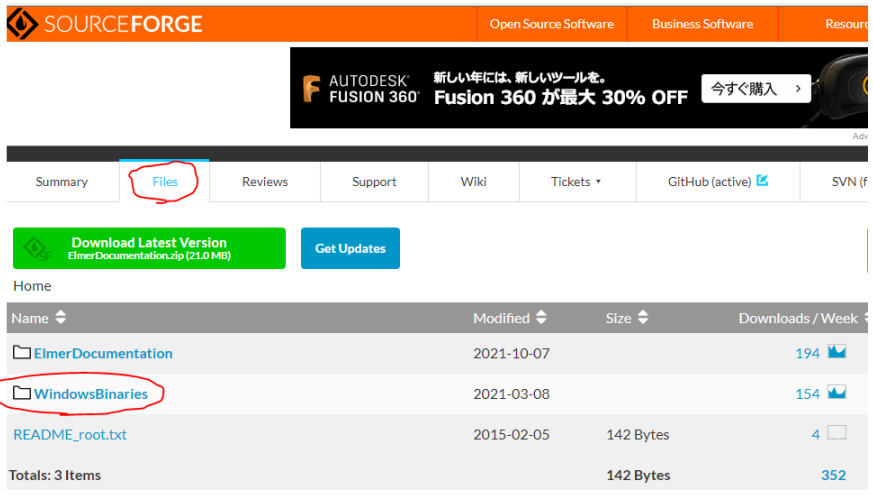
1. ソフトウェアのインストール

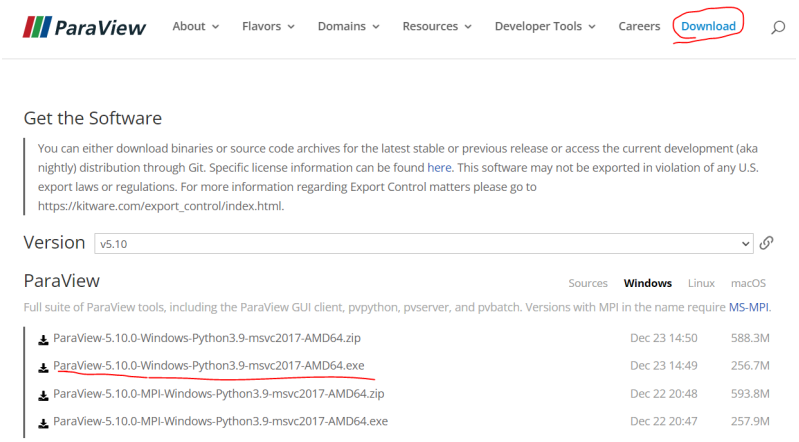
Elmerのインストール
https://www.csc.fi/web/elmer
にあるリンクから
https://sourceforge.net/projects/elmerfem/



DLしたEXEファイルを実行するだけでインストールされる。

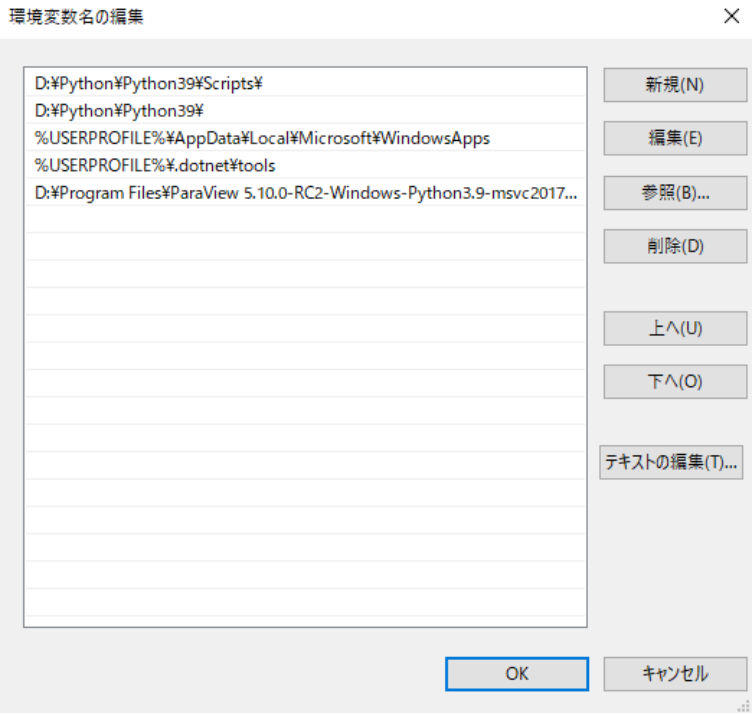
ParaView
グラフの可視化のためにParaViewをインストールする。

https://www.paraview.org/



ダウンロードしたファイルを実行するとインストールされる。続いて、環境変数の編集
paraviewのインストールフォルダのbinフォルダを環境変数に追加する。
正しく設定されているとparaviewとコマンドプロンプトからタイプすると起動してくる。

環境変数の設定



GMSH

https://gmsh.info/

モデル形状の作製に今回はGMSHを使う。
Pythonがインストールされ、環境変数にpythonのインストールフォルダとスクリプトにPATHが通っており、pipのインストールをしてある事が前提。環境に依存するが、下記のようなフォルダ。
C:\Python\PythonXX\Scripts\; C:\Python\PythonXX\
にパスが通っており、
コマンドプロンプトから
pip -V
と入力した際、pipのバージョン情報を見る事ができれば問題はない。
GMSHのインストール自体は、
pip install –upgrade gmsh
とコマンドプロンプトから入力するだけ。

pipがインストールされていない、pipがバージョン違いで動かない場合は、
https://pip.pypa.io/en/stable/installation/

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
として、get-pipをDLし
python get-pip.py
で実行する。pipがインストールされる。

環境変数の編集
paraviewのインストールフォルダのbinフォルダを環境変数に追加する。
正しく設定されているとparaviewとタイプすると起動してくる。

2. 熱解析の流れ
石英系PLCにおけるTOスイッチにおける消費電力を求める事をめざす。

JOURNAL OF LIGHTWAVE TECHNOLOGY, VOL. 26, NO. 14, JULY 15, 2008 2235

が具体的な実験結果になり、これを再現する事を目指す。最終的には、実験結果は計算より常に10mW程消費電力が大きく、電極への熱流出等が示唆される結果が得られる。
計算は、上面から一定の熱流入（Flux）を与え、底面が定常25℃に保たれているとした場合の温度分布をFEMで計算する事になる。

計算の流れ
熱分布の計算の手順は
① 計算する構造をGMSHのスクリプトをpythonで記載して定義し、GMSHでメッシュを切る。
② メッシュを切ったデータをElmerにインポートして、材料特性、与える熱量や定常境条件等を与えて熱分布を計算する。
③ Paraviewを用いて熱分布の解析を行う。

