Paraview 覚書

16 June, 2025 revised

Paraview はフリーの可視化ソフト．linux 対応であるが，ここでは，windows 版を推奨する．

1. インストール

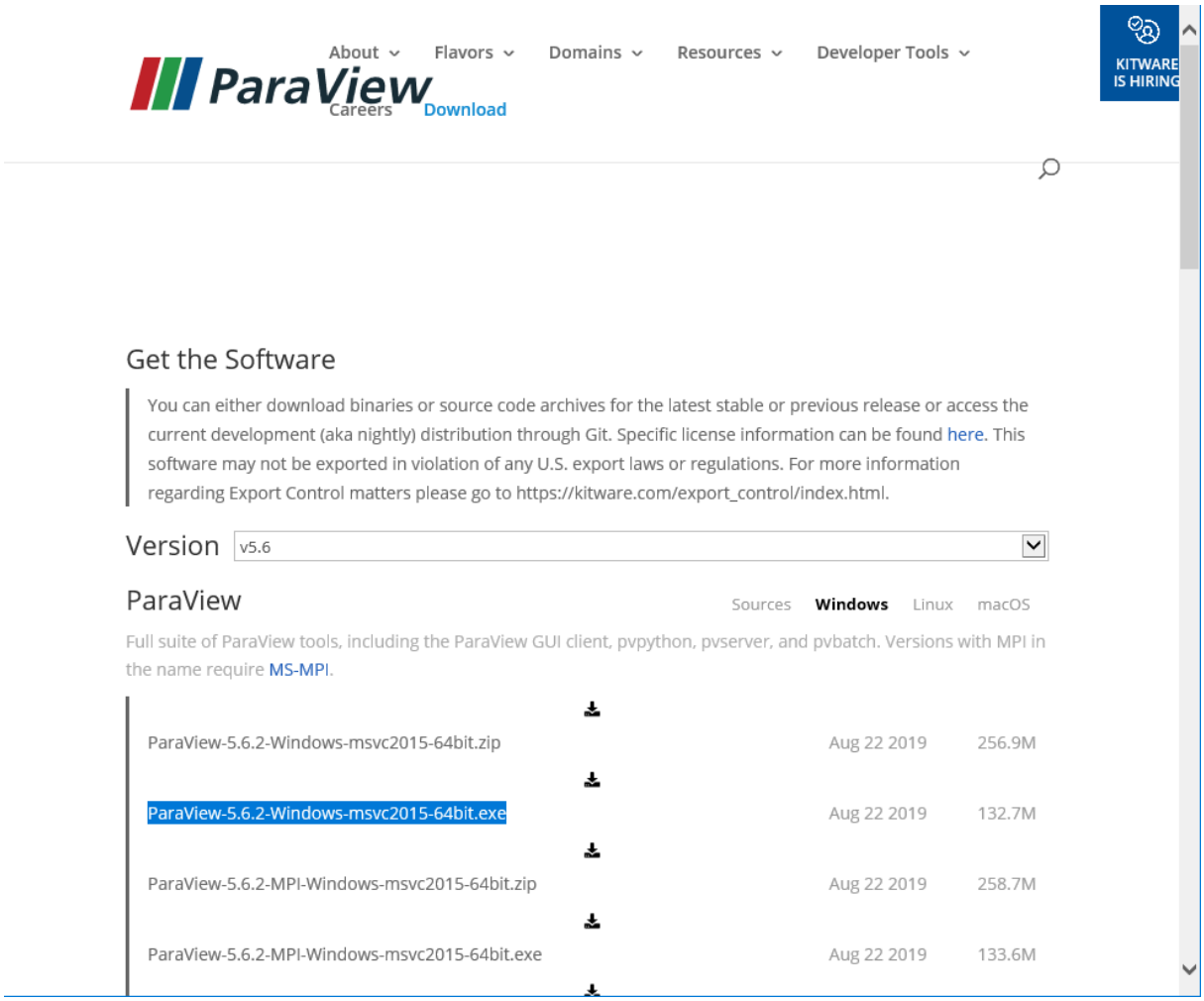
以下からダウンロードできる．

https://www.paraview.org/download/

2021 年現在最新は，v5.9.*であるが，多くは動かない．動作確認できたのは，v5.6.2 であるので，上記から，Version で **v5.6** を選択し，選択画面から，

ParaView-5.6.2-Windows-msvc2015-64bit.exe

をダウンロードして，windows 上でインストールする．



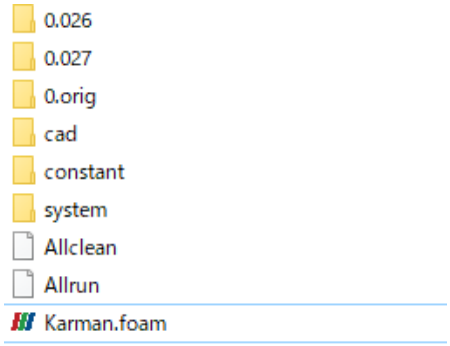
なお，paraview は，VCOMP140.DLL を呼び出すが，これが無い場合には，例えば，この辺からダウンロードしてインストールしておく．

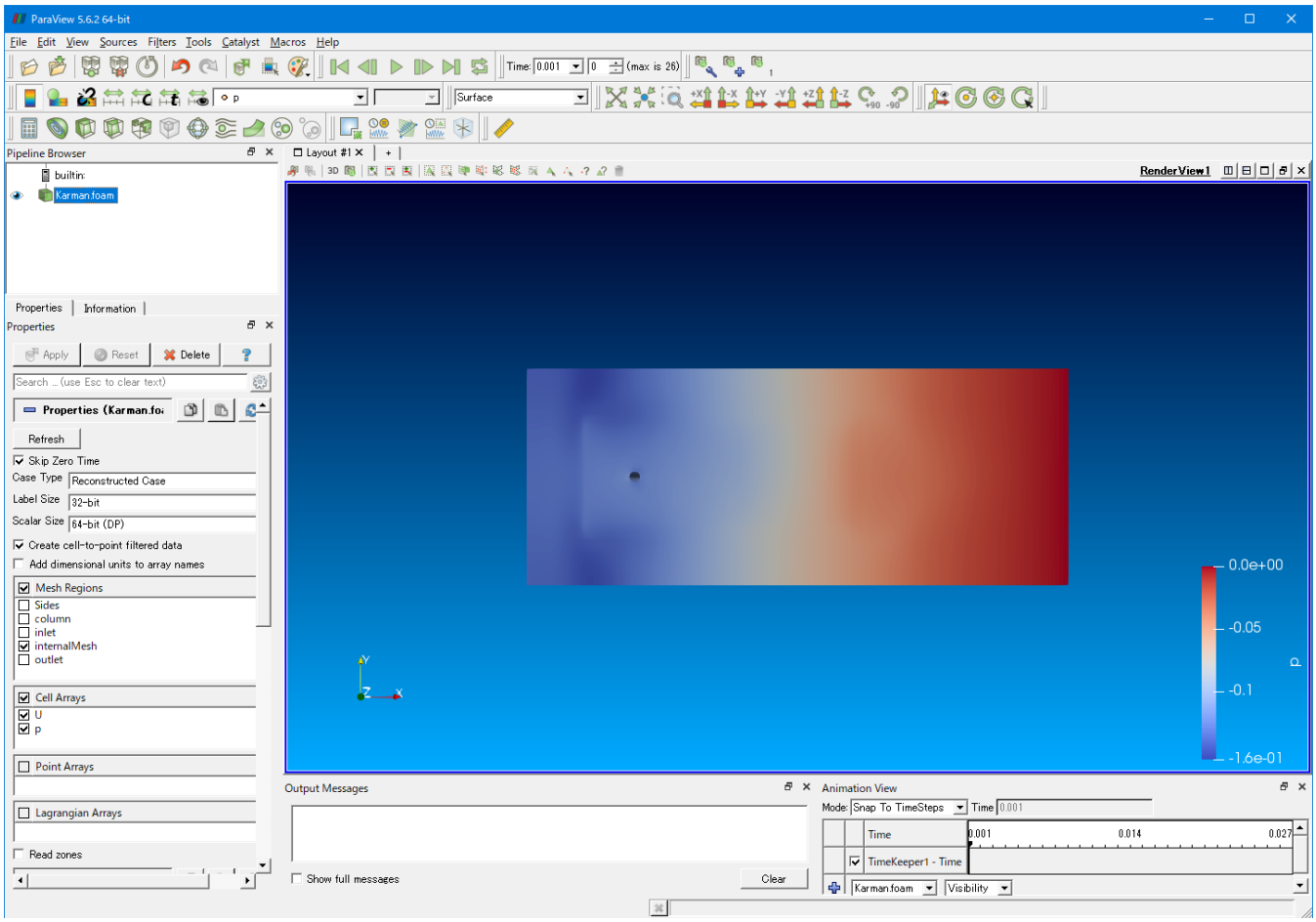
https://www.microsoft.com/ja-jp/download/details.aspx?id=48145

注：2023 現在では，gltf フォーマットを認識できるという理由から，ver. 5.11.0 を使っている．

2.　データの読み込み

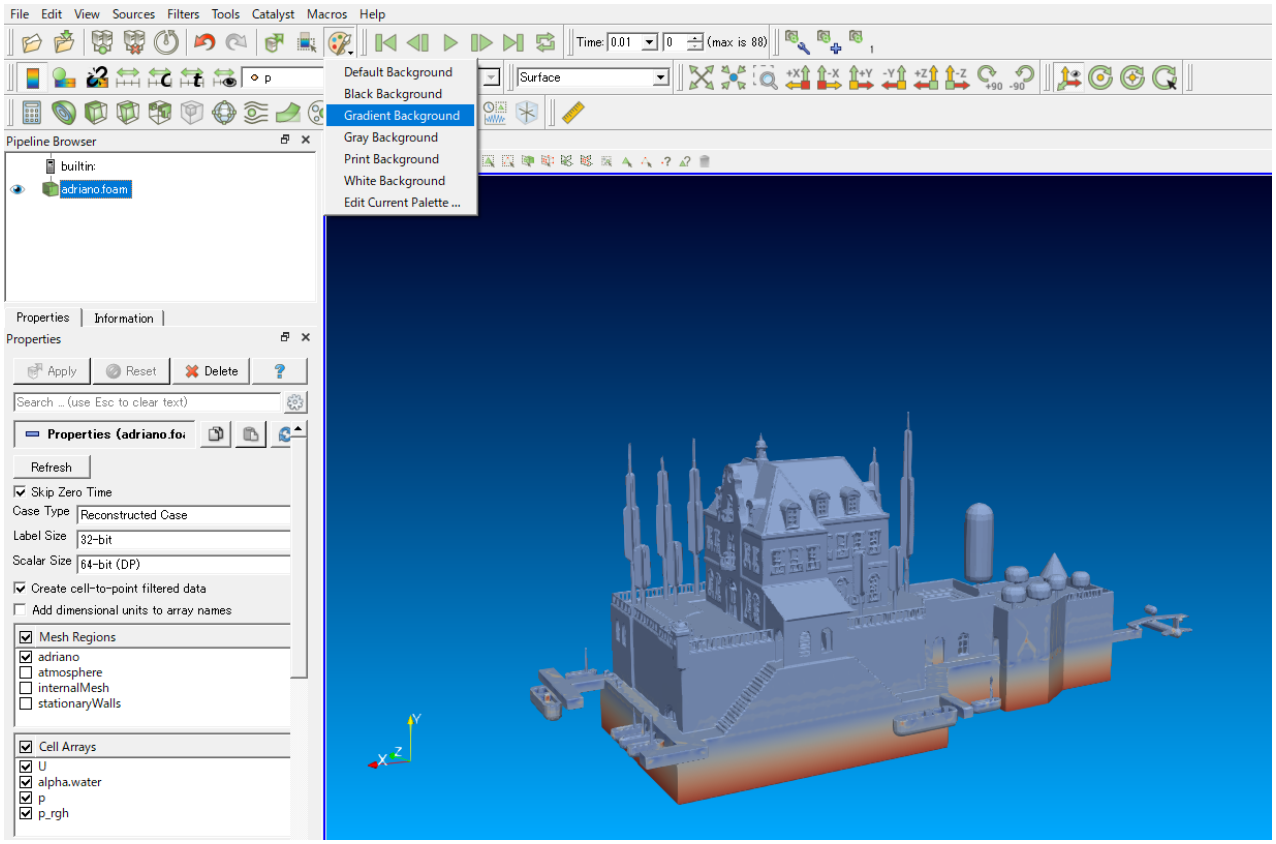
拡張子が".foam"をダブルクリックして，同じディレクトリにあるデータファイルを読み込み「Apply」する．***.foam は，中身は空のテキストファイルでよい．以下は，0.026 ms のデータを読み込んでいる．

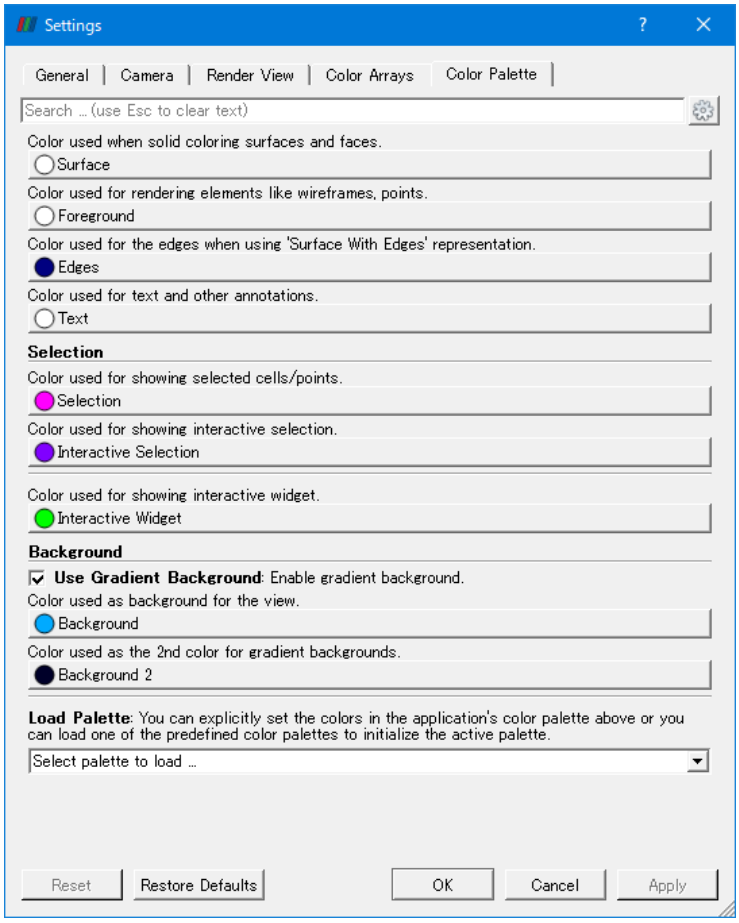




3.　背景の変更

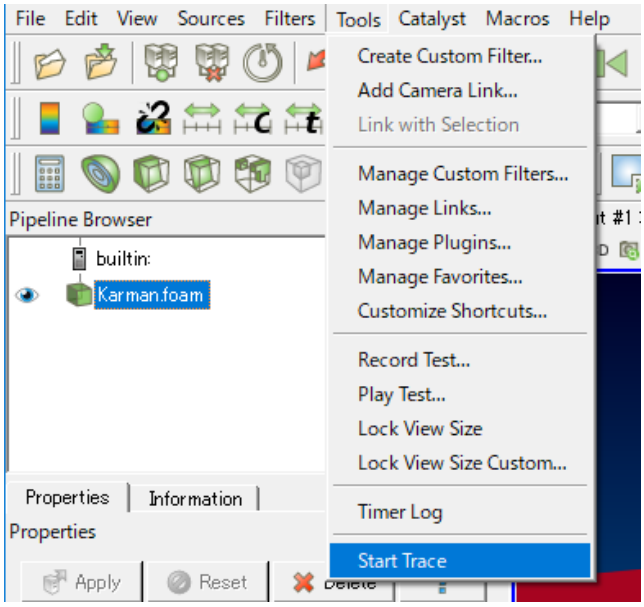
以下のコマンドで背景を変更することができる．「Edit current palette」で二色の配色のグラデーションができる．

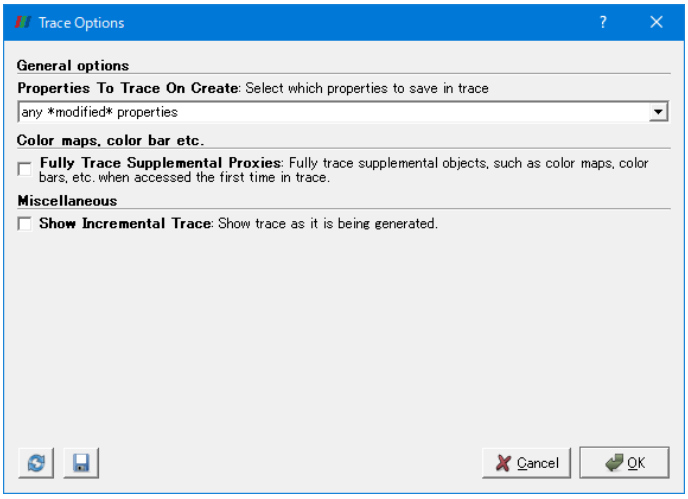




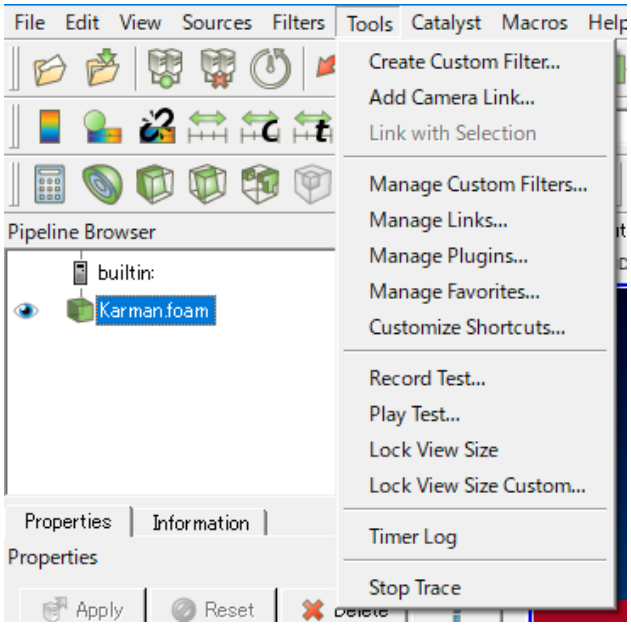
4. マクロ

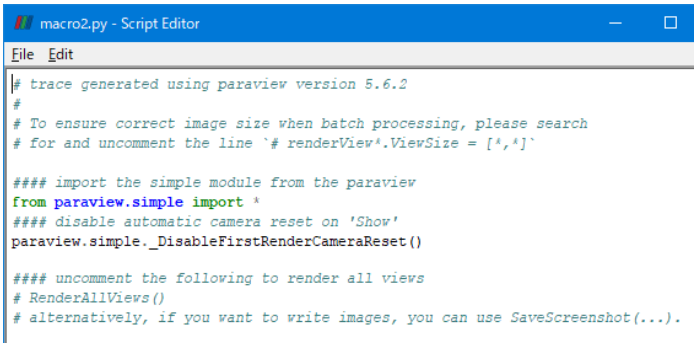
Tool の Start Trace でマクロの記録を開始し，Option を決めて OK する．

