機能仕様：自動増分調整・カットバック機能

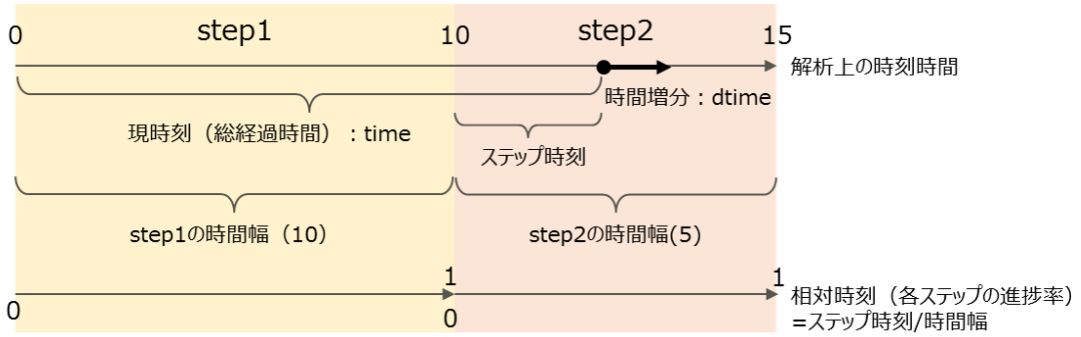
概要

FrontISTR の非線形静解析において、収束状況に応じた時間増分の自動調整と収束に失敗した場合のカットバックを行う。 また、別途指定した時刻においてつり合い計算実行および結果出力を行う。

解析上の時間について

ここでは FrontISTR の解析上の時間について、以下の通り用語の定義を行う：

- 現時刻：解析初期からの総経過時間
- ステップ時刻：ステップ開始からの経過時間
- 時間幅：ステップで解析する時間
- 相対時刻：時間幅に対するステップ開始からの経過時間の割合
- 時間増分：現時刻からつり合いを求める時刻までの増分



time

自動増分調整・カットバック

非線形静解析の流れ

自動増分調整・カットバック機能が有効な場合の非線形静解析の流れは以下の通りである。

1. ステップ 1 から最終ステップ N_step まで、下記 2.以降の手続きを繰り返す：
2. 時間増分の基準値 dtime_base を、現在の dtime_base と前サブステップでの収束状況から定める。初回は初期時間増分 initdt を用いる。
3. 実際の時間増分 dtime を、ステップ終了または直近の出力指定時刻までの残り時間と dtime_base の小さい方で定める。
4. 時刻 time+dtime でのつり合い計算を試みる
5. 収束に成功した場合は時刻 time に時間増分 dtime を加算して時刻を進める。失敗した場合は時刻 time の状態を復元して 2.に戻る。
6. time がステップ終了時刻に到達したらステップを終了する

解析途中で下記に挙げるケースに該当した場合、非線形静解析の手続きは失敗と判断され、エラー終了する。

1. time がステップ終了時刻に到達する前にサブステップ数が上限に到達した場合
2. 時間増分の基準値 dtime_base が時間増分下限 mindt を下回った場合
3. 指定された N_C 回連続して収束に失敗した場合

時間増分基準値 dtime_base の調整

dtime_base は以下に述べる通り設定される。

ステップ初回は、指定された初期時間増分 initdt の値に設定される。 それ以外の場合は、前のサブステップの収束状況に応じて次の通り設定される。

1. 収束に失敗した場合（カットバックされた場合）...dtime_base にカットバック縮小率 R_C を乗じた値
2. 収束に成功した場合
    1. 減少条件に該当する場合：dtime_base に減少率 R_S を乗じた値
    2. 減少条件に該当せず、増加条件に該当する場合：dtime_base に増加率 R_L を乗じた値と、時間増分上限 maxdt の小さい方
    3. 減少条件にも増加条件にも該当しない場合：dtime_base は変化しない

増加・減少条件

自動増分調整機能では、増分を増加・減少させる条件を以下の変数を用いて判定する：

- N_max：前サブステップにおける Newton 法反復回数の最大値
- N_sum：前サブステップにおける Newton 法反復回数の合計値（接触反復が無い場合は N_max に一致）
- N_cont：前サブステップにおける接触反復回数

減少条件は以下の両方が満たされるときである：

- N_max, N_sum, N_contのいずれか一つが、各々の閾値$N^S_max, N^S_sum, N^S_cont$を上回る
- 上記の状態が、N_S 回以上連続したサブステップで満たされる

