Running PERIGEE with the Greenshields-Weller Benchmark

0. When getting a brand new machine, please go through steps 0. ~ 0.7.

We assume your system is Linux, such as Ubuntu. If you are a OS X user, there are some different steps which will not be introduced here.

Learn about basic terminal commands and vi commands.

In this guide, **filename** will be noted in bold. Operations in Terminal will be shown as follows:

```
$ (command1)
$ (command2)
```

We also assume you have a handy editor such as VsCode.

0.1. Install external libraries following this. And here are some tips：

MPICH:

If you have a hard life with error info about "Fortran compiler", and gfortran installation doesn't work, try to download another version of mpich:

```
$ wget https://www.mpich.org/static/downloads/3.4.2/mpich-3.4.2.tar.gz
```

then add a new configuration option:

```
$ ./configure --prefix=$HOME/lib/mpich-3.4.2 FFLAGS=-fallow-argument-mismatch 2>&1 |
tee c.txt
```

VTK:

We'd better install the OpenGL(Glut) library before installing VTK.

If you install VTK on a cluster such as Taiyi, do:

```
$ cmake $HOME/lib/VTK-7.1.1-src -DCMAKE_BUILD_TYPE:STRING=Release
-DCMAKE_INSTALL_PREFIX:PATH=$HOME/lib/VTK-7.1.1-OPT -DVTK_Group_StandAlone:BOOL=OFF
-DVTK_Group_Rendering:BOOL=OFF -DModule_vtkCommonMath:BOOL=ON
-DModule_vtkCommonMisc:BOOL=ON -DModule_vtkCommonCore:BOOL=ON
-DModule_vtkCommonSystem:BOOL=ON -DModule_vtkIOCore:BOOL=ON -DModule_vtkIOLegacy:BOOL=ON
-DModule_vtkIOXML:BOOL=ON
```

METIS:

Install metis before installing petsc, then you can use the option `"--with-metis-dir="` rather than `"--download-metis"` later.

PETSc:

We prefer another version now:

```
$ wget http://ftp.mcs.anl.gov/pub/petsc/release-snapshots/petsc-3.16.6.tar.gz
```

And an advanced installation configuration:

```
$ ./configure --with-x=0 -with-pic --with-make-np=6 --with-mpi-compilers=0
--with-mpi-dir=$HOME/lib/mpich-3.4.2/ --with-scalar-type=real --with-precision=double
```

```
--with-chaco=1 --download-chaco --with-hypre=1 --download-hypre --with-spai=1
--download-spai --with-sundials2=1 --download-sundials2 --with-mumps=1 --download-mumps
--with-scalapack=1 --download-scalapack --with-blacs=1 --download-blacs --with-spooles=1
--download-spooles --with-superlu_dist=1 --download-superlu_dist --with-superlu=1
--download-superlu --download-fblaslapack --with-metis-dir=$HOME/lib/metis-5.0.3
--with-ml=1 --download-ml --with-eigen=1 --download-eigen --with-debugging=no
COPTFLAGS="-O3 -march=native -mtune=native" CXXOPTFLAGS="-O3 -march=native -mtune=native"
FOPTFLAGS="-O3 -march=native -mtune=native" --prefix=$HOME/lib/petsc-3.16.6-opt
```

(It's a long command, please confirm your lib version, fit them where are highlighted.)

If you want to update your cmake to version 3.20, add an option:

```
....... --download-cmake ......
```

Similarly：

```
....... --download-make ......
```

If you have a hard life with error info about Fortran/mpich-dir/MPI etc. , change the highlighted options to:

```
...... --with-mpi-compilers=1 --with-cc=gcc --downnload-mpich ......
```

(Of course, you installed gcc in advance since you have been a C/C++ programmer.)

For the further learning of petsc, we suggest you doing:

```
$ ./configure --help
```

to have a look, then you can try your individual configuration, dealing with error info by yourself.

SLEPc:

The version of slepc should be suited to your petsc. If your petsc is version 3.16.6:

```
$ wget https://slepc.upv.es/download/distrib/slepc-3.16.3.tar.gz
```

Somtimes the environment variations created by temporary *export* commands cannot be detected, you should write them into **~/.bashrc** file:

```
$ sudo vi ~/.bashrc
```

(Move to the bottom, press 'a', write:

```
export PETSC_DIR=$HOME/lib/petsc-3.16.6-opt
export SLEPC_DIR=$HOME/lib/slepc-3.16.3-src
export PETSC_ARCH= [your arch](If your PETSC_ARCH is required, view thier website,
download a users manual to know what is PETSC_ARCH. Otherwise ignore this line.)
```

then press 'esc', input ':wq!', press 'enter' .)

```
$ source ~/.bashrc
```

If the document's name is **slepc-3.16.3-src**, you must add ***"-src"*** to the end of SLEPC_DIR. If it doesn't work either, write the variations into **/etc/environment** without *export* (It's a system-wide profile, take it carefully), then restart the machine.

Gmsh and ParaView:

View Gmsh's website and download a package for installation is not bad. Same for ParaView.

You can call them in Terminal to check your installation on linux(Mac users need alias):

```
$ gmsh
$ paraview
```

0.2. View this page, download **machine_name.cmake**, rename it appropriately:

Open it with an editor, replace these arguments according to your libraries' path and configuration:

```
# In the guide, vtk directory is $HOME/lib/VTK-7.1.1-shared
set(VTK_DIR /home/jliu/lib/VTK-7.1.1-shared/lib/cmake/vtk-7.1)

# Modify the PETSC_DIR variable to point to the location of PETSc.
set(PETSC_DIR /home/jliu/lib/petsc-3.11.3)

# Modify the PETSC_ARCH variable. You can find it in your configuration
# output. If you forget it, go to your PETSc home director and open
# configure.log. Go the end of the file, and you shall find the value
# of PETSC_ARCH
set(PETSC_ARCH arch-linux2-c-debug)

# Modify the METIS_DIR.
# Note: If your PETSc has METIS installed, the conf
# file will directly load that METIS; otherwise this METIS will
# be used for PERIGEE. This means, if you are sure that you have
# METIS in PETSc, you do not have to specify the METIS_DIR variable.
set(METIS_DIR /home/jliu/lib/metis-5.0.3)

# Modify the HDF5_ROOT, pointing to your hdf5 library location
set(HDF5_ROOT /home/jliu/lib/hdf5-1.8.16)

```

```
# $PETSC_DIR/$PETSC_ARCH/bin, or the mpich you specified for
# PETSc install.
set(CMAKE_C_COMPILER  /home/jliu/lib/petsc-3.11.3/bin/mpicc)
set(CMAKE_CXX_COMPILER /home/jliu/lib/petsc-3.11.3/bin/mpicxx)
set(CMAKE_CXX_STANDARD 11)
if( ${CMAKE_BUILD_TYPE} MATCHES "Release" )
```

then save it.

