V4.6 の公開と V5.0 に向けた開発

FrontISTR研究会

V4.5とV4.6の差分情報

1.1 解析機能・アルゴリズム①

- 回転変位の境界条件追加
  - 次に示すフォーマットで，ある節点†に対する X, Y, Z 軸周りの回転変位を実現可能
    - !BOUNDARY, ROT_CENTER=*回転中心となる節点のID*, GRPID=*グループID*
    - *回転変位を与える節点グループ名*, 1, 1, *X 軸周りの回転角度*
    - *回転変位を与える節点グループ名*, 2, 2, *Y 軸周りの回転角度*
    - *回転変位を与える節点グループ名*, 3, 3, *Z 軸周りの回転角度*
- トルクの境界条件追加
  - 次に示すフォーマットで，ある節点†に対する X, Y, Z 軸周りのトルクを実現可能
    - !CLOAD, ROT_CENTER=*回転中心となる節点のID*, GRPID=*グループID*
    - *トルクを与える節点グループ名*, 1, 1, *回転ベクトルの X 軸方向成分*
    - *トルクを与える節点グループ名*, 2, 2, *回転ベクトルの Y 軸方向成分*
    - *トルクを与える節点グループ名*, 3, 3, *回転ベクトルの Z 軸方向成分*

† 節点グループを指定した場合，グループ内の節点の座標値を平均した点が中心となる.

1.1 解析機能・アルゴリズム②

- 3次元線形静的解析の場合における反力計算の計算方法変更
  - 修正前は，変位境界条件が指定された節点・自由度上に荷重境界条件を指定すると，荷重境界条件の影響を無視した反力が出力されていた
  - Abaqus の定義に倣い，変位境界条件が指定された節点・自由度上に荷重境界条件を指定すると，従来の反力に荷重境界条件の値を加えた値が出力されるように変更した
- 接触解析時の scan_contact_state 関数がメモリリークを起こすバグの修正

1.1 解析機能・アルゴリズム③

- 非圧縮性流体解析機能の導入
  - メッシュファイル (.msh) に3414要素でメッシュ情報を記述
    - 四面体1次要素(341)と節点数，コネクティビティは同じ
    - 1節点につき4つの自由度 (速度 + 圧力) を持つ
  - 非線形陰的動解析のみ対応
    - 制御ファイルに !SOLUTION, TYPE=DYNAMIC と !DYNAMIC, TYPE=NONLINEARを指定
  - 非圧縮性流体 の材料物性を定義
    - !MATERIAL, NAME=FLUID
    - !FLUID, TYPE=INCOMP_NEWTONIAN
    - 1.0e-3　　粘度
    - !DENSITY
    - 1.0e3　　質量密度

1.1 解析機能・アルゴリズム③

- 速度境界条件は次のように指定
  - !BOUNDARY, GRPID=1
  - *節点グループ名*, 1, 1, 0.0　　節点グループの X 方向速度を 0.0 に固定
  - !BOUNDARY, GRPID=2
  - *節点グループ名*, 4, 4, 0.0　　節点グループの圧力を 0.0 に固定
- 流速，圧力は次のように可視化
  - !OUTPUT_VIS
  - DISP, ON　　流速の可視化
  - PRESSURE, ON　　圧力の可視化
- 線形ソルバーはBiCGSTAB (METHOD=BICGSTAB)，前処理はブロック SSOR (PRECOND=1) とブロック対角スケーリング (PRECOND=3) のみ対応

1.2 要素

- 非線形静解析から固有値解析を行う (!SOLUTION, TYPE=STATICEIGEN) 際，六面体1次要素を使うときのバグを修正
  - 修正前は!ELEM_OPTカードの指定に依らず，完全適合要素が使用されていた
  - 修正後は!ELEM_OPTカードの指定に依らず，B-bar要素が使用されるように修正

1.3 線形ソルバ

- 4 × 4ブロック線形ソルバの追加
  - 対角スケーリング前処理を追加
- 線形ソルバにMUMPSを使用した場合のログ出力方法を細分化
  - !SOLVER, METHOD=MUMPS, TIMELOG=NO/YES/VERBOSE
    - NO：ログを出力しない
    - YES：エラーメッセージのみ出力
    - VERBOSE：エラーメッセージ，警告，MUMPSの計算時間といった統計情報を出力

