FrontISTRのカスタマイズ
～Element／Material追加および
ユーザーサブルーチン使用～

東京大学
新領域創成科学研究科
人間環境学専攻
橋本　学

2015年6月2日
第19回FrontISTR研究会

この講演では (1/2)

- 『FrontISTRに実装されている定式化を十分に理解し，解きたい問題に対してソースコードを自由にカスタマイズ(要素タイプを追加，材料を追加，ユーザサブルーチンを利用)できるようになること』を最終目標とします
- 第3回・第7回・第10回では弾性解析 (等方弾性体)，第11回では弾性解析 (直交異方弾性体)，第15回では熱応力解析，第16回では弾塑性解析，今回は第17回のFrontISTRプログラムのカスタマイズの続きを説明します
  - 第3回FrontISTR研究会　プログラミング編，2013/5/22開催
  - 第7回FrontISTR研究会　産業応用事例，有限変形定式化，ユーザーの声への対応編，2013/12/3開催
  - 第10回FrontISTR研究会　有限変形定式化と実装，Ver.4.3公開編，2014/2/21開催
  - 第11回FrontISTR研究会　機能・例題・定式化・プログラム解説編「直交異方弾性体を中心に」，2014/7/30開催
  - 第15回FrontISTR研究会　機能・例題・定式化・プログラム解説編「熱応力解析／弾塑性解析」，2014/10/31開催
  - 第16回FrontISTR研究会　ユーザー事例紹介編／機能・例題・定式化・プログラム解説編「弾塑性解析」，2015/01/16開催
  - 第17回FrontISTR研究会　ユーザー事例紹介編／機能・例題・定式化・プログラム解説編「FrontISTRのカスタマイズ (Element／Material追加およびユーザーサブルーチン使用)」，2015/03/23開催

この講演では (2/2)

- 「27節点六面体要素の追加」と「3次式のMooney-Rivlin超弾性体のu/p定式化 (8/1要素) の追加」を題材にして，FrontISTRプログラムに要素タイプや材料を追加します
  (ここでは，ユーザサブルーチンを使用しません)
- 最後に，ユーザマニュアルを見ながら，ユーザーサブルーチン使用方法を確認します

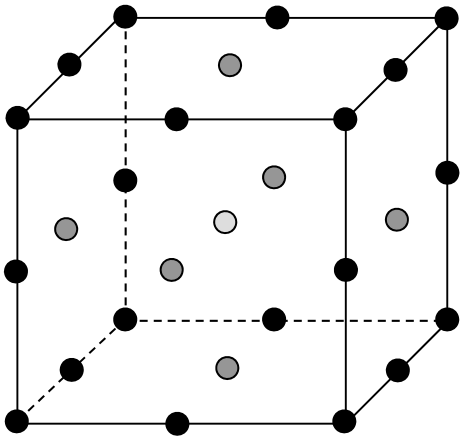
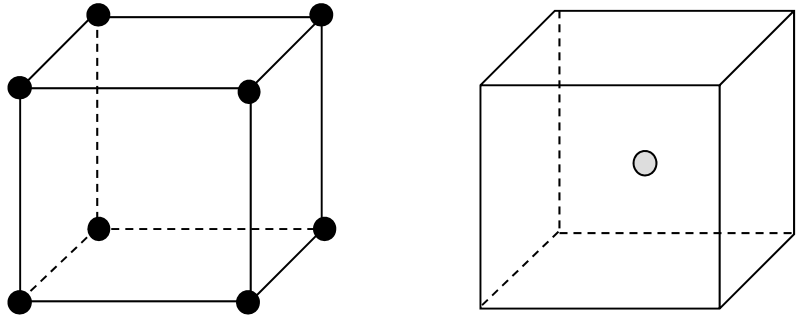


Fig. 27-node hexahedral element

要素タイプの追加



(a) displacement ***u*** (b) pressure *p*

Fig. 8/1 element (***u***: 8 nodes, *p*: 1 node)

要素タイプと材料の追加

内容

1. FrontISTRのソースプログラム
2. FrontISTRへの新しい要素タイプの追加
   「27節点六面体要素の追加」
3. FrontISTRへの新しい材料の追加
   「3次式のMooney-Rivlin超弾性体 (8/1要素) 」
4. FrontISTRのユーザサブルーチンの使用方法の確認

内容

1. FrontISTRのソースプログラム
2. FrontISTRへの新しい要素タイプの追加
   「27節点六面体要素の追加」
3. FrontISTRへの新しい材料の追加
   「3次式のMooney-Rivlin超弾性体 (8/1要素) 」
4. FrontISTRのユーザサブルーチンの使用方法の確認

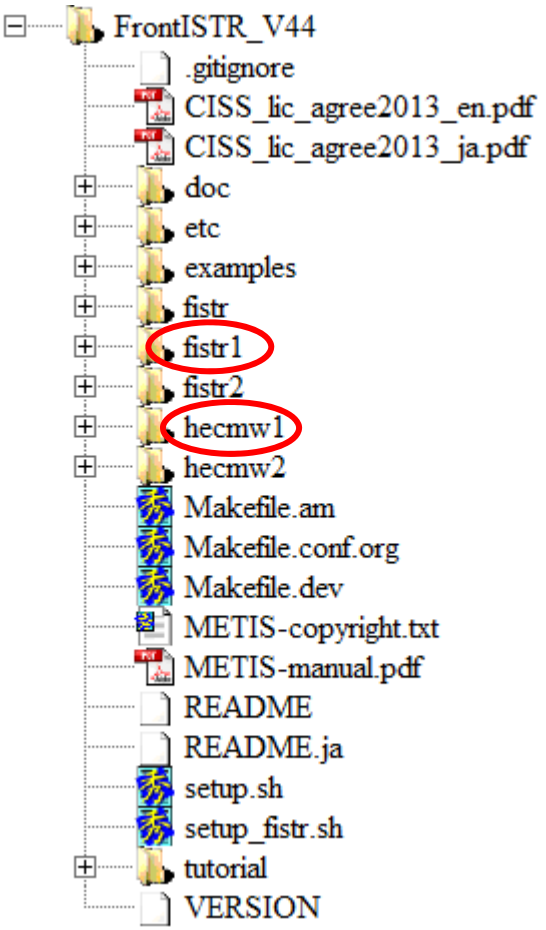
FrontISTR Ver.4.4のソースプログラム

ソースプログラムFrontISTR_V44.tar.gzは，
FrontISTR研究会のホームページ (http://www.multi.k.u-tokyo.ac.jp/FrontISTR/) からダウンロードできます

ファイル
FrontISTR_V44.tar.gz

解凍

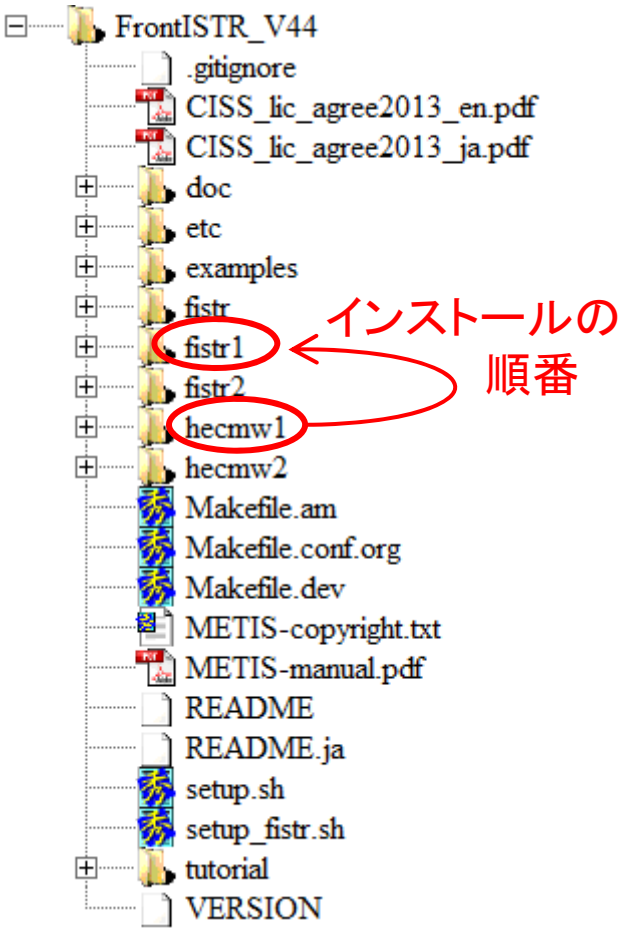
ディレクトリ
FrontISTR_V44



この講演では，ディレクトリfistr1と
ディレクトリhecmw1の中のファイルを
対象とします

FrontISTRのインストール手順

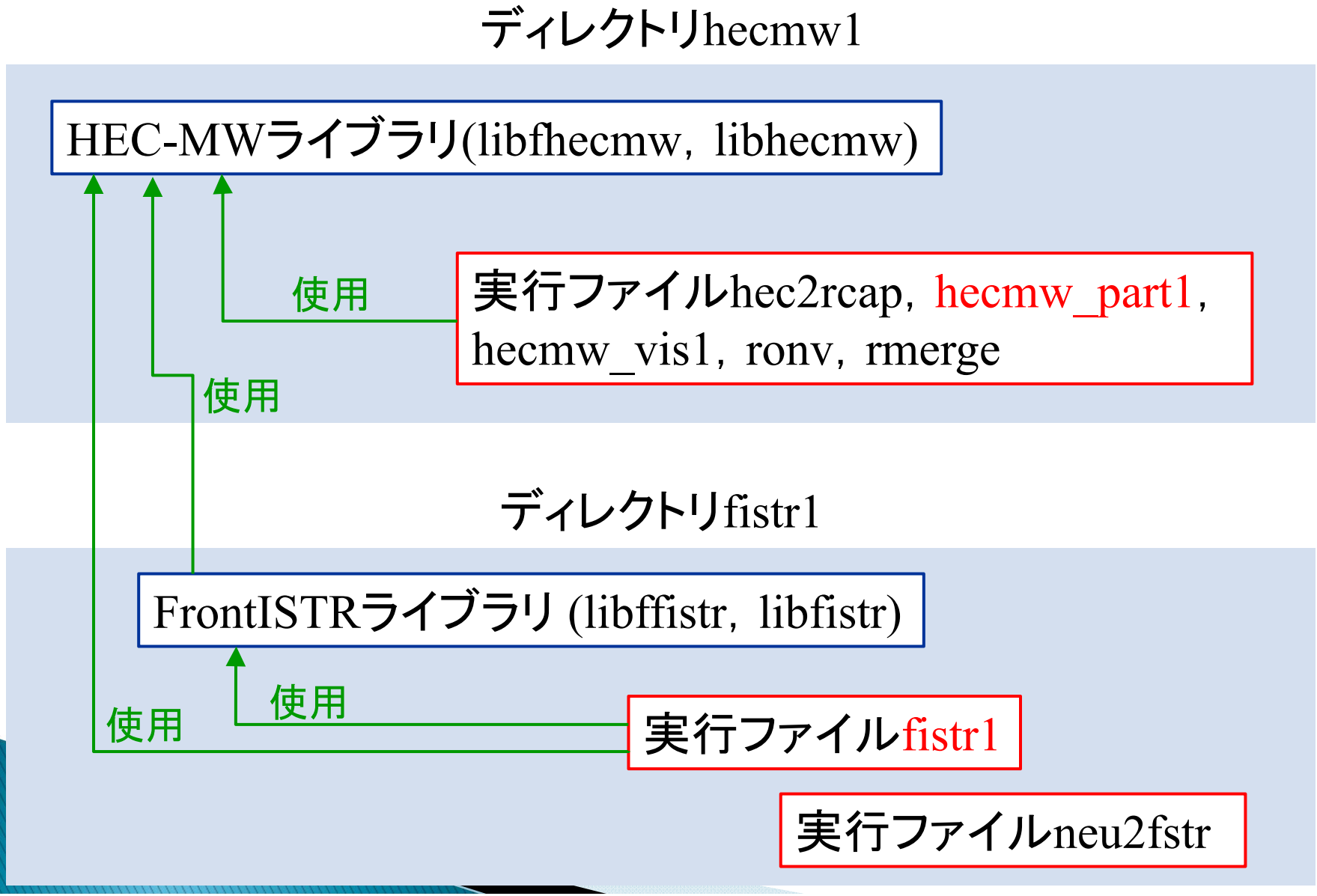
(1) ディレクトリhecmw1内のソースプログラムをコンパイルする
(2) HEC-MWライブラリ(libfhecmw, libhecmw) が生成される
(3) 実行ファイルhec2rcap, hecmw_part1, hecmw_vis1, ronv, rmergeが生成される
(4) ディレクトリfistr1内のソースプログラムをコンパイルする
(5) FrontISTRライブラリ (libffistr, libfistr) が生成される
(6) 実行ファイルfistr1, neu2fstrが生成される



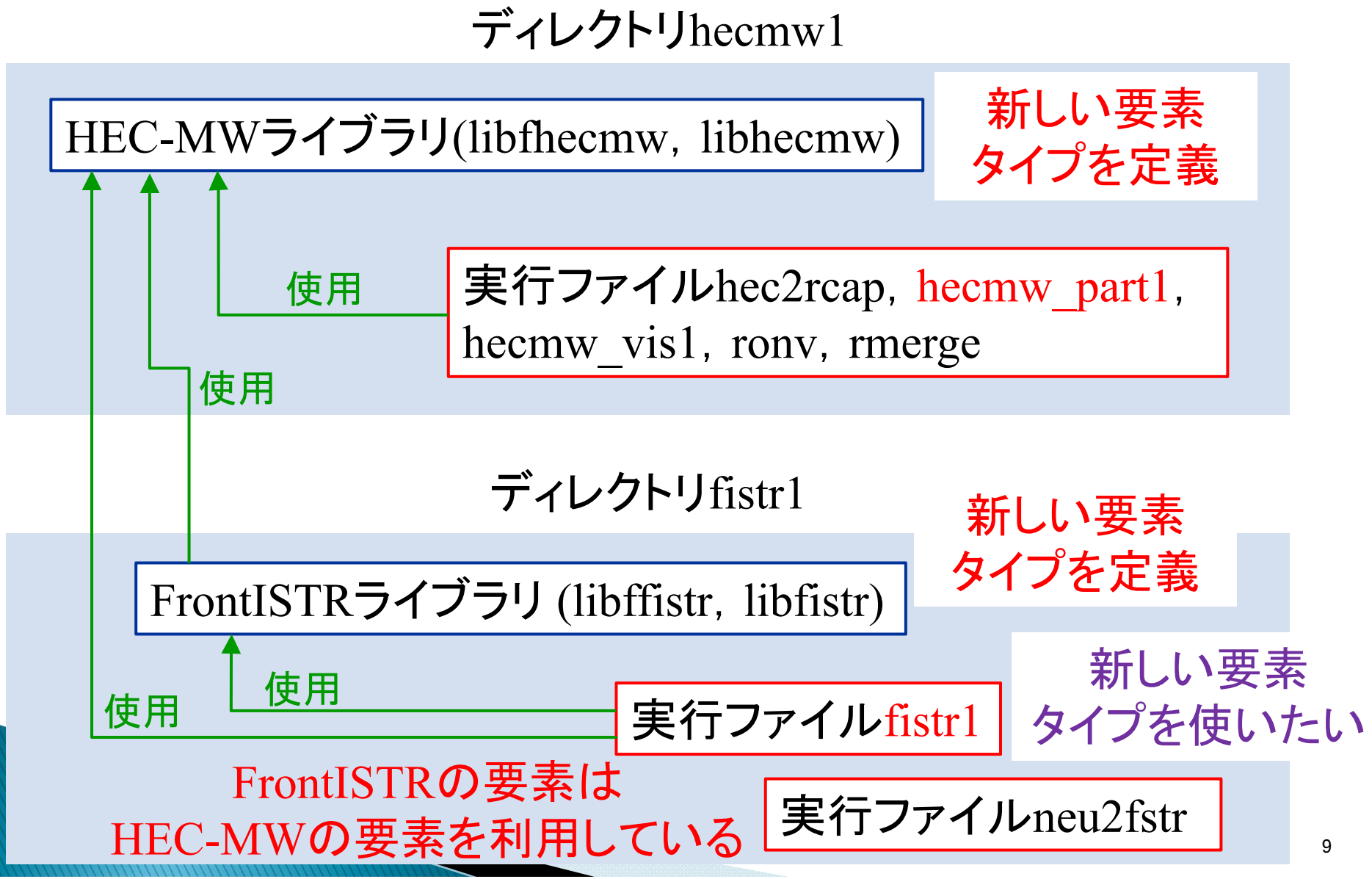
(注1) インストールオプションで--with-toolsを指定しない場合，hec2rcap, hecmw_part1, hecmw_vis1, ronv, rmergeは生成されません