3. 解析構造の作成

構造はGMSHを用いて定義する。pipを用いてインストールしたが、https://gmsh.info/にwindows用のインストーラーありこれをダウンロードするとTutorialsがあり、その下にpythonによる構造定義の例が見えるのでこれを開いて実行、コードを読む事を推奨する。

今回、計算の対象とする構造は以下の構造である。Si基板上に形成されたPLCのTO位相変調器の断面図となる。断熱溝を有し、断熱溝で区切られるリッジ部分にコアがあり、その上面にヒータが設置されている。
アンダークラッドの厚さは43um、オーバークラッドの厚さ20um、ヒータ厚さは1.0um、ヒータ幅20um、リッジ幅30um

実際この構造をGMSHで構造を入力して、メッシュを切る。具体的なpythonのコードは以下になる。コードをコピーし適当な名前+".py"の拡張子で保存する。

```
# ----------------------------------------------------------------------------
#  Gmsh Python
#   Kei Watanabe   NTT
#   This is a test for JOURNAL OF LIGHTWAVE TECHNOLOGY, VOL. 26, NO. 14, JULY 15, 2008 2235
#   Ultralow Power Consumption Silica-Based PLC-VOA/Swithes
# ----------------------------------------------------------------------------

import gmsh
import sys

# Before using any functions in the Python API, Gmsh must be initialized:
gmsh.initialize()

# add new model
gmsh.model.add("tmesh")

#------POINT------
# mesh size
lc=0.000001

si_thick=0.00015
si_width=0.0005

ridge_width=0.00003
underclad  =0.000043
overclad    =0.000020
coresize    =0.0000045
ridge_hight=overclad+underclad

corehight=underclad+coresize/2
t_heater=0.000001
w_heater=0.00002
```

```
p1=gmsh.model.geo.addPoint(-si_width/2, -corehight-si_thick, 0, 10*lc)
p2=gmsh.model.geo.addPoint( si_width/2, -corehight-si_thick, 0, 10*lc)
p3=gmsh.model.geo.addPoint( si_width/2, -corehight, 0, 10*lc)
p4=gmsh.model.geo.addPoint(-si_width/2, -corehight, 0, 10*lc)

p5=gmsh.model.geo.addPoint(-ridge_width/2, -corehight, 0, lc)
p6=gmsh.model.geo.addPoint( ridge_width/2, -corehight, 0, lc)

p7=gmsh.model.geo.addPoint( ridge_width/2,  overclad-coresize/2, 0, lc)
p8=gmsh.model.geo.addPoint( w_heater/2,       overclad-coresize/2, 0, lc)
p9=gmsh.model.geo.addPoint(-w_heater/2,       overclad-coresize/2, 0, lc)
p10=gmsh.model.geo.addPoint(-ridge_width/2, overclad-coresize/2, 0, lc)

p11=gmsh.model.geo.addPoint(-w_heater/2,t_heater+overclad-coresize/2, 0, lc/2)
p12=gmsh.model.geo.addPoint( w_heater/2,t_heater+overclad-coresize/2, 0, lc/2)



#------LINE------
L1=gmsh.model.geo.addLine(p1, p2)
L2=gmsh.model.geo.addLine(p2, p3)
L3=gmsh.model.geo.addLine(p3, p6)
L4=gmsh.model.geo.addLine(p6, p5)
L5=gmsh.model.geo.addLine(p5, p4)
L6=gmsh.model.geo.addLine(p4, p1)

L7=gmsh.model.geo.addLine(p6, p7)
L8=gmsh.model.geo.addLine(p7, p8)
L9=gmsh.model.geo.addLine(p8, p9)
L10=gmsh.model.geo.addLine(p9, p10)
L11=gmsh.model.geo.addLine(p10, p5)

L12=gmsh.model.geo.addLine(p8, p12)
L13=gmsh.model.geo.addLine(p12, p11)
L14=gmsh.model.geo.addLine(p11, p9)

#------LOOP------
LOOP1=gmsh.model.geo.addCurveLoop([ L1, L2, L3, L4, L5 ,L6])
LOOP2=gmsh.model.geo.addCurveLoop([-L4, L7, L8, L9, L10,L11])
LOOP3=gmsh.model.geo.addCurveLoop([-L9,L12,L13,L14])
#------SURFACE------
SURFACE1=gmsh.model.geo.addPlaneSurface([LOOP1])
SURFACE2=gmsh.model.geo.addPlaneSurface([LOOP2])
SURFACE3=gmsh.model.geo.addPlaneSurface([LOOP3])
```

```
#------SYNCHRONIZATION------
gmsh.model.geo.synchronize()
# addPhysicalGroup(demension, group)
PG1=gmsh.model.addPhysicalGroup(1, [L1])
PG2=gmsh.model.addPhysicalGroup(1, [L2])
PG3=gmsh.model.addPhysicalGroup(1, [L3])
PG4=gmsh.model.addPhysicalGroup(1, [L4])

PG5=gmsh.model.addPhysicalGroup(1, [L5])
PG6=gmsh.model.addPhysicalGroup(1, [L6])
PG7=gmsh.model.addPhysicalGroup(1, [L7])
PG8=gmsh.model.addPhysicalGroup(1, [L8])
PG9=gmsh.model.addPhysicalGroup(1, [L9])

PG6=gmsh.model.addPhysicalGroup(1, [L10])
PG7=gmsh.model.addPhysicalGroup(1, [L11])
PG8=gmsh.model.addPhysicalGroup(1, [L12])
PG9=gmsh.model.addPhysicalGroup(1, [L13])

PGS1=gmsh.model.addPhysicalGroup(2, [SURFACE1])
PGS2=gmsh.model.addPhysicalGroup(2, [SURFACE2])
PGS3=gmsh.model.addPhysicalGroup(2, [SURFACE3])



# Generate a 2D mesh
gmsh.model.mesh.generate(2)

# Save meshdata
gmsh.write("tmesh.msh")

# Visualizing of the genarated mesh
#
if '-nopopup' not in sys.argv:
    gmsh.fltk.run()

#------FINISH------
```

GMSHコード内容
ポイント
まずは、構造中の”点”を定義する。点をaddPointを用いる。
p1=gmsh.model.geo.addPoint(x, y, z, m)
点のx,y座標（3次元の場合はzも）が第1、2引数であり、最後のｍはメッシュの細かさを示す。p1の戻り値には、この点を指す番号が入る。

点が定義できたら、次にそれらを結んで線を定義する。addLineで線を定義する。

L1=gmsh.model.geo.addLine(p1, p2)

点で指定した番号2つを指定して線を定義していく。
同様に、線をひとまとめとして、ループを作る。ループは閉じている必要がある。

LOOP1=gmsh.model.geo.addCurveLoop([L1, L2, L3, L4, L5 ,L6])

