チュートリアル

フリーの可視化ソフトとして今日広く利用されているParaViewの使い方についてその初歩の初歩を解説します。ParaViewに関する予備知識が全く無い人が四面体などの非構造格子を用いた有限要素解析結果をLegacy VTKファイル形式で書き出してParaViewで表示出来るようになることを目指し、なるべく丁寧に説明します。一般的なParaViewの操作方法についてはあまり深く解説を行いませんので、書籍およびウェブ上にあるParaViewの操作方法に関するドキュメント類と併せてお読みください。

ParaView超入門

大西 有希

1　はじめに

有限要素法（FEM）を始めとする数値解析を実施する場合、私達は大きく分けて「プリ」・「ソルバ」・「ポスト」の3種類のソフトウェアを利用します。「プリ」は空間離散化（メッシュ切り）と解析条件（境界条件、初期条件など）の割り当てを行い、解析条件入力ファイルを作ります。「ソルバ」は解析条件入力ファイルを読み込んでその問題を解き、解析結果出力ファイルに書き出します。「ポスト」は解析結果出力ファイルを読み込んで様々な画像やグラフを表示して可視化します。この記事で取り上げるParaViewというソフトは上記3種類の中の「ポスト」の役割を専門に担うソフトウェアです。

ParaViewをご存知無かった方は先ず一度ウェブページのギャラリー（https://www.paraview.org/gallery/）を覗いてみてください。綺麗な可視化例が幾つも展示されています。しかも、このソフトはフリー（修正BSDライセンス）で入手できるのです。ソースコードをダウンロードして読んだり書き換えたりすることも基本的に自由です。

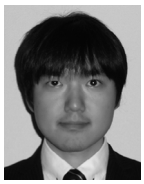
その名に“Para”の文字が入っていることからも分かる通り、ParaViewは複数のCPU（あるいはコア）を用いた並列処理に対応しており、大規模な解析モデルの可視化に対応しています。ただし、並列処理を用いない環境でも全く問題なく動作し、「普段使い」にも適したソフトです。例えば数万要素程度のモデルであれば、一般的なノートパソコンでも充分に使うことが出来ます（本稿では並列可視化には触れません）。

本稿ではParaViewを用いて自作のFEMソルバ等の解析結果を可視化する方法に焦点を絞って解説します（ParaViewの基本的操作方法については参考文献等をご参照ください）。前半はParaViewが読み込むことのできる最も基本的なファイル形式であるLegacy VTKファイル（拡張子がvtk）で可視化モデルを作り、それをParaViewで可視化するまでの流れを丁寧に説明します。後半はアニメーションの作成、グループデータの利用、バイナリ出力およびVTK XMLファイル出力のためのPythonスクリプトについて掻い摘んで説明します。

汎用解析ソフトを触ったことのある人の中には、「プリ」・「ソルバ」・「ポスト」は1つのパッケージソフトとしてまとまっているのが当たり前で、可視化専用ソフトを単独で使う機会など自分には無いと思われるかもしれません。確かに、特定の汎用解析ソフトの1ユーザーとして研究・開発を行うのであればその通りかも知れません。しかし、その汎用ソフトでは解けない問題を自作または非汎用のソルバで解かなければならない時や、数種類のソルバの解析結果を統一的に比較したい時など、ParaViewの様なソフトを利用した方が良い状況が今後出てくるかも知れません。この機会に是非ご一読頂けると幸いです。

続きはWebで
日本計算工学会誌「計算工学（Vol.23, No.4）」HP:
http://www.jsces.org/activity/journal/

筆者紹介



おおにし ゆうき
2005年東京工業大学大学院情報理工学研究科博士課程修了、みずほ情報総研株式会社を経て、現在、東京工業大学工学院助教。固体の超大変形解析、腐食や電着塗装などの電気化学解析、熱／光インプリントのプロセス解析などの研究に従事。