1.4 入力

- メッシュファイル (.msh ファイル) 内の材料定義 (!MATERIAL カード) の読み込みを高速化
  - FrontISTRでは，メッシュファイル (.msh) 内で解析領域に対する材料の名を !MATERIAL カードで定義し，制御ファイル (.cnt) 内で同じく !MATERIAL カードを使って材料物性値を具体的に与える必要がある
  - 従来は，制御ファイルから対応するメッシュファイル内の !MATERIAL カードを探す際，材料名が格納された配列を最初のインデックスから順に探索して対応する材料を探していた
  - メッシュファイルを読み込んだ段階でハッシュテーブルを作り，制御ファイルから読み込む際はハッシュテーブルを参照するようにコードを変更
- Abaqus 形式フォーマット (foo.inp) の入力ファイルへの準拠
  - 対応するカードの追加

1.5 プログラムの実装

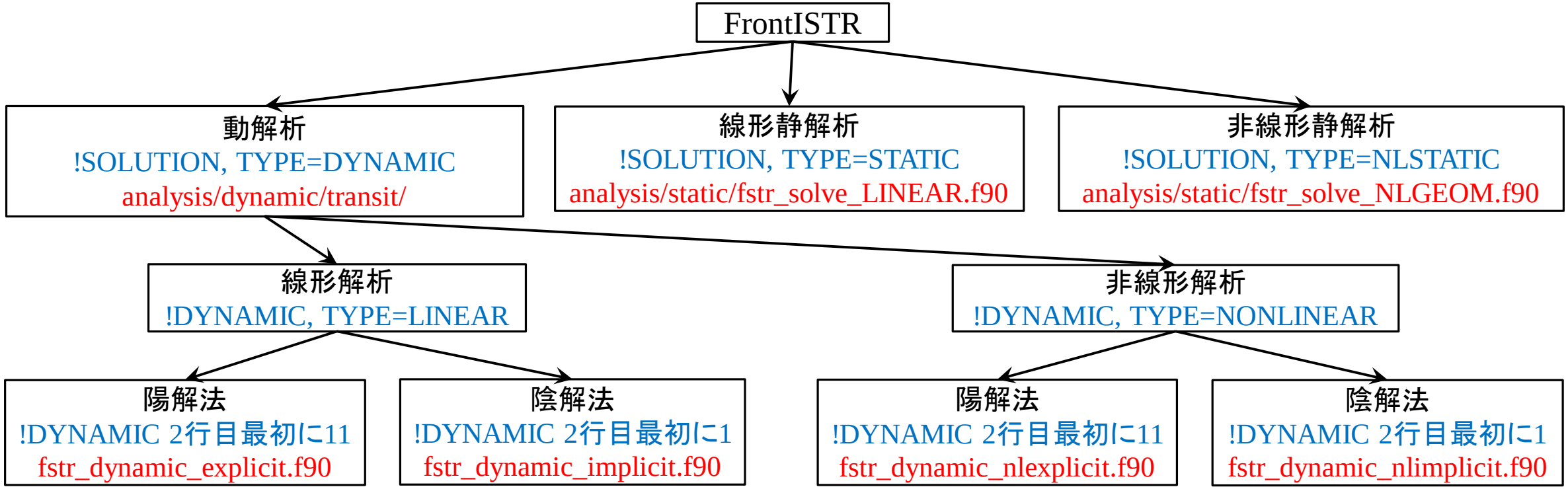
- gccでコンパイルした際，warning が出力される部分を書き直し，warning が出力されないように修正
- コードファイルの先頭部分に記述されていたコメントを修正
  - ライセンスに関する記述は LICENSE.txt に記述し，コメントを簡潔に変更した
- 大規模なモデルをリファイン処理した場合の不具合を修正
  - 節点数に依存する変数が 32 bit 整数で定義されており，節点数が増大した場合オーバーフローが発生していた.
  - 問題となっていた変数の型を 64 bit 整数に変更