増加条件は以下の両方が満たされるときである：

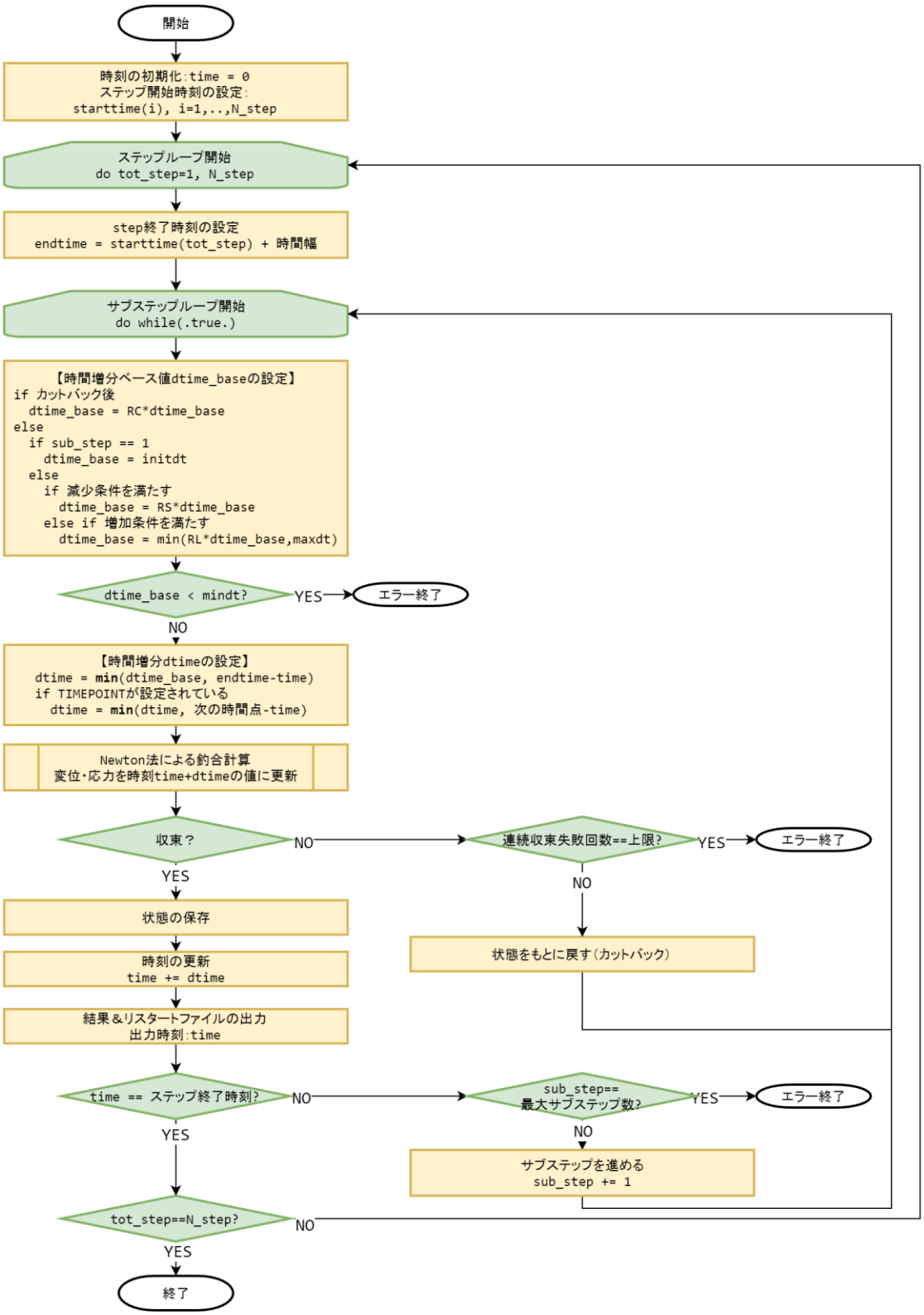
- N_max, N_sum, N_contのすべてが、各々の閾値$N^L_max, N^L_sum, N^L_cont$以内である
- 上記の状態が、N_L 回以上連続したサブステップで満たされる

計算および出力時刻の指定

自動時間増分では収束状況によって増分が変化するため、ステップの途中ではどの時刻につりあい計算および結果出力が行われるか予測できない。 これが不便である場合に、出力時刻のリストを与えることによって、希望する時刻におけるつり合い計算および結果出力を実行させることができる。 出力時刻のリストが与えられたステップでは、指定された時刻にて必ず計算が行われるように、時間増分 dtime の値が調整される。