2　ParaViewのインストール

ParaViewのウェブページにある“Download”のリンクからダンロードのページに移ることが出来ます。最新のバージョン（2018年8月末時点ではv5.5）を選択し、Windows/Linux/macOSのいずれかのバイナリをダウンロードします。最近のバージョンでは64bit版のみしか配布されていないため、32bitのOSをお使いの場合は注意が必要です（少し古いv5.2であれば32bitのWindows用バイナリが配布されています）。Windowsユーザーでどれをダウンロードすべきか全く見当がつかない方は“ParaView-5.2.0-Qt4-OpenGL2–Windows -32bit.exe”をダウンロードすると良いでしょう。また、Linuxユーザーであればお使いのディストリビューションが配布しているParaViewのパッケージをインストールすれば良いでしょう。

同ページにはDocumentationとしてチュートリアルやガイドのPDF[1-3]が用意されています。英語が不得意な初学者には難解かと思いますが、ある程度使い慣れたら是非目を通しておくべき文書です。

3　VTKファイルについて

ParaViewはVisualization Toolkit (VTK)と呼ばれるライブラリを用いてモデルの描画を行っています。従って、VTK純正のファイルフォーマットを使ってParaViewにデータを読み込ませるのが最も基本的な使い方となります。ただし、VTKのファイルフォーマットにも様々な種類が存在します。詳細はVTKの解説書[4]を参照ということで省略しますが、本稿では最も簡単なLegacy VTKファイルフォーマット（拡張子vtk）を利用します。

Legacy VTKファイルは基本的にはテキストファイルで、改行およびスペース区切りの形式です。浮動小数点の実数は単精度（float）および倍精度（double）をサポートしますが、ParaViewで表示する時点で多くの環境ではfloatに丸め込まれるため、実数値の与え方はfloatを用いることになります。改行コードの指定は無く、ParaViewではどの改行コードでも適切に処理されるようです。

4　4節点四面体1要素モデルの作成と可視化

4.1　形状の定義

では、ParaViewで可視化するモデルを作っていきます。まずは4節点四面体要素1つだけから成るモデルを作ってみます。お好みのテキストエディタ（Windowsのメモ帳でも可）で下記のテキストファイルを作成し、適当なファイル名（ここでは“sample-a.vtk”とします）で保存してください。

```
----------
# vtk DataFile Version 4.1
This is a comment line.
ASCII
DATASET UNSTRUCTURED_GRID

POINTS 4 float
0. 0. 0.
1. 0. 0.
0. 1. 0.
0. 0. 1.

CELLS 1 5
4  0 1 2 3

CELL_TYPES 1
10
----------
```

1行目はVTKファイルであることを宣言する「おまじない」です。2行目はコメント行なので、読み飛ばされます。3行目はデータがアスキー（ASCII）形式かバイナリ（BINARY）形式のどちらで記述されているかを指定する行で、ここではアスキー形式を用いています（バイナリ形式については後述）。4行目は形状データの構造を指定する行で、一般的なFEMでは非構造格子（UNSTRUCTURED_GRID）を使用します。

“POINTS”で始まるブロックは節点数、精度、および各節点の座標を宣言しています。“POINTS”に続く“4”は節点が全部で4個あることを意味し、続く“float”は座標を単精度の浮動小数点として読み込むよう指示しています。その下の4行は各行ともxyz座標値を表しており、節点0、1、2、3の座標がそれぞれ(0,0,0)、(1,0,0)、(0,1,0)、(0,0,1)であることを意味しています。ここで、VTKの節点番号は1からではなく0から始まることに注意してください。

“CELLS”で始まるブロックは要素数、要素リストに用いられる数字の個数、および要素リストを表しています。“CELLS”に続く“1”は要素が全部で1個であることを意味し、続く“5”はその下に続く要素リストの定義に用いられる数字の個数が5個であることを意味しています。その下の要素リストの最初の“4”はその要素の節点数が4個であることを意味し、それに続く“0 1 2 3”はその要素の構成節点の番号を表します。