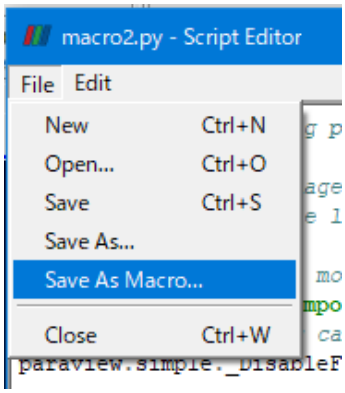




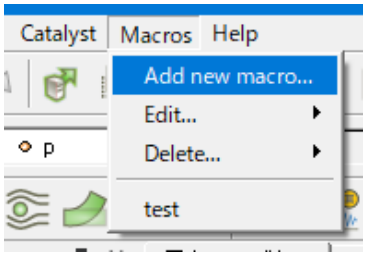
一連の動作が終わったら，Stop Trace すると，ファイルが表示されるので，適当に名前を付けて，Macro ファイル（***.py）として保存する．







Macros の Add new macro から保存したファイルを登録する．test というマクロファイル名にしたので，test というマクロボタンが出来ている．ボタンを押すとマクロが実行される．



残念ながら，カメラのオペレーションは登録できないようだ....

5. 渦度の表示

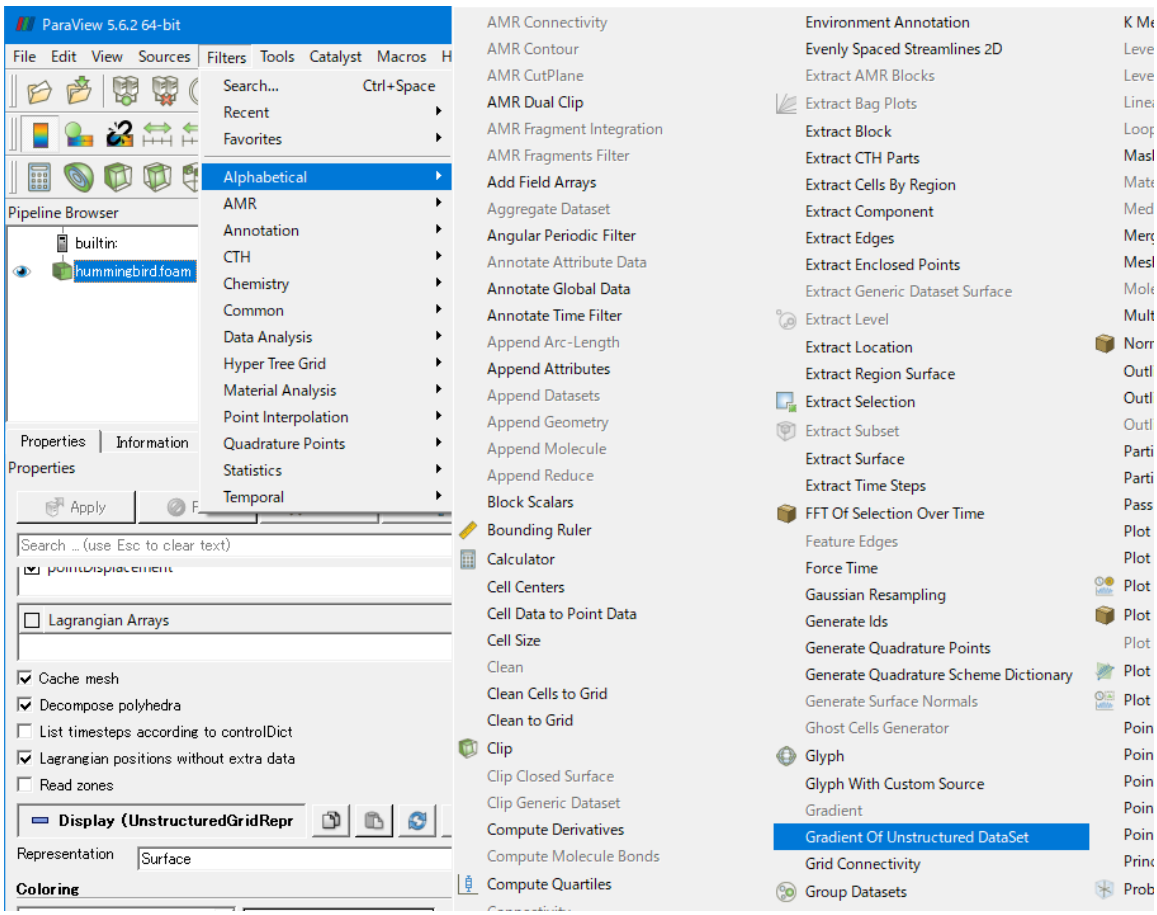
渦度ベクトルは，

$$\omega = \nabla \times U = \begin{pmatrix} \frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} \\ \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \\ \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \end{pmatrix}$$

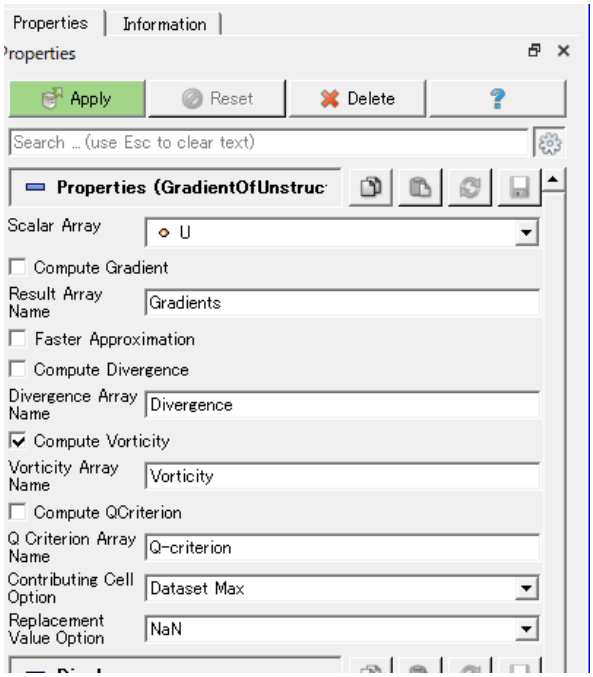
で計算され，U が計算結果の速度ベクトル．U の回転を計算して，そのノルムをスカラー量として表示する．

．手法 A：

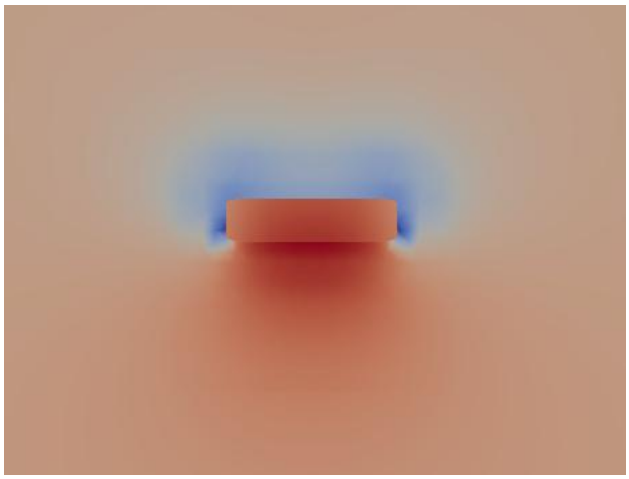
「Filters」->「Alphabetical」->「Gradient Of Unstructured DataSet」を選択する（回転を計算したいのに，勾配を選択するのが引っかかる...）．



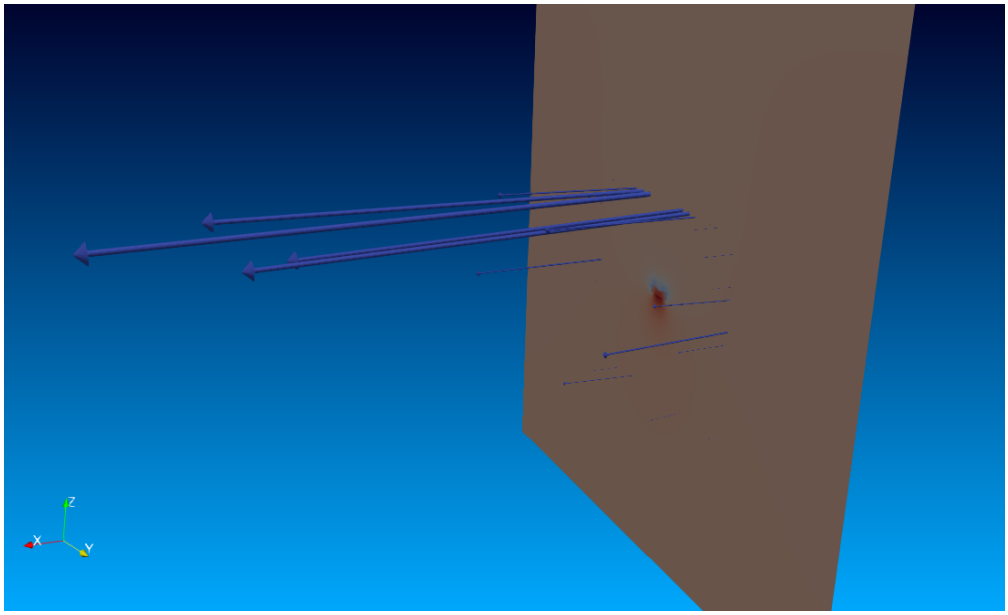
．「Scalar Array」で流速「U」を選択し，「Compute Vorticity」を選択し，「Apply」する（速度ベクトル U なのに，Scalar Array と書かれているのも気になる）．



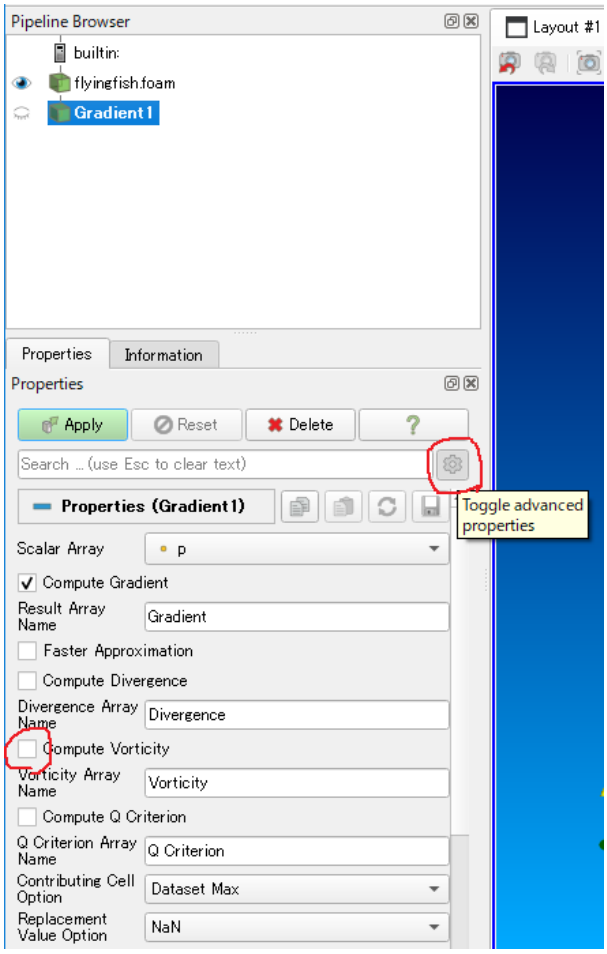
．Coloring に「Vorticity」が現れるので，選択して，適当な平面を表示すると，渦度が表示される．



．Glyph を使ってベクトル表示するとこんな感じ．ちょうどよい矢印の長さと位置を決めるのが難しい...



．ver. 5.10 以降は，「Gradient Of Unstructured DataSet」は，「Gradient」に統合されている．
Properties の歯車のマークをクリックすると，これまでと同じことができる．



なお，格子の細分化の状態により，やたらノイズのような値が出るケースがある．細分化のレベルを上げているところなど．

．手法 B：

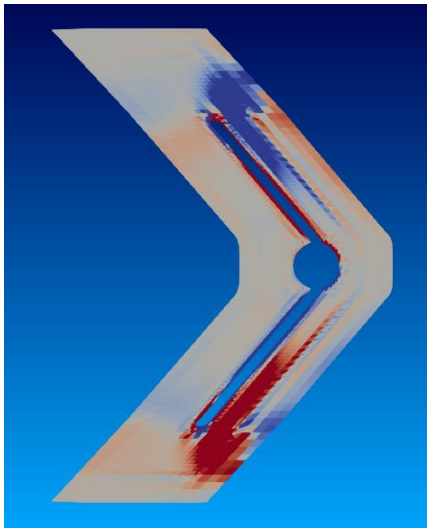
導関数計算（Compute derivatives）を使って同様に渦度を表示することができる．

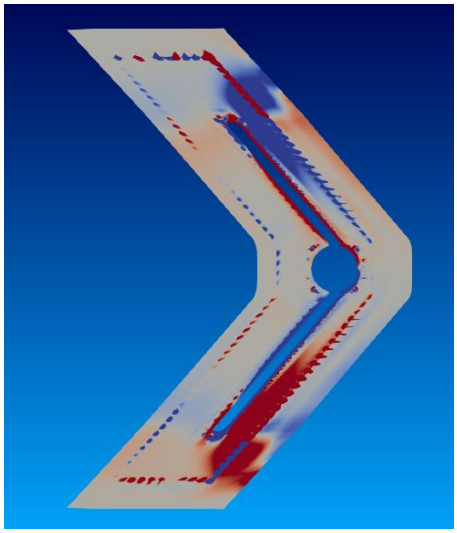
「Filters」->「Alphabetical」->「Compute derivatives」から，

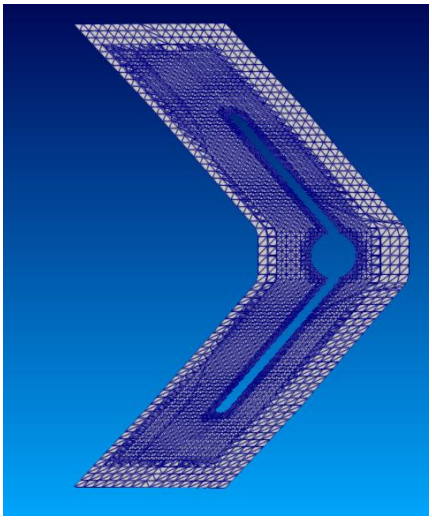
Vectors 「U」

Output Vector Type 「Vorticity」

を選択すると，Coloring で「Vorticity」を選択できるようになる．方法 A は，細分化 level を変更したところにノイズが出ていることが分かる．



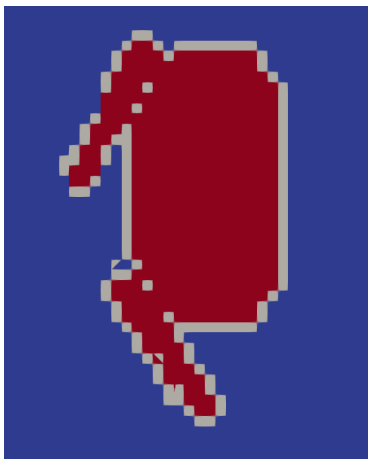




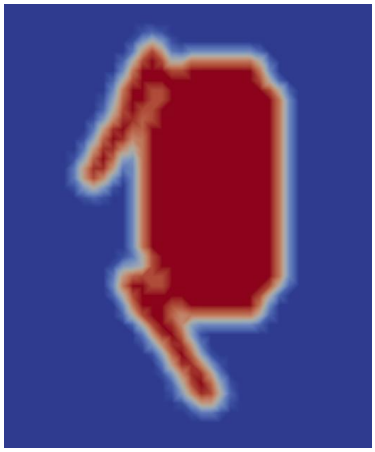
(a) Compute derivatives　　(b) Gradient Of Unconstructed Dataset

