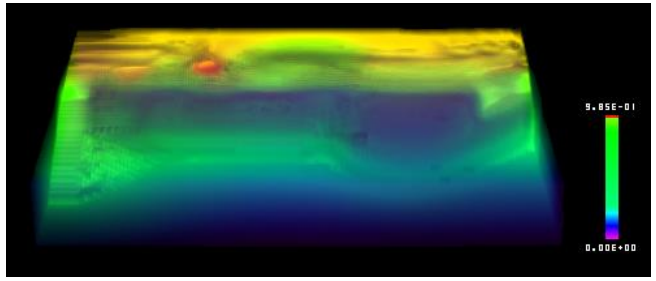
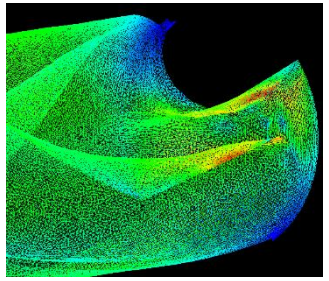
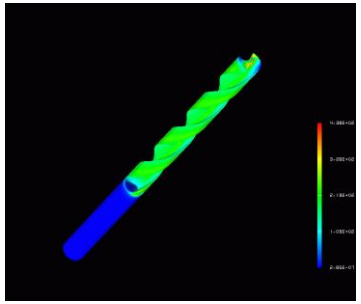
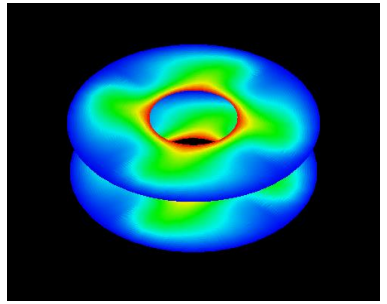
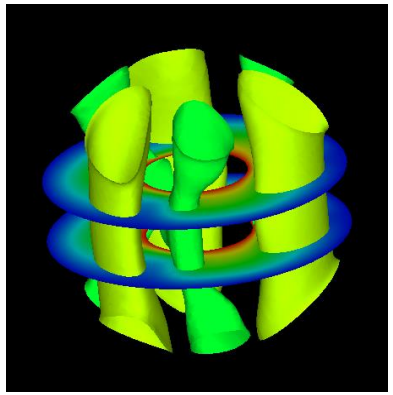
第33回FrontISTR研究会
2017年1月30日（月）
東京大学工学部8号館

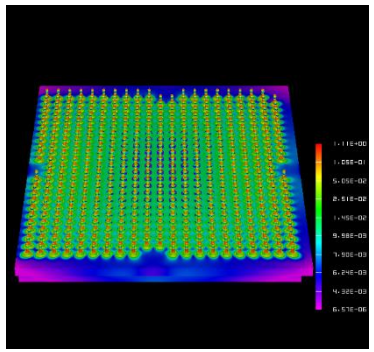





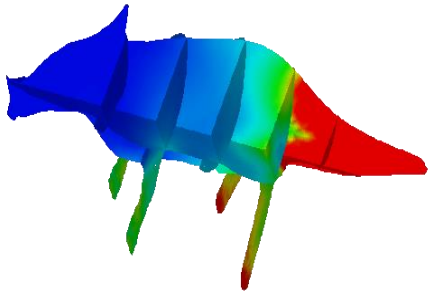
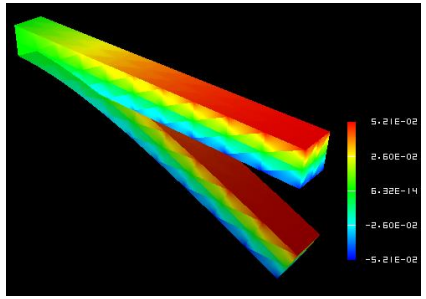
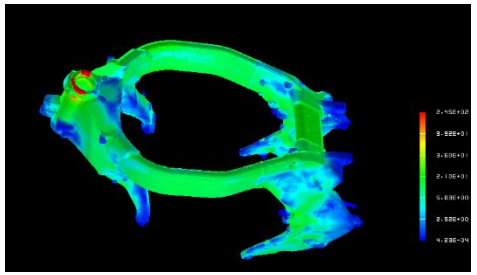
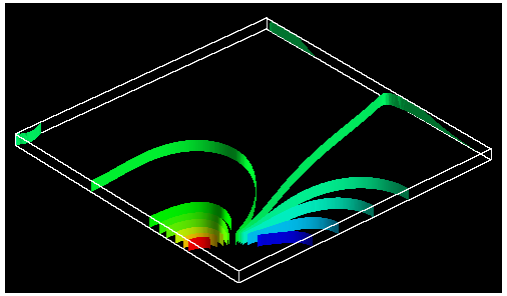
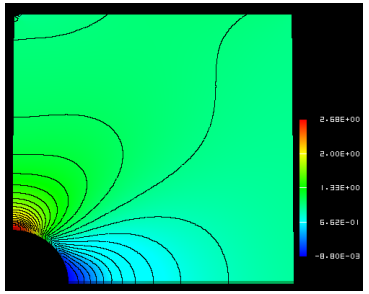
FrontISTR可視化ライブラリ利用方法