自動時間増分・カットバックのフローチャート

上記の流れをまとめると次の図のようになる：


開始
時刻の初期化:time = 0
ステップ開始時刻の設定:
starttime(i), i=1,..,N_step
ステップループ開始
do tot_step=1, N_step
step終了時刻の設定
endtime = starttime(tot_step) + 時間幅
サブステップループ開始
do while(.true.)
【時間増分ベース値dtime_baseの設定】
if カットバック後
dtime_base = RC*dtime_base
else
if sub_step == 1
dtime_base = initdt
else
if 減少条件を満たす
dtime_base = RS*dtime_base
else if 増加条件を満たす
dtime_base = min(RL*dtime_base,maxdt)
dtime_base < mindt?
YES
エラー終了
NO
【時間増分dtimeの設定】
dtime = min(dtime_base, endtime-time)
if TIMEPOINTが設定されている
dtime = min(dtime, 次の時間点-time)
Newton法による釣合計算
変位・応力を時刻time+dtimeの値に更新
収束？
NO
連続収束失敗回数==上限?
YES
エラー終了
NO
YES
状態の保存
時刻の更新
time += dtime
結果＆リスタートファイルの出力
出力時刻:time
状態をもとに戻す(カットバック)
time == ステップ終了時刻?
NO
sub_step==
最大サブステップ数?
YES
エラー終了
NO
YES
サブステップを進める
sub_step += 1
tot_step==N_step?
NO
YES
終了

autoinc_flow

時間増分の使用方法

自動増分調整・カットバック機能は!STEP カードで TYPE_INC=AUTO を指定することで有効になる。時間増分の調整関連のパラメータは!AUTOINC_PARAM カードで指定する。 出力時刻の指定は、まず!TIME_POINTS カードで時刻リストの定義を行った上で、 !STEP, TIMEPOINTS パラメータでそれを指定することにより、当該ステップにおいて有効になる。

入力カード仕様

本機能に関する設定はすべて解析制御ファイルで行う。

解析ステップの設定・!STEP（既存カードの拡張）

パラメータ

※印が今回追加されるパラメータ

パラメータ名	属性	内容
TYPE	C	STATIC（default 値）／VISCO（準静的解析）
SUBSTEPS	I	境界条件の分割ステップ数（デフォルト：1）
CONVERG	R	収束判定閾値（デフォルト：1.0e-6）
MAXITER	I	非線形解析における最大反復計算回数（デフォルト：50）
AMP	C	時間関数名（!AMPLITUDE で指定）
※INC_TYPE	C	FIXED（固定増分・デフォルト）／AUTO（自動増分）
※MAXRES	R	最大許容残差の設定（デフォルト：1.0e+10）
※TIMEPOINTS	C	時刻リスト名（!TIME_POINTS, NAME で指定）
※AUTOINCPARAM	C	自動接触パラメータセット名（!AUTOINC_PARAM, NAME で指定）
※MAXCONTITER	I	接触解析における最大接触反復回数（デフォルト：10）

2 行目以降

INC_TYPE=FIXED の場合（TYPE=STATIC の場合は省略可）

（2 行目） `DTIME, ETIME`

変数名	属性	内容
DTIME	R	時間増分（デフォルト：1/SUBSTEPS）
ETIME	R	ステップ時間幅（デフォルト：1）

INC_TYPE=AUTO の場合（TYPE によらず指定）

（2 行目） `DTIME_INIT, ETIME, MINDT, MAXDT`

変数名	属性	内容
DTIME_INIT	R	初期時間増分
ETIME	R	ステップ時間幅
MINDT	R	時間増分下限
MAXDT	R	時間増分上限

3 行目以降

```
BOUNDARY, id        id=!BOUNDARY で定義した GRPID
LOAD, id            id=!CLOAD, !DLOAD, !TEMPERATURE で定義した GRPID
CONTACT, id         id=!CONTACT で定義した GRPID
```

使用例

自動増分調整を有効にし、初期時間増分 0.01、ステップ時間幅 2.5、時間増分下限 1E-5、時間増分上限 0.3、 最大サブステップ数を 200 に設定する

```
!STEP, INC_TYPE=AUTO, SUBSTEPS=200
0.01, 2.5, 1E-5, 0.3
```

自動増分調整を有効にし、時刻リスト TP1 を計算・結果出力時刻として指定する

```
!STEP, INC_TYPE=AUTO, TIMEPOINTS=TP1
0.1, 2.0, 1E-3, 0.2
```

備考

- 自動増分調整の場合、SUBSTEPS は最大サブステップ数として扱われる
- 時刻リスト名 TIMEPOINTS および自動接触パラメータセット名 AUTOINCPARAM の指定は INC_TYPE=AUTO のときのみ有効
- TIMEPOINTS を指定する場合、指定先の!TIME_POINT は!STEP カードより前に定義されていなければならない。
- AUTOINCPARAM を指定する場合、指定先の!AUTOINC_PARAM は!STEP カードより前に定義されていなければならない。また、本パラメータ省略時はデフォルトの自動増分パラメータセットが使用される