引数の部分は配列で渡し、上記の場合はSi基板の部分をループとして定義している事になる。
これが閉じた平面であるとして最後にsurfaceを定義する。

SURFACE1=gmsh.model.geo.addPlaneSurface([LOOP1])

複数のループを一つの面として定義する事も可能であるが、本例では一つの閉じた面を一つのSURFACEとして定義している。

これでおおむね構造の入力（CADによる図形の描画）は終了となる。

gmsh.model.geo.synchronize()

このコマンドは、メッシュを切るため内部のCADデータをGMSHの形式に変換するコマンドになる。

続いて、実際ElmerでFEMの計算を実施する際に、境界条件を与えたり、材料の定義を行うために、PhysicalGroupを定義しておく。addPhysicalGroup(demension, group)で指定する。
demensionは構造の次元（点ならゼロ、線は1、面は2、立体は3）で、groupは配列で構造を定義する。

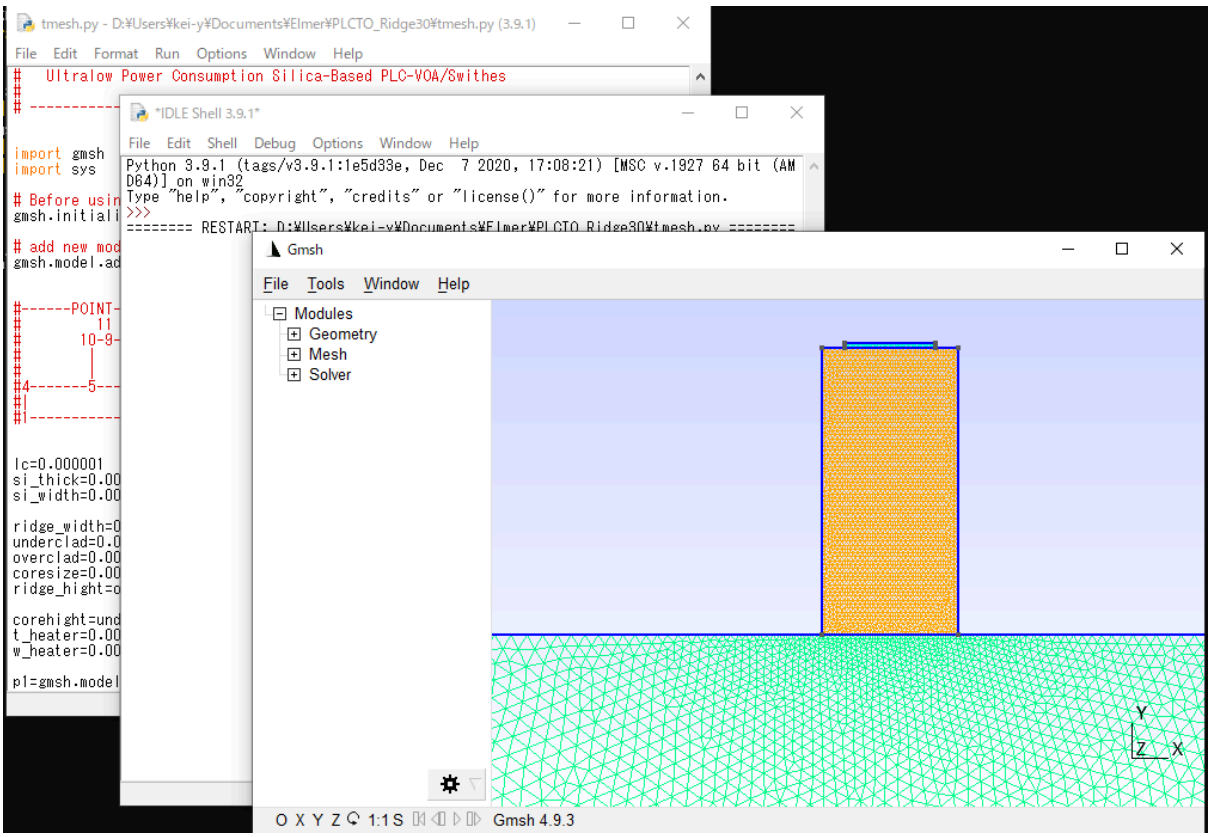
PG1=gmsh.model.addPhysicalGroup(1, [L1])

といった具合である。
これで、メッシュを切り、出力をファイルに保存し終了する。

gmsh.model.mesh.generate(2)　←()の中は次元
gmsh.write("tmesh.msh")　←” ”の中は保存するファイル名
gmsh.finalize()　←最後の行に配置する。

終了前に、
　gmsh.fltk.run()
を呼び出す事で、GMSHのGUIが立ち上がり表示が行われる。表示が必要ない場合は割愛する。

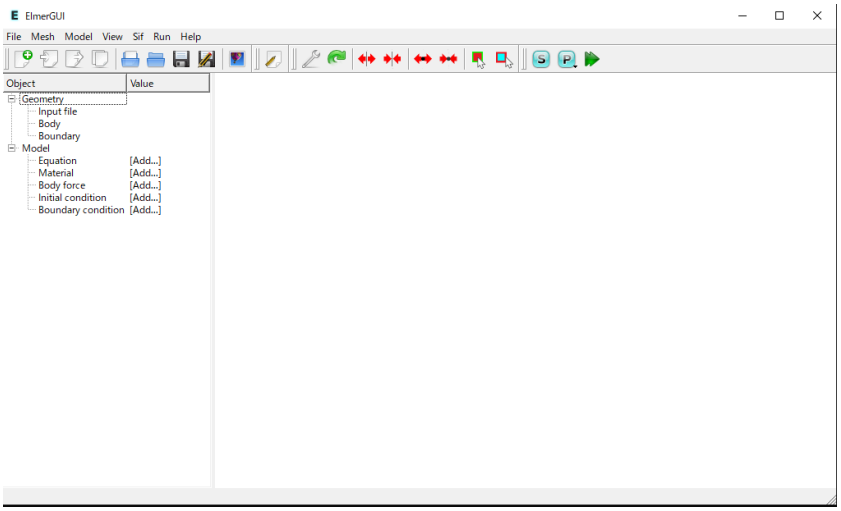
保存した.pyファイルをIDLEで開き（ファイルを右クリックし、Edit with IDLE）、Run-Run moduleで実行する。問題なければ、IDLE shellが立ち上がり、GMSHのGUIも立ち上がりメッシュが切られた図形が表示される。



注意点は、2つの面が接している場合、例えばこの例の場合、Si基板とガラスからなるリッジ部分が点4，5、線4でつながっている。同じ形を作るだけであれば、LOOPをP1、P2、P3、P4とだけつなげればよい。しかしこのようにモデルをCAD構築すると、この境界でFEMの計算が連続的に行われない。つまり物理的につながっていない計算になってしまう。そのため各点で分割し、LOOPを形成する際に、それらを繋ぎ合わせる必要がある。正しい計算をするためにこの点には注意されたい。