6. 物理量のセル間グラデーション表示と非表示

有限体積法の場合，セルの中心に物理量が保存されているが，色（coloring）表示の時，  を選択すると，セル内が単一の色で着色され，



のように表示され，  を選択すると，セル内でグラデーションがかかり（色が補間され），



のように表示される．

7.　重合格子のデータと計算空間（background）のデータの表示

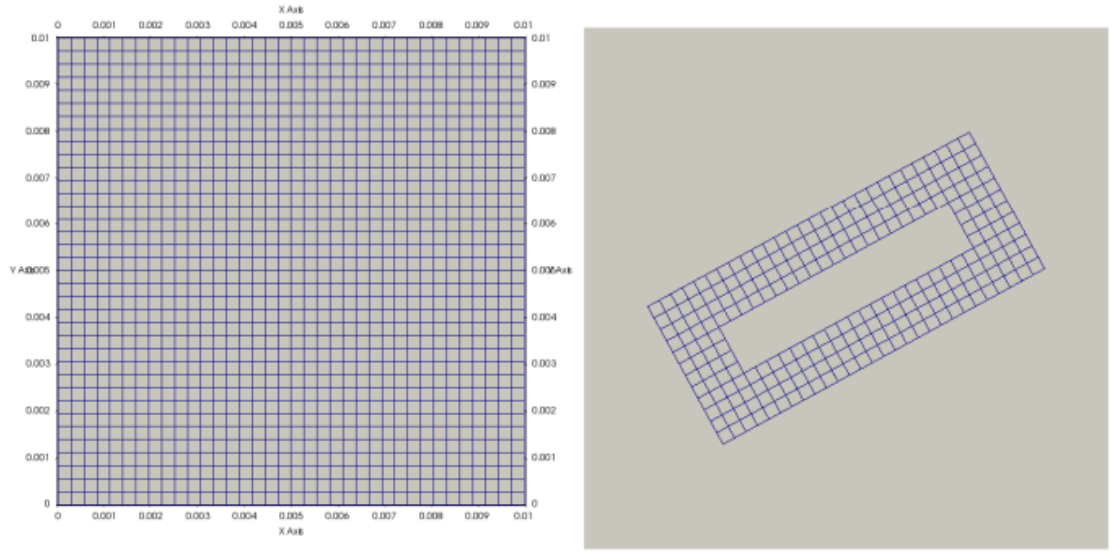
重合格子と計算空間のデータを別々に表示したいときがある．重合格子の計算では，セル領域を以下の三つに分けている．tutorial の simpleRotor で説明すると，以下の図のような計算空間の領域（background：図左）と重合格子の領域（overset：右）は，

計算領域（Calculated）：　　運動方程式を解く領域

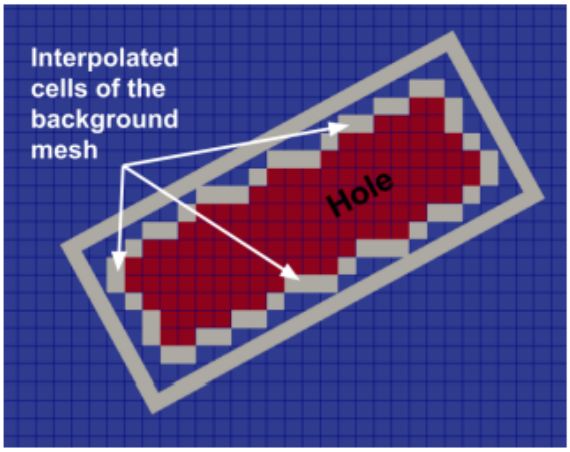
内挿領域（Interpolated）：他方の領域（overset なら background，background なら overset）の要素の近傍から内挿によって求められる領域．

空洞領域（Holes）：ここでは計算しない領域．おそらく，以前の値がそのまま残っている．

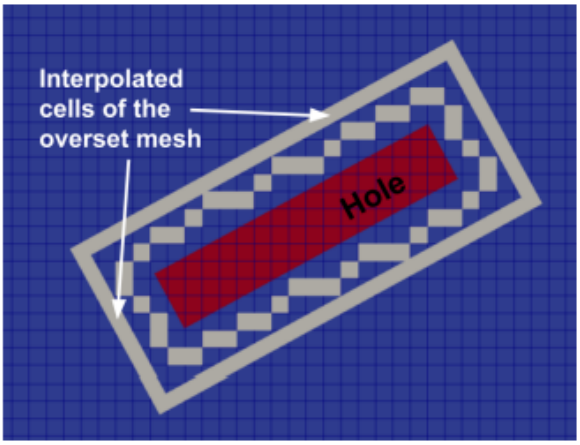
となっている．



(a) Background mesh　　(b) Overset mesh

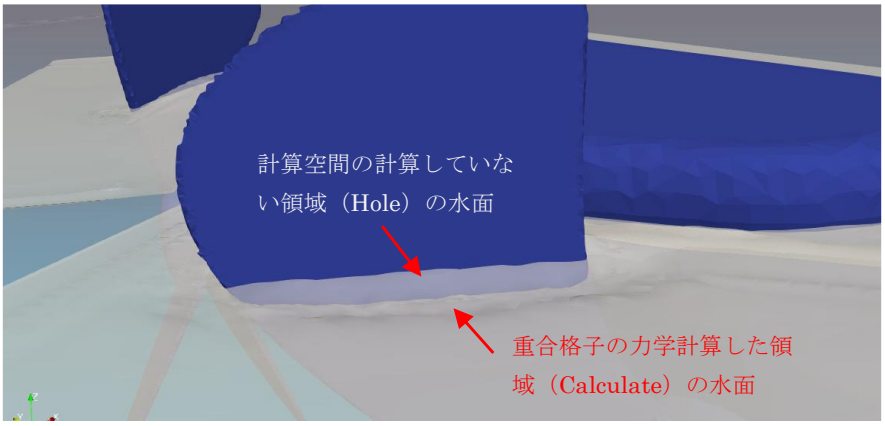


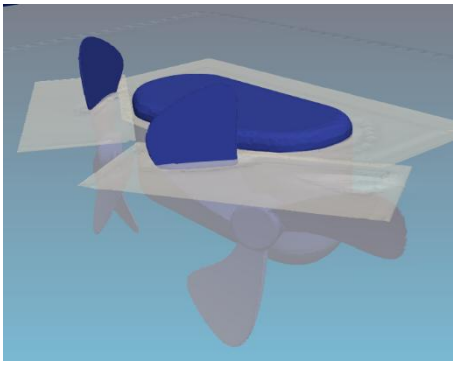
(a) Hole (red cells) and interpolated cells (white cells) defined for background mesh



(b) Hole (red cells) and interpolated cells (white cells) defined for overset mesh

例えば, 水中ロボでこの水面をそのまま表示すると, 以下のようになり, 計算していない水面まで二重に表示されることになる.





これは，上記の計算領域と内挿領域のみを表示することで回避することができる.

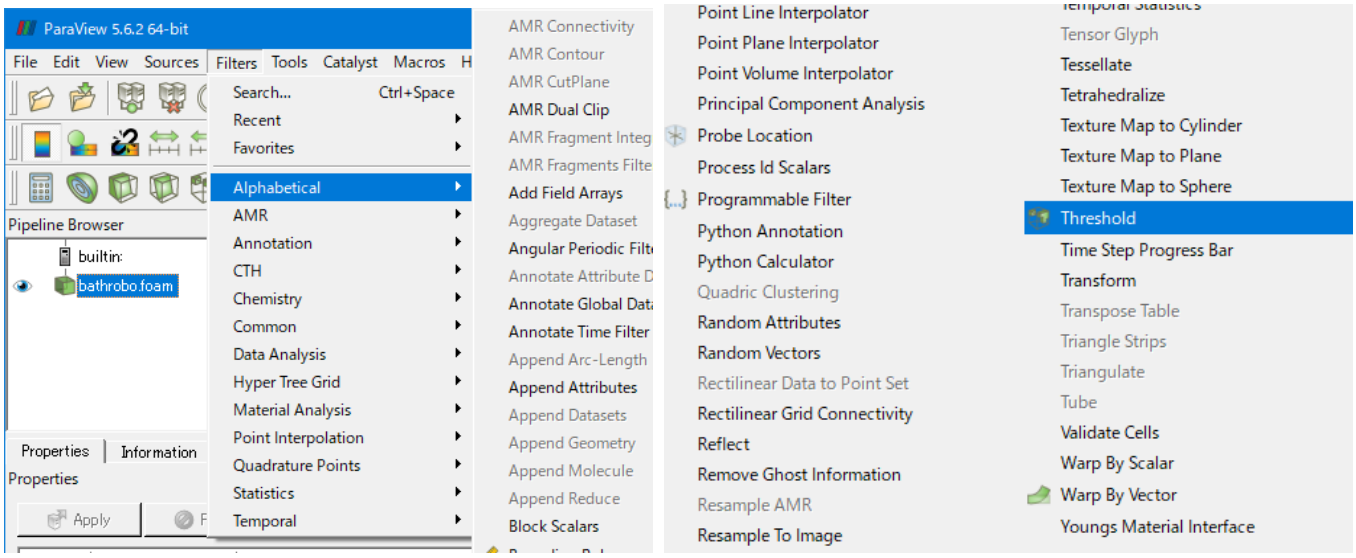
「Filters」の「Alphabetical」の「Threshold」を選択し，Scalars で「cellTypes」を選択し，min を「0」，max を「1.5」にして「Apply」する．これにより，cellType が 0 と 1 の物理量のみ表示されるようになる（2 がマスクされる）．なお,

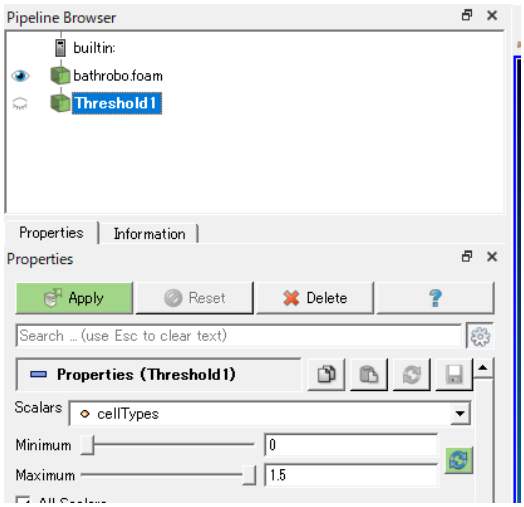
計算領域が 0

内挿領域が 1

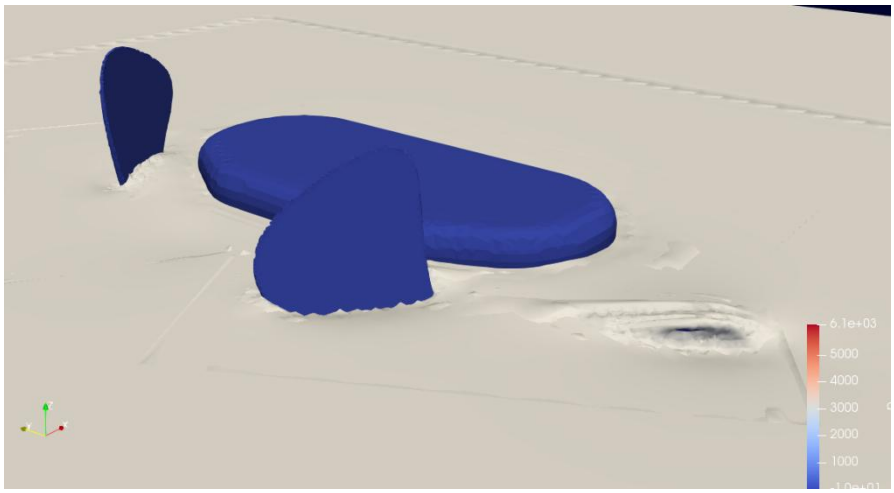
空洞領域が 2

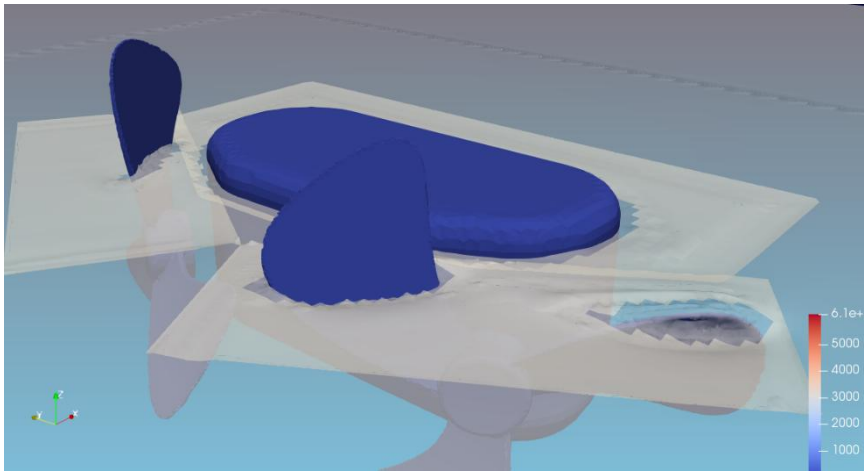
である.





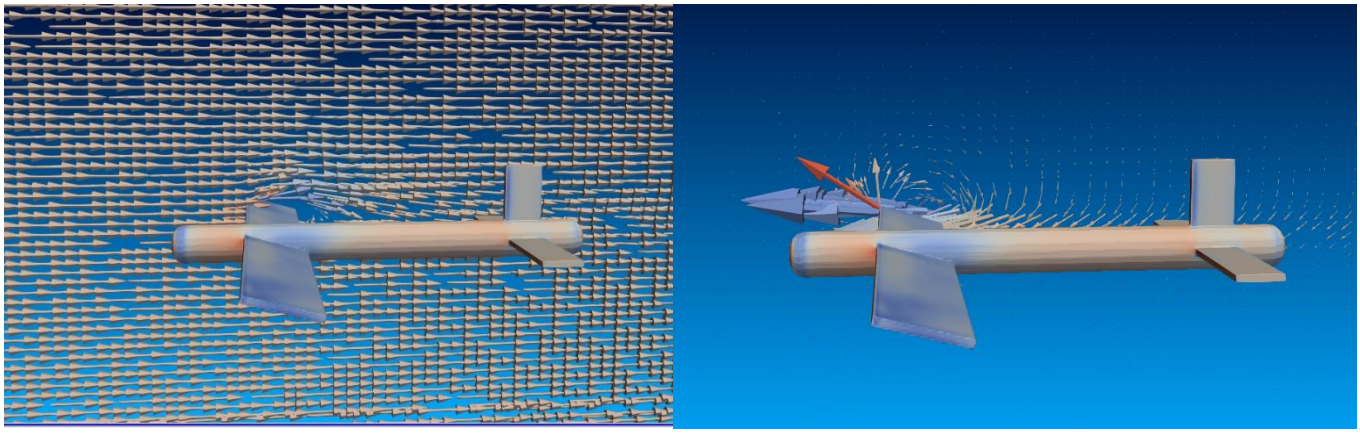
次に，「Contour」で「alpha.water」を選択し，値を「0.5」にして「Apply」すると，空洞領域を排除して水面を表示できる．ただし，透過表示すると計算領域と重合格子領域の物理量が二重に表示されて濃くなるのは回避できていない．





8.　速度の表示あれこれ（Calculator の使い方）

航空機の翼周りの流れを定常計算した場合の流速ベクトル（動いている座標系から見た流速ベクトル）を表示すると以下のようになるが，流入速度 vx=10m/s が強く，翼周りの流れの変化が分かり難い場合がある．この場合は，絶対座標（止まっている座標系）から見た速度ベクトルを表示すると分かりやすい場合がある．



航空機から見た翼周りの流速ベクトル（左）と地上から見た翼周りの流速ベクトル（右）

．表示法

データを読み込んだ後，「Filters」の「Alphabetical」から「Calculator」を選択する．

「Result Array Name」で新しく作る変数の名前を記入する．ここでは，「RelativeVelocity」というベクトル変数を作る．

計算式のスペースに，Scalars のプルダウンメニューを利用しながら

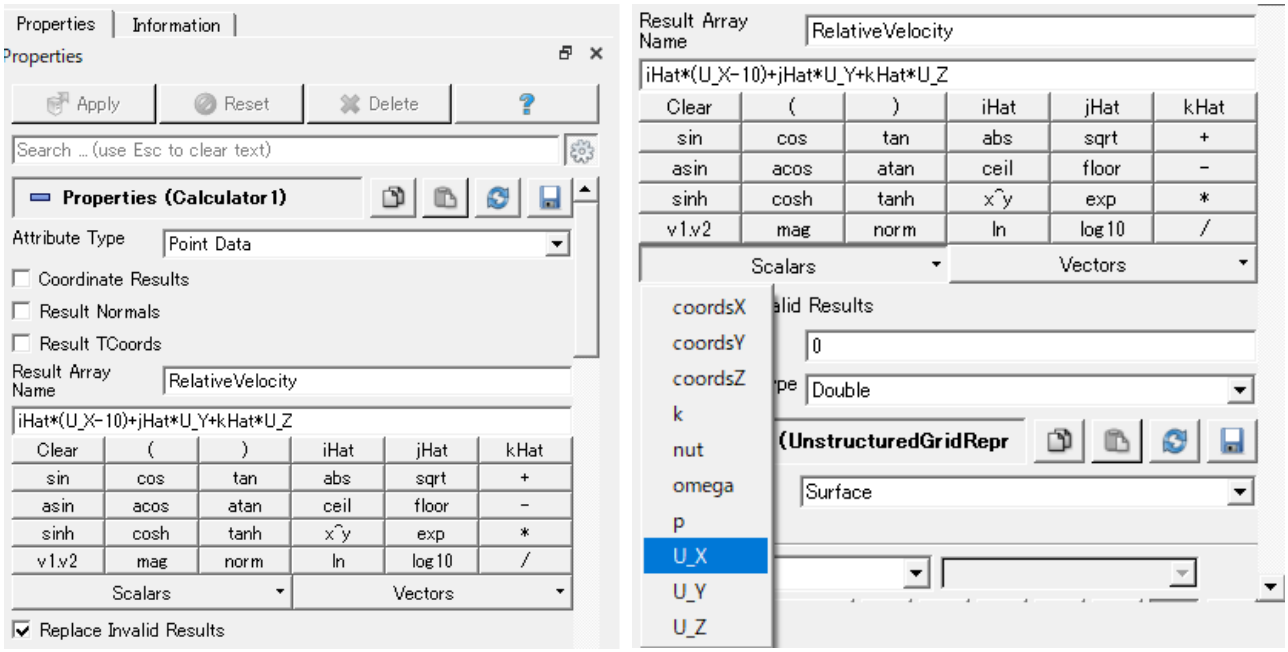
iHat*(U_X-10)+ jHat*U_Y+ kHat*U_Z

と入力する．

$$\mathrm{Re}\,lativeVelocity = \hat{i} * (U_x - 10) + \hat{j} * U_y + \hat{k} * U_z$$

$$where\ \ U = (U_x, U_y, U_z), \hat{i} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \hat{j} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \hat{k} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