奥田 洋司，生野 達大

東京大学大学院新領域創成科学研究科
人間環境学専攻

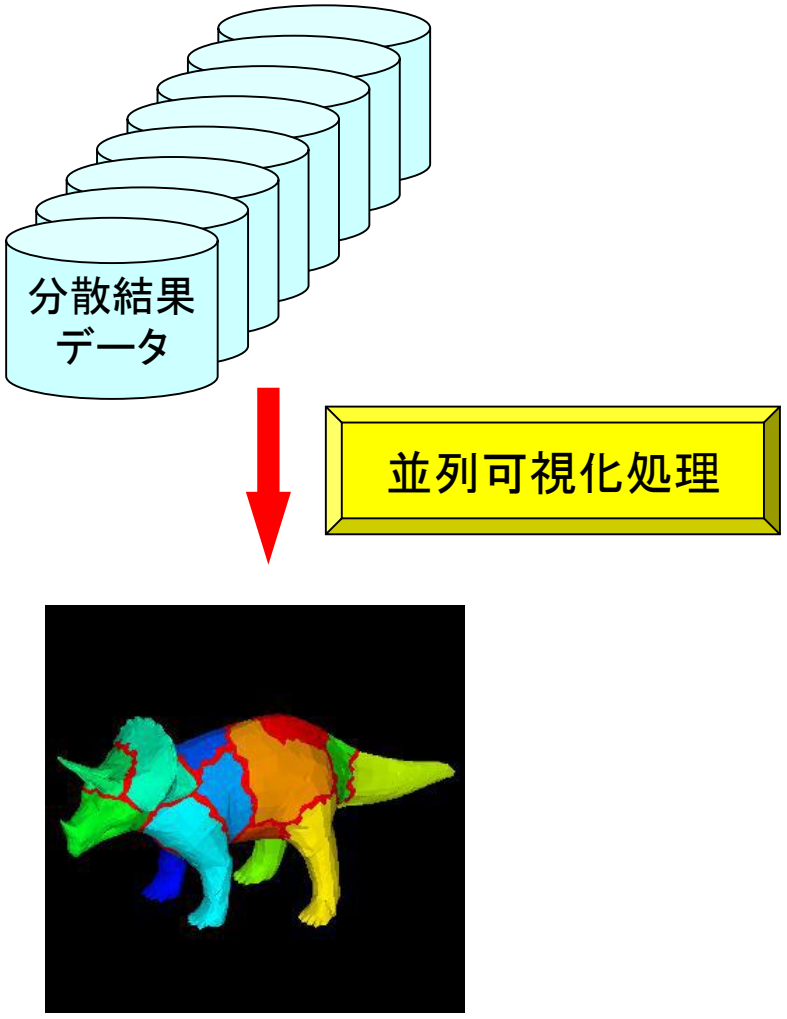


目次

HEC-MW VIS 「並列」可視化とは？ （1/2）

- 並列FEMの結果を，並列計算機を用いて，バッチ処理で可視化
- 解析は非常に大規模で，単一領域のデータは処理不可能
- 解析結果の分散データ(ファイルまたはメモリイメージ)を処理して，一枚の画像を作成する



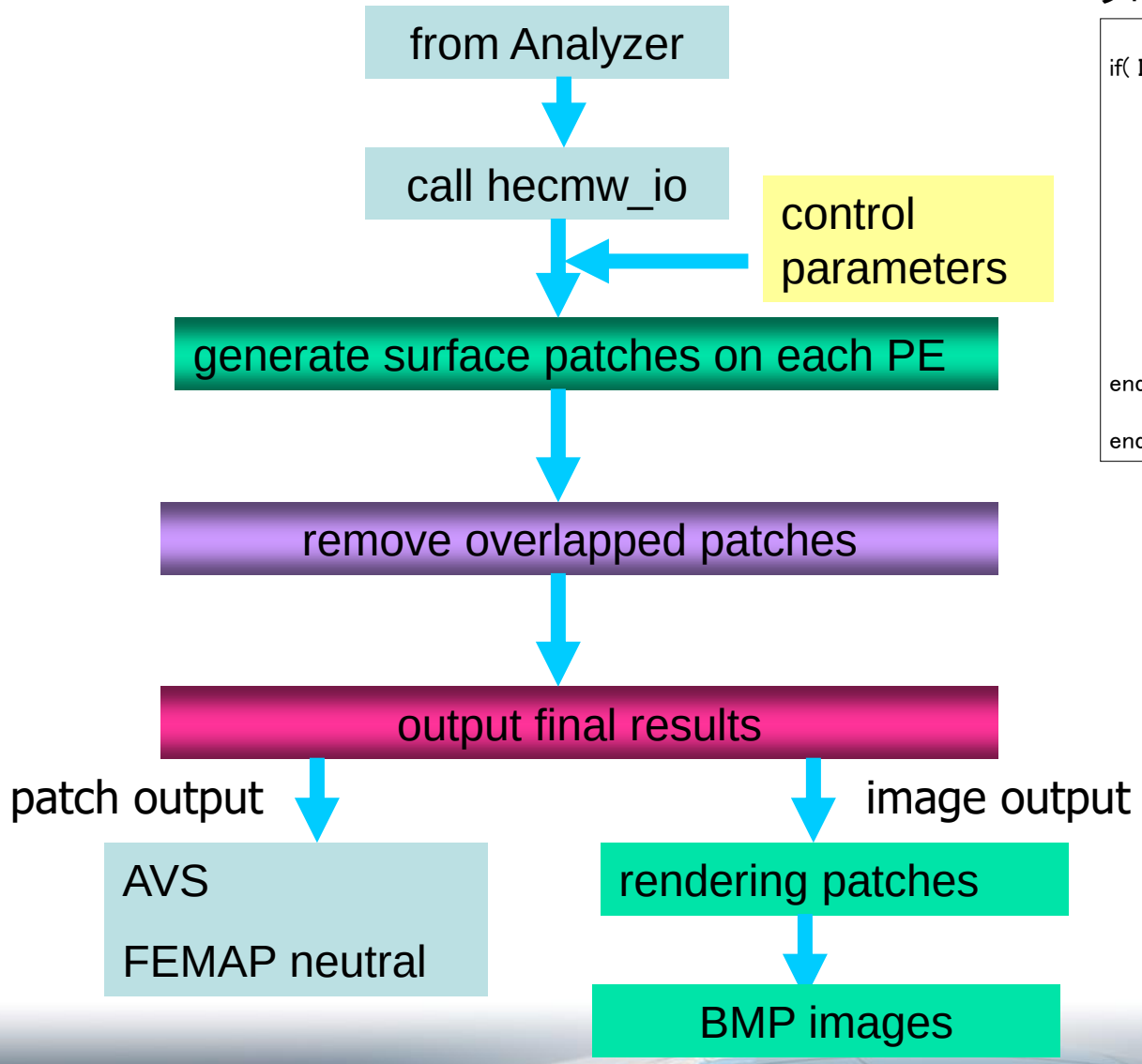
HEC-MW VIS 「並列」可視化とは？ （2/2）

- 「メモリ渡し」「ファイル渡し」
- メモリ渡し（on memory）
  - FEM解析に続けて可視化処理
  - 計算結果と同時に可視化画像も出力される
- ファイル渡し（via file）
  - FEM解析終了後，いったんファイル出力された計算結果を再度読み込んで可視化処理

```
        :
! WRITE, RESULT
解析結果をファイル出力（「ファイル渡し」のために必要）
! WRITE, VISUAL
（「メモリ渡し」のために必要）
! VISUAL, method=PSR
(各種の可視化パラメータ)
        :
```

