日本建築学会技術報告集　第5号，281-286，1997年12月
AIJ J. Technol. Des. No. 5, 281-286, Dec., 1997

コンピュータによる建築構造設計支援のあり方に関する研究
―(その2)　FACT法の優位性―

STUDY REGARDING THE REQUIREMENTS FOR BUILDING STRUCTURAL DESIGN SUPPORTED BY THE COMPUTER
(Part 2)　Superiority of the FACT method

稲田達夫 ――*1


*1

Tatsuo INADA

In this report, the notation and the principle of operation of the FACT method will be explained first by means of an extremely simple example of application of the FACT method. Then the superiority of the FACT method to other programming methods will be clarified by giving an example of application of the FACT method to the vibration response analysis program.

キーワード：
建築構造設計支援プログラム，プログラム開発手法，履歴減衰装置

Keywords:
Building structural design suported by computer, Program development method, Hysteretic damper

1．はじめに

建築構造設計において，ｺﾝﾋﾟｭｰﾀが不可欠であることは，改めて言うまでもない事であろう。しかし一方，ｺﾝﾋﾟｭｰﾀ利用が高度化する中で，新たな問題が発生しつつあるのも事実である。

既報（その1）で筆者は，人間の典型的な知的活動としての建築構造設計を支援するｺﾝﾋﾟｭｰﾀﾌﾟﾛｸﾞﾗﾑはいかにあるべきかという事柄について考察を行い，建築構造設計支援ﾌﾟﾛｸﾞﾗﾑの要件として，

・「構造」的把握に対する支援
・拡張性及び再利用性
・正確さ　・柔軟な操作性
・fortran言語との親和性

の5点が重要であることを述べた。そして，このような要件を満足するためには，既存のﾌﾟﾛｸﾞﾗﾑ方式では困難であり，それを解決する方法として，ＦＡＣＴ法（Fortran Auto-Coding Tool）と呼ぶ新しいﾌﾟﾛｸﾞﾗﾑ方式を提案した。

本報では，まず最初にＦＡＣＴ法の極めて簡単な適用例を示すことにより，ＦＡＣＴ法の記述形式と動作方式を説明する。次に，ＦＡＣＴ法を具体的な建築構造設計支援ﾌﾟﾛｸﾞﾗﾑに適用した事例を示すことにより，ＦＡＣＴ法を他のﾌﾟﾛｸﾞﾗﾑ方式と比較した場合の優位性を明らかにする。

2．既存のプログラム方式の問題点

本章では，極めて簡単なﾌﾟﾛｸﾞﾗﾑ例を対象として考察を進めることにより，既存のﾌﾟﾛｸﾞﾗﾑ方式が，本来の人間の要求に対し，いかに柔軟性を欠いているかを明らかにする。

a）簡単なﾌﾟﾛｸﾞﾗﾑ例

ここでは，極めて簡単なﾌﾟﾛｸﾞﾗﾑ例として，「円柱の底面の半径Ｒと高さＨを入力し，円柱の底面積Ａと体積Ｖを出力する」ﾌﾟﾛｸﾞﾗﾑを記述することを考える。

このﾌﾟﾛｸﾞﾗﾑを作成するために必要な機能としては，

・半径Ｒより底面積Ａを求める機能
・底面積Ａと高さＨより体積Ｖを求める機能

の2つが必要である。

b）既存のプログラム方式による記述と問題点

ここでは，第1の機能をｻﾌﾞﾌﾟﾛｸﾞﾗﾑ MENSEK，第2の機能をｻﾌﾞﾌﾟﾛｸﾞﾗﾑ TAISEKで記述することにしよう。そうすると，上記の問題をFORTRAN言語で記述すれば，図1のようなﾌﾟﾛｸﾞﾗﾑが書けるであろう。

しかしこのﾌﾟﾛｸﾞﾗﾑは，利用者の意図の所在の如何によっては，きわめて不親切なﾌﾟﾛｸﾞﾗﾑと見ることもできる。というのは，利用者の意図に基づく様々な入力操作と出力要求の可能性を列記すれば，例えば図2に示すような多くのｹｰｽが想定される。しかし，図1で示し

*1 三菱地所㈱丸ビル改築設計室　副室長・博士(工学)
（〒100　千代田区丸の内2-7-3　東京ビル3F）

*1 Mitsubishi Estate Co., Ltd.

