# FrontISTR のビルド
# 虎の巻
# (Ubuntu/Metis4 の書)

海洋研究開発機構 地球情報基盤センター

小川 道夫

## はじめに

FrontISTR は非線形構造解析機能が充実したオープンソースの構造解析ソフトウェアです。

大規模並列 FEM 基盤ミドルウェア上に構築され、先進性と実用性を兼ね備えています。

京・地球シミュレータ・FX10 などのスーパーコンピュータや各種クラウドサービスから身近にあるパソコンまでのスケーラビリティを備え、並列環境をあまり意識しないシンプルで使いやすい解析手順が提供されています。

ソースコードは公開され、ドキュメントも充実しているため必要であれば新たに機能を実装し、独自のニーズに対応することもできます。

以下では FrontISTR のビルド手順を説明します。

- Ubuntu-15.10 (64 ビット版)環境へのインストール方法を説明します。
- fistr1、fistr2 をビルドします。
- FrontISTR に含まれるインストールマニュアルの補助資料としてご利用ください。
- 作成する FrontISTR は MPI、OpenMP、パーティショナ、リファイナー、カプラー Metis、LAPACK、MUMPS、ML を有効にします。

## ソフトウェアのダウンロード

以下のリストを参照して、構築に必要なソフトウェアをダウンロードして下さい。

| ソフトウェア名 | ダウンロードサイト | ダウンロードするもの |
|---|---|---|
| OpenBLAS | http://www.openblas.net/ | OpenBLAS-0.2.15.tar.gz |
| METIS | http://glaros.dtc.umn.edu/gkhome/metis/metis/download | metis-5.1.0.tar.gz |
| ScaLAPACK | http://www.netlib.org/scalapack/ | scalapack-2.0.2.tgz |
| MUMPS | http://mumps.enseeiht.fr/<br>※要ユーザ登録 | MUMPS_5.0.1.tar.gz |
| ML(Trilinos) | https://trilinos.org/<br>※要ユーザ登録 | trilinos-12.4.2-Source.tar.bz2 |
| REVOCAP_Refiner | http://www.multi.k.u-tokyo.ac.jp/FrontISTR/reservoir_f/revisions.php<br>FrontISTR リザーバ内 ※要ユーザ登録 | REVOCAP_Refiner-1.1.03.tar.gz |
| FrontISTR | http://www.multi.k.u-tokyo.ac.jp/FrontISTR/reservoir_f/revisions.php<br>FrontISTR リザーバ内 ※要ユーザ登録 | FrontISTR_V44.tar.gz |
| FrontISTR 用パッチ | 文中参照 | fix_omp_for_gfortran.patch |

# ビルド環境の構築

初めにビルドに必要なパッケージをインストールします。

```
(MINGW64)$ cd $HOME
(MINGW64)$ mkdir Software
(MINGW64)$ sudo apt-get install build-essential gfortran cmake cmake-gui
(MINGW64)$ sudo apt-get install libopenmpi-dev openmpi-bin
(MINGW64)$ sudo apt-get install libboost-all-dev
```

これで準備は完了です。

# OpenBLAS のビルド

Ubuntu では apt-get でインストール出来るバイナリパッケージの OpenBLAS が用意されていますが、OpenMP を有効にしたライブラリを作成するため、今回はソースからビルドします。

```
$ cd $HOME/Software
$ tar xvf OpenBLAS-0.2.15.tar.gz
$ cd OpenBLAS-0.2.15
$ make BINARY=64 NO_SHARED=1 USE_OPENMP=1
$ make PREFIX=$HOME/Software install
```

以上でライブラリとヘッダファイルがインストールされます。

他のマシンで実行させる場合、その CPU に合わせた TARGET を指定する必要があるかもしれません。

TargetList.txt にサポートされている CPU の一覧からキーワードを探し、動作させるマシンの TARGET を指定してください。

```
Nehalem なら
$ make BINARY=64 NO_SHARED=1 USE_OPENMP=1 USE_THREAD=1 TARGET=NEHALEM
Sandy Bridge なら
$ make BINARY=64 NO_SHARED=1 USE_OPENMP=1 USE_THREAD=1 TARGET=SANDYBRIDGE
```