“CELL_TYPES”で始まるブロックは要素数および各要素の要素タイプ番号を表しています。“CELL_TYPES”に続く“1”は要素が全部で1個であることを意味しています。その下の“10”は要素タイプ番号で、先に“CELLS”で定義した要素が4節点四面体であることを指定しています。

ではこの“sample-a.vtk”をParaViewで可視化してみます。ParaViewを起動し、画面左上のアイコン「Open」から“sample-a.vtk”を開いてください。画面左側の“Apply”ボタンを押して画面上にグレーの四面体が1個表示されれば成功です（図1）。もしファイルフォーマットに誤りがある場合は画面上にエラーメッセージが現れますので見直してください。マウスの左・中・右ボタンをドラッグして動かせば回転・移動・拡大縮小などが出来ると思います。詳細は参考文献[5, 6]およびweb検索をご参照ください。

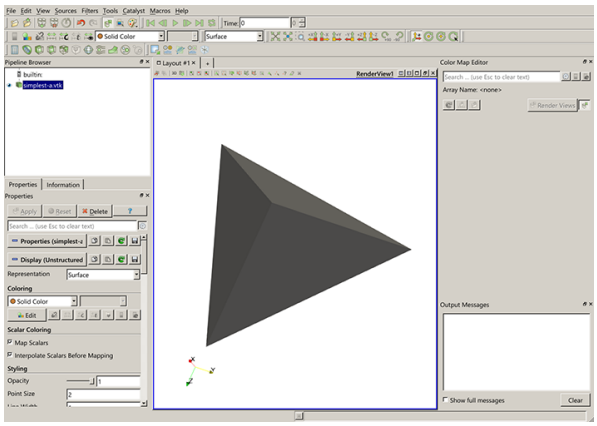


図1　4節点四面体要素形状の可視化例

4.2　スカラーの節点量および要素量の定義

現状の“sample-a.vtk”は幾何学的な情報しか含まれておらず、グレー一色の物体が表示されているだけでした。そこで次は何かしらの量（例えば温度など）をこの物体上に表示させたいと思います。節点量ならびに要素量としてスカラー値をこのvtkファイルに追加してみます。先程の“sample-a.vtk”の末尾に下記のテキストを追記して保存してください。

＜先程の“sample-a.vtk”の末尾への追記分＞

```
POINT_DATA 4
SCALARS Point_Scalar_Data float
LOOKUP_TABLE default
0.123
1.234
2.345
3.456

CELL_DATA 1
SCALARS Cell_Scalar_Data float
LOOKUP_TABLE default
3.14
```

“POINT_DATA”で始まるブロックは節点量を定義しています。“POINT_DATA”に続く“4”は節点数を表します。“SCALARS”の行は“Point_Scalar_Data”という名前（任意に設定可）で“float”の精度を持つデータを定義することを宣言しています。続く“LOOKUP_TABLE”はやや高度なカラーリングをしたい場合に利用される機能なので、本稿では解説を省略します（“LOOKUP_TABLE default”の1行は「おまじない」だと思ってください）。その下の4行は節点0~3の“Point_Scalar_Data”という量のスカラー値を順に並べて指定しています。

“CELL_DATA”で始まるブロックは要素量を定義しています。記述方法は“POINT_DATA”とほとんど同じなので容易に想像がつくかと思いますが、“Cell_Scalar_Data”という名前（任意に設定可）のスカラー量を1個の要素に与えています。FEMに明るい方であれば「要素量ではなく積分点量を定義したい」と考えるかと思いますが、残念ながらVTKフォーマットで積分点量を定義することは出来ません。

では、ParaViewを起動して先程と同じ手順で追記された“sample-a.vtk”を可視化してみてください。画面上側にコンター図として表示する量を選ぶプルダウンのボックスがあるので、追記した“Point_Scalar_Data”と“Cell_Scalar_Data”を選んでそれぞれ可視化してみます。“Point_Scalar_Data”は節点間で補間された分布が表示される（図2）のに対し、“Cell_Scalar_Data”は要素内でベタ塗りに表示される（図3）ことが確認できると思います。