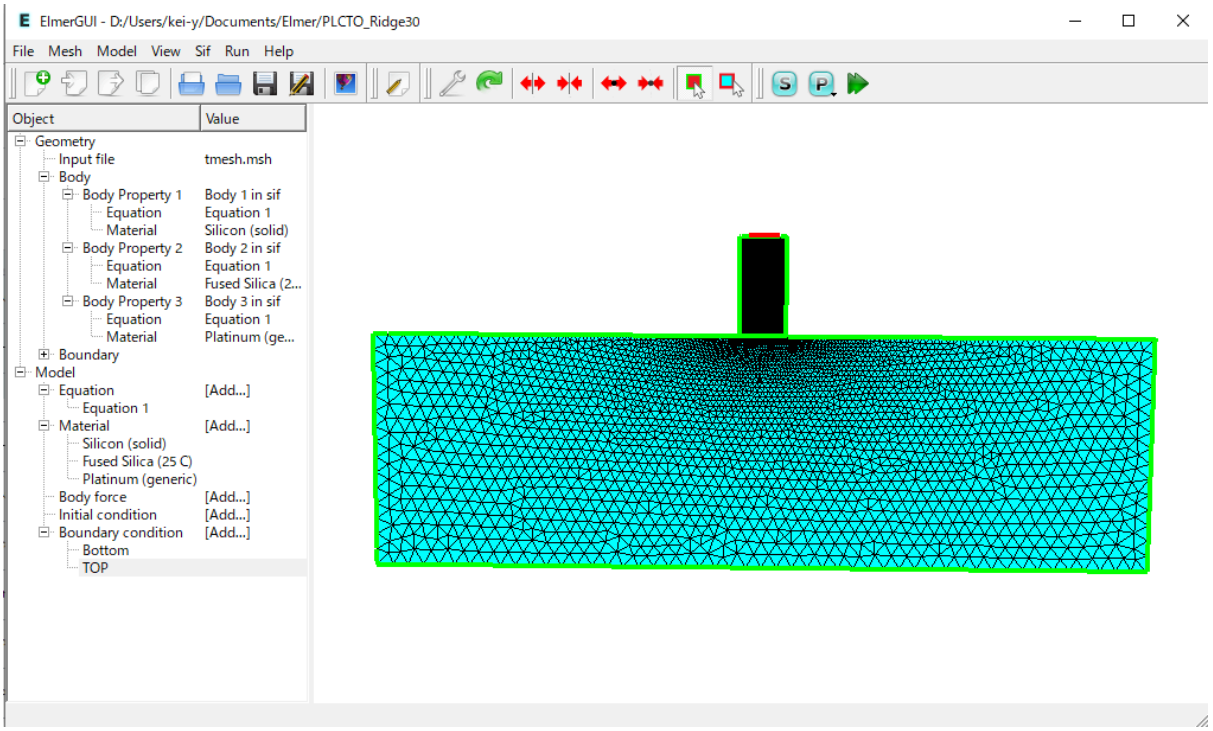
4．Elmerによる熱分布の計算

ElmerGUIを立ち上げる。



第3章で作製したメッシュデータの読み込みを行う。

File- Open を指定するとダイアログが開くため、前章で保存されたメッシュデータを読み込む。拡張しは.mshとなっている。この例の場合tmesh.mshとなる。読み込みができればElmer上に構造が表示される。この時、GMSHで定義した物理グループ（PhysicalGroup）が構造物として認識される。



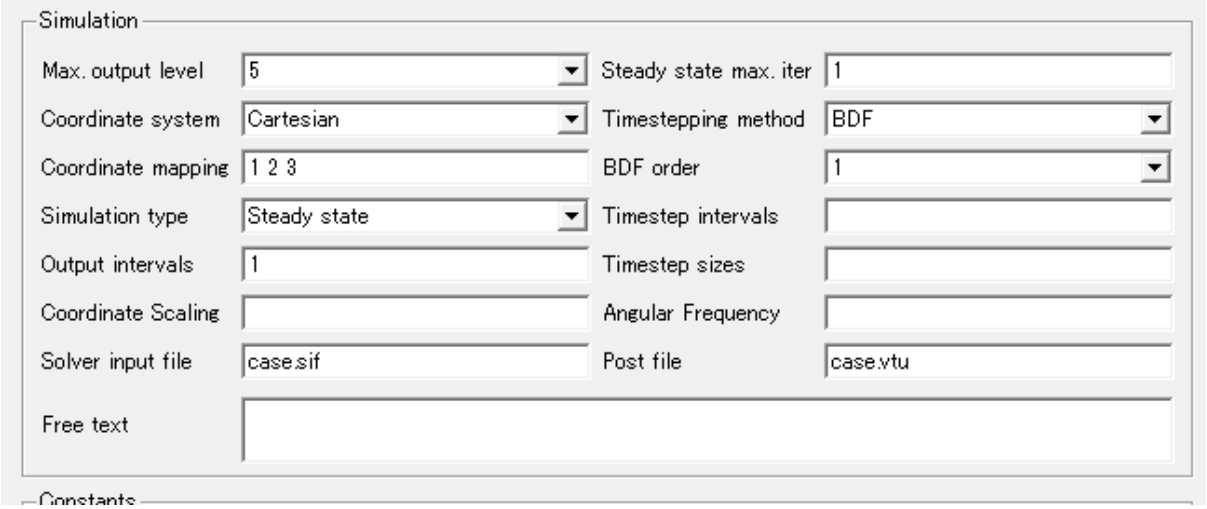
この例題では定常状態について計算する。時間でどのように変化するか計算する場合は別途チュートリアルを参考にすれば時間応答も求められる。

ElmerでのFEM計算の流れは以下となる。
初期設定（SetUp）
微分方程式の指定（Equation)
物性値の設定（Material)
境界条件の設定（Boundary condition)
ここまでが設定事項となる。
基本的にModelのタグを上から順番に設定をすれば良い事になる。
この例題（定常状態の計算）では用いないが、
body force:その表面自体から発熱する場合等
Initial condition: 時間応答を計算する場合は初期状態が必要でその定義を行う。
がある。詳しく知りたい場合はチュートリアルを参照されたい。