(注2) fistr1がFrontISTRの実行ファイル，hecmw_part1がパーティショナの実行ファイルです

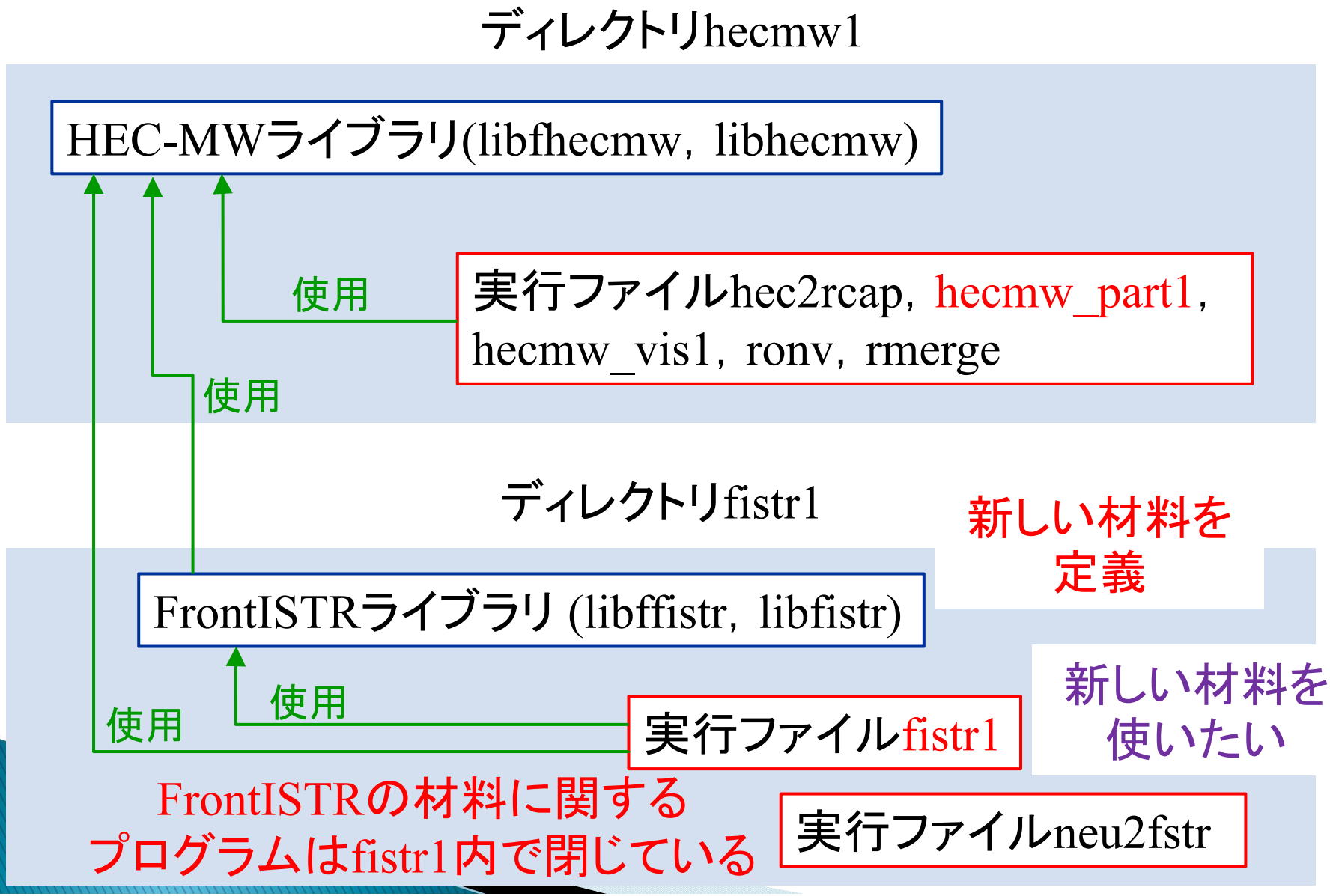
hecmw1とfistr1のライブラリと実行ファイル



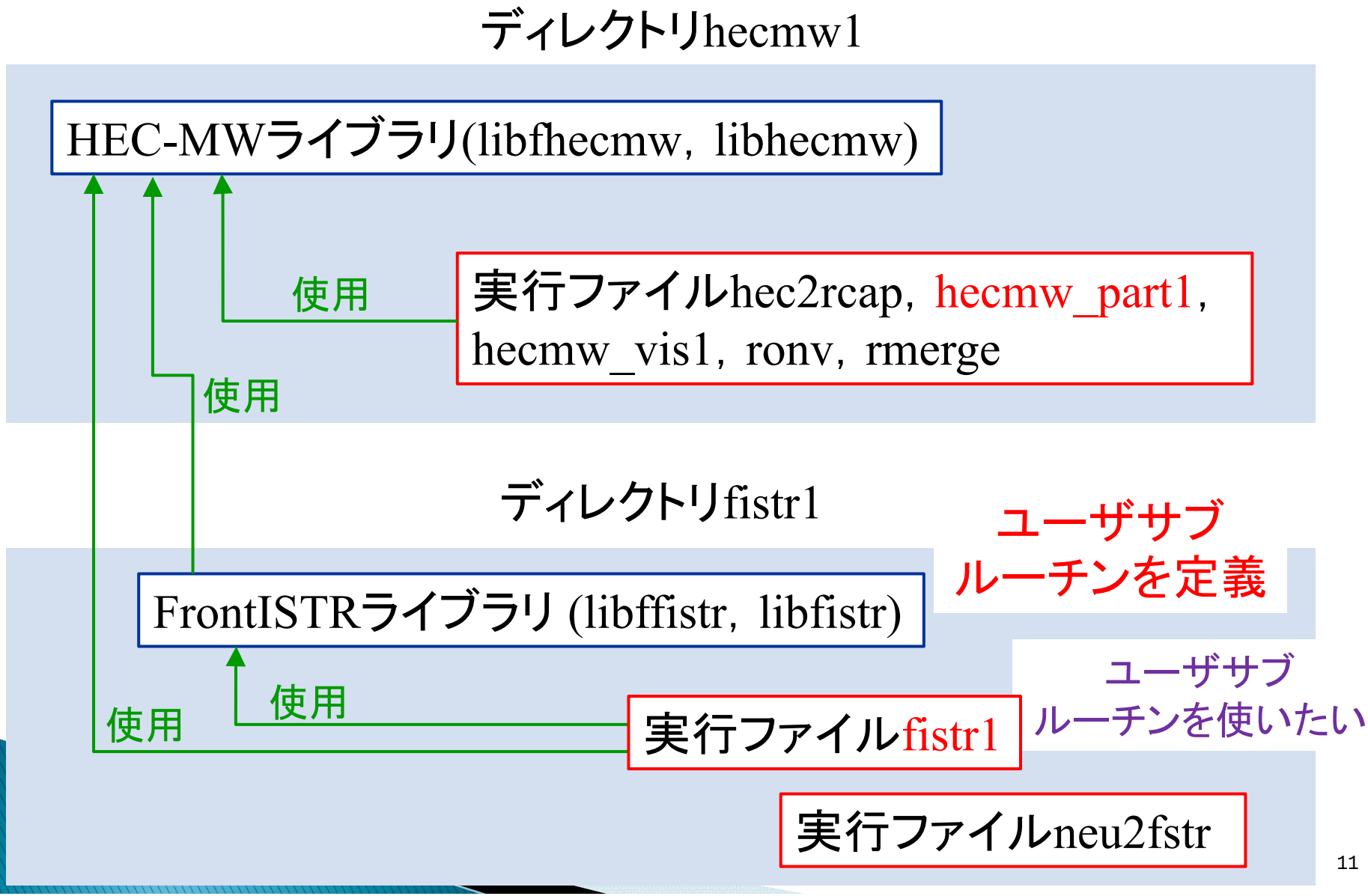
FrontISTRに新しい要素タイプを追加する場合



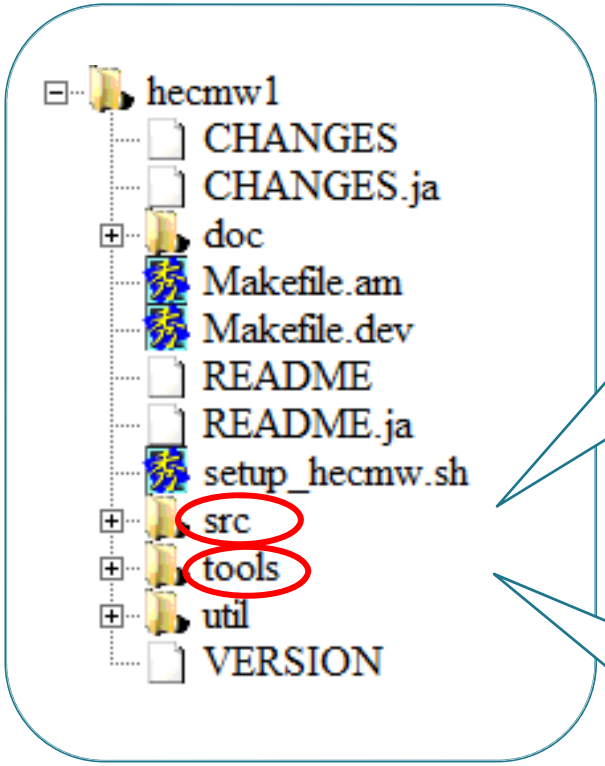
FrontISTRに新しい材料を追加する場合

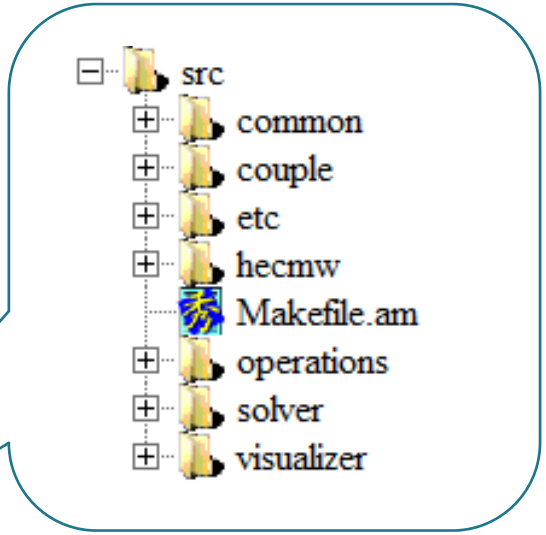


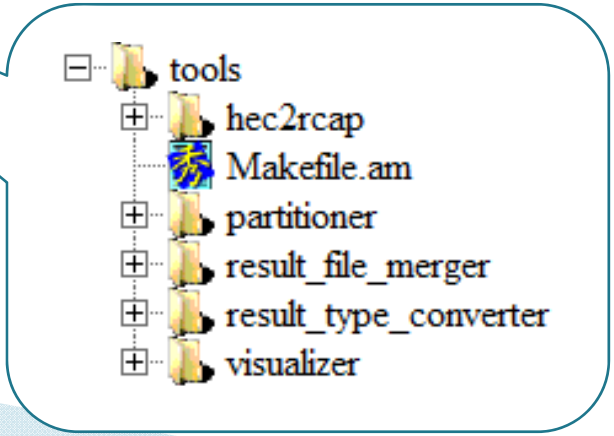
FrontISTRのユーザサブルーチンを定義する場合



ディレクトリhecmw1





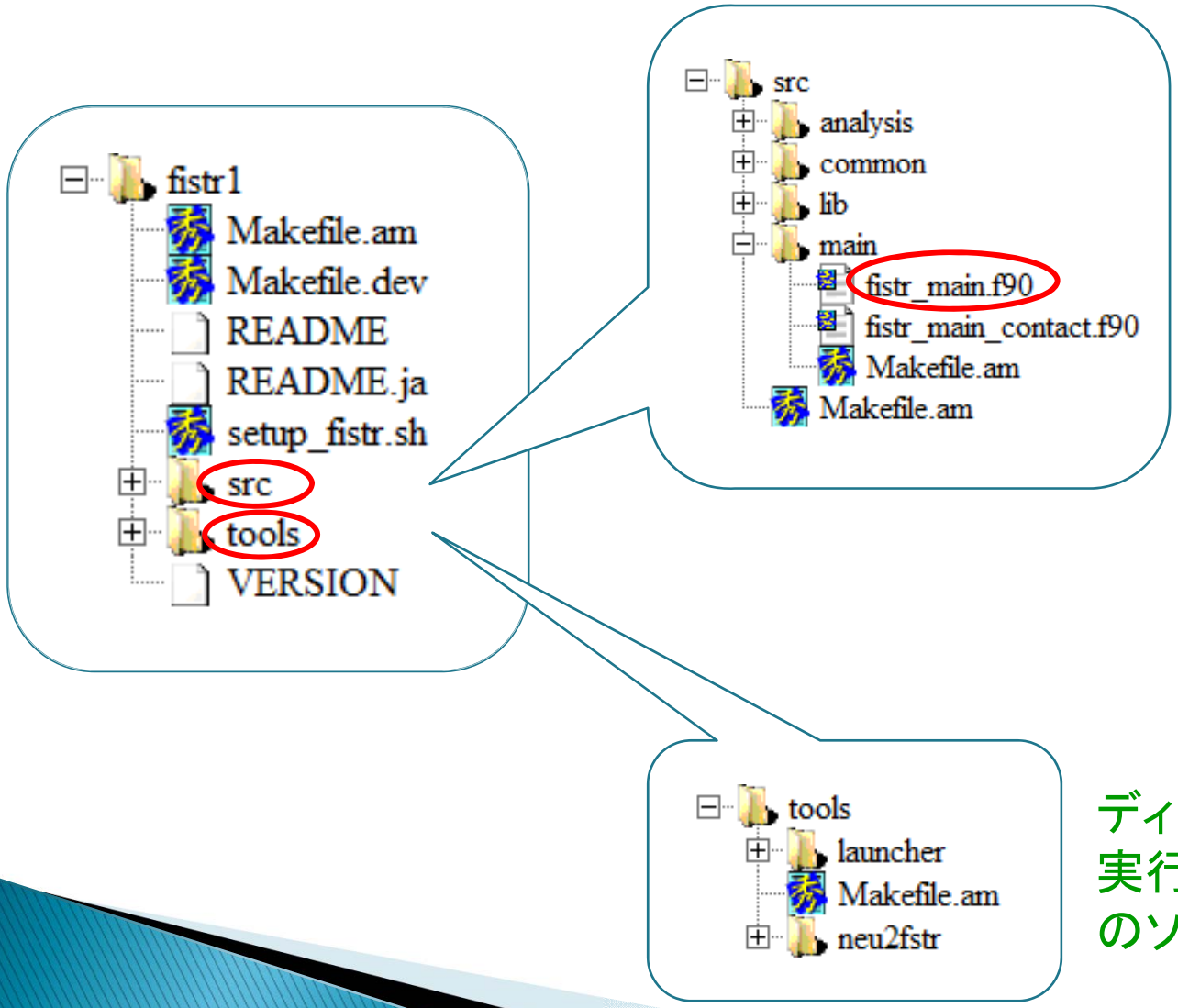


ディレクトリsrcの下が
HEC-MWライブラリ
(libfhecmw, libhecmw)
のソースプログラム

重要なディレクトリは
「common」,
「hecmw」,
「solver」,
「visualizer」
の四つ

ディレクトリtoolsの
下が実行ファイル
hec2rcap,
hecmw_part1,
hecmw_vis1, ronv,
rmerge
のソースプログラム

ディレクトリfistr1 (1/2)