DO NOT use "$HOME" to replace "/home/xxx/" as follows:

```
# In the guide, vtk directory is $HOME/lib/VTK-7.1.1-shared
set(VTK_DIR $HOME/lib/VTK-7.1.1-shared/lib/cmake/vtk-7.1)

# Modify the PETSC_DIR variable to point to the location of PETSc.
set(PETSC_DIR $HOME/lib/petsc-3.16.6-opt)

# Modify the PETSC_ARCH variable. You can find it in your configuration
# output. If you forget it, go to your PETSc home director and open
# configure.log. Go the end of the file, and you shall find the value
# of PETSC_ARCH
set(PETSC_ARCH .)

# Modify the METIS_DIR.
# Note: If your PETSc has METIS installed, the conf
# file will directly load that METIS; otherwise this METIS will
# be used for PERIGEE. This means, if you are sure that you have
# METIS in PETSc, you do not have to specify the METIS_DIR variable.
set(METIS_DIR $HOME/lib/metis-5.0.3)

# Modify the HDF5_ROOT, pointing to your hdf5 library location
set(HDF5_ROOT $HOME/lib/hdf5-1.8.16)

```

```
# PETSc install.
set(CMAKE_C_COMPILER  $HOME/lib/petsc-3.16.6-opt/bin/mpicc)
set(CMAKE_CXX_COMPILER $HOME/lib/petsc-3.16.6-opt/bin/mpicxx)
set(CMAKE_CXX_STANDARD 11)
if( ${CMAKE_BUILD_TYPE} MATCHES "Release" )
```

0.3. Open your Terminal, do:

```
$ echo $MACHINE_NAME
```

to check whether there is an environment variation called MACHINE_NAME. If there isn't, think about a good name of your machine, and write it into **~/.bashrc** :

```
$ sudo vi ~/.bashrc
```

(write)

```
export MACHINE_NAME=HXMmac
```

(for me)

(save and exit) (Do you remenber how to use vi to do that?)

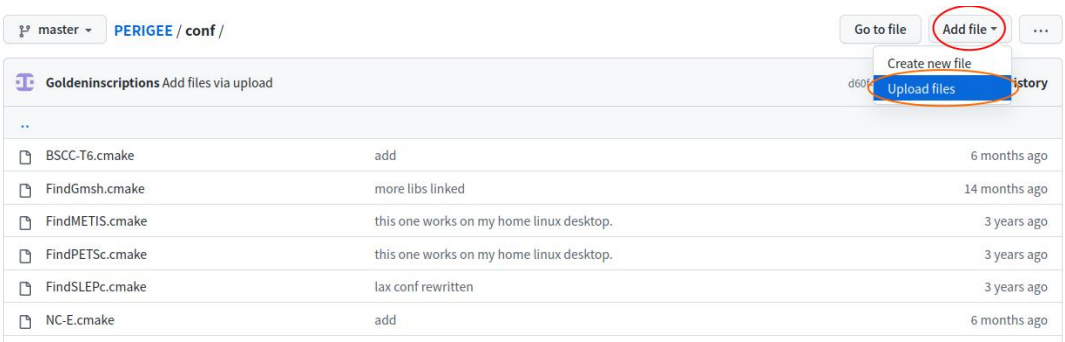
```
$ source ~/.bashrc
```

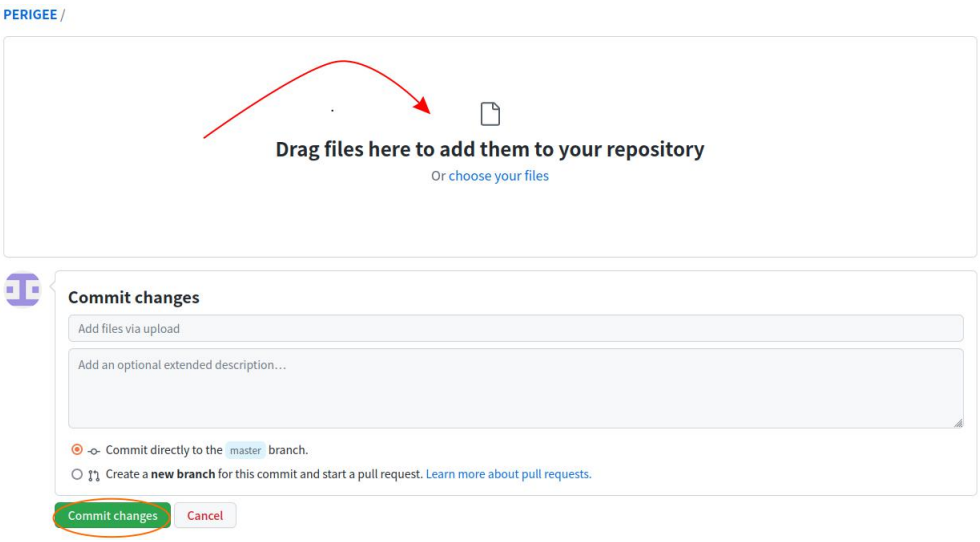
View this page, download **system_lib_loading.cmake**, open it with an editor, move to the last " elseif " branch, add a similar branch under it:

```
elseif( $ENV{MACHINE_NAME} MATCHES "NC-E")
  message(STATUS "NC-E ningxia")
  include(${CMAKE_CURRENT_LIST_DIR}/NC-E.cmake)
elseif( $ENV{MACHINE_NAME} MATCHES "HXMLinux")
  message(STATUS "Huangxm's Linux")
  include(${CMAKE_CURRENT_LIST_DIR}/huangxm-Linux.cmake)
elseif( $ENV{MACHINE_NAME} MATCHES "HXMTaiyi")
  message(STATUS "Huangxm's Taiyi")
  include(${CMAKE_CURRENT_LIST_DIR}/huangxm-Taiyi.cmake)

elseif( $ENV{MACHINE_NAME} MATCHES "HXMmac")  # Your MACHINE_NAME
  message(STATUS "Huangxm's Mac")
  include(${CMAKE_CURRENT_LIST_DIR}/huangxm-Mac.cmake)  # Your CmakeFile's name

else($ENV{MACHINE_NAME} MATCHES "poincare")
  message(STATUS "The system cannot be identified.")
endif( $ENV{MACHINE_NAME} MATCHES "poincare")

# End of the file
```