という意味である．全ての流速の x 成分が 10m/s 引かれている．



これ以降は，「Glyph」で新しくつくって「RelativeVelocity」を表示する．

9.　データを間引く（background）

時系列データを 0.1ms 毎保存したが，簡易的に 1ms 毎で動画を作りたいなど，データを間引きたいときがある．この場合は，

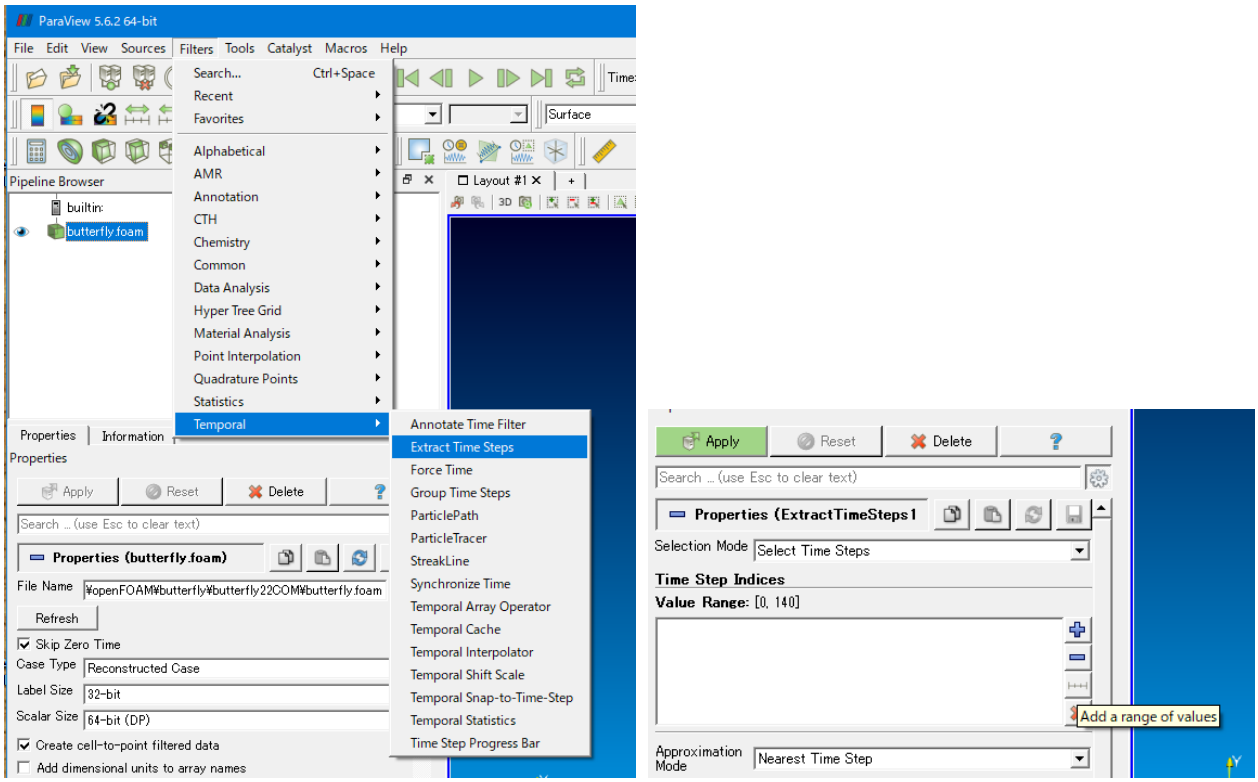
時系列データを読み込んだ後，

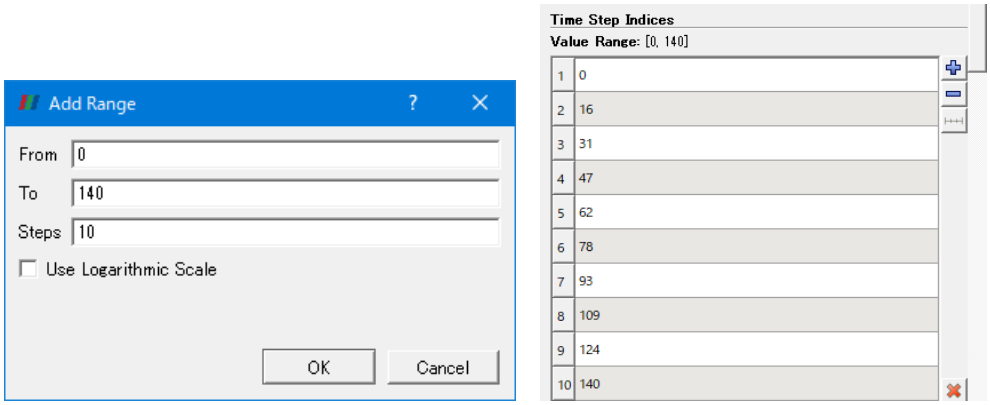
「Filters」->「Temporal」->「Extract Time Steps」

を選択し，

デフォルトの Value Range の設定を消し，「Add a range of values」のアイコンをクリックし，分割数を設定する．

ex. 0～140 まで 141 個のデータがあったとき，10 と入力すると間引かれて 10 個のデータになる．合わないデータは，「Nearest Time Step」だと近い時間の値になる．この後は，このデータに対して処理を行う．





10. 時系列データ間の差分を取りたい（Python calculator）

任意の時刻間のデータ処理（差分を取るとか）をしたい場合がある．例えば，加速度を求めたい場合，

$$\frac{U(t)-U(t-1)}{\Delta t}$$

という式が書ければ，後方差分により近似的に加速度を求められる．当然，前方差分も中央差分もできる．

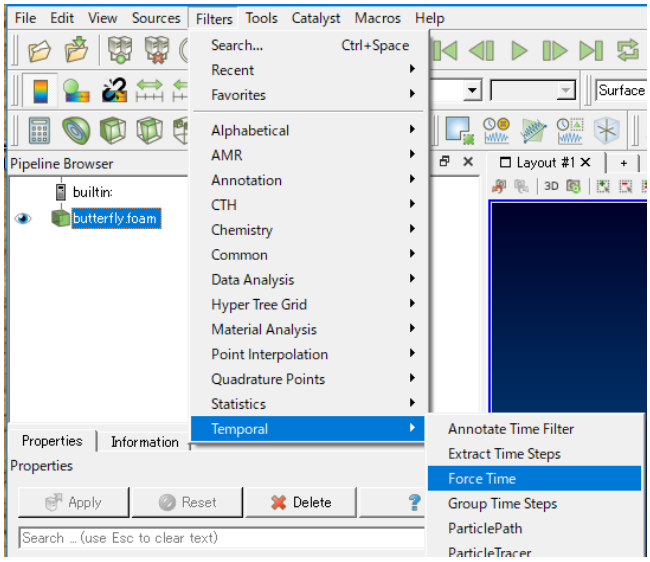
データを読み込んだ後，

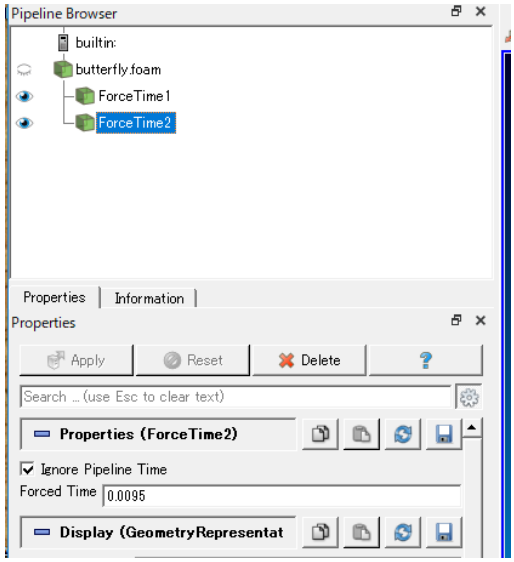
「Filters」->「Temporal」->「force Time」

を選択し，Apply する．0.01s の加速度を求めたい場合，Forced Time に「0.01」を入力して Apply する．同様に，

「Filters」->「Temporal」->「force Time」

を選択し，後退差分ということで，0.0095s を入力する．ここで，dt=0.0005 である．パイプラインのつながりに注意する．





次に，コントールキーを押しながら，「ForceTime1」と「ForceTime2」を同時に選択し，その状態で，

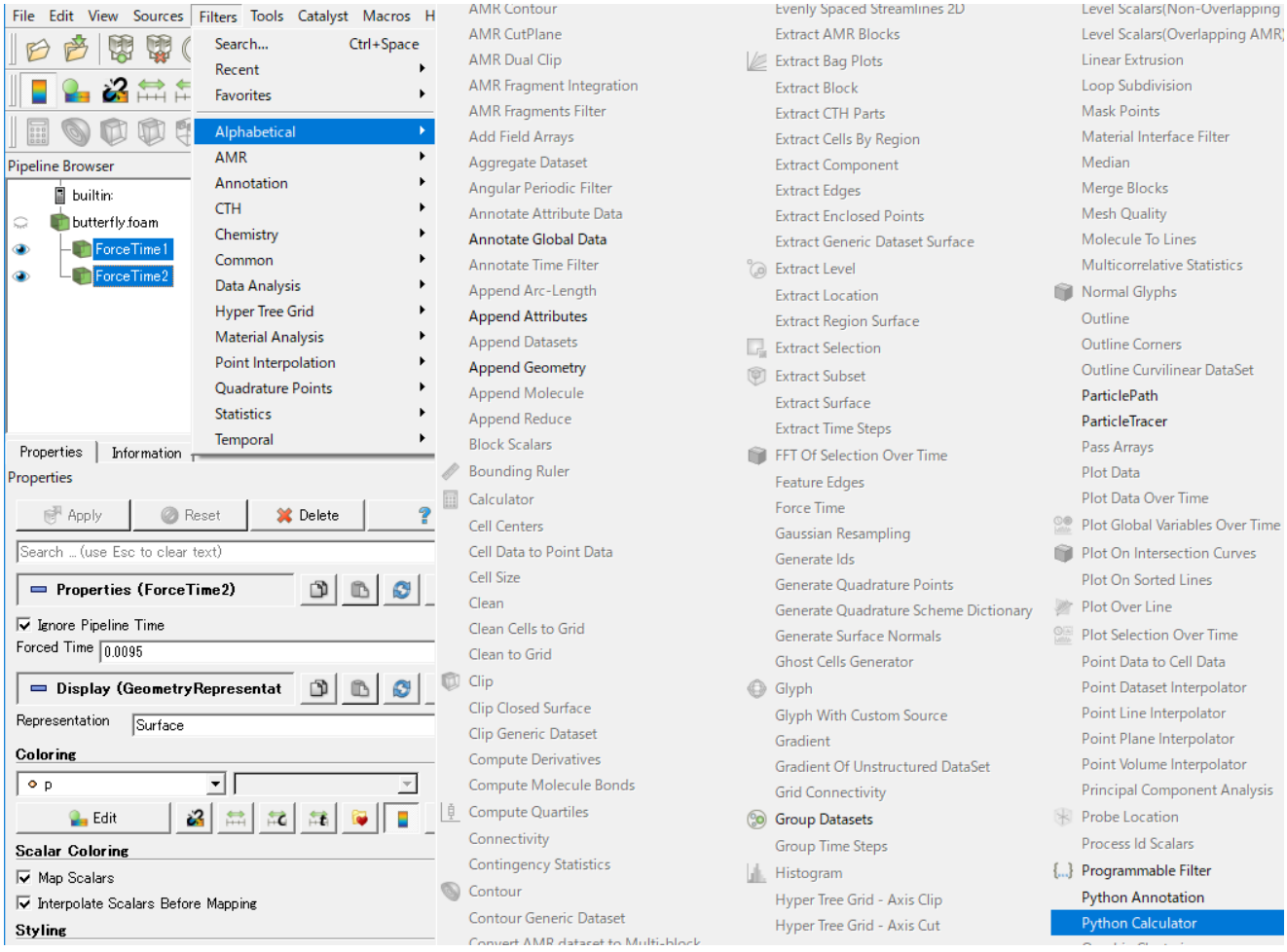
「Alphabetical」->「Python Calculator」

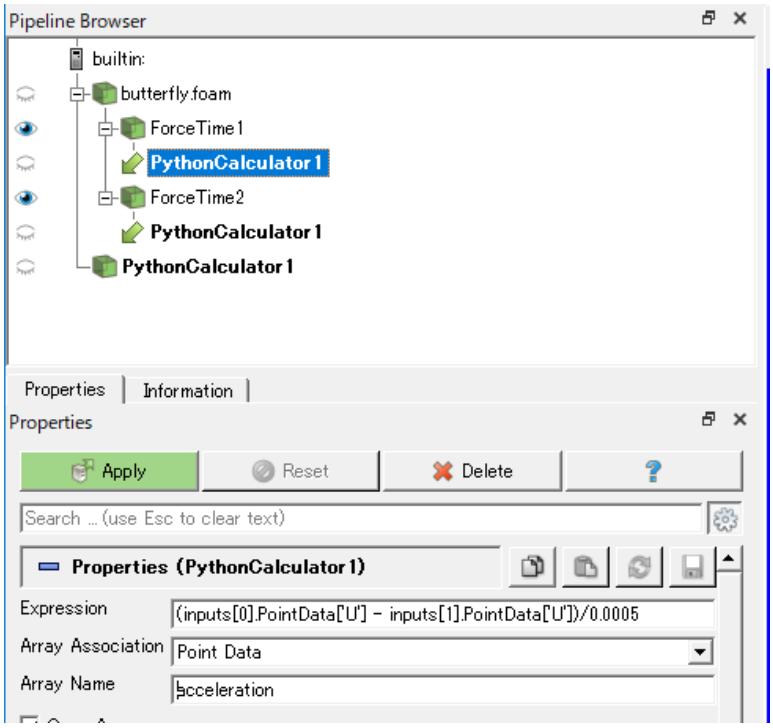
を選択する．「Expression」に加速度の式を入力する．

(inputs[0].PointData['U'] - inputs[1].PointData['U'])/0.0005

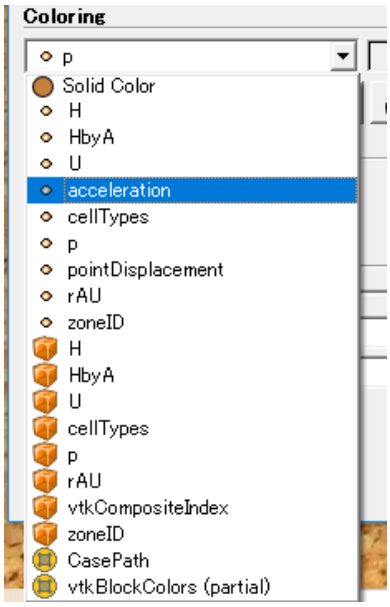
U は速度，0, 1 はデータ番号である．（ForceTime1=0, ForceTime2=1）

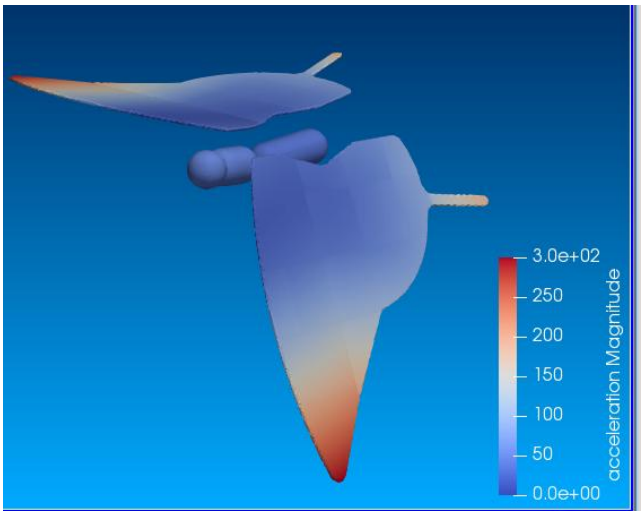
「Array Name」は任意の名前を入力する．ここでは，「Acceleration」．その後，Apply する．





カラーリングなどで，「Acceleration」を選択すると，加速度を表示することができる．加速度自体はベクトル量であるが，ここではスカラー量として大きさをカラーリングしている．翅先が 300m/s^2，およそ，30G 出ていることが分かる．流れ場の加速度を表示することもできる．



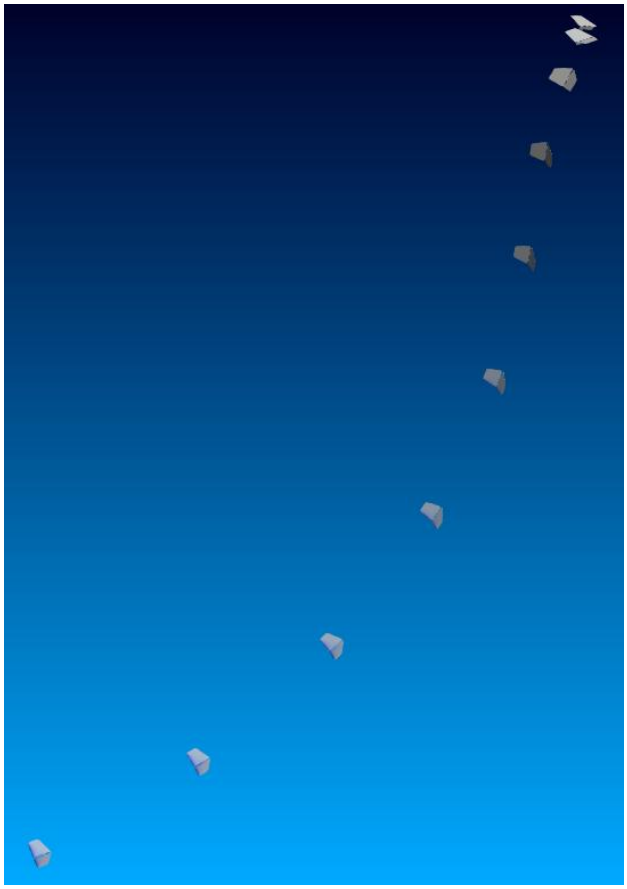


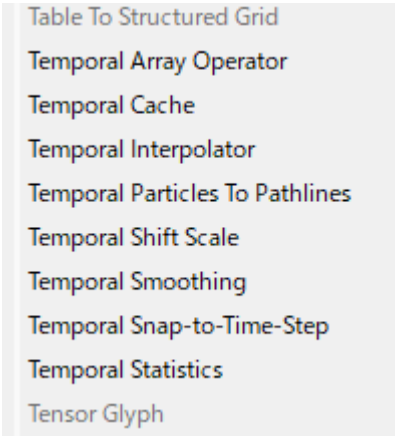
11. 時系列データを重ねて表示したい：ストロボ表示（ForceTime）

「Filters」->「Temporal」->「force Time」

で表示したい時間を入力し，同じことを繰り返すと，データを重ねて表示することができる．滑空のストロボ表示に便利．

注：Paraview5.13.2 では，「Temporal Array Operator」に変更されている．





12. Programmable filter の使い方

Filters にない処理は，programmable filter で直接プログラムすることによって実現できる．言語は Python.