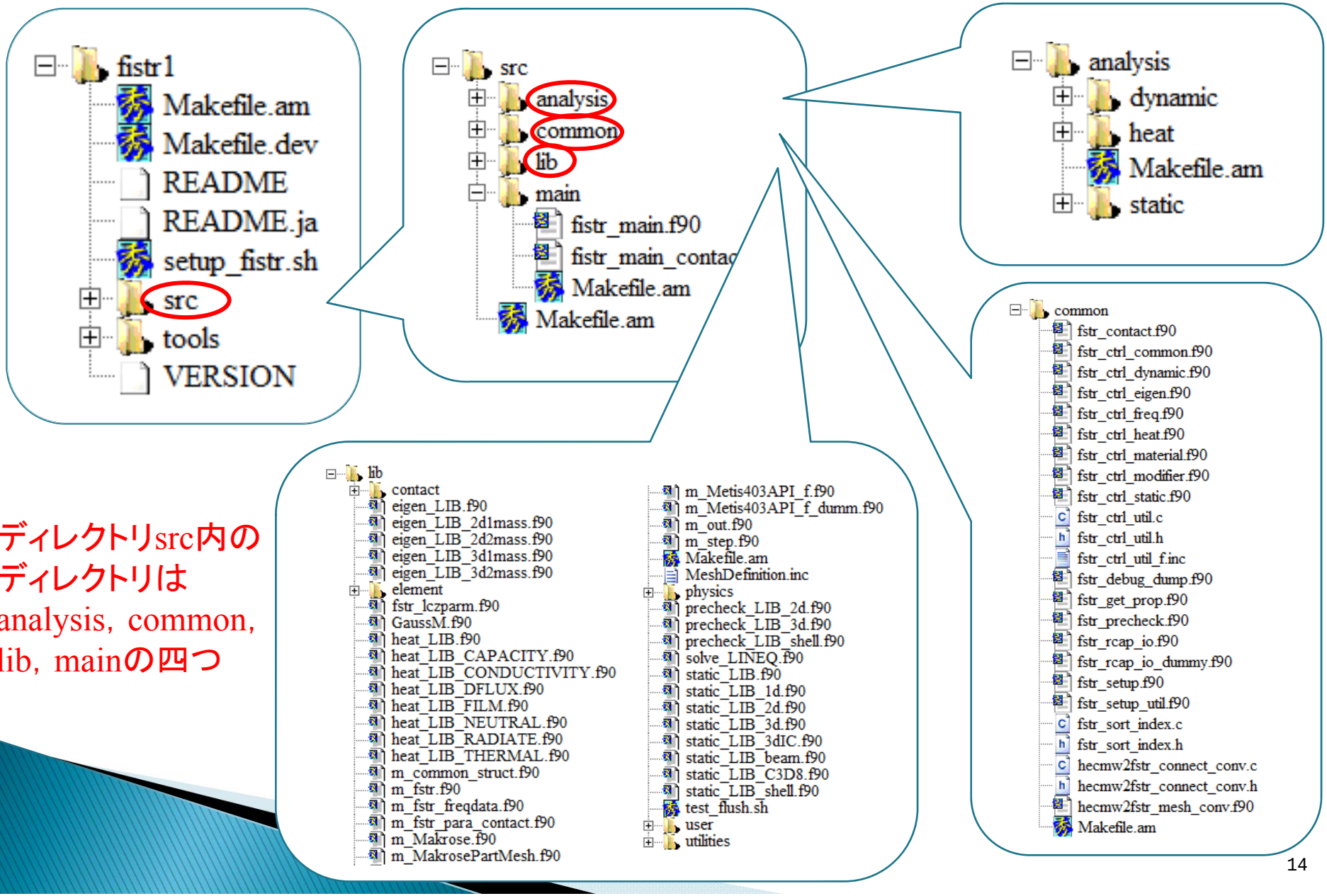
ディレクトリsrcの下が
FrontISTRライブラリ
(libffistr, libfistr) の
ソースプログラム

ディレクトリsrc内の
ディレクトリは
analysis, common,
lib, mainの四つ

ディレクトリsrc/mainの
下が実行ファイルfistr1
のソースプログラム

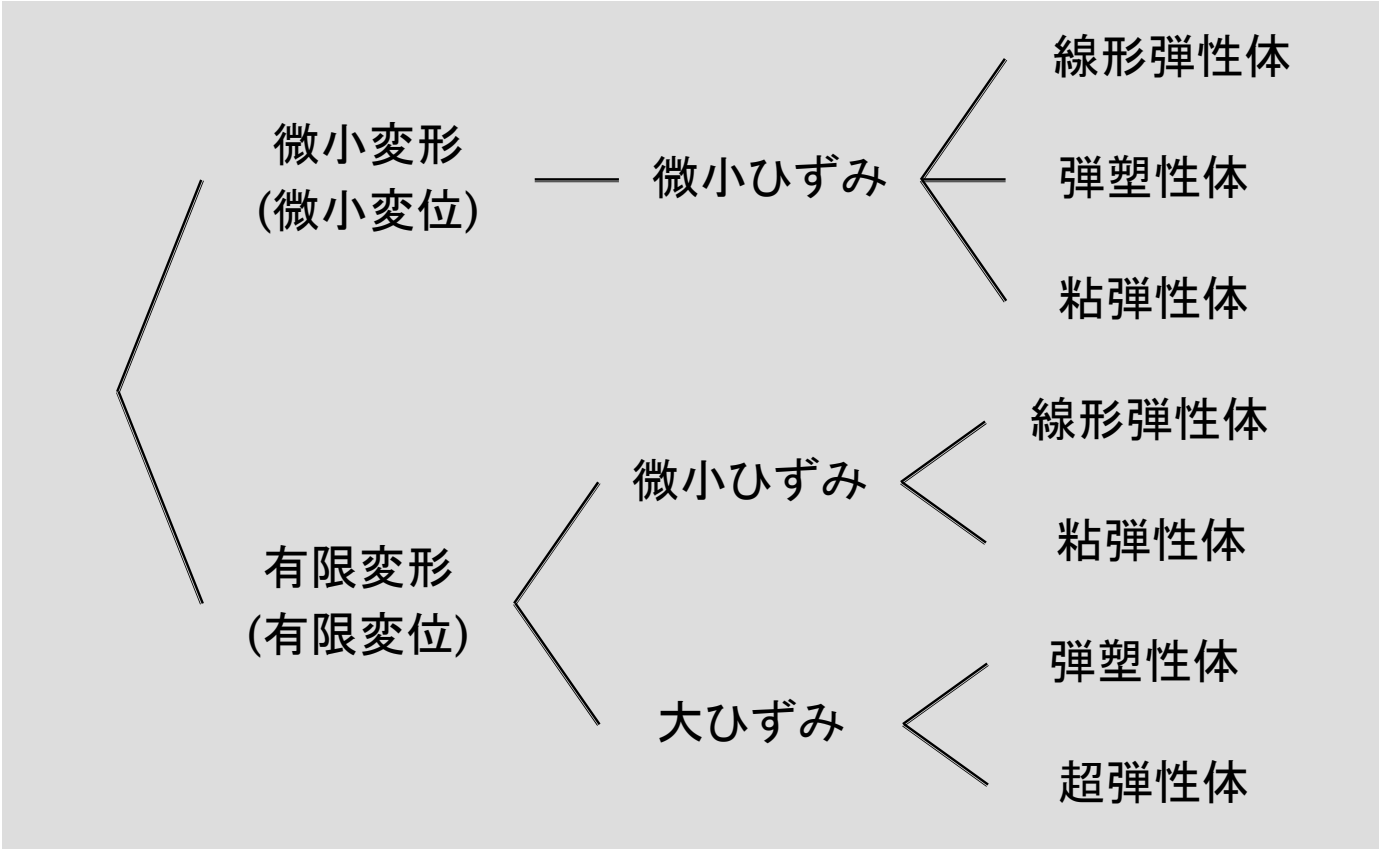
ディレクトリtoolsの下が
実行ファイルneu2fstr
のソースプログラム

ディレクトリfistr1 (2/2)



ディレクトリsrc内のディレクトリは
analysis，common，lib，mainの四つ

静応力解析の分類 (1/2)



有限変形

$${}_0^t\boldsymbol{E} = \frac{1}{2}\left\{({}^0\nabla \otimes {}^t\boldsymbol{u}) + ({}^0\nabla \otimes {}^t\boldsymbol{u})^{\mathrm{T}} + ({}^0\nabla \otimes {}^t\boldsymbol{u}) \cdot ({}^0\nabla \otimes {}^t\boldsymbol{u})^{\mathrm{T}}\right\}$$

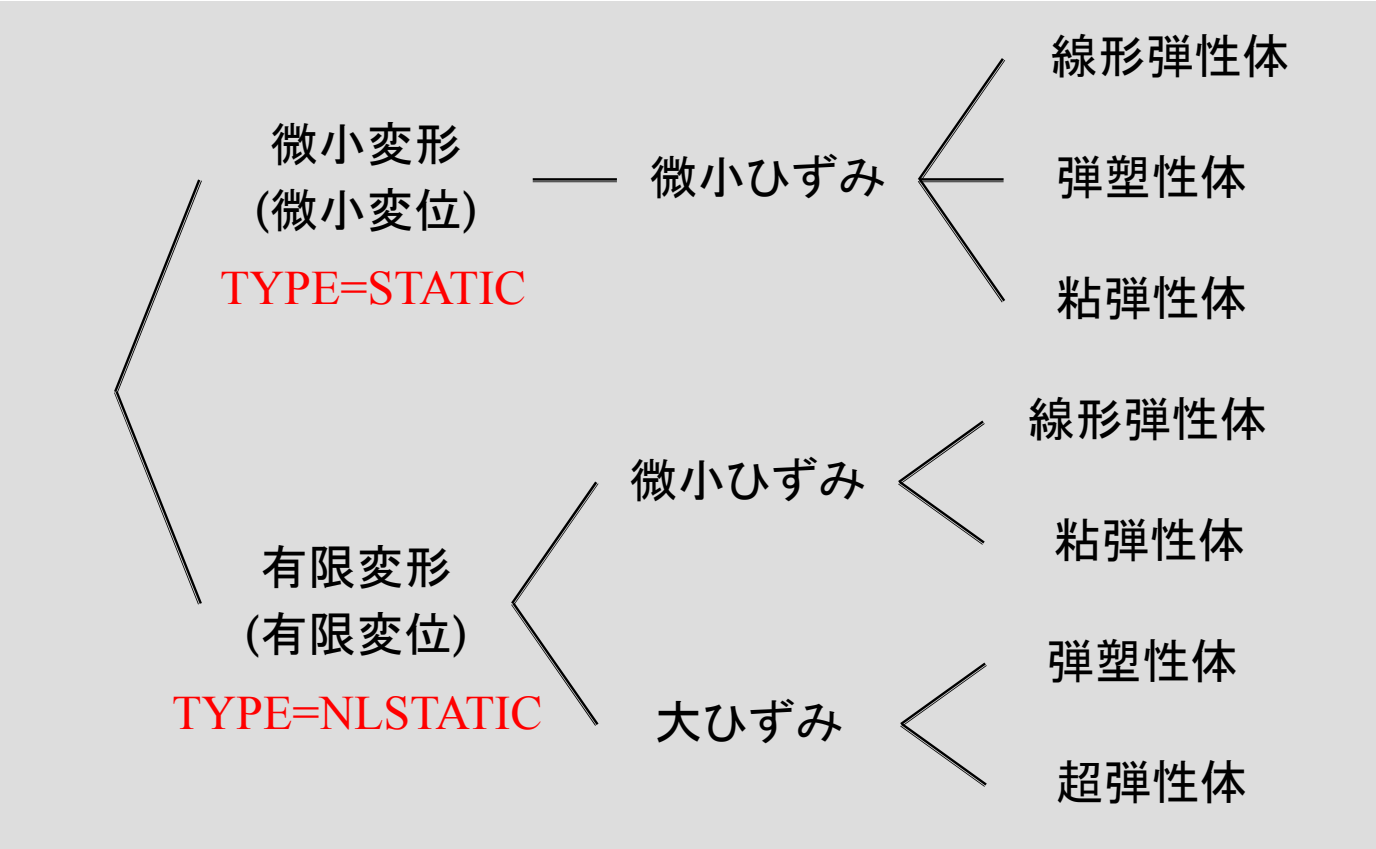
ひずみ　　変位こう配の2次項がある

大ひずみ

$${}_0^t\boldsymbol{S} = f({}_0^t\boldsymbol{E},\ {}_0^t\boldsymbol{E} \cdot {}_0^t\boldsymbol{E},\ \ldots)$$

応力　ひずみの2次以上の項がある

静応力解析の分類 (2/2)