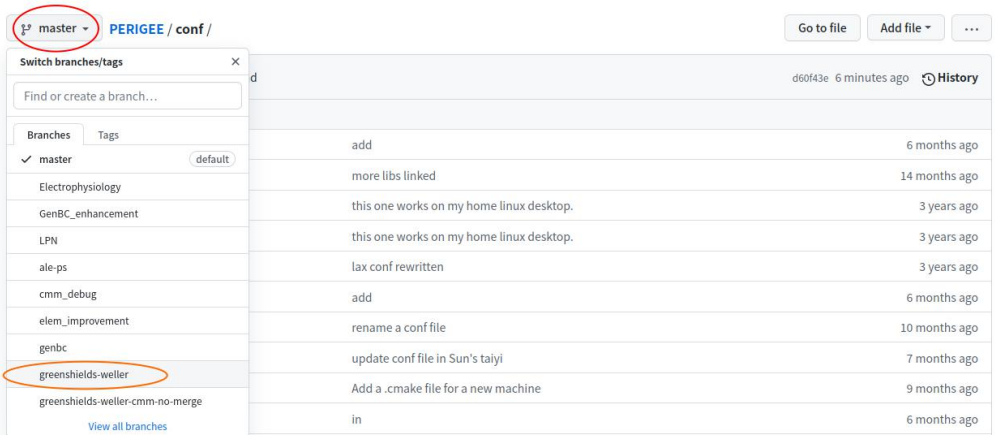
"STATUS" in the middle line will be shown while running PERIGEE, name it as you like, then save the file.

0.4. View this page again, upload **your cmakefile** and the edited **system_lib_loading.cmake**:





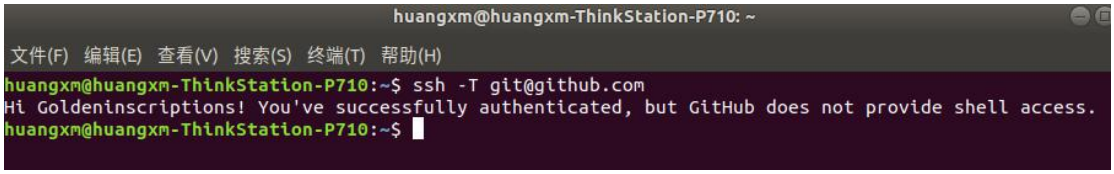
As we want to test the Greenshields-Weller benchmark this time, we shift to the branch "greenshields-weller", and upload two files once again.



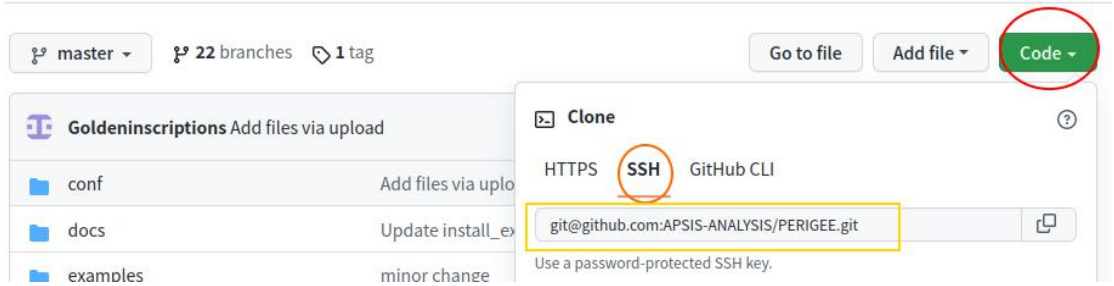
0.5. Let's download PERIGEE source codes to you machine, we suggest using git:

```
$ sudo apt install git
```

After installation, set a ssh-key to your github account, and test it:



View the page of PERIGEE, copy this line:



Select a place to download our source codes, for me:

```
$ cd $HOME
$ mkdir codes
$ cd codes
$ git clone git@github.com:APSIS-ANALYSIS/PERIGEE.git
```

As we want to test the Greenshields-Weller benchmark, we have to shift to the branch "greenshields-weller":

```
$ cd PERIGEE
$ git branch
$ git checkout greenshields-weller
$ git branch
```

0.6. There is one more environment variation which should be set in your machine:

```
$ sudo vi ~/.bashrc
```

(write)

```
export LD_LIBRARY_PATH=$HOME/VTK-7.1.1-shared/lib:$LD_LIBRARY_PATH
```

(save and exit)

```
$ source ~/.bashrc
```

0.7. Now check your preparation:

Compiler: cmake

Libs: mpich, VTK, hdf5, metis, PETSc, SLEPc, Gmsh, ParaView

Environment variations: PATH, LD_LIBRARY_PATH, MACHINE_NAME

Files uploaded: **your cmakefile, system_lib_loading.cmake**

Source codes downloaded: PERIGEE

Furthermore, some shell scripts are prepared to simplify following steps.
To run a shell script in Terminal:

```
$ sh [script_name.sh]
```

On a cluster such as Taiyi, some operations must be done with scripts.

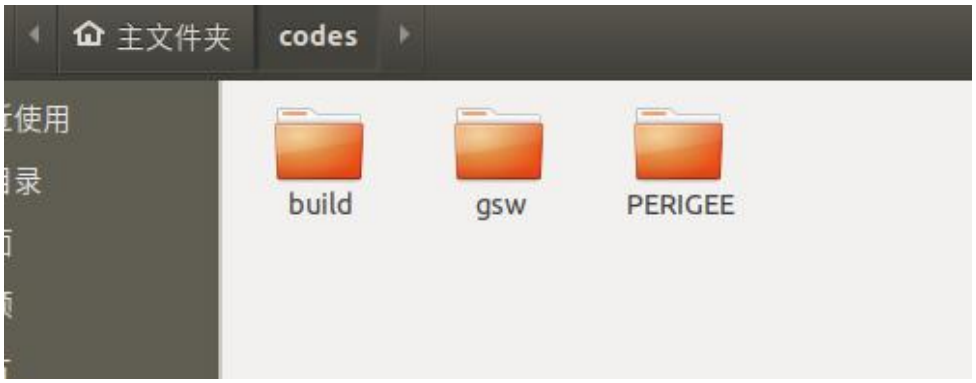
1. Our tutorial starts here formally:

Open a new Terminal, go to the dictory you downloaded PERIGEE in:

```
$ cd $HOME/codes          (for me)
$ ls
$ mkdir gsw build
$ ls
```

We built a document called “build” where we will compile PERIGEE to get a executable program, and another document called “gsw” where we will put our project files and data.

