

FrontISTR のビルド 虎の巻 (Windows/シリアルの本)

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ルビリティを備え、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。

- Windows (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 はパーティショナ、リファイナー、Metis、LAPACK、MUMPS 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。

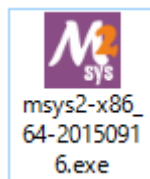
ソフトウェア名	ダウンロードサイト	ダウンロードするもの
MSYS2	https://msys2.github.io	msys2-x86_64-20150916.exe
OpenBLAS	http://www.openblas.net/	OpenBLAS-0.2.15.tar.gz
METIS	http://glaros.dtc.umn.edu/gkhome/metis/metis/download	metis-5.1.0.tar.gz
MUMPS	http://mumps.enseiht.fr/ ※要ユーザ登録	MUMPS_5.0.1.tar.gz
REVOCAP_Refiner	http://www.multi.k.u-tokyo.ac.jp/FrontISTR/reservoir_f/visions.php 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/visions.php 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

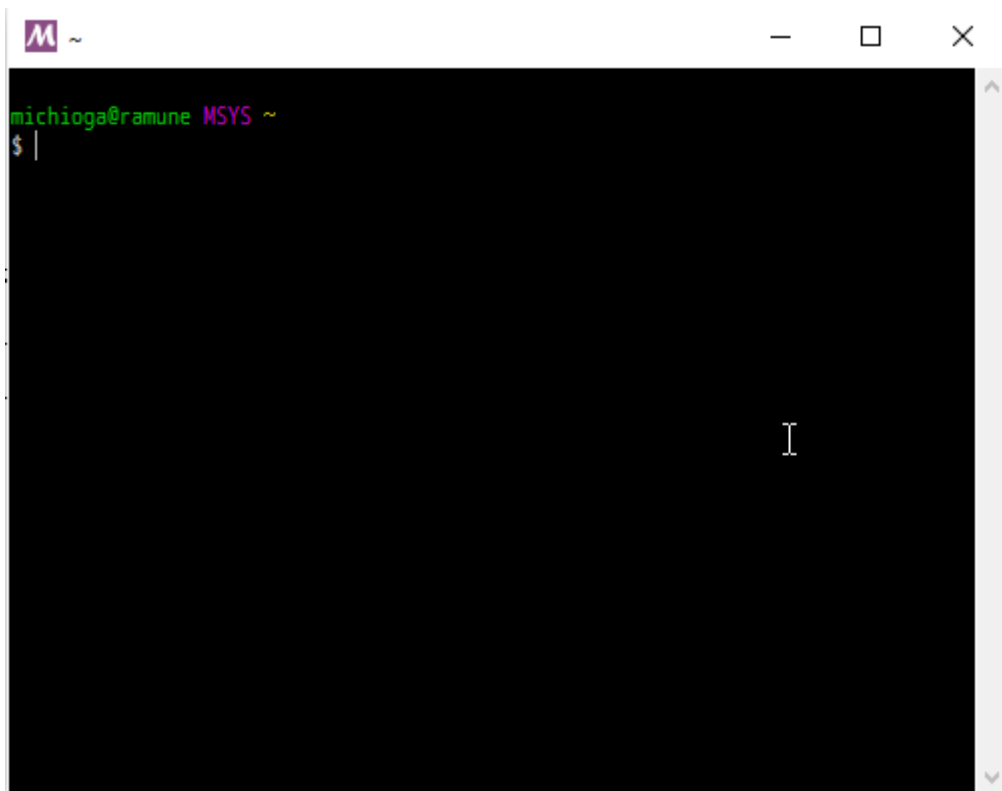
MSYS2 を用いて MinGW 64 ビット版の環境を構築します。

Unix 系のソフトウェアを Windows 上でビルドする環境として Cygwin が有名ですが、ラ
ンタイムへの依存があるため今回は使いません。

ダウンロードしたインストーラ `msys2-x86_64-20150916.exe` をダブルクリックし、ガイ
ドに従ってインストールして下さい。



インストーラが終了すると「MSYS2 Shell」が実行されます。



MSYS2 は archlinux 由来の「pacman」というパッケージマネージャで管理されています。

pacman を使って環境を最新の状態にしてください。

```
(MSYS)$ pacman -Syu
```

```
michioga@ramune MSYS ~
$ pacman -Syu
:: パッケージデータベースの同期中...