有限変形静応力解析を行う場合，FrontISTRの解析制御データ(*.cntファイル) 内で!SOLUTION,TYPE=NLSTATICと記述します

微小変形静解析の流れ

```
[main/fistr_main.f90] PROGRAM fstr_main        ・・・メインプログラム
    hecmw_init()
    hecmw_get_mesh()
    [main/fistr_main.f90] fstr_init()        ・・・ 変数初期化・入力データ読み込み
        hecmw_nullify_matrix()
        hecmw_nullify_result_data()
        [main/fistr_main.f90] fstr_init_file()
        hecmw_mat_con()
        [main/fistr_main.f90] fstr_condition()
            hecmw_ctrl_get_control_file()
    [main/fistr_main.f90] fstr_linear_static_analysis()        ・・・ 線形静解析用のルーチンへ
        [analysis/static/fstr_solve_LINEAR.f90] m_fstr_LINEAR::fstr_solve_LINEAR()
            [analysis/static/static_mat_ass.f90] m_static_mat_ass::fstr_mat_ass()        ・・・ 剛性マトリックスの作成
                [analysis/static/static_mat_ass_main.f90] m_static_mat_ass_main::fstr_mat_ass_main()
                    hecmw_mat_clear()
                    [analysis/static/static_mat_ass_main.f90] m_static_mat_ass_main::fstr_local_stf_create()
                        【ここで要素タイプに応じて分岐します】
                        [analysis/static/static_LIB_〇〇〇.f90] m_static_LIB_〇〇〇::STF_〇〇〇()    ・・・要素剛性マトリックスの計算
                            [lib/physics/calMatMatrix.f90] m_MatMatrix::MatlMatrix()    ・・・ Dマトリックス
                                【ここで材料タイプに応じて分岐します】
                                [lib/physics/〇〇〇.f90] m_〇〇〇::cal 〇〇〇()
                    hecmw_mat_ass_elem()    ・・・ 要素剛性マトリックスをassemble
            [analysis/static/fstr_ass_load.f90] m_fstr_ass_load::fstr_ass_load()    ・・・ 外力ベクトルの計算
            [analysis/static/fstr_AddBC.f90] m_fstr_AddBC::fstr_AddBC()    ・・・ 境界条件の処理
            hecmw_allREDUCE_R1()
            [lib/solve_LINEQ.f90] m_solve_LINEQ::solve_LINEQ()    ・・・ 線形ソルバーによる求解
                hecmw_solve_〇〇〇()
            hecmw_update_〇_R()
            [analysis/static/fstr_Update.f90] m_fstr_Update::fstr_Update3D()
                【ここで要素タイプに応じて分岐します】
                [lib/static_LIB_〇〇〇.f90] m_static_LIB_〇〇〇::UpdateST_〇〇〇()    ・・・ 応力の計算
                [lib/static_LIB_〇〇〇.f90] m_static_LIB_〇〇〇::STF_〇〇〇()    ・・・要素剛性マトリックスの計算
                    [lib/physics/calMatMatrix.f90] m_MatMatrix::MatlMatrix()    ・・・ Dマトリックス
                        【ここで材料タイプに応じて分岐します】
                        [lib/physics/〇〇〇.f90] m_〇〇〇::cal 〇〇〇()
            [analysis/static/static_output.f90] m_static_output :: fstr_static_Output()    ・・・ 結果の出力
                [analysis/static/static_make_result.f90] m_static_make_result::fstr_write_static_result()
    [main/fistr_main.f90] fstr_main::fstr_finalize()    ・・・ 変数の削除
    hecmw_finalize()
```

FrontISTRの解析制御データ
(*.cntファイル) 内で
!SOLUTION,TYPE=STATICと記述

[ディレクトリ/ファイル名]
モジュール名::サブルーチン名()を意味しています

有限変形静解析の流れ (1/2)

FrontISTRの解析制御データ
(*.cntファイル) 内で
!SOLUTION,TYPE=NLSTATICと記述

[main/fistr_main.f90] PROGRAM fstr_main ・・・ メインプログラム
hecmw_init()
hecmw_get_mesh()
[main/fistr_main.f90] fstr_init() ・・・ 変数初期化・入力データ読み込み
hecmw_nullify_matrix ()
hecmw_nullify_result_data ()
[main/fistr_main.f90] fstr_init_file()
hecmw_mat_con ()
[main/fistr_main.f90] fstr_condition()
hecmw_ctrl_get_control_file ()
[main/fistr_main.f90] fstr_nonlinear_static_analysis() ・・・ 非線形静解析用のルーチンへ
[analysis/static/fstr_solve_NLGEOM.f90] m_fstr_solve_NLGEOM::fstr_solve_nlgeom() ・・・ 荷重増分のループ
[analysis/static/fstr_solve_Nonlinear.f90] m_fstr_NonLinearMethod::fstr_Newton()
} loading step (substep) 1
[analysis/static/fstr_solve_Nonlinear.f90] m_fstr_NonLinearMethod::fstr_Newton()
} loading step (substep) 2
....
....
[analysis/static/static_output.f90] m_static_output::fstr_static_Output() ・・・ 計算結果の出力
[analysis/static/static_make_result.f90] m_static_make_result::fstr_write_static_result()
[main/fistr_main.f90] fstr_finalize() ・・・ 変数の削除
hecmw_finalize ()

[ディレクトリ/ファイル名] モジュール名::サブルーチン名()を意味しています

有限変形静解析の流れ (2/2)

[analysis/static/fstr_solve_Nonlinear.f90] m_fstr_NonLinearMethod::fstr_Newton()　・・・Newton-Raphson反復

hecmw_allreduce_I1()
[analysis/static/fstr_ass_load.f90] m_fstr_ass_load::fstr_ass_load()　・・・ 外力ベクトルの計算
[analysis/static/fstr_StfiffMatrix.f90] m_fstr_StiffMatrix::fstr_StiffMatrix()　・・・ 接線剛性マトリックスの計算
　　hecmw_mat_clear()
　　【ここで要素タイプに応じて分岐します】
　　[lib/static_LIB_○○○.f90] m_static_LIB_○○○::STF_○○○()　・・・要素接線剛性マトリックスの計算
　　　[lib/physics/calMatMatrix.f90] m_MatMatrix::MatlMatrix()　・・・ Dマトリックス
　　　【ここで材料タイプに応じて分岐します】
　　　[lib/physics/○○○.f90] m_○○○::cal○○○()
　　hecmw_mat_ass_elem()　・・・ 要素接線剛性マトリックスをassemble
[analysis/static/fstr_AddBC.f90] m_fstr_AddBC::fstr_AddBC()　・・・ 境界条件の処理
[lib/solve_LINEQ.f90] m_solve_LINEQ::solve_LINEQ()　・・・ 線形ソルバーによる求解
hecmw_update_○_R()
[analysis/static/fstr_Update.f90] m_fstr_Update::fstr_UpdateNewton()　・・・ 内力ベクトルの計算
　　【ここで要素タイプに応じて分岐します】
　　[lib/static_LIB_○○○.f90] m_static_LIB_○○○::Update_○○○()・・・応力・要素内力ベクトルの計算
　　　[lib/physics/calMatMatrix.f90] m_MatMatrix::MatlMatrix()　・・・ Dマトリックス
　　　【ここで材料タイプに応じて分岐します】
　　　[lib/physics/○○○.f90] m_○○○::cal○○○()
[analysis/static/fstr_Residual.f90] m_fstr_Residual::fstr_Update_NDForce()　・・・ 残差ベクトルの計算
hecmw_allreduce_R1()
[analysis/static/fstr_StfiffMatrix.f90] m_fstr_StiffMatrix::fstr_StiffMatrix()
　　….
hecmw_allreduce_R1()
[analysis/static/fstr_StfiffMatrix.f90] m_fstr_StiffMatrix::fstr_StiffMatrix()
　　….

Newton-Raphson iteration (iter) 1

Newton-Raphson iteration (iter) 2

[ディレクトリ/ファイル名] モジュール名::サブルーチン名()を意味しています

内容

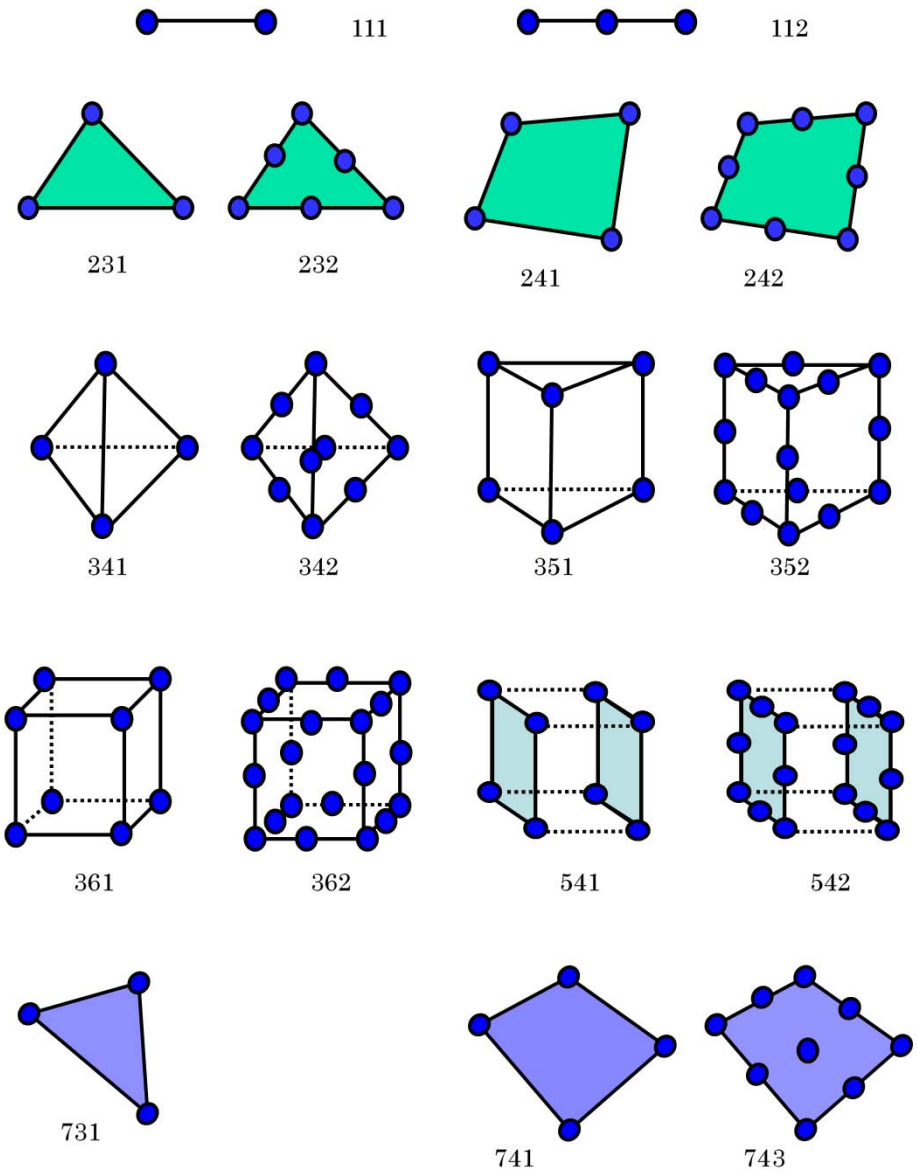
1. FrontISTRのソースプログラム
2. FrontISTRへの新しい要素タイプの追加
   「27節点六面体要素の追加」
3. FrontISTRへの新しい材料の追加
   「3次式のMooney-Rivlin超弾性体 (8/1要素) 」
4. FrontISTRのユーザサブルーチンの使用方法の確認

FrontISTRで使用可能な要素タイプ (1/2)

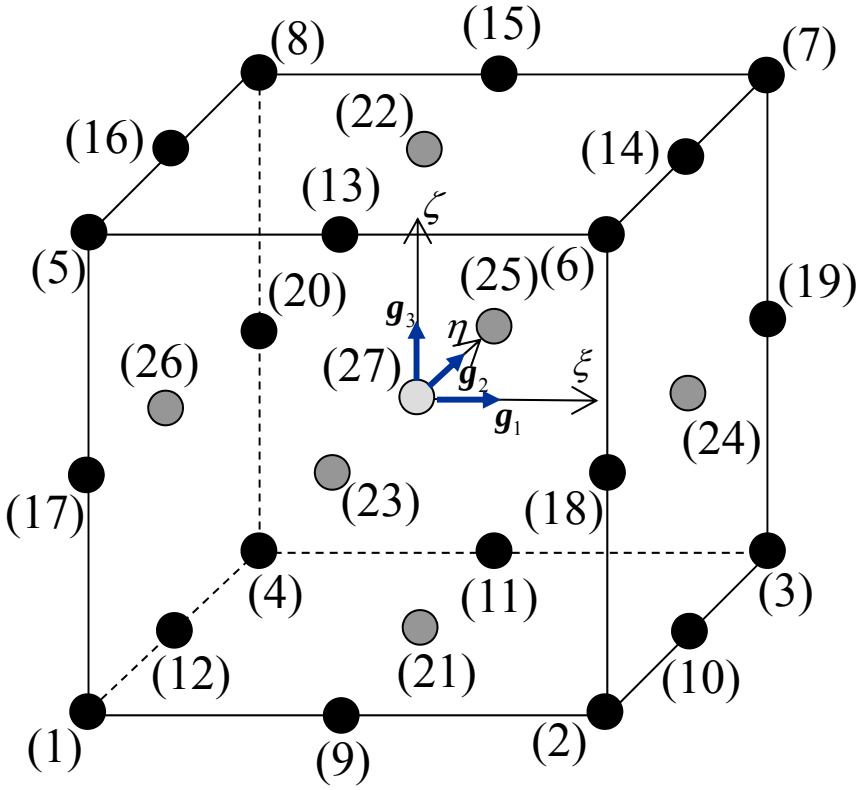
要素種類	要素タイプ番号	節点数	節点自由度数	説明
線要素	111	2	3	2節点リンク要素
	112	3	3	3節点リンク要素
平面要素	231	3	3	三角形1次要素
	232	6	3	三角形2次要素
	241	4	3	四角形1次要素
	242	8	3	四角形2次要素 (Serendipity族)
ソリッド要素	301	2	3	2節点トラス要素
	341	4	3	四面体1次要素
	342	10	3	四面体2次要素
	351	6	3	プリズム1次要素
	352	15	3	プリズム2次要素
	361	8	3	六面体1次要素
	362	20	3	六面体2次要素 (Serendipity族)
インターフェース要素	541	4×2	3	四角形面1次要素
	542	8×2	3	四角形面2次要素
梁要素	611*	2	6	2節点梁要素 (Bernoulli-Euler梁)
	641**	2×2	3	2節点梁要素 (Bernoulli-Euler梁)
シェル要素	731*	3	6	三角形1次要素 (MITC3シェル)
	761**	3×2	3	三角形1次要素 (MITC3シェル)
	741*	4	6	四角形1次要素 (MITC4シェル)
	781**	4×2	3	四角形1次要素 (MITC4シェル)
	743	9	6	四角形2次要素 (MITC9シェル)

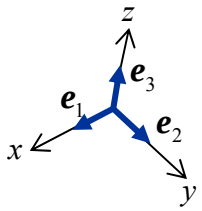
(注意) 解析メッシュにソリッド要素，梁要素，シェル要素が混在する場合，
節点自由度数を3に揃えるため，*ではなく**の要素タイプ番号を使用してください

FrontISTRで使用可能な要素タイプ (2/2)



六面体27節点要素





要素タイプ363を
追加します

$$\boldsymbol{g}_1=\frac{\partial \boldsymbol{x}}{\partial \xi},\ \boldsymbol{g}_2=\frac{\partial \boldsymbol{x}}{\partial \eta},\ \boldsymbol{g}_3=\frac{\partial \boldsymbol{x}}{\partial \zeta}$$