You can also see their directly on the window:



Do:

```
$ cd PERIGEE/input/gmsh
$ ls
```

You can see many .geo files, they restore geometric information and mesh information of our models. What we need is **fsi_cylinder_wBL.geo**, let’s copy it to “gsw”:

```
$ cd ../../../gsw
$ cp ../PERIGEE/input/gmsh/fsi_cylinder_wBL.geo .          (Here is a dot at the end)
$ ls
```

Open it with gmsh to have a look:

```
$ gmsh fsi_cylinder_wBL.geo
```

And open it with editor:

```
$ xdg-open fsi_cylinder_wBL.geo
```

Try to reset some digital arguments and save it, then observe the difference with gmsh.

Here are some recommeded trying:

```
打开(O)
Point(1) = {0, 0, 0};
Point(2) = {1, 0, 0};
Point(3) = {0, 1, 0};
Point(4) = {-1, 0, 0};
Point(5) = {0, -1, 0};
Point(6) = {1.2, 0, 0};
Point(7) = {0, 1.2, 0};
Point(8) = {-1.2, 0, 0};
Point(9) = {0, -1.2, 0};

Circle(1) = {2, 1, 3};
Circle(2) = {3, 1, 4};
Circle(3) = {4, 1, 5};
Circle(4) = {5, 1, 2};
Circle(5) = {6, 1, 7};
Circle(6) = {7, 1, 8};
Circle(7) = {8, 1, 9};
Circle(8) = {9, 1, 6};

Line(9) = {7, 3};
Line(10) = {6, 2};
Line(11) = {9, 5};
Line(12) = {8, 4};

h_bl = 0.8;

Point(10) = {h_bl, 0, 0, 1.0};
Point(11) = {0, h_bl, 0, 1.0};
Point(12) = {-h_bl, 0, 0, 1.0};
Point(13) = {0, -h_bl, 0, 1.0};
```

```
Line Loop(9) = {16, 13, 14, 15};
Plane Surface(9) = {9};

Transfinite Surface {1,2,3,4,5,6,7,8};

Extrude {0, 0, 10} {
  Surface{1,2,3,4,5,6,7,8,9}; Layers{150};
}


Physical Surface("ftop") = {218, 152, 130, 196, 174};
Physical Surface("fbot") = {9, 5, 6, 7, 8};
```

More importantly, these three parameters determine the mesh size. You can try to set a larger ---but not too large---number, otherwise the memory cost and the time cost would exponentially increase.

For beginners, we recommand 4, 4, 20.

```
Transfinite Line {9,10,11,12} = 4 Using Progression 1;
Transfinite Line {-17,-18,-19,-20} = 4 Using Progression 1.05;
Transfinite Line {1,2,3,4,5,6,7,8,13,14,15,16} = 20 Using Progression 1;
```

Now (in the directory of “gsw”) do:

```
$ gmsh fsi_cylinder_wBL.geo -3 -format msh2
$ ls
```

which means to convert the .geo file to a .msh file with 3-D modeling.

Our PERIGEE will use **fsi_cylinder_wBL.msh** later. You can have a look at the mesh by:

```
$ gmsh fsi_cylinder_wBL.msh
```

2. Let's complie the source codes of PERIGEE:

```
$ cd $HOME/codes/build
$ cmake ../PERIGEE/examples/tet4_fsi
```

You must see the identifications info as follows:

```
-- CMAKE_RANLIB: /usr/bin/ranlib
-- Huangxm's Linux
-- Found PETSc: /home/huangxm/lib/petsc-3.16.6-opt/./lib/libpetsc.s
-- Found HDF5: /home/huangxm/lib/hdf5-1.8.16/lib/libhdf5.so;/usr/li
(found version "1.8.16")
-- Use METIS in PETSc: /home/huangxm/lib/petsc-3.16.6-opt/./lib/lib
-- External Libraries: vtkChartsCorevtkCommonColorvtkCommonCorevtks
onModelvtkFiltersGeneralvtkCommonComputationalGeometryvtkFiltersCor
tkRenderingContext2DvtkRenderingCorevtkFiltersGeometryvtkFiltersSou
tkIOXMLParservtkexpatvtkDomainsChemistryOpenGL2vtkRenderingOpenGL2v
athsvtkFiltersGenericvtkFiltersHybridvtkImagingSourcesvtkFiltersHyp
agingvtkFiltersPointsvtkFiltersProgrammablevtkFiltersSMPvtkFilters
kInteractionStylevtkInteractionWidgetsvtkImagingColorvtkRenderingAn
kexoIIcvtkNetCDFvtkNetCDF_cxxvtkIOExportvtkRenderingGL2PSOpenGL2vtk
IONetCDFvtkIOPLYvtkIOParallelvtkjsoncppvtkIOParallelXMLvtkIOSQLvtks
ingStencilvtkInteractionImagevtkRenderingContextOpenGL2vtkRendering
ome/huangxm/lib/petsc-3.16.6-opt/./lib/libpetsc.so/home/huangxm/lib
inux-gnu/libz.so/usr/lib/x86_64-linux-gnu/libdl.so/usr/lib/x86_64-l
-- ------------------- Compiler setup -------------------------
-- CMAKE_CXX_COMPILER: /home/huangxm/lib/mpich-3.3/bin/mpicxx
-- CMAKE_C_COMPILER: /home/huangxm/lib/mpich-3.3/bin/mpicc
-- CMAKE_C_FLAGS:
```

Otherwise, your libs may not be identified successfully, you should select another name as your MACHINE_NAME, reset it in **~/.bashrc** and **system_lib_loading.cmake**. Try "cmake" again until your libs are identified.

For the Greenshields-Weller benchmark, we complied **tet4_fsi**, there are other examples in the directory: /PERIGEE/examples .

Then, do:

```
$ make -j 6
$ ls
```

to get executable programs.

3. Now we do the preprocessing, let's move to "gsw" then all data will be created there:

```
$ cd ../gsw
$ ../build/gmshIO -gmsh_file fsi_cylinder_wBL.msh
$ ls
```

gmshIO is the first executable program we run, which converts the .msh file to some .vtu/.vtp files. It will take a few minutes.