自動時間増分調整制御・!AUTOINC_PARAM

パラメータ

パラメータ名	属性	内容
NAME	C	自動増分パラメータ名（必須）

2 行目

減少条件およびその際の時間増分減少率を指定する。

（2 行目） `RS, NS_MAX, NS_SUM, NS_CONT, N_S`

変数名	属性	内容
RS	R	時間増分減少率（デフォルト：0.25）
NS_MAX	I	Netwon 法最大反復数の閾値（デフォルト：10）
NS_SUM	I	Netwon 法合計反復数の閾値（デフォルト：50）
NS_CONT	I	接触反復数の閾値（デフォルト：10）
N_S	I	減少条件成立までのサブステップ数（デフォルト：1）

3 行目

増加条件およびその際の時間増分増加率を指定する。

（3 行目） `RL, NL_MAX, NL_SUM, NL_CONT, N_L`

変数名	属性	内容
RL	R	時間増分増加率（デフォルト：1.25）
NL_MAX	I	Netwon 法最大反復数の閾値（デフォルト：1）
NL_SUM	I	Netwon 法合計反復数の閾値（デフォルト：1）
NL_CONT	I	接触反復数の閾値（デフォルト：1）
N_L	I	増加条件成立までのサブステップ数（デフォルト：2）

4 行目

カットバックの設定を行う。

（3 行目） `RC, N_C`

変数名	属性	内容
RC	R	カットバック時の時間増分減少率（デフォルト：0.25）
N_C	I	連続カットバック回数の許容上限（デフォルト：5）

使用例

デフォルト設定と同じ設定の場合

```
!AUTOINC_PARAM, NAME=AP1
0.25, 10, 50, 10, 1
1.25,  1,  1,  1, 2
0.25,  5
```

時刻リストの定義・!TIME_POINTS

パラメータ

パラメータ名	属性	内容

NAME	C	時刻リスト名（必須）
TIME	C	STEP（ステップ開始時刻起点の時刻による入力・デフォルト値）／TOTAL（初期からの通算時刻による入力）
GENERATE	-	開始時間、終了時間、時間間隔による時刻点の自動生成

2 行目以降（GENERATE を使用しない場合）

（2 行目以降） `TIME`

変数名	属性	内容
TIME	R	時刻

2 行目（GENERATE を使用する場合）

（2 行目） `STIME, ETIME, INTERVAL`

変数名	属性	内容
STIME	R	開始時刻
ETIME	R	終了時刻
INTERVAL	R	時刻点の間隔

使用例

時刻 1.5, 2.7, 3.9 を通算時刻として GENERATE を使用せず定義

```
!TIME_POINTS, TIME=TOTAL, TIME=,NAME=TP1
1.5
2.7
3.9
```

時刻 1.5, 2.7, 3.9 をステップ時刻として GENERATE を使用して定義

```
!TIME_POINTS, TIME=STEP, GENERATE, NAME=TP1
1.5, 3.9, 1.2
```

備考

- 時刻点の入力は昇順に行わなければならない

機能仕様補足：自動増分調整・カットバック機能

本開発で実施したその他の変更点

初期状態での結果出力

初期時刻の状態をステップカウント 0 の結果ファイルに出力するようにした。 出力の有無は!WRITE, RESURT/VISUAL の設定に準じるが、出力の指定がある場合には、 最終ステップ同様 FREQUENCY の指定によらず必ず出力される。

接触反復回数が上限に到達した際の挙動変更

接触反復を上限回数まで繰り返しても収束しなかった場合、 Newton 法同様に未収束エラーとみなしてつり合い試行を終了するように仕様を変更する。 なお、従来は反復を抜けて解析を継続する仕様であったが、 これは意図してそのようになったものではない（すなわち不具合であった）と確認している。