新しい CPU を TARGET に指定したライブラリは、古い CPU 上では正常に動かない可能性があります。

また、他の CPU の最適化に対応したライブラリを作成する事も出来ます。

```
$ make BINARY=64 NO_SHARED=1 USE_OPENMP=1 USE_THREAD=1 DYNAMIC_ARCH=1
```

DYNAMIC_ARCH=1 を指定した場合、ビルドにかかる時間が若干増え、出来上がったライブラリのサイズも大きくなります。

また、上記のオプションでビルドしたライブラリには LAPACK も含まれます。LAPACK をリンクしたい場合、-llapack の代わりに-lopenblas を指定します。

## METIS のビルド

古い metis-4.0.3 をビルドします。

```
$ cd $HOME/Software
$ tar xvf metis-4.0.3.tar.gz
$ metis-4.0.3
```

ファイルの修正は特に必要ありません。

```
$ make
```

以上で METIS のビルドは終了です。

## ScaLAPACK のビルド

scalapack-2.0.2 をビルドします。

```
$ cd $HOME/Software
$ tar xvf scalapack-2.0.2.tgz
$ cd scalapack-2.0.2
```

サンプルの SLmake.inc.example をコピーし、環境に合わせた内容に書き換えます。

```
$ cp SLmake.inc.example SLmake.inc
$ vi SLmake.inc
FC              = gfortran -fopenmp
CC              = gcc -fopenmp
FCFLAGS         = -O3 -I/usr/lib/openmpi/include
CCFLAGS         = -O3 -I/usr/lib/openmpi/include
FCLOADER        = $(FC) -lmpi -lmpi_f90
CCLOADER        = $(CC) -lmpi

BLASLIB         = -L$(HOME)/Software/lib -lopenblas
LAPACKLIB       =
※変更点のみ記載
```

変更が済んだらビルドをします。

```
$ make
```

以上で ScaLAPACK のビルドは完了です。

## MUMPS のビルド

MUMPS_5.0.1 をビルドします。

```
$ cd $HOME/Software
$ tar xvf MUMPS_5.0.1.tar.gz
$ cd MUMPS_5.0.1
```

テンプレート Makefile.inc.generic を Makefile.inc としてコピーし、環境に合わせた内容に書き換えます。

```
$ cp Make.inc/Makefile.inc.generic Makefile.inc
$ vi Makefile.inc
LMETISDIR = $(HOME)/Software/metis-4.0.3
IMETIS = -I$(LMETISDIR)/Lib
LMETIS = -L$(LMETISDIR) -lmetis

ORDERINGSF = -Dmetis4 -Dpord
```

```
CC = gcc -fopenmp
FC = gfortran -fopenmp
FL = gfortran -fopenmp

SCALAP = -L$(HOME)/Software/scalapack-2.0.2 -lscalapack

INCPAR = -I/usr/lib/openmpi/include
LIBPAR = $(SCALAP) -lmpi

LIBBLAS = -L$(HOME)/Software/lib -lopenblas
LIBOTHERS = -lpthread -fopenmp

OPTF    = -O -DMUMPS_OPENMP
OPTC    = -O -I. -DMUMPS_OPENMP
OPTL    = -O
```

※変更点のみ記載

変更が済んだらビルドします。

```
(MINGW64)$ make
```

以上で MUMPS のビルドは完了です。

## ML(Trilinos)のビルド

trilinos-12.4.2 をビルドします。FrontISTR では ML を用いますが、zoltan も一緒にビルドします。

```
$ cd $HOME/Software
$ tar xvf trilinos-12.4.2-Source.tar.bz2
$ cd trilinos-12.4.2-Source
$ mkdir build
```