"nElem" means the number of elements. This case is with mesh arguments: 4, 4, 20.

```
    names: fluid        solid
    nElem: 753300       205200
    etype: 4    4
    nLocBas: 4  4
=== Total node number : 166704
=== Total element number : 1008360
=== Gmsh FileIO::write vtp for ftop a
```

4. We will deal with these .vtu/.vtp files with **preprocess_fsi**:

```
huangxm@huangxm-ThinkStation-P710:~/codes/gsw$ ls
fbot_fluid.vtp  fsi_cylinder_wBL.geo  ftop_fluid.vtp   fwall_solid.vtp  solid.vtu       swall_solid.vtp
fluid.vtu       fsi_cylinder_wBL.msh  fwall_fluid.vtp  sbot_solid.vtp   stop_solid.vtp  whole_vol.vtu
```

Bofore that, do:

$ xdg-open ../PERIGEE/examples/tet4_fsi/preprocess_fsi.cpp

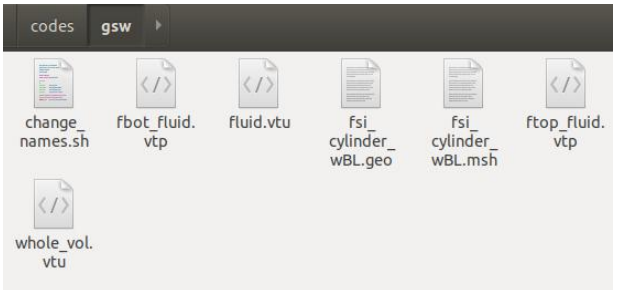
You can find that it requires we change the name of files we input:

```
// Input files
std::string geo_file("./whole_vol.vtu");

std::string geo_f_file("./lumen_vol.vtu");
std::string geo_s_file("./tissue_vol.vtu");

std::string sur_f_file_wall("./lumen_wall_vol.vtp");
std::string sur_f_file_in_base( "./lumen_inlet_vol_" );
std::string sur_f_file_out_base("./lumen_outlet_vol_");

std::string sur_s_file_interior_wall("./tissue_interior_wall_vol.vtp");

std::string sur_s_file_wall("./tissue_wall_vol.vtp");
std::string sur_s_file_in_base( "./tissue_inlet_vol_" );
std::string sur_s_file_out_base("./tissue_outlet_vol_");
```

Therefore, we have a shell script **change_names.sh** to do that, and here are the rules:

```
# After command 'gmshIO -gmsh_file ...', we got some vtu/vtp files.
# Change their names by refering to '../PERIGEE/examples/tet4_fsi/preprocess_fsi.cpp'.
# 'f' means 'fluid' ------> lumen
# 's' means 'solid' ------> tissue
# 'f_s' ------> interior
# KEEP 'whole_vol.vtu'.

mv fluid.vtu lumen_vol.vtu
mv fwall_fluid.vtp lumen_wall_vol.vtp
mv fbot_fluid.vtp lumen_inlet_vol_000.vtp
mv ftop_fluid.vtp lumen_outlet_vol_000.vtp

mv fwall_solid.vtp tissue_interior_wall_vol.vtp

mv solid.vtu tissue_vol.vtu
mv swall_solid.vtp tissue_wall_vol.vtp
mv sbot_solid.vtp tissue_inlet_vol_000.vtp
mv stop_solid.vtp tissue_outlet_vol_000.vtp
```

You should copy it to "gsw":



Do:

$ sh. change_names.sh

$ ../build/preprocess_fsi -cpu_size 10

It will take a few minutes.

"-cpu_size" means the number of cpu cores you request, if you are not sure about how many cores your have, do:

$ cat /proc/cpuinfo

5. After preprocessing, we can run the solver **fsi_tet4_3d** directly:

```
$ ../build/fsi_tet4_3d -cpu_size 10
```

or use a run-scipt.sh:

```
$ sh run-script.sh
```
(recommended)

you can check every argument of it by comparing with their comments in .cpp files:

```
$ xdg-open run-script.sh
$ xdg-open ../PERIGEE/examples/tet4_fsi/fsi_driver.cpp
```

It will print all arguments, and when you see this:

```
-----------------------------------------------------------------------
===> Start Finite Element Analysis:
Time = 0.000000e+00, dt = 1.000000e-06, index = 0, 12:53:07
  --- M updated  Init res 2-norm: 5.379476e+03
  --- KSP: 3, 2.265395e+01  --- KSP: 4, 8.804061e-09  --- nl_res: 2.265416e+01
  --- KSP: 3, 2.940734e-02  --- KSP: 6, 2.165685e-11  --- M updated  --- nl_res: 2.940652e-02
  --- KSP: 3, 4.324923e-05  --- KSP: 6, 7.299010e-14  --- nl_res: 4.324871e-05
  === NR ite: 3, r_error: 8.039576e-09, a_error: 4.324871e-05
```

The solver is working well, and we shall wait for a long time.

(Waiting...)

After the solving process, you will get some files wilth prefix “SOL”:

```
huangxm@huangxm-ThinkStation-P710:~/codes/gsw$ ls
apart                    dot_SOL_velo_900000500    node_mapping_v.h5     SOL_pres_900002000
change_names.sh          dot_SOL_velo_900001000    npart.h5              SOL_velo_900000000
dot_SOL_disp_900000000   dot_SOL_velo_900001500    preprocessor_cmd.h5   SOL_velo_900000500
dot_SOL_disp_900000500   dot_SOL_velo_900002000    run-script.sh         SOL_velo_900001000
dot_SOL_disp_900001000   epart.h5                  SOL_disp_900000000    SOL_velo_900001500
dot_SOL_disp_900001500   fsi_cylinder_wBL.geo      SOL_disp_900000500    SOL_velo_900002000
dot_SOL_disp_900002000   fsi_cylinder_wBL.msh      SOL_disp_900001000    solver_cmd.h5
dot_SOL_pres_900000000   lumen_inlet_vol_000.vtp   SOL_disp_900001500    tissue_inlet_vol_000.vtp
dot_SOL_pres_900000500   lumen_outlet_vol_000.vtp  SOL_disp_900002000    tissue_interior_wall_vol.vtp
dot_SOL_pres_900001000   lumen_vol.vtu             SOL_pres_900000000    tissue_outlet_vol_000.vtp
dot_SOL_pres_900001500   lumen_wall_vol.vtp        SOL_pres_900000500    tissue_vol.vtu
dot_SOL_pres_900002000   material_model.h5          SOL_pres_900001000    tissue_wall_vol.vtp
dot_SOL_velo_900000000   node_mapping_p.h5          SOL_pres_900001500    whole_vol.vtu
```

Their quantity and the suffixes depend on your time/step and recording configurations:

```
打开(O)                                                    run-script.sh
                                                           ~/codes/gsw
mpirun -np 10 ../build/fsi_tet4_3d -nz_estimate 300 \
  -nqp tet 5 -nqp tri 13 \
  -init_step 1.0e-6 -fina_time 2.0e-3 \
  -nl_refreq 2 -nl_rtol 1.0e-6 -nl_atol 1.0e-15 -nl_dtol 1.0e2 \
  -nl_maxits 8 \
  -log_view -ttan_freq 100 -sol_rec_freq 500 \
  -is_restart NO -restart_index 79500 -restart_time 7.95e-3 \
  -restart_step 1.0e-7 \
```

