、**binary** fileも扱える

gridEditor画面

ダブルクリックすると、Editorが開く

gridEditor: pitzDaily/0/. (0:0)

ファイル(F) 編集(E) 表示(V)

アルファベット順

アルファベット順

	define patch at constant (boundary)	U	epsilon	k	nuTilda	nut
field type dimensions		volVectorField; [0 1 -1 0 0 0 0];	volScalarField; [0 2 -3 0 0 0 0];	volScalarField; [0 2 -2 0 0 0 0];	volScalarField; [0 2 -1 0 0 0 0];	volScalarField; [0 2 -1 0 0 0 0
internal Field <sort patch>		uniform (0 0 0);	uniform 14.855;	uniform 0.375;	uniform 0;	uniform 0;
frontAndBack	type empty; inGroups 1(empty);	type empty;	type empty;	type empty;	type empty;	type empty;
inlet	type patch;	type fixedValue; value uniform (10 0 0);	type fixedValue; value uniform 14.855;	type fixedValue; value uniform 0.375;	type fixedValue; value uniform 0;	type calculated; value uniform 0
lowerWall	type wall;	type fixedValue; value uniform (0 0 0);	type epsilonWallFunction; value uniform 14.855;	type kqRWallFunction; value uniform 0.375;	type zeroGradient;	type nutkWallFunction; value uniform 0
outlet	type patch;	type zeroGradient;	type zeroGradient;	type zeroGradient;	type zeroGradient;	type calculated; value uniform 0
upperWall	type wall;	type fixedValue; value uniform (0 0 0);	type epsilonWallFunction; value uniform 14.855;	type kqRWallFunction; value uniform 0.375;	type zeroGradient;	type nutkWallFunction; value uniform 0

Field名

field type & dimension

internalField

Boundary Field

patch名

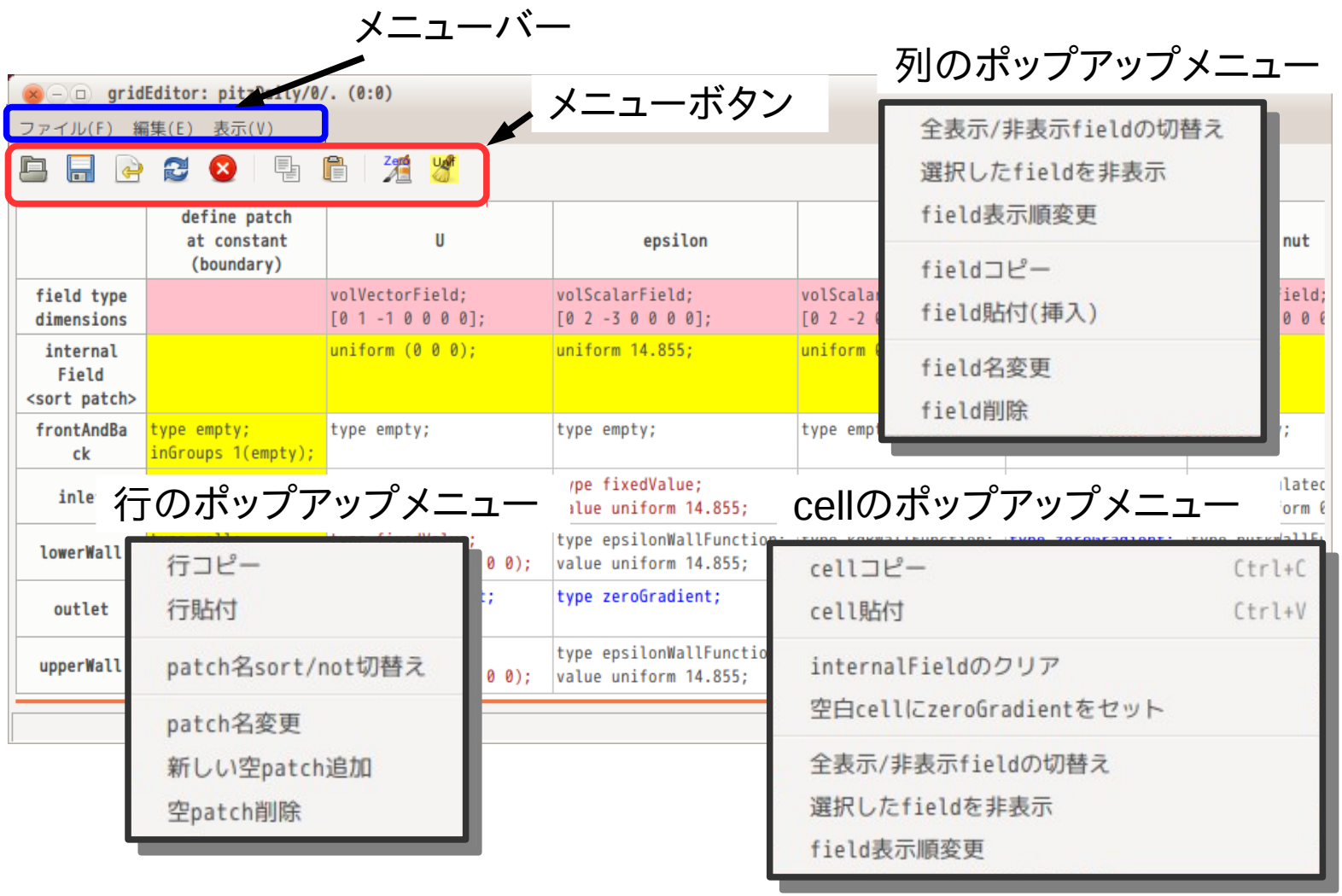
patchType

ダブルクリックすると、
patch名が変更できる

Excelの様にダブルクリックして、
cell内容を直接編集できる。

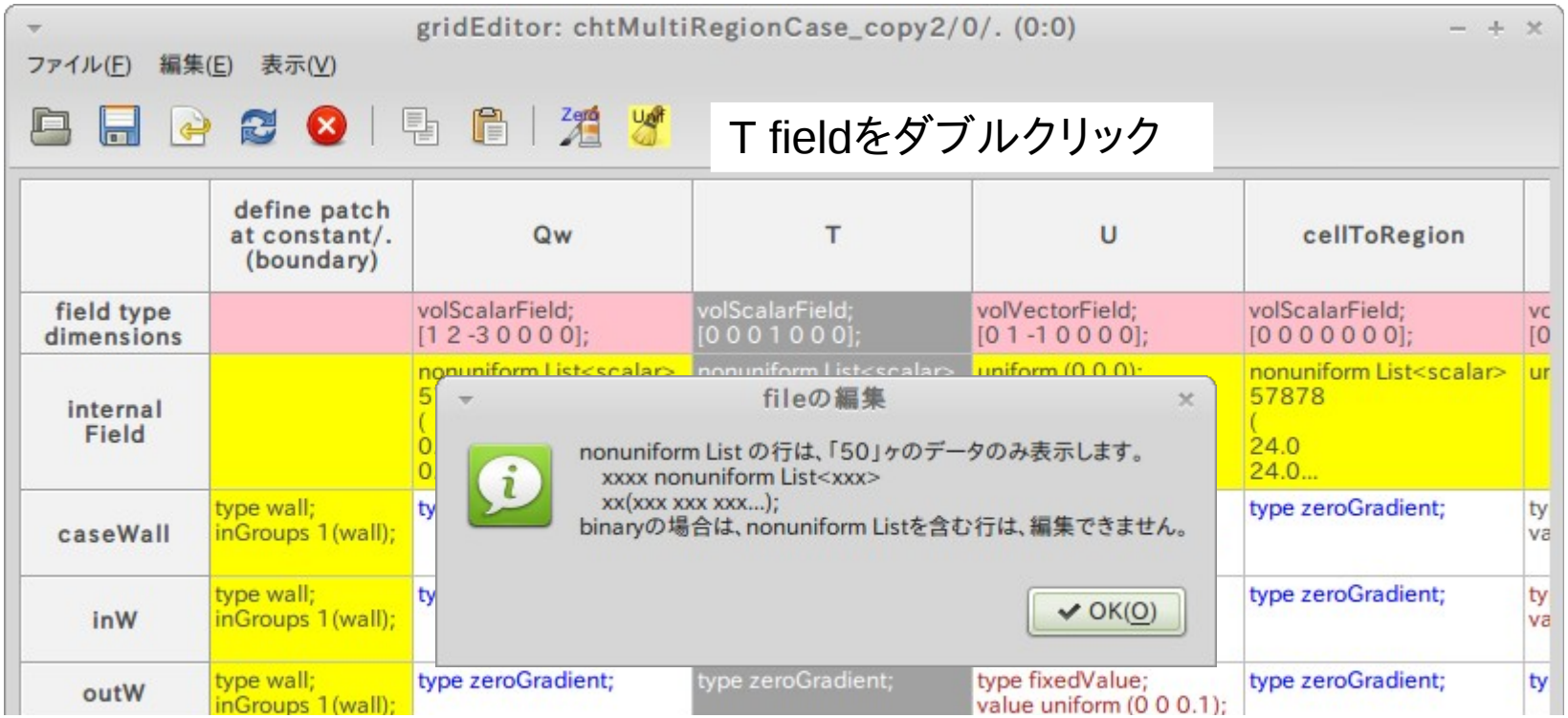
gridEditorのメニュー

・patch名の変更、空patchの追加、削除ができる。（行のポップアップメニュー）
・fieldの非表示、表示順の変更も可能。
（列のポップアップメニュー、起動時にfieldを選択）



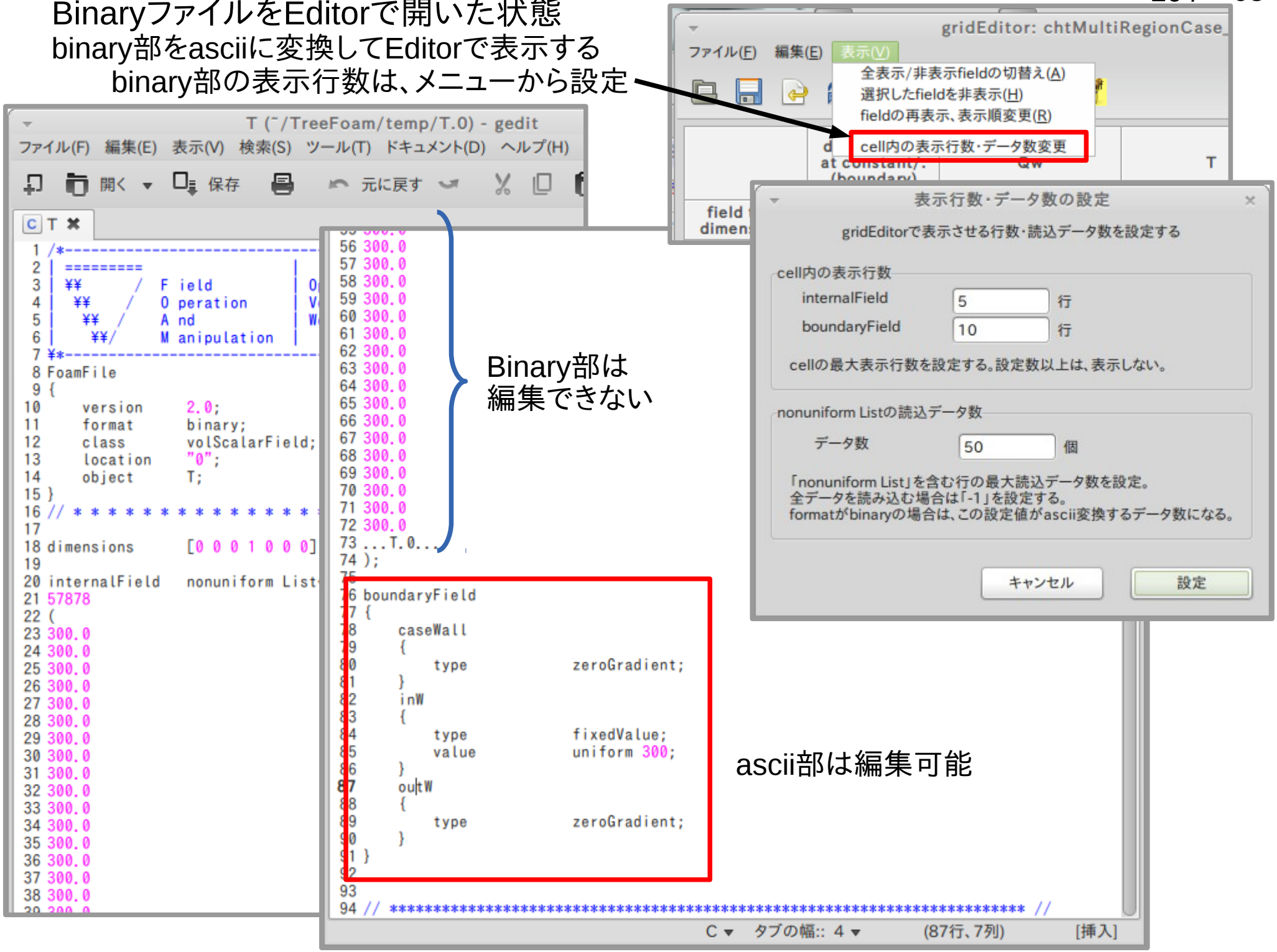
binaryファイルの扱い（Ediorで読み込み、編集する方法）

gridEditorで扱う場合：field名、または、cellをダブルクリックして読み込む



端末からコマンド入力で扱う場合：「editFoamFile.py <file名>」コマンドで読み込む

BinaryファイルをEditorで開いた状態
binary部をasciiに変換してEditorで表示する
binary部の表示行数は、メニューから設定



ワイルドカード、include、inGroupsの扱い

boundaryField内の

".*"

#include "${WM_PROJECT_DIR}/etc/caseDicts/setConstraintTypes"

inGroups 1(wall)

を解釈して、gridEditorで表示する。

tutorials/incompressible/pimpleDyMFoam/propeller の例

U fieldの内容

```
18 dim 0 0 0];
19
20 internalField   uniform (0 0 0);
21
22 boundaryField
23 {
24     //- Set patchGroups for constraint patches
25     #include "${WM_PROJECT_DIR}/etc/caseDicts/setConstraintTypes"
26
27     inlet
28     {
29         type
30         value
31     }
32
33     outlet
34     {
35         type
36         inletValue
37         value
38     }
39
40     outerCylinder
41     {
42         type
43         value
44     }
45
46     "propeller.*"
47     {
48         type
49         value
50     }
51 }
52
53
54 // ******************
```

boundaryの内容

```
53     p
54     {
55         type            wall;
56         inGroups        1(wall);
57         nFaces          576;
58         startFace       1600283;
59     }
60     propellerStem3
61     {
62         type            wall;
63         inGroups        1(wall);
64         nFaces          1536;
65         startFace       1600859;
66     }
67     AMI1
68     {
69         type            cyclicAMI;
70         inGroups        1(cyclicAMI);
71         nFaces          18496;
72         startFace       1602395;
73         matchTolerance  0.0001;
74         transform       noOrdering;
75         neighbourPatch  AMI2;
76     }
77     AMI2
78     {
```

gridEditorでの表示

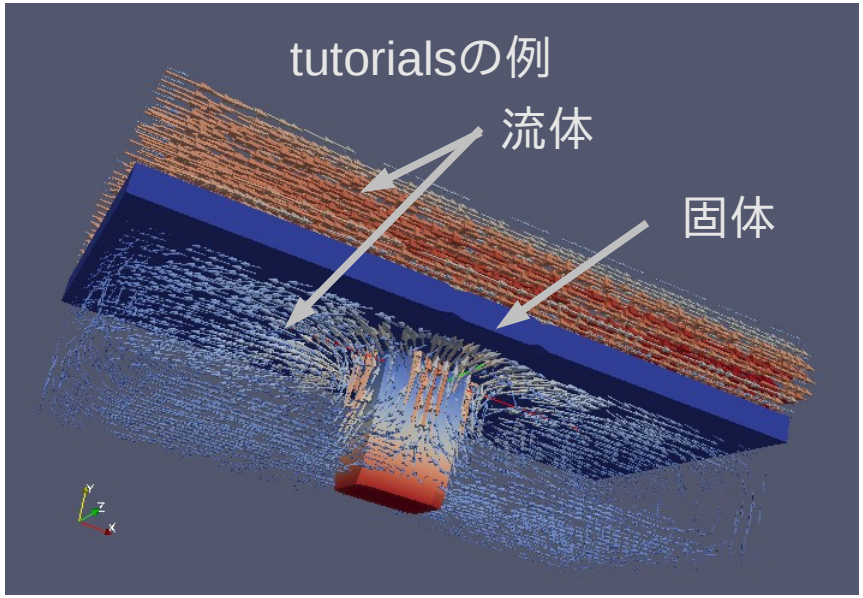
	define patch at constant/. (boundary)	U	epsilon
field type dimensions		volVectorField; [0 1 -1 0 0 0 0];	volScalarField; [0 2 -3 0 0 0 0];
internal Field		uniform (0 0 0);	uniform 0.0495;
inlet	type patch;	type fixedValue; value uniform (0 -5 0);	type fixedValue; value $internalFie
outlet	type patch;	type inletOutlet; inletValue uniform (0 0 0); value uniform (0 0 0);	type inletOutlet; inletValue $intern value $internalFie
outerCylinder	type wall; inGroups 1(wall);	type fixedValue; value uniform (0 0 0);	type epsilonWallF value $internalFie
propellerTip	type wall; inGroups 1(wall);	type movingWallVelocity; value uniform (0 0 0);	type epsilonWallF value $internalFie
propellerStem1	type wall; inGroups 1(wall);	type movingWallVelocity; value uniform (0 0 0);	type epsilonWallF value $internalFie
propellerStem2	type wall; inGroups 1(wall);	type movingWallVelocity; value uniform (0 0 0);	type epsilonWallF value $internalFie
propellerStem3	type wall; inGroups 1(wall);	type movingWallVelocity; value uniform (0 0 0);	type epsilonWallF value $internalFie
AMI1	type cyclicAMI; inGroups 1(cyclicAMI); matchTolerance 0.0001; transform noOrdering; neighbourPatch AMI2;	type cyclicAMI;	type cyclicAMI;
AMI2	type cyclicAMI; inGroups 1(cyclicAMI); matchTolerance 0.0001; transform noOrdering; neighbourPatch AMI1;	type cyclicAMI;	type cyclicAMI;

2. 演習概要

chtMultiRegionFoam（流体・個体熱連成解析）について

利点　：本格的に固体・流体熱錬成解析を行うことができる。

欠点　：操作が煩雑で本格利用（多くの領域を含む）する為には、**根気と労力**が必要。



TreeFoamを改造し、領域が多くなっても楽に処理できるように修正

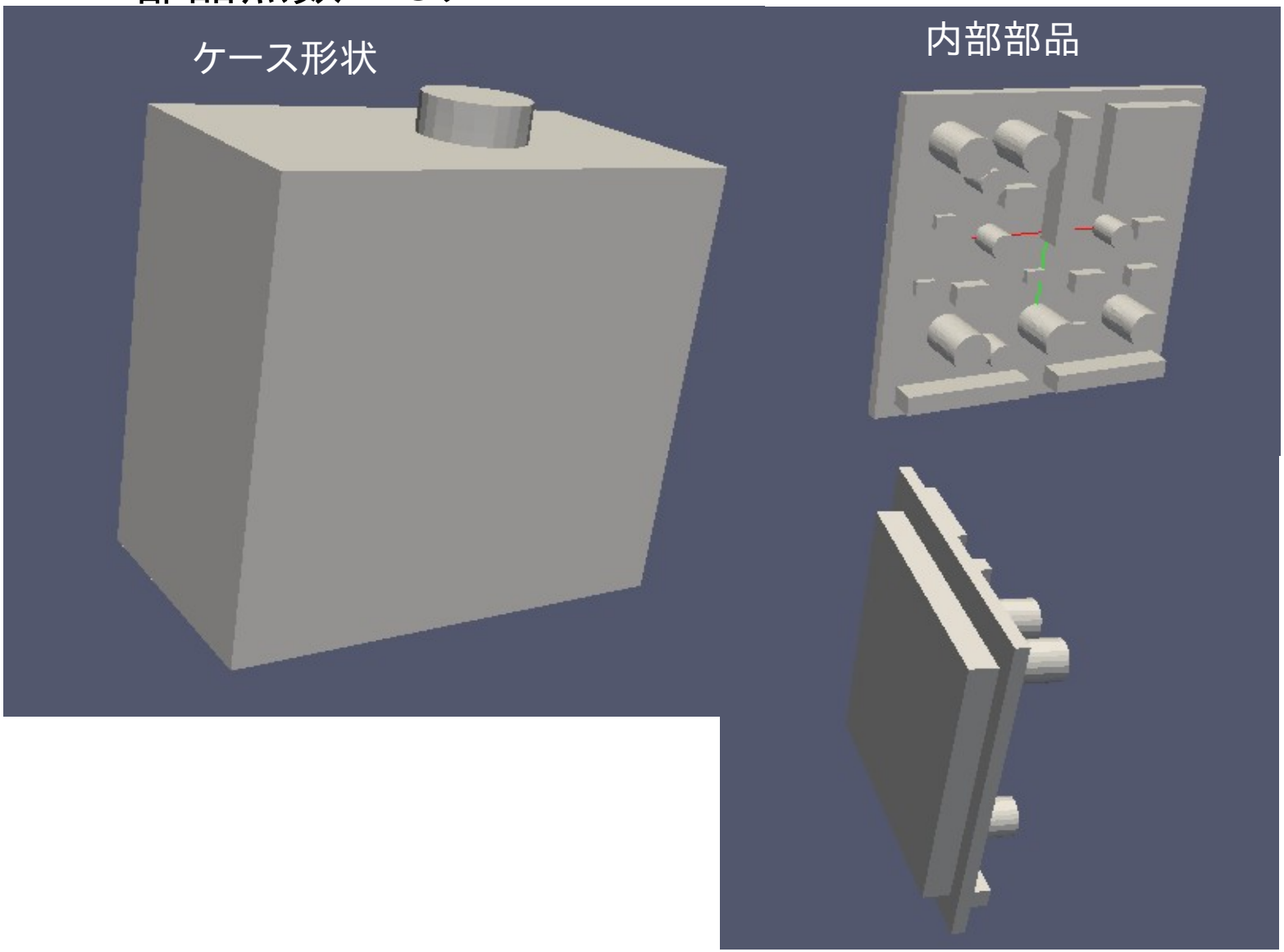
- メッシュ作成
- データセット（setFields）
- 領域分割
- 領域間の境界条件設定
- 結果の確認

領域（region）が多くなると操作が煩雑になってくる。

モデル形状

多くの部品（領域）を含むモデル

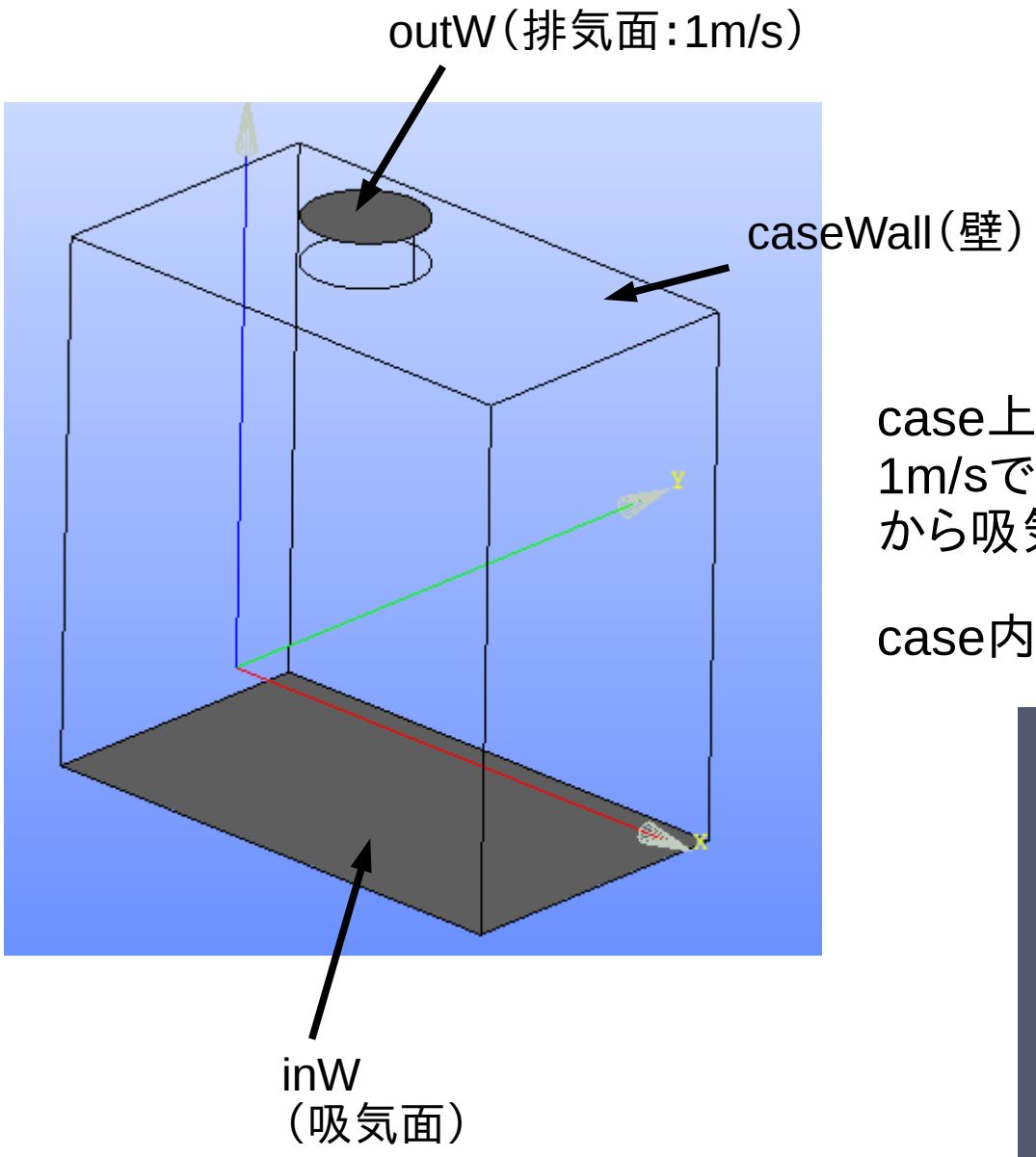
部品点数：25ヶ



ベースに多数の
部品配置

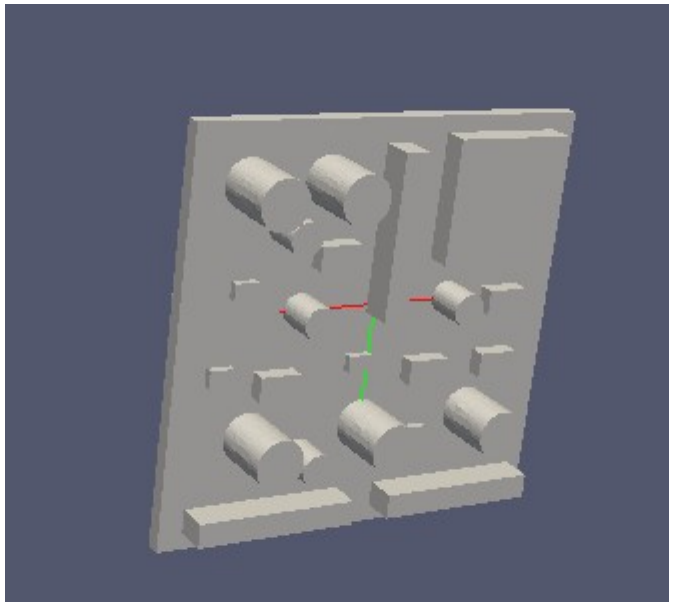
ベース裏面に
ヒートブロック配置

計算条件の設定内容



case上面の排気面（outW）から1m/sで排気し、case下面全体（inW）から吸気する構造。

case内部には、発熱部品を配置



演習の流れ

1) stlファイル作成
   モデルのpatch（surface）、領域（solid）のstlファイルを作成
   → **作成済み**

2) **メッシュ作成**
   → cellサイズを指定してメッシュを作成（csvファイルを作成）
   csvファイルからblocjMeshDict、snappyHexMeshDictをつくりだす