ファイルを展開したら cmake、make をして下さい。

```
$ cd build
$ cmake \
-DCMAKE_INSTALL_PREFIX=$HOME/Software/trilinos \
-DCMAKE_CXX_FLAGS="-DNO_TIMES" \
-DCMAKE_C_FLAGS="-DNO_TIMES" \
-DTrilinos_ENABLE_ML=ON -DTrilinos_ENABLE_Zoltan=ON -DTrilinos_ENABLE_OpenMP=ON \
```

```
-DTrilinos_ENABLE_Kokkos=OFF -DTrilinos_ENABLE_Teuchos=OFF \
-DTrilinos_ENABLE_AztecOO=OFF -DTrilinos_ENABLE_Epetra=OFF \
-DTPL_BLAS_LIBRARIES=$HOME/Software/lib/libopenblas.a \
-DTPL_LAPACK_LIBRARIES=$HOME/Software/lib/libopenblas.a \
-DML_ENABLE_MPI=ON \
-DTPL_ENABLE_MPI=ON \
..
$ make
$ make install
```

以上で ML(Trilinos)のビルドは完了です。

## REVOCAP_Refiner のビルド

REVOCAP_Refiner-1.1.03 をビルドします。

```
$ cd $HOME/Software
$ tar xvf REVOCAP_Refiner-1.1.03.tar.gz
$ cd REVOCAP_Refiner-1.1.03
```

更に足りない記述を追加します。

```
$ vi MeshDB/kmbMeshOperation.h
#include <cstdlib>
#include <cstring>
を追加
$ vi MeshDB/kmbMeshBrep.h
#include <cstdlib>
を追加
```

変更が済んだらビルドします。

```
$ make
```

以上で REVOCAP_Refiner のビルドは完了です。

## FrontISTR のビルド

FrontISTR_V44.tar.gz の中には、Version4.6(fistr2)と Version3.6(fistr1)が同梱されています。今回は Version3.6(fistr1)のみビルドします。

FrontISTR はコンパイル時に有効にする機能が多数あります。今回は

- MPI
- OpenMP
- パーティショナ等のツール類
- リファイナー
- METIS
- MUMPS
- LAPACK
- ML

を有効にします。

FrontISTR_V44 をビルドします。

```
$ cd $HOME/Software
$ tar xvf FrontISTR_V44.tar.gz
$ cd FrontISTR_V44
```