mingw32                224.7 KiB  53.5K/s  00:04 [#####] 100%
mingw32.sig             96.0 B    0.00B/s  00:00 [#####] 100%
mingw64                223.6 KiB  51.3K/s  00:04 [#####] 100%
mingw64.sig             96.0 B    0.00B/s  00:00 [#####] 100%
msys                   128.1 KiB  62.5M/s  00:00 [#####] 100%
msys.sig                96.0 B    0.00B/s  00:00 [#####] 100%
:: システム全体の更新を開始...
依存関係を解決しています...
衝突するパッケージがないか確認しています...

パッケージ (12) bash-4.3.042-3  curl-7.45.0-1  flex-2.6.0-1  gcc-libs-4.9.2-6
                  gmp-6.1.0-1  grep-2.22-1  libcurl-7.45.0-1
                  libreadline-6.3.008-6  mintty-1~2.2.1-1
                  msys2-runtime-2.4.0.16752.6eb10ef-1  ncurses-6.0.20151121-1
                  pacman-4.2.1.6258.f5bbd79-1

合計ダウンロード容量:  11.56 MiB
合計インストール容量:  59.12 MiB
最終的なアップグレード容量:  -0.36 MiB

:: インストールを行いますか? [Y/n] y
```

パッケージのアップデートが完了したら、一度ウインドウを閉じて下さい。

再度「MSYS2 Shell」を起動し、その他のパッケージをインストールしていきます。

```
(MSYS)$ pacman -S base-devel mingw-w64-x86_64-toolchain mingw-w64-x86_64-cmake \
           mingw-w64-x86_64-extra-cmake-modules mingw-w64-x86_64-boost
```

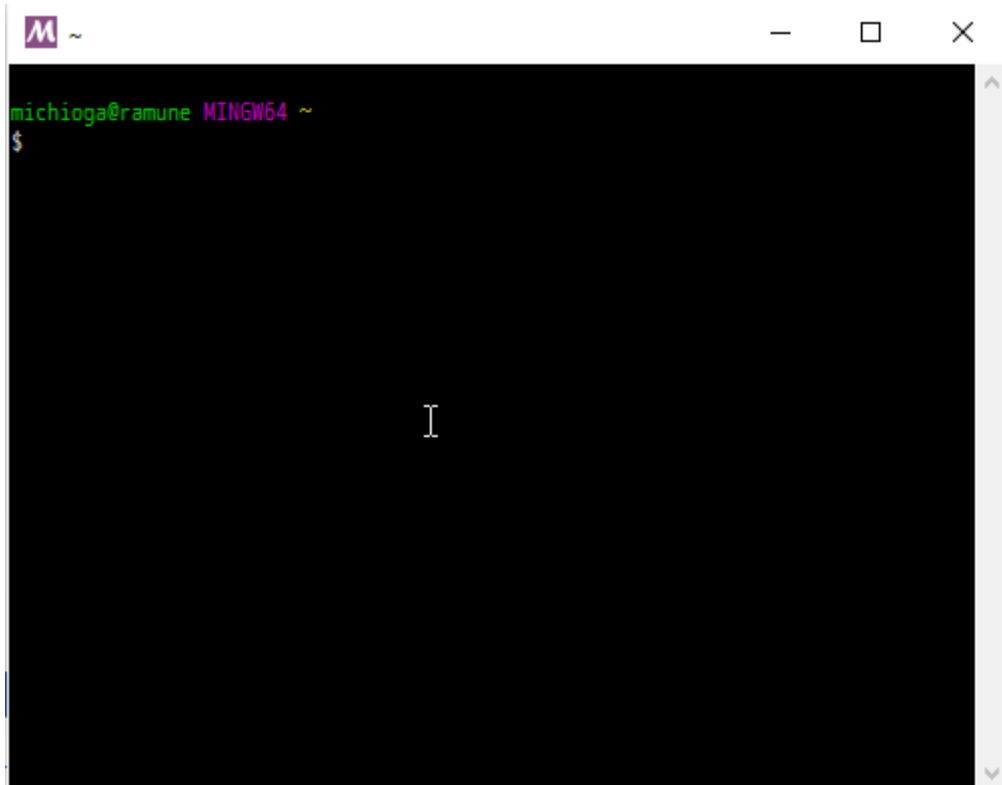
pacman を使い、構築済みのパッケージをインストールする事も出来ます。

```
キーワードでパッケージ名を検索
(MSYS)$ pacman -Ss <キーワード>

パッケージをインストール
(MSYS)$ pacman -S <パッケージ名>
```