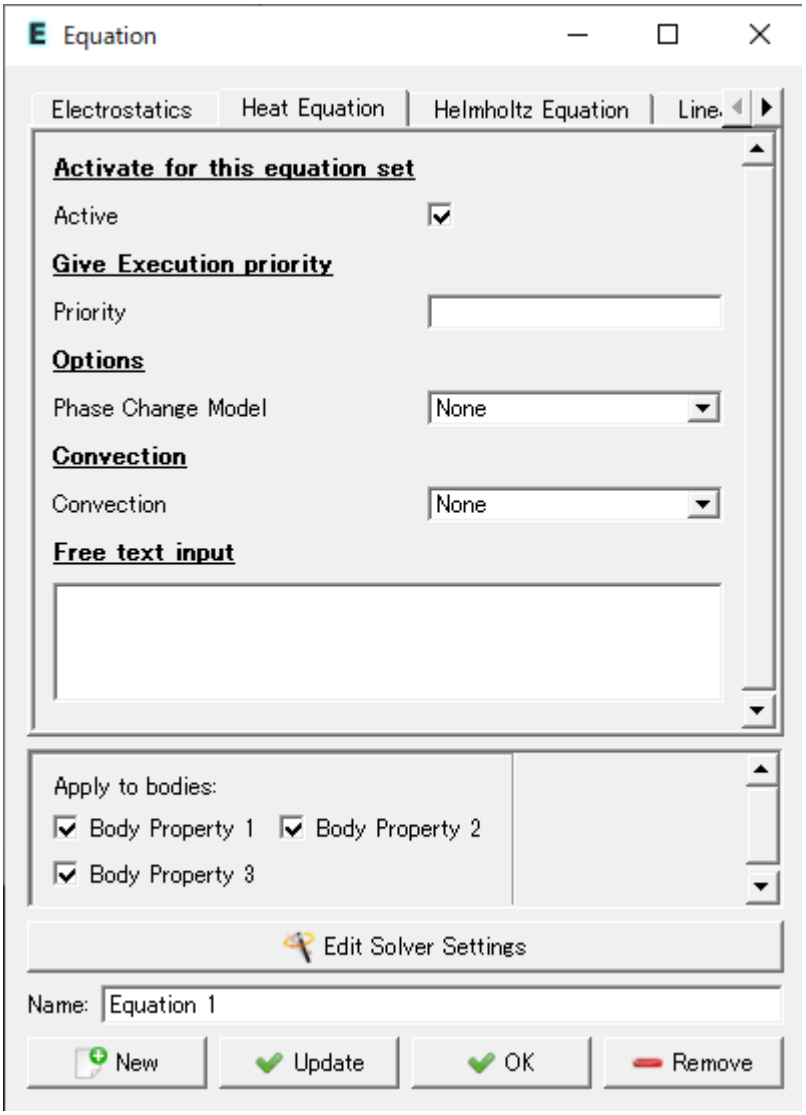
その後、
実行ファイルの生成（Sif）
上記設定が記載されたファイルでそのファイルに記載された内容で計算をする。そのファイルの書き出し作業。この時点で編集を行い内容を変更する事もできる。
上書き保存。この作業が重要で、実行する前に保存しないと反映されない。
そしてFEMの計算をするというのが全体の流れとなる。

実際の各種の設定をはじめる。

設定
Setup
Simulation type = Steady state
BDF order = 1
Applyを押す



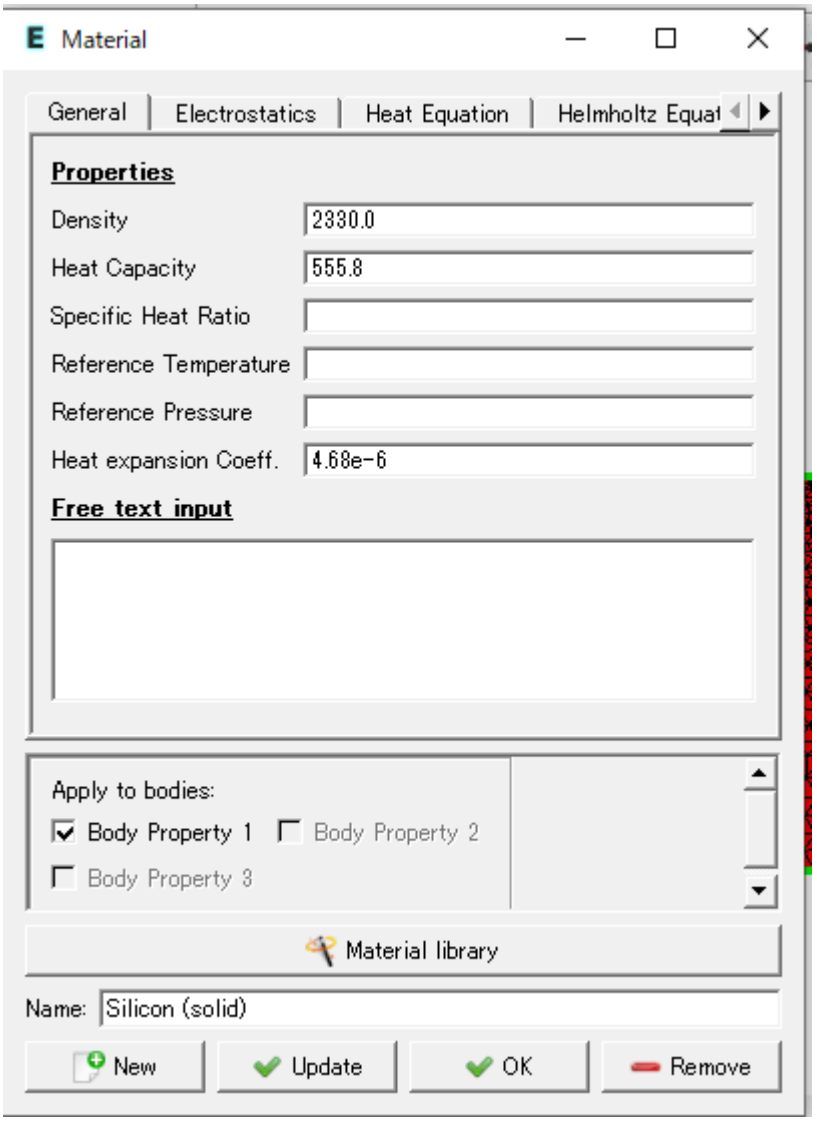
Equation一Add
Heat Equation のタブを選択



Name：Equation1
Activeのチェックを入れる
Body Property1、Body Property2, Body Property3のチェックを入れる。
OKを押す。

それぞれの領域において適応する微分方程式の選択を行う。ここではすべての領域で熱伝導方程式を適応するため3つのBodyにチェックを入れる。Nameは呼称のため適当な区別できる適当な名前であればよい。

Material- Add
Material Libraryのボタン
リストからSilicon(solid)を選択し、OKを押す
Body Property1にチェックを入れる
OKを押す