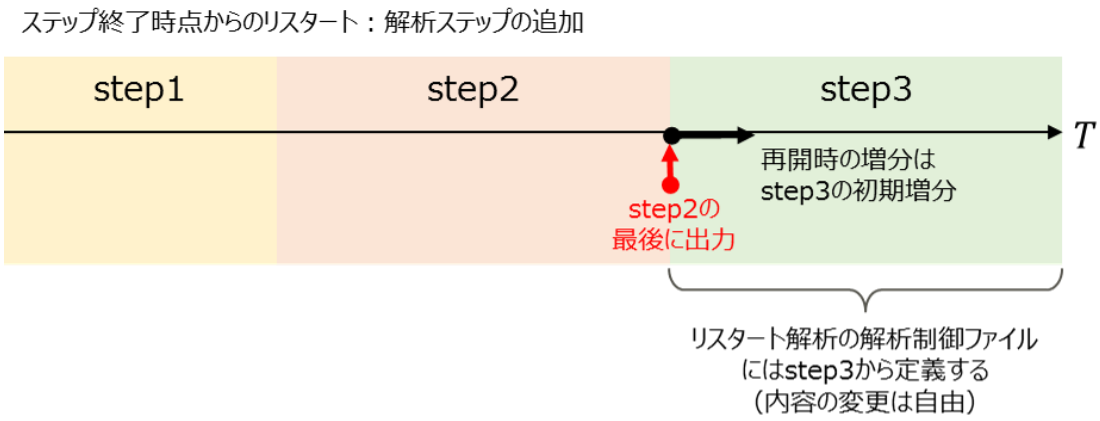
リスタートの仕様変更

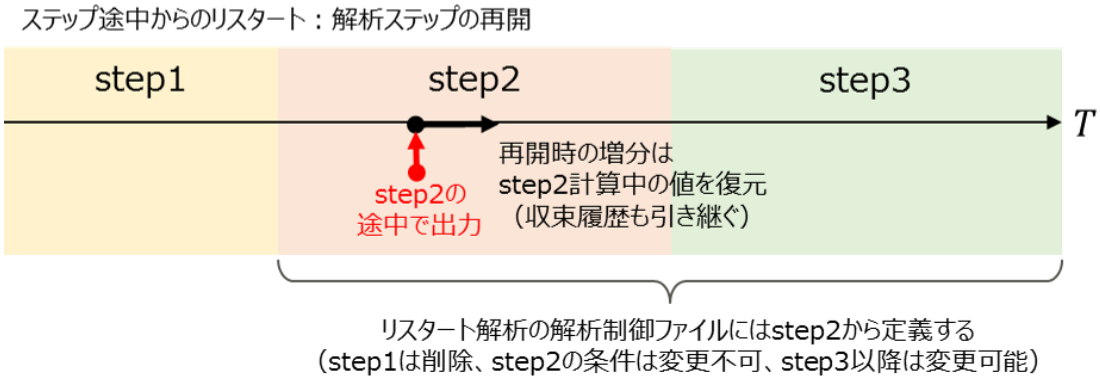
現在の仕様でリスタートを行うとき、解析制御ファイルに終了した!STEP の情報をそのまま残しておかなければならないが、 これを再開するステップ以降の情報だけ入力するよう仕様変更を行う：

- リスタート解析を有効にするには!RESTAT, FREQUENCY に負の値を設定する（現行通り）
- 解析制御ファイルには再開するステップ以降の!STEP のみを記述する。
  - リスタートファイルがステップの最後で出力されたものである場合
    - リスタート解析の解析制御ファイルに定義された最初のステップが、次のステップであるものとして解析が再開される
    - 時間増分はその!STEP で定義された初期増分に設定される
  - リスタートファイルがステップの途中で出力されたものである場合
    - リスタート解析の解析制御ファイルに定義された最初のステップが、中断ステップであるものとして解析が再開される
    - 中断ステップの定義はリスタート実行前のものから変更してはならない
    - リスタート解析は、中断ステップのファイル出力時刻から開始される
    - 時間増分はリスタート前の値が復元される。Newton 法の収束情報（自動増分で用いる情報）も引き継がれる
- （ステップを削除しても）リスタート解析の時刻はリスタート前の解析から継続される
- リスタートファイルがステップの最後で出力されたか否かは、読み込み実行時にメッセージを出力して知らせる

重要な制約事項

- 荷重条件定義!CLOAD,!DLOAD,!TEMPERATURE は、次の場合を除いて先行解析での定義を変更、削除、追加してはならない。
  - 先行ステップで使用されていない GRPID の荷重条件を変更・追加する場合





restart

内部仕様は「実装に関する説明」の「リスタートの仕様変更」に記載する。

実施例

例えば、以下に示す 2 ステップからなる解析をリスタート機能を使って継続することを考える

【目標とする解析の解析制御ファイル】

```
!RESTART, FREQUENCY=1
# BOUNDARY, LOAD 等の定義
# step1
!STEP, SUBSTEPS=100, INC_TYPE=AUTO
0.1, 1.0, 0.001, 1.0
BOUNDARY, 1
LOAD, 1
# step2
!STEP, SUBSTEPS=100, INC_TYPE=AUTO
0.3, 2.0, 0.001, 1.0
BOUNDARY, 1
LOAD, 1
LOAD, 2
!END
```