3) **データセット（setFields）**
   → 各領域に初期温度をセット

4) **領域分割（splitMeshRegions）**
   → cellZone毎に領域分割
   領域間の境界面の境界条件を設定

5) **各regionの設定**
   → 材料のDBから材料設定

6) **境界条件設定**
   → 境界条件のみを保存、設定

7) 計算開始、結果の確認

8) 見本となるcase（templateCase）の作成

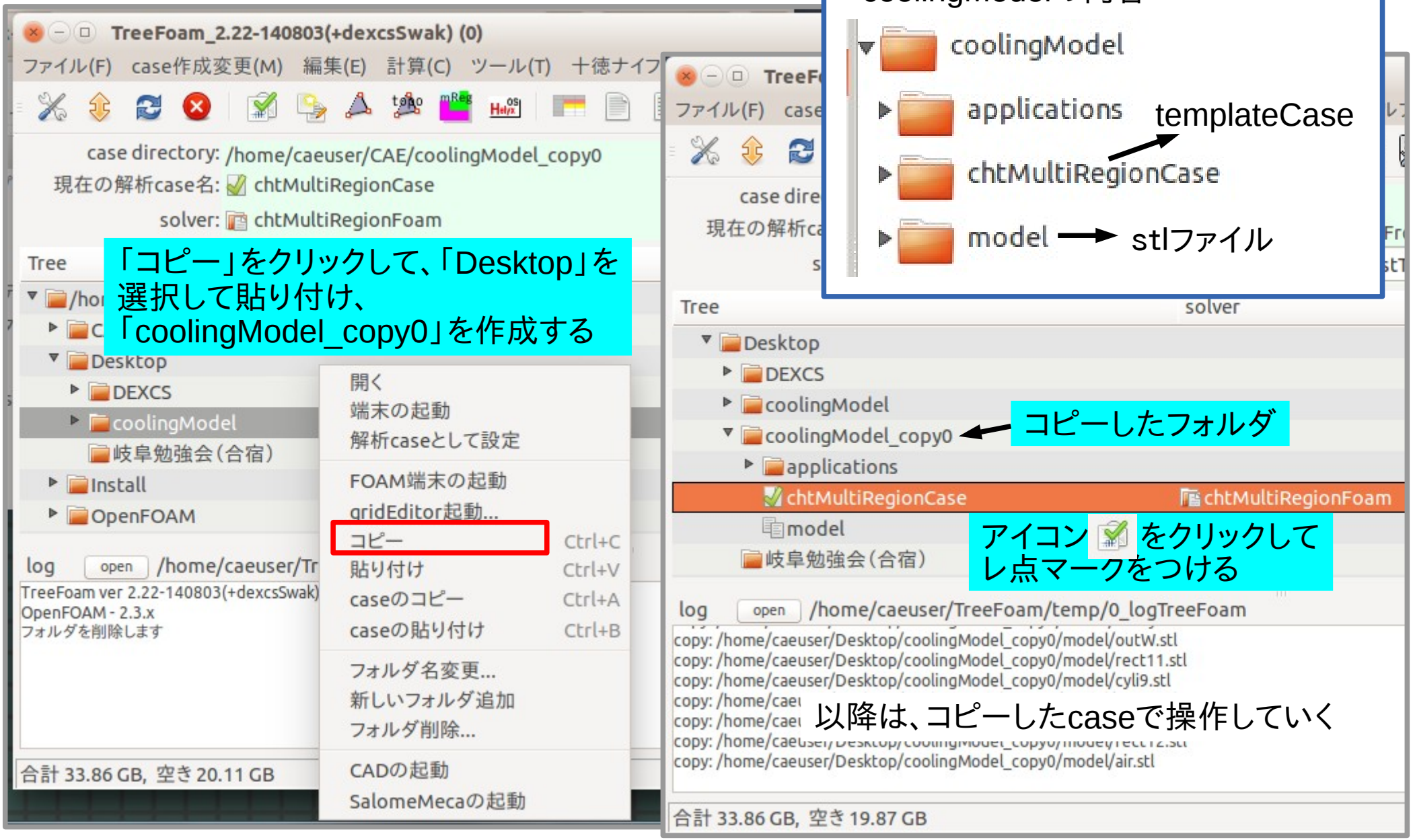
9) 部品の発熱を考慮した場合の計算

TreeFoamを改造して
簡単に設定が可能

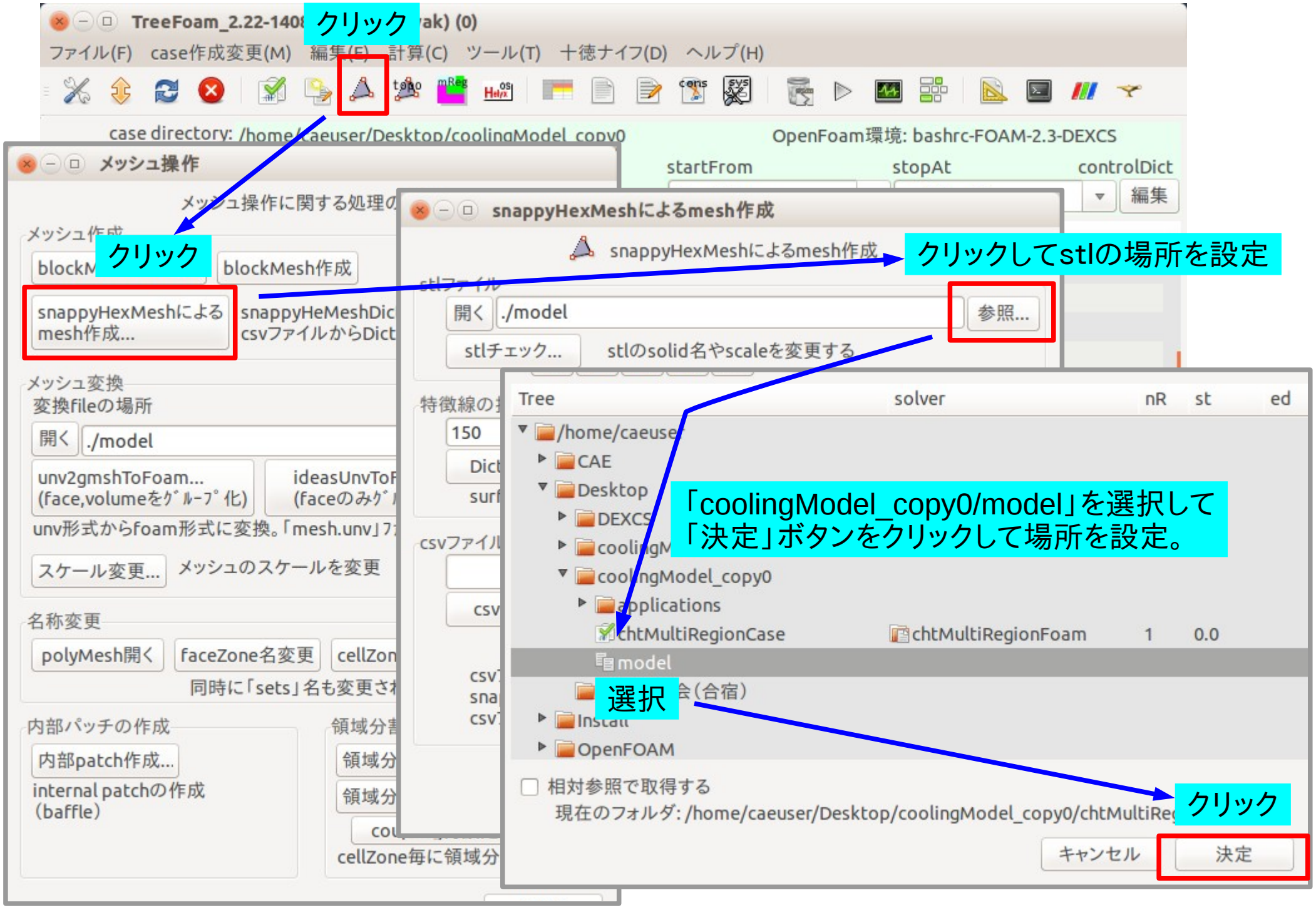
- メッシュ作成
- データセット
- 領域分割
- 各regionの設定
- 境界条件の設定

2） メッシュ作成

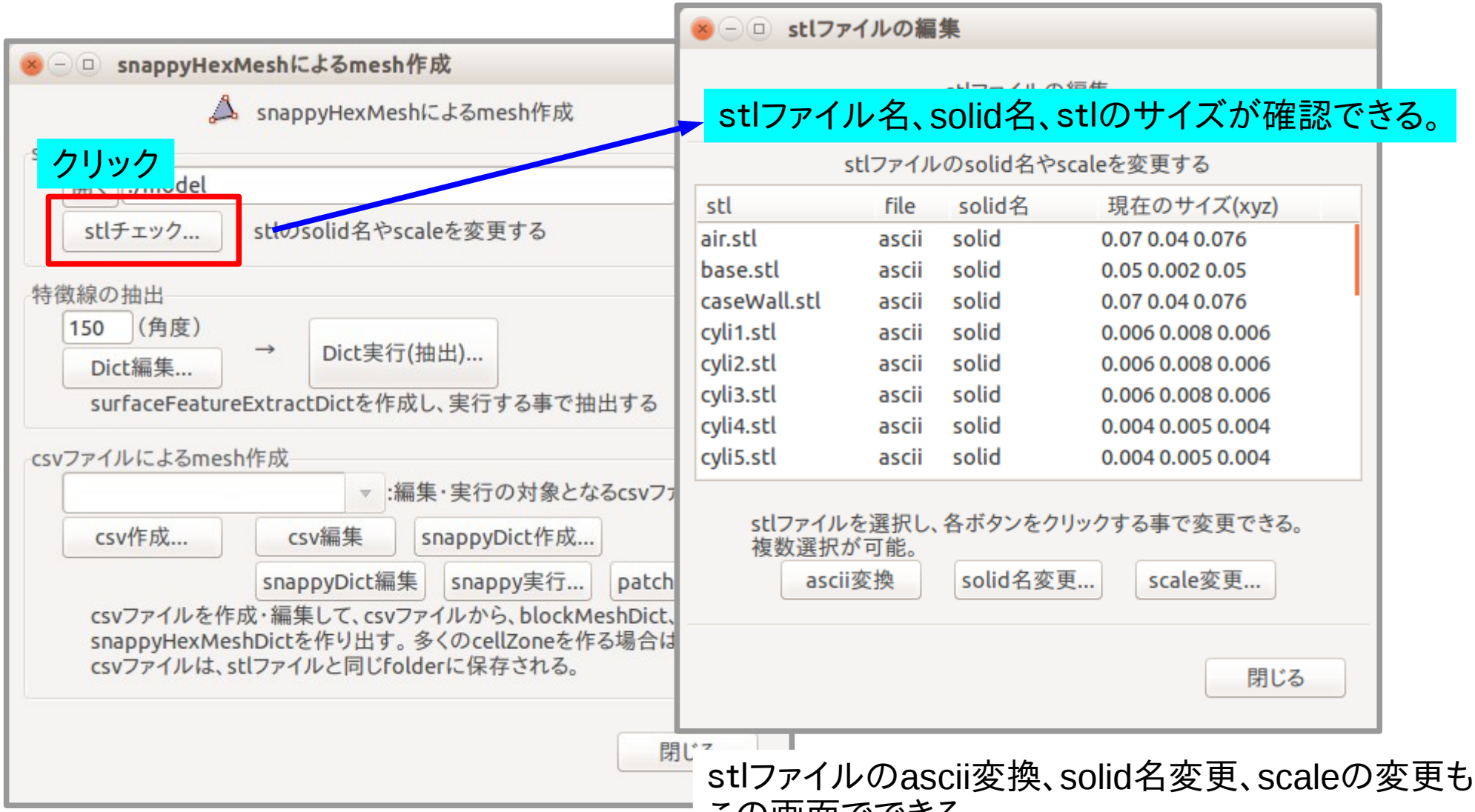
Desktop上の「coolingModel」のコピーを作成する。
コピーした「coolingModel_copy0/chtMultiRegionCase」を解析caseに設定する
（レ点マークをつける）
　　コピーしたcase内にメッシュを作成する。



stlファイルの場所を設定する



stlファイル内容を確認



stl	file	solid名	現在のサイズ(xyz)
air.stl	ascii	solid	0.07 0.04 0.076
base.stl	ascii	solid	0.05 0.002 0.05
caseWall.stl	ascii	solid	0.07 0.04 0.076
cyli1.stl	ascii	solid	0.006 0.008 0.006
cyli2.stl	ascii	solid	0.006 0.008 0.006
cyli3.stl	ascii	solid	0.006 0.008 0.006
cyli4.stl	ascii	solid	0.004 0.005 0.004
cyli5.stl	ascii	solid	0.004 0.005 0.004

stlファイルのascii変換、solid名変更、scaleの変更も、この画面でできる

＜ascii形式＞

solid名

1行目：solid solid
2行目： facet normal -0.996655 0 0.0817245
3行目： outer loop
4行目： :

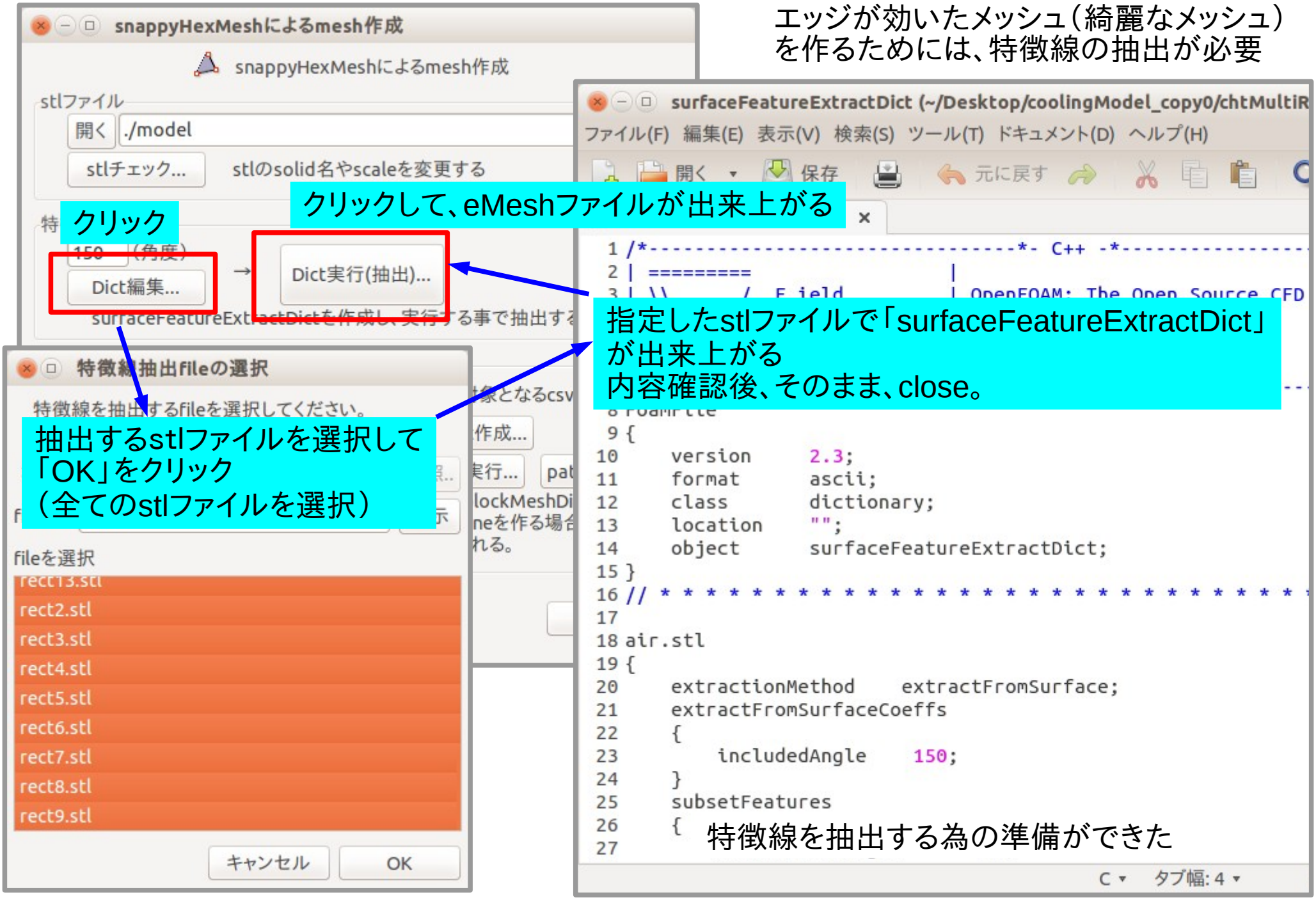
＜binary形式＞

solid名

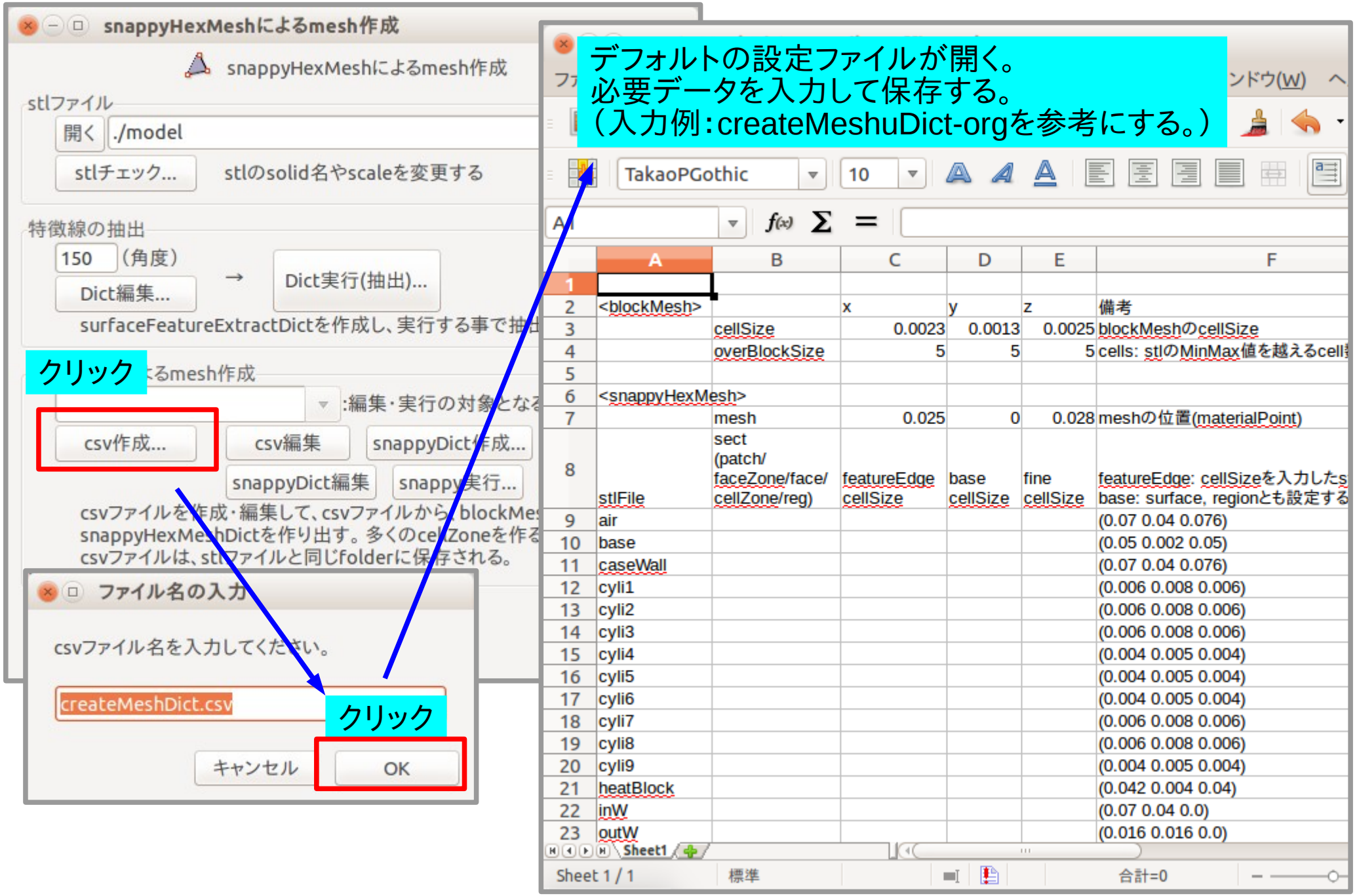
80 byte：header（string）
4 byie：三角形の数（int）
4 byte：三角形の向き（float）
4 byte： :

特徴線を抽出する為のeMeshファイルを作成

エッジが効いたメッシュ（綺麗なメッシュ）を作るためには、特徴線の抽出が必要

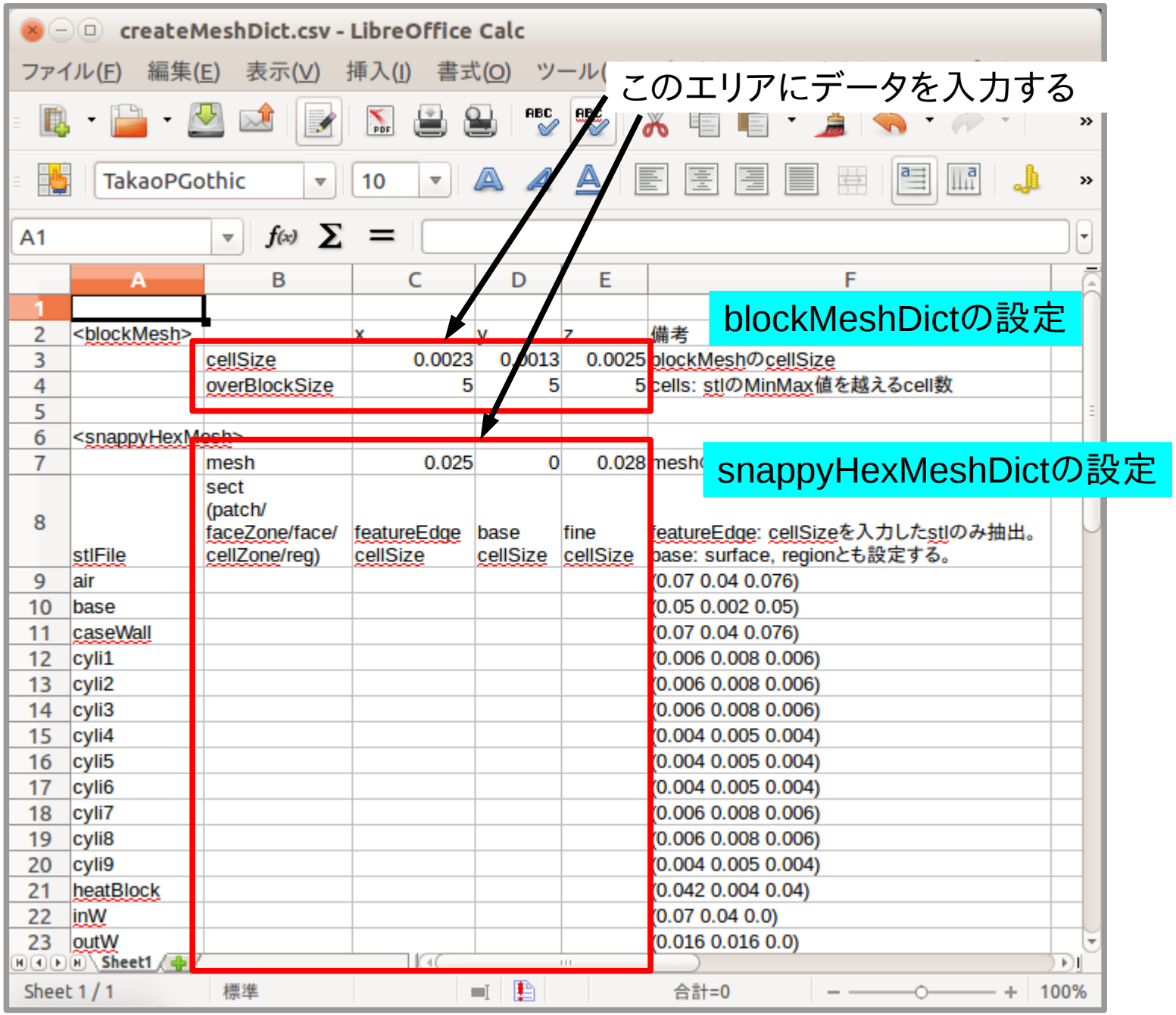


メッシュ作成する為のcsvファイルを作成する。



	A	B	C	D	E	F
1						
2	<blockMesh>		x	y	z	備考
3		cellSize	0.0023	0.0013	0.0025	blockMeshのcellSize
4		overBlockSize	5	5	5	cells: stlのMinMax値を越えるcell
5						
6	<snappyHexMesh>					
7		mesh	0.025	0	0.028	meshの位置(materialPoint)
8	stlFile	sect (patch/ faceZone/face/ cellZone/reg)	featureEdge cellSize	base cellSize	fine cellSize	featureEdge: cellSizeを入力した base: surface, regionとも設定する
9	air					(0.07 0.04 0.076)
10	base					(0.05 0.002 0.05)
11	caseWall					(0.07 0.04 0.076)
12	cyli1					(0.006 0.008 0.006)
13	cyli2					(0.006 0.008 0.006)
14	cyli3					(0.006 0.008 0.006)
15	cyli4					(0.004 0.005 0.004)
16	cyli5					(0.004 0.005 0.004)
17	cyli6					(0.004 0.005 0.004)
18	cyli7					(0.006 0.008 0.006)
19	cyli8					(0.006 0.008 0.006)
20	cyli9					(0.004 0.005 0.004)
21	heatBlock					(0.042 0.004 0.04)
22	inW					(0.07 0.04 0.0)
23	outW					(0.016 0.016 0.0)