: Covariant basis vectors

$$N^{(\alpha)}=\left\{\frac{\xi^{(\alpha)}\xi}{2}\left(1+\xi^{(\alpha)}\xi\right)+\left(1-\xi^{(\alpha)2}\right)\left(1-\xi^{2}\right)\right\}$$
$$\left\{\frac{\eta^{(\alpha)}\eta}{2}\left(1+\eta^{(\alpha)}\eta\right)+\left(1-\eta^{(\alpha)2}\right)\left(1-\eta^{2}\right)\right\}$$
$$\left\{\frac{\zeta^{(\alpha)}\zeta}{2}\left(1+\zeta^{(\alpha)}\zeta\right)+\left(1-\zeta^{(\alpha)2}\right)\left(1-\zeta^{2}\right)\right\}$$

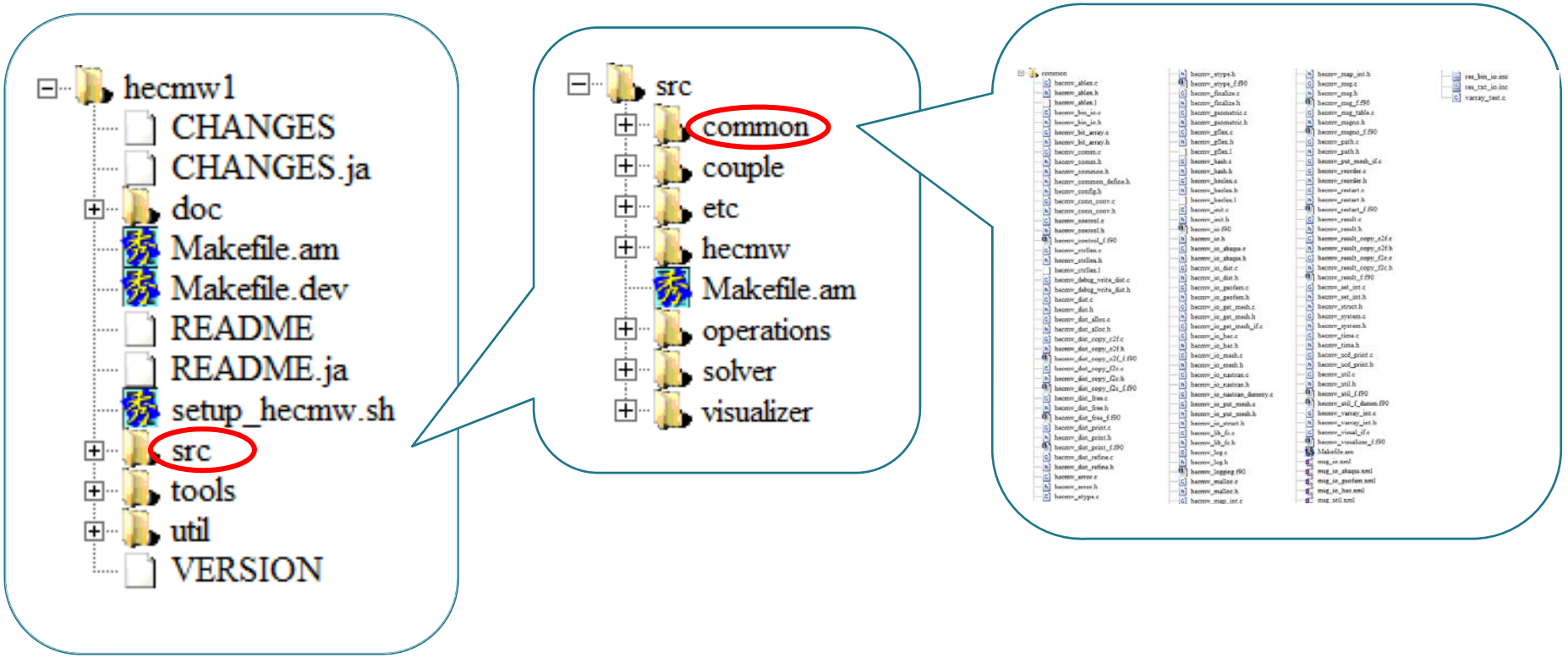
$$\left\{\xi^{(\alpha)}\,|\alpha=1,2,\ldots,27\right\}=\begin{Bmatrix}-1,&1,&1,-1,-1,&1,&1,-1,\\0,&1,&0,-1,&0,&1,&0,-1,\\-1,&1,&1,-1,&0,&0,&0,&1,\\0,-1,&0\end{Bmatrix}$$

$$\left\{\eta^{(\alpha)}\,|\alpha=1,2,\ldots,27\right\}=\begin{Bmatrix}-1,-1,&1,&1,-1,-1,&1,&1,\\-1,&0,&1,&0,-1,&0,&1,&0,\\-1,-1,&1,&1,&0,&0,-1,&0,\\1,&0,&0\end{Bmatrix}$$

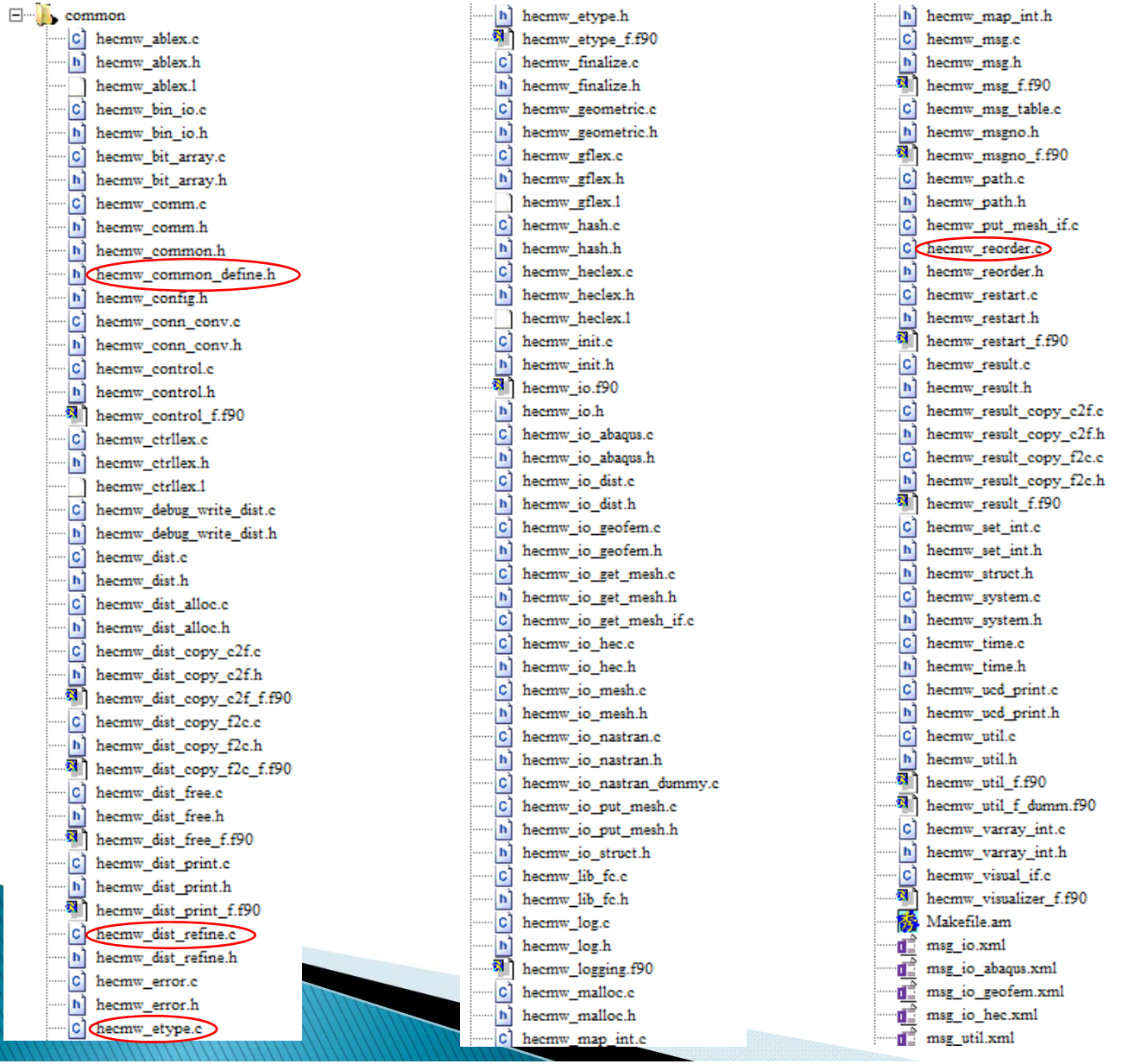
$$\left\{\zeta^{(\alpha)}\,|\alpha=1,2,\ldots,27\right\}=\begin{Bmatrix}-1,-1,-1,-1,&1,&1,&1,&1,\\-1,-1,-1,-1,&1,&1,&1,&1,\\0,&0,&0,&0,-1,&1,&0,&0,\\0,&0,&0\end{Bmatrix}$$

: Shape function

ディレクトリhecmw1/src/common (1/2)



ディレクトリhecmw1/src/common (2/2)



四つのファイル
hecmw_common_define.h,
hecmw_etype.c,
hecmw_reorder.c,
hecmw_dist_refine.cを変更します

hecmw_common_define.hの変更箇所

定数名	意味
HECMW_ETYPE_MAX	HEC-MWの要素タイプ番号最大値
HECMW_ETYPE_XXX	HEC-MWの要素タイプ番号
HECMW_GEOFEM_ETYPE_MAX	GeoFEMでの要素タイプ番号最大値
HECMW_GEOFEM_ETYPE_XXX	GeoFEMでの要素タイプ番号
HECMW_MESH_ETYPE_MAX	メッシュユーティリティにおける要素タイプ番号
HECMW_MESH_ETYPE_XXX	メッシュユーティリティにおける要素タイプ番号
HECMW_UCD_LABEL_XXX	UCDデータの要素ラベル
HECMW_MAX_NODE_MAX	HEC-MWの要素の節点数最大値
HECMW_MAX_NODE_XXX	HEC-MWの要素の節点数
HECMW_MAX_EDGE_MAX	HEC-MWの要素のエッジ数最大値
HECMW_MAX_EDGE_XXX	HEC-MWの要素のエッジ数
HECMW_MAX_SURF_MAX	HEC-MWの要素の面数最大値
HECMW_MAX_SURF_XXX	HEC-MWの要素の面数
HECMW_MAX_TSUF_MAX	HEC-MWの要素の三角形面数最大値
HECMW_MAX_TSUF_XXX	HEC-MWの要素の三角形面数
HECMW_MAX_QSUF_ MAX	HEC-MWの要素の四角形面数の最大値
HECMW_MAX_QSUF_XXX	HEC-MWの要素の四角形面数
HECMW_MESH_DOF_XXX	HEC-MWの節点自由度数

hecmw_etype.cの変更箇所

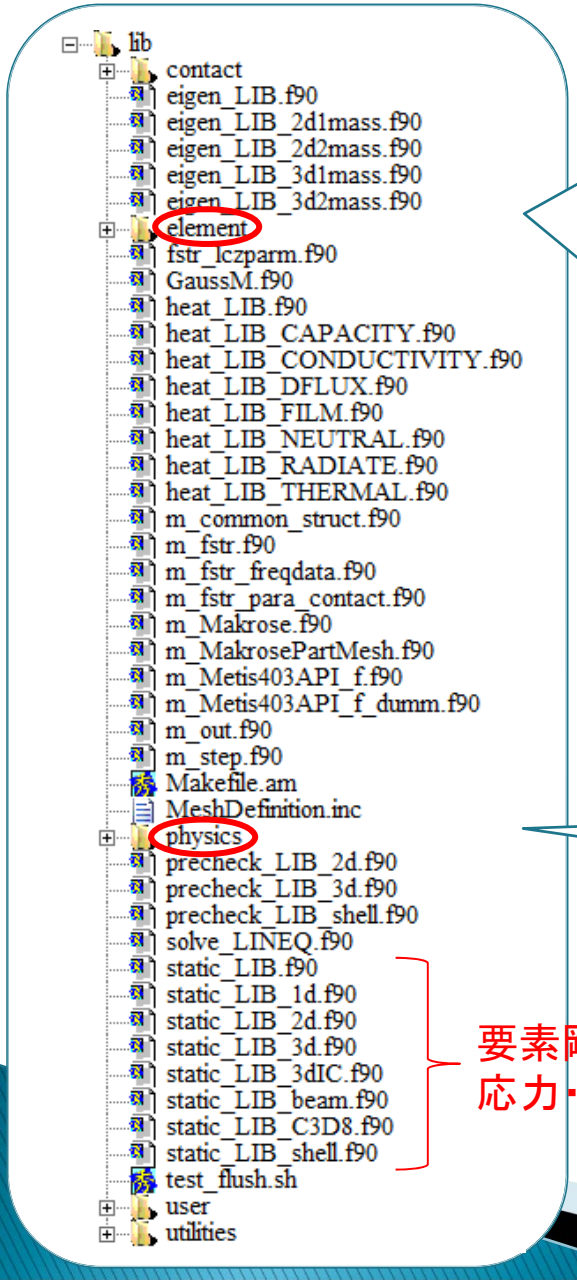
hecmw_etype.cの関数名	機能
HECMW_get_etype_UTIL2HECMW	入力されたHECMW_MESH_ETYPE_XXXに対して，対応するHECMW_ETYPE_XXXを返す
HECMW_get_etype_HECMW2UTIL	入力されたHECMW_ETYPE_XXXに対して，対応するHECMW_MESH_ETYPE_XXXを返す
HECMW_get_etype_GeoFEM2HECMW	入力されたHECMW_GEOFEM_ETYPE_XXXに対して，対応するHECMW_ETYPEXXXを返す
HECMW_get_max_node	入力されたHECMW_ETYPE_XXXに対して，対応するHECMW_MAX_NODE_XXXを返す
HECMW_get_max_edge	入力されたHECMW_ETYPE_XXXに対して，対応するHECMW_MAX_EDGE_XXXを返す
HECMW_get_max_surf	入力されたHECMW_ETYPE_XXXに対して，対応するHECMW_MAX_SURF_XXXを返す
HECMW_get_max_tsuf	入力されたHECMW_ETYPE_XXXに対して，対応するHECMW_MAX_TSUF_XXXを返す
HECMW_get_max_qsuf	入力されたHECMW_ETYPE_XXXに対して，対応するHECMW_MAX_QSUF_XXXを返す
HECMW_get_ucd_label	入力されたHECMW_ETYPE_XXXに対して，対応するHECMW_UCD_LABEL_XXXを返す
HECMW_is_etype_rod	入力された要素が線要素なら1を，そうでなければ0を返す
HECMW_is_etype_surface	入力された要素が平面要素なら1を返し，そうでなければ0を返す
HECMW_is_etype_solid	入力された要素がソリッド要素なら1を返し，そうでなければ0を返す
HECMW_is_etype_interface	入力された要素がインターフェイス要素なら1を返し，そうでなければ0を返す
HECMW_is_etype_beam	入力された要素が梁要素なら1を返し，そうでなければ0を返す
HECMW_is_etype_shell	入力された要素がシェル要素なら1を返し，そうでなければ0を返す
HECMW_is_etype_link	入力された要素が連結用の要素なら1を返し，そうでなければ0を返す
HECMW_is_etype_33struct	入力された要素が自由度混在用の要素なら1を返し，そうでなければ0を返す