シナリオ１：ステップの追加

リスタートファイルを step1 の終了時点で出力して解析を一旦終了する。その後、リスタート機能を用いて step2 の解析を追加実行する。 例えば、step2 の条件を色々変えて解析を試行したい場合など。

【最初の解析の解析制御ファイル】

```
!RESTART, FREQUENCY=1
# BOUNDARY, LOAD 等の定義
# step1
!STEP, SUBSTEPS=100, INC_TYPE=AUTO
0.1, 1.0, 0.001, 1.0
BOUNDARY, 1
LOAD, 1
!END
```

【リスタート解析の解析制御ファイル】step2 の情報から記載を開始する。!RESTART, FREQUENCY に負の値を設定する。 定義された最初のステップが step2 として扱われ、初期増分は 0.3 に設定される。このとき、GRPID=1 の荷重条件は変更してはならない。GRPID=2 の荷重条件および step2 の!STEP の設定については、リスタートに関する制限はない。

```
!RESTART, FREQUENCY=-1
# BOUNDARY, LOAD 等の定義…※GRPID=1 の荷重は最初の解析の解析制御ファイルと同一でなければならない
# step2
!STEP, SUBSTEPS=100, INC_TYPE=AUTO
0.3, 2.0, 0.001, 1.0
BOUNDARY, 1
LOAD, 1
LOAD, 2
!END
```

シナリオ２：ステップの再開

外部的な要因による中断や、途中での発散などが原因で step1 の途中（仮に時刻 0.5 とする）で 出力されたリスタートファイルがある場合に、step1 の解析を再開する。 定義された最初のステップは step1 として扱われ、時刻 0.5 から解析が再開する。増分はリスタートファイルの値が復元される。

【最初の解析の解析制御ファイル】…【目標とする解析の解析制御ファイル】を使用しているとする。

【リスタート解析の解析制御ファイル】…【目標とする解析の解析制御ファイル】で そのまま継続する場合は、!RESTART, FREQUENCY に負の値を設定してリスタートする。定義された最初のステップは step1 として扱われ、増分はリスタート時点のものが復元される。発散したなどの理由によって条件を修正したい場合には、GRPID=1 の荷重条件定義および step1 の時間幅、有効な境界条件の組「以外」の情報（step1 の最小時間増分、!AUTOINC_PARAM の設定など。GRPID=2 の荷重条件および step2 の!STEP の設定も）が変更可能である。

```
!RESTART, FREQUENCY=-1
# BOUNDARY, LOAD 等の定義…※
# step1
```

```
!STEP, SUBSTEPS=100, INC_TYPE=AUTO
0.1, 1.0, 0.001, 1.0
BOUNDARY, 1
LOAD, 1
# step2
!STEP, SUBSTEPS=100, INC_TYPE=AUTO
0.3, 2.0, 0.001, 1.0
BOUNDARY, 1
LOAD, 1
LOAD, 2
!END
```

ログ出力

FSTR.sta に現在の解析の進捗を出力する。具体的には、サブステップの試行計算が実行される度に 以下に挙げる情報が追記される。

- STEP：ステップ番号
- SUB STEP：サブステップ番号
- STAT：収束状態...S は収束、F は収束に失敗。F の前に付く数字は連続して失敗した回数。
- # of CONT ITER：接触反復回数
- MAX # NEWTON ITER：Newton 法反復回数の最大値
- TOT # NEWTON ITER：Newton 法反復回数の合計値
- START TIME：開始時刻
- TIME INC：時間増分
- END TIME：終了時刻（カットバックが実行された場合は開始時刻から変化しない）
- MESSAGE：メッセージが出力される

従来出力されていたサブステップの計算時間は FSTR.msg に出力先が変更になっている。

```
-----------+-------------------------------------------------------------+------------
----------------------------
           |       # of  MAX #  TOT #                                     |

 STEP  SUB |  STAT CONT NEWTON NEWTON       START        TIME         END | MESSAGE

      STEP |       ITER   ITER   ITER        TIME         INC        TIME |

-----------+-------------------------------------------------------------+------------
----------------------------
    1    1 |    1F    0      7      7  0.0000E+00  2.0000E-01  0.0000E+00 | Failed to c
onverge due to MAXITER.
    1    1 |     S    0      7      7  0.0000E+00  1.0000E-01  1.0000E-01 |

    1    2 |     S    0      7      7  1.0000E-01  1.0000E-01  2.0000E-01 |

    1    3 |     S    0      7      7  2.0000E-01  1.0000E-01  3.0000E-01 |

    1    4 |     S    0      7      7  3.0000E-01  1.0000E-01  4.0000E-01 |
```

```
----------+--------------------------------------------------------------+-----------
----------------------------
FSTR_SOLVE_NLGEOM HAS COMPLETED SUCCESSFULLY
```