解析制御ファイル（.cnt）の可視化部分

PSRの処理の流れ（メモリ渡し）



プログラム内での可視化ライブラリの呼ばれ方

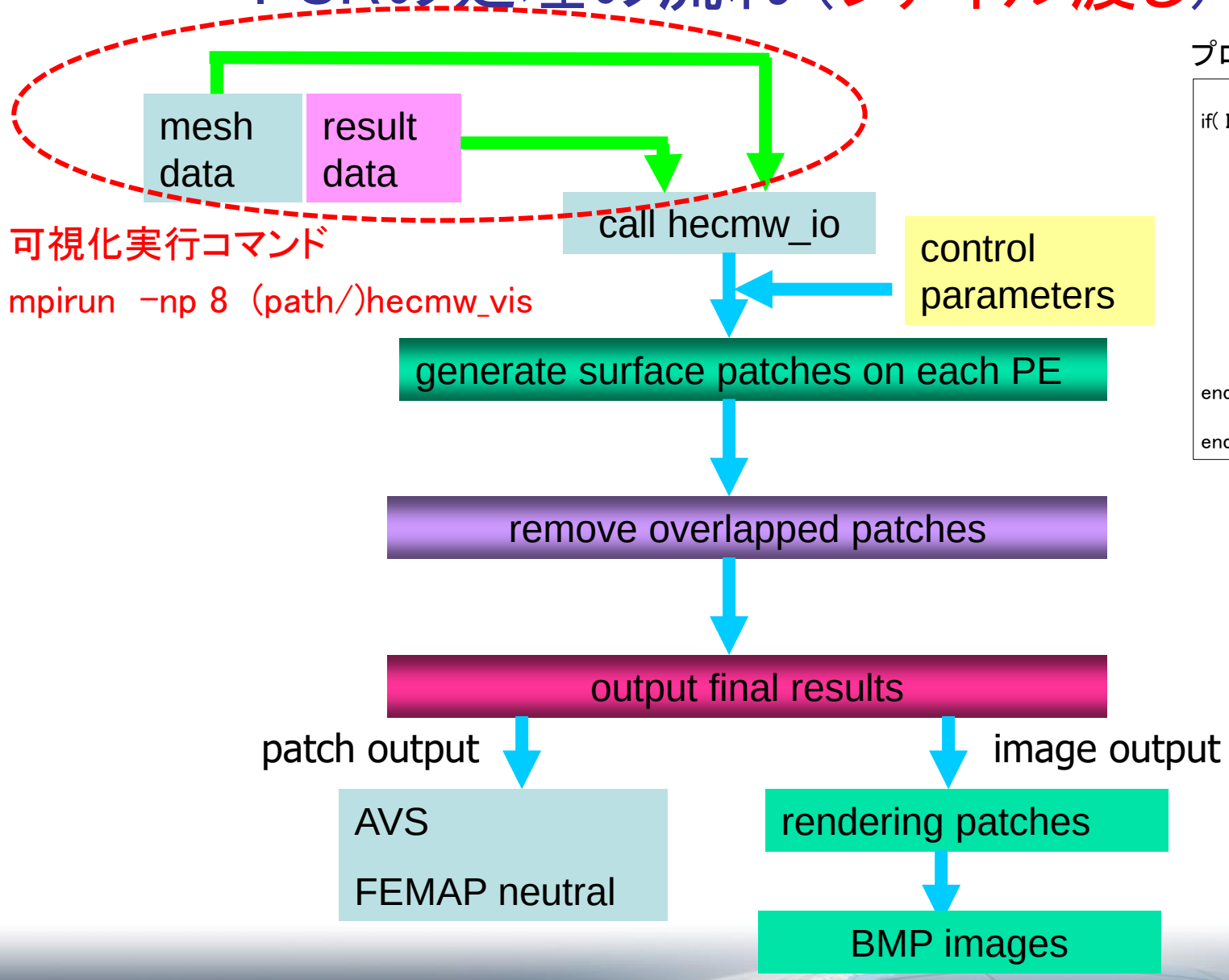
```
      :
if( IVISUAL .eq. 1 ) then
      :
    call hecmw_visualize_init       - 可視化制御パラメータの読込

    call hecmw_visualize            - 計算結果を可視化プログラムにメモリ経由で転送
                                    - PSR, PVRモジュールをコール
                                    - 可視化画像（イメージデータ）を出力
                                    - 可視化ツール用ファイル（パッチデータ）を出力

    call hecmw_visualize_finalize   - メモリ解放

endif
      :
end subroutine FSTR_SOLVE_LINEAR
```

PSRの処理の流れ（ファイル渡し）



プログラム内での可視化ライブラリの呼ばれ方

```
      :
if( IVISUAL .eq. 1 ) then
      :
    call hecmw_visualize_init       – 可視化制御パラメータの読込

    call hecmw_visualize            – 計算結果を可視化プログラムにメモリ経由で転送
                                    – PSR, PVRモジュールをコール
                                    – 可視化画像（イメージデータ）を出力
                                    – 可視化ツール用ファイル（パッチデータ）を出力

    call hecmw_visualize_finalize   – メモリ解放

endif
      :
end subroutine FSTR_SOLVE_LINEAR
```

プログラム内での可視化ライブラリの呼ばれ方

```
      :
if( IVISUAL .eq. 1 ) then
      :
    call hecmw_visualize_init          - 可視化制御パラメータの読込

    call hecmw_visualize               - 計算結果を可視化プログラムにメモリ経由で転送
                                       - PSR, PVRモジュールをコール
                                       - 可視化画像(イメージデータ)を出力
                                       - 可視化ツール用ファイル(パッチデータ)を出力

    call hecmw_visualize_finalize      - メモリ解放

endif
      :
end subroutine FSTR_SOLVE_LINEAR
```

解析制御ファイル（.cnt） 可視化部分の概要

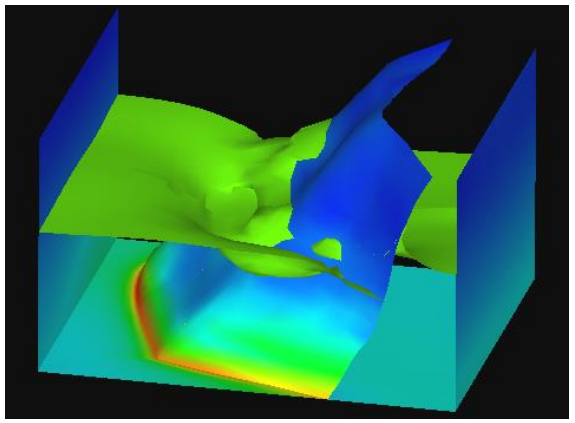
! VISUAL, method=PSR, visual_start_step=2, visual_interval_step=5, visual_end_step=20