デフォルトのcsvファイル内容



	A	B	C	D	E	F
1						
2	<blockMesh>		x	y	z	備考
3		cellSize	0.0023	0.0013	0.0025	blockMeshのcellSize
4		overBlockSize	5	5	5	cells: stlのMinMax値を越えるcell数
5						
6	<snappyHexMesh>					
7		mesh	0.025	0	0.028	mesh(
8	stlFile	sect (patch/ faceZone/face/ cellZone/reg)	featureEdge cellSize	base cellSize	fine cellSize	featureEdge: cellSizeを入力したstlのみ抽出。 base: surface, regionとも設定する。
9	air					(0.07 0.04 0.076)
10	base					(0.05 0.002 0.05)
11	caseWall					(0.07 0.04 0.076)
12	cyli1					(0.006 0.008 0.006)
13	cyli2					(0.006 0.008 0.006)
14	cyli3					(0.006 0.008 0.006)
15	cyli4					(0.004 0.005 0.004)
16	cyli5					(0.004 0.005 0.004)
17	cyli6					(0.004 0.005 0.004)
18	cyli7					(0.006 0.008 0.006)
19	cyli8					(0.006 0.008 0.006)
20	cyli9					(0.004 0.005 0.004)
21	heatBlock					(0.042 0.004 0.04)
22	inW					(0.07 0.04 0.0)
23	outW					(0.016 0.016 0.0)

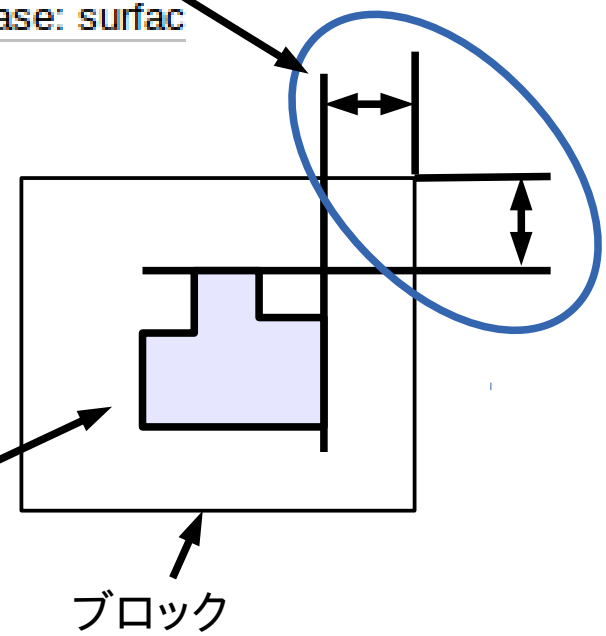
blockMeshDictの設定

セルサイズ　2mm

A	B	C	D	E	
<blockMesh>		x	y	z	備考
	cellSize	0.002			blockMeshの
	overBlockSize	5			cells: stlのM
<snappyHexMesh>					
	mesh	0.			
stlFile	sect (patch/ faceZone/face/ CellZone/reg)	featureEdge cellSize	base cellSize	fine cellSize	featureEdge: base: surfac

stlモデルサイズよりもcell数5ヶ分大きい
サイズでブロックを作成

通常：blockの座標とXYZの分割数を入力
→　座標を意識する必要がある
　　セルサイズが直感的にわかりにくい



座標を意識すること無く、
セルサイズを直接入力できる

snappyHexMeshでメッシュを作るのであれば、blockMeshの座標は、意味を持たない。

snappyHexMeshDictの設定

mesh内のポイント
（デフォルトで中心座標が設定される）

<snappyHexMesh>					
	mesh	0.025	0	0.028	meshの位置(materialPoint)
stlFile	sect (patch/ faceZone/face/ CellZone/reg)	featureEdge cellSize	base cellSize	fine cellSize	featureEdge: cellSizeを入力した base: surface, regionとも設定す
base	cellZone	0.001	0.001		(0.05 0.002 0.05)
caseWall	patch	0.002	0.002		(0.07 0.04 0.076)
cyli1	cellZone	0.001	0.001		(0.006 0.008 0.006)
cyli2	cellZone	0.001	0.001		(0.006 0.008 0.006)
cyli3	cellZone	0.001	0.001		(0.006 0.008 0.006)
rect9	cellZone	0.001	0.001		(0.005 0.002 0.003)
air	cellZone	0.002	0.002		(0.07 0.04 0.076)

cellZoneの大きさ

stlファイル名

区分
patch
faceZone
cellZone
face(面を定義：部分的にcellサイズを変更する場合に使用）
reg（領域を定義：部分的にcellサイズを変更する場合に使用）

特徴線の抽出
空白：抽出しない
値：抽出する

基本のcellサイズ
表面、内部共

最小cellサイズ
表面のcell

airは最終行に移動

airは外表面のstl為、
最終行にする事で、余りの
空間がairになる。

直接cellサイズを入力するように設定
（blockMeshサイズが変わっても影響を受けない）
データ入力後、csv形式で保存する。

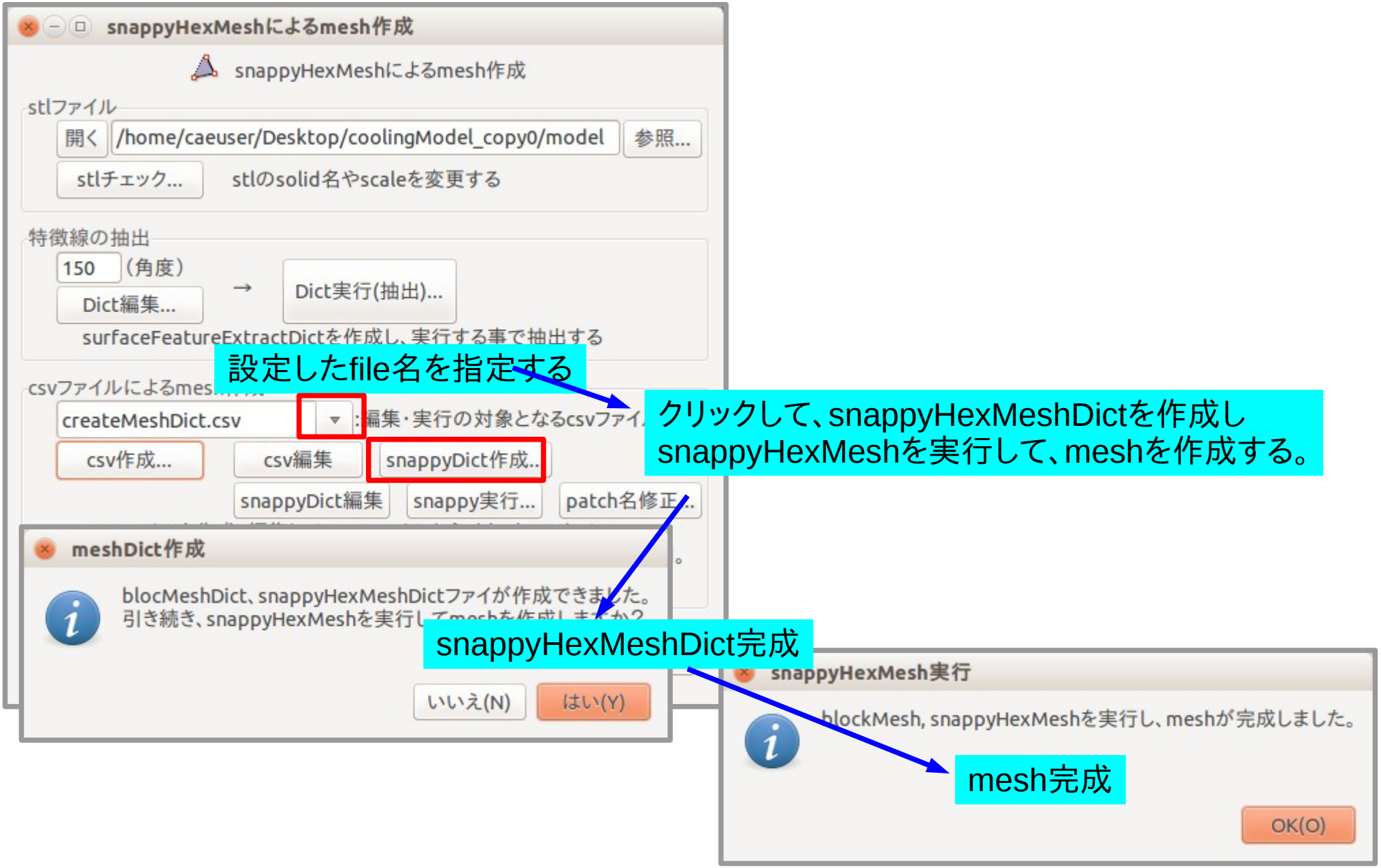
設定内容（1／2）

stlFile	sect (patch/ faceZone/face cellZone/reg)	featureEdge cellSize	base cellSize	fine cellSize	cellZoneの大きさ
base	cellZone	0.001	0.001		(0.05 0.002 0.05)
caseWall	patch	0.002	0.002		(0.07 0.04 0.076)
cyli1	cellZone	0.001	0.001		(0.006 0.008 0.006)
cyli2	cellZone	0.001	0.001		(0.006 0.008 0.006)
cyli3	cellZone	0.001	0.001		(0.006 0.008 0.006)
cyli4	cellZone	0.0005	0.0005		(0.004 0.005 0.004)
cyli5	cellZone	0.0005	0.0005		(0.004 0.005 0.004)
cyli6	cellZone	0.0005	0.0005		(0.004 0.005 0.004)
cyli7	cellZone	0.001	0.001		(0.006 0.008 0.006)
cyli8	cellZone	0.001	0.001		(0.006 0.008 0.006)
cyli9	cellZone	0.0005	0.0005		(0.004 0.005 0.004)
heatBlock	cellZone	0.001	0.001		(0.042 0.004 0.04)
inW	patch	0.002	0.002		(0.07 0.04 0.0)
outW	patch	0.002	0.002		(0.016 0.016 0.0)

設定内容（2／2）

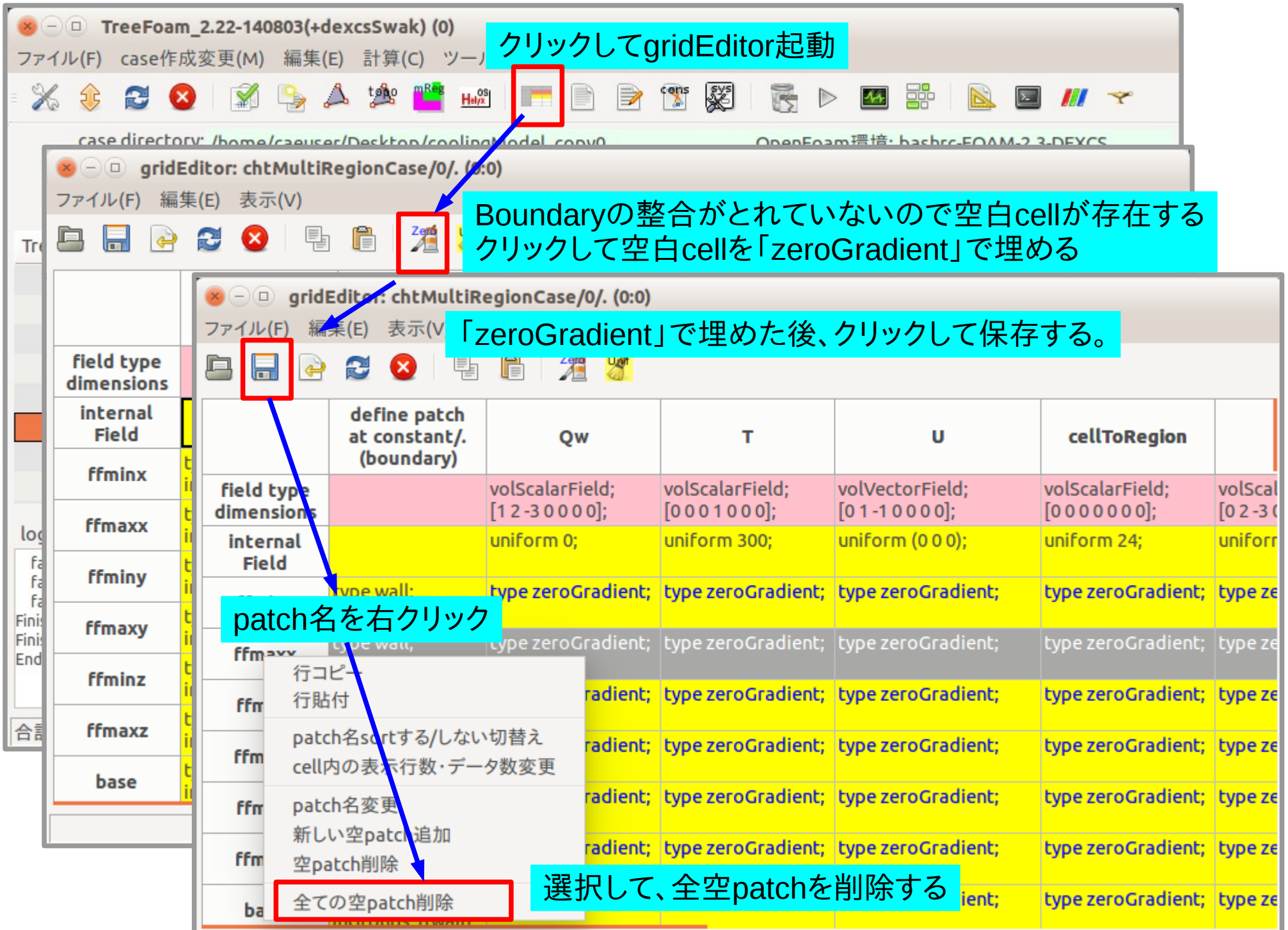
stlFile	sect (patch/ faceZone/face cellZone/reg)	featureEdge cellSize	base cellSize	fine cellSize	cellZoneの大きさ
rect1	cellZone	0.001	0.001		(0.02 0.004 0.003)
rect10	cellZone	0.0005	0.0005		(0.003 0.001 0.002)
rect11	cellZone	0.001	0.001		(0.015 0.003 0.015)
rect12	cellZone	0.001	0.001		(0.005 0.002 0.003)
rect13	cellZone	0.0005	0.0005		(0.003 0.001 0.002)
rect2	cellZone	0.001	0.001		(0.02 0.004 0.003)
rect3	cellZone	0.001	0.001		(0.005 0.002 0.003)
rect4	cellZone	0.001	0.001		(0.005 0.002 0.003)
rect5	cellZone	0.001	0.001		(0.005 0.002 0.003)
rect6	cellZone	0.0005	0.0005		(0.003 0.001 0.002)
rect7	cellZone	0.001	0.001		(0.003 0.004 0.02)
rect8	cellZone	0.0005	0.0005		(0.003 0.001 0.002)
rect9	cellZone	0.001	0.001		(0.005 0.002 0.003)
air	cellZone	0.002	0.002		(0.07 0.04 0.076)

meshを作成する。



エラーが発生する場合は、case内の「blockMeshDict」、「snappyHexMeshDict」を削除して、再確認する。（削除すると、実行時にデフォルトのDictファイルに置き換えられる。）

boundaryの整合と空patchの削除



最終的な内容を保存

nCase/0/. (0:0)

ファイル(F) 編集(E) 表示(V)

	define patch at constant/. (boundary)	Qw	
field type dimensions		volScalarField; [1 2 -3 0 0 0 0];	vo [0
internal Field		uniform 0;	ur
caseWall	type wall; inGroups 1(wall);	type zeroGradient;	ty
inW	type wall; inGroups 1(wall);	type zeroGradient;	type fixedValue; value uniform 300;
outW			

同じ事（boundaryの整合、空patch削除）が、
「patch名修正...」ボタンでもできる

csvファイルによるmesh作成

createMeshDict.csv :編集・実行の対象となるcsvファイル名

csv作成... csv編集 snappyDict作成...

snappyDict編集 snappy実行... patch名修正...

csvファイルを作成・編集して、csvファイルから、blockMeshDict、snappyHexMeshDictを作り出す。多くのcellZoneを作る場合は、有用。
csvファイルは、stlファイルと同じfolderに保存される。

TreeFoam_2.22-140803(+dexcsSwak) (0)

ファイル(F) case作成変更(M) 編集(E) 計算(C) ツール(T) 十徳ナイフ(D) ヘルプ(H)

case directory: /home/caeuser/Desktop/coolingModel_copy0

現在の解析case名: chtMultiRegionCase

solver: chtMultiRegionFoam

startFrom

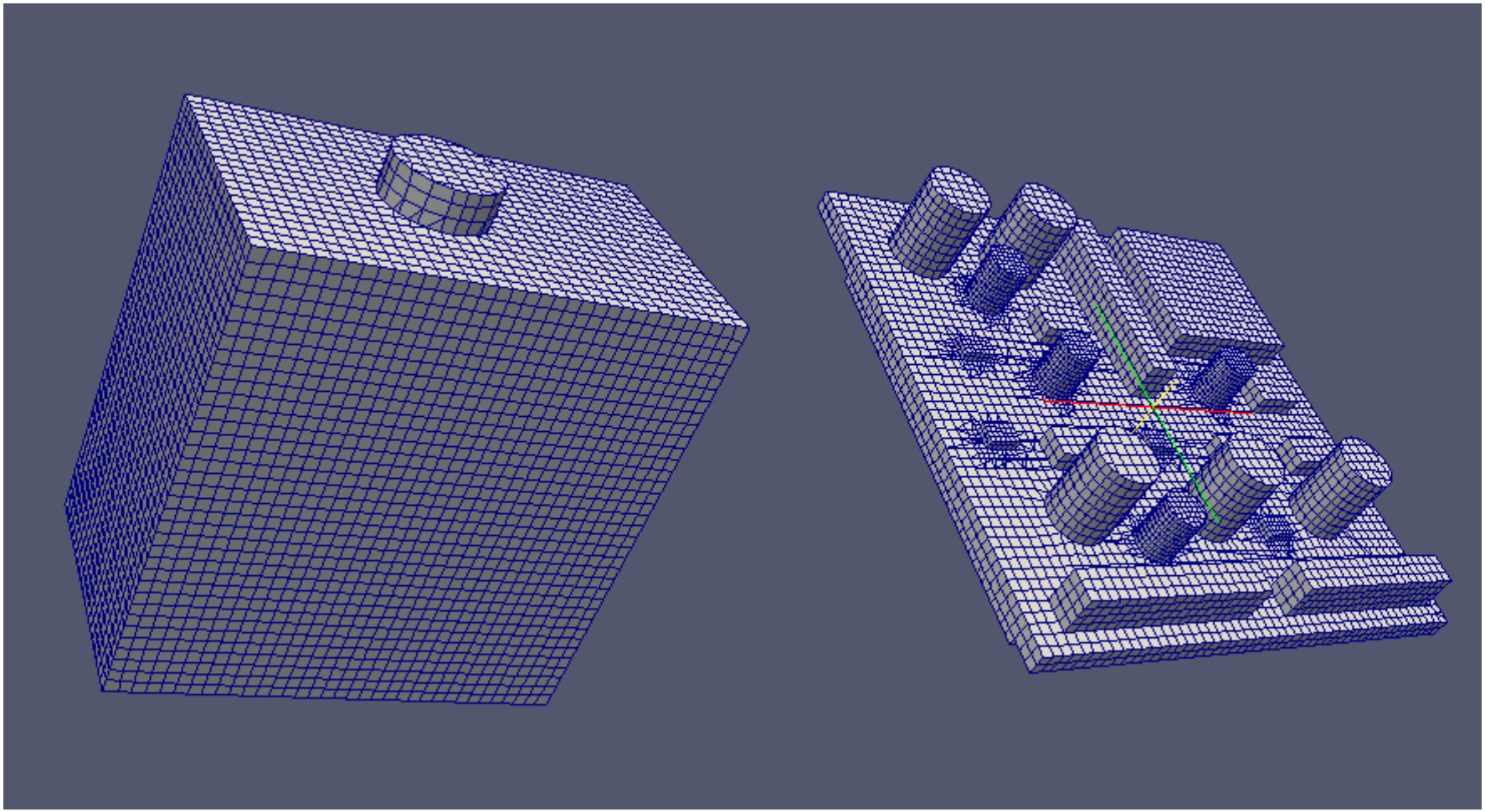
latestTime

クリックしてparaViewを起動してmesh確認

Tree solver BCPn nR st ed

- /home/caeuser
  - CAE
  - Desktop
    - DEXCS
    - coolingModel
    - coolingModel_copy0
      - applications

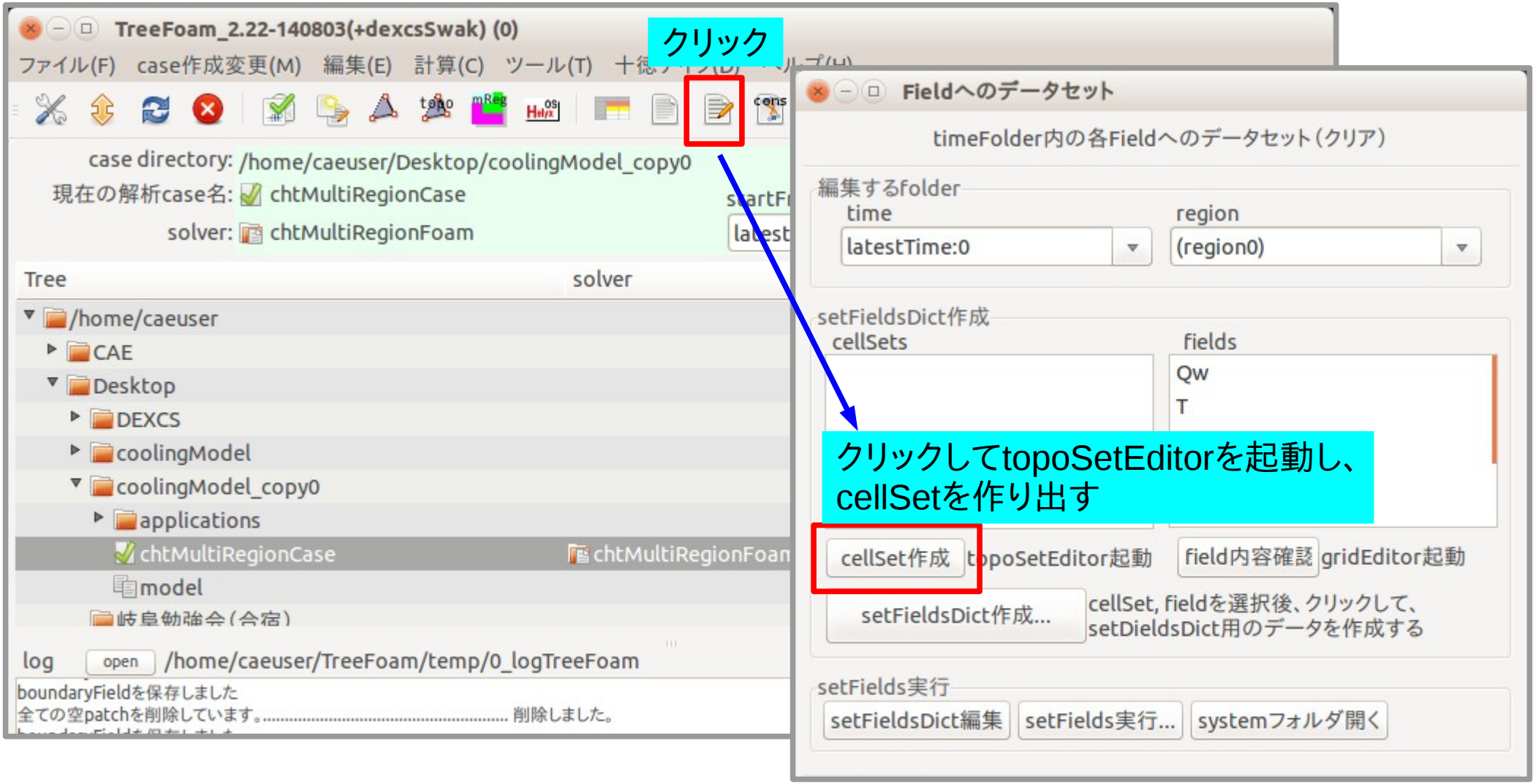
出来上がったmesh



このメッシュを使って、解析する。

3） データセット（setFields）

出来上がったメッシュにデータをセットする。
セットするfieldは、
　　T：温度（初期温度を設定）
　　Qw：発熱させるワット数（後で使用）
にセットする。
データセットするためには、cellSetが必要なので、cellZoneからcellSetを作り出す。



topoSetEditorを使って、cellZoneからcellSetを作り出す

mesh抽出 :coolingModel_copy0/chtMultiRegionCase

topoSet Editor（topoSetDictを作成し、meshを抽出）

time latestTime:0 region (region0) (-regionを設定して、topoSetを実行)

<Action> コマンド
source: new, add, delete, subset
no source: clear, invert, remove
combined: renameSetZone, new, newCellToFace, newZonesToSets

<Source> 入力 type mesh:constant
sets: cellSet, faceSet, pointSet
zones: cellZone, faceZone
幾何図形: box, cylinder, sphere
その他: surface ..., rotatedBox, region, field, patch, label, shape, normal, nearest

rect11, rect12, rect13, rect2, rect3, rect4, rect5, rect6, rect7, rect8, rect9
クリックして選択

<Result> 出力 type mesh:constant/.
sets: cellSet, faceSet, pointSet
zones: cellZoneSet
set name
↑ listから取得

code出力

コード確認、編集 topoSetDictクリア topoSetDict追加 topoSet実行 クリア.追加.実行

```
// new To cellSet
{
  name  air;
  type  cellSet;
  action new;
  // Cells in cell zone
  source zoneToCell;
  sourceInfo
  {
    name "air";    // Name of cellZone, regular expressions allowed
  }
```