熱伝導解析結果ファイルを使用した熱応力解析の取り扱い

1. 当該機能は!TEMPERATURE, READRESULT が指定された場合に有効になり、その際は 1 ステップ目のみの解析となる。
2. 初期温度は!INITIAL CONDITION で指定する。読み込まれる温度分布は READRESULT, SSTEP, INTERVAL で指定される result ファイル一式である。
3. 2.で指定された result ファイルから読み込まれる温度分布は時間軸上等間隔に配置される。
4. 中間的な時刻の解析を行う場合、解析時刻の前後の温度分布を、時刻によって線形補間した時刻を用いる。

実装に関する説明

メインループ：FSTR_SOLVE_NLGEOM

仕様に示す自動時間増分調整およびカットバック処理のフローは FSTR_SOLVE_NLGEOM において実装されている。 時刻管理・調整、カットバック処理、入力ファイル処理等の要素機能は別途モジュールを作成・改修して実装している。

解析上の時間について

解析上の時間管理は fistr1/src/analysis/static/fstr_Ctrl_TimeInc.f90 に定義されている m_fstr_TimeInc で行う。 現在時刻 current_time、時間増分 time_inc およびその基準値 time_inc_base が private なモジュール変数として定義され、このモジュール内の関数を通じて外部から設定を行うようになっている。

カットバック処理について

解析結果は fstrSOLID に新たに追加された変数

```
real(kind=kreal), pointer :: unode_bkup(:)     => null() !< disp at the beginning of cu
rr step (backup)
real(kind=kreal), pointer :: QFORCE_bkup(:)    => null() !< equivalent nodal force (bac
kup)
real(kind=kreal), pointer :: last_temp_bkup(:) => null()
type( tElement ), pointer :: elements_bkup(:)  => null() !< elements information (backu
p)
type( tContact ), pointer :: contacts_bkup(:)  => null() !< contact information (backu
p)
```

に格納される。順に変位、内力、温度、要素積分点状態、接触状態のバックアップ先であり、変数末尾に_bkup を追加している。 バックアップとしては若干冗長な変数の保存方法である（例えば unode はバックアップしなくとも更新処理をスキップするだけでよい）が、ソースの可読性・拡張性の観点から保存の必要な領域を単純に 2 重に保持する実装としている。

カットバック処理の実体（状態保存変数の初期化、終了、状態の保存、読み出し）は fistr1/src/analysis/static/fstr_Cutback.f90 に定義されている。

今後の開発において状態変数の追加を行う際には、 節点変数であれば unode, QFORCE にならって fstrSOLID 配下に real ポインタ配列として_bkup 変数を追加した上で、カットバック処理の実体コードを改修する。

積分点変数であれば、モジュール mMechGauss に定義されている tGaussStatus のメンバ変数を増やし、同モジュール内で定義されているコピーサブルーチン fstr_copy_gauss をそれに応じて改修すれば 自動的にカットバック処理が適用されるようになる。

接触状態変数も同様に、モジュール m_contact_lib に定義されている tContactState のメンバ変数を増やし、 同モジュール内で定義されているコピーサブルーチン contact_state_copy をそれに応じて改修すれば 自動的にカットバック処理が適用されるようになる。

時刻点リストについて

時刻点リストは fstr1/src/lib/m_timepoint.f90 に定義されている m_timepoint モジュールに実装されている。 外部仕様上は GENERATE で生成されている場合でも、内部的には時間を格納する real 配列として確保している。

現在時刻が時刻点であるかどうかを返す is_at_timepoints、次の時間点までの残り時刻を返す get_remain_to_next_timepoints は、倍精度浮動小数点数同士の一致比較に相当する処理を含むが、 これはしばしば数値誤差によって誤りの結果を返す場合があるため、1d-10 だけ値をシフトしている。

その他の内部仕様変更

時刻管理変数の一元化

積分点時刻 element(i)%gausses(j)%ttime は廃止とし、時間管理は m_fstr_TimeInc に統一した。 その際、積分点時刻を参照していたルーチンに時刻を受け渡すインターフェースを追加した。 詳細は git の[To remove gauss%ttime]から始まる一連のコミットを参照のこと。

関係した主なサブルーチン：fstr_solve_nonlinear, fstr_Stiff_matrix, fstr_Update_Newton, MatlMatrix, StressUpdate, STF_C3* および Update_C3*