たﾌﾟﾛｸﾞﾗﾑ1は上記のｹｰｽ2の操作にはよく対応しているが，他のｹｰｽの場合については，それぞれの操作の手順とは対応しておらず，もしﾌﾟﾛｸﾞﾗﾑ1で利用者の意図を満足させるためには，利用者に対し何らかの形での余計な操作を強いることになる。

それでは，利用者の持つ様々な意図を満足させるためにはどのようなﾌﾟﾛｸﾞﾗﾑを準備すればよいだろうか。それで，利用者の意図に基づく様々な入出力操作の要求に応えることが可能なﾌﾟﾛｸﾞﾗﾑとして，図３に示すﾌﾟﾛｸﾞﾗﾑ2を提示する。ﾌﾟﾛｸﾞﾗﾑ2を利用して，画面に表示されたﾒﾆｭｰに従って入出力操作を進めれば，利用者の意図に基づく様々な要求に応えることが可能となる。

```
      REAL R, H, A, V
      READ(5, *) R
      READ(5, *) H
      CALL MENSEK(R, A)
      CALL TAISEK(A, H, V)
      WRITE(6, *) A
      WRITE(6, *) V
      STOP
      END
```

図１）ﾌﾟﾛｸﾞﾗﾑ1

しかし，ﾌﾟﾛｸﾞﾗﾑ2もまた利用者にとって余り親切なﾌﾟﾛｸﾞﾗﾑとは言えない一面を持っている。それは，ﾌﾟﾛｸﾞﾗﾑ2の中程で網かけで示した２行であるが，このﾌﾟﾛｸﾞﾗﾑではどの様な出力を行う場合でも必ずｻﾌﾞﾌﾟﾛｸﾞﾗﾑ MENSEKとTAISEKが呼ばれるようになっている。しかし，入出力操作の手順によっては必ずしも両方のｻﾌﾞﾌﾟﾛｸﾞﾗﾑを呼ぶ必要の無い場合も少なくない。これは，場合よっては利用者に対し意味のない実行を強制することにもなる。MENSEK，TAISEKがもし多大な実行時間を要する処理であるとしたならば，ﾌﾟﾛｸﾞﾗﾑ2は利用者にとって耐え難いﾌﾟﾛｸﾞﾗﾑとなるであろう。

それでは，上記で示した利用者の意図に基づいて想定される様々な入出力操作の各ｹｰｽに対応して，無駄のない実行が可能なﾌﾟﾛｸﾞﾗﾑとしては，どのような方式が考えられるであろうか。その１つの方法として，図４に示すﾌﾟﾛｸﾞﾗﾑ3を提示する。このﾌﾟﾛｸﾞﾗﾑ3では，利用者にｻﾌﾞﾌﾟﾛｸﾞﾗﾑを呼ぶﾀｲﾐﾝｸﾞを指示させることにより，上記の問題を解決している。