## OpenMP 用パッチ(fix_omp_gfortran.patch)

OpenMP に対応した FrontISTR を gfortran でビルドするためのパッチを適用します。

このパッチは、合同会社 PExProCS(http://www.pexprocs.jp)の後藤様より提供して頂きました。

fix_omp_gfortran.patch

```
diff --git a/hecmw1/src/solver/matrix/hecmw_mat_ass.f90
b/hecmw1/src/solver/matrix/hecmw_mat_ass.f90
index e22b422..363ca2f 100644
--- a/hecmw1/src/solver/matrix/hecmw_mat_ass.f90
+++ b/hecmw1/src/solver/matrix/hecmw_mat_ass.f90
@@ -302,9 +302,9 @@ module hecmw_matrix_ass
subroutine hecmw_mat_ass_bc(hecMAT, inode, idof, RHS, conMAT)
type (hecmwST_matrix)      :: hecMAT
integer(kind=kint) :: inode, idof
-       real(kind=kreal) :: RHS
+       real(kind=kreal) :: RHS, val
type (hecmwST_matrix),optional     :: conMAT
```

```
-       integer(kind=kint) :: NDOF, in, i, ii, iii, ndof2, k, iS, iE, iiS, iiE, ik
+       integer(kind=kint) :: NDOF, in, i, ii, iii, ndof2, k, iS, iE, iiS, iiE, ik, idx

NDOF = hecMAT%NDOF
if( NDOF < idof ) return
@@ -318,13 +318,14 @@ module hecmw_matrix_ass

DO i = NDOF-1,0,-1
IF( i .NE. NDOF-idof ) THEN
+           idx = NDOF*inode-i
+           val = hecMAT%D(ndof2*inode-ii)*RHS
!$omp atomic
-           hecMAT%B(NDOF*inode-i) = hecMAT%B(NDOF*inode-i)         &
-                                  - hecMAT%D(ndof2*inode-ii)*RHS
+           hecMAT%B(idx) = hecMAT%B(idx) - val
if(present(conMAT)) then
+             val = conMAT%D(ndof2*inode-ii)*RHS
!$omp atomic
-             conMAT%B(NDOF*inode-i) = conMAT%B(NDOF*inode-i)         &
-                                    - conMAT%D(ndof2*inode-ii)*RHS
+             conMAT%B(idx) = conMAT%B(idx) - val
endif
ENDIF
ii = ii - NDOF
@@ -373,14 +374,15 @@ module hecmw_matrix_ass
if (hecMAT%itemU(ik) .eq. inode) then
iii = ndof2 - idof
DO i = NDOF-1,0,-1
+               idx = NDOF*in-i
+               val = hecMAT%AU(ndof2*ik-iii)*RHS
!$omp atomic
-               hecMAT%B(NDOF*in-i) = hecMAT%B(NDOF*in-i)       &
-                                   - hecMAT%AU(ndof2*ik-iii)*RHS
+               hecMAT%B(idx) = hecMAT%B(idx) - val
hecMAT%AU(ndof2*ik-iii)= 0.d0
if(present(conMAT)) then
+                 val = conMAT%AU(ndof2*ik-iii)*RHS
!$omp atomic
-                 conMAT%B(NDOF*in-i) = conMAT%B(NDOF*in-i)       &
-                                     - conMAT%AU(ndof2*ik-iii)*RHS
```

```
+                conMAT%B(idx) = conMAT%B(idx) - val
conMAT%AU(ndof2*ik-iii)= 0.d0
endif
iii = iii - NDOF
@@ -412,14 +414,15 @@ module hecmw_matrix_ass
iii  = ndof2 - idof

DO i = NDOF-1, 0, -1
+               idx = NDOF*in-i
+               val = hecMAT%AL(ndof2*ik-iii)*RHS
!$omp atomic
-               hecMAT%B(NDOF*in-i) = hecMAT%B(NDOF*in-i)      &
-                                   - hecMAT%AL(ndof2*ik-iii)*RHS
+               hecMAT%B(idx) = hecMAT%B(idx) - val
hecMAT%AL(ndof2*ik-iii) = 0.d0
if(present(conMAT)) then
+                 val = conMAT%AL(ndof2*ik-iii)*RHS
!$omp atomic
-                 conMAT%B(NDOF*in-i) = conMAT%B(NDOF*in-i)      &
-                                     - conMAT%AL(ndof2*ik-iii)*RHS
+                 conMAT%B(idx) = conMAT%B(idx) - val
conMAT%AL(ndof2*ik-iii) = 0.d0
endif
iii = iii - NDOF
```

このパッチファイルを適用します。

```
$ cd $HOME/Software
$ vi fix_omp_for_gfortran.txt
上記のパッチをファイルにする
$ cd FrontISTR_V44
$ patch -p1 < ../fix_omp_for_gfortran.txt
```