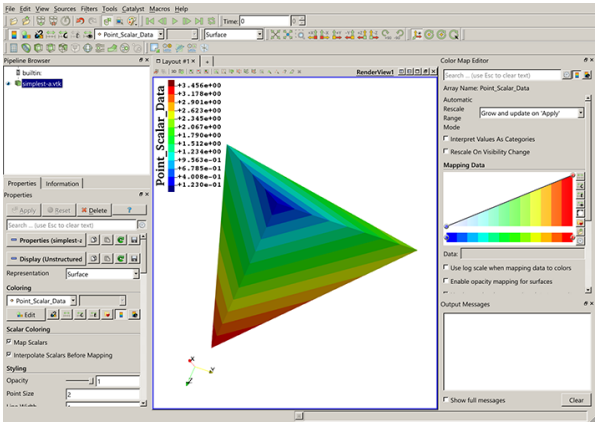


図2　節点スカラー量の可視化例

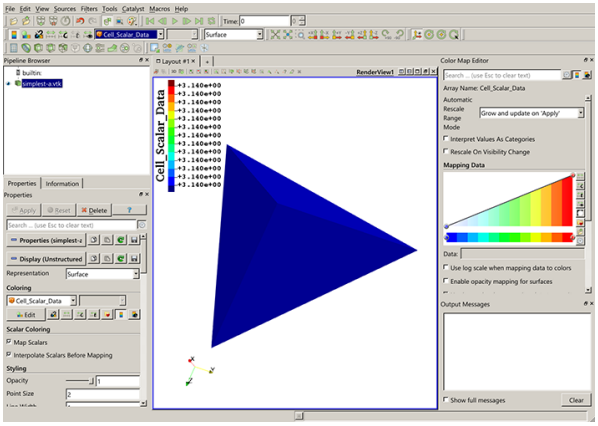


図3　要素スカラー量の可視化例

コンターの色使いやカラーバーの表示には様々なオプションが用意されており、Color Map Editorをいじることで変更できます。直感的に分かりやすい仕様ですので、皆さんの手で試してみてください。

4.3　ベクトルの節点量および要素量の定義

続いて、ベクトル量（例えば変位ベクトル）を表示させたいと思います。節点量ならびに要素量としてベクトル値をvtkファイルに更に追加してみます。先程の“sample-a.vtk”の末尾に下記のテキストを更に追記して保存してください。

＜先程の“sample-a.vtk”の末尾への更なる追記分＞

```
POINT_DATA 4
```

```
VECTORS Point_Vector_Data float
0.0  0.1 -0.2
0.1  0.2  0.3
0.2  0.3  0.4
0.3  0.4  0.5

CELL_DATA 1
VECTORS Cell_Vector_Data float
1.2 -2.3 -3.4
```

スカラーの場合とほとんど同じですが、“SCALARS”の代わりに“VECTORS”になっている点、“LOOKUP_TABLE”の行が不要になった点、および各節点／要素に与える数値が1個ずつではなく3個ずつになった点が異なります。

では、ParaViewを起動して可視化してみましょう。先程と同じ手順で可視化すると、ベクトルの各成分あるいはベクトルの大きさ(Magnitude)でコンター表示を行うことが出来ます。節点量でも要素量でも手順は同じです。でもベクトル量ですから、ベクトルで表示したいことも多いかと思います。そんな時は「Glyph」のアイコンをクリックします。画面左側にGlyph1の項目が現れるので、Active AttributesのVectorsを“Point_Vector_Data”に設定すると節点からのびる4本のベクトルが表示されます(図4)。ベクトルの形・長さ・色などは様々に変えることが出来るので、皆さんでいじって試してみてください。