引数

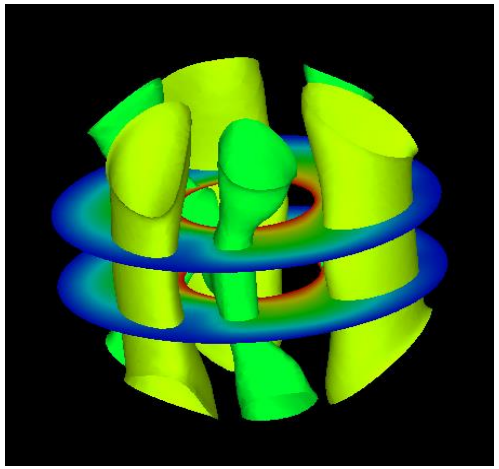
- method { PSR, PVR }
- visual_start_step 可視化開始ステップ （default : -1）
- visual_end_step 可視化終了ステップ
- visual_interval_step 可視化ステップ間隔 （default: 1）

Parallel Surface Rendering (PSR)

- 境界面
- 等値面
  - 複数面の抽出
  - 半透明表示
- その他
  - 方程式を用いた指定
  - 断面



断層データにおける等値面と境界面



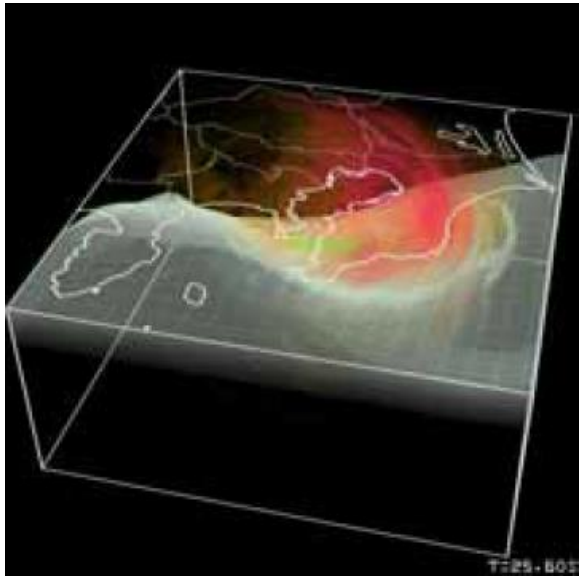
マントルコアデータにおける
複数枚の等値面と断面

Parallel Volume Rendering (PVR)

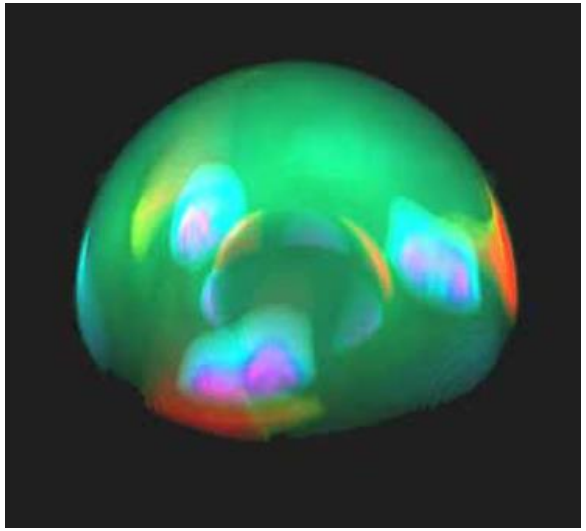
- 適用可能な格子構造
  規則的/階層的/非構造的/複合的

- 高効率
  - ハイブリッド複合並列化
  - 動的負荷分散

- 高品質
  豊富な伝達関数の提供



構造格子データのPVR例（地震波動伝搬シミュレーション．データ提供：古村孝史教授）



非構造格子データのPVR例（マントルコア熱流動シミュレーション）

本日は PSR の事例を紹介します

目次

PSR のパラメータ設定 （共通(1/2)）

!surface_num : 抽出する面面の数

!surface : 新しい面の識別

!surface_style : 抽出する面のタイプ

!color_comp_name : 色付けに使用する成分名

!color_comp : 色付けに使用する成分名のID number

!color_subcomp : color_comp>1の場合のスカラへの変換法 (subcompnent ID)

!display_method : 描画方法

!isoline_number : 等値線の数

!specified_color : 特定色の指定

!output_type : 出力ファイルタイプ

PSR のパラメータ設定 （共通(2/2)）

!deform_display_on :　　変形表示の有無

!deform_comp_name :　　変形を指定する際の採用する属性

!deform_comp :　変形を指定する際の変数識別番号

!deform_scale :　変形を表示する際の変位スケール

!initial_style :　初期変形表示のタイプ

!deform_style :　変形後の形状表示スタイル

!initial_line_color :　　初期メッシュを表示する際のカラー指定

!deform_line_color :　　変形メッシュを表示する際のカラー指定

!deform_num_frames :　　変形メッシュを表示する際の回転に要するフレーム数

PSR のパラメータ設定 （等値面(surface_style = 2 のとき)）

!data_comp_name : 等値面を抽出する要素名

!data_comp : 等値面を抽出する要素名のID number

!data_subcomp : data_comp>1の場合のスカラへの変換法(subcomponent ID)