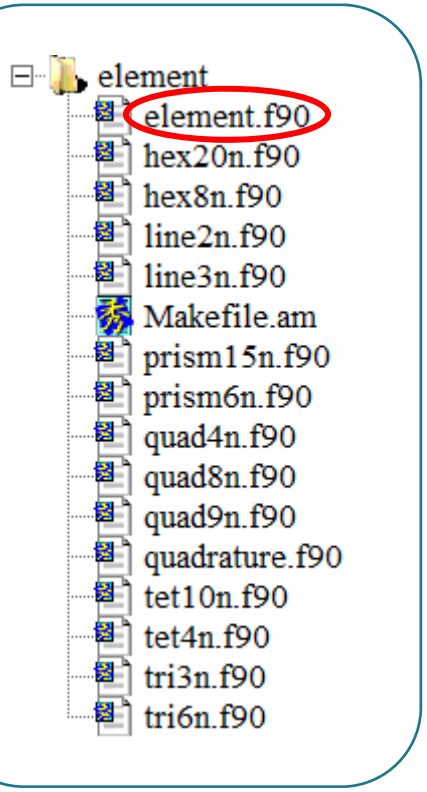
hecmw_reorder.c／hecmw_dist_refine.c の変更箇所

hecmw_reorderの関数名	機能
mask_node_dof	入力されたHEC-MWの要素タイプに対して，節点自由度数を識別する数字 (ビット) を返す

hecmw_dist_refine.cの関数名	機能
get_enode_h2r	入力されたHEC-MWの要素の節点番号に対して，REVOCAP_Refinerの要素の節点番号を返す
get_enode_r2h	REVOCAP_Refinerの要素の節点番号に対して，HEC-MWの要素の節点番号を返す
get_sid_h2r	HEC-MWの面番号に対して，REVOCAP_Refinerの面番号を返す
get_sid_r2h	REVOCAP_Reifinerの面番号に対して，HEC-MWの面番号を返す
elem_type_hecmw2rcap	HEC-MWの要素タイプ番号に対して，REVOCAP_Refinerの要素タイプ番号を返す
elem_type_rcap2hecmw	REVOCAP_Refinerの要素タイプ番号に対して，HEC-MWの要素タイプ番号を返す

ディレクトリfistr1/src/lib/element

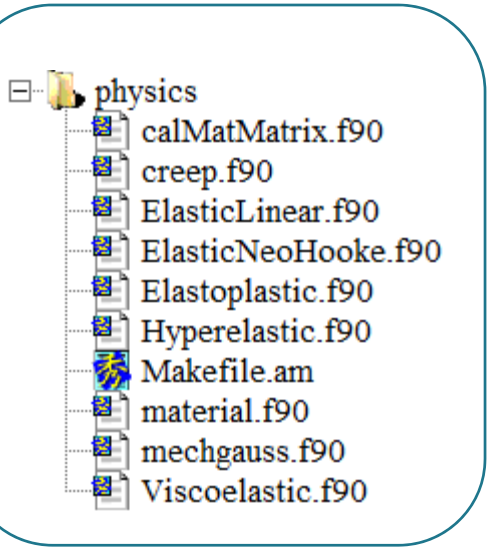




有限要素の幾何情報を計算する
プログラム
Bマトリックスの計算で使用

要素剛性マトリックスや
応力・内力の計算で使用

材料情報を計算する
プログラム
Dマトリックスの計算で使用



element.f90の修正箇所

USE shape_hex27n　・・・形状関数を計算するモジュールをUSEする

fe_hex27n ＝363　・・・HEC-MWで定義した要素タイプを追加

element.f90の関数名／サブルーチン名	機能
elementInfo::getSpaceDimension	入力されたHEC-MWの要素タイプ番号に対して，次元数を返す
elementInfo::getNumberOfNodes	入力されたHEC-MWの要素タイプ番号に対して，要素の節点数を返す
elementInfo::getNumberOfSubface	入力されたHEC-MWの要素タイプ番号に対して，要素の面数を返す
elementInfo::getSubFace	入力されたHEC-MWの要素タイプ番号と要素の面番号に対して，その面を構成する要素内節点番号を返す
elementInfo::NumOfQuadPoints	入力されたHEC-MWの要素タイプ番号に対して，Gauss積分点の数を返す
elementInfo::getQuadPoint	入力されたHEC-MWの要素タイプ番号およびGauss積分点の数に対して，Gauss積分点の計算空間座標値を返す
elementInfo::getWeight	入力されたHEC-MWの要素タイプ番号およびGauss積分点の数に対して，Gauss積分点の重みの値を返す
elementInfo::getShapeDeriv	入力されたHEC-MWの要素タイプ番号および計算空間座標値に対して，形状関数の微分値を返す
elementInfo::getShapeFunc	入力されたHEC-MWの要素タイプ番号および計算空間座標値に対して，形状関数の値を返す

hex27n.f90の作成

hex27n.f90のサブルーチン名	機能
shape_hex27n::ShapeFunc_hex27n	入力された要素タイプ番号fe_hex27nおよび計算空間座標値に対して，形状関数の値を返す $N^{(\alpha)} = \left\{\frac{\xi^{(\alpha)}\xi}{2}\left(1+\xi^{(\alpha)}\xi\right)+\left(1-\xi^{(\alpha)2}\right)\left(1-\xi^2\right)\right\}$ $\left\{\frac{\eta^{(\alpha)}\eta}{2}\left(1+\eta^{(\alpha)}\eta\right)+\left(1-\eta^{(\alpha)2}\right)\left(1-\eta^2\right)\right\}$ $\left\{\frac{\zeta^{(\alpha)}\zeta}{2}\left(1+\zeta^{(\alpha)}\zeta\right)+\left(1-\zeta^{(\alpha)2}\right)\left(1-\zeta^2\right)\right\}$ $\left\{\xi^{(\alpha)} \mid \alpha = 1,2,\ldots,27\right\} = \begin{Bmatrix} -1, & 1, & 1, -1, -1, & 1, & 1, -1, \\ 0, & 1, & 0, -1, & 0, & 1, & 0, -1, \\ -1, & 1, & 1, -1, & 0, & 0, & 0, & 1, \\ 0, -1, & 0 \end{Bmatrix}$ $\left\{\eta^{(\alpha)} \mid \alpha = 1,2,\ldots,27\right\} = \begin{Bmatrix} -1, -1, & 1, & 1, -1, -1, & 1, & 1, \\ -1, & 0, & 1, & 0, -1, & 0, & 1, & 0, \\ -1, -1, & 1, & 1, & 0, & 0, -1, & 0, \\ 1, & 0, & 0 \end{Bmatrix}$ $\left\{\zeta^{(\alpha)} \mid \alpha = 1,2,\ldots,27\right\} = \begin{Bmatrix} -1, -1, -1, -1, & 1, & 1, & 1, & 1, \\ -1, -1, -1, -1, & 1, & 1, & 1, & 1, \\ 0, & 0, & 0, & 0, -1, & 1, & 0, & 0, \\ 0, & 0, & 0 \end{Bmatrix}$
shape_hex27n::ShapeDeriv_hex27n	入力された要素タイプ番号fe_hex27nおよび計算空間座標値に対して，形状関数の微分値を返す $\frac{\partial N^{(\alpha)}}{\partial \xi}, \frac{\partial N^{(\alpha)}}{\partial \eta}, \frac{\partial N^{(\alpha)}}{\partial \zeta}$

要素に関するその他の修正

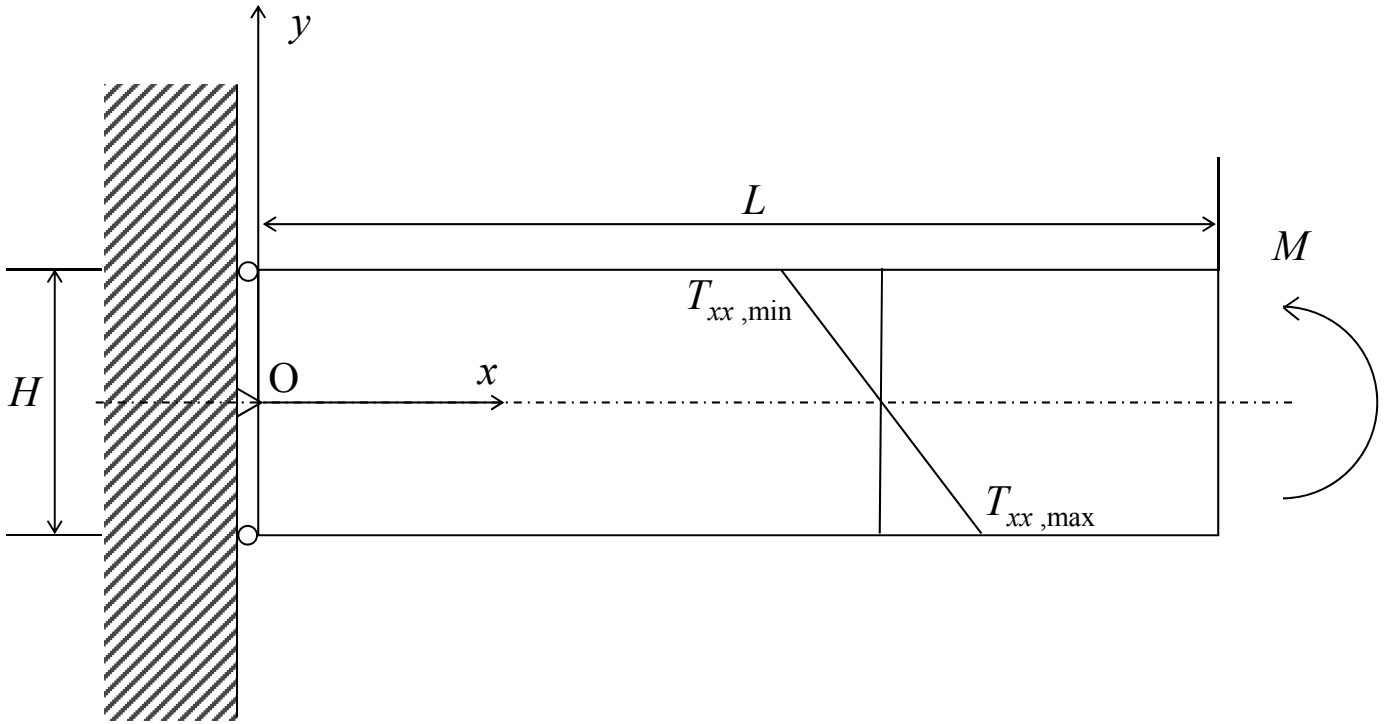
- analysis/static/fstr_ass_load.f90
  - サブルーチンm_fstr_ass_load::fstr_ass_load()
    要素タイプが363であるなら，static_LIB_3d.f90のサブルーチン
    m_static_LIB_3d::call DL_C3()を呼ぶ
    要素タイプが363であるなら，static_LIB_3d.f90のサブルーチン
    m_static_LIB_3d::TLOAD_C3()を呼ぶ

- analysis/static/fstr_StiffMatrix.f90
  - サブルーチンm_fstr_StiffMatrix::fstr_StiffMatrix()
    要素タイプが363であるなら，static_LIB_3d.f90のサブルーチン
    m_static_LIB_3d::STF_C3()を呼ぶ

- analysis/static/fstr_Update.f90
  - サブルーチンm_fstr_Update::fstr_UpdateNewton()
    要素タイプが363であるなら，static_LIB_3d.f90のサブルーチン
    m_static_LIB_3d::Update_C3()を呼ぶ
  - サブルーチンm_fstr_Update::fstr_Update3D()
    要素タイプが363であるなら，static_LIB_3d.f90のサブルーチン
    m_static_LIB_3d::Update_C3()を呼ぶ

- analysis/static/static_mat_ass_main.f90
  - サブルーチンm_static_mat_ass_main::fstr_mat_ass_main()
    要素タイプが363であるなら，static_LIB_3d.f90のサブルーチン
    m_static_LIB_3d::STF_C3()を呼ぶ