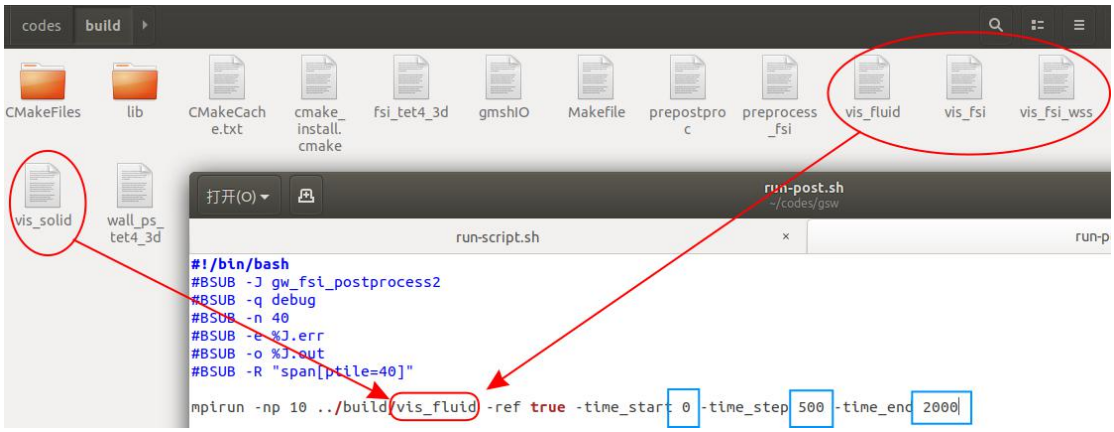
Of cource you can scroll up to read the summary to get more infomations:

```
  === NR ite: 3, r_error: 9.378994e-07, a_error: 1.892985e-03
Time = 2.001000e-03, dt = 1.000000e-06, index = 2001, 22:21:27
************************************************************************************************************************
***                WIDEN YOUR WINDOW TO 120 CHARACTERS.  Use 'enscript -r -fCourier9' to print this document            ***
************************************************************************************************************************

---------------------------------------------- PETSc Performance Summary: ----------------------------------------------

../build/fsi_tet4_3d on a  named huangxm-ThinkStation-P710 with 10 processors, by huangxm Mon Oct 24 22:21:27 2022
Using Petsc Release Version 3.16.6, Mar 30, 2022

                         Max       Max/Min     Avg       Total
Time (sec):           2.699e+04     1.000   2.699e+04
Objects:              1.331e+05     1.000   1.331e+05
Flop:                 1.599e+12     1.038   1.574e+12  1.574e+13
```

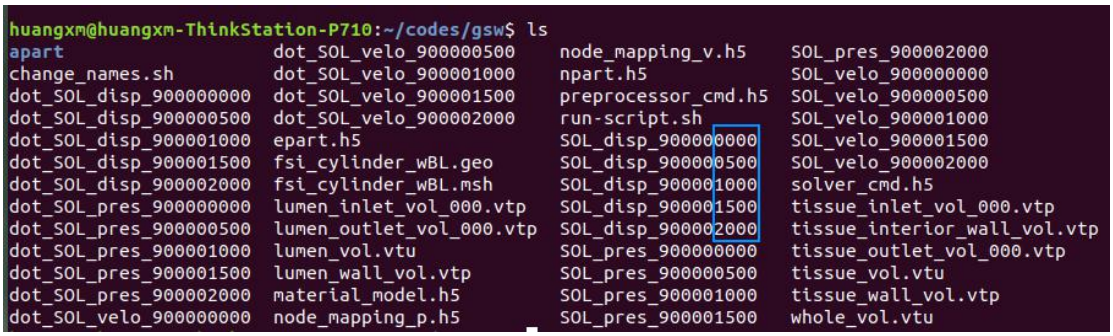
6. Now, we do the postprocessing:

```
$ ../build/prepostproc -cpu_size 10
```

prepostproc does the "preprocessing" for our postprocess. When it is done, copy the script **run-post.sh** to "gsw", open it with your editor and open "build" on the window:



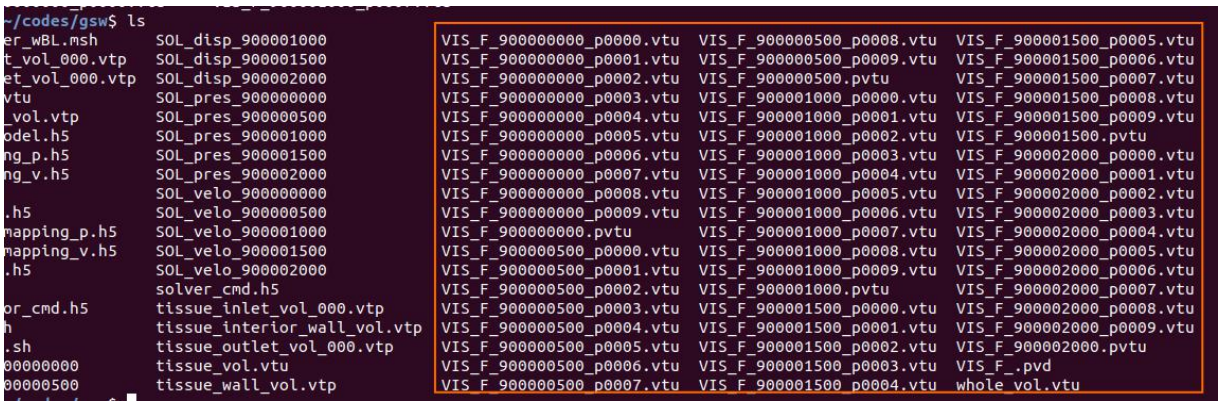
These four files with prefix "vis_" are all the postprocessor, you should select one of them according to your requirement. The time parameter should strictly match with your "SOL_file" sequence:



Save the script and run it:

```
$ sh run-post.sh
$ ls
```

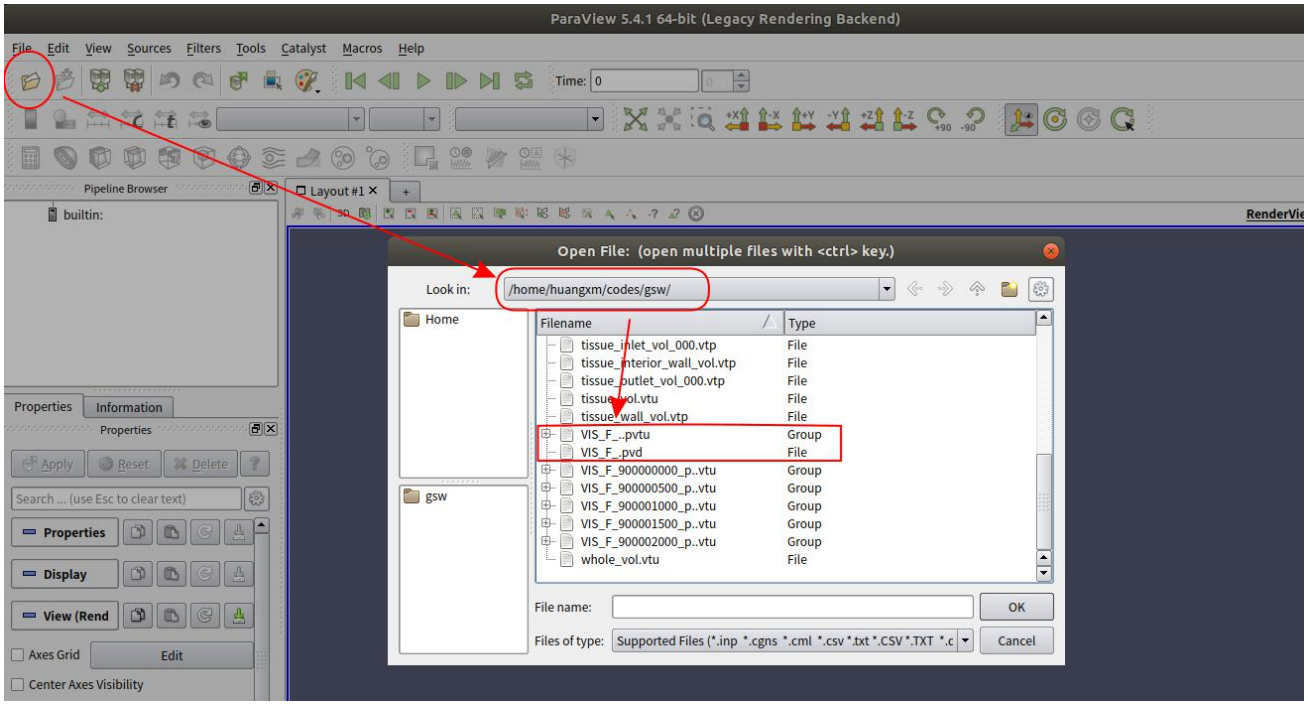
More .vtu/.vtp files are created with prefix "VIS_" , which restore our solutions and can be viewed with ParaView:



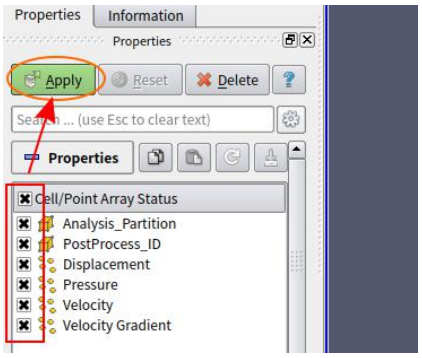
7. Finally, we visualize our solutions:

```
$ paraview
```

Then open the file **VIS_F_.pvd** or **VIS_F_..pvtu**, both of them consist of all solutions. If you just want a part of your solutions, open the VIS_files with number suffixes such as **VIS_F_900000500_p..vtu** .

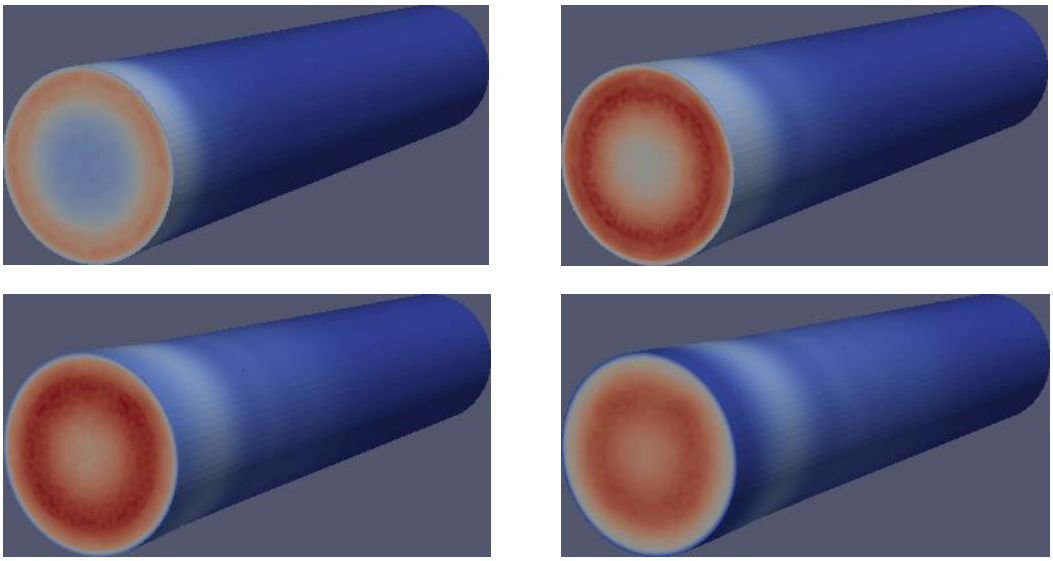


For example, we choose **VIS_F_..pvtu**, select the information you want, then click “Apply” :