<使用法>
・action（ｺﾏﾝﾄﾞ）を選択。
・source type, nameを選択。
・result type, nameを決定。
・「topoSetDictに追加」

topoSetDict編集

閉じる

全て選択

選択

選択

クリックしてコードを出力

順番にクリックしてtopoSet実行する
（topoSetの実行により、cellSetが出来上がる）

この内容が
topoSetDictの内容

TopoSetDictを作り、topoSetを実行。
CellSetができあがった

cellSetの領域にデータをセットする

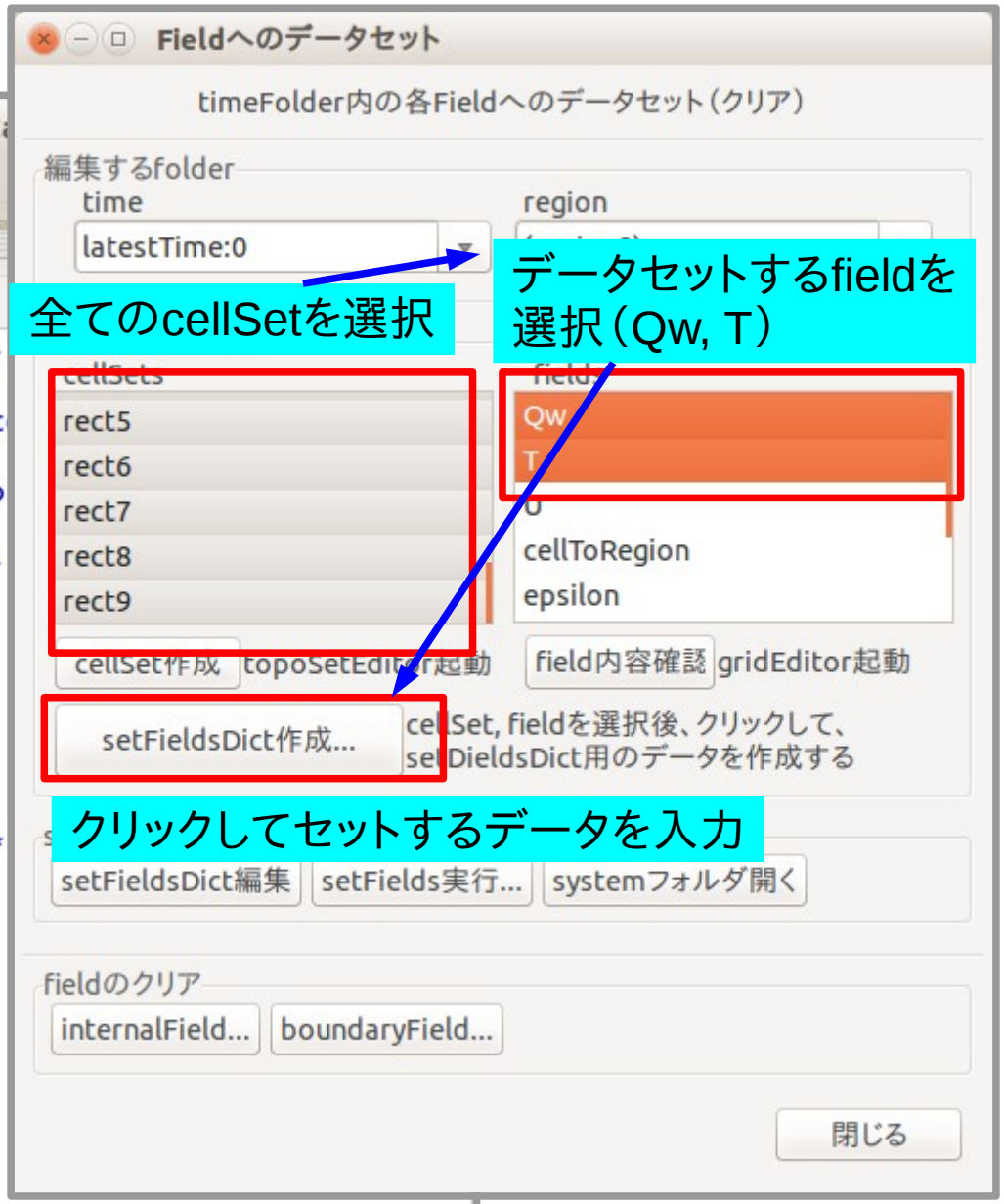
出来上がったtopoSetDict
この内容でtopoSetを実行した

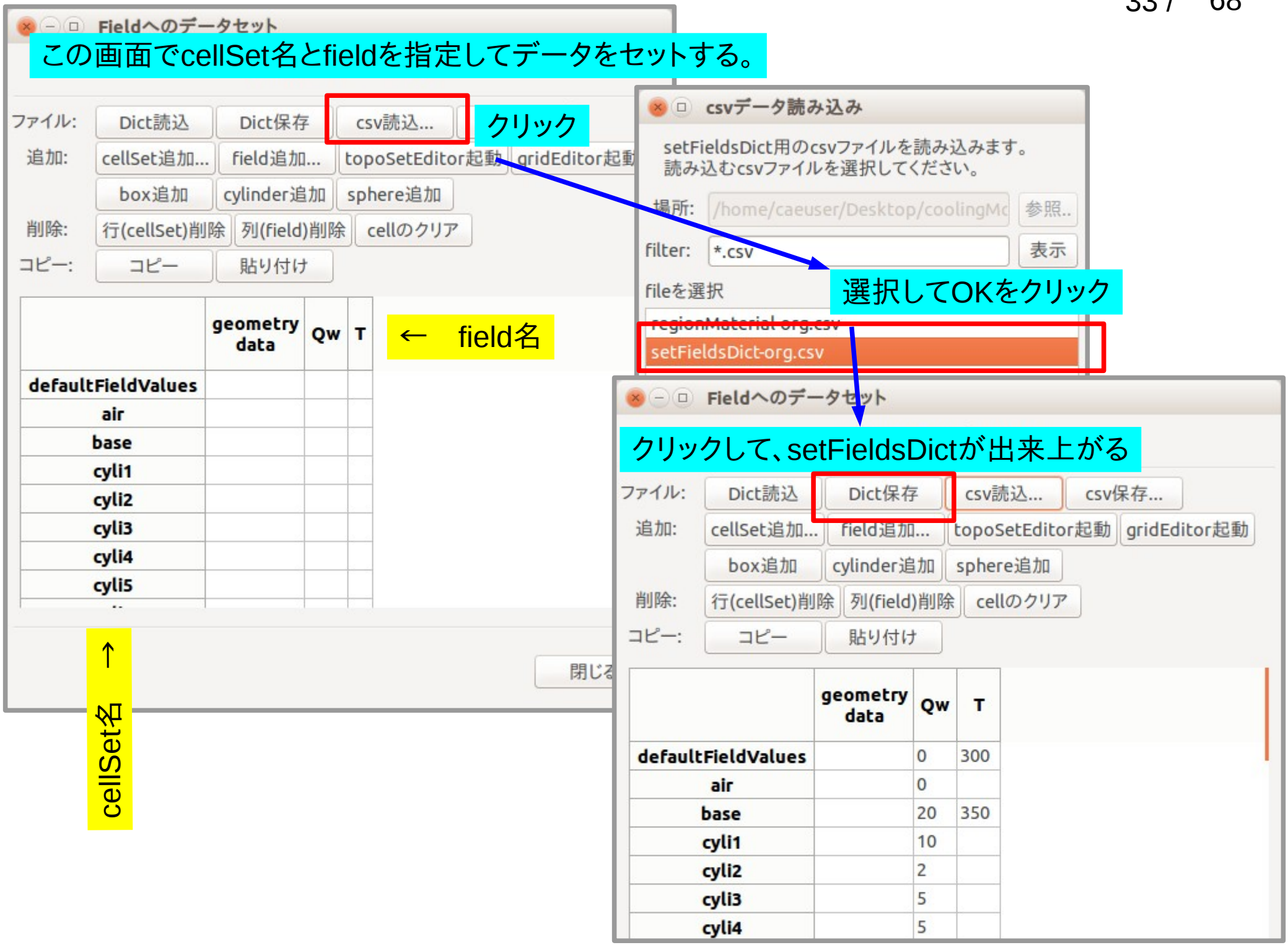
ファイル(F) (D) ヘルプ(H)

topoSetDict

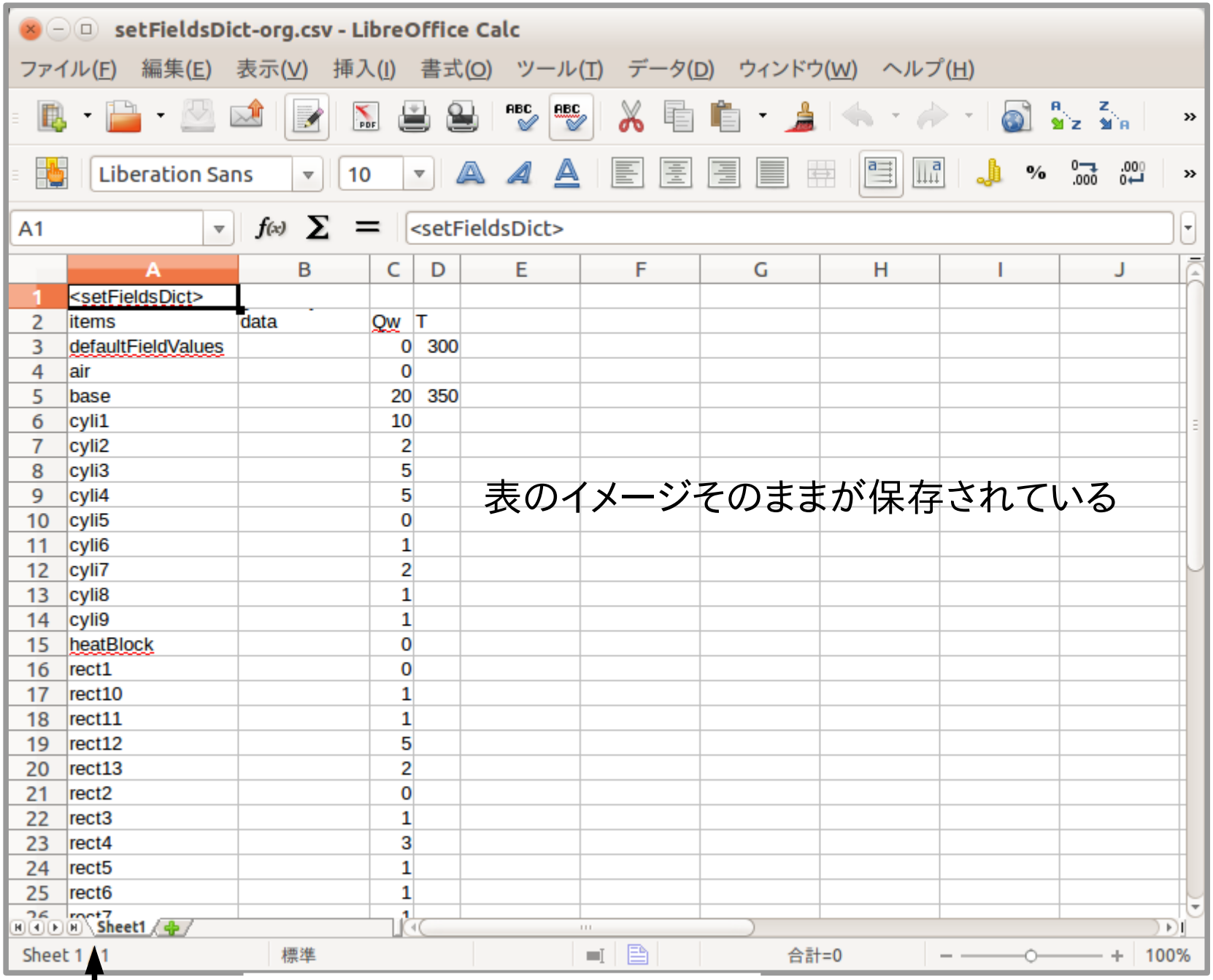
```
/*--------------------------------*- C++ -*----------
| =========                 |
| \\      /  F ield         | OpenFOAM: The Open Sourc
|  \\    /   O peration     | Version:  2.1.0
|   \\  /    A nd           | Web:      www.OpenFOAM.o
|    \\/     M anipulation  |
\*---------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      topoSetDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * *

actions
(

    // new To cellSet
    {
        name    air;
        type    cellSet;
        action  new;
        // Cells in cell zone
        source zoneToCell;
        sourceInfo
```

C ▾ タブ幅: 4 ▾ (40行、19列) [挿入]




Fieldへのデータセット
この画面でcellSet名とfieldを指定してデータをセットする。
ファイル:
Dict読込
Dict保存
csv読込...
クリック
追加:
cellSet追加...
field追加...
topoSetEditor起動
gridEditor起動
box追加
cylinder追加
sphere追加
削除:
行(cellSet)削除
列(field)削除
cellのクリア
コピー:
コピー
貼り付け
geometry data
Qw
T
← field名
defaultFieldValues
air
base
cyli1
cyli2
cyli3
cyli4
cyli5
閉じる
cellSet名 →
csvデータ読み込み
setFieldsDict用のcsvファイルを読み込みます。
読み込むcsvファイルを選択してください。
場所:
/home/caeuser/Desktop/coolingMc
参照..
filter:
*.csv
表示
fileを選択
選択してOKをクリック
regionMaterial-org.csv
setFieldsDict-org.csv
Fieldへのデータセット
クリックして、setFieldsDictが出来上がる
ファイル:
Dict読込
Dict保存
csv読込...
csv保存...
追加:
cellSet追加...
field追加...
topoSetEditor起動
gridEditor起動
box追加
cylinder追加
sphere追加
削除:
行(cellSet)削除
列(field)削除
cellのクリア
コピー:
コピー
貼り付け
geometry data
Qw
T
defaultFieldValues 0 300
air 0
base 20 350
cyli1 10
cyli2 2
cyli3 5
cyli4 5

データセットファイル「setFieldsDict-org.csv」の内容

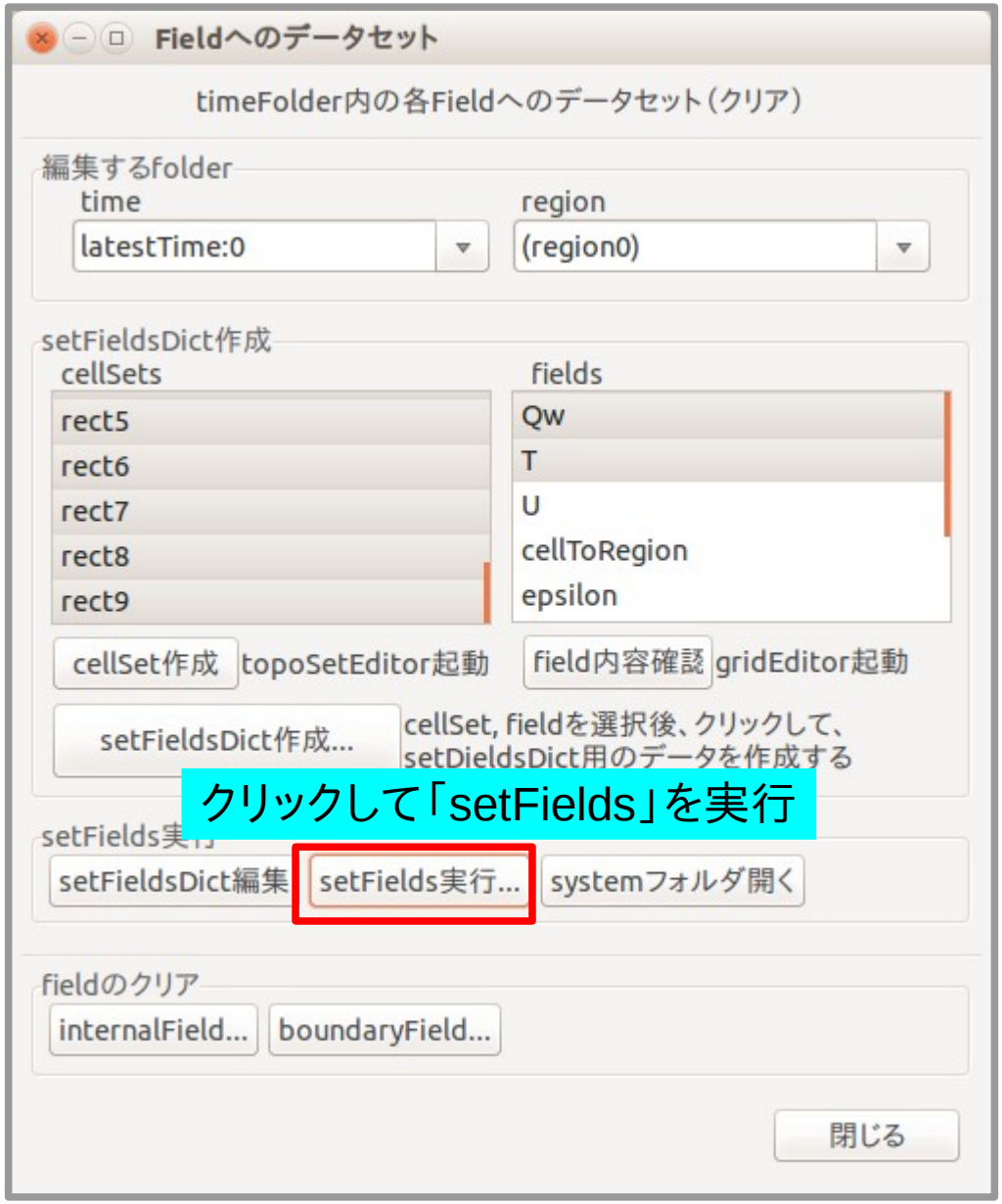


	A	B	C	D
1	<setFieldsDict>			
2	items	data	Qw	T
3	defaultFieldValues		0	300
4	air		0	
5	base		20	350
6	cyli1		10	
7	cyli2		2	
8	cyli3		5	
9	cyli4		5	
10	cyli5		0	
11	cyli6		1	
12	cyli7		2	
13	cyli8		1	
14	cyli9		1	
15	heatBlock		0	
16	rect1		0	
17	rect10		1	
18	rect11		1	
19	rect12		5	
20	rect13		2	
21	rect2		0	
22	rect3		1	
23	rect4		3	
24	rect5		1	
25	rect6		1	
26	rect7		1	