以上でビルド環境の構築は完了しましたので、「MSYS2 Shell」を閉じて下さい。

これからの作業は「MinGW-w64 Win64」で行います。



また、ビルド作業は「`$HOME/Software`」で行います。

```
(MINGW64)$ cd $HOME
(MINGW64)$ mkdir Software
```

OpenBLAS のビルド

Windows 版 OpenBLAS はバイナリ配布がありますが、シングルスレッドのライブラリを作成するためソースからビルドします。

```
(MINGW64)$ cd $HOME/Software
(MINGW64)$ tar xvf OpenBLAS-0.2.15.tar.gz
(MINGW64)$ cd OpenBLAS-0.2.15
(MINGW64)$ make BINARY=64 NO_SHARED=1 USE_OPENMP=0 USE_THREAD=0
(MINGW64)$ 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
(MINGW64)$ make BINARY=64 NO_SHARED=1 USE_OPENMP=0 USE_THREAD=0 TARGET=NEHALEM
Sandy Bridge なら
(MINGW64)$ make BINARY=64 NO_SHARED=1 USE_OPENMP=0 USE_THREAD=0 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。

```
(MINGW64)$ make BINARY=64 NO_SHARED=1 USE_OPENMP=0 USE_THREAD=0 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。

```
(MINGW64)$ cd $HOME/Software
(MINGW64)$ tar xvf metis-4.0.3.tar.gz
(MINGW64)$ metis-4.0.3
```

Makefile.in を修正します。

```
(MINGW64)$ vi Makefile.in
CC = gcc
COPTIONS = -D__VC__
※変更点のみ記載
```

また、MinGW には srand48 と drand48 の関数がありませんので、代わりに srand と drand を使います(乱数の品質は下がります)。

以上の修正が済んだらビルドします。

```
(MINGW64)$ make
```

以上で METIS のビルドは終了です。

MUMPS のビルド

MUMPS_5.0.1 をビルドします。

```
(MINGW64)$ cd $HOME/Software
(MINGW64)$ tar xvf MUMPS_5.0.1.tar.gz
(MINGW64)$ cd MUMPS_5.0.1
```

テンプレート `Makefile.inc.generic.SEQ` 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。

```
(MINGW64)$ cp Makefile.inc/Makefile.inc.generic.SEQ Makefile.inc
(MINGW64)$ vi Makefile.inc
LMETISDIR = $(HOME)/Software/metis-4.0.3
IMETIS = -I$(LMETISDIR)/Lib
LMETIS = -L$(LMETISDIR) -lmetis

ORDERINGSF = -Dmetis4 -Dpord

CC = gcc
FC = gfortran -fno-range-check
FL = gfortran

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

OPTF = -O
OPTC = -O -I.
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。

REVOCAP_Refiner のビルド

REVOCAP_Refiner-1.1.03 をビルドします。

```
(MINGW64)$ cd $HOME/Software
(MINGW64)$ tar xvf REVOCAP_Refiner-1.1.03.tar.gz
```

```
(MINGW64)$ cd REVOCAP_Refiner-1.1.03
```

MakefileConfig.in を環境に合わせた内容に書き換えます。

```
(MINGW64)$ vi MakefileConfig.in
ARCH = x86_64-mingw-w64
CXXFLAGS = -O -Wall $(DEBUGFLAG)
※変更点のみ記載
```

更に足りない記述を kmbMeshOperation.cpp に追加します。