リスタートの仕様変更

まず現行仕様において、解析制御ファイルに終了した!STEP の情報を そのまま残しておかなければならないのは、以下のような内部仕様が原因である。

1. 現行仕様ではリスタート地点の算出は fstr_read_restart で行われる。その際、リスタートファイルに格納されている番号が ステップの最終サブステップであるか否かを判定するために、（解析制御ファイルから取得した）前ステップにおけるサブステップ数を参照する。
2. リスタート解析はステップカウント tot_step をリスタート前の値を復元して実行される。このため、 tot_step を index に用いてアクセスされる!STEP 情報構造体 fstrSOLID%step_ctrl(:)のデータ構成は、 リスタート前と全く同じでなければならない。!STEP を削除するとこの index がずれてしまい、正しく解析条件設定ができない。

3. さらに荷重条件処理 fstr_ass_load では、前ステップでの荷重状態を引き継ぐ仕様になっている。すなわち、2 ステップ目以降では 1 つ前のステップで有効であった荷重条件はすべて 100%の荷重がかかった状態で有効とする。そのために、前ステップの荷重情報を参照する。

それぞれ、以下のように改修して対応する

1. リスタートファイルに、リスタート解析で冒頭に実行されるステップの開始時刻 starttime_restartstep を追記する。
   - リスタートファイル出力における starttime_restartstep の値は、ステップ終了時には現時刻を、それ以外は当該ステップの starttime を設定する。
   - リスタートファイル読み込み時の処理は次の通り
     - starttime_restartstep = リスタート時刻であればステップ終了時に出力されたファイルと判断し、 再開時は restart_step_num を+1、restart_substep_num を 1 に設定する
     - starttime_restartstep < リスタート時刻であればステップ途中に出力されたファイルと判断し、 再開時は restart_step_num をそのままに、restart_substep_num を+1 に設定する
     - リスタート解析では stepinfo のスタート時間を starttime_restartstep だけシフトする。これにより、 外部仕様通りの挙動が実現できる。
2. FSTR_SOLVE_NLGEOM におけるステップカウント変数 tot_step の解釈を、解析全体のステップカウントから実行中の解析におけるステップカウントに変更する。すなわち、tot_step はリスタート開始ステップカウント restart_step_num ではなく常に 1 からカウントが始まる。このとき、stepinfo 構造体へアクセスする index は tot_step、外部表示用の通算ステップカウントは tot_step+restart_step_num-1 となる。
3. リスタートファイルに有効な荷重条件の grpid を記録する。リスタートの先頭ステップにおける fstr_ass_load では、前ステップからの継続荷重として記録された grpid を参照する
   - ステップ終了時の出力であれば、現ステップでの有効な荷重条件を出力する
   - ステップ途中の出力であれば、前ステップでの有効な荷重条件を出力する
   - 最初に有効な制約条件の個数を出力し、次にその個数分だけ有効な制約条件の grpid を出力する

あわせて下記の修正を行う

- リスタートファイルの出力が指定されている場合、出力間隔の指定によらずステップの最後では必ずリスタートファイルを出力する （自動増分により出力間隔指定ではステップ最後の出力が保証されないため、上記項目 3.の対応のため）
- fstrSOLID%temp_bak(:)=fstrSOLID%temperature(:)の更新箇所を、現在のステップ冒頭から解析冒頭＋ステップ末尾に変更する。 ステップ途中からリスタートした場合に、fstrSOLID%temp_bak が正しく設定されないため。

仕様検討の記録

先行ステップの荷重条件の記録については、有効な節点荷重ベクトルを保存する節点変数 fstrSOLID%GL_bak の新設も検討を行った。 しかしながら現在の FrontISTR の仕様では、リスタートフ

ァイルから引き継いだ荷重をその後ずっと継続させることが不可能になるため、 以下の案は却下された。同時に、制約事項「先行ステップで使う荷重定義の変更不可」が追加された。

3. 前ステップで有効な節点荷重ベクトルを保存する節点変数 fstrSOLID%GL_bak(:)を新設し、fstr_ass_load における 「if(cstep > 1) if(fstr_isLoadActive(fstrSOLID, grpid, cstep-1)) factor = 1.0d0」の処理を廃止する。 fstrSOLID%GL_bak はリスタートファイルに書き出される。解析冒頭に 0 かリスタートファイルの値が格納され、 ステップ終了時にその時点の fstrSOLID%GL の値に更新される。temp_bak についても同様にリスタート出力・復元を追加する。
   - なお、GL=factor×今ステップで有効な荷重+前ステップで有効な荷重=factor*F_n+GL_bak であるので、ステップをまたぐとき(factor=1) の GL_bak の更新は GL_bak=GL-GL_bak である点に注意する（GL_bak を引かないと前前ステップや前前前ステップの荷重が引き継がれてしまう）

判断した。