梁の曲げ解析（解析モデル）



Young's modulus: $E = 200{,}000$ MPa

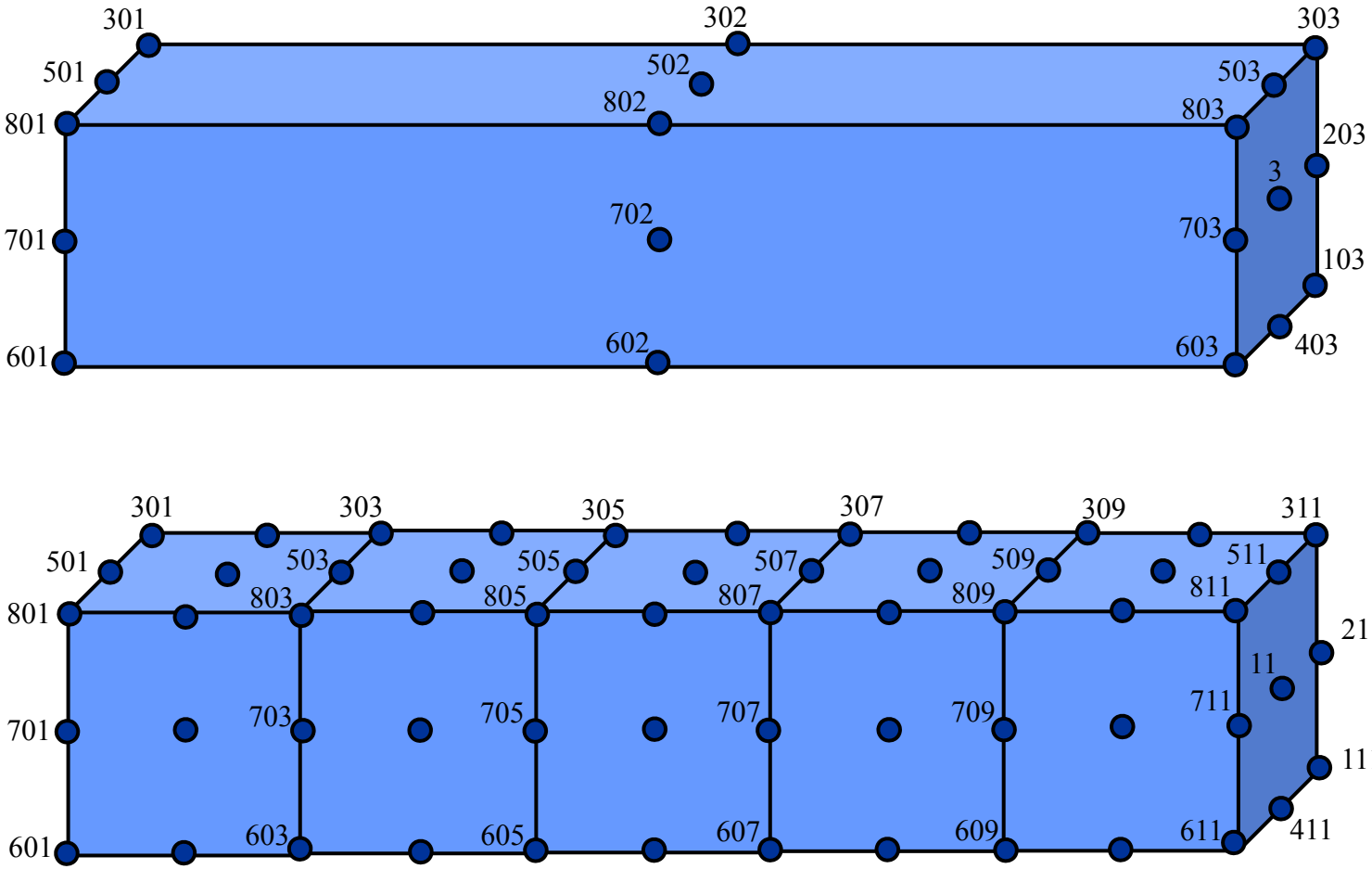
Poisson's ratio: $\nu = 0$

Geometry: $L = 100$ mm, $H = 10$ mm, $b = 10$ mm

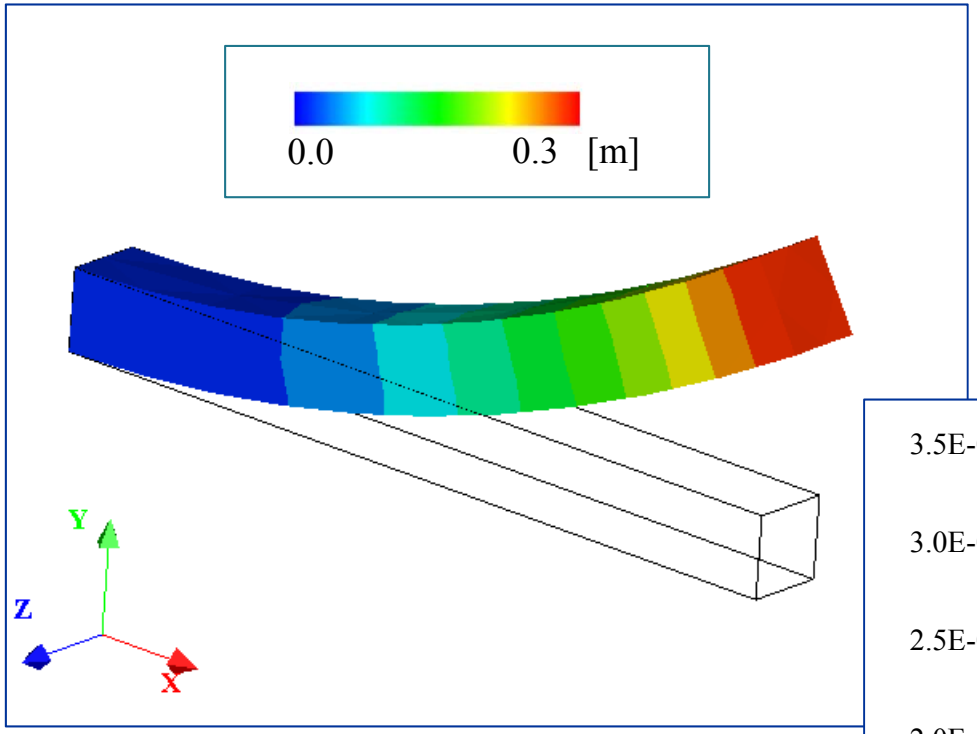
$T_{xx,\max} = 60$ MPa

$T_{xx,\min} = -60$ MPa

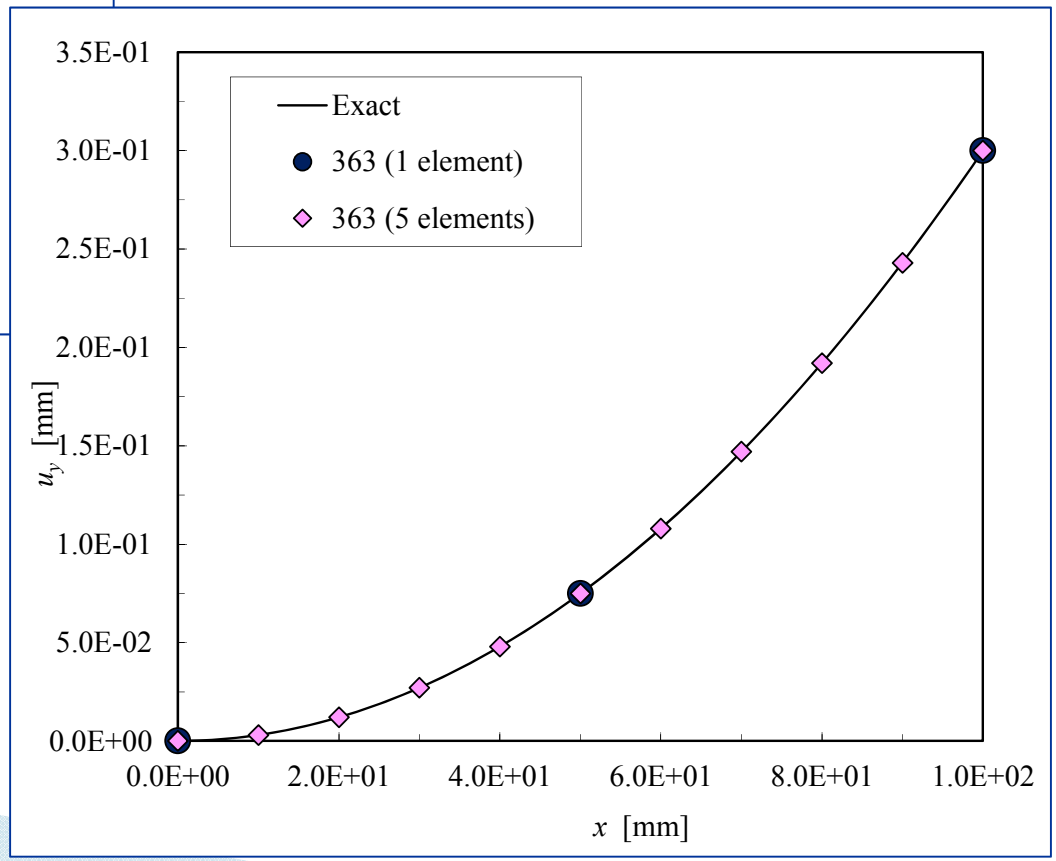
梁の曲げ解析（解析メッシュ）



梁の曲げ解析（計算結果）



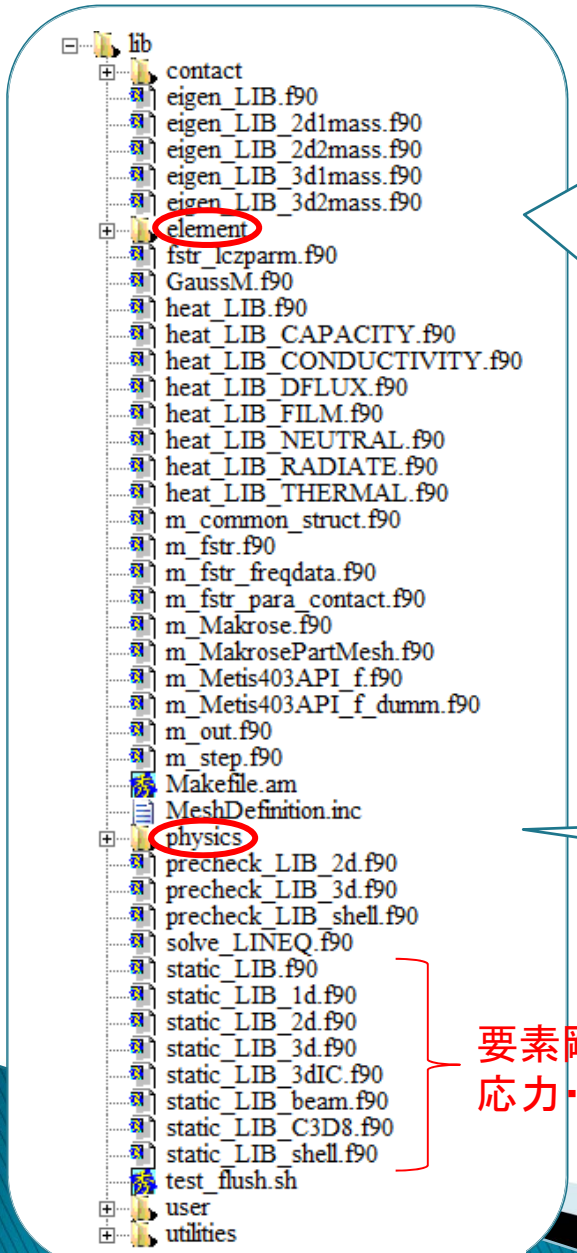
変形の倍率：100倍

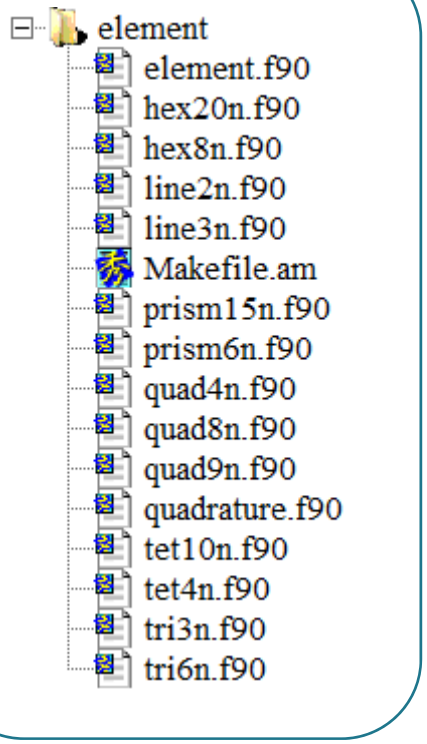


内容

1. FrontISTRのソースプログラム
2. FrontISTRへの新しい要素タイプの追加
   「27節点六面体要素の追加」
3. FrontISTRへの新しい材料の追加
   「3次式のMooney-Rivlin超弾性体 (8/1要素) 」
4. FrontISTRのユーザサブルーチンの使用方法の確認

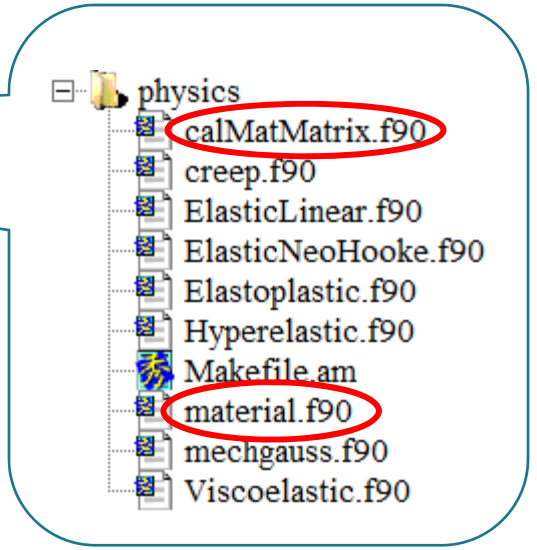
ディレクトリfistr1/src/lib/material





有限要素の幾何情報を計算する
プログラム
Bマトリックスの計算で使用

材料情報を計算する
プログラム
Dマトリックスの計算で使用



要素剛性マトリックスや
応力・内力の計算で使用

要素の追加に関する変更 (1/2)

要素タイプ3610を
追加します

- lib/element/element.f90
  - 定数fe_hex8n_up = 3610

- lib/physics/mechgauss.f90
  - 構造体tElement
    u/p定式化では，*p*は各要素に定義される
    メンバ変数としてポインタp(:)を追加する (プログラムlib/m_fstr.f90内の構造体fstr_solidのメンバ変数として，構造体tElementのインスタンスelements(:)が含まれている)

- analysis/static/fstr_StiffMatrix.f90
  - サブルーチンm_fstr_StiffMatrix::fstr_StiffMatrix()
    変数nn_pと変数lambda(1:nn_p)を定義する
    要素タイプが3610の場合，fstrSOLID%element(i)%p(j)をlambda(j)に渡し，static_LIB_3d_up8.f90のサブルーチンm_static_LIB_3d_up::STF_C3_up()を呼ぶ

- analysis/static/fstr_Update.f90
  - サブルーチンm_fstr_Update::fstr_UpdateNewton()
    変数nn_p，変数ddlambda(1:nn_p)，変数lambda(1:nn_p)を宣言する
    要素タイプが3610の場合，fstrSOLID%element(i)%p(j)をlambda(j)に渡し，static_LIB_C3D8.f90のサブルーチンm_static_LIB_3d::Update_C3_up()を呼ぶ
    ddlambda(1:nn_p)が算出されたら，lambda(j)を更新し，lambda(j)をfstrSOLID%element(i)%p(j)に渡す

要素の追加に関する変更 (2/2)

要素タイプ3610を
追加します

- lib/static_LIB_3d_up.f90
  - サブルーチンm_static_LIB_3d_up::STF_C3_up()
    サブルーチンm_static_LIB_3d_up::STF_C3_up()を追加する
    接線剛性マトリックスへのpの寄与も計算する
    calMatMatrix.f90のサブルーチンm_MatMatrix::MatlMatrix_up()を呼ぶ
  - サブルーチンm_static_LIB_3d_up::Update_C3_up()
    サブルーチンm_static_LIB_3d_up::Update_C3_up()を追加する
    内力へのpの寄与も計算し，ddlambdaも計算する
    calMatMatrix.f90のサブルーチンm_MatMatrix::StressUpdate_up()を呼ぶ

- lib/static_LIB.f90
  - モジュールm_static_LIB_3d_upのUSE

- lib/m_out.f90
  - サブルーチンm_out::initOutInfo()
    outinfo%keyWordにEPRESSURという名前を追加する

- analysis/static/static_make_result.f90
  - サブルーチンm_static_make_result::fstr_write_static_result()
    解析制御ファイルのヘッダー!ELEMENT_OUTPUTにElementalPRESSURE
    と書かれていたら，出力ファイルに圧力pの値を出力するようにする

材料の追加に関する修正

材料をMOONEYRIVLIN_CUBICを追加します

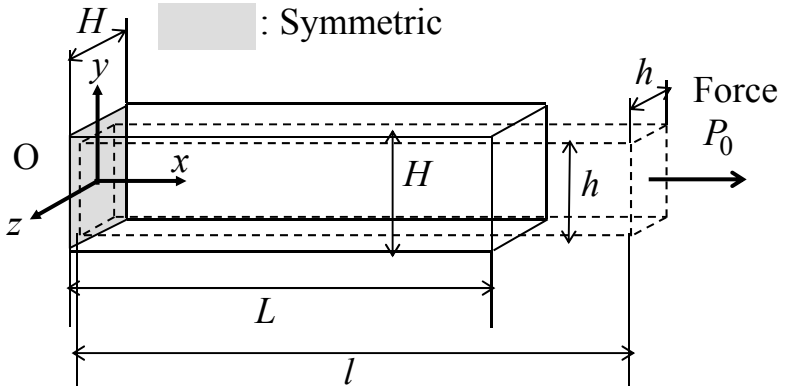
- lib/physics/material.f90
  3次式のMooney-Rivlin超弾性体の名前MOONEYRIVLIN_CUBICを追加する
  Mooney-Rivlin定数を2個から9個に増やす