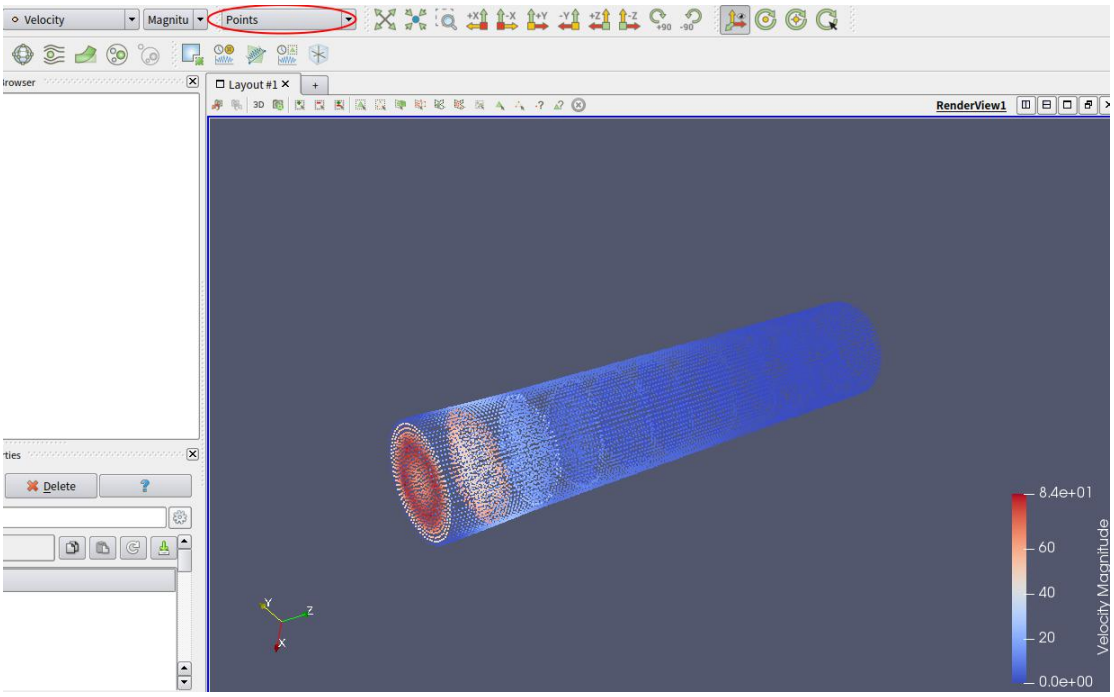


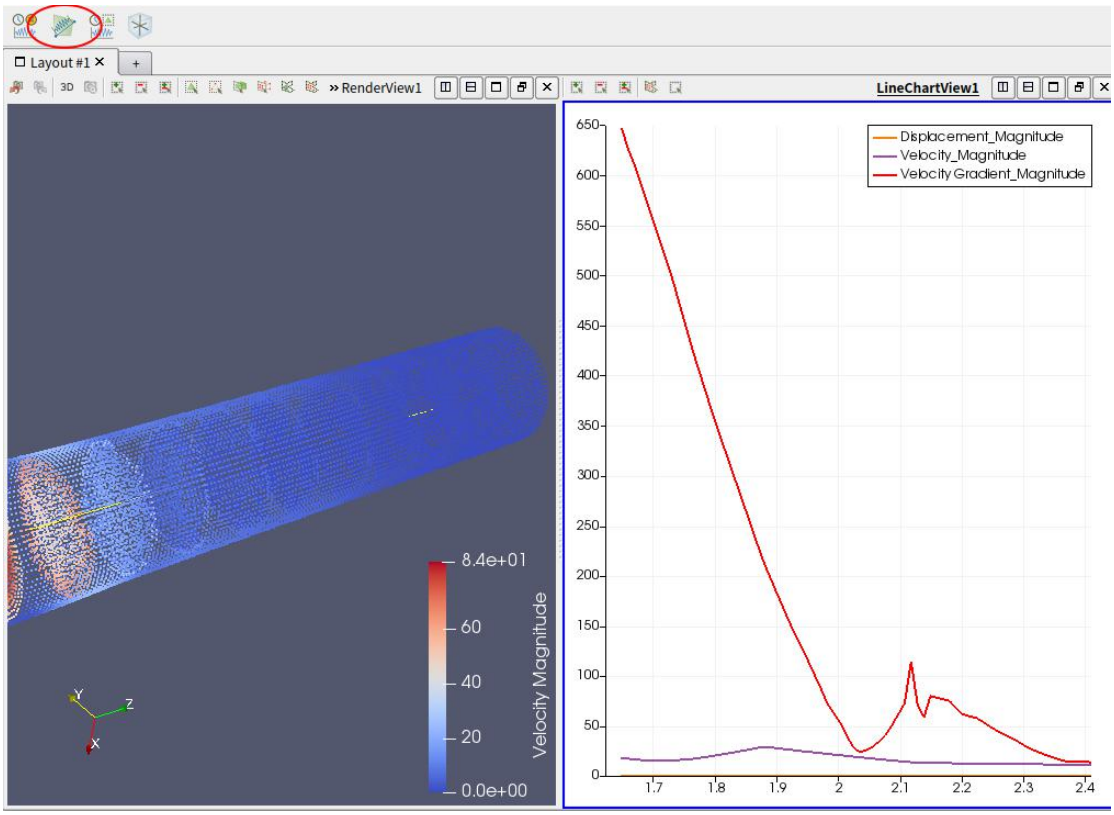
When a grey and ugly cylinder appears on your window, select a physical quantity on the head, click “PLAY” :

Then you will see a “pulse” flow into our model like this:



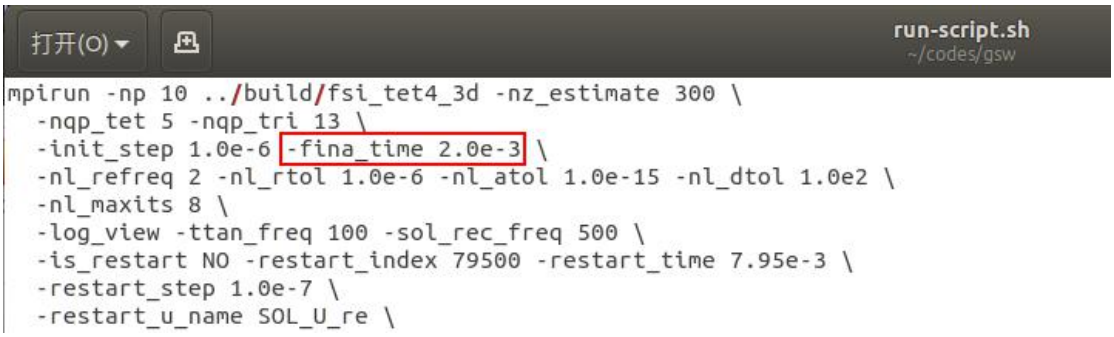
Explore ParaView by yourself, for example:





8. If you followed this guide much too strictly and got the same result as we showed, maybe you found that the "pulse" came to a halt as soon as it just started.

That's because we set a short time for analysis with purpose of teaching:




```
mpirun -np 10 ../build/fsi_tet4_3d -nz_estimate 300 \
  -nqp_tet 5 -nqp_tri 13 \
  -init_step 1.0e-6 -fina_time 2.0e-3 \
  -nl_refreq 2 -nl_rtol 1.0e-6 -nl_atol 1.0e-15 -nl_dtol 1.0e2 \
  -nl_maxits 8 \
  -log_view -ttan_freq 100 -sol_rec_freq 500 \
  -is_restart NO -restart_index 79500 -restart_time 7.95e-3 \
  -restart_step 1.0e-7 \
  -restart_u_name SOL_U_re \
```

With 10 cores, 1*10^6 elements and 2000 steps, this progress would be finished in one day if everything goes well.

Now, set a larger "-fina_time", try to get a complete "pulse". (we'd better not change "-init_step", unfit steps would make iterations divergent.)

By the way, try to get rid of this guide.

End