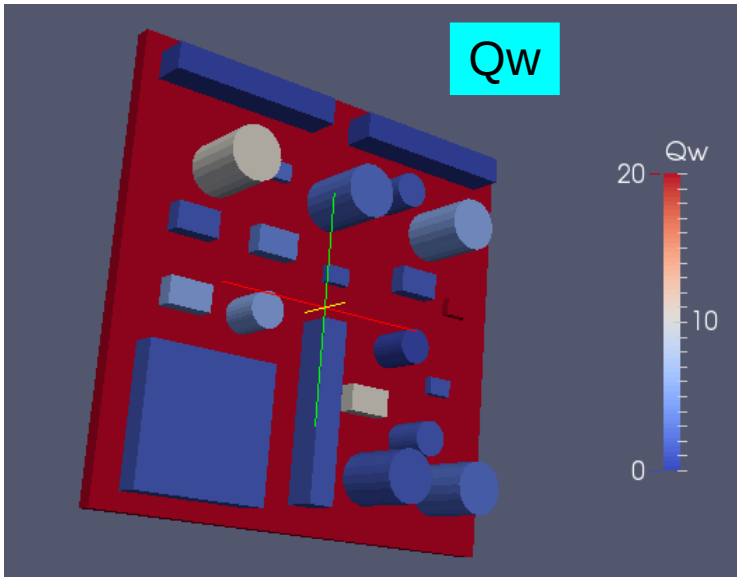
表のイメージそのままが保存されている

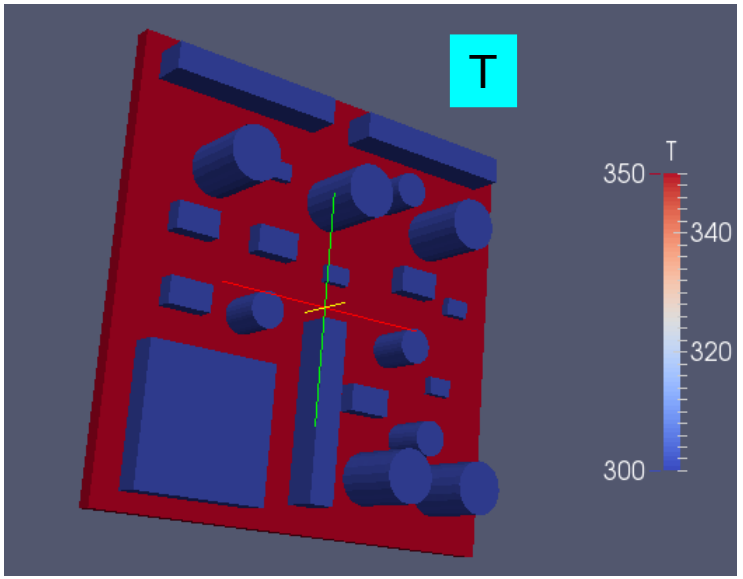
cellSet名：stlファイル名と同じ

setFieldsを実行してデータをセットする



データセット結果

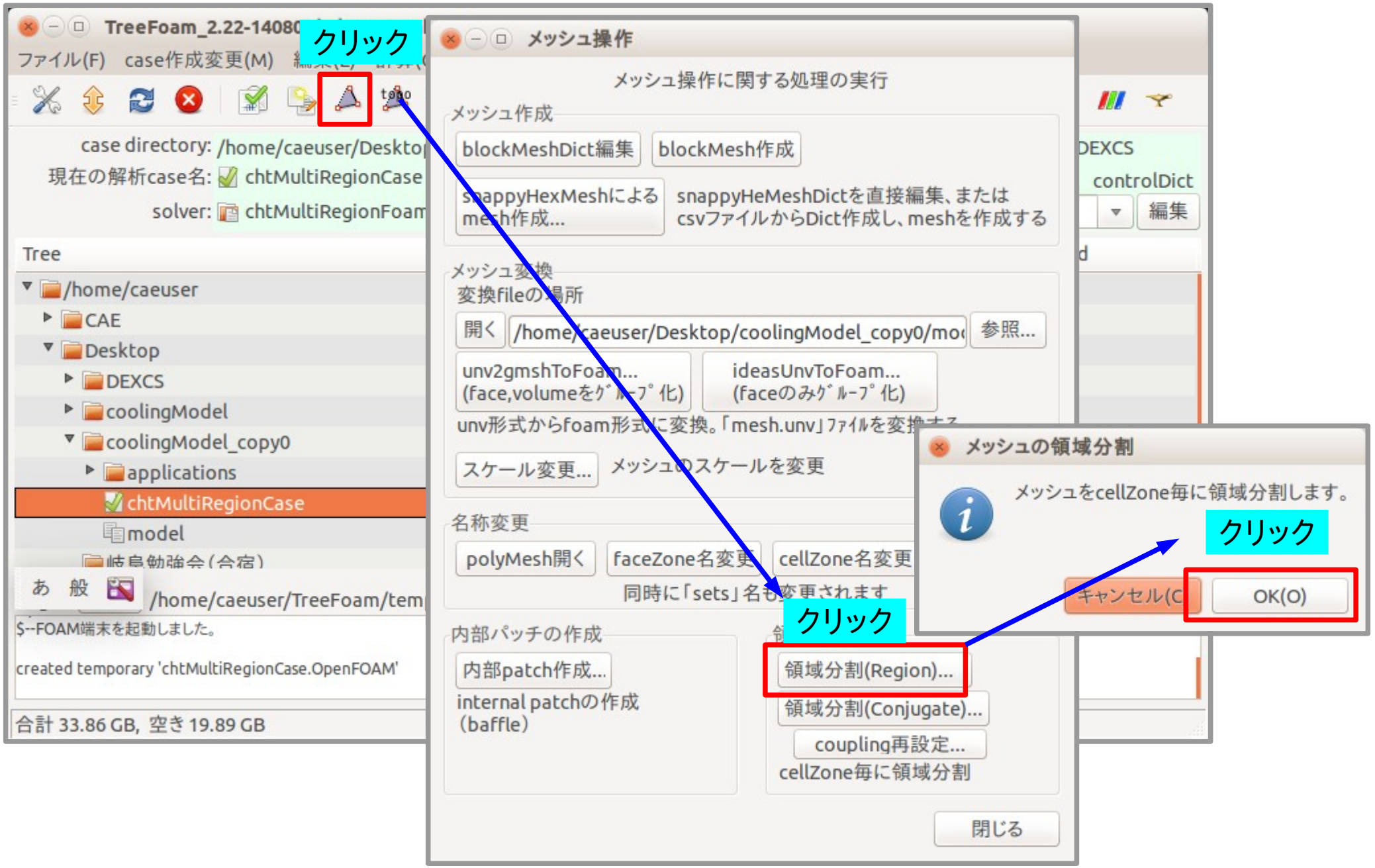




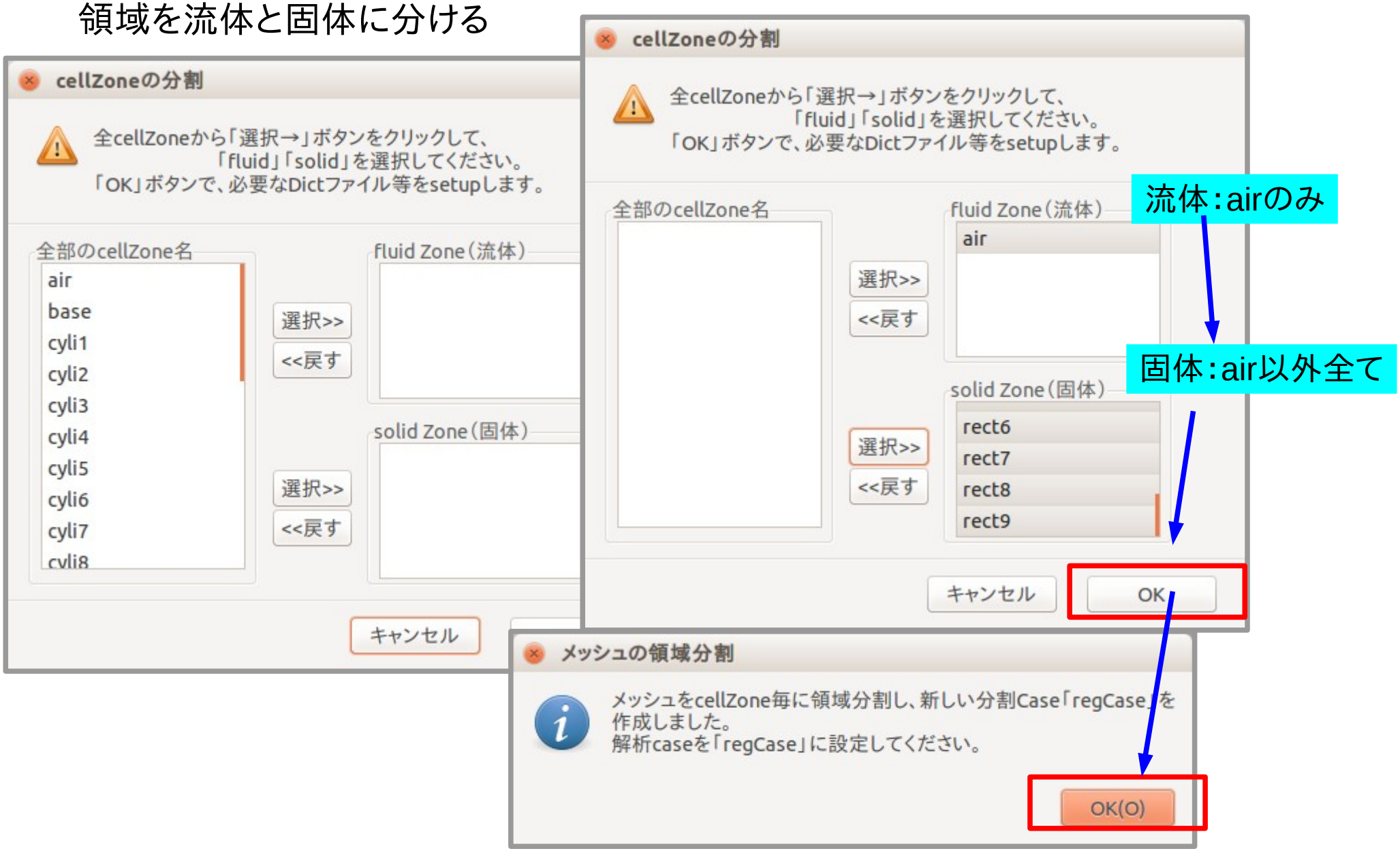
うまくデータがFieldにセットできている。

4） 領域分割

cellZone（部品）毎に領域分割する。

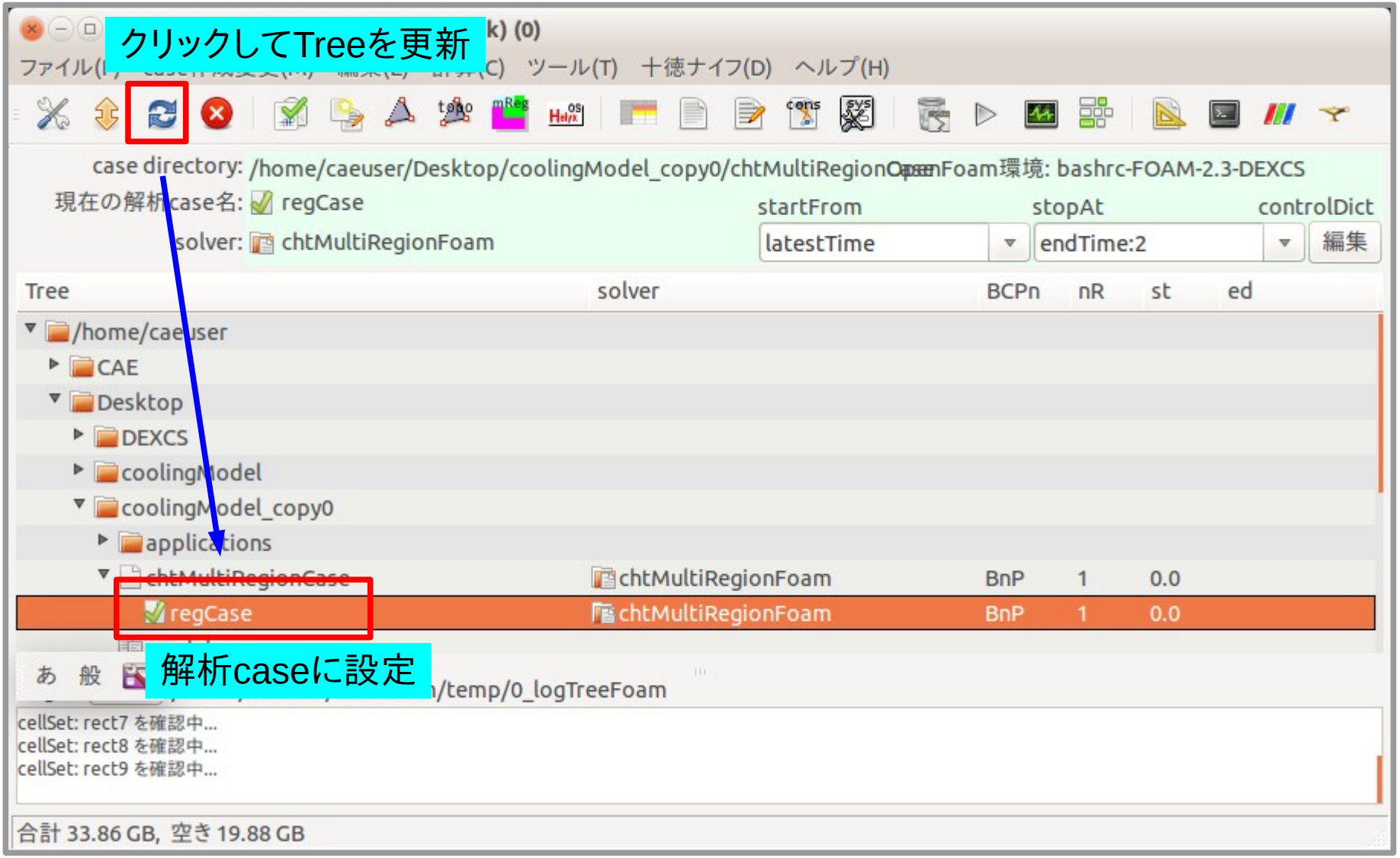


領域を流体と固体に分ける



領域分割できたことになる。
領域分割されたcaseは、「chtMultiRegionCase_copy0/**regCase**」になる。

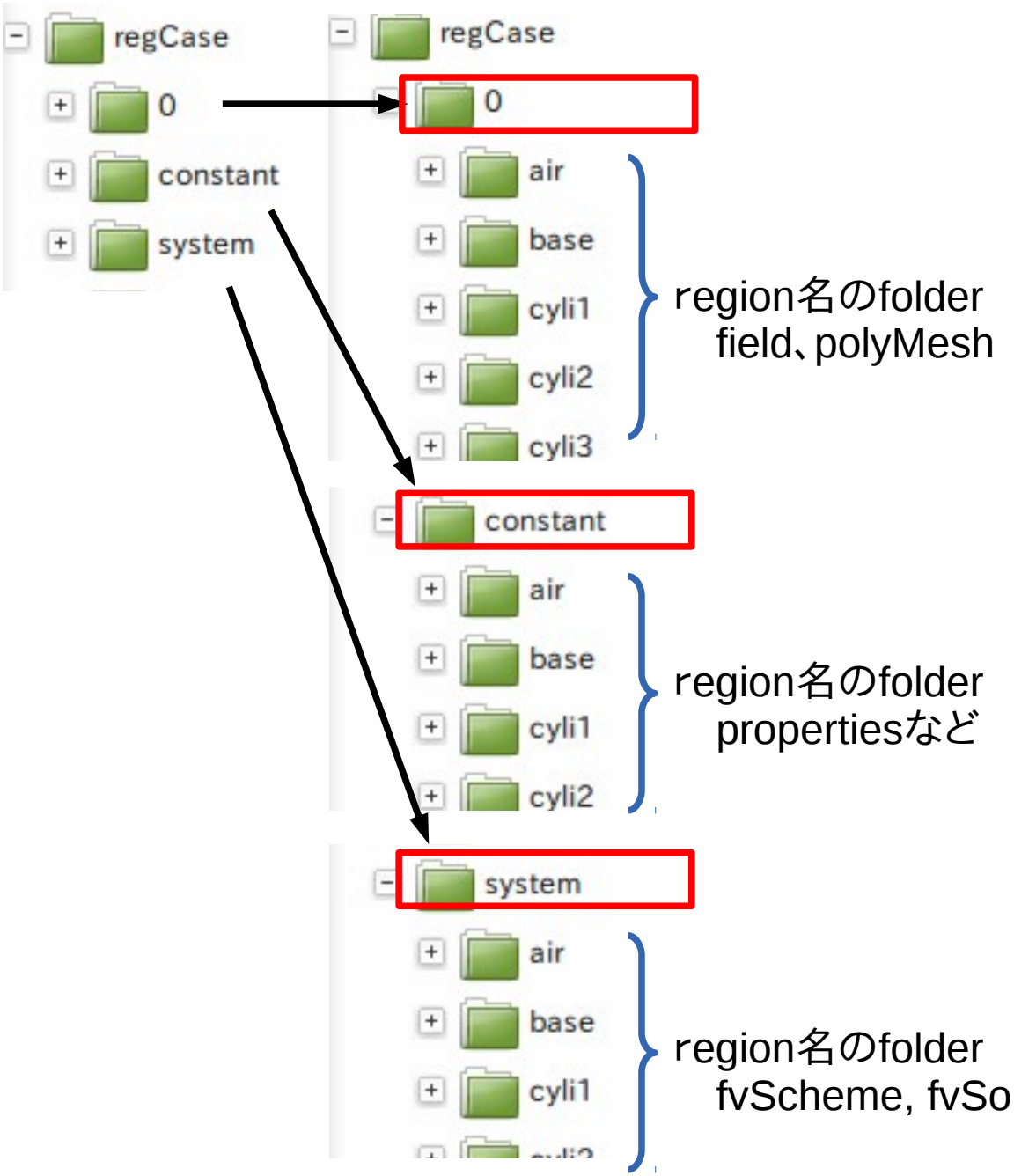
領域分割case「regCase」を解析caseに設定（レ点マークをつける）



以後、「regCase」で解析を進めていく。

5） 各regionの設定

regCase（multiRegion）内のfolder構成

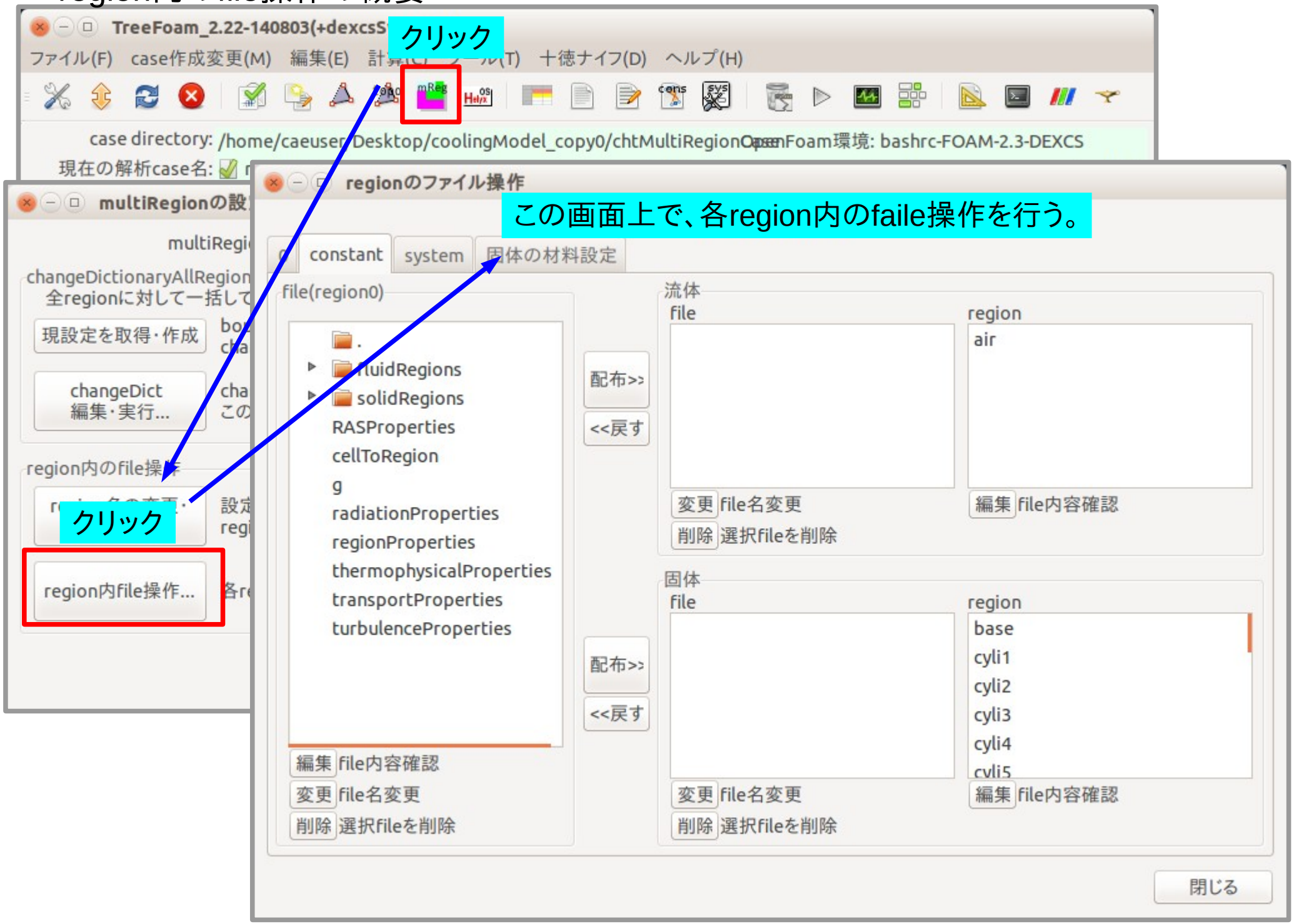


・各folder内にregion名folder存在する
・Region名folder内に必要なファイルを配置する必要がある

region名が多くあると、file操作が煩雑（fileコピー、編集が煩雑）

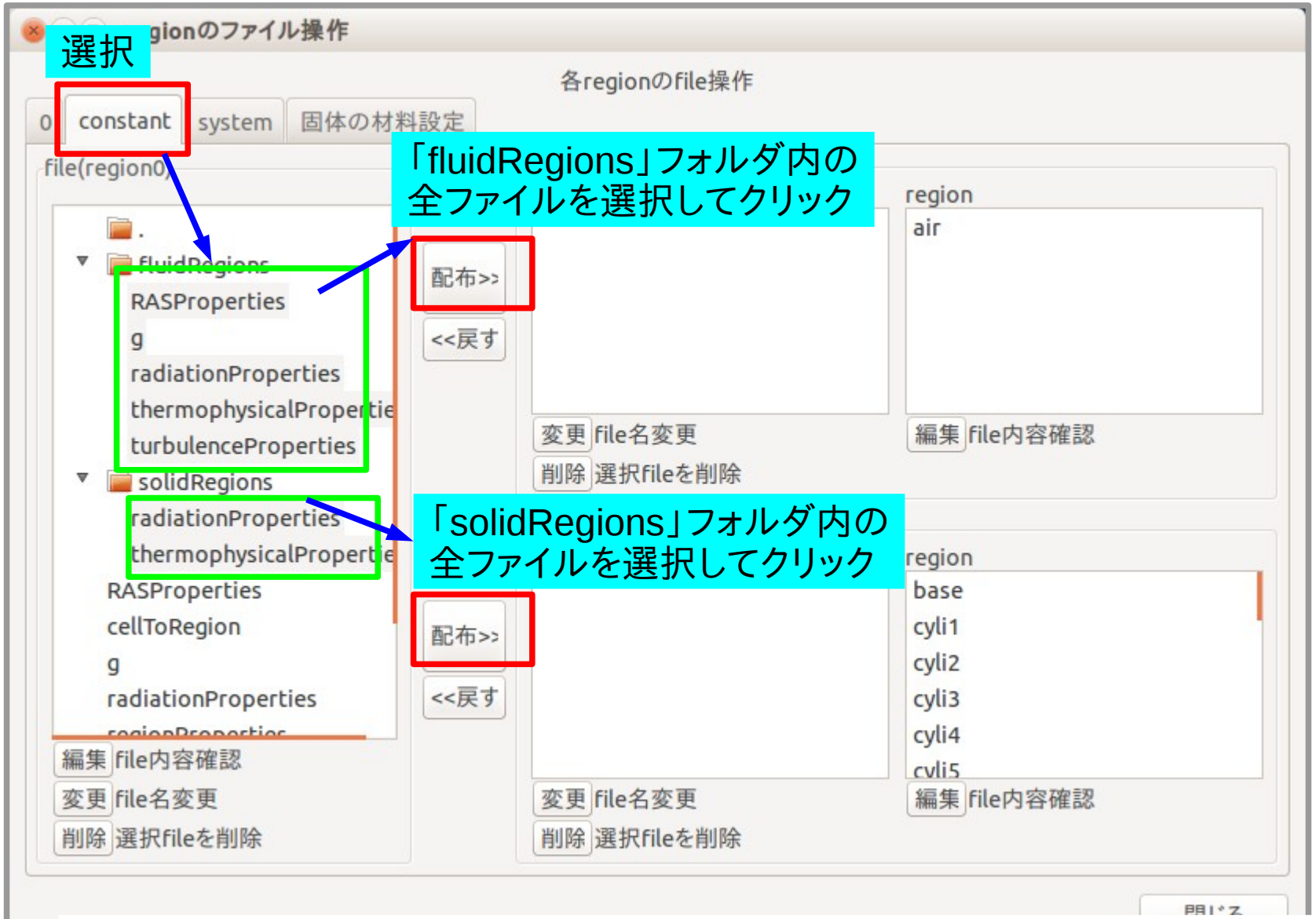
各regin内のfile操作設定が楽にできるように、TreeFoamに組み込み

region内のfile操作の概要



constant内のfile操作

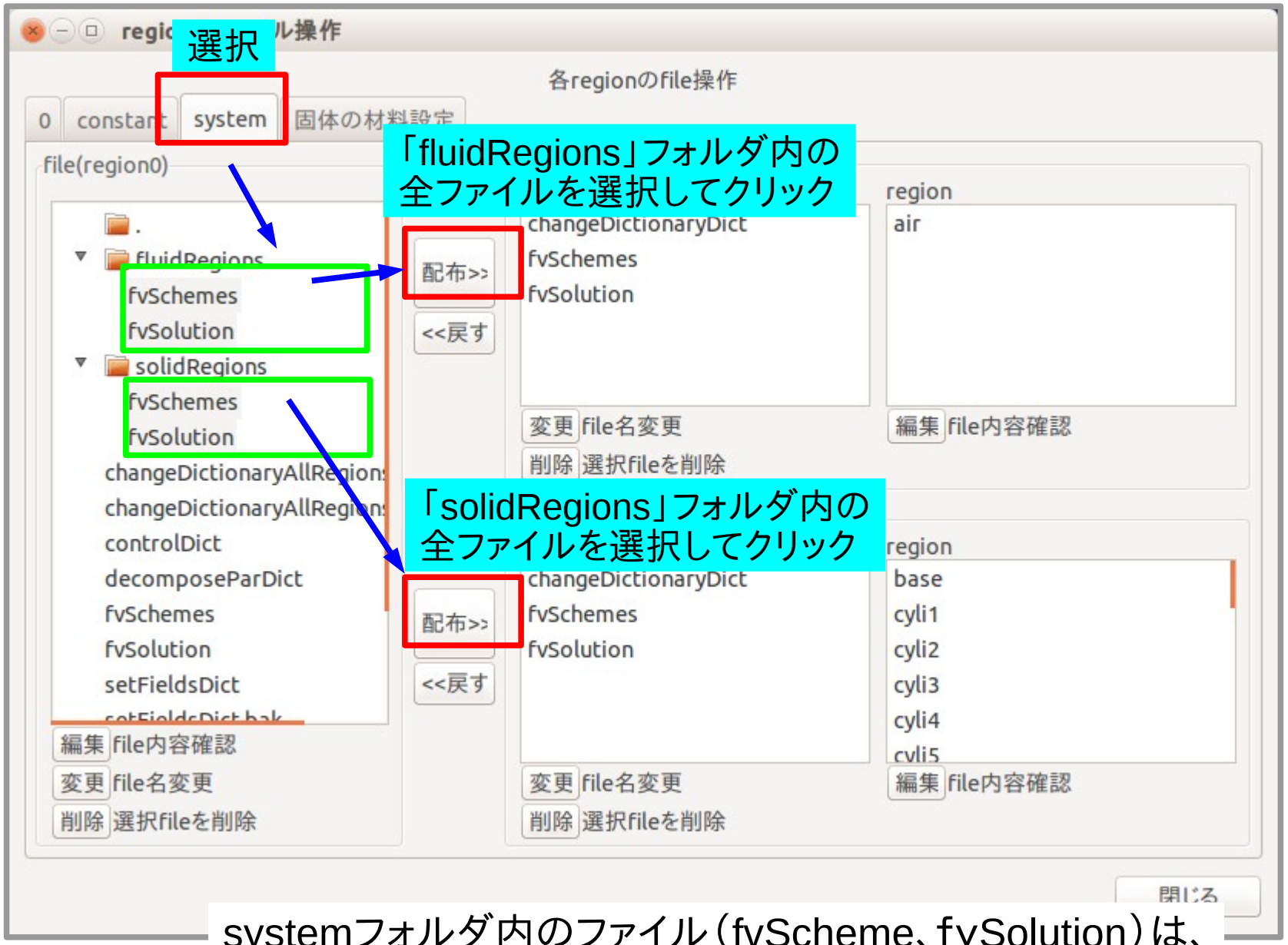
各regionが必要としているファイルは、「fluidRegions」「solidRegions」フォルダ内に準備されているので、これをコピーする。
（timeFolder「0」の内容は、領域分割時に既にコピーされているので、file操作は不要。）



同じファイルが各regionにコピーされたが、
region固有のデータ（thermoPhysicalProperties内のデータ）は、後で設定する。

system内のfile操作

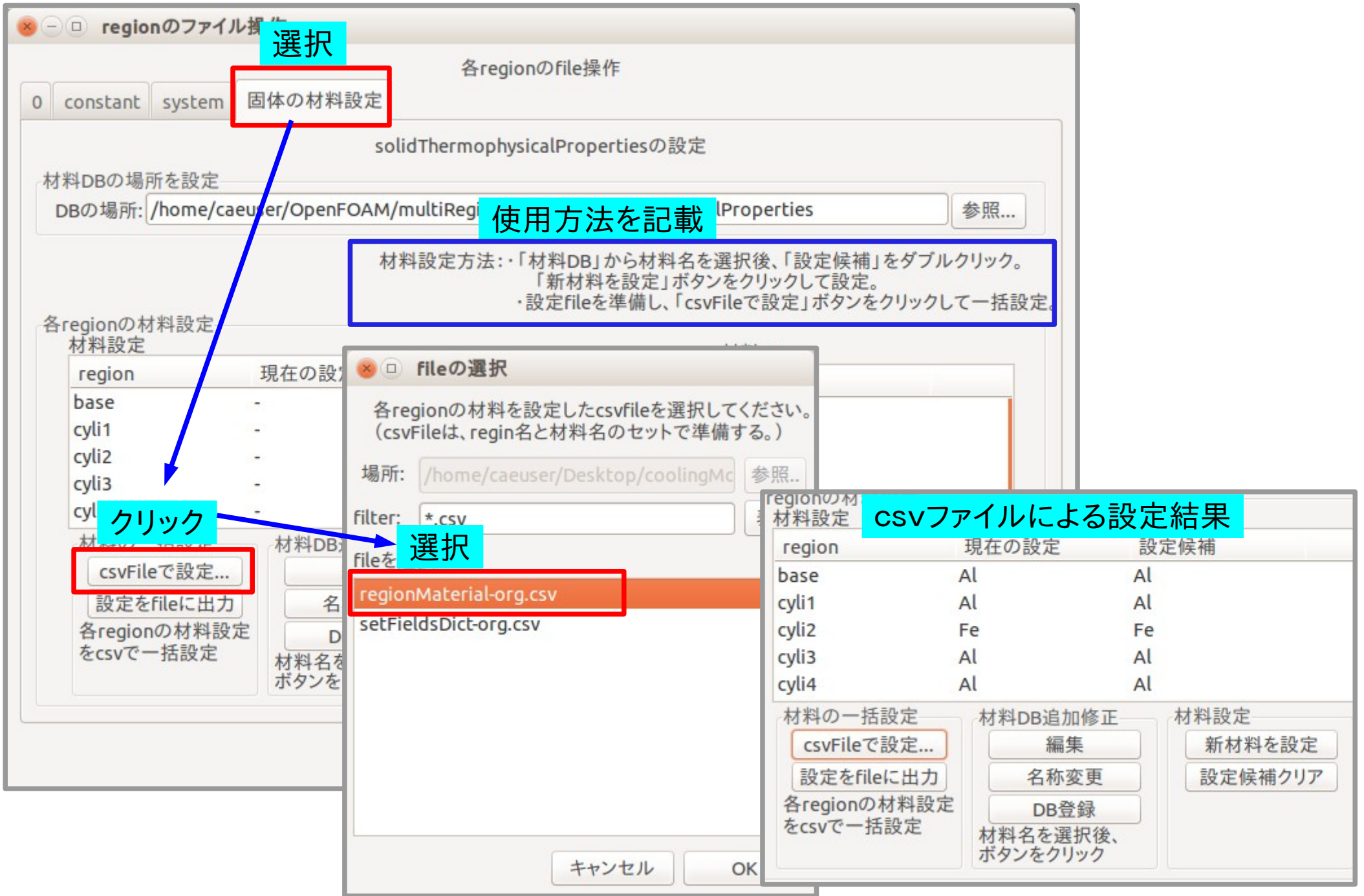
各regionが必要としているファイルは、「fluidRegions」「solidRegions」フォルダ内に準備されているので、これをコピーする。



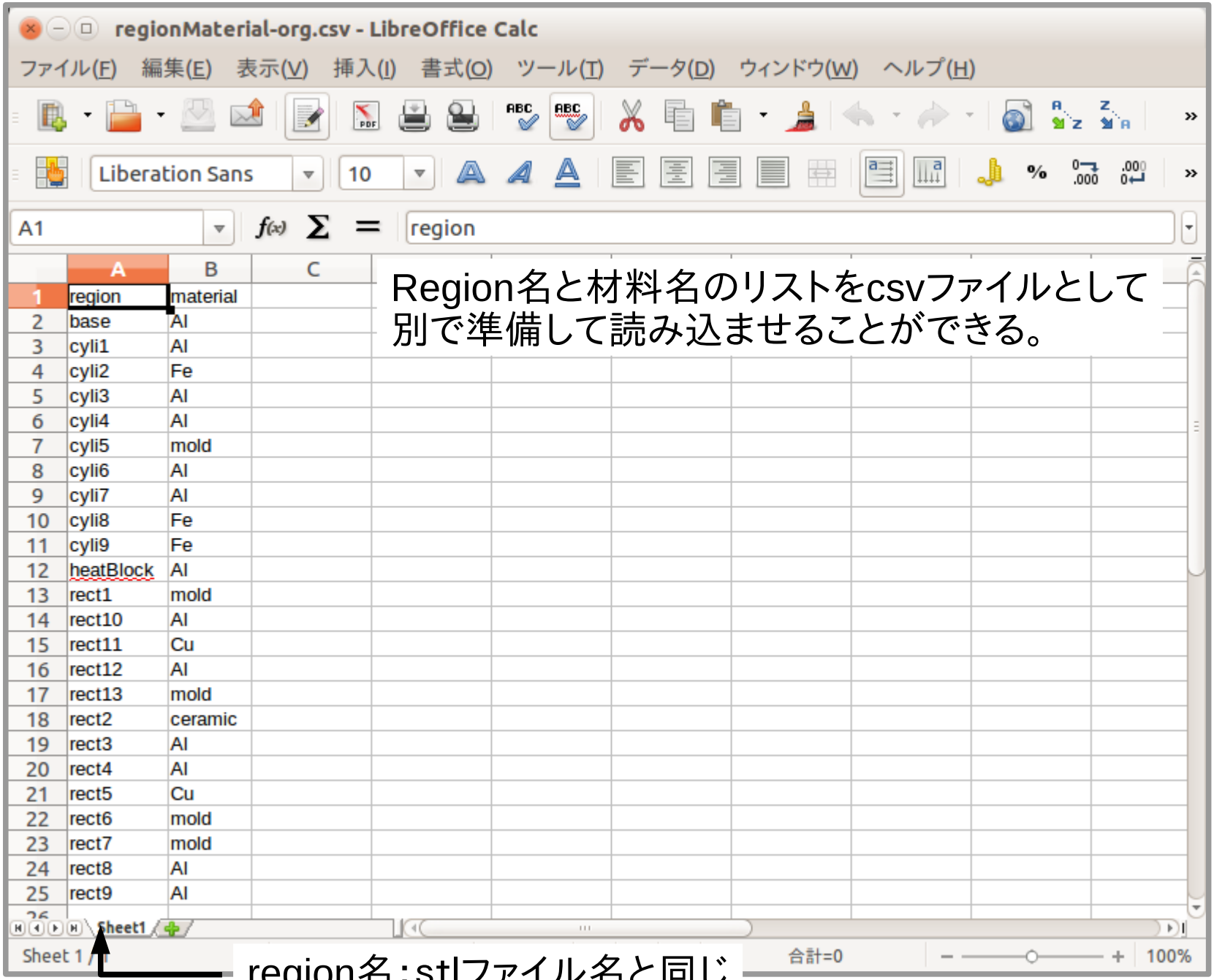
systemフォルダ内のファイル（fvScheme、fvSolution）は、特にファイル内容の編集は不要。