. 「Filters」→「Alphabetical」→「Programmable filter」を選択

. Output Data Set Type で出力するデータタイプを選択.

. Script にプログラムを書く.

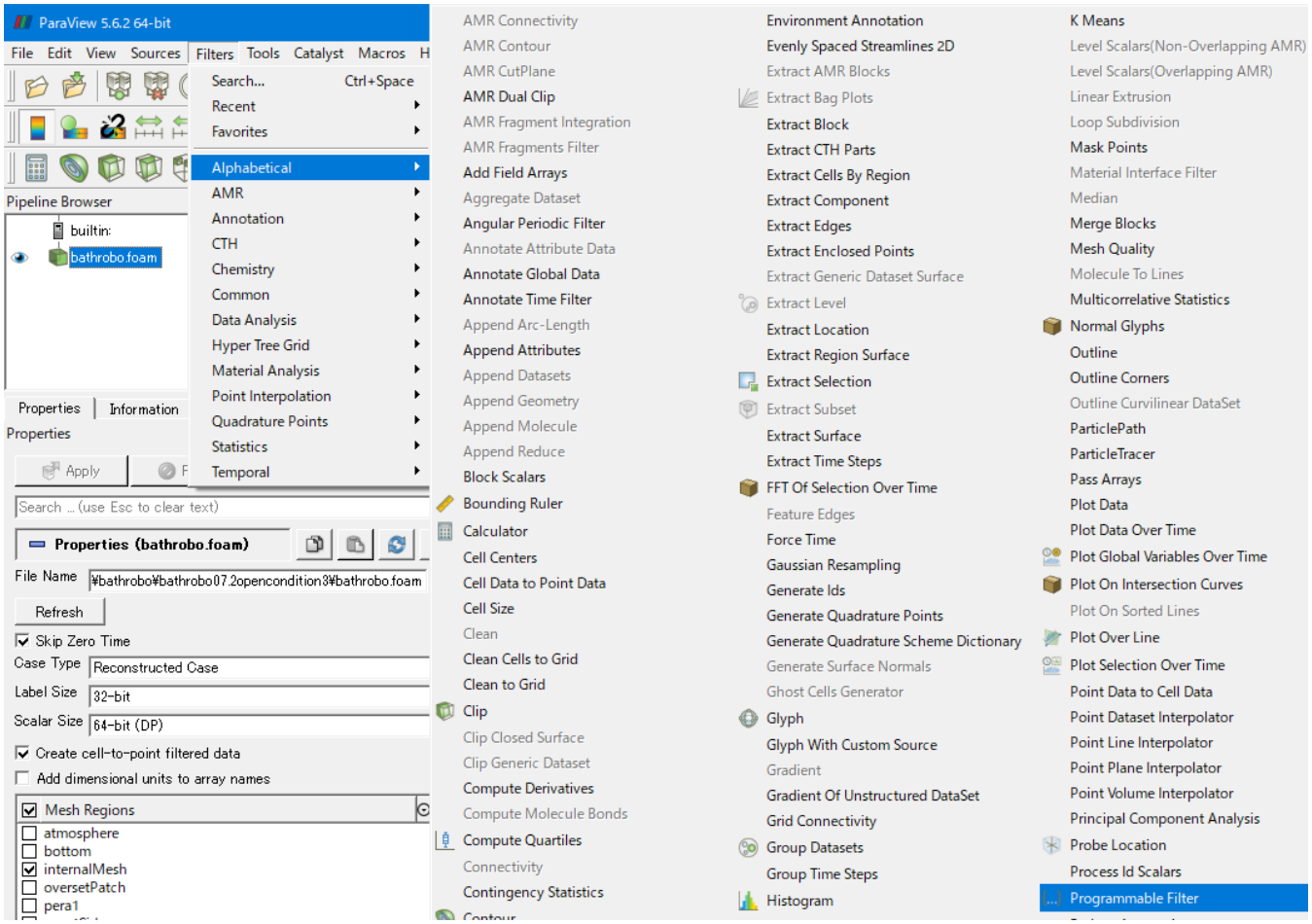
参考：

https://www.paraview.org/Wiki/Python_Programmable_Filter

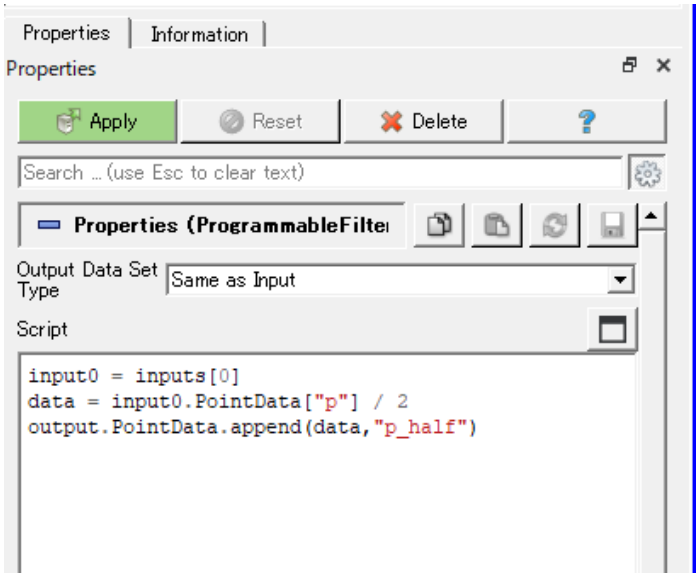
https://docs.paraview.org/en/latest/ReferenceManual/pythonProgrammableFilter.html

12.1 圧力が半分の p_half という Cell Arrays を作る

例えば，気液二相流の風呂ロボの計算データを読み，「Filters」の「Alphabetical」から「Programmable Filter」を選択する.



Output Data Set Type で「Same as Input」を選択して出力データを入力データと同じにし，Script に以下のプログラムを記述して「Apply」する．

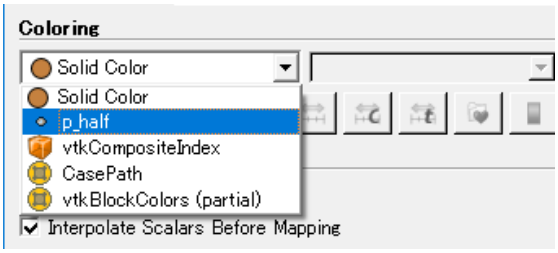


input0 = inputs[0]　// 変数 inputs の 0 番目（並列計算していなければ常に 0？）のデータを，変数 input0 を定義して代入する．

data = input0.PointData[“p”]/2 // input0 クラスの点データ（有限体積の中心データ）の p(圧力)データを 2 で割って定義変数 data に代入．

output.PointData.append(data,”p_half”) // 出力先（output）の点データに p_half という名前で data をアサイン

これより，圧力を半分にした p_half を選択して処理できるようになる．



12.2 場合分け（if 文）と for ループしてみるスクリプト

Script に以下を書いて「Apply」すると，空洞領域の格子の値を 2 から 10 に変更した kikut という変数が使えるようになる．

```
import numpy as np                          // np を使えるようにする宣言
input0 = inputs[0]                          // 0 番のデータ inputs[0]を，input0 という変数を定義して代入
data = input0.PointData["cellTypes"]        // input0 クラスの有限体積の中心データ cellTypes のデータを data を宣言して代入
num = input0.GetNumberOfPoints()            //  有限体積の最大値を取得

pointArray = np.empty(num,dtype=np.int32)       //  空の 32bit 整数型配列を作成．データ数は num

for i in range(num):                            // num 回ループ
    if data[i] < 1.5:                           // if 分
        pointArray[i] = data[i]
    else:
        pointArray[i] = 10

output.PointData.append(pointArray,"kikut")     // kikut というデータ配列を作って出力．
```

12.3 点の移動

```
        pdi = self.GetPolyDataInput()
        pdo =   self.GetPolyDataOutput()
        newPoints = vtk.vtkPoints()
        numPoints = pdi.GetNumberOfPoints()
        for i in range(0, numPoints):
            coord = pdi.GetPoint(i)
            x, y, z = coord[:3]
            x = x * 1
            y = y * 1
            z = 1 + z*0.3
            newPoints.InsertPoint(i, x, y, z)
    pdo.SetPoints(newPoints)
```

12.4 Labeling common points between two datasets

```
import numpy as np                          // np を使えるようにする宣言
        # Code for 'Script'

        # Get the two inputs
        A = inputs[0]
        B = inputs[1]           // 恐らく，並列化していないと 0 のみ．
        # use len(inputs) to determine now many inputs are connected
        # to this filter.

        # We use numpy.in1d to test all which point coordinate components
```

```
        # in    B are present in A as well.
        maskX = np.in1d(B.Points[:,0], A.Points[:,0])
        maskY = np.in1d(B.Points[:,1], A.Points[:,1])
        maskZ = np.in1d(B.Points[:,2], A.Points[:,2])

        # Combining each component mask, we get the mask for point
        # itself.
        mask = maskX & maskY & maskZ

        # Now convert it to uint8, since bool arrays
        # cannot be passed back to the VTK pipeline.
        mask = np.asarray(mask, dtype=np.uint8)

        # Initialize the output and add the labels array

        # This ShallowCopy is needed since by default the output is
        # initialized to be a shallow copy of the first input (inputs[0]),
        # but we want it to be a description of the second input.
        output.ShallowCopy(B.VTKObject)
    output.PointData.append(mask, "labels")
```

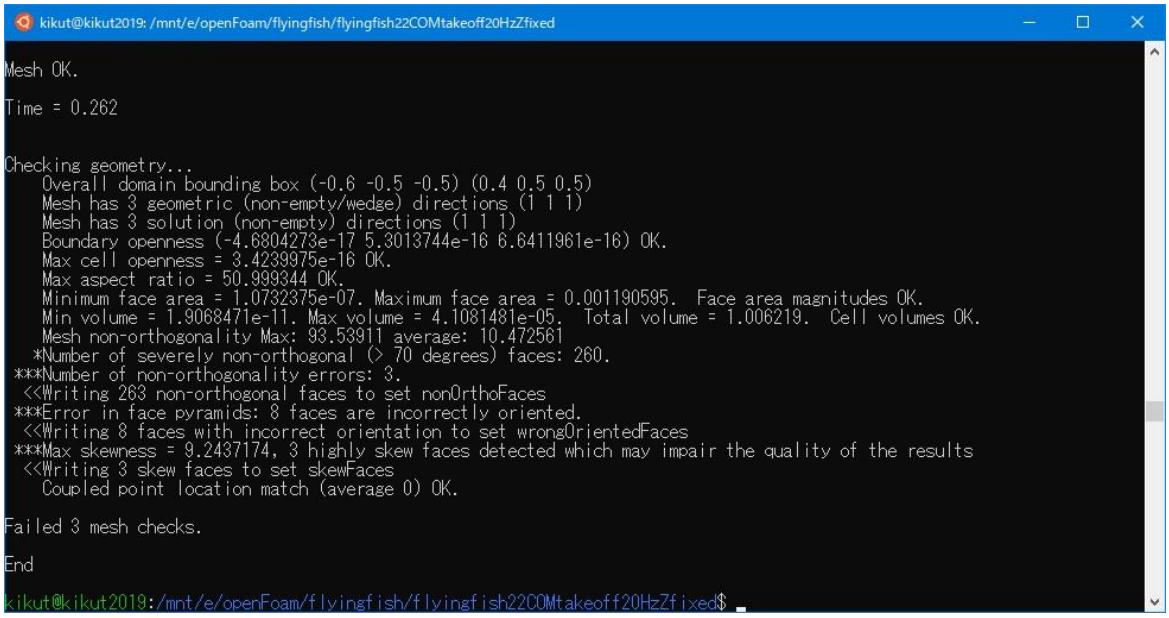
13. 歪んだ格子の表示

FSI 計算において格子の変形量が大きいと，様々な問題が起こり，数値計算が止まる．極端な典型例として，凸形状が凹形状になると計算が飛ぶ．これは，数値積分法に依存している．立方体だった格子がぺらっぺらの平面に近づいても計算精度が極端に落ちる．数値計算が止まった場合，これを可視化することができる．

計算結果がある（constant や system がある）ディレクトリから，

- chekMesh

を実行すると，全てのデータディレクトリに含まれる格子の品質をチェックする．大抵は，計算が止まった後で実行するので，最後のディレクトリで問題が発見される．以下は，0.262 ディレクトリに保存されている格子に問題が見つかった例である．



三つの問題が示されている．

(1) Number of severely non-orthogonal (> 70 degrees) faces: 260

深刻な非直交格子面が 260 個見つかったということ．ただし，これは，計算精度が落ちるが，直接的な計算エラーにはつながらない．以下のファイルに記録されている．

.....¥0.262¥polyMesh¥sets¥nonOrthoFaces

```
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       faceSet;
    location    "0.262/polyMesh/sets";
    object      nonOrthoFaces;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


263         // 問題のあった格子面の数
(
1050968    // 格子面番号
1050971
1050974
1051443
1051444
1051446
```

(2) Number of non-orthogonality errors: 3

直交していない格子が 3 個見つかったということ．これは，深刻なエラー．多くの場合，数値計算を継続できない．以下のファイルに記録されている．

.....¥0.262¥polyMesh¥sets¥ skewFaces

```
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       faceSet;
    location    "0.262/polyMesh/sets";
    object      skewFaces;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

3(1051933 1074895 1074897)
```

(3) Error in face pyramids: 8 faces are incorrectly oriented.