- common/fstr_ctrl_material.f90
  - サブルーチンfstr_ctrl_get_HYPERELASTIC_INCOMP
    サブルーチンfstr_ctrl_get_HYPERELASTIC内にTYPE=MOONEYRIVLIN_CUBICの場合，data_fmt = "RRRRRRRRRR "として，9個のMooney-Rivlin定数と体積弾性率を入力できるようにする

- lib/physics/calMatMatrix.f90
  - サブルーチンMatlMatrix_up
    Hyperelastic.f90のサブルーチンcalElasticMooneyRivlin_cubicを呼ぶ
  - サブルーチンStressUpdate_up
    Hyperelastic.f90のサブルーチンcalUpdateElasticMooneyRivlin_cubicを呼ぶ

- lib/physics/Hyperelastic.f90
  - サブルーチンcderiv
    右Cauchy-Green変形テンソルのインバースも得られるようにする
  - サブルーチンcalElasticMooneyRivlin_Incomp
    Dマトリックスへのpの寄与も計算する
  - サブルーチンcalUpdateElasticMooneyRivlin_Incomp
    応力へのpの寄与も計算する

ブロックの引張変形（解析モデル）

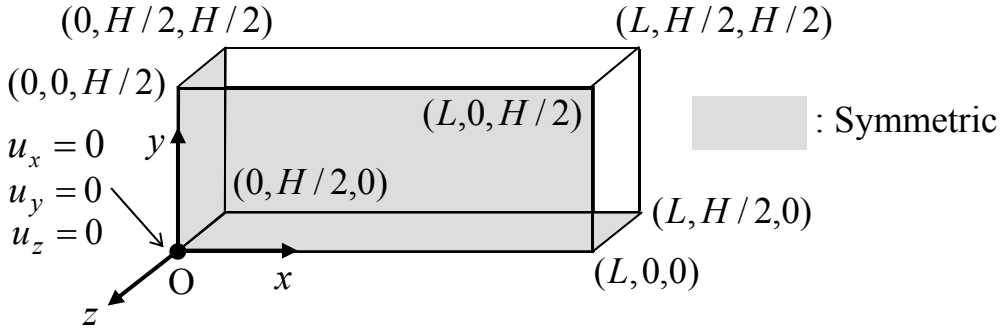


$L = 0.3$ m
$H = 0.1$ m
$P_0 = 2.0 \times 10^4$ N

$c_{10} = 0.3$ Pa
$c_{01} = 0.15$ Pa
$\nu = 0.4999$

(a) Analysis model

$$u_x = 0 \quad {}_0^t t_y = 0 \quad {}_0^t t_z = 0 \quad \text{on } x = 0$$
$${}_0^t t_x = 0 \quad u_y = 0 \quad {}_0^t t_z = 0 \quad \text{on } y = 0$$
$${}_0^t t_x = 0 \quad {}_0^t t_y = 0 \quad u_z = 0 \quad \text{on } z = 0$$
$${}_0^t t_x = P_0 / H^2 \quad {}_0^t t_y = 0 \quad {}_0^t t_z = 0 \quad \text{on } x = L$$
$${}_0^t t_x = 0 \quad {}_0^t t_y = 0 \quad {}_0^t t_z = 0 \quad \text{on the other boundaries}$$



(b) Boundary conditions

Fig. Analysis model and boundary conditions in tensile deformation problem

$$g({}_0^t III_C) = 2({}_0^t III_C{}^{1/2} - 1)$$

ブロックの引張変形（解析モデル）

Table Analysis meshes

	8/1 elements
Element number	$2 \times 2 \times 2$
Node number ($\boldsymbol{u}$)	27
Node number (p)	8

Table Relative errors (FrontISTR)

Exact solution	8/1 elements
$l = 8.4657356 \times 10^{-1}$ m	2.576×10^{-3} %
$h = 2.9764479 \times 10^{-2}$ m	5.913×10^{-4} %
$p = -1.8812746 \times 10^{6}$ Pa	1.188×10^{-3} %

Tolerance : 1.0×10^{-10}

Newton-Raphson iteration : 7

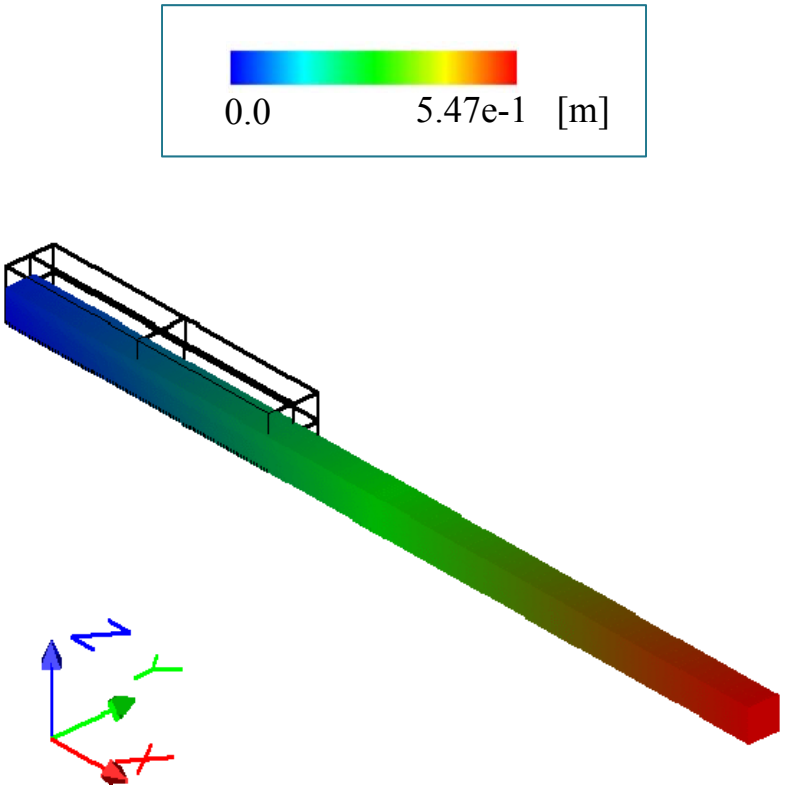


Fig. Deformation and initial mesh (FrontISTR)

内容

1. FrontISTRのソースプログラム
2. FrontISTRへの新しい要素タイプの追加
   「27節点六面体要素の追加」
3. FrontISTRへの新しい材料の追加
   「3次式のMooney-Rivlin超弾性体 (8/1要素) 」
4. FrontISTRのユーザサブルーチンの使用方法の確認

ユーザサブルーチンの使用 (1/4)

8. ユーザーサブルーチン

ユーザーが FrontISTR の機能をプログラミングにより拡張するためのインターフェースを提供する。これらのインターフェースは、基本的にサブルーチンヘッダを含む FORTRAN サブルーチンで、入出力変数の記述とこれらの変数のための宣言文である。ルーチンの主要部は、ユーザーによって書かなければならない。

FrontISTR は以下のユーザサブルーチンインターフェースを提供している。

8.1 ユーザー定義材料の入力

ユーザー定義材料を使用する場合、最大 100 のユーザー定義材料定数が使用可能である。材料定数の入力は以下のように、制御データファイル内の 1 行 10 数値、最大 10 行まで入力可能である。

2 行目～最大 10 行目
v1, v2, v3, v4, v5, v6, v7, v8, v9, v10
.....

8.2 弾塑性変形に関わるサブルーチン（uyield.f90）

「!PLASTIC, YIELD=USER」が正しい

弾塑性剛性マトリクスおよび応力の return mapping を計算するためのサブルーチンを提供している。ユーザー定義降伏関数を利用する場合、まず入力ファイルに!PLASTIC, TYPE=USER を設定して必要な材料定数を入力し、次にサブルーチン uElastoPlasticMatrix および uBackwardEuler を作成する必要がある。

（1）弾塑性剛性マトリクスの計算サブルーチン

```
subroutine uElastoPlasticMatrix( matl, stress, istat, fstat, D )
    REAL(KIND=kreal), INTENT(IN)  :: matl(:)
    REAL(KIND=kreal), INTENT(IN)  :: stress(6)
    INTEGER, INTENT(IN)           :: istat
    REAL(KIND=kreal), INTENT(IN)  :: fstat(:)
    REAL(KIND=kreal), INTENT(OUT) :: D(:,:)
```

matl: 材料定数を保存する配列（最大 100）
stress: 2nd Piola-Kirchhoff 応力
istat: 降伏状態(0: 未降伏； 1: 降伏した)
fstat: 状態変数. fstat(1)=塑性ひずみ、fstat(2:7)= back stress(移動または複合硬化時)
D: 弾塑性マトリクス

FrontISTRのユーザマニュアルの
163ページ～166ページ

FrontISTRで提供している
ユーザサブルーチン

(1) 弾塑性解析用
[uyield.f90]uElastoPlasticMatrix()
[uyield.f90]uBackwardEuler()

(2) 線形弾性・超弾性解析用
[uelastic.f90]uElasticMatrix()
[uelastic.f90]uElasticUpdate()

(3) [umat.f90]uMatlMatrix()
[umat.f90]uUpdate()

(4) [uload.f90]mULoad::ureadload
[uload.f90]mULoad::uloading
[uload.f90]mULoad::uResidual

ユーザサブルーチンの使用 (2/4)

（2）応力の Return mapping 計算サブルーチン

```
subroutine uBackwardEuler ( matl, stress, istat, fstat )
      REAL(KIND=kreal), INTENT(IN)    :: matl(:)
      REAL(KIND=kreal), INTENT(INOUT) :: stress(6)
      INTEGER, INTENT(INOUT)          :: istat
      REAL(KIND=kreal), INTENT(IN)    :: fstat(:)
```

matl: 材料定数を保存する配列（最大 100）
stress: trial stress 弾性変形を仮定し得られた 2nd Piola-Kirchhoff 応力
istat: 降伏状態(0: 未降伏； 1: 降伏した)
fstat: 状態変数． fstat(1)=塑性ひずみ、fstat(2:7)= back stress(移動または複合硬化時)

8.3 弾性変形に関わるサブルーチン（uelastic.f90）

弾性および超弾性問題の弾性剛性マトリクスおよび応力の更新計算をするためのサブルーチンを提供している。ユーザー弾性または超弾性構成式を利用する場合、まず入力ファイルに!ELASTIC, TYPE=USER または!HYPERELASTIC, TYPE=USER を設定して必要な材料定数を入力し、次にサブルーチン uElasticMatrix および uElasticUpdate を作成する必要がある。

（1）弾性剛性マトリクスの計算サブルーチン

```
subroutine uElasticMatrix( matl, strain, D )
      REAL(KIND=kreal), INTENT(IN)  :: matl(:)
      REAL(KIND=kreal), INTENT(IN)  :: strain(6)
      REAL(KIND=kreal), INTENT(OUT) :: D(6,6)
```

matl: 材料定数を保存する配列（最大 100）
strain: Green-Lagrange ひずみ
D: 弾性マトリクス

（2）応力の計算サブルーチン

```
subroutine uElasticUpdate ( matl, strain, stress )
      REAL(KIND=kreal), INTENT(IN)  :: matl(:)
      REAL(KIND=kreal), INTENT(IN)  :: strain(6)
      REAL(KIND=kreal), INTENT(OUT) :: stress(6)
```

matl: 材料定数を保存する配列（最大 100）
strain: Green-Lagrange ひずみ
stress: 応力

FrontISTRのユーザマニュアルの
163ページ～166ページ

ユーザサブルーチンの使用 (3/4)

8.4 ユーザー定義材料に関わるサブルーチン（umat.f)

!USER_MATERIAL

弾性、超弾性、弾塑性材に拘らず一般的な材料の変形解析のインターフェースを提供する。

FrontISTRのユーザマニュアルの
163ページ～166ページ

（1）剛性マトリクスの計算サブルーチン

```
subroutine uMatlMatrix( mname, matl, ftn, stress, fstat, D, temperature, dtime )
      CHARACTER(len=*), INTENT(IN)   :: mname
      REAL(KIND=kreal), INTENT(IN)   :: matl(:)
      REAL(KIND=kreal), INTENT(IN)   :: ftn(3,3)
      REAL(KIND=kreal), INTENT(IN)   :: stress(6)
      REAL(KIND=kreal), INTENT(IN)   :: fstat(:)
      REAL(KIND=kreal), INTENT(OUT)  :: D(:,:)
      REAL(KIND=kreal), optional     :: temperature
      REAL(KIND=kreal), optional     :: dtime
```

「uMatlMatrix(mname, matl, strain, stress, fstat, D, dtime, ttime, temperature)」
が正しい

mname: 材料名
matl: 材料定数を保存する配列（最大 100）
ftn: 変形勾配テンソル
stress: 2nd Piola-Kirchhoff 応力
fstat: 状態変数
D: 構成式
temperature: 温度
dtime: 時間増分

（2）ひずみおよび応力の更新計算サブルーチン

```
subroutine uUpdate( mname, matl, ftn, strain, stress, fstat, temperature, dtime )
      character(len=*), intent(in)     :: mname
      real(KIND=kreal), intent(in)     :: matl
      real(kind=kreal), intent(in)     :: ftn(3,3)
      real(kind=kreal), intent(inout)  :: strain(6)
      real(kind=kreal), intent(inout)  :: stress(6)
      real(kind=kreal), intent(inout)  :: fstat(:)
      real(KIND=kreal), optional       :: temperature
      real(KIND=kreal), optional       :: dtime
```

「uUpdate(mname, matl, strain, stress, fstat, dtime, ttime, temperature)」
が正しい

mname: 材料名
matl: 材料定数を保存する配列（最大 100）
ftn: 変形勾配テンソル
strain: ひずみ
stress: 2nd Piola-Kirchhoff 応力

ユーザサブルーチンの使用 (4/4)

fstat: 状態変数
temperature: 温度
dtime: 時間増分

8.5 ユーザー定義外部荷重の処理サブルーチン（uload.f）

ユーザー定義外部荷重を処理するインターフェースを提供する。
ユーザー定義外部荷重を利用するため、まず外部荷重を定義するための数値構造 tULoad を定義し、入力ファイルの!ULOAD を利用してその定義を読み込む。その後、以下のインターフェースを利用して、外部荷重を組み込む。

（1）外部荷重の読み込みサブルーチン

```
integer function ureadload( fname )
    character(len=*), intent(in)      :: fname
```

fname: 外部ファイル名。このファイルからユーザー定義外部荷重を読み込む。

（2）外部荷重を全体荷重ベクトルへ組み込むサブルーチン

```
subroutine uloading( cstep, factor, exForce )
    integer, INTENT(IN)                 :: cstep
    REAL(KIND=kreal), INTENT(IN)        :: factor
    REAL(KIND=kreal), INTENT(INOUT) :: exForce(:)
```

cstep: 現時点の解析ステップ数
factor: 現ステップの荷重係数
exForce: 全体荷重ベクトル

（3）残差応力の計算サブルーチン

```
subroutine uResidual( cstep, factor, residual )
    integer, INTENT(IN)                 :: cstep
    REAL(KIND=kreal), INTENT(IN)        :: factor
    REAL(KIND=kreal), INTENT(INOUT) :: residual(:)
```

cstep: 現時点の解析ステップ数
factor: 現ステップの荷重係数
residual: 全体残差力ベクトル

FrontISTRのユーザマニュアルの
163ページ～166ページ

!ULOAD, FILE=‘ファイル名’