!iso_value : 抽出する等値面の値

PSR のパラメータ設定 （レンダリングパラメタ（BMP outputのとき））

!x_resolution : 画像における x方向ピクセル数

!y_resolution : 画像における y方向ピクセル数

!num_of_lights : 光源数

!position_of_lights : 光源の座標

!viewpoint : 視点の座標

!look_at_point : 注視点の座標

!up_direction : 上方向の座標

!ambient_coef : 環境係数

!diffuse_coef : 拡散反射係数

!specular_coef : 鏡面反射係数

!color_mapping_style : カラーマッピング方法

!interval_mapping_num : カラーマッピング区間数

!interval_mapping : フィールド区間のフィールド値，RGB空間の値

!rotate_style : アニメーションにおける回転方法

!rotate_num_of_frames : 回転に要するフレーム数

!color_mapping_bar_on : カラーバー表示の有無

!scale_marking_on : カラーバーのフィールド値表示の有無

!num_of_scales : カラーバーのフィールド値表示の数

!color_mapping_system : カラーマッピングシステムの定義

!font_size : カラーバーフィールド値のフォントサイズ

!font_color : カラーバーフォントのRGB値

!background_color : 背景のRGB値

!fixed_range_on : フィールド範囲値を固定する有無

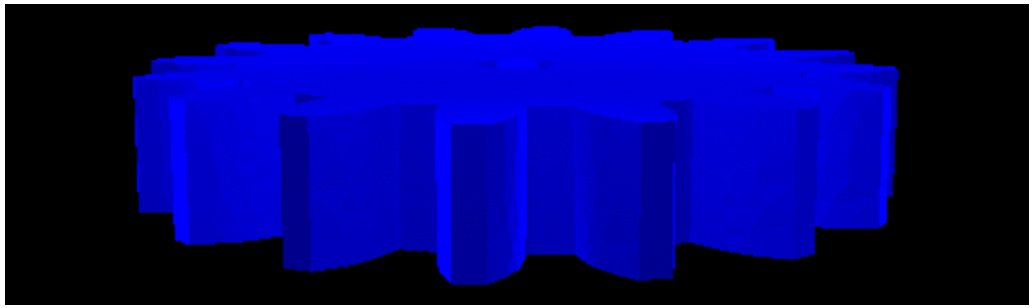
!histogram_on : データ値分布のヒストグラム情報出力有無

!boundary_line_on : 境界線の有無

目次

(1) PSR の最低限のパラメータ設定

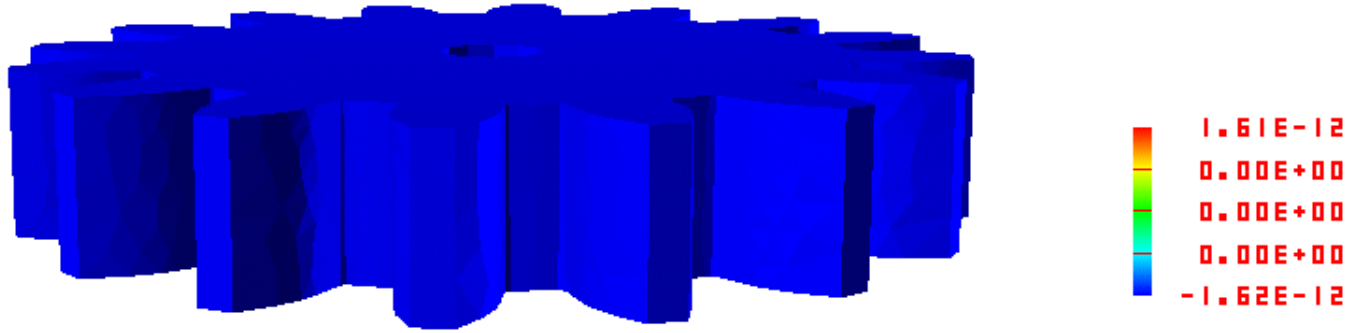
```
!VISUAL, method=PSR
!surface_num = 1
!surface 1
!output_type = BMP （デフォルトは AVS）
!END
```



- 画像ファイル（.bmp）が出力される
- 他のパラメータはデフォルト値となる
- 境界面表示
- 第1成分が色表示 される

(2) (1)にパラメータ追加

```
!display_method=1
!surface_style=1
!initial_style=1
!x_resolution = 800
!y_resolution = 600
!num_of_lights = 1
!position_of_lights = 100.000, 100.000, 50.0000
!up_direction = 0.00000, 0.00000, 1.00000
!ambient_coef = 0.3
!diffuse_coef = 0.7
!specular_coef = 0.5
!color_mapping_bar_on = 1
!scale_marking_on = 1
!num_of_scale = 5
!font_size = 2.0
!font_color = 1.00000, 0.00000, 0.00000
!background_color = 1.00000, 1.00000, 1.00000
```