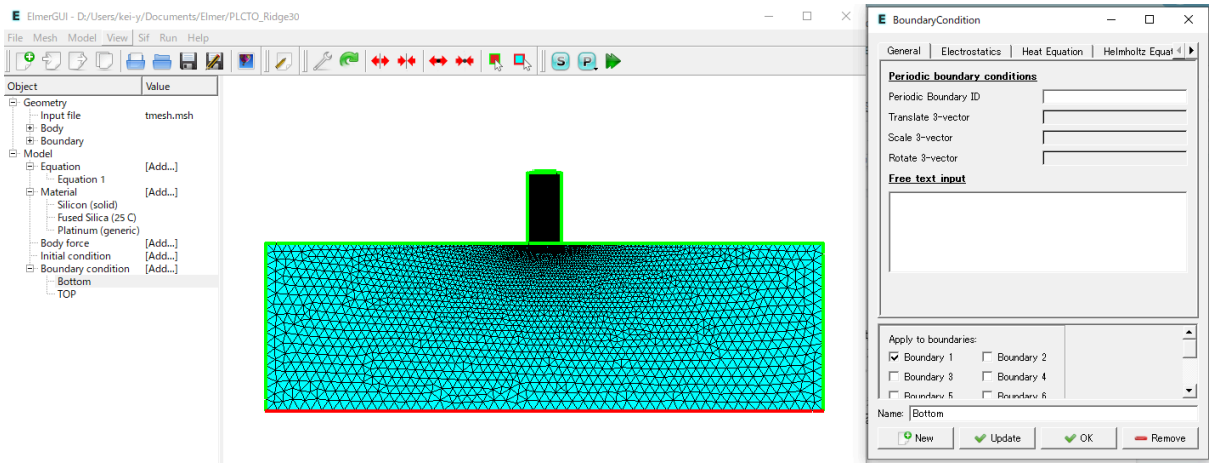
Material- Add
Material Libraryのボタン
リストからFused Silica (25 C)を選択し、OKを押す
Body Property2にチェックを入れる
OKを押す

Material- Add
Material Libraryのボタン
リストからSilicon(solid)を選択し、OKを押す
リストからPlatinum (generic)を選択し、OKを押す
Body Property3にチェックを入れる
OKを押す

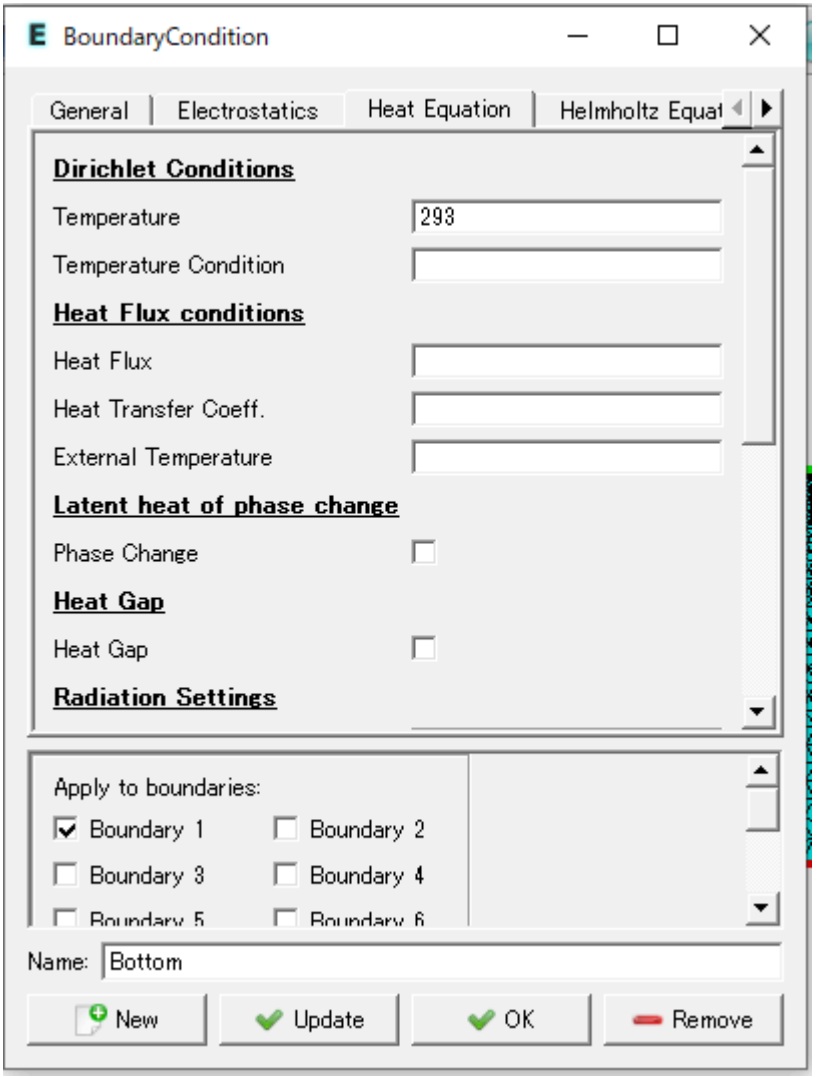
これで領域1はSi基板なのでSiの物性値、リッジ部分の領域2は石英、ヒータ部分はプラチナの物性値が設定される。なお、ヒータは別の材料であるがガラス等に比べて金属は圧倒的な熱伝導を有しているため適当な金属を指定したまでである。他の金属に変更しても計算結果は全くと言って良い程変わらない。