固体の材料設定（thermoPhysicalProperties）

DBから材料名を選びダブルクリックする事で材料を設定できる
今回は、材料設定用のcsvファイルがあるので、これを使って設定する。



材料設定ファイル「regionMaterial-org.csv」の内容



	A	B
1	region	material
2	base	Al
3	cyli1	Al
4	cyli2	Fe
5	cyli3	Al
6	cyli4	Al
7	cyli5	mold
8	cyli6	Al
9	cyli7	Al
10	cyli8	Fe
11	cyli9	Fe
12	heatBlock	Al
13	rect1	mold
14	rect10	Al
15	rect11	Cu
16	rect12	Al
17	rect13	mold
18	rect2	ceramic
19	rect3	Al
20	rect4	Al
21	rect5	Cu
22	rect6	mold
23	rect7	mold
24	rect8	Al
25	rect9	Al

Region名と材料名のリストをcsvファイルとして別で準備して読み込ませることができる。

region名：stlファイル名と同じ

6） 境界条件の設定

流体・固体の境界面のTとU fieldについては、領域分割時に設定されている。
「全fieldの境界面以外のpatch」と「T, U field以外のpatch」を設定する。
今回の場合、patchを流体「air」のみに設定した為、「air」のみの設定で済む。

gridEditor: regCase/0/air (0:1)

ファイル(F) 編集(E) 表示(V)

T, U field ：境界面以外のpatchを設定する

	define patch at constant/air (boundary)	T	U
field type dimensions		volScalarField; [0 0 0 1 0 0 0];	volVectorField; [0 1 -1 0 0 0 0];
internal Field			uniform (0 0 0);
caseWall	type wall; inGroups 1(wall);	type zeroGradient;	type fixedValue; value uniform (0 0 0);
inW	type wall; inGroups 1(wall);	type fixedValue; value uniform 300;	type zeroGradient;
outW	type wall; inGroups 1(wall);	type zeroGradient;	type fixedValue; value uniform (0 0 0.1);
air_to_cyli9	type mappedWall; inGroups 1(wall); sampleMode nearestPatchFace; sampleRegion cyli9; samplePatch cyli9_to_air;	type compressible::turbulentTemperatureCoupledBaffleMixed; value uniform 300; Tnbr T; kappa fluidThermo; kappaName none;	type fixedValue; value uniform (0 0 0);
air_to_rect9	type mappedWall; inGroups 1(wall); sampleMode nearestPatchFace; sampleRegion rect9; samplePatch rect9_to_air;	type compressible::turbulentTemperatureCoupledBaffleMixed; value uniform 300; Tnbr T; kappa fluidThermo; kappaName none;	type fixedValue; value uniform (0 0 0);
	type mappedWall;	type compressible::turbulentTemperatureCoupledBaffleMixed;	type fixedValue;

gridEditor: regCase/0/air (0:3)

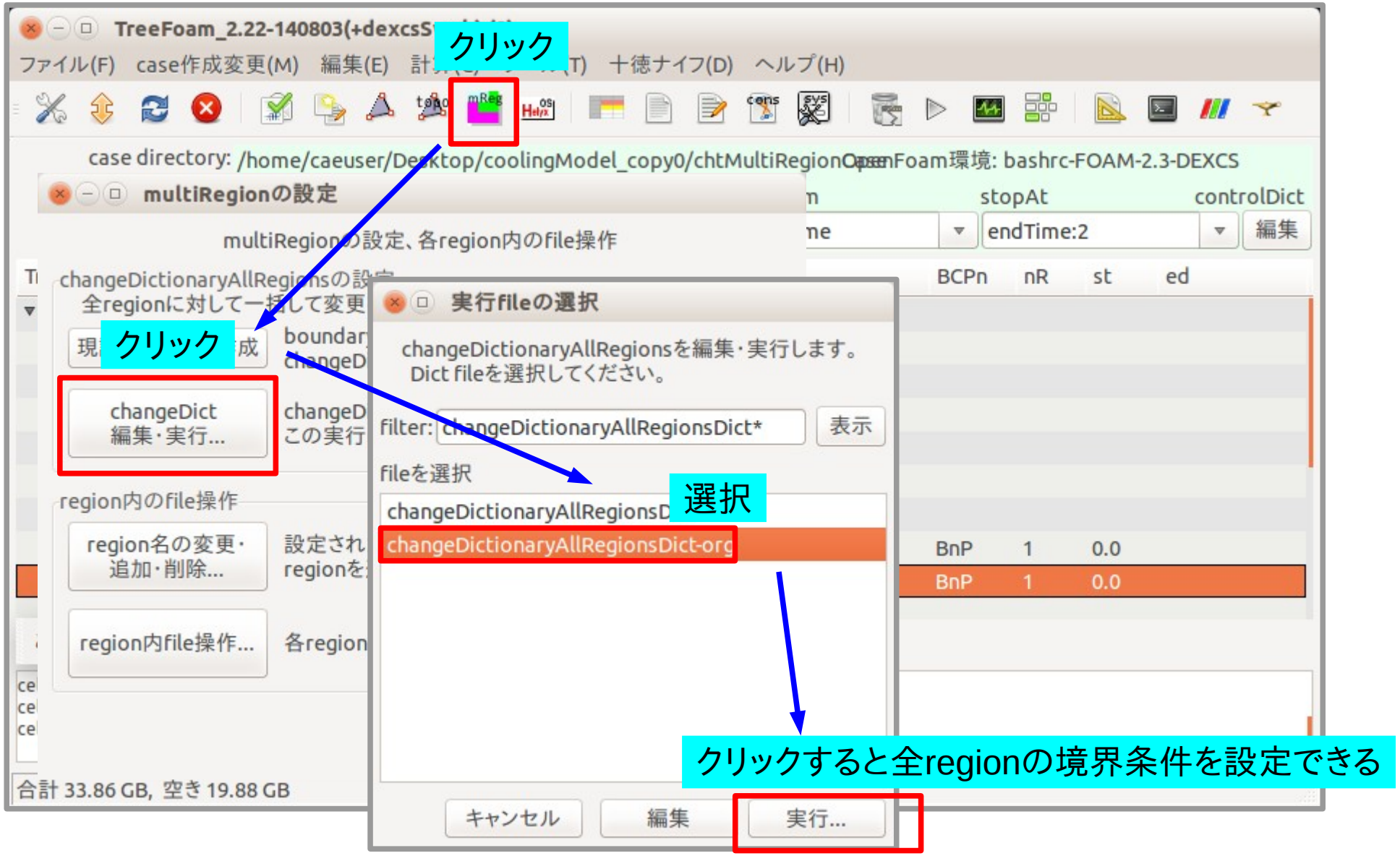
ファイル(F) 編集(E) 表示(V)

epsilon, k, p, p_rgh field ：全てのpatchを設定する

	epsilon	k	p	p_rgh
field type dimensions	volScalarField; [0 2 -3 0 0 0 0];	volScalarField; [0 2 -2 0 0 0 0];	volScalarField; [1 -1 -2 0 0 0 0];	volScalarField; [1 -1 -2 0 0 0 0];
internal Field	uniform	100000;		uniform 100000;
caseWall	type compressible::epsilonWallFunction; value uniform 0.01; Cmu 0.09; kappa 0.41; E 9.8;	type compressible::kqRWallFunction; value uniform 0.1;	type calculated; value uniform 100000;	type fixedFluxPressure; gradient uniform 0; value uniform 100000;
inW	type fixedValue; value uniform 0.01;	type inletOutlet; inletValue uniform 0.1; value uniform 0.1;	type calculated; value uniform 100000;	type fixedValue; value uniform 100000;
outW	type zeroGradient;	type zeroGradient;	type calculated; value uniform 100000;	type zeroGradient;
air_to_cyl i9	type compressible::epsilonWallFunction; value uniform 0.01;	type compressible::kqRWallFunction; value uniform 0.1;	type calculated; value uniform 100000;	type fixedFluxPressure; value uniform 100000; gradient uniform 0;
air_to_rec t9	type compressible::epsilonWallFunction; value uniform 0.01;	type compressible::kqRWallFunction; value uniform 0.1;	type calculated; value uniform 100000;	type fixedFluxPressure; value uniform 100000; gradient uniform 0;

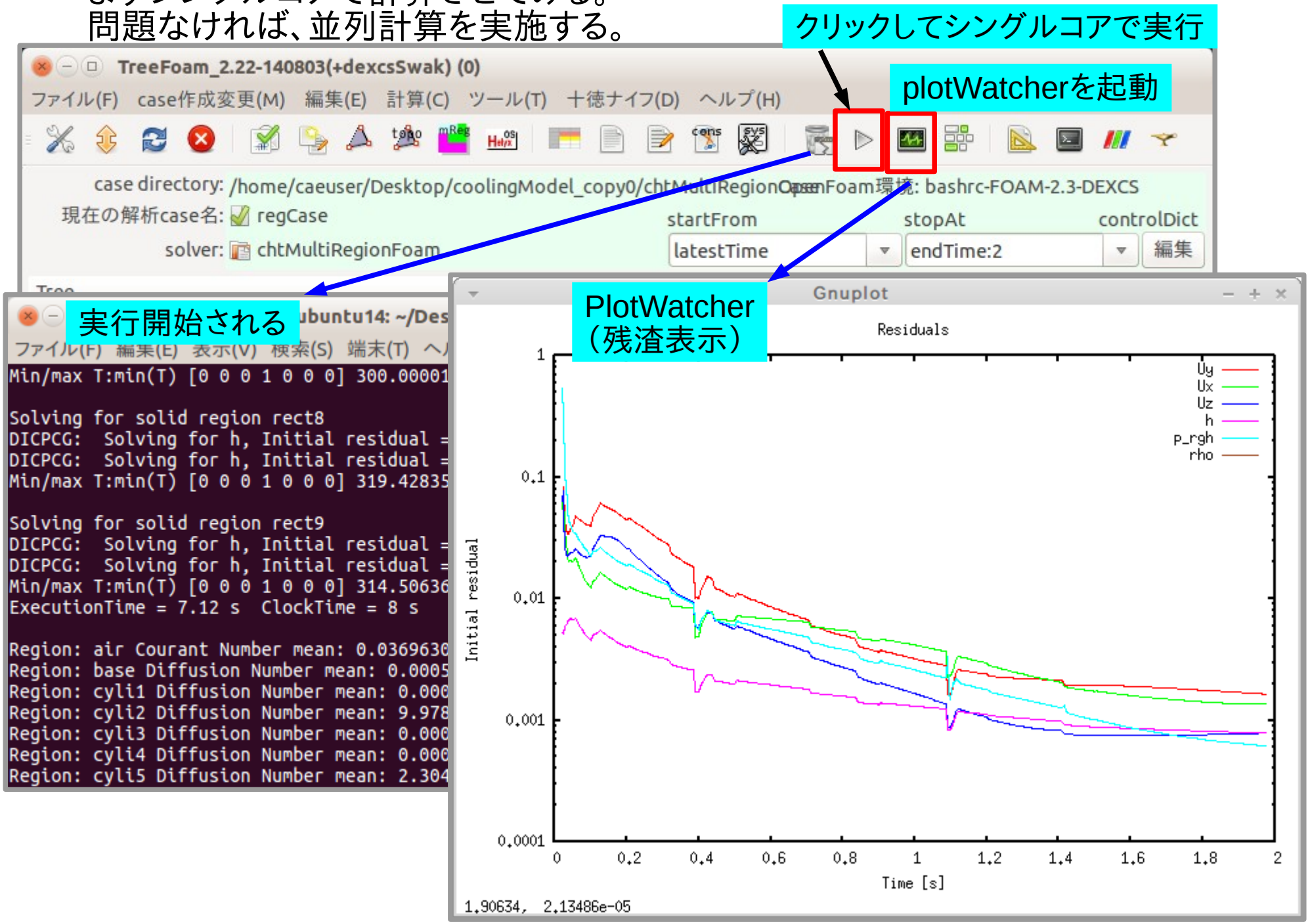
_A 般

zeroGradient以外の境界条件は、その設定が保存できる。
今回の場合、その保存fileがあるので、読み込んで設定できる。

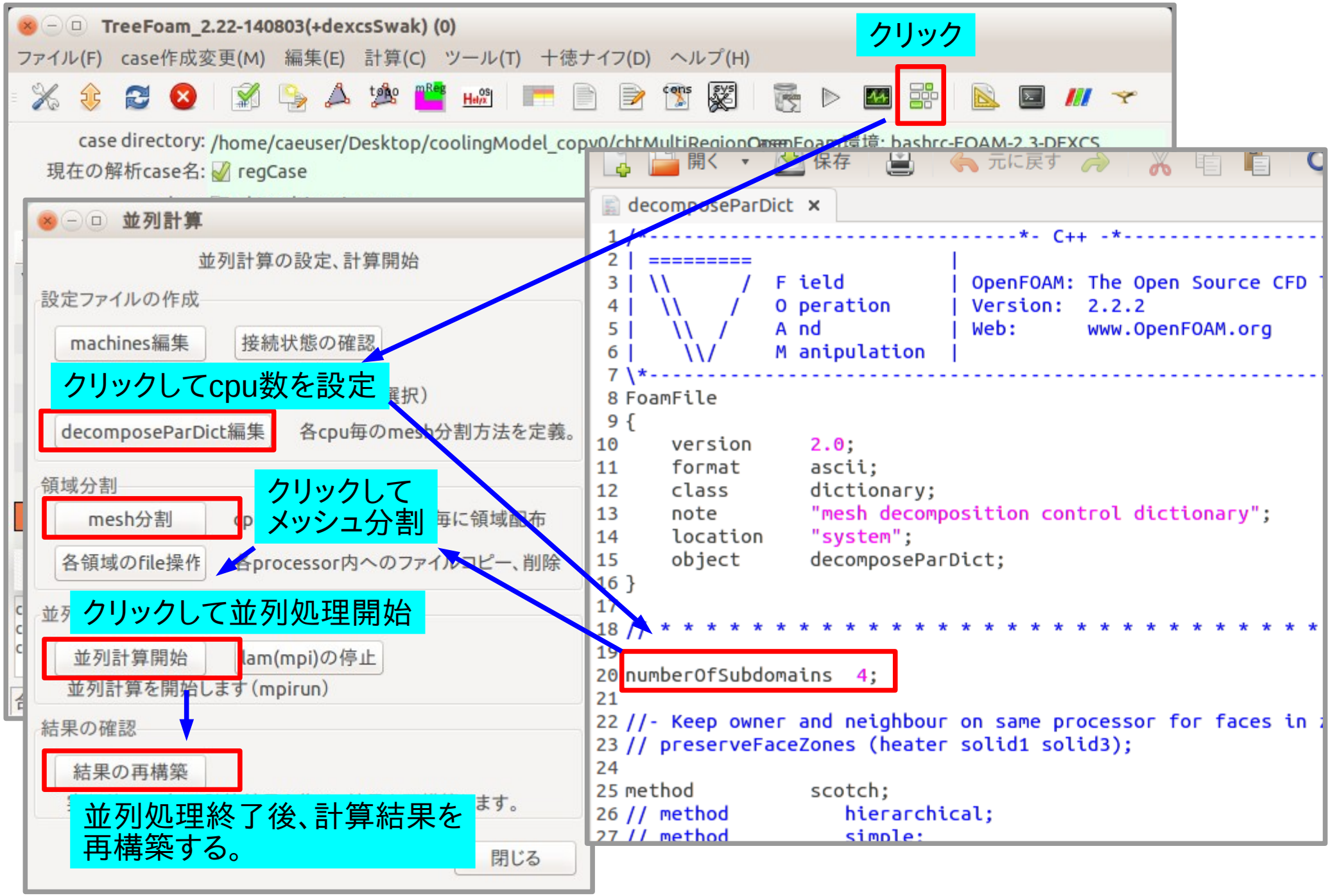


7） 計算開始、結果の確認

まずシングルコアで計算させてみる。
問題なければ、並列計算を実施する。



並列処理で実行する。


TreeFoam_2.22-140803(+dexcsSwak) (0)
ファイル(F) case作成変更(M) 編集(E) 計算(C) ツール(T) 十徳ナイフ(D) ヘルプ(H)
クリック
case directory: /home/caeuser/Desktop/coolingModel_copy0/chtMultiRegionOpenFoam
現在の解析case名: regCase
並列計算
並列計算の設定、計算開始
設定ファイルの作成
machines編集
接続状態の確認
クリックしてcpu数を設定
decomposeParDict編集
各cpu毎のmesh分割方法を定義。
領域分割
mesh分割
クリックして
メッシュ分割
各領域のfile操作
各processor内へのファイルコピー、削除
クリックして並列処理開始
並列計算開始
lam(mpi)の停止
並列計算を開始します(mpirun)
結果の確認
結果の再構築
並列処理終了後、計算結果を
再構築する。
閉じる
開く
保存
元に戻す
decomposeParDict
/*------------------------------------*- C++ -*----------
| ========= | |
| \\ / F ield | OpenFOAM: The Open Source CFD
| \\ / O peration | Version: 2.2.2
| \\ / A nd | Web: www.OpenFOAM.org
| \\/ M anipulation | |
*---
FoamFile
{
version 2.0;
format ascii;
class dictionary;
note "mesh decomposition control dictionary";
location "system";
object decomposeParDict;
}
// *
numberOfSubdomains 4;
//- Keep owner and neighbour on same processor for faces in
// preserveFaceZones (heater solid1 solid3);
method scotch;
// method hierarchical;
// method simple;

paraFoamによる計算結果の確認

multiRegionの場合、paraFoamで全regionのfieldを表示させようとする時、regionが多くあると操作が非常に煩雑。　以下のどちらかで対処する。

→　マクロ（pythonスクリプト）を作成して実行
→　「paraFoam -bultin」オプション付きでparaFoamを起動

ParaFoamのoption内容

```
$ paraFoam -help

Usage: paraFoam [OPTION] [PARAVIEW_OPTION]
options:
  -block          use blockMesh reader  (uses .blockMesh extension)
  -builtin        use VTK builtin OpenFOAM reader  (uses .foam extension)
  -case <dir>       specify alternative case directory, default is the cwd
  -region <name>    specify alternative mesh region
  -touch          only create the file  (eg, .blockMesh, .OpenFOAM, etc)
  -touchAll       create .blockMesh, .OpenFOAM files (and for all regions)
  -help           print the usage
```

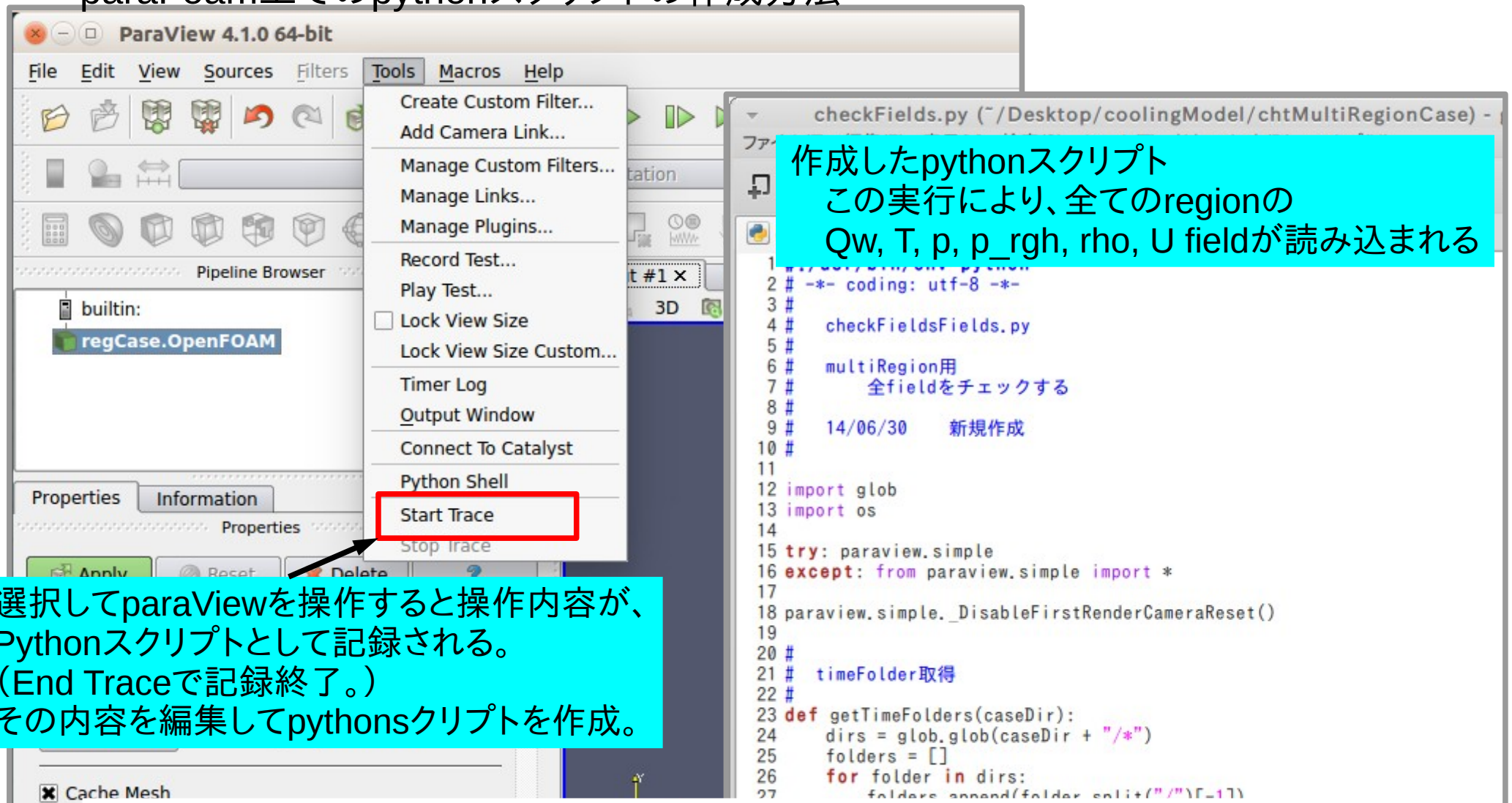
端末を起動して確認すると良くわかる。

```
$ paraFoam -touchAll
$ paraFoam -region air
```

```
$ paraFoam -builtin
```

マクロ（pythonスクリプト）を作成する場合

paraFoam上でのpythonスクリプトの作成方法



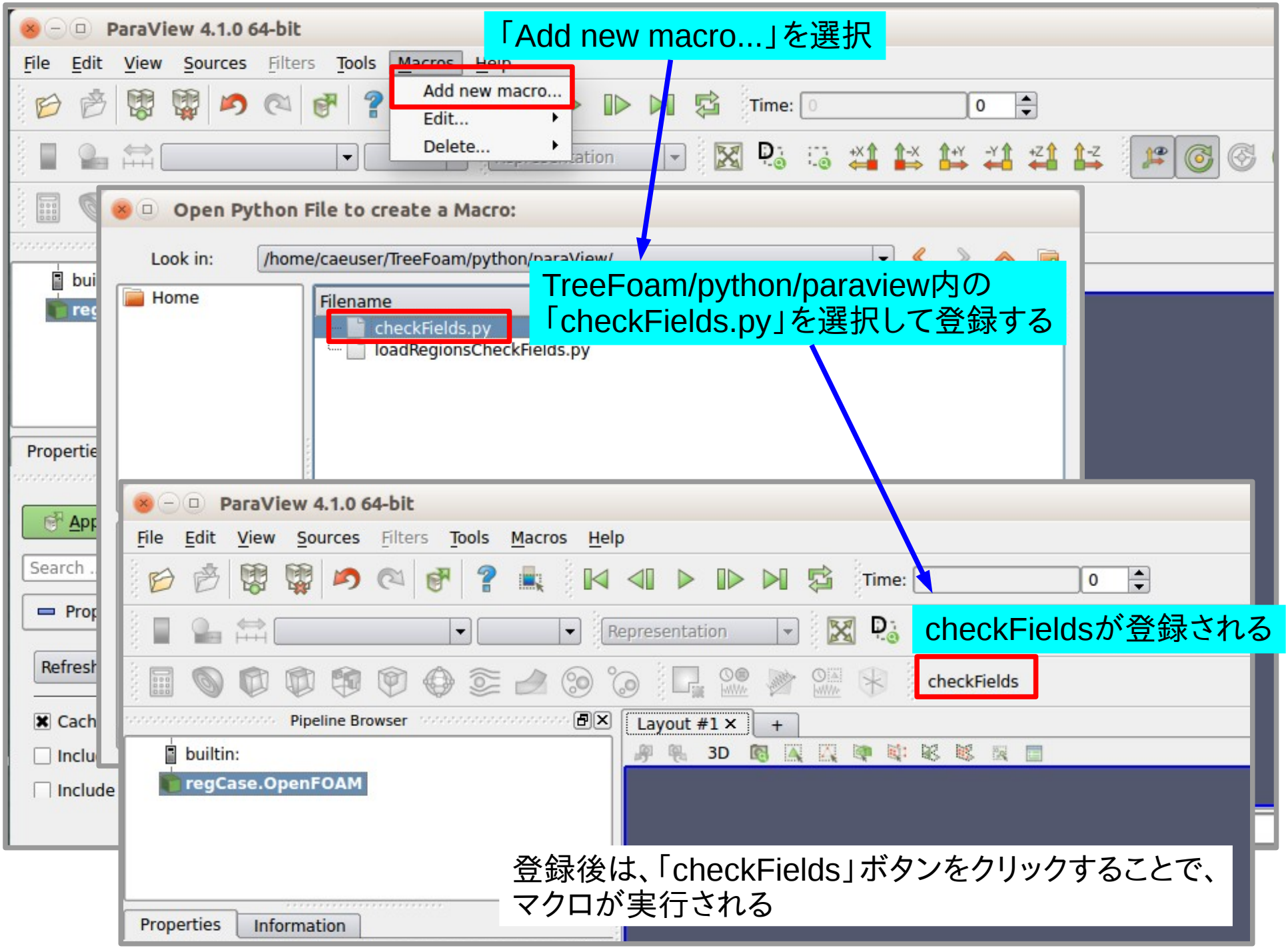
作成したpythonスクリプト
この実行により、全てのregionの
Qw, T, p, p_rgh, rho, U fieldが読み込まれる

選択してparaViewを操作すると操作内容が、
Pythonスクリプトとして記録される。
（End Traceで記録終了。）
その内容を編集してpythonsクリプトを作成。