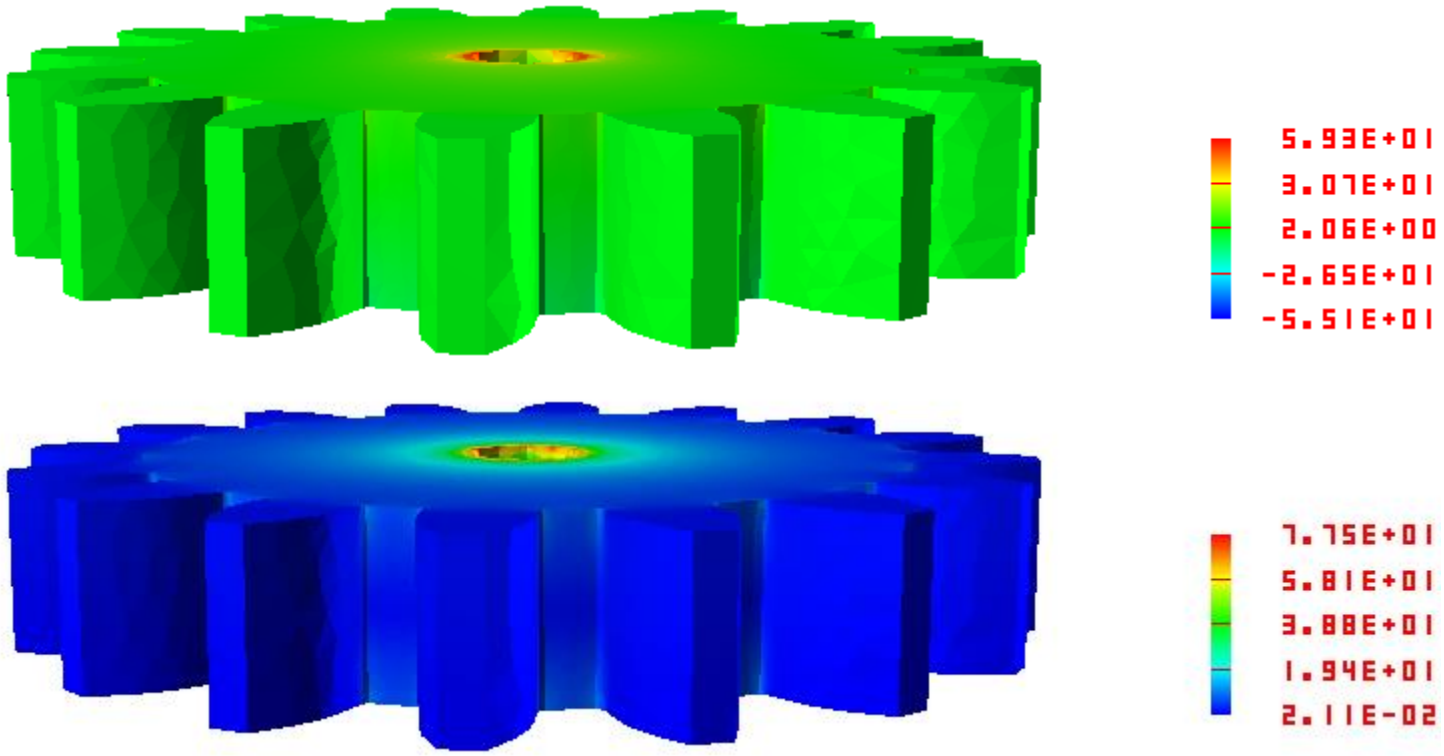
- カラーバー表示
- 背景色の変更

(3) (2)にパラメータ追加

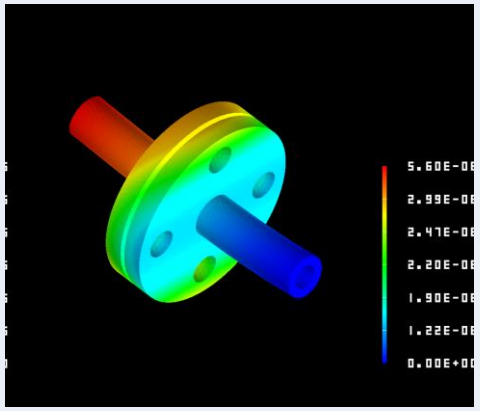
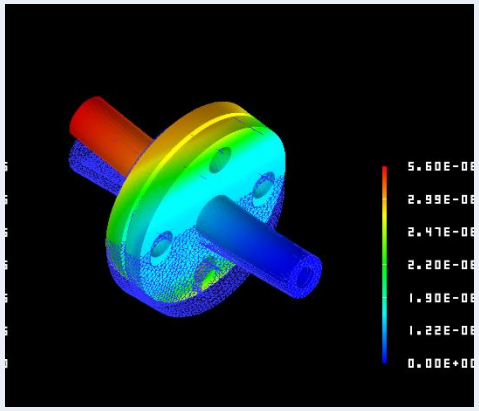
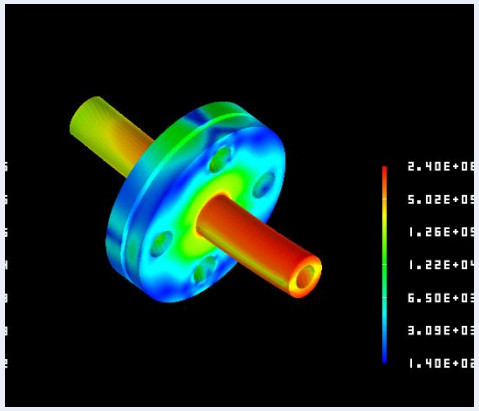
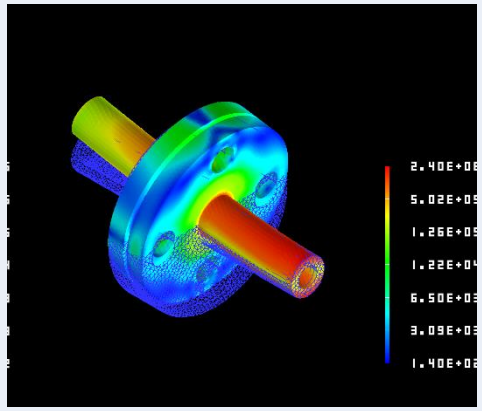
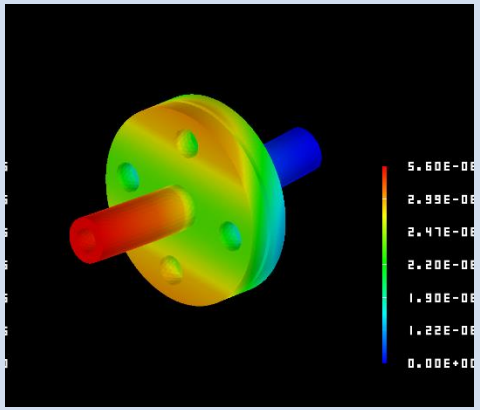
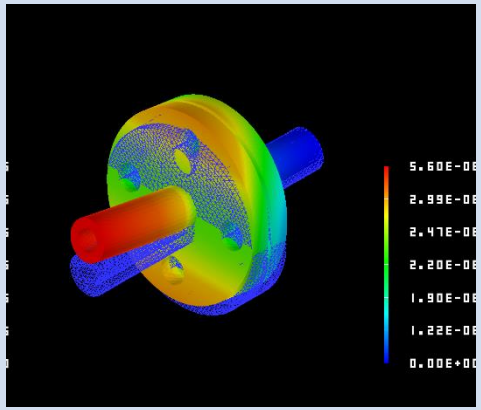
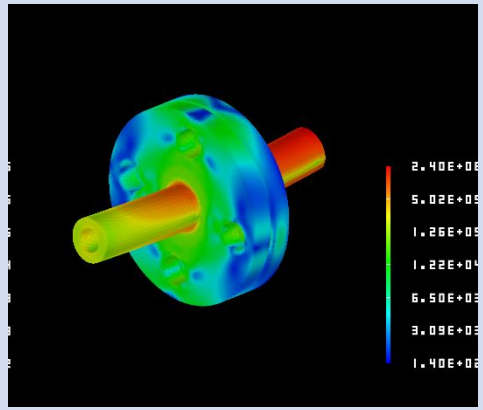
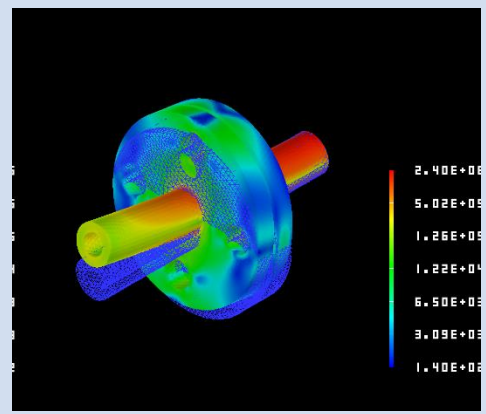
```
!color_comp_name    =   displacement
!color_subcomp_name    =   norm
!color_comp=1

上から
!color_subcomp          1
!color_subcomp          0
```