また物性値はリストから選択したが、リストに無い材料を利用する場合は直接、Propertiesにある物性値を調べて入力し、Nameに適当な名前をいれるようにする。

Boundary condtion―Add

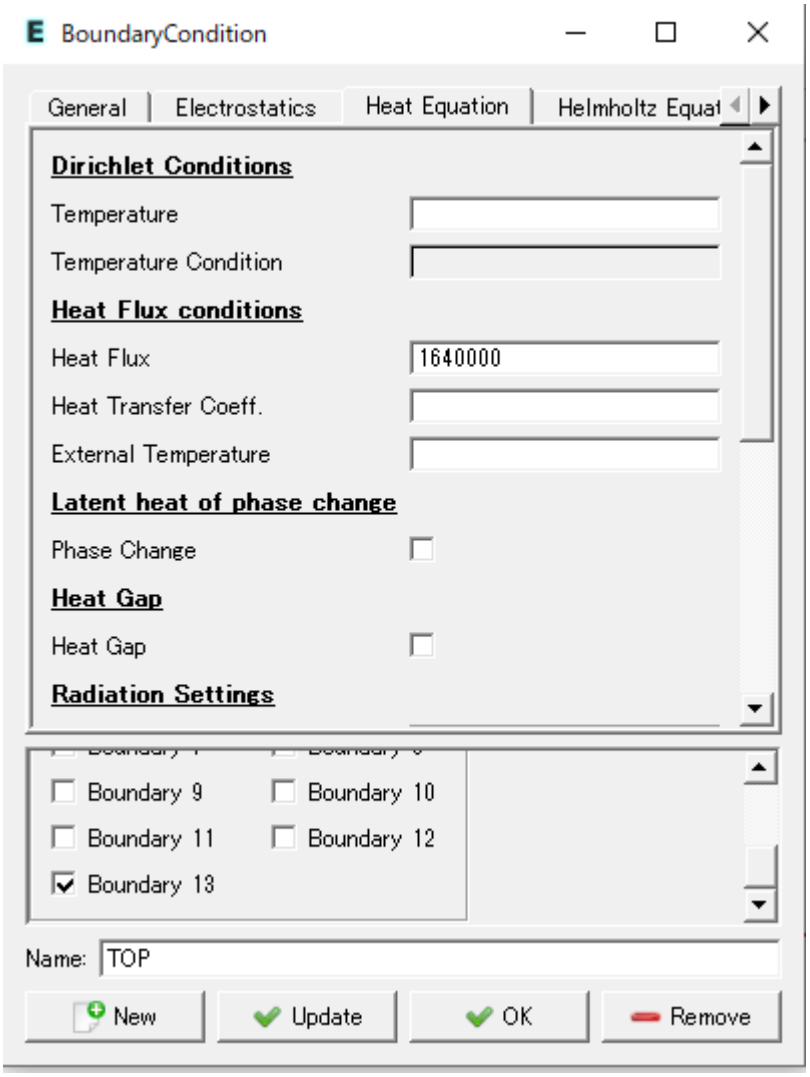


Heat Equationのタブを選択
Dirichlet Conditonの
Temperature = 298
Apply to boundaries:のBoundary1にチェックを入れる。図のように選択した境界が色がかわり選択される。
Name：Bottomにする
OKを押す



Boundary condition一Add
Heat Equationのタブを選択
Heat Flux condtion
Heat Fluxに1640000を入力
Apply to boundaries:のBoundary13にチェック
Name:TOPと記載する
OKを押す

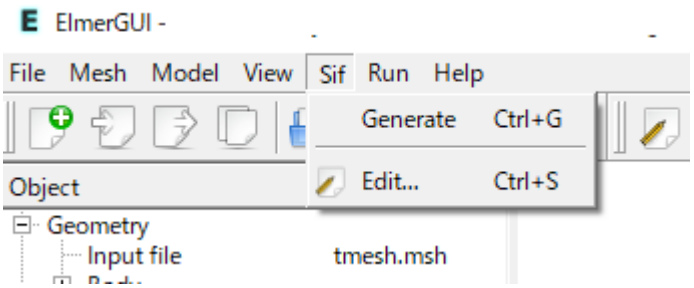
これで底面は境界条件として常に20度に設定される。
（この温度は相対的なため上昇温度だけを知りたい場合はゼロでもかまわない。）
続いてヒータ上面部分に熱流入Fluxを設定する。1640000という熱流入を入れているがこの値について後ほど解説を行う。



境界条件を有効にするために、

Set Boundary conditionのボタンを押して反転させる。
（またはModel- Set Boundary conditionを選択する）

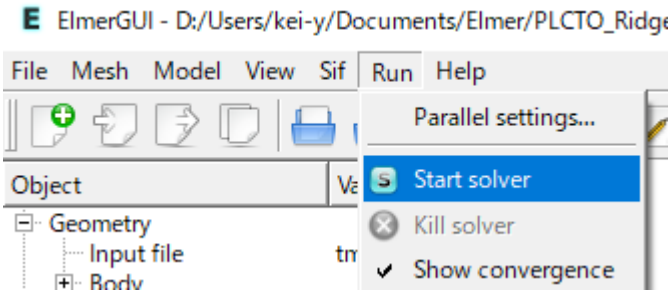
Sifファイルの生成



Sif- Generate
続いてSif- Editを選択すると生成されたSifが確認できる。必要があれば編集する。（編集する事はほとんどない。）

File- Save project
プロジェクト自体を上書き保存する。これをしないとFEMの計算に反映されないので注意。

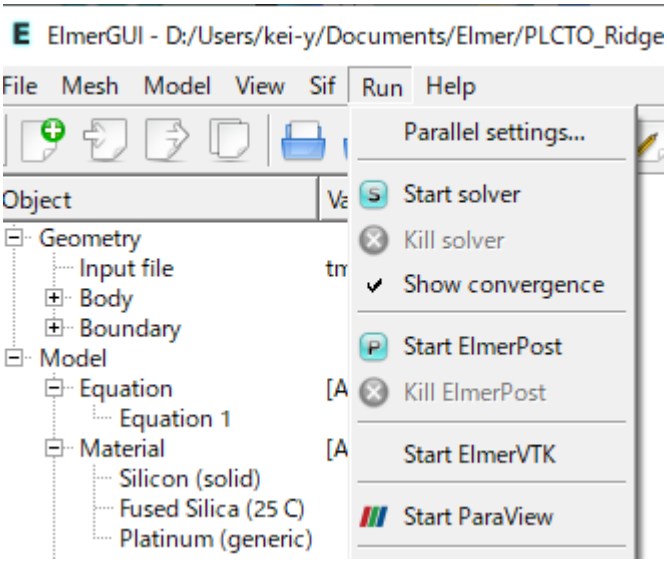
実行
Run - Start solver



計算が完了すれば、
ElmerSolver: The end
SOLVER TOTAL TIME(CPU,REAL): 0.71 0.71
ELMER SOLVER FINISHED AT: 20**/**/** **:**:**

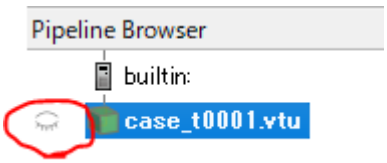
ELMER SOLVER FINISHED ATとエラーが無ければ計算終了時刻が表示される。