これで gfortran でも OpenMP 対応の FrontISTR をビルドできるようになりました。

## ファイルの編集

テンプレート Makefile.conf.org を Makefile.conf としてコピーし、環境に合わせた内容に書き換えます。

```
$ cp Makefile.conf.org Makefile.conf
$ vi Makefile.conf
```

```
################################################
#                                              #
#     Setup Configulation File for FrontISTR     #
#                                              #
################################################

# MPI
MPIDIR         = /usr
MPIBINDIR      = $(MPIDIR)/bin
MPILIBDIR      = $(MPIDIR)/lib
MPIINCDIR      = $(MPIDIR)/lib/openmpi/include
MPILIBS        = -lmpi -lmpi_f90 -lmpi_cxx

# for install option only
PREFIX         = $(HOME)/FrontISTR
BINDIR         = $(PREFIX)/bin
LIBDIR         = $(PREFIX)/lib
INCLUDEDIR     = $(PREFIX)/include

# Metis
METISDIR       = $(HOME)/Software/metis-4.0.3
METISLIBDIR    = $(METISDIR)
METISINCDIR    = $(METISDIR)/Lib

# ParMetis
PARMETISDIR    = $(HOME)/ParMetis-3.1
PARMETISLIBDIR = $(PARMETISDIR)
PARMETISINCDIR = $(PARMETISDIR)/ParMETISLib

# Refiner
REFINERDIR     = $(HOME)/Software/REVOCAP_Refiner-1.1.03
REFINERINCDIR  = $(REFINERDIR)/Refiner
REFINERLIBDIR  = $(REFINERDIR)/lib/x86_64-linux

# Coupler
REVOCAPDIR     = $(HOME)/Software/REVOCAP_Coupler-2.1
REVOCAPINCDIR  = $(REVOCAPDIR)/librcap
REVOCAPLIBDIR  = $(REVOCAPDIR)/librcap

# MUMPS
```

```
MUMPSDIR        = $(HOME)/Software/MUMPS_5.0.1
MUMPSINCDIR     = $(MUMPSDIR)/include
MUMPSLIBDIR     = $(MUMPSDIR)/lib
MUMPSLIBS       = -ldmumps -lmumps_common -lpord -L$(HOME)/Software/scalapack-2.0.2/ -
lscalapack


# ML
MLDIR           = $(HOME)/Software/trilinos
MLINCDIR        = $(MLDIR)/include
MLLIBDIR        = $(MLDIR)/lib


# C compiler settings
CC              = mpicc -fopenmp
CFLAGS          =
LDFLAGS         = -L$(HOME)/Software/lib -lopenblas -lm -lstdc++
OPTFLAGS        = -O3


# C++ compiler settings
CPP             = mpic++ -fopenmp
CPPFLAGS        =
CPPLDFLAGS      = -L$(HOME)/Software/lib -lopenblas
CPPOPTFLAGS     = -O3


# Fortran compiler settings
F90             = mpif90 -fno-range-check -fopenmp
F90FLAGS        =
F90LDFLAGS      = -L$(HOME)/Software/lib -lopenblas -lstdc++
F90OPTFLAGS     = -O2
F90LINKER       = mpif90 -fopenmp


MAKE            = make
AR              = ar ruv
CP              = cp -f
RM              = rm -f
MKDIR           = mkdir -p
```

gfortran でコンパイルする時に問題になる箇所を修正します。

```
$ vi fistr2/src/analysis/dynamic/transit/dynamic_output.f90
subroutine dynamic_nodal_stress_2d
      real(kind=KREAL) :: s11, s22, s33, s12, s23, s13, ps, smises
の下に
```

```
      integer :: tmp1_ielem, tmp2_ielem, tmp3_ielem, tmp4_ielem
```

宣言を追加。

```
            fstrSOLID%ESTRAIN(6*ielem-5:6*ielem) = estrain
            fstrSOLID%ESTRESS(7*ielem-6:7*ielem-1) = estress
```

を

```
            tmp1_ielem = 6*ielem-5
            tmp2_ielem = 6*ielem
            fstrSOLID%ESTRAIN(tmp1_ielem:tmp2_ielem) = estrain
            tmp3_ielem = 7*ielem-6
            tmp4_ielem = 7*ielem-1
            fstrSOLID%ESTRESS(tmp3_ielem:tmp4_ielem) = estress
```

と変更

## ビルド

FrontISTR をビルドします。

```
$ ./setup.sh -p --with-tools --with-refiner \
               --with-lapack --with-metis --with-mumps --with-ml
$ make
$ make install
```

$HOME/FrontISTR/bin にソルバー(fistr1, fistr2)やパーティショナ(hecmw_part1, hecmw_part2)がインストールされます。

```
$ cd $HOME/FrontISTR/bin
$ ls
conv2mw3  fistr2    hecmw_part1  hecmw_vis1  neu2fstr  rmerge
fistr1    hec2rcap  hecmw_part2  hecmw_vis2  rconv
```