恐らく，向きがひっくり返っている格子面が 8 個あったということ．これも恐らく深刻なエラー．多くの場合数値計算を継続できない．以下のファイルに記録されている．

.....¥0.262¥polyMesh¥sets¥ wrongOrientedFaces

```
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       faceSet;
    location    "0.262/polyMesh/sets";
    object      wrongOrientedFaces;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

8(1051925 1051928 1051930 1051933 1051936 1052407 1074895 1074897)

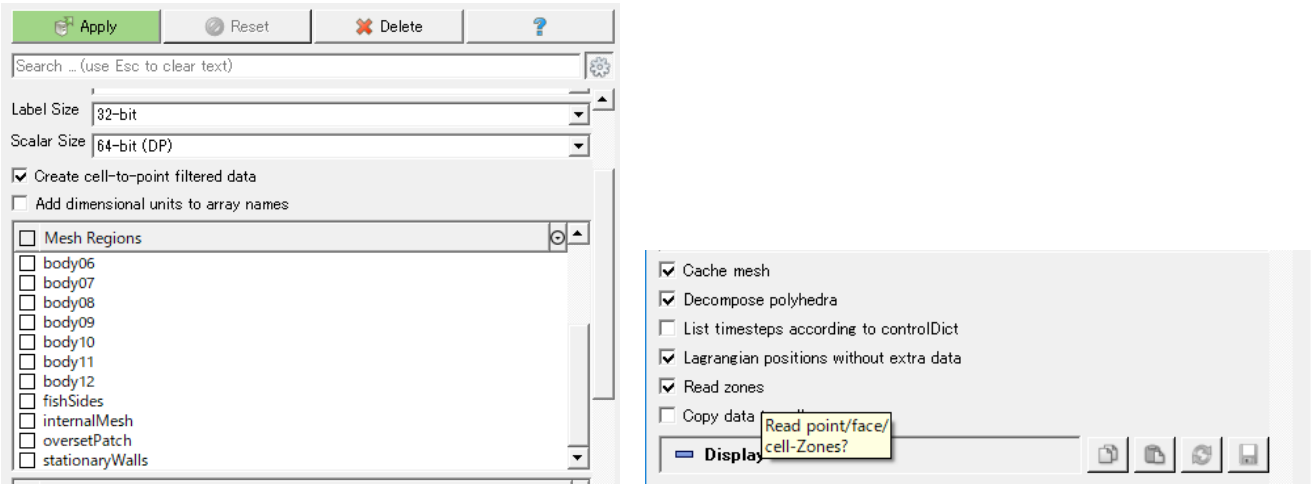
これらは，Paraview で読み込むことができる．以下を実行する．

- setsToZones -noFlipMap

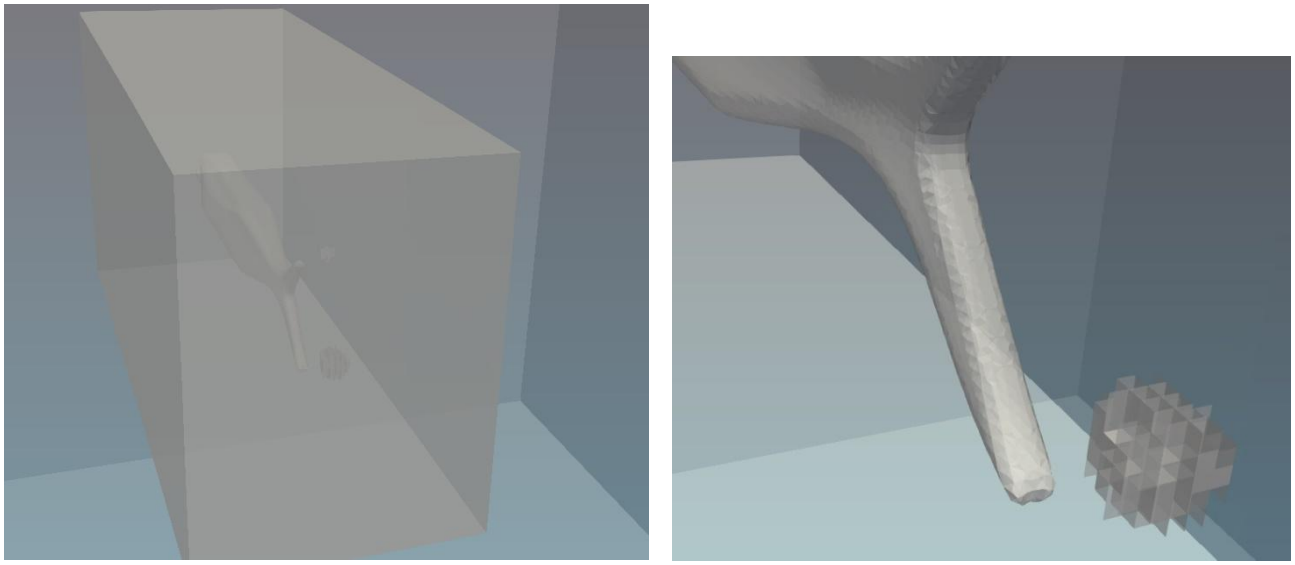
以下の region ファイルが作成され，Paraview で読み込めるようになる．

....¥constant/polyMesh/sets/region**

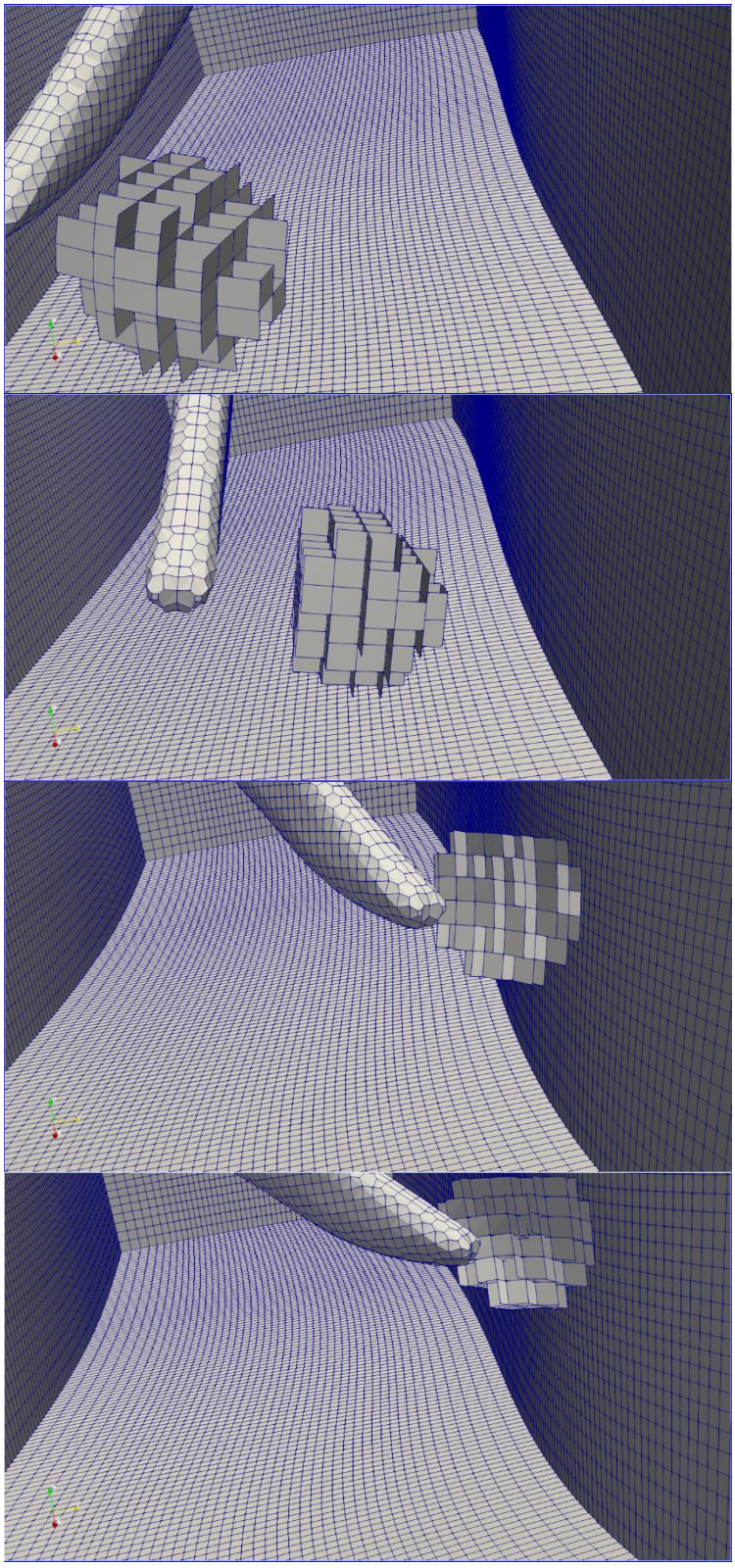
.foam を実行して，Paraview を立ち上げ，Mesh Regions のチェックを外し（”internalMesh”を読み込まず），”Read zones”をチェックして，上記の regionを読み込む．



Opacity を 0.5 程度にして透過させると，問題の格子が表示される．t=0 では，まだ歪んではいない．

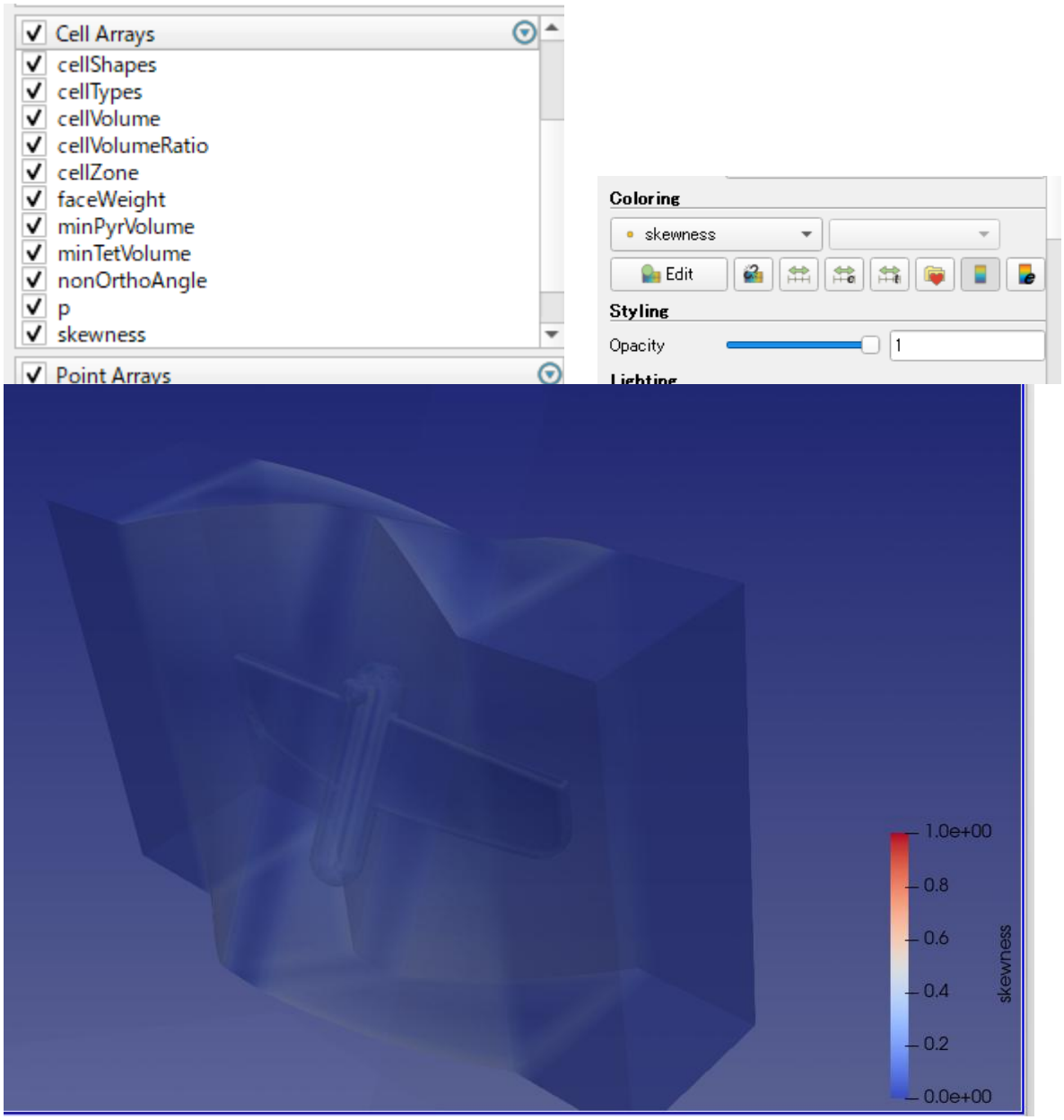


時間と共に，歪んでいくのが分かる．これは，トビウオのカランギフォーム運動での歪の例．



- chekMesh -writeAllFields

のコマンドを実行後，paraview を立ち上げると，変数に skewness，nonOrthoAngle 等のパラメータが追加されているので，coloring で可視化することができる．以下は，skewness の表示例．openFOAM では 4 を超えるとエラーになる．

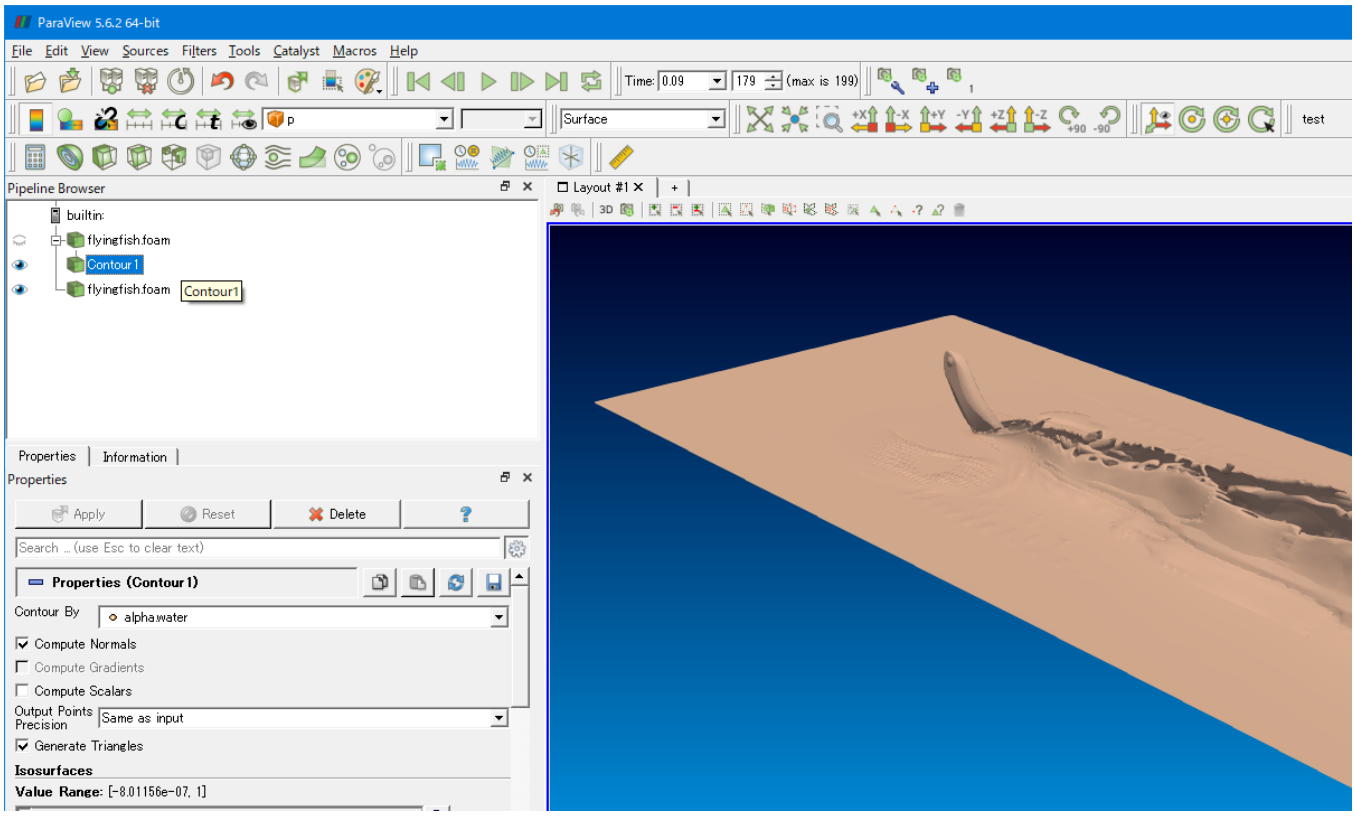


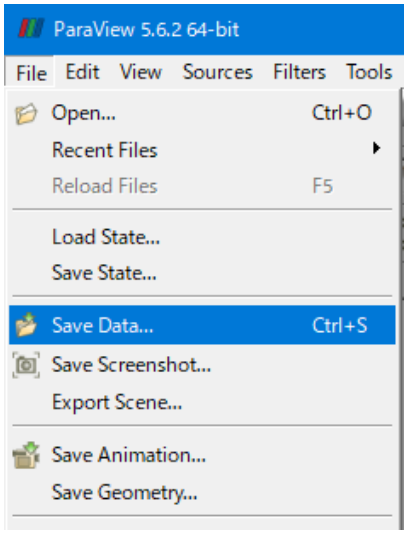
14. 計算結果を stl ファイルで保存して，3D プリントする

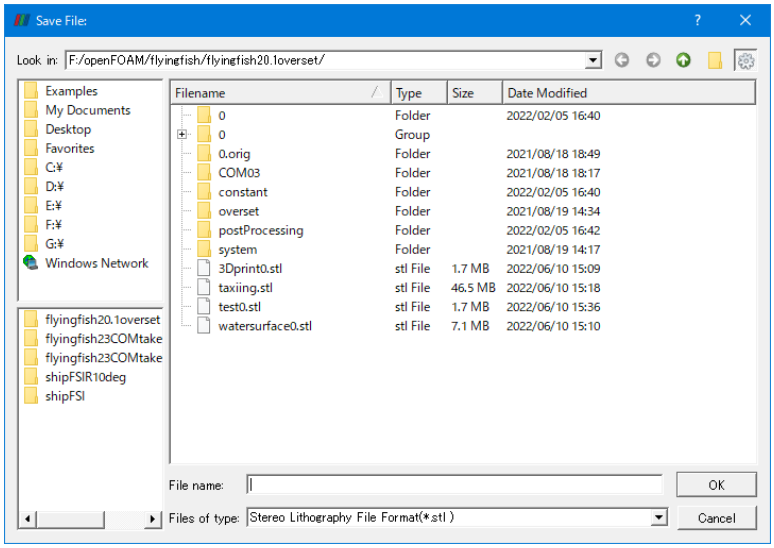
例：トビウオ（1 剛体モデル）のタキシング

．水面とトビウオをそれぞれ paraview から stl ファイルで保存する．

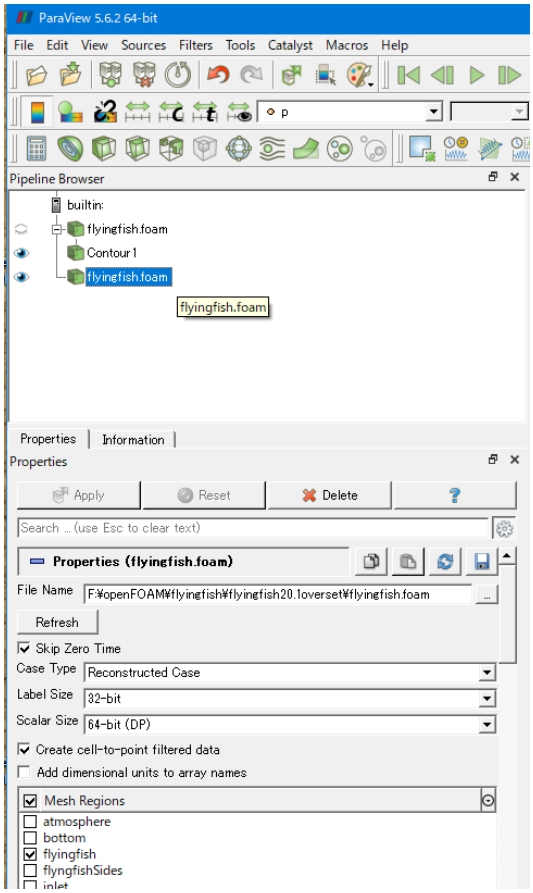
コンタ面（「Contour 1」）を選択し，「ファイル」の「Save Data..」を選択し，File of types: から「Stereo Lithography File Format (*.stl)」を選択し，適当な名前を付けて保存する．



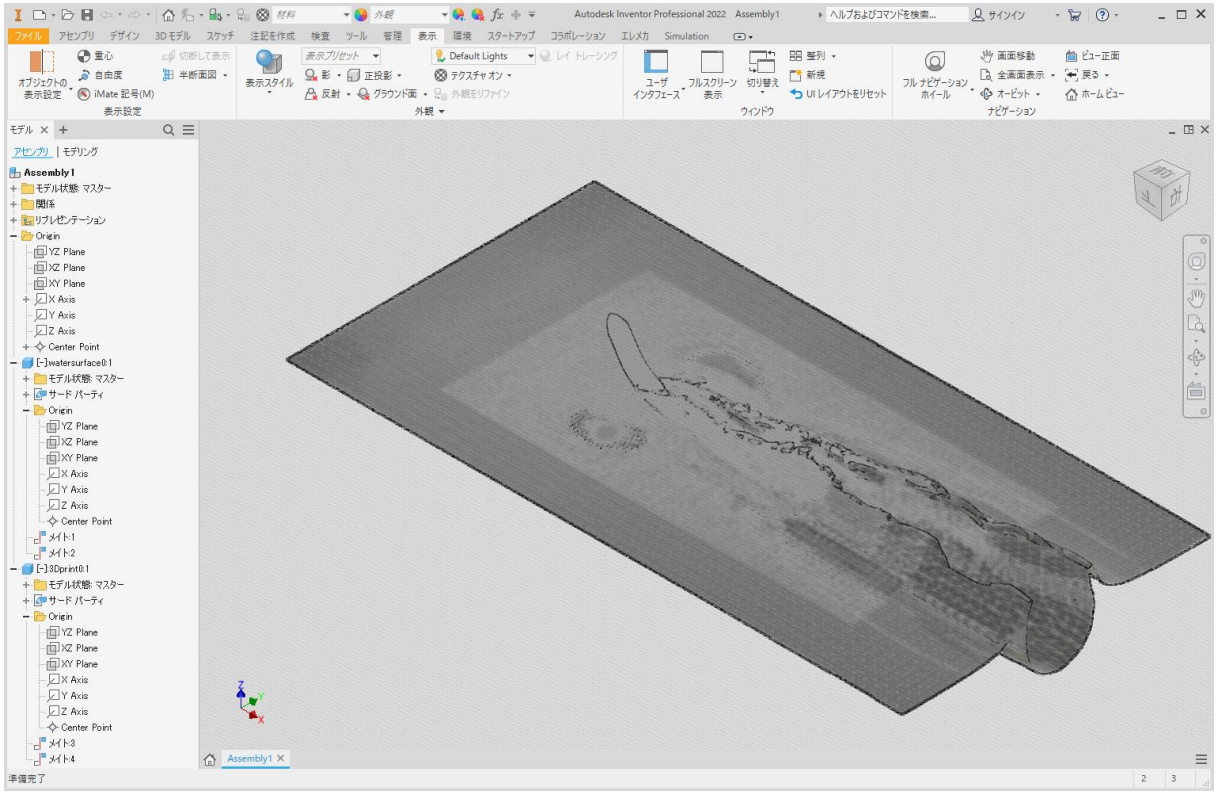