V5.0 について①

2.1 解析機能・アルゴリズム ①: 上位レベルでのルーチンの統合化

- 静解析における線形・非線形解析のルーチン統合
- 動解析における線形・非線形解析のルーチン統合
  - 線形解析と非線形解析でコードが別々に実装されている
    - 次スライドのFig. 1参照
  - 静解析の場合と動解析の場合で，非線形解析の実施方法が異なる
    - 静解析の場合，次のように !SOLUTION カードで区別：
      - !SOLUTION, TYPE=STATIC
      - !SOLUTION, TYPE=NLSTATIC
    - 動解析の場合，!SOLUTION, TYPE=DYNAMIC としたうえで，次のように区別：
      - !DYNAMIC, TYPE=LINEAR
      - !DYNAMIC, TYPE=NONLINEAR
  - Fig. 2に示すようにこれらを統合

2.1 解析機能・アルゴリズム ①: 上位レベルでのルーチンの統合化



青: 制御ファイル内の記述　赤: 実装されているコードファイル

Fig. 1 現在 (V4.6) の FrontISTR における線形/非線形・静/動解析の実装形式

2.1 解析機能・アルゴリズム ①: 上位レベルでのルーチンの統合化

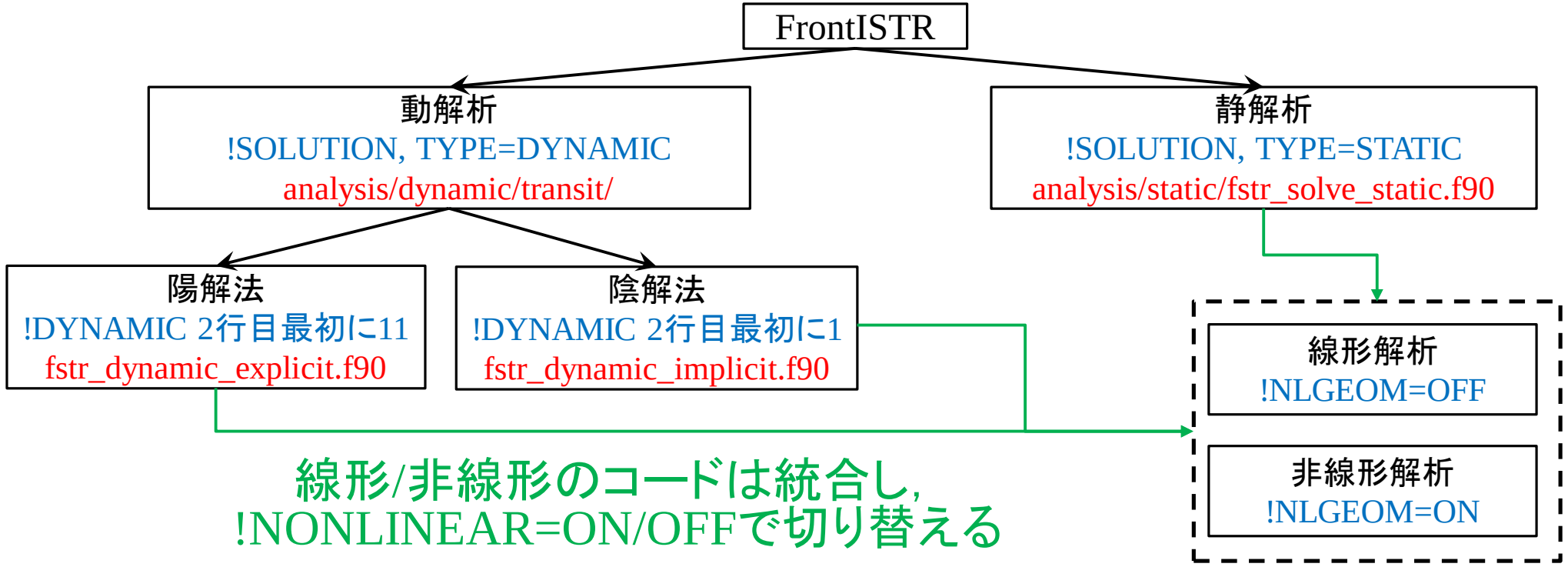


Fig. 2 V5.0 における線形/非線形 静/動 解析の実装形式

2.1 解析機能・アルゴリズム ②: 非線形静解析の荷重増分制御

- 非線形静解析の自動増分機能
  - 非線形方程式を解くための Newton-Raphson (NR) 法のループで，収束までに要する反復回数が増大した場合，荷重増分を自動的に小さくとる
  - 反復回数が減少した場合，荷重増分を自動的に大きくとる
- 非線形静解析のカットバック機能
  - 非線形方程式を解くための NR 法のループで，収束せずに解が発散した場合，直前のNR法の反復ステップに戻り，荷重増分を自動的に小さくして，再度求解する