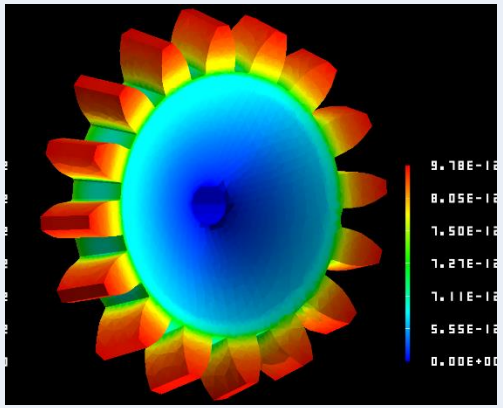
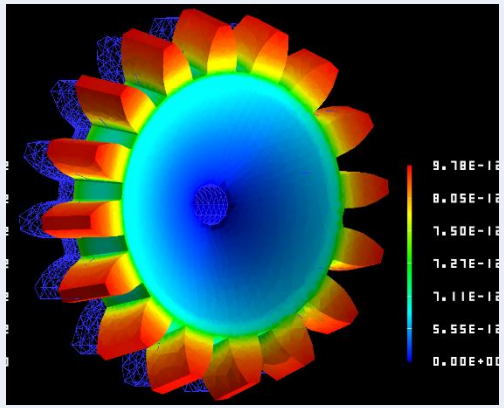
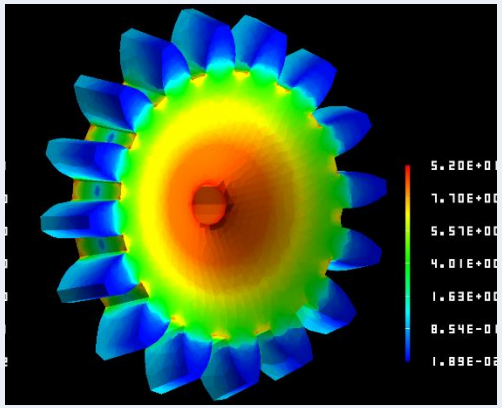
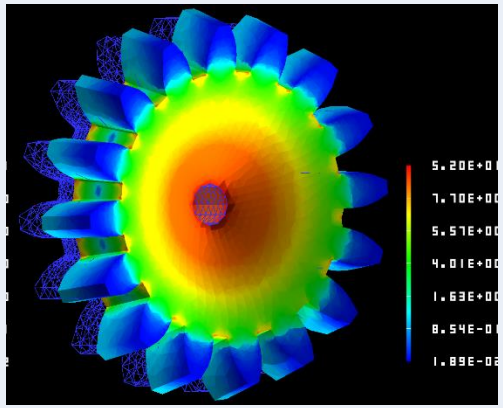
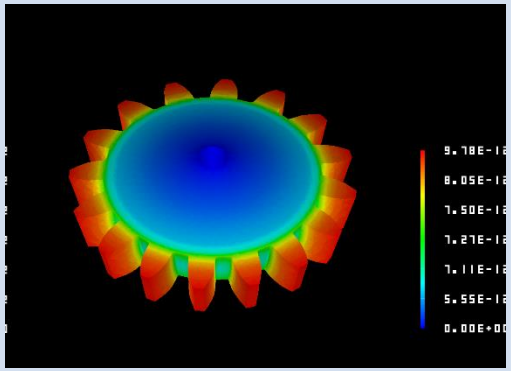
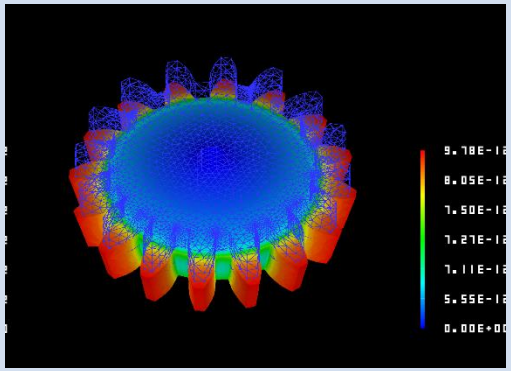
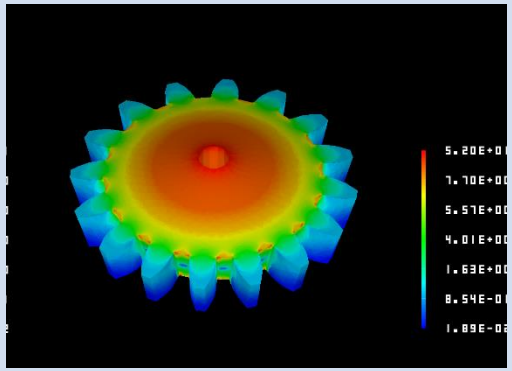
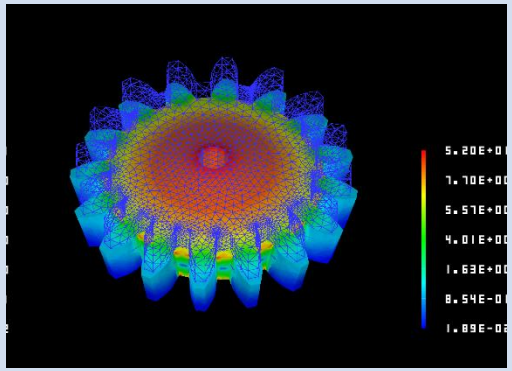
- 表示させる成分の指定