一方、“Cell_Vector_Data”の可視化には一手間必要です。“Point_Vector_Data”と同様の手順で可視化すると要素量にも関わらずあたかも節点量かの様に4本のベクトルが表示されてしまいます。要素ベクトルを表示する時は、まず画面左上の“sample-a.vtk”が選択された状態でメニューバーからFilters→Alphabetical→Cell Centersをクリックします。これで要素中心に仮想的に節点が追加され、そこに定義したCELL_DATAがコピーされたCellCenters1というデータが出来ます。次に、このCelCenters1に対してGlyphを実施します。そうすると要素中心からのびる1本のベクトルが表示出来ます。図5では“simplest-a.vtk”の表示をWireframeにしてベクトルを見やすくしています。

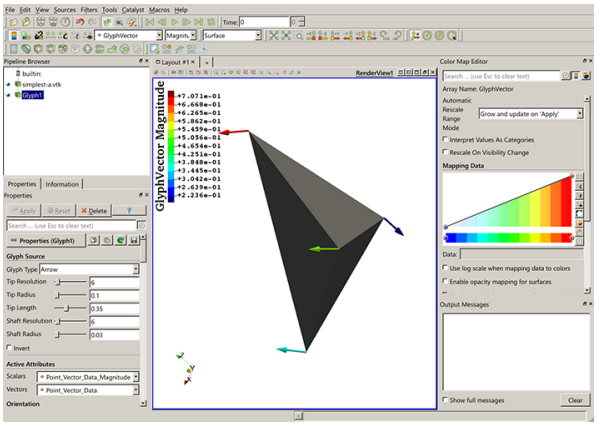


図4　節点ベクトル量の可視化例

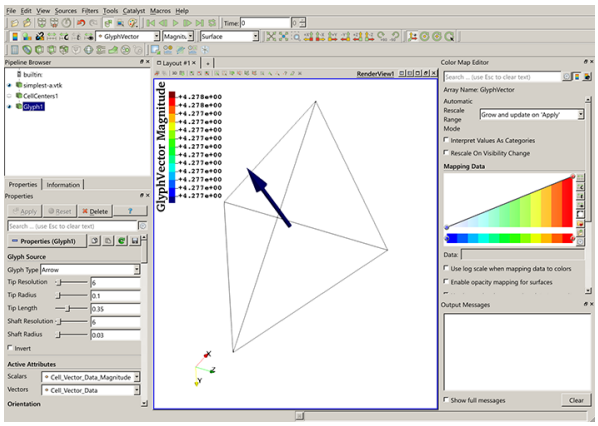


図5　要素ベクトル量の可視化例

テンソルの節点量および要素量については説明を割愛しますが、ベクトルと同様に定義することが出来ます。ただし、ベクトルの様に矢印等で可視化することは難しく、例えば応力テンソルからMises応力や静水圧応力(圧力の逆符号)の様なスカラー値を抜き出して可視化することになります。VTKの解説書[4]等を参考にしてみてください。

5　複数要素モデルの作成と可視化

次に、要素が4節点四面体1個ではなく異なるタイプの要素が複数個ある場合のモデルを作ってみます。ここでは、4節点四面体、6節点五面体、および8節点六面体が1個ずつ計3個の要素から成るモデルを作ってみます。先程と同様に、下記のテキストファイルを適当なファイル名(ここでは“sample-b.vtk”とします)で保存してください。

```
# vtk DataFile Version 4.1
This is a comment line.
ASCII
DATASET UNSTRUCTURED_GRID

POINTS 11 float
 0.   0.   0.
 1.   0.   0.
 0.   1.   0.
 0.   0.   1.
-1.   0.   0.
-1.   0.   1.
-1.   1.   0.
 0.   2.   1.
-1.   2.   1.
-1.   1.   2.
 0.   1.   2.

CELLS 3 21
4  0 1 2 3
6  0 3 2 4 5 6
```

```
8  2 6 5 3 7 8 9 10
CELL_TYPES 3
10
13
12
```

節点数が11個に、要素数が3個に増えています。“CELLS”の下の3行が4節点四面体、6節点五面体、8節点六面体を行毎に定義しており、行頭に構成節点数を、以降行末まで構成節点番号が並んでいます。“CELLS”の行末の“21”はその下に続く要素リストの定義に用いられる数字の個数を表していますから、(1+4)+(1+6)+(1+8)=21という内訳です。また、“CELL_TYPES”の下の3行の“10”、“13”、“12”はそれぞれ4節点四面体、6節点五面体、8節点六面体の要素タイプ番号を表します。