2.2 要素:

- 要素定式化の切り替え
  - 解析制御ファイル内の!ELEMOPT カードで，要素の定式化を変更可能にする
    - 例:六面体1次要素の場合
      - !ELEMOPT, 361=IC　　完全適合積分要素
      - !ELEMOPT, 361=Bbar　B-bar要素

2.3 並列化

- 節点番号リオーダリング
  - 分散メッシュを立方体領域で分割し，その領域内でメッシュの節点番号を振り直す
  - 行列演算における節点番号アクセスリストのデータ参照の局所性が向上することにより，キャッシュ効果が向上する
- パーティショナの並列化

2.4 入力

- 並列計算時の全体制御ファイルの書式変更
  - hecmw_ctrl.dat では，逐次，並列ごとに次のように記述を変える必要がある
    - 逐次 : !MESH, NAME=fstrMSH, TYPE=HECMW-ENTIRE
    - 並列 : !MESH, NAME=fstrMSH, TYPE=HECMW-DIST
  - さらに，並列計算時は hecmw_part_ctrl.dat を準備し，並列数などの情報を記述する必要がある.
  - hecmw_part_ctrl.dat の内容は hecmw_ctrl.dat に統合し，hecmw_ctrl.dat 内では全体メッシュ，分散メッシュの区別なく記述できるように変更する.
- msh ファイルに記述すべき情報と cnt ファイルに記述すべき情報の整理
  - 例 : 熱 - 構造片方向連成の場合の初期温度は，msh ファイル内で定義されているが，幾何学的な情報ではないので本来は cnt ファイルで記述するべき

2.5 出力

- リスタートファイルの書式変更
  - 規定の書式が存在しない
    - 各ルーチン毎に吐き出す情報やその並びが異なる
  - リスタートファイルの書式ルールを作り，それに従って各ルーチンのリスタートファイル出力を修正する
- 各解析における出力ルーチンの統合
  - 例えば，static_output.f90 (静解析) と dynamic_output.f90 (動解析) は，実質的な内容がほとんど同じである
  - 出力のルーチンを統合してfstr_output.f90 (仮) を作る

2.6 可視化

- visualizer の ini ファイルがランク0からしか読めないバグの修正
  - 例えば京コンピュータでは各 CPU がお互いのファイルを読み込むことが出来ない
    - デッドロックが発生する
  - また，現在の実装では，可視化のためにFrontISTR内で新たに配列の確保を行っている
  - vtk や inp などのデータを FrontISTRから直接書き出すように変更する

2.7 プログラムの実装

- オブジェクト指向に則って作られたコードファイル (hecmw2, fistr2) の削除
  - 現在整備されていない

2.8 マニュアル

- マニュアルの修正
  - マニュアルに未記載の機能や，マニュアルに記載されている式通りに実装されていない場合があるため，マニュアルを修正する
- Markdown 言語形式のマニュアル作成
  - 現在は Microsoft Word を使って記述されている
    - ページ数・数式が多いため，修正作業が煩雑
    - .docx ファイルはバイナリファイルであるため，git によるバージョン管理が困難
  - Markdown (.md ファイル) を使ってマニュアルを作成し直す
    - 容易に wiki のようなマニュアルを作成可能
    - LaTeX 形式に準じた数式入力が可能
    - .md ファイルはテキストファイルであり，git によるバージョン管理が容易