以上で FrontISTR のビルドは完了です。

## テスト

チュートリアルデータを用いて解析をしてみます。

先ほどのディレクトリにパスを通してから、FrontISTR のチュートリアルデータがある場所に移動して下さい。

```
$ export PATH=$HOME/FrontISTR/bin:$PATH
$ cd $HOME/Software/FrontISTR_V44/tutorial
```

```
$ ls
01_elastic_hinge              02_elastic_hinge_parallel
02_elastic_hinge_parallel_02 03_hyperelastic_cylinder
04_hyperelastic_spring       05_plastic_cylinder
06_plastic_can               07_viscoelastic_cylinder
08_creep_cylinder            09_contact_hertz
10_contact_2tubes            11_contact_2beam
12_dynamic_beam              13_dynamic_beam_nonlinear
14_dynamic_plate_contact     15_eigen_spring
16_heat_block                17_freq_beam
```

この中の 01_elastic_hinge と 02_elastic_hinge_parallel を実行してみて下さい。

```
$ cd 01_elastic_hinge
$ ls
hecmw_ctrl.dat  hinge.cnt  hinge.msh
```

正しくパスが通っていれば fistr1 と打つだけで解析が始まります。

```
$ fistr1
Step control not defined! Using default step=1
 fstr_setup: OK
### 3x3 B-SSOR-CG(0)  1
      1    1.903375E+00
      2    1.974378E+00
...
...
   2967    1.072387E-08
   2968    9.994170E-09
### Relative residual = 1.02411E-08

### summary of linear solver
      2968 iterations      9.994170E-09
    set-up time      :     2.124152E-01
    solver time      :     7.884661E+01
    solver/comm time :     4.373577E-02
    solver/matvec    :     3.441607E+01
    solver/precond   :     3.714435E+01
    solver/1 iter    :     2.656557E-02
    work ratio (%)   :     9.994453E+01

Start visualize PSF 1 at timestep 1
```

```
 ====================================
    TOTAL TIME (sec) :     82.02
           pre (sec) :      0.85
         solve (sec) :     81.17
 ====================================
 FrontISTR Completed !!
$
```

何も指定していなければ、コンピュータに搭載されたコア数の OpenMP スレッドで解析をします。top コマンドなどで確認してみてください。

スレッド数を指定する場合は、OMP_NUM_THREADS という環境変数を設定します。

```
$export OMP_NUM_THREADS=4
echo $OMP_NUM_THREADS
4
$ fistr1
```

Hyper-threading が ON になってと実際の倍のコアが有るように見えますが、スレッドの数は実際のコア数に指定するのが一番効率良いようです。

次に MPI の実行例を示します。最初に領域分割をします。

```
$ cd ..\02_elastic_hinge_parallel
$ hecmw_part1
Dec 04 17:38:31 Info: Reading mesh file...
Dec 04 17:38:31 Info: Starting domain decomposition...
Dec 04 17:38:31 Info: Creating local mesh for domain #0 ...
Dec 04 17:38:32 Info: Creating local mesh for domain #1 ...
Dec 04 17:38:32 Info: Creating local mesh for domain #2 ...
Dec 04 17:38:32 Info: Creating local mesh for domain #3 ...
Dec 04 17:38:32 Info: Domain decomposition done
>
```

MPI でソルバを実行するときは、OpenMP スレッドの数を 1 つにした方が良いようです(環境によります)。

実行には mpirun を用います。

```
$ export OMP_NUM_THREADS=1
$ mpirun -np 4 fistr1
 Step control not defined! Using default step=1
 Step control not defined! Using default step=1
```

```
 Step control not defined! Using default step=1
 Step control not defined! Using default step=1
 fstr_setup: OK
 fstr_setup: OK
 fstr_setup: OK
 fstr_setup: OK
### 3x3 B-SSOR-CG(0)  2
      1    2.179037E+00
      2    2.412720E+00
...
...
   2084    1.003548E-08
   2085    9.112093E-09
### Relative residual = 9.29985E-09

### summary of linear solver
      2085 iterations      9.112093E-09
    set-up time      :     1.456044E-02
    solver time      :     9.929363E+01
    solver/comm time :     2.988770E+00
    solver/matvec    :     4.605539E+01
    solver/precond   :     4.684918E+01
    solver/1 iter    :     4.762284E-02
    work ratio (%)   :     9.698997E+01


 ====================================
    TOTAL TIME (sec) :    101.14
           pre (sec) :      0.32
         solve (sec) :    100.82
 ====================================
 FrontISTR Completed !!
Start visualize PSF 1 at timestep 1

$
```