結果の表示

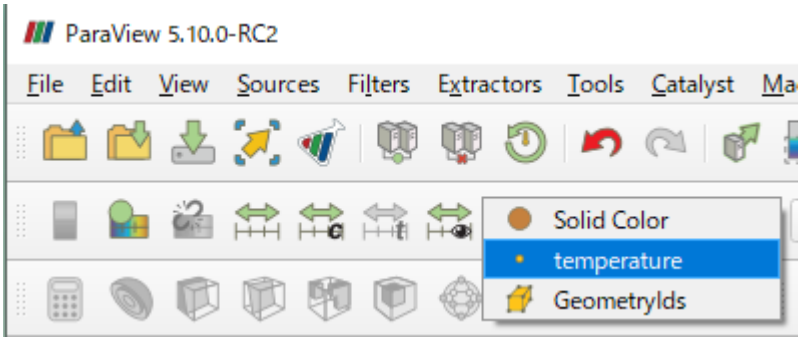


Run- Start ParaViewを選択すると、ParaViewが立ち上がり結果が表示される。

ParaViewの操作

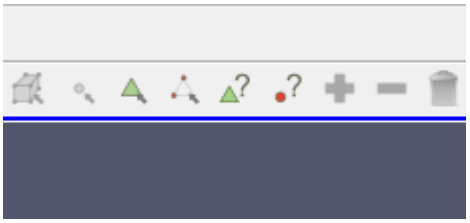


立ち上げ直後は、不可視になっているので、目のマークをクリックすると、構造が表示される。

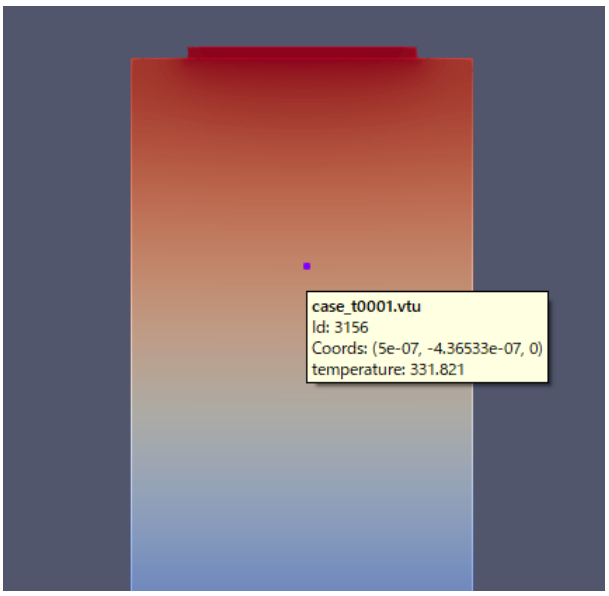


温度表示にしたいため、Temperatureを選択する。

温度分布が計算できている事が確認できる。



コア座標の温度を知るために、赤い丸印＋？のアイコンをクリックする。



コア中心がy軸ゼロになるようにCADで作図しているため、原点付近（(0, 0)はメッシュ交点に必ずしもならないので、近い交点）の温度を知る事ができる。
ライン上の分布を表示する等、ParaView自体の使い方はネット上の記事を参考にされたい。

この例の場合は、Si基板の底面が298度（摂氏20度）、熱量Qとして1640000を与えると331.18度と計算されて38.5度上昇している事になる。

流入熱量Q

上記ではHeat Fluxとして1,850,000をヒータの上面に熱流入させる計算を行っている。
Heat Fluxについて考察を行う。

Heat Fluxのため単位面積あたりの流入熱量として定義される。Heat fluxの単位はW/m2で与えられるとElmer Discussion Forumでは記述がある。3次元モデルの際は、単位面積あたりの熱量という定義で理解しやすい。2次元の場合も同じであり、

Heat flux＝　P[mW] / S [m2]

(dxはヒータ横幅の単位長さ、dzはヒータ長さの単位長さ)
考えるべきは、2次元のモデルの時で次元が一つ減少した場合になるが、
本計算において、与えるべきHeat Flux Fは、幅W長さLのヒータに電力Pを加えた場合
F= P[mW] / W/L
で与えられ、Elmerの中では、これにWを掛けた熱量が2次元の場合与えられる。つまり仮に20umのヒータ幅に熱量QのFluxを与えた場合と、10umで同じだけ与えるには、Fluxとしては倍の値を与えると、結果的にヒータ全体で与えられる熱量は同じになる。

Power	80	mW
Power （P)	8.00E-02	W
Lheater（L)	2.00E-03	m

Wheater[um]	Area of heater[m2]	Flux
ヒータ幅（W)	ヒータ面積（S)	単位あたり（P/S）
5	1.00E-08	8,000,000
10	2.00E-08	4,000,000
15	3.00E-08	2,666,667
20	4.00E-08	2,000,000
25	5.00E-08	1,600,000
30	6.00E-08	1,333,333

実験結果との差の検証

PLCからなるMZI型の光スイッチを考える。

長さLのヒータの下で、半波長の光路長差を発生させるには、どれぐらいコアの温度を温める必要があるかを計算する。

ガラスの屈折率温度依存性　dn/dTとし、位相変調器長さをLとする。ヒータによりΔTの温度差がつけられるとする。
ガラスの屈折率の変化量Δnは

$$\Delta n = \frac{\delta n}{\delta T} \cdot \Delta T$$

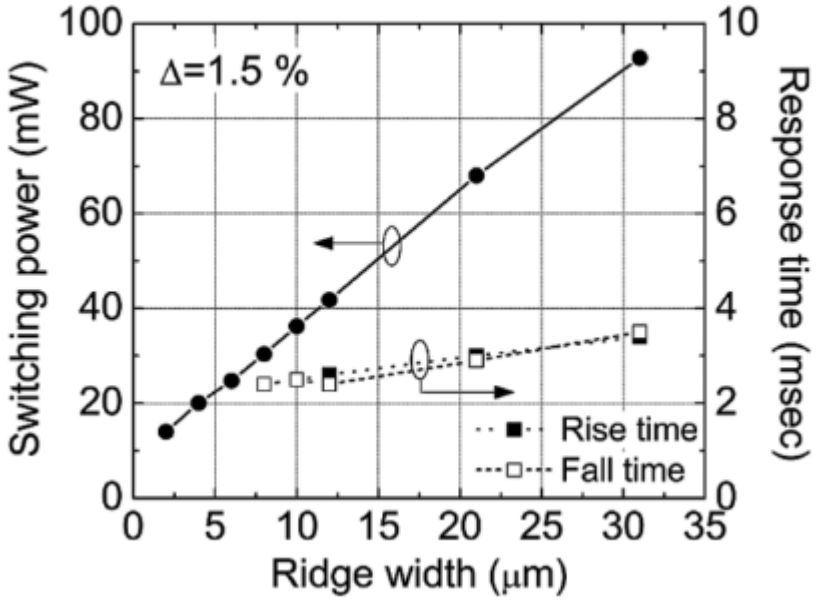
で与えられる。長さLの位相変調器で半波長の光路長差を発生させた場合以下の式が成り立つ。

$$\frac{\lambda}{2neff} = \Delta n \cdot L = \frac{\delta n}{\delta T} \cdot \Delta T \cdot L$$

λ＝1.55x10-6、neff=1.46(導波路の実効屈折率）、dn/dT＝8x10-6、L＝2x10-3とし上記の式をΔTについて解くと、ΔT＝33.2で与えられる。つまりヒータ直下でコアをMZIの他方のコアより33.18度温め、2mmにわたり伝搬させればスイッチとしてオンからオフになることになる（この時の電力はパイ電力と呼ばれる）。周辺温度を298[K]とした場合、331.18[K]になる事を意味する。

先のシュミレーションの結果Fとして、1640000（65.6mW)を与えた場合、コア中心温度は331.18[K]となる。（そうなるようにFの値を入れていたため）

つまり計算上はリッジ幅65.6mWとなる。だが実際の実験結果ではリッジ幅30umの際には同構造で90mWであり、実際の計算よりも


Δ=1.5 %
Switching power (mW)
Response time (msec)
Ridge width (μm)
Rise time
Fall time
0
20
40
60
80
100
0
2
4
6
8
10
0
5
10
15
20
25
30
35