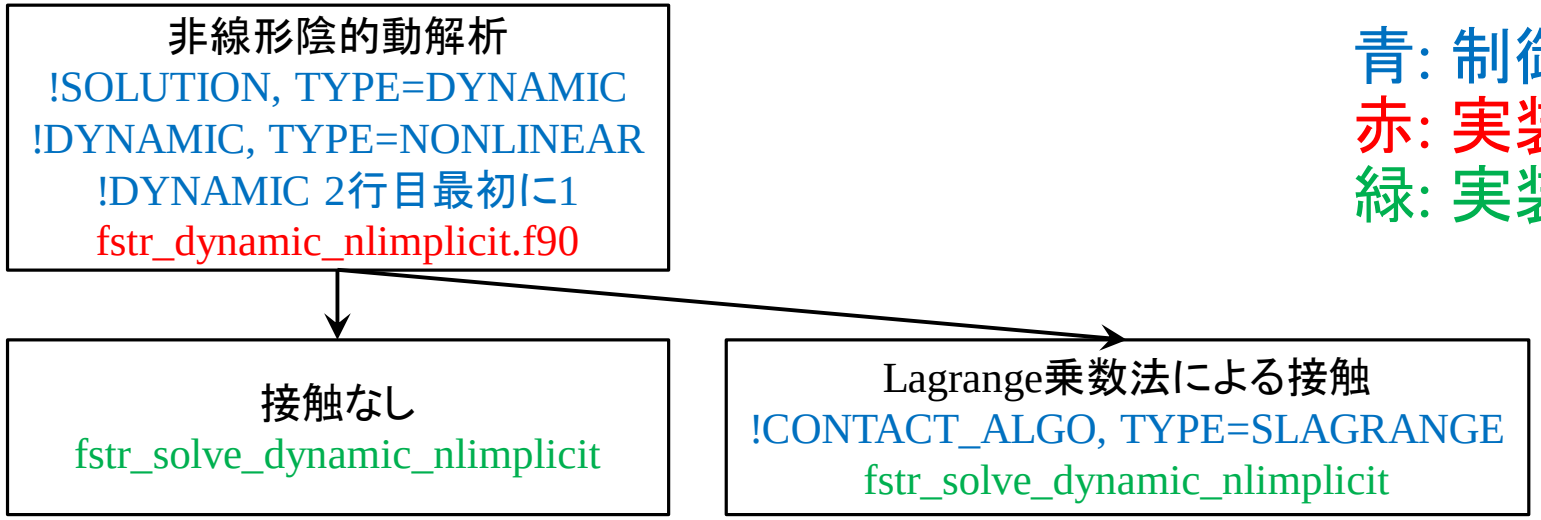
V5.0 について②

追加で議論する内容

3.1 解析・アルゴリズム ①: 接触ルーチンの統合

- 接触解析ルーチンの統合
  - Fig. 3に示すように，現在は接触の有無，接触アルゴリズムの種類毎に，別のサブルーチンが実装されている.
  - これらを統合して一つのサブルーチンにする

3.1 解析・アルゴリズム ①: 接触ルーチンの統合



青: 制御ファイル内の記述
赤: 実装されているコードファイル
緑: 実装されているサブルーチン

接触解析の有無，アルゴリズム毎に別のサブルーチンとして実装されている
(非線形陽的動解析，非線形静解析も同様)

⇓

一つのサブルーチンにまとめて記述する

Fig. 3 現在 (V4.6) の FrontISTR における接触解析の実装形式

3.1 解析・アルゴリズム ②: 片方向連成解析

- リスタートファイルを利用した熱-構造連成解析の追加
  - 従来は，熱伝導解析を実施した後，その結果を読み込んで熱から構造への影響のみを考慮した片方向連成解析が可能
  - リスタートファイルを利用した形式に書き直し

3.2 ソルバ

- !SOLVER,TYPE=DIRECTの削除
  - FrontISTRに直接実装された直接法ソルバ
  - 現在のバージョンではそもそも動作しない

3.3 プログラムの実装

- ファイル名・ディレクトリ構造の変更
  - 例：fstr_StiffMatrix.f90は接線剛性マトリクスを計算するためのコードで，動解析でも静解析でも使うが，これは静解析のディレクトリ (analysis/static) にある
  - ファイル名とディレクトリ構造を適切なものに変更する
    - 上の例であれば，analysis/ に配置し直す
- 熱伝導・固有値応答，周波数応答のリファクタリング
  - FrontISTRで定義されている関数を全く使っておらず，モジュール化がされていない．
    - globalDerivやgetShapeFuncなど
  - これらを使って書き直す

3.4 テスト

- テストプログラムの網羅性
  - 全てのパターンを網羅しているかを確認
- テストプログラムのVerification
  - 妥当な解が得られているかを確認