ディレクトリ内を見ると結果が出力されているのが分かります。

```
$ ls
0.log       FSTR.dbg.3           hinge.cnt      hinge_4.1
1.log       FSTR.msg             hinge.msh      hinge_4.2
```

```
2.log         FSTR.sta              hinge.res.0.1   hinge_4.3
3.log         hecmw_ctrl.dat        hinge.res.1.1   hinge_vis_psf.0001.inp
FSTR.dbg.0    hecmw_part.log        hinge.res.2.1   part.inp
FSTR.dbg.1    hecmw_part_ctrl.dat   hinge.res.3.1
FSTR.dbg.2    hecmw_vis.ini         hinge_4.0
$
```

hinge.cnt 内の!output_type を下の様に書き換えて下さい。

```
!output_type = COMPLETE_AVS
```

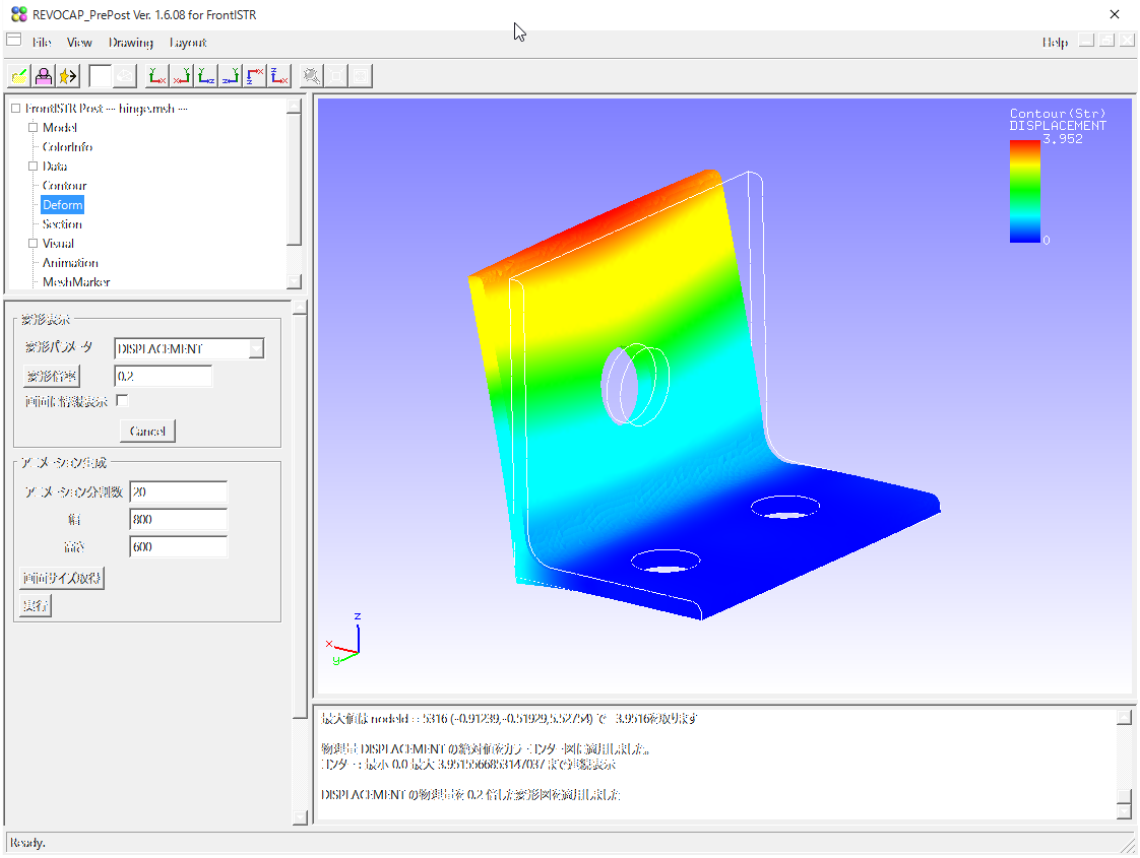
を

```
!output_type = COMPLETE_REORDER_AVS
```