では、ParaViewで“simplest-b.vtk”を可視化してみます。図6では表示を“Surface With Edges”としています。3つの異なる要素が表示されていることが確認できます。

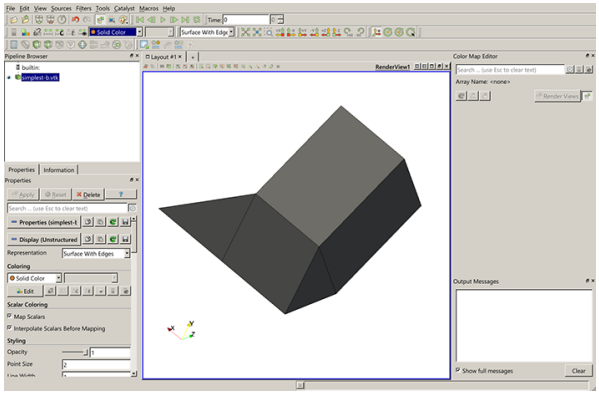


図6　3要素モデル形状の可視化例

使用できる要素タイプや要素タイプ番号および各要素の構成節点番号の並び順などについてはVTKの解説書[4]を参照してください。

6　連番ファイルによるアニメーション

例えば時間発展計算を行う動的問題の解析結果であれば、アニメーションを表示させたくなります。ParaViewでは時相毎のVTKファイルを連番で用意するだけで簡単にアニメーションを表示させることが出来ます。

例えば時刻t=0のデータを“test-000.vtk”に、t=⊿tのデータを“test-001.vtk”に、t=2⊿tのデータを“test-002.vtk”に、……といった具合に保存します。これをParaViewで開こうとすると、「test-..vtk」という一塊の連番ファイルとして認識されます。これを開くと全データを1つの時刻歴データとして開くことが出来ます。ビデオプレーヤーにある様なアイコン達をクリックすれば再生したり巻き戻したりの操作が可能です。

各時相の時刻を画面に表示するにはメニューバーからFilters→Temporal→Annotate Time Filterをクリックします。画面左側に現れたウィンドウのパラメータを調整すれば望みの時間が表示できます。その他にもFilters→Temporalの下にはアニメーション向けの様々なフィルター（ツール）が用意されているので試してみてください。

7　複数ファイルの同時可視化

Legacy VTKファイルの不便な点の1つに、パートやグループという概念が無いことが挙げられます。沢山のパーツから成る物体であれば、指定したパーツのみを表示したいという状況が頻繁に現れますが、単一のvtkファイルではそれが実現できません。その様な場合にはパーツ毎に独立した複数のvtkファイルを用いると良いでしょう。

例えば、A,B,Cの3つのパーツから成る場合は“A.vtk”と“B.vtk”と“C.vtk”を別々に用意し、ParaViewで3つを同時に開きます。そうすればA,B,Cの3つが画面上に同時に表示されます。画面左側の各ファイル名の左にある目玉のマークをクリックすればパーツ毎に表示／非表示を切り替えられます。

この状態で何かの操作（コンター表示する量の変更など）を行うと、A,B,Cのいずれか1つのみにその操作が反映されます。多くの場合、この様な操作は全てのパーツに対して同時に行いたい事の方が多いため、このままでは何度も同じ操作をせねばならず面倒です。そんな時は、画面左側の全ファイル名を選択した状態で「Group Datasets」のアイコンをクリックします。すると一番下にGroupDatasets1という項目が現れます。これを選択した状態で何かの操作をすれば全てのパーツに一括して同じ操作を実行出来ます。