```
(MINGW64)$ $ 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。

```
(MINGW64)$ 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。

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

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

を有効にします。

FrontISTR_V44 をビルドします。


```
(MINGW64)$ cd $HOME/Software
(MINGW64)$ tar xvf FrontISTR_V44.tar.gz
(MINGW64)$ cd FrontISTR_V44
```

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。

```
(MINGW64)$ cp Makefile.conf.org Makefile.conf
(MINGW64)$ vi Makefile.conf
#####
#                                     #
#   Setup Configuration File for FrontISTR   #
#                                     #
#####

# MPI
MPIDIR      =
MPIBINDIR   =
MPILIBDIR   =
MPIINCDIR   =
MPILIBS     =

# 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-mingw-w64

# 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    = $(MUMPSDIR)/lib
MUMPSLIBS      = -ldmumps -lmumps_common -lpord -L$(MUMPSDIR)/libseq -lmpiseq

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

# C compiler settings
CC              = gcc
CFLAGS          = -D_WINDOWS
LDLDFLAGS      = -L$(HOME)/Software/lib -lopenblas -lm -lstdc++
OPTFLAGS       = -O3

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

# Fortran compiler settings
F90             = gfortran -fno-range-check
F90FLAGS        =
F90LDLDFLAGS   = -L$(HOME)/Software/lib -lopenblas -lstdc++
F90OPTFLAGS    = -O2
F90LINKER      = gfortran
```

```
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。

```
(MINGW64)$ vi fistr2/src/analysis/dynamic/transit/dynamic_output.f90
subroutine dynamic_nodal_stress_2d 内の
    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。

```
(MINGW64)$ ./setup.sh --with-tools --with-refiner \  
                --with-lapack --with-metis --with-mumps  
(MINGW64)$ make  
(MINGW64)$ make install
```

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

```
(MINGW64)$ cd $HOME/FrontISTR/bin  
(MINGW64)$ ls  
conv2mw3.exe  hec2rcap.exe      hecmw_vis1.exe  rconv.exe  
fistr1.exe    hecmw_part1.exe  hecmw_vis2.exe  rmerge.exe
```

```
fistr2.exe    hecmw_part2.exe  neu2fstr.exe
```

「MinGW-w64 Win64 Shell」内であれば、この状態で動作します。

このシェルを起動せずに実行するには、必要になる DLL をこのディレクトリコピーしておく便利です。

```
(MINGW64)$ mkdir /c/FrontISTR
(MINGW64)$ cp $HOME/FrontISTR/bin /c/FrontISTR
(MINGW64)$ cd /mingw64/bin
(MINGW64)$ cp libgcc_s_seh-1.dll libgomp-1.dll libstdc++-6.dll $HOME/FrontISTR/bin
(MINGW64)$ cp libgfortran-3.dll libquadmath-0.dll $HOME/FrontISTR/bin
(MINGW64)$ cp libwinpthread-1.dll $HOME/FrontISTR/bin
(MINGW64)$ cd /c/FrontISTR
(MINGW64)$ ls
fistr1.exe      libgcc_s_seh-1.dll  libstdc++-6.dll      rmerge.exe
hec2rcap.exe    libgfortran-3.dll   libwinpthread-1.dll
hecmw_part1.exe libgomp-1.dll       neu2fstr.exe
hecmw_vis1.exe  libquadmath-0.dll  rconv.exe
```

上の例では C: ドライブ直下に **FrontISTR** というディレクトリを作り、そこへ実行ファイルと DLL をコピーしました。

Windows の環境変数で C:\FrontISTR へのパスを加えておけば、MinGW をインストールしていない環境でも実行できます。

テスト

スタートメニューから「Windows システム ツール」⇒「コマンドプロンプト」を起動して下さい。

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。

```
C:>set %PATH%=C:\FrontISTR;%PATH%
C:>cd msys64\home<ログオン名>\Software\FrontISTR_V44\tutorial
C:>dir /w
ドライブ C のボリューム ラベルは Windows です
ボリューム シリアル番号は XXXX-XXXX です