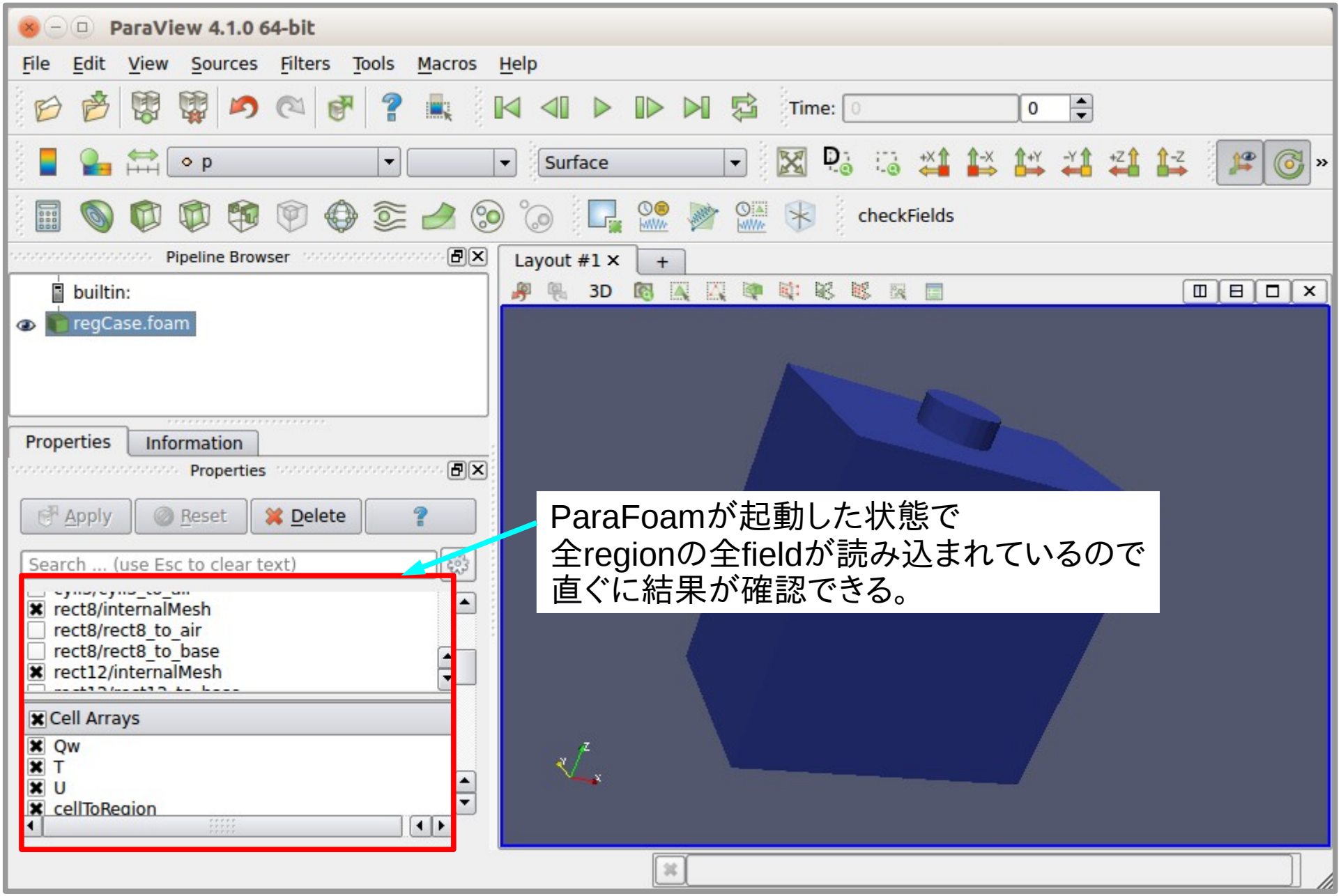
以下のpythonスクリプトを作成しているので、マクロに登録しておくと操作が楽。

- checkFields.py
  表示されているregionの全fileldにチェックを付ける
- loadRegionsCheckFields.py
  全regionを読み混んだ後、全fieldにチェックを付ける

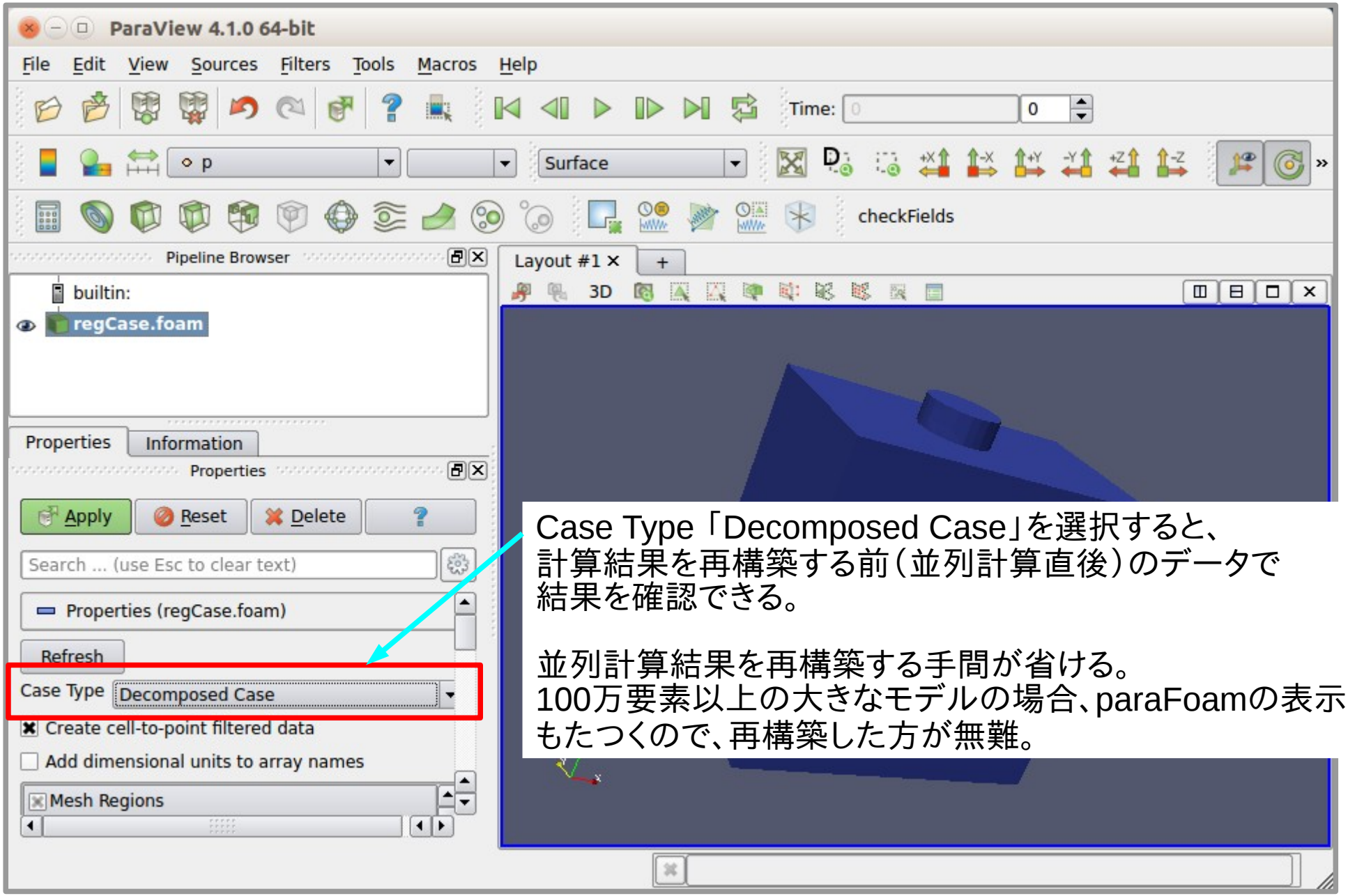
paraFoamのマクロ登録



「paraFoam -builtin」で起動した状態



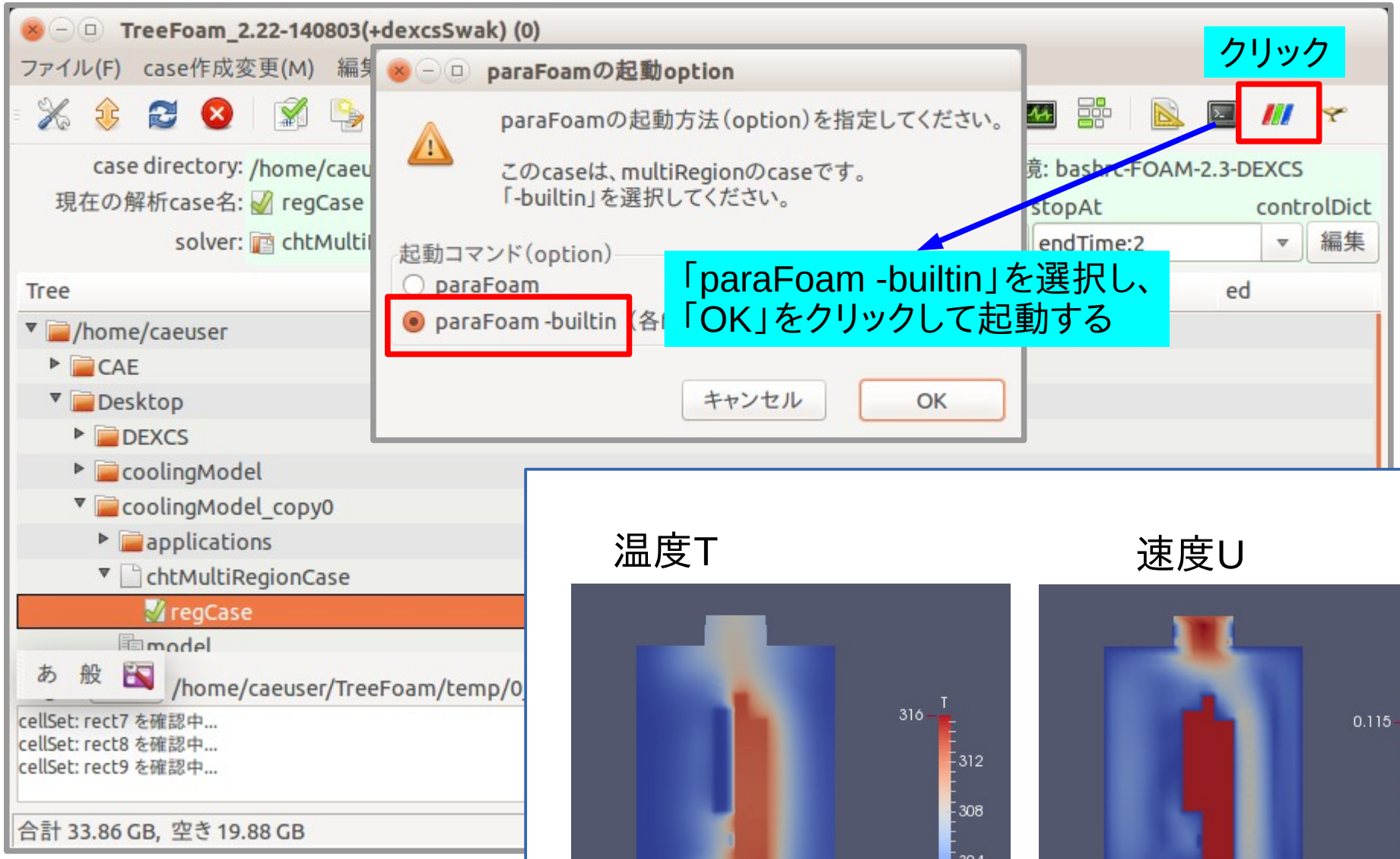
「paraFoam -builtin」で起動した状態



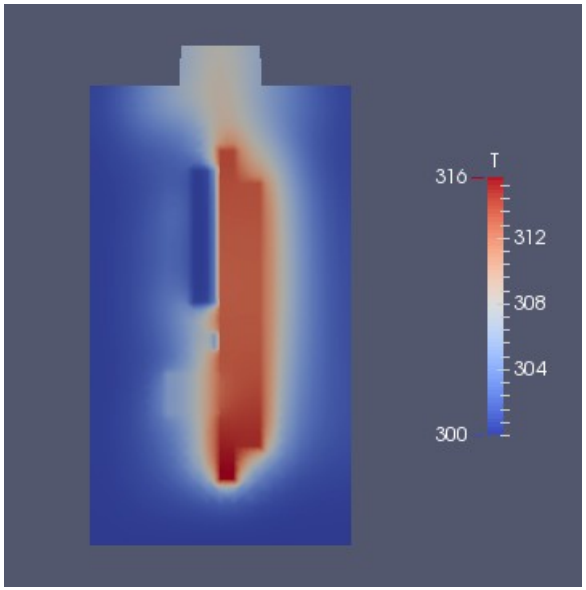
Case Type「Decomposed Case」を選択すると、
計算結果を再構築する前（並列計算直後）のデータで
結果を確認できる。

並列計算結果を再構築する手間が省ける。
100万要素以上の大きなモデルの場合、paraFoamの表示が
もたつくので、再構築した方が無難。

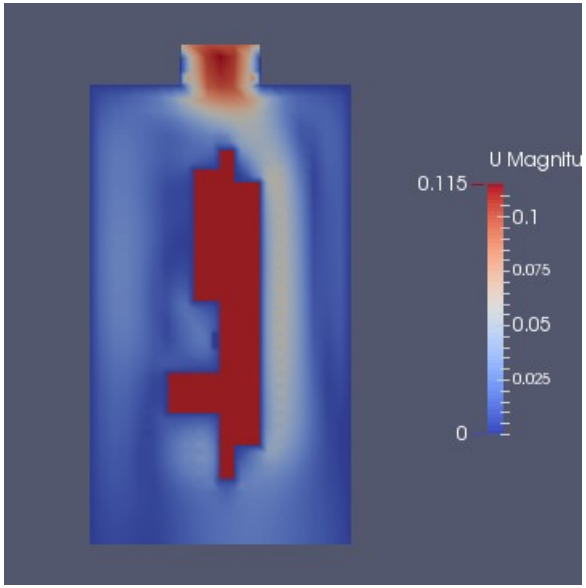
計算結果の確認方法



温度T

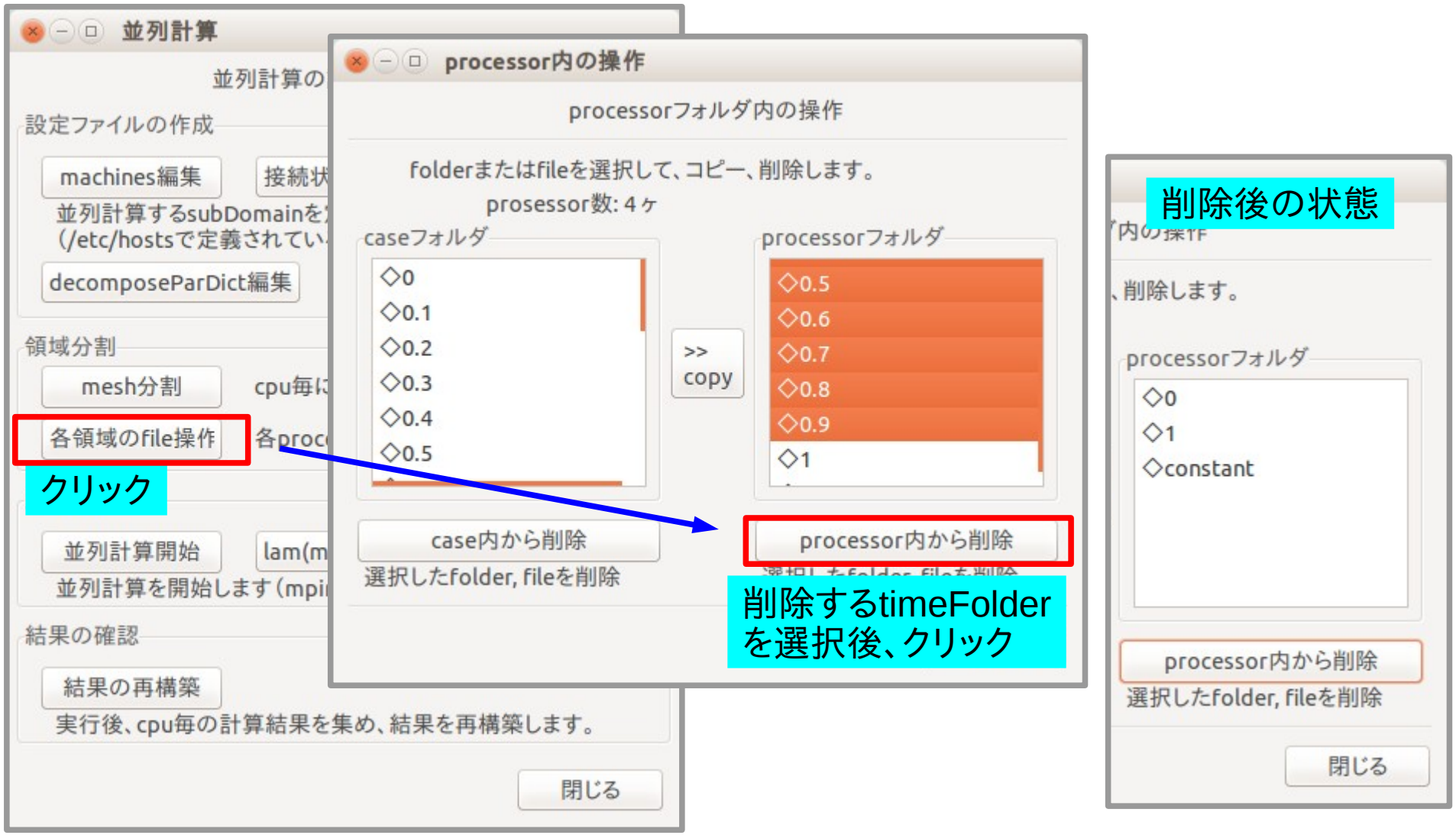


速度U



各processor内のfile操作

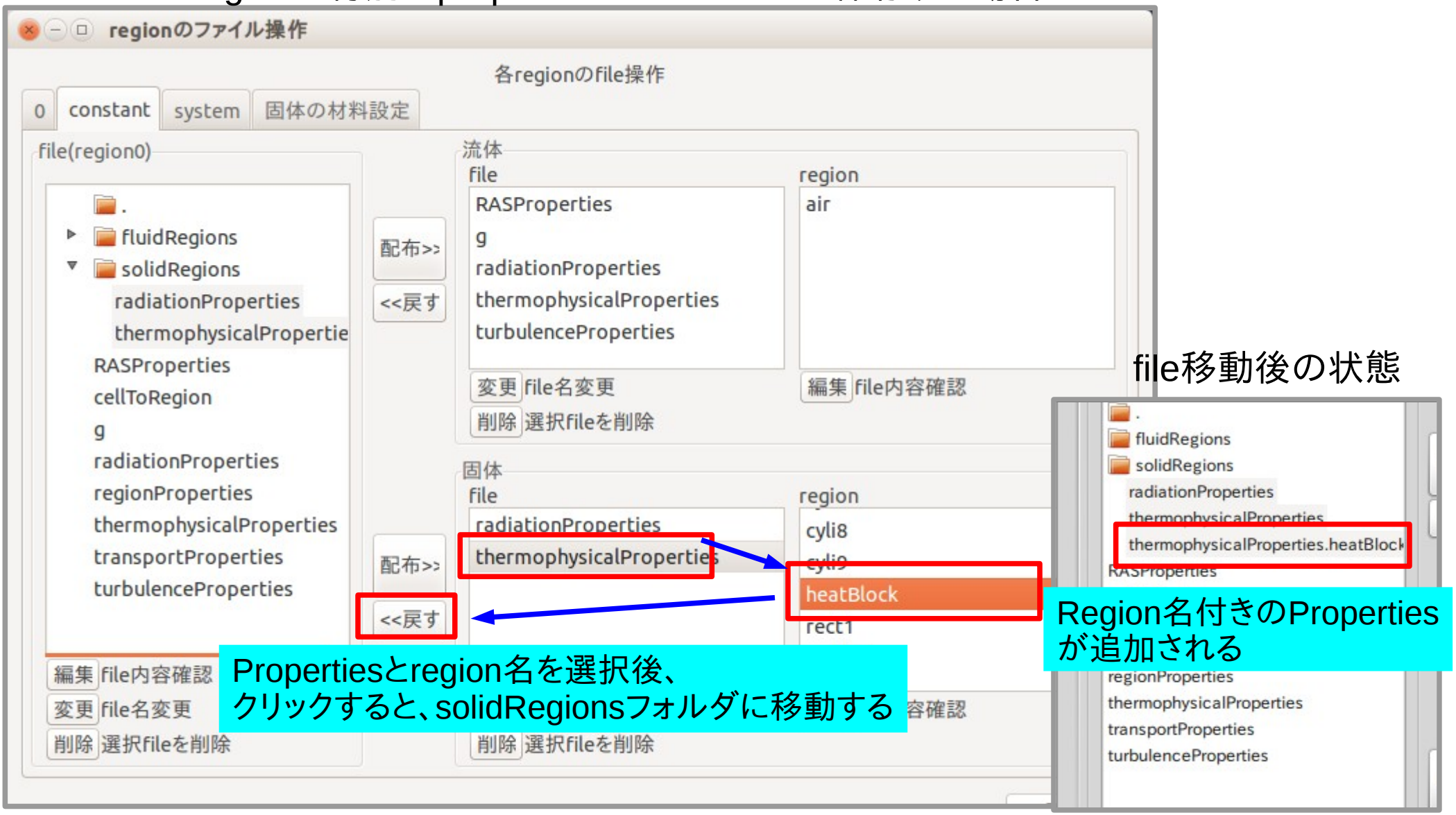
並列数が増えた場合、
processor内の一部のtimeFolderを削除する時、操作が煩雑になる。
　計算結果を再構築した後は、基本的にprocessor内のtimeFolderは、
　最初と最後のみ残しておけば、問題ない。
以下の操作で、容易に削除できる。



8） 見本となるcase（templateCase）の作成

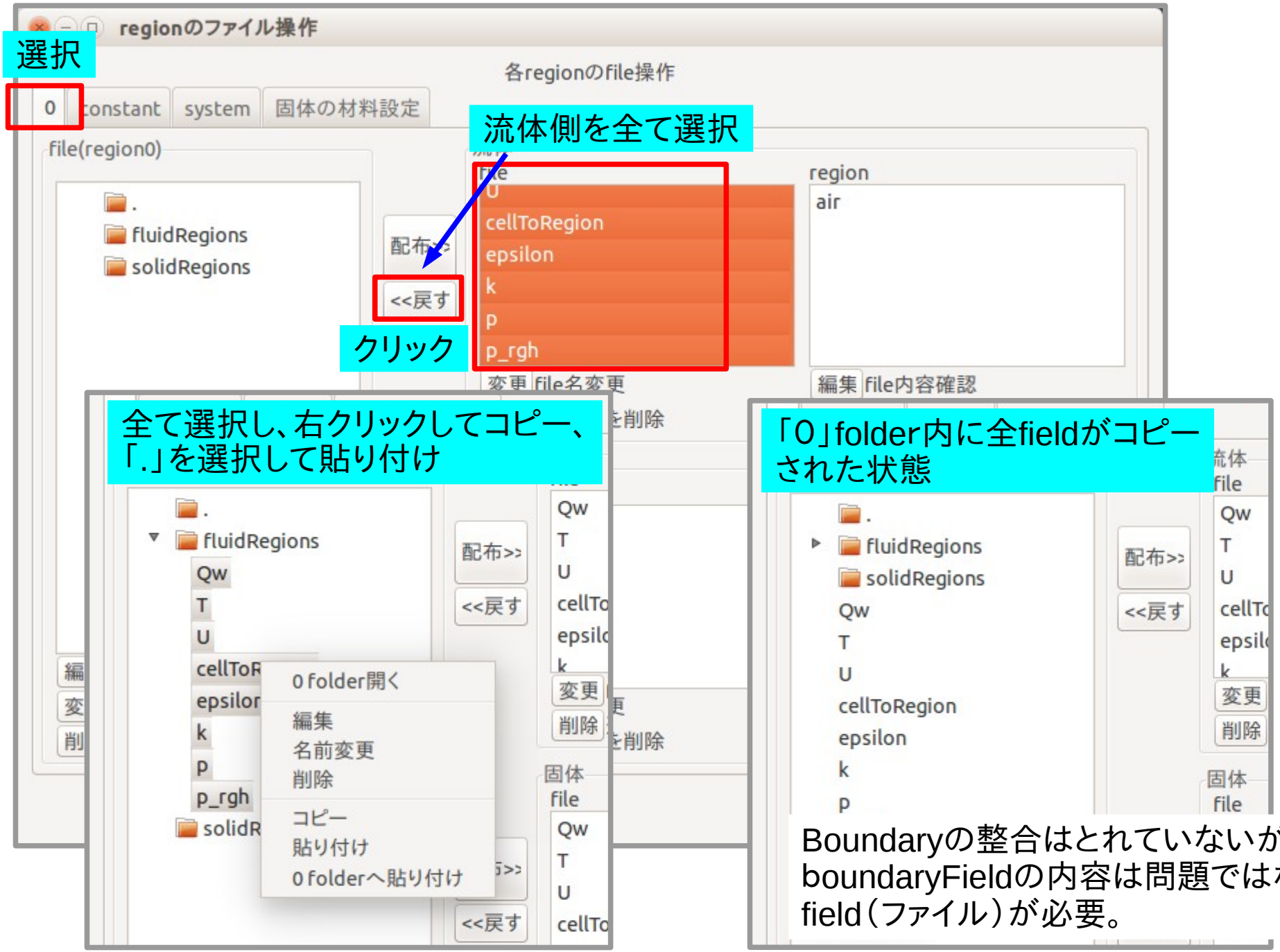
計算が走る事が確認できたので、見本となるcaseを作成する。
（今回の例で使用した「multiRegionCase」を作成してみる。）
caseの内容が変更されるので、caseコピーした上で、操作する