同様に，トビウオ境界面を選択し，stl として保存する．



Inventor を立ち上げ，アセンブリファイルを新規作成し，二つのファイルを読み込んで，原点どおしを拘束する．

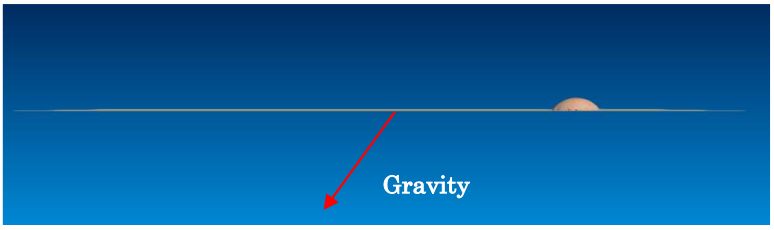


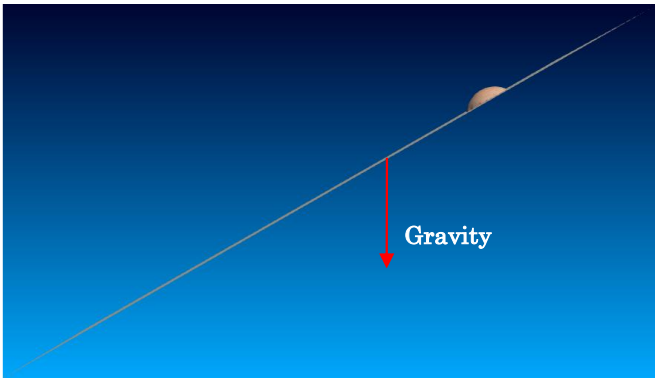
「ファイル」,「書き出し」,「CAD 形式」を選択し，stl ファイルとして保存する．このファイルをそれぞれの 3D printer 用の形式に変換すれば，3D プリントできる．

1 ファイルのみの場合には，paraview からの stl をそのまま利用してもよい．蝶の 10*10 の翅モデルなどは，100 回のアセンブリ拘束が必要なことになる．原点どおしの拘束でつながるが，全て手動で行う必要がある．

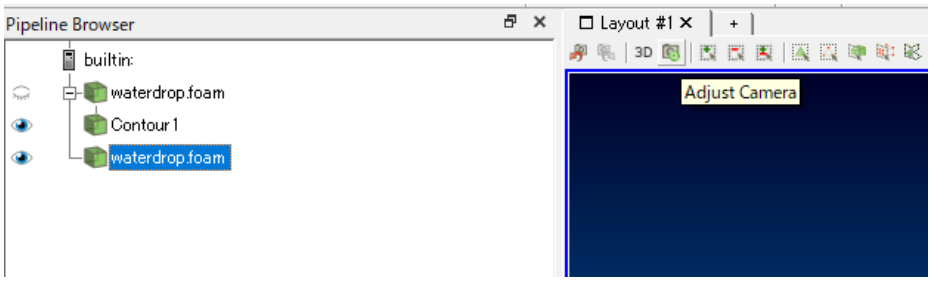
15. 物体を任意の角度で回転する（任意回転表示）

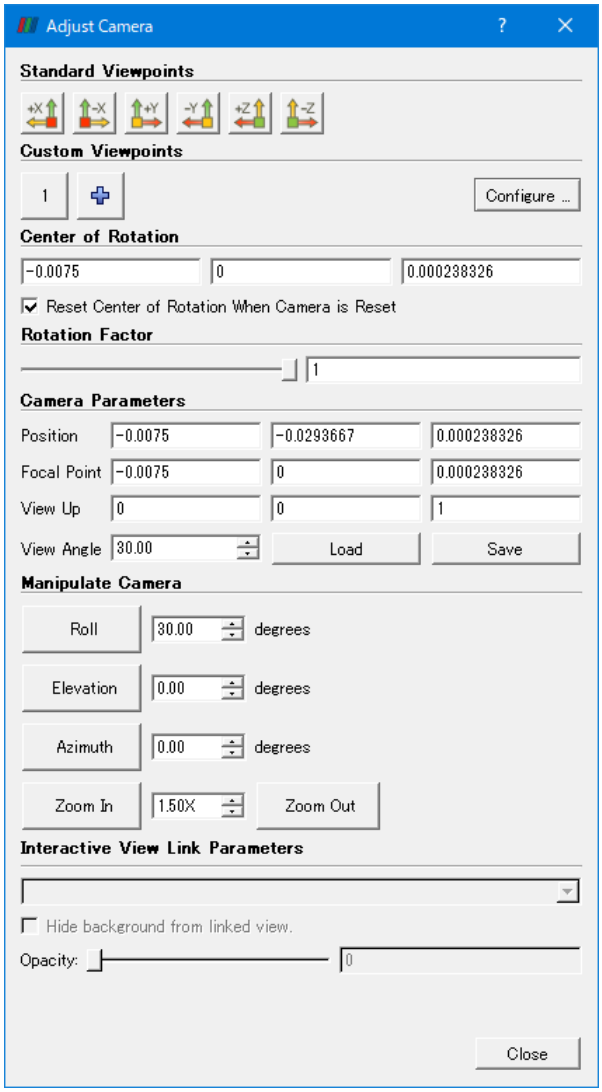
計算空間に対して，重力方向を任意方向に変えている場合に，表示においてその方向に回転させたい時がある．例えば，こういうこと．デフォルトの回転ボタンには 90deg 回転しかない．





「Adjust Camera」を選択し，オープンした Adjust Camera window の Roll の角度を例えば”30”degree にし，「Roll」ボタンを押すと，物体が 30deg ロール回転する．カメラがロール回転しているということ．もう一度クリックすると，さらに 30deg，初期角度からは 60deg 回転する．その他，一般的なカメラ設定ができる．





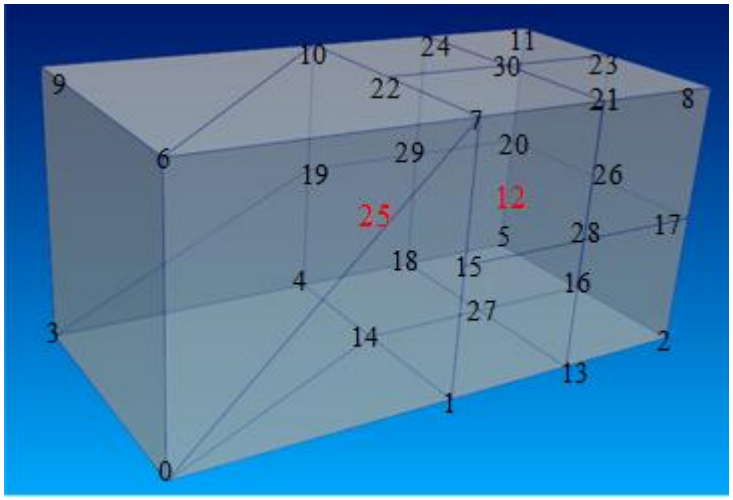
16. Paraview の表示方法に関して

面表示に関しては，実際にないエッジまで直線として表示されるので注意が必要である．頂点番号が図のようにナンバリングされているとする（赤は内部の頂点）と：

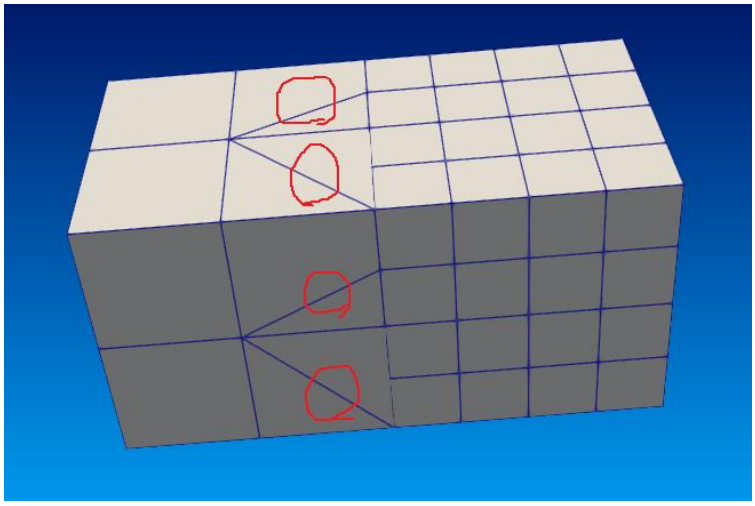
. snappy は，左右の隣り合うセル同士で頂点を共有していない（ex. 頂点 5）場合，n 角形（この場合は 5 頂点で四角形）を作る．0-1-15-7-6 は，ひとつの面.

. この場合，0-7 という線は存在していないが，描画する．6-10 の線，3-19 の線，0-14 の線も実際は存在していない.

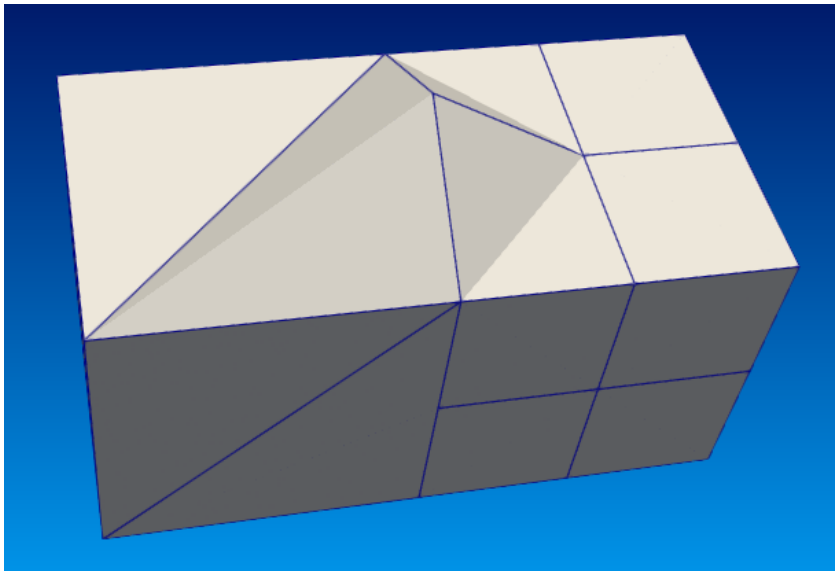
. 面を曲面として描画できず，三角形の面を組み合わせて描画している．このため，四角形は二つの三角形で，五角形は三つの三角形で描画する.



以下の赤丸の線は実質存在しないが，描画される．



以下の図を見ると，6-7-22-10-9 の面が三つの三角形で表示され，その一つのエッジを間違って表示してしまっているのが分かる．



17. Paraview の可視化情報の共有化マニュアル

Paraview で可視化したものを pvsm ファイル形式で保存することにより，可視化情報の共有化及び保存が可能となり，研究活動，研究室メンバーや教授との議論を円滑に進めることができるようになる．ここでは，「pvsm ファイルの出力」から「別の PC から出力された pvsm ファイルの読み込み」までの流れを説明する．