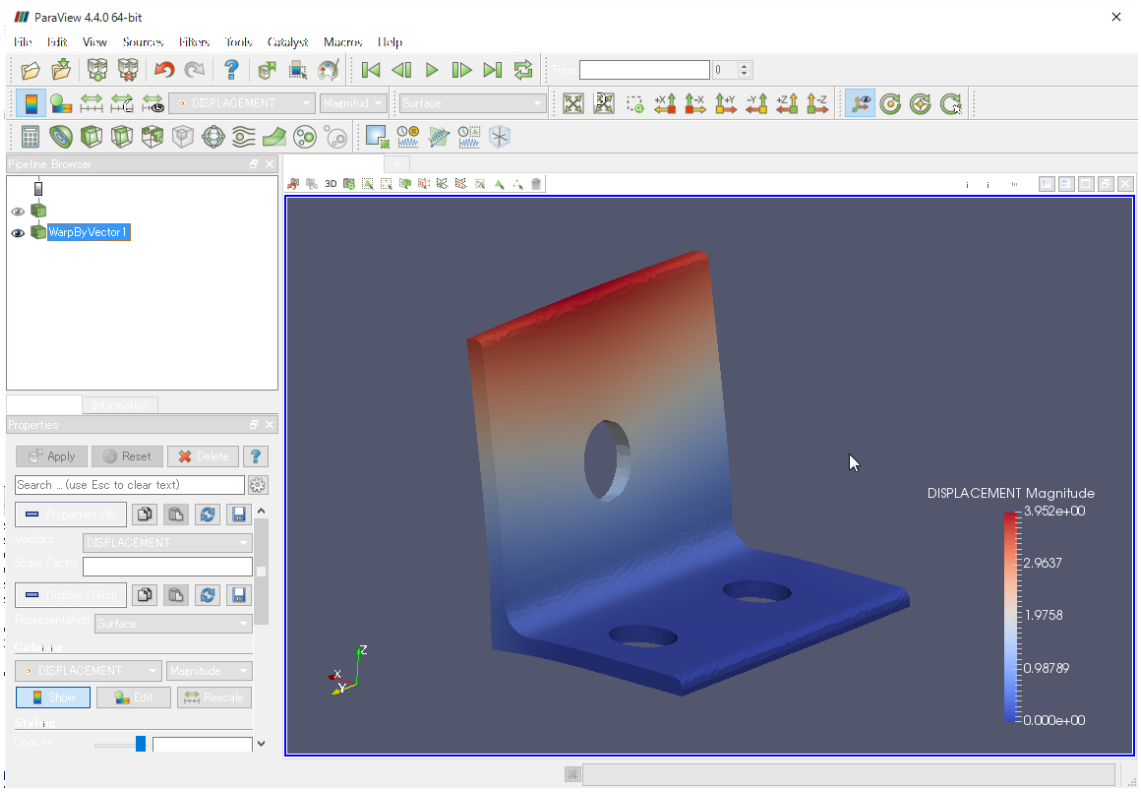
変更後 fistr1 を実行すると、解析結果を ParaView からも見ることが出来ます。

REVOCAP_PrePost はどちらの形式にも対応しているので、変更の必要はありません。

REVOCAP_PrePost での可視化結果は下の図のようになります。




ParaView ではこのように表示されます。


ParaView 4.4.0 64-bit
File Edit View Sources Filters Tools Catalyst Macros Help
Pipeline Browser
WarpByVector1
Properties
Apply
Reset
Delete
Search ... (use Esc to clear text)
DISPLACEMENT
Surface
Show
Edit
Rescale
DISPLACEMENT Magnitude
3.952e+00
2.9637
1.9758
0.98789
0.000e+00