あるregionに特別なpropertiesやfvSchemeを保存する場合



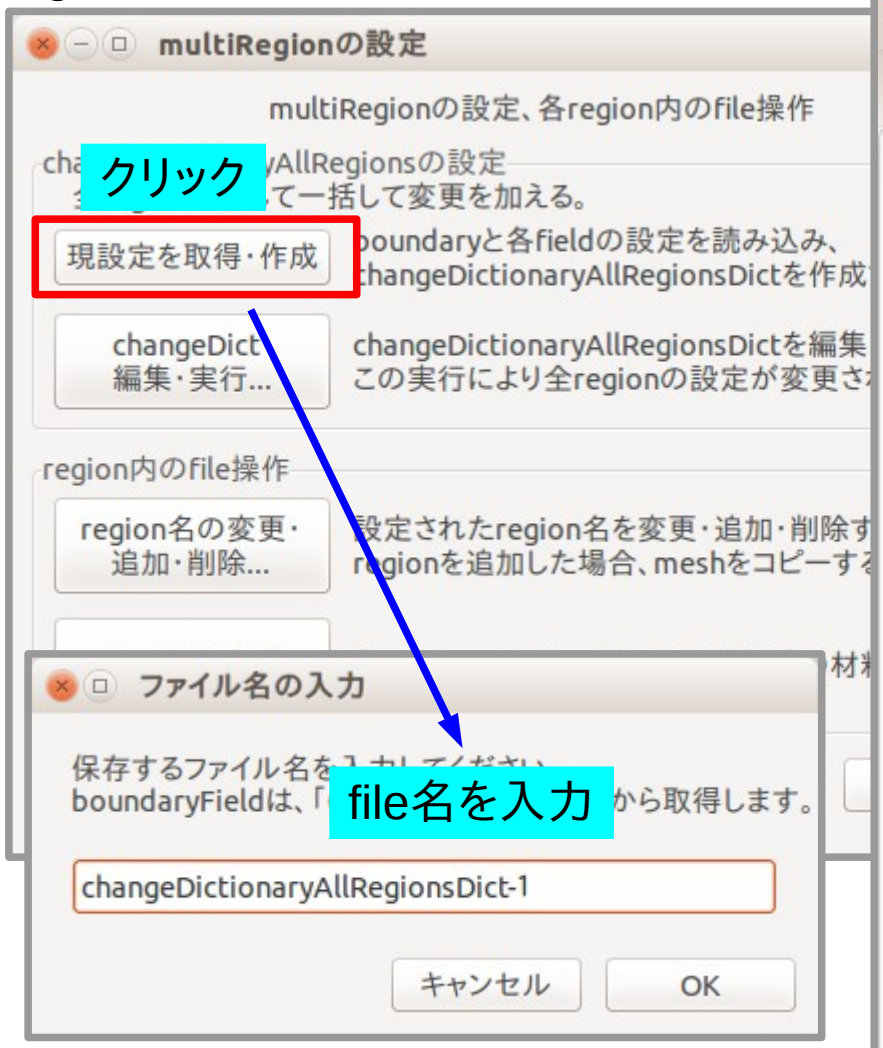
timeFolder内のFieldは、流体側の全fieldをregionの外に出しておく

全fieldを「0」folder直下に移動しておく



Boundaryの整合はとれていないが、boundaryFieldの内容は問題ではなくfield（ファイル）が必要。

各regionに設定した境界条件を保存する



保存された境界条件
（system/changeDictionaryAllRegionsDictの内容）

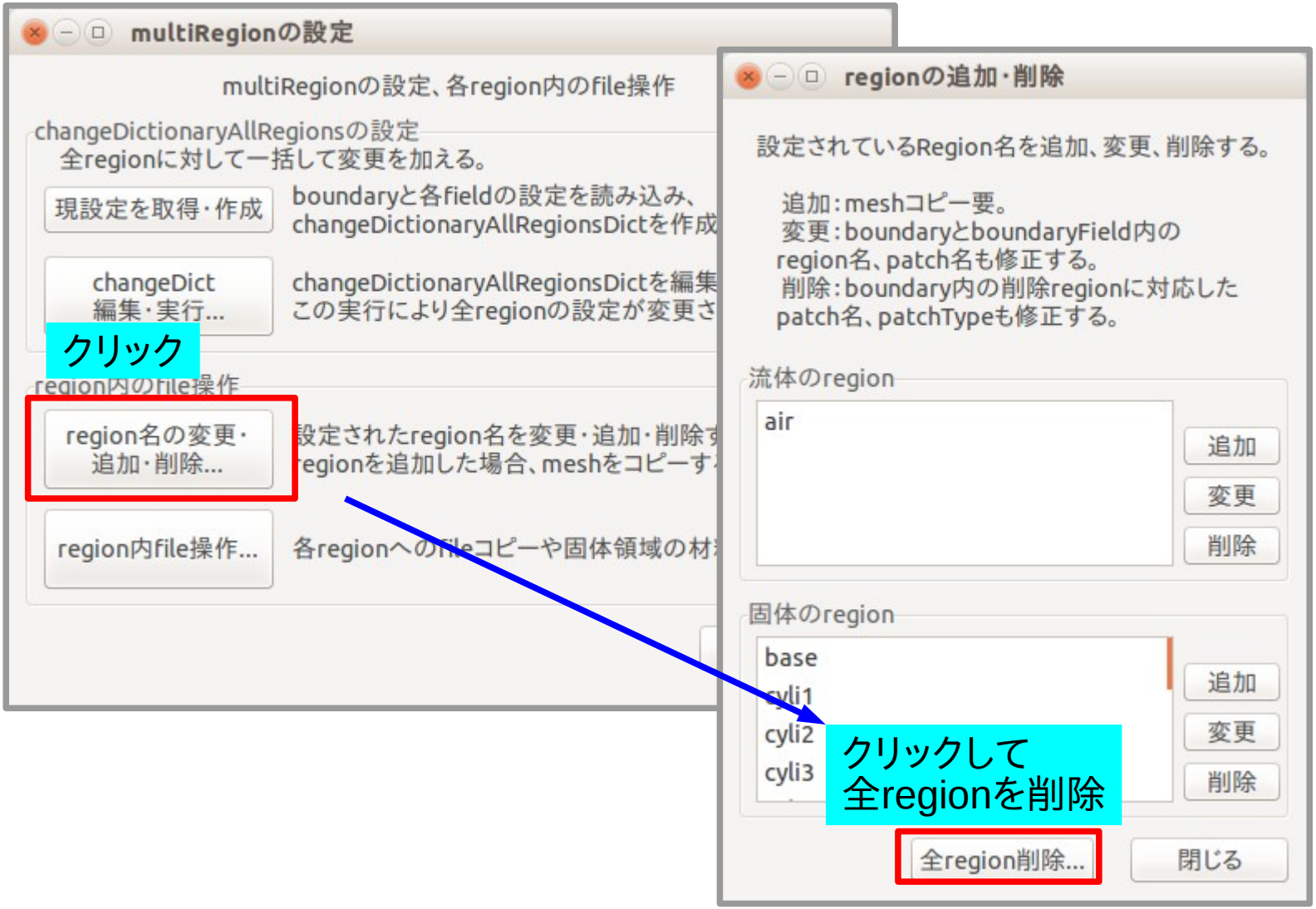
changeDictionaryAllRegionsDict-1

```
/*--------------------------------*- C++ -*----------------
| =========                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD
|  \\    /   O peration     | Version:  1.6.x
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|    \\/     M anipulation  |
\*---------------------------------------------------------
FoamFile
{
    version     2.3.x;
    format      ascii;
    class       dictionary;
    location    "system";
    object      changeDictionaryAllRegionsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

air
{
    dictionaryReplacement
    {
        boundary
        {
            air_to_base
            {
                type mappedWall;
                inGroups 1 ( wall );
                sampleMode nearestPatchFace;
                sampleRegion base;
                samplePatch base to air;
```

タブ幅: 4

全regionのchangeDictionaryDictを結合して
1ヶの「changeDictionaryAllRegionsDict」を作り保存する

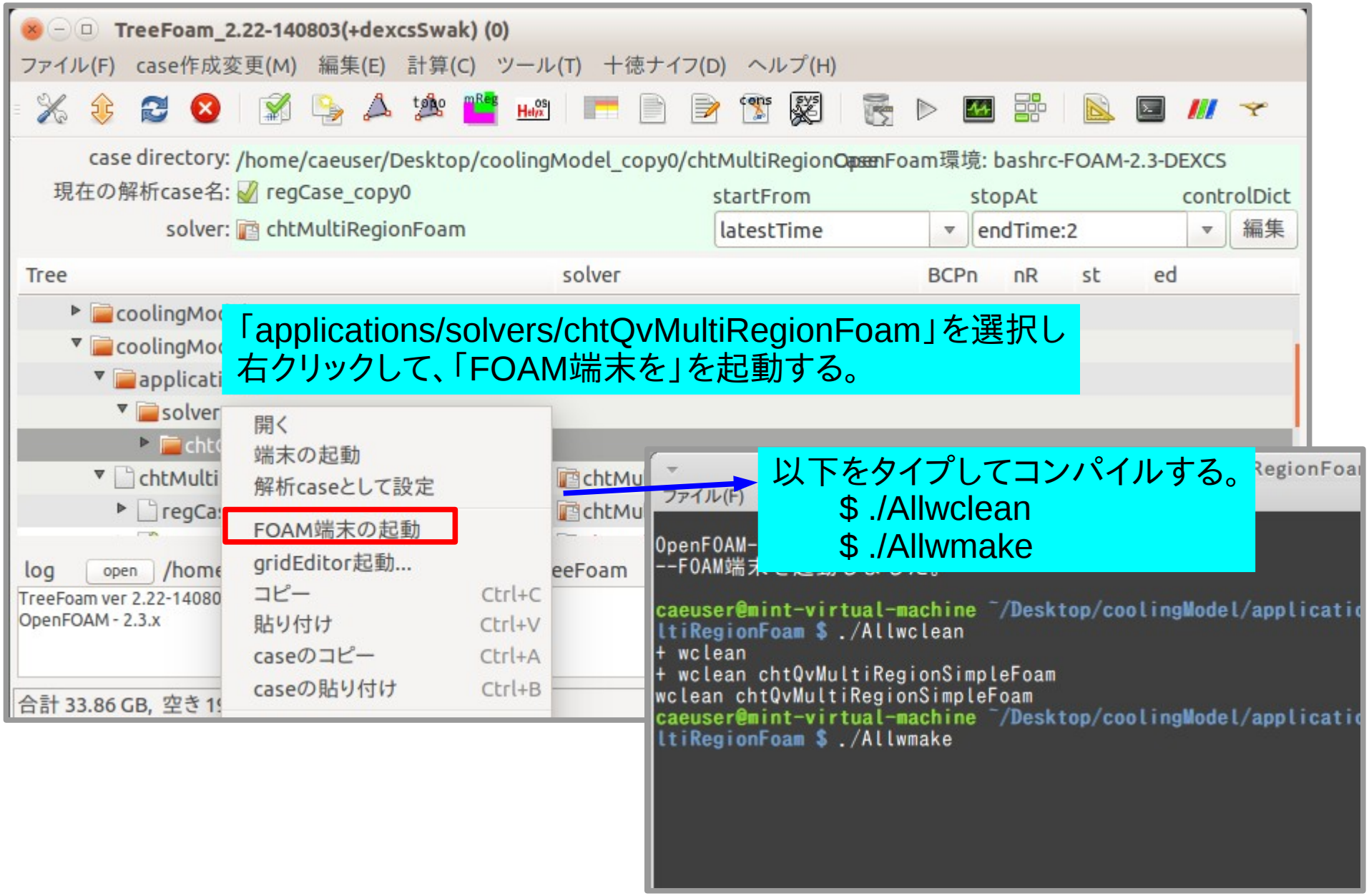
regionを削除して見本用のcaseを作成する。



見本用のcaseが完成したことになる。

9） regionが発熱する場合の計算

Qw fieldに発熱量（ワット数）をセットしているので、これを使って計算する。
発熱計算ができるように、solverを改造する。
（ソースコードがあるので、これをコンパイルする）



ソースコードの修正内容
＜chtQvMultiRegionFoam＞

createFluidFields.H

```
PtrList<fv::IOoptionList> fluidFvOptions(fluidRegions.siz

//----追加-----------
PtrList<volScalarField> QwFluid(fluidRegions.size());
//------------------


// Populate fluid field pointer lists
forAll(fluid Regions, i)
```

pointer追加

```
    );

    //-----追加-------
    Info<< "    Adding to QwFluid¥n" << endl;
    QwFluid.set
    (
        i,
        new volScalarField
        (
            IOobject
            (
                "Qw",
                runTime.timeName(),
                fluidRegions[i],
                IOobject::MUST_READ,
                IOobject::AUTO_WRITE
            ),
            fluidRegions[i]
        )
    );
    //-------------------
}
```

field追加

C/C++/ObjC のヘッダー ▾ タブの幅:: 4 ▾

createSolidFields.H

```
PtrList<volScalarField> betavSolid(solidRegions.size());

//-----追加------------
PtrList<volScalarField> QwSolid(solidRegions.size());
//--------------------

// Populate solid field pointer lists
forAll(solidRegions, i)
```

pointer追加

```
    //-----追加-------
    Info<< "    Adding to QwSolid¥n" << endl;
    QwSolid.set
    (
        i,
        new volScalarField
        (
            IOobject
            (
                "Qw",
                runTime.timeName(),
                solidRegions[i],
                IOobject::MUST_READ,
                IOobject::AUTO_WRITE
            ),
            solidRegions[i]
        )
    );
    //-------------------

}
```

field追加

C/C++/ObjC のヘッダー ▾ タブの幅:: 4 ▾

setRegionFluidFields.H

```
24
25      //---追加---------
26      volScalarField& Qw = QwFluid[i];
27      //----------------
28
29      const dimensionedScalar initialMass
30      (
31          "initialMass",
```

追加

setRegionSolidFields.H

```
43
44      const volScalarField& betav = betavSolid[i];
45
46      fv::IOoptionList& fvOptions = solidHeatSources[i];
47
48      //---追加---------
49      volScalarField& Qw = QwSolid[i];
50      //----------------
```

追加

EEqn.H

```
1  {
2      volScalarField& he = thermo.he();
3
4      //----追加-----------
5      dimensionedScalar partVol
6          ("partVol", dimVolume, gSum(mesh.V()));
7      volScalarField Qv = Qw / partVol;
8      //-------------------
9
10     fvScalarMatrix EEqn
11     (
12         fvm::ddt(rho, he) + fvm::div(phi, he)
13       + fvc::ddt(rho, K) + fvc::div(phi, K)
14       + (
15             he.name() == "e"
16           ? fvc::div
17             (
18                 fvc::absolute(phi/fvc::interpolate(rho), U),
19                 p,
20                 "div(phiv,p)"
21             )
22           : -dpdt
23         )
24       - fvm::laplacian(turb.alphaEff(), he)
25      ==
26         rad.Sh(thermo)
27       + fvOptions(rho, he)
28       //---追加----------
29       + Qv
30       //----------------
31     );
32
33     EEqn.relax();
```

追加

追加

solveSolid.H

```
6  {
7      //---追加------------
8      dimensionedScalar partVol
9          ("partVol", dimVolume, gSum(mesh.V()));
10     volScalarField Qv = Qw / partVol;
11     //-------------------
12
```

ワット数をregionの体積で除してW/m^{3}に単位変換

```
h=0; nonOrth<=nNonOrthCorr; nonOrth++)

Matrix> hEqn

17         fvm::ddt(betav*rho, h)
18       - (
19           thermo.isotropic()
20         ? fvm::laplacian(betav*thermo.alpha(), h, "laplacian(
21         : fvm::laplacian(betav*tAnialpha(), h, "laplacian(alp
22         )
23       ==
24         fvOptions(rho, h)
25       //---追加----------
26       + Qv
27       //----------------
28     );
29
30     hEqn().relax();
s.constrain(hEqn());
34     hEqn().solve(mesh.solver(h.select(finalIter)));
35
36     fvOptions.correct(h);
```

追加

発熱分を方程式に追加

ソースコードの修正内容
＜chtQvMultiRegionSimpleFoam＞

createFluidFields.H

```
PtrList<fv::IOoptionList> fluidFvOptions(fluidRegions.size());

//----追加-----------
PtrList<volScalarField> QwFluid(fluidRegions.size());
//------------------

// Populate fluid field pointer lists
forAll(fluidRegions, i)
{
```

追加

```
        new fv::IOoptionList(fluidRegions[i])
    );
    //-----追加-------
    Info<< "    Adding to QwFluid¥n" << endl;
    QwFluid.set
    (
        i,
        new volScalarField
        (
            IOobject
            (
                "Qw",
                runTime.timeName(),
                fluidRegions[i],
                IOobject::MUST_READ,
                IOobject::AUTO_WRITE
            ),
            fluidRegions[i]
        )
    );
    //------------------
}
```

追加

C/C++/ObjC のヘッダー ▾ タブの幅:: 4 ▾

setRegionFluidFields.H

```
fv::IOoptionList& fvOptions = fluidFvOptions[i];

//---追加---------
volScalarField& Qw = QwFluid[i];
//----------------

const dimensionedScalar initialMass
(
    "initialMass",
```

追加

EEqn.H

```
{
    volScalarField& he = thermo.he();

    //----追加-----------
    dimensionedScalar partVol
        ("partVol", dimVolume, gSum(mesh.V()));
    volScalarField Qv = Qw / partVol;
    //------------------

    fvScalarMatrix EEqn
    (
        fvm::div(phi, he)
      + (
            he.name() == "e"
          ? fvc::div(phi, volScalarField("Ekp", 0.5*magSqr(U) + p/rho))
          : fvc::div(phi, volScalarField("K", 0.5*magSqr(U)))
        )
      - fvm::laplacian(turb.alphaEff(), he)
     ==
        rad.Sh(thermo)
      + fvOptions(rho, he)
      //---追加----------
      + Qv
      //----------------
    );

    EEqn.relax();
```

追加

追加

solveSolid.H

solveSolid.H

```
{
    //---追加------------
    dimensionedScalar partVol
        ("partVol", dimVolume, gSum(mesh.V()));
    volScalarField Qv = Qw / partVol;
    //-------------------

    for (int nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)
    {
        fvScalarMatrix hEqn
        (
            thermo.isotropic()
            ? -fvm::laplacian(betav*thermo.alpha(), h, "laplacian(alpha,h)")
            : -fvm::laplacian(betav*tAnialpha(), h, "laplacian(alpha,h)")
          ==
            fvOptions(rho, h)
          //---追加----------
          + Qv
          //-----------------
        );

        hEqn.relax();

        fvOptions.constrain(hEqn);

        hEqn.solve();

        fvOptions.correct(h);
    }
}

thermo.correct();

Info<< "Min/max T:" << min(thermo.T()) << ' ' << max(thermo.T()) << endl;
```

追加

追加

C/C++/ObjC のヘッダー ▾ タブの幅:: 4 ▾ (1行、1列) [挿入

「chtMultiRegionFoam」が走ったregCaseを「caseコピー」し、新しくcaseを作成する。
controlDict内のapplicationを修正する。

TreeFoam_2.22-140803(+dexcsSwak) (0)

ファイル(F) case作成変更(M) 編集(E) 計算(C) ツール(T) 十徳ナイフ(D) ヘルプ(H)

case directory: /home/caeuser/Desktop/coolingModel_copy0/chtMultiRegionCase OpenFoam環境: bashrc-FOAM-2.3-DEXCS

現在の解析case名: regCase_copy1

solver: chtMultiRegionFoam

startFrom: latestTime | stopAt: endTime:2 | controlDict: 編集

クリックしてctrolDictを開く

Tree | solver | BCPn | nR

- applications
- chtMultiRegionCase — chtMultiRegion
  - regCase — chtMultiRegion
  - regCase_copy0 — chtMultiRegion
  - regCase_copy1 — chtMultiRegion

コピーしたcaseを解析caseに設定する
（レ点マークを付ける）

/home/caeuser/TreeFoam/temp/0_logTreeFoam

copy: /home/caeuser/Desktop/coolingModel_copy0/chtMultiRegionCase/regCase_copy1/r
copy: /home/caeuser/Desktop/coolingModel_copy0/chtMultiRegionCase/regCase_copy1/r
copy: /home/caeuser/Desktop/coolingModel_copy0/chtMultiRegionCase/regCase_copy1/s

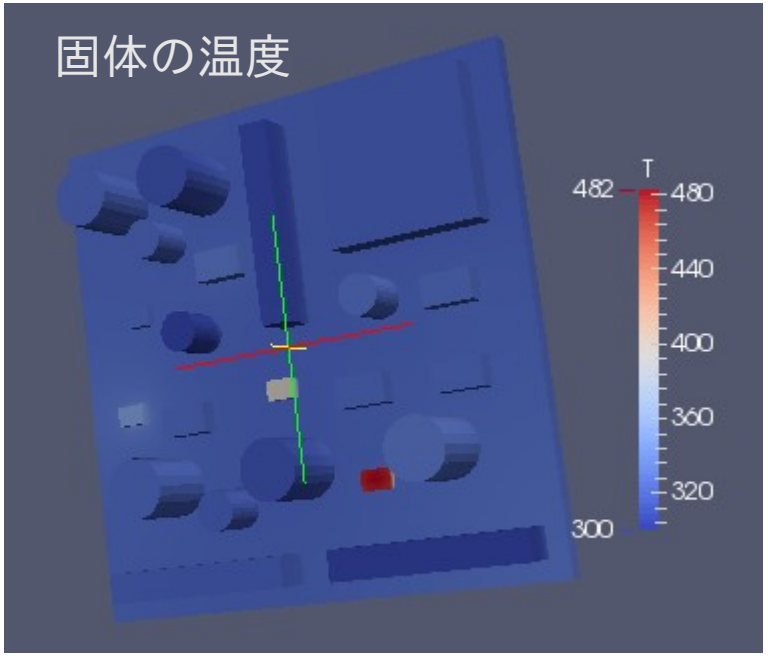
合計 33.86 GB, 空き 19.76 GB

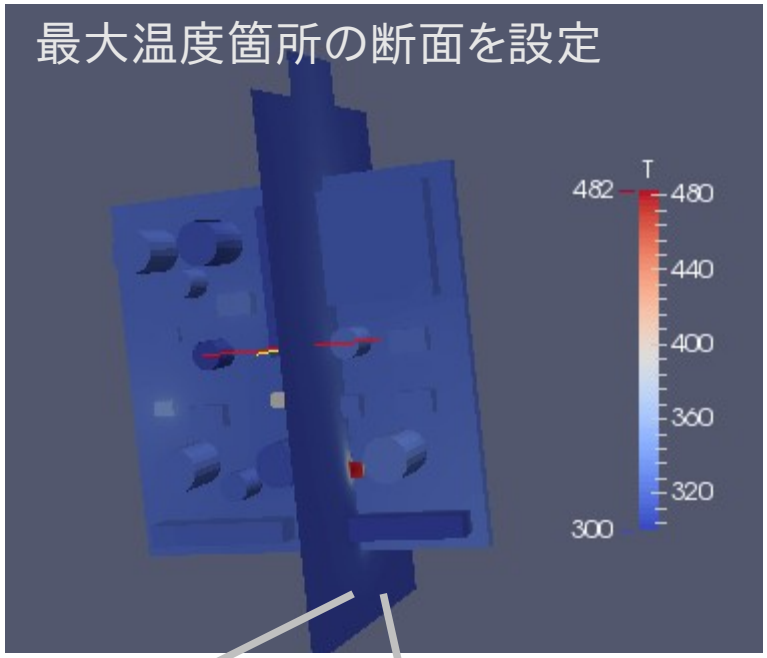
```
 4 |  ¥¥    /   O peration     | Version:  2.2.2
 5 |   ¥¥  /    A nd           | Web:      www.OpenFOAM.org
 6 |    ¥¥/     M anipulation  |
 7 ¥*---------------------------------------------------------------
 8 FoamFile
 9 {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     location    "system";
14     object      controlDict;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
17
18 libs
19 (
20     "libcompressibleTurbulenceModel.so"
21     "libcompressibleRASModels.so"
22 );
23
24 //application     chtMultiRegionFoam;
25 application     chtQvMultiRegionFoam;
26
27 startFrom
28
29 startTime       0;
30
31 stopAt          endTime;
```

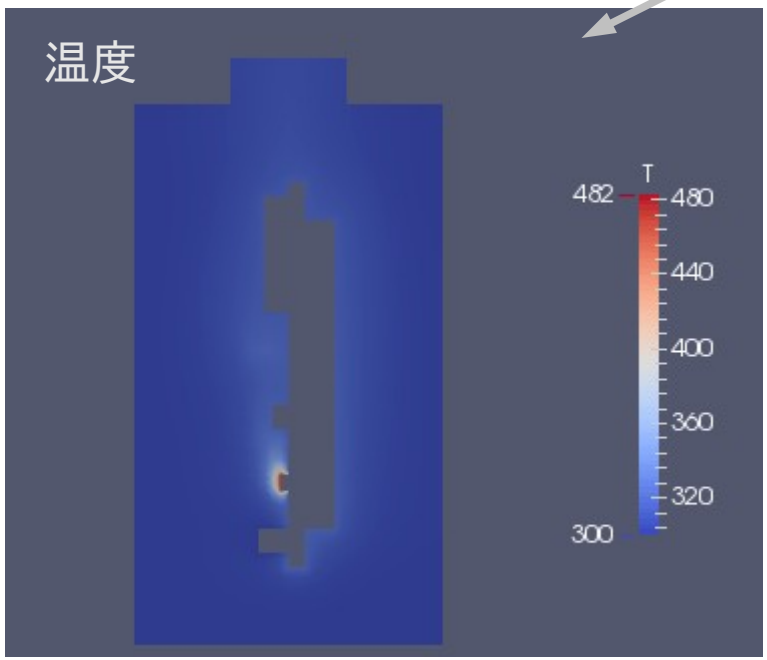
修正して保存する

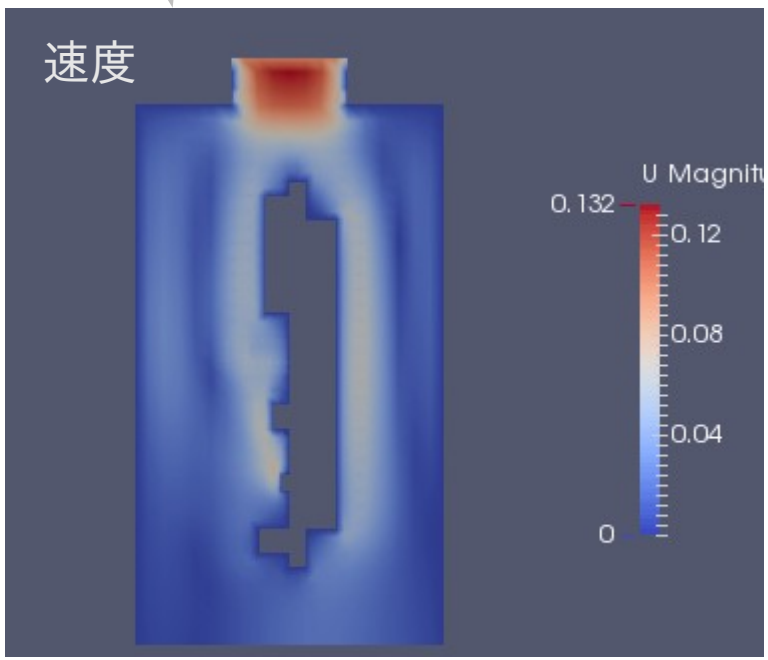
設定は、全てできているので、controlDict修正後、solverを実行する

計算結果（発熱を含む場合）









3. まとめ

chtMultiRegionFoamが簡単に使える様にTreeFoamを改造してきた。

今回のように、部品点数が多いと、editorで条件設定をするのは、不可能に近い。 TreeFoam上からでは、数時間で可能。

メッシュ作成、データセット、固体の材料設定に関しては、予め、設定用のcsvファイルを作成する事で設定できる。これらcvsファイルは、stlファイル名がそのまま利用できるので、モデルが出来上がった時点で設定できる。