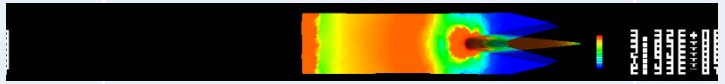
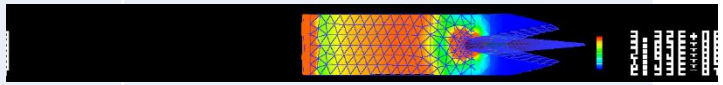
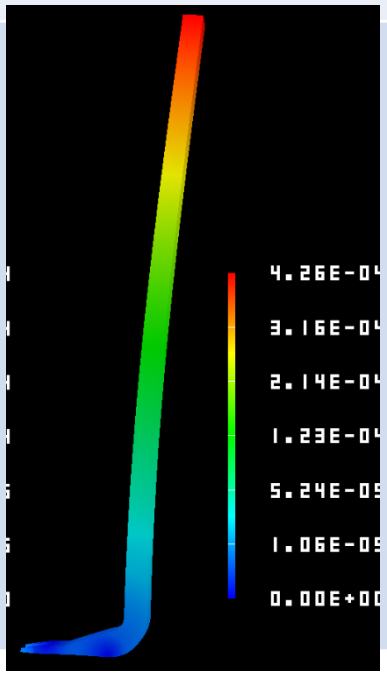
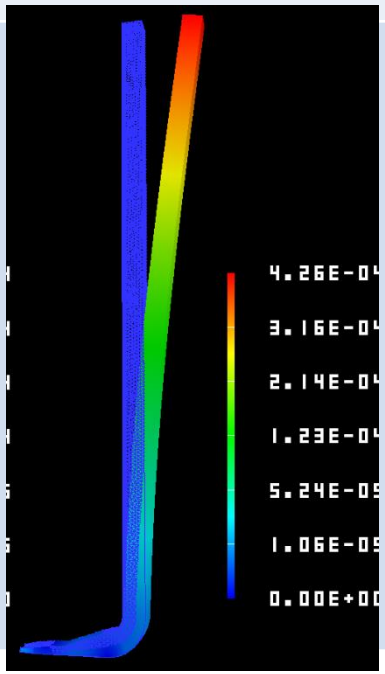
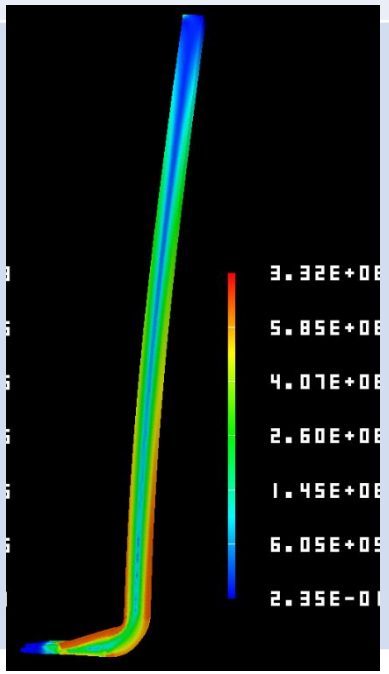
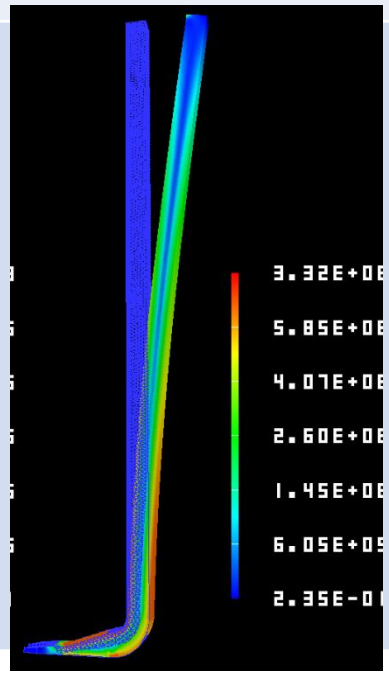
フランジモデル　flange (90,106節点，55,043要素，四面体2次要素)，1個の制御ファイルで8枚の絵を描画

	変位		ミーゼス応力	
	変形形状	変形前と変形後	変形形状	変形前と変形後
カメラ1				
カメラ2				

ギヤモデル　gear (58,273節点，36,311要素，四面体2次要素)，1個の制御ファイルで8枚の絵を描画

	変位		ミーゼス応力	
	変形形状	変形前と変形後	変形形状	変形前と変形後
カメラ1				
カメラ2				

バールモデル　bar (50,300節点，29,410要素，四面体1次要素)，1個の制御ファイルで8枚の絵を描画

	変位		ミーゼス応力	
	変形形状	変形前と変形後	変形形状	変形前と変形後
カメラ1				
カメラ2				

付録

この発表のための作業環境

東京大学（奥田研）のPCクラスタ

モデル，計算結果とその配置場所(1/2)

workspace/demo/flange　　90,106節点，55,043要素，四面体2次
workspace/demo/gear　　58,273節点，36,311要素，四面体2次
workspace/demo/bar　　50,300節点，29,410要素，四面体2次

各ディレクトリ下に計算結果のサブディレクトリ　ref0/　ref1/　ref2/
　ref0/ リファインなし(オリジナル)　4並列で計算
　ref1/ リファイン1回　　8並列で計算
　ref2/ リファイン2回　　16並列で計算

モデル，計算結果とその配置場所(2/2)

配置されているファイルは以下の通り

- hecmw_ctrl.dat　　全体制御ファイル
- モデル名.{msh, cnt}　　メッシュファイル，解析制御ファイル
- hecmw_part_ctrl.dat　　領域分割のための制御ファイル
- モデル名.inp　　領域分割図
- fstrRES/ resファイル　　– 部分領域ごとに {モデル名}.res.{領域ID}.1
  – rmergeで統合した計算結果 {モデル名}.res.1
- vis_out/ inpファイル　　ParaView用に，COMPLETE_REORDER_AVSで出力させた inpファイル

（リファインすると，inpファイル，resファイル共に，節点，要素もリファイン後の情報が記載される）

プログラムの配置

/usr/local/FrontISTR_V45/

FrontISTRの実行ファイル（fistr1）

パーティショナ（hecmw_part1）

可視化プログラム（hecmw_vis1）

並列計算時のresファイル統合プログラム（rmerge）

バージョンは2017/01/30時点の最新版

プログラムの実行手順

- 逐次計算　/usr/local/FrontISTR_V45/fistr1

- 並列計算　ジョブスケジューラを利用する

  job_fistr1.sh　FrontISTRを実行するジョブスクリプト

  job_part1.sh　パーティショナを実行するジョブスクリプト

  job_vis1.sh　可視化プログラムを実行するジョブスクリプト

ジョブの制御コマンド　qsub job_fistr1.sh，qstat，qdel ジョブID，など

ログ　{p/f}1_{モデル名} R{Refine回数}.{e/o}{ジョブID}　part1.log　fistr1.log

rmerge　（hecmw_ctrl.dat のあるディレクトリで）

usr/local/FrontISTR_V45/rmerge {モデル名}　（fstrRES以下にデータが出力される）

job_fistr1.sh　FrontISTRを実行するジョブスクリプト

```
#!/bin/bash
#PBS -q batch
#PBS -N f1_GearR0
#PBS -l walltime=00:30:00
#PBS -l nodes=1:ppn=4:hc


cd $PBS_O_WORKDIR
export OMP_NUM_THREADS=1


/usr/mpi/gcc/openmpi-2.0.1/bin/mpirun -machinefile $PBS_NODEFILE -np $PBS_NP -x
OMP_NUM_THREADS=$OMP_NUM_THREADS /usr/local/FrontISTR_V45/fistr1 2>&1 | tee fistr1.log
```

job_vis1.sh　可視化プログラムを実行するジョブスクリプト

```
#!/bin/bash
#PBS -q batch
#PBS -N f1_GearR0
#PBS -l walltime=00:30:00
#PBS -l nodes=1:ppn=4:hc


cd $PBS_O_WORKDIR
export OMP_NUM_THREADS=1


/usr/mpi/gcc/openmpi-2.0.1/bin/mpirun -machinefile $PBS_NODEFILE -np $PBS_NP -x
OMP_NUM_THREADS=$OMP_NUM_THREADS /usr/local/FrontISTR_V45/hecmw_vis1 2>&1 | tee
vis1.log
```