8　バイナリ形式のLegacy VTKとVTK XMLファイル出力

Legacy VTKファイルの別の不便な点の1つに、ファイルサイズが大きいことが挙げられます。VTK XMLファイルであればzlibによるデータ圧縮が可能になるのですが、フォーマットがやや難解で初学者には不向きです。Legacy VTKファイルのままファイルサイズの低減と読込速度の向上を図るにはバイナリ形式を用いることになります。

まずは先程の“sample-a.vtk”をバイナリ形式に変換したファイルをテキストエディタで開いた場合、どの様に見えるのかを示します。

```
# vtk DataFile Version 4.1
This is a comment line.
BINARY
DATASET UNSTRUCTURED_GRID

POINTS 4 float
この行は文字化けして見える

CELLS 1 5
```

```
この行は文字化けして見える
CELL_TYPES 1
この行は文字化けして見える

POINT_DATA 4
SCALARS Point_Scalar_Data float
LOOKUP_TABLE default
この行は文字化けして見える
VECTORS Point_Vector_Data float
この行は文字化けして見える

CELL_DATA 1
SCALARS Cell_Scalar_Data float
LOOKUP_TABLE default
この行は文字化けして見える
VECTORS Cell_Vector_Data float
この行は文字化けして見える
```

バイナリ形式と言っても全てがバイナリで記述される訳ではなく、アスキー形式で数字が羅列されていた行のみがバイナリで記述され、文字化けして見えることになります。注意が必要なのはバイナリのエンディアンがBig endianであるという点です。一般的なPCはLittle endianなので、自作プログラム等でバイナリ部分を出力する際はスワッピングを行ってから出力を行う必要があります。

さて、プログラムの初学者だと自作プログラムではアスキー形式で出すのが精一杯かと思います。その様な場合は一旦アスキー形式で出力した後、別途バイナリ形式に変換するのが良いでしょう。非構造格子のLegacy VTKファイルをバイナリ形式に変換するPythonスクリプトを紹介しておきます(VTK 7.1.1で動作を確認)。

****

```
#!/usr/bin/env python2
import sys
import vtk

if len(sys.argv) != 3:
  print 'Usage:', sys.argv[0], 'inp.vtk out.vtk'
  sys.exit(1)

reader = vtk.vtkUnstructuredGridReader()
reader.SetFileName(sys.argv[1])
reader.ReadAllScalarsOn()
reader.ReadAllVectorsOn()
reader.ReadAllTensorsOn()
reader.Update()

writer = vtk.vtkUnstructuredGridWriter()
writer.SetFileName(sys.argv[2])
writer.SetInputData(reader.GetOutput())
writer.SetFileTypeToBinary()
writer.Write()
```

****

最後に、おまけとして、非構造格子のLegacy VTKファイルをVTK XMLのVTUファイル形式に変換するPythonスクリプトも紹介しておきます。

****

```
#!/usr/bin/env python2
import sys
import vtk

if len(sys.argv) != 3:
  print 'Usage:', sys.argv[0], 'inp.vtk out.vtu'
  sys.exit(1)

reader = vtk.vtkUnstructuredGridReader()
reader.SetFileName(sys.argv[1])
reader.ReadAllScalarsOn()
reader.ReadAllVectorsOn()
reader.ReadAllTensorsOn()
reader.Update()

writer = vtk.vtkXMLUnstructuredGridWriter()
writer.SetFileName(sys.argv[2])
writer.SetInputData(reader.GetOutput())
writer.Write()
```

****

■参考文献

[1] Utkarsh Ayachit, “The ParaView Guide”, https://www.paraview.org/download/.

[2] Kenneth Moreland, “The ParaView Tutorial”, https://www.paraview.org/download/.

[3] “ParaView Getting Started Guide”, https://www.paraview.org/download/.

[4] “VTK File Formats”, http://www.vtk.org/VTK/img/file-formats.pdf.

[5] 林真, “はじめてのParaView改訂版”, 工学社.

[6] 大嶋拓也, “ParaView導入ガイド”, 日本音響学会誌, 69(8), 407-412.

[7] Python Examples, https://lorensen.github.io/VTKExamples/site/Python/.