C:\msys64\home<ログオン名>\Software\FrontISTR_V44\tutorial のディレクトリ
```

```
[.]                [...]
[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 を実行してみてください。

```
> cd 01_elastic_hinge
> dir
2015/12/04 17:19 <DIR>      .
2015/12/04 17:19 <DIR>      ..
2015/02/16 18:04                183 hecmw_ctrl.dat
2015/02/16 18:04                511 hinge.cnt
2015/02/16 18:04           9,266,446 hinge.msh
          3 個のファイル           9,267,140 バイト
          2 個のディレクトリ  xxx,xxx,xxx,xxx バイトの空き領域
```

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 !!
>
```

正常に終了すればテストは成功です。

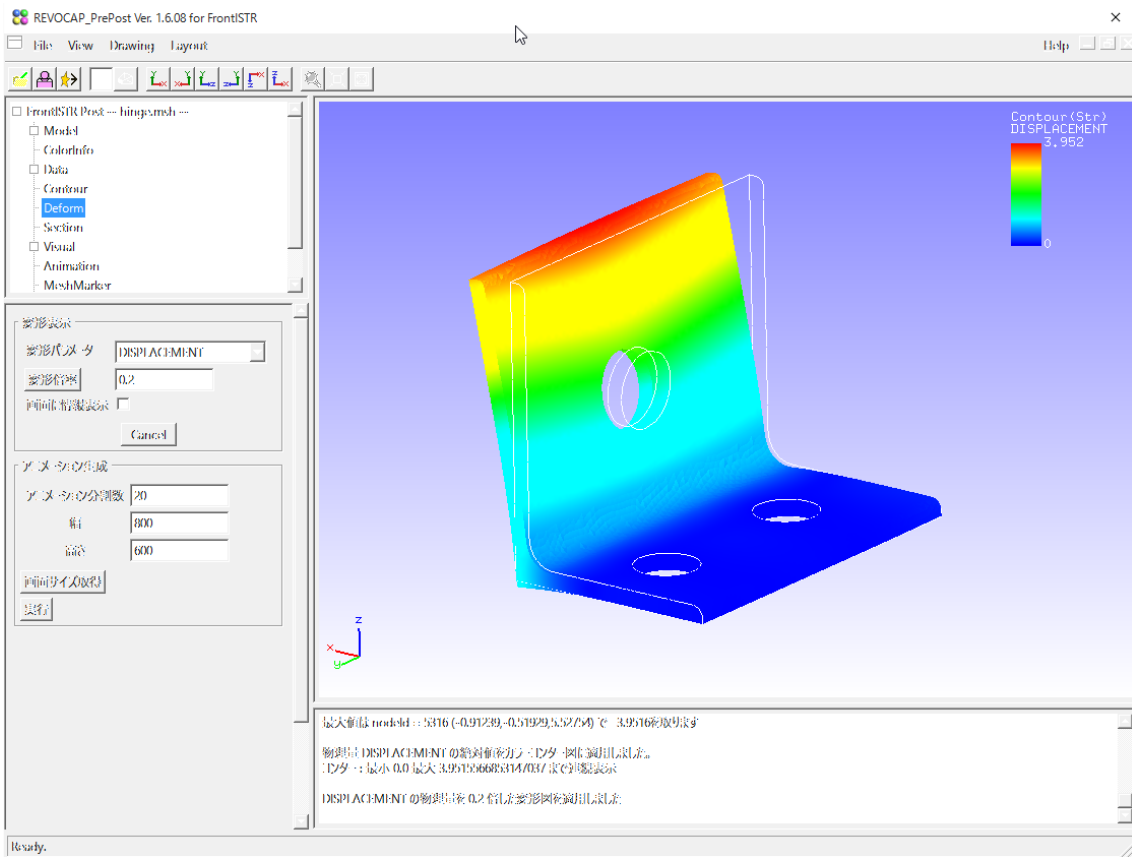
次に `hinge.cnt` 内の `!output_type` を下の様書き換えて下さい。

```
!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** から見る事が出来ます。

`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。