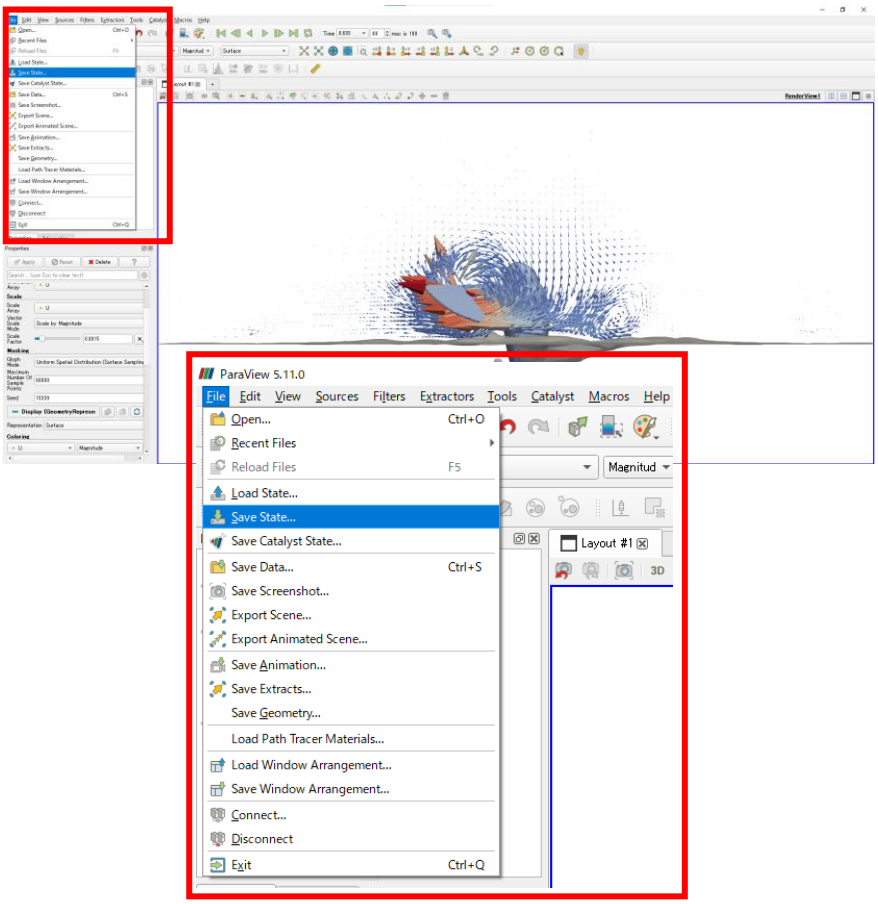
- 注意事項
  - ✓ **Paraview のバージョンを揃える．**バージョンが揃っていないと，圧力などの range やカラー情報などが正しく読み込まれない．
  - ✓ **Paraview のバージョンは最新のものから一つ前のバージョンがおすすめ．**
    → ダウンロードはこちらから（https://www.paraview.org/download/）
  - ✓ 共有する相手側のパソコンまたは HDD 内に同様の解析データが必要．

Step.1　psvm ファイルの出力

保存，共有化したいものを可視化し終えたら，

「File」→「Save State…」を選択し，**解析データがあるディレクトリに保存する．**

保存名は，何を可視化したのかが分かるようにする．

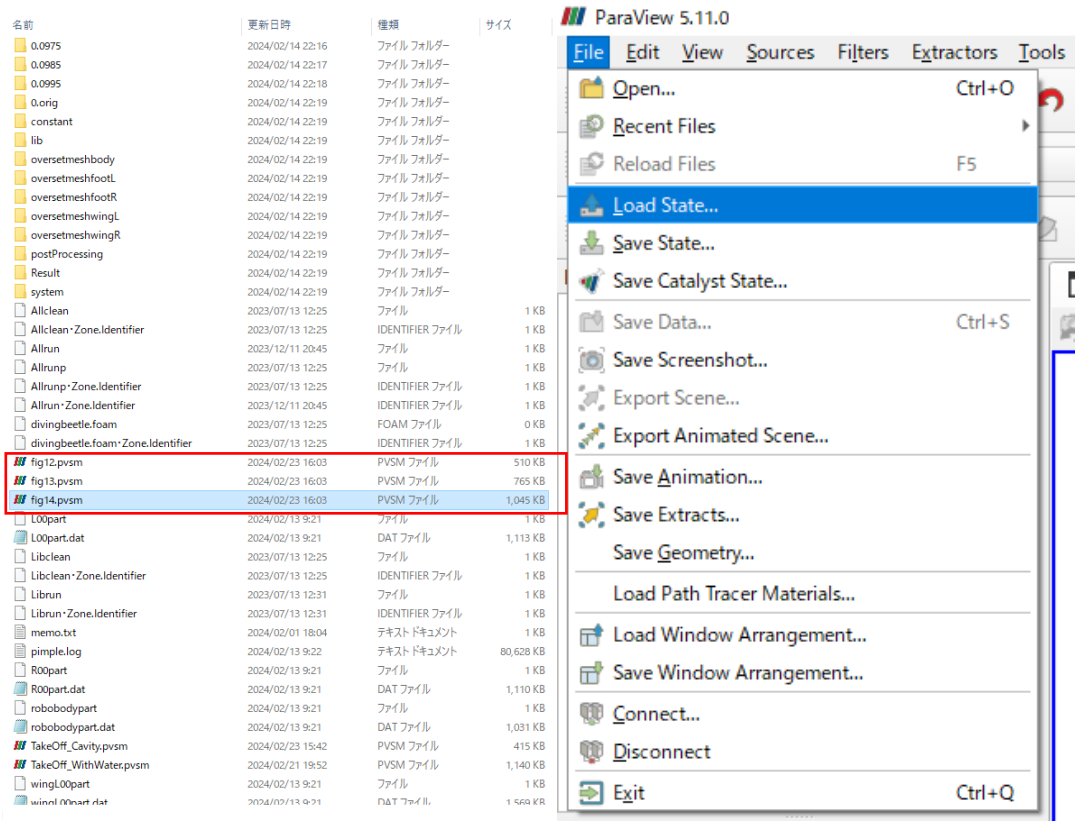


Step.2　pvsm ファイルの読み込み

出力された pvsm ファイルはこんな感じ（左図の赤枠）．複数の pvsm ファイルを作成することができ，異なった可視化情報を個々に保存できる．

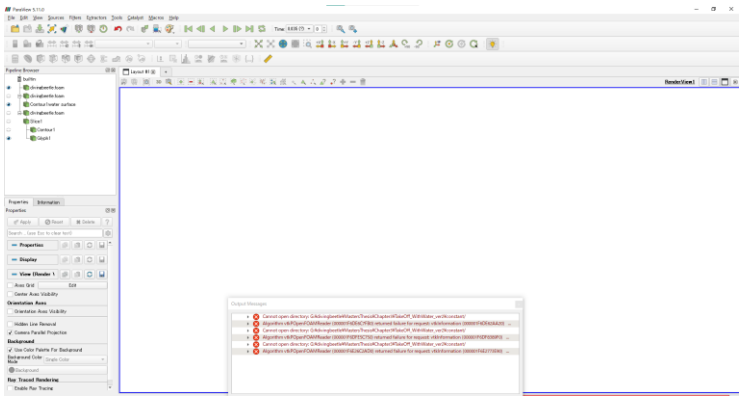
任意の pvsm ファイルをダブルクリックで開く．

※上記の方法で開かない場合は，paraview を起動し，「File」→「Load State…」から，Step1 で出力した pvsm ファイルを開く．（右図）



Step.3　読み込んだ後の処理

読み込んだ際に警告が出るが無視してよい.

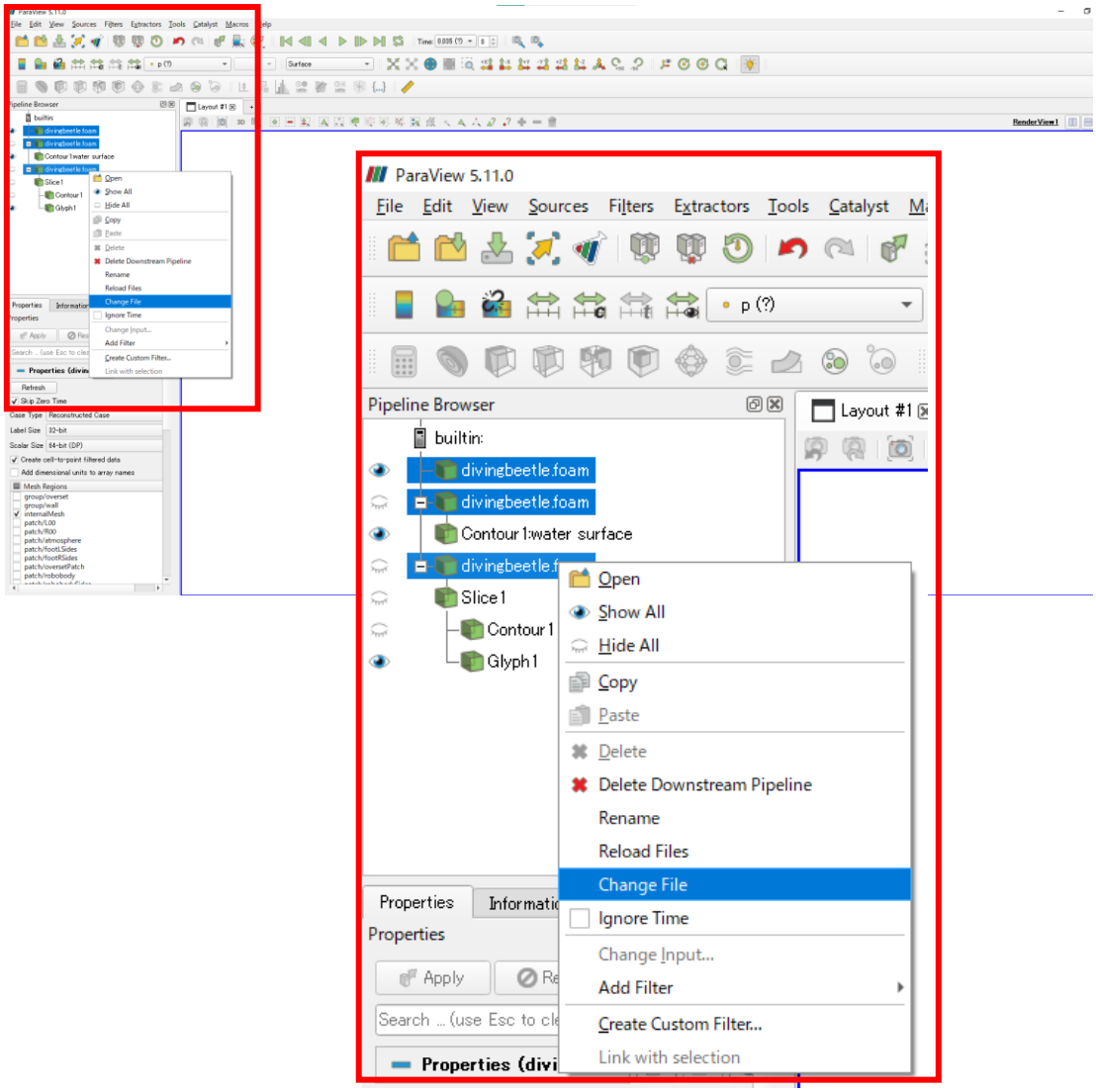


ここで，何も表示されていない状態になっているため，以下の手順を実行する.

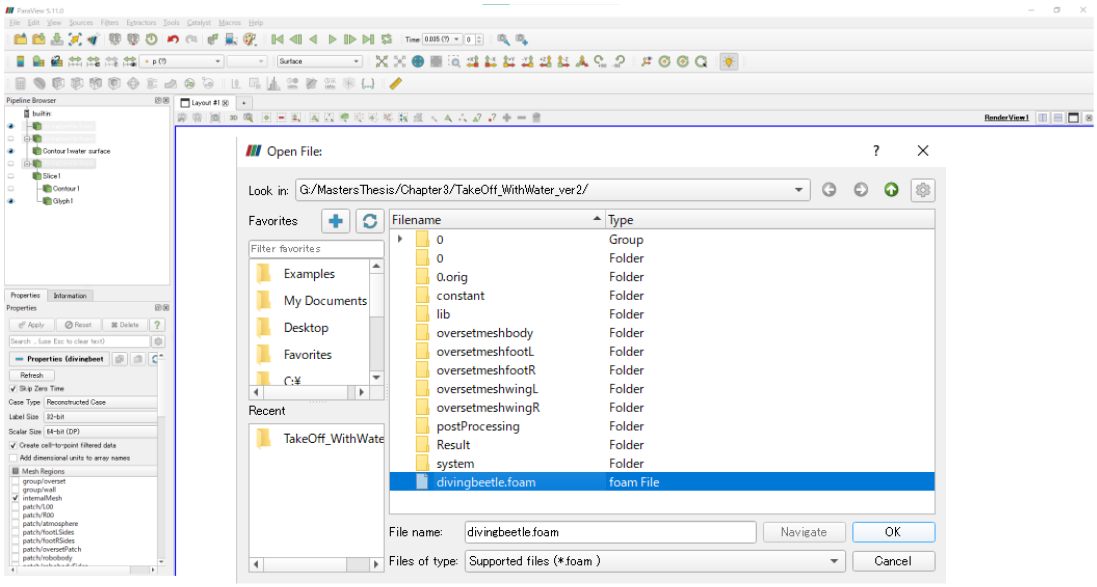
1. Paraview 上の左タブの foam ファイルを全て選択した状態で右クリックし，「Chage File」を選択
2. Pvsm ファイルが保存されているディレクトリへ移動し，******.Foam ファイルを読み込む

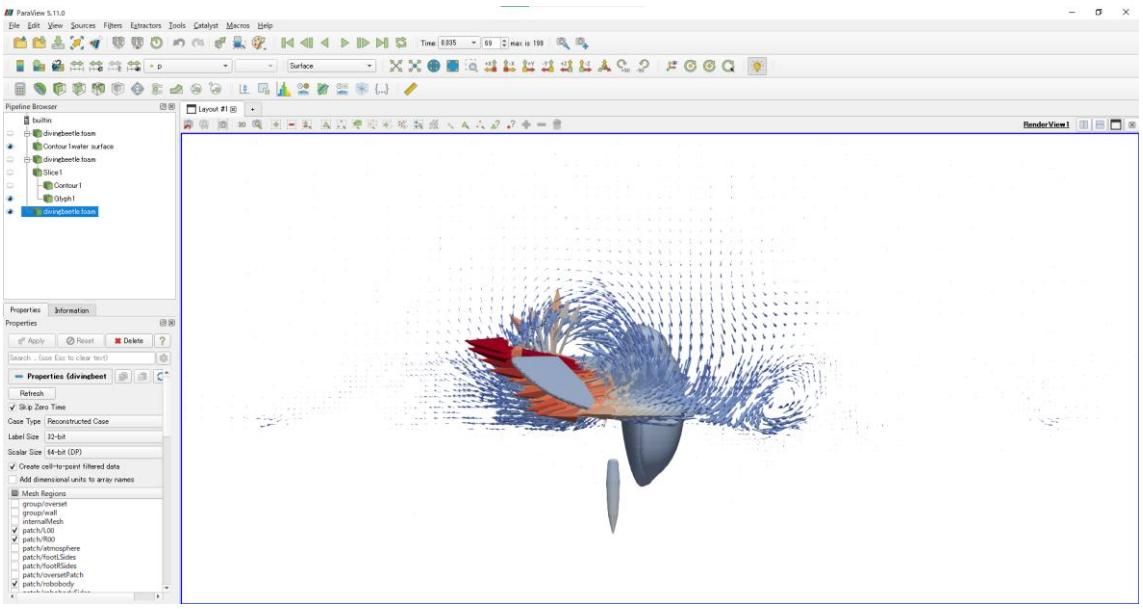
　⇒　Step1 で保存したものが可視化される.

1.



2.





18. 流線（Stream line）のデータの表示

流線を表示したい三次元空間を選択し，「Stream Tracer」を選択する．

次に，流線を描く初期点群を表示する方法を決定する．ここでは，線を指定し，その線上に点群を配置する．

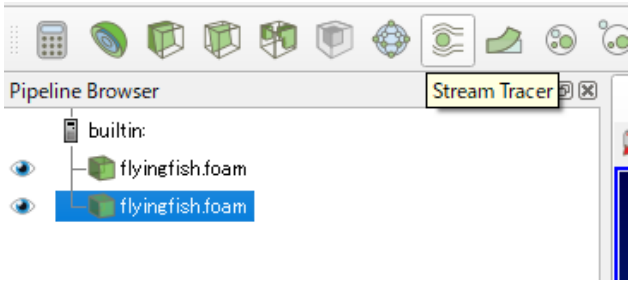
Seed Type を「Line」にし，線分の開始点と終点を入力する．ここでは，

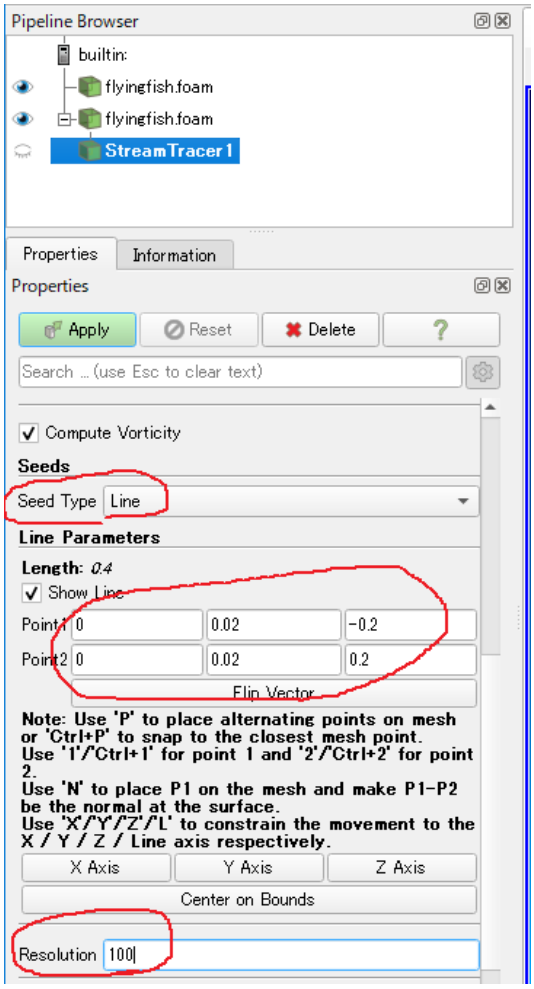
Point 1 が（0, 0.02, -0.2） [m]

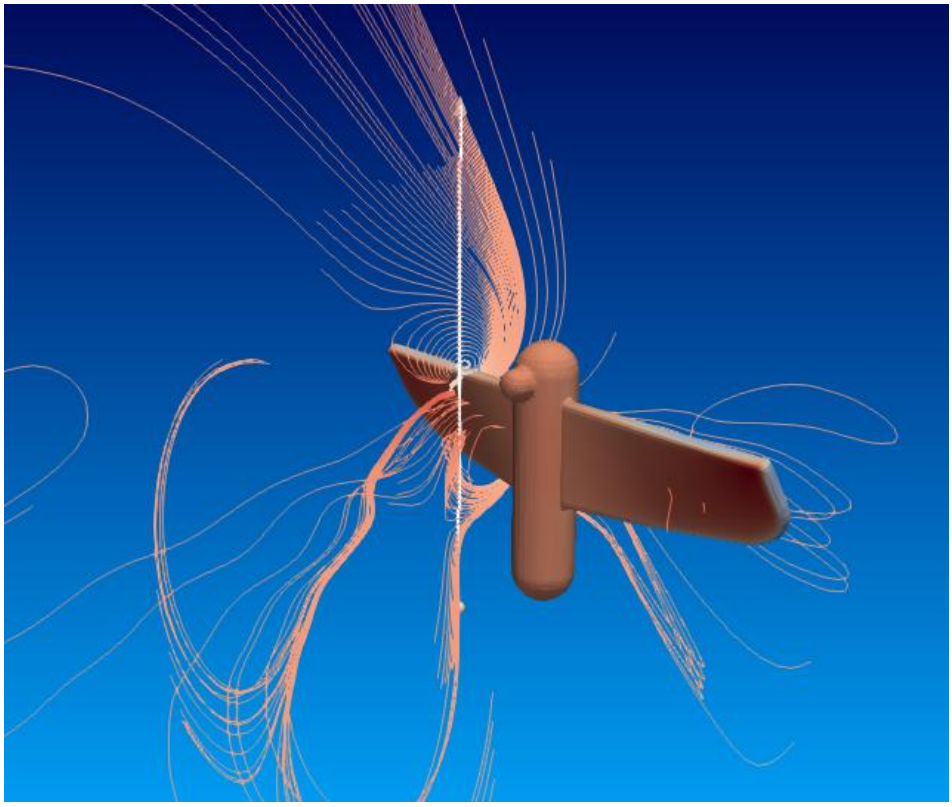
Point 2 が（0, 0.02, 0.2） [m]

またこの線分上の配置点の数を

Resolution で「100」点にして，「Apply」する．







なお，Integration Parameters で，積分方向（この例は BOTH: 未来と過去），積分法，積分時間など設定できる．