しかし，このﾌﾟﾛｸﾞﾗﾑ3は別の意味で大きな問題を含んでいる。それは，無駄のないﾌﾟﾛｸﾞﾗﾑの起動のﾀｲﾐﾝｸﾞの選定を，ﾌﾟﾛｸﾞﾗﾑ3では利用者の判断に委ねている点である。このことは，このﾌﾟﾛｸﾞﾗﾑを使用する場合には，利用者は常に自分にとって必要なｻﾌﾞﾌﾟﾛｸﾞﾗﾑの実行のﾀｲﾐﾝｸﾞを意識しておかなければならず，利用者に対しﾌﾟﾛｸﾞﾗﾑ2の場合とは異なる種類の負担を強いることになる。また，別の見方をすれば，ﾌﾟﾛｸﾞﾗﾑ3では利用者が判断を誤れば，利用者に対し誤った情報を提供する危険性を秘めていることにもなる。

```
      REAL R, H, A, V
   10 WRITE(6, 1010)
 1010 FORMAT(1H , '*****************'/
     ¥       1H , '*DATA INPUT (1)*'/
     ¥       1H , '*DATA OUTPUT(2)*'/
     ¥       1H , '*SYUURYOU   (0)*'/
     ¥       1H , '*****************')
      READ(5, *) IOP
      IF(IOP.EQ.0) THEN
          STOP
      ELSEIF(IOP.EQ.1) THEN
          WRITE(6, 1020)
 1020   FORMAT(1H , '**********************'/
     ¥         1H , '*TEIMEN NO HANKEI(1)*'/
     ¥         1H , '*ENCHUU NO TAKASA(2)*'/
     ¥         1H , '**********************')
          READ(5, *) JOP
          IF(JOP.EQ.1) THEN
              READ(5, *) R
              GO TO 10
          ELSEIF(JOP.EQ.2) THEN
              READ(5, *) H
              GO TO 10
          ELSE
              GO TO 10
          ENDIF
      ELSEIF(IOP.EQ.2) THEN
          CALL MENSEK(R, A)
          CALL TAISEK(A, H, V)
          WRITE(6, 1030)
 1030   FORMAT(1H , '**************************'/
     ¥         1H , '*ENCHUU NO TAISEKI   (1)*'/
     ¥         1H , '*ENCHUU NO TEIMENSEKI(2)*'/
     ¥         1H , '**************************')
          READ(5, *) KOP
          IF(KOP.EQ.1) THEN
              WRITE(6, *) '**V=', V
              GO TO 10
          ELSEIF(KOP.EQ.2) THEN
              WRITE(6, *) '**A=', A
              GO TO 10
          ELSE
              GO TO 10
          ENDIF
      ELSE
          GO TO 10
      ENDIF
      END
```

図３）ﾌﾟﾛｸﾞﾗﾑ2

ケース１（Ｒを入力し直ちにＡを確認する。次にＨを入力しＶを表示する）
①Ｒを入力する。
②Ａを表示する。
③Ｈを入力する。
④Ｖを表示する。

ケース２（Ｒ，Ｈを順次入力し，引き続きＡ，Ｖを順次表示する）
①Ｒを入力する。
②Ｈを入力する。
③Ａを表示する。
④Ｖを表示する。

ケース３（Ｒ，Ｈを順次入力した後，Ｖを先に表示しＡを後で表示する）
①Ｒを入力する。
②Ｈを入力する。
③Ｖを表示する。
　動する。
④Ａを表示する。

ケース４（Ｒを固定し，Ｈの変化に伴うＶの変化を見る）
①Ｒを入力する。
②Ｈを入力する。
③Ｖを表示する。
④Ｈを入力する。
⑤Ｖを表示する。

ケース５（Ｈを固定し，Ｒの変化に伴うＶの変化を見る）
①Ｈを入力する。
②Ｒを入力する。
③Ｖを表示する。
④Ｒを入力する。
⑤Ｖを表示する。

図２

３．ＦＡＣＴ法の記述形式と動作原理

さて，それではどのようにすれば以上述べた様な問題を解決することができるであろうか。それでまず，利用者の入力操作と出力要求に対応する，無駄のないﾌﾟﾛｸﾞﾗﾑの起動のﾀｲﾐﾝｸﾞの決定の方法について考えてみよう。固有の機能を有するﾌﾟﾛｸﾞﾗﾑの起動の必要性は，利用者が出力要求を発行した時点において始めて生じると考えるのが自然であろう。従って，利用者の出力要求に対応して，その出力を行うために必要な最小限のﾌﾟﾛｸﾞﾗﾑの起動を行うようにすれば，この問題は解決する。図２のｹｰｽ1から5の問題について，利用者の出力要求に即応したﾌﾟﾛｸﾞﾗﾑ起動の手順を示せば，図５の通りとなる。

しかし，このような出力要求に即応して最も無駄のないﾌﾟﾛｸﾞﾗﾑ起動の手順を選択する問題を，既存の手続き型のﾌﾟﾛｸﾞﾗﾑ言語を使用して記述することは困難である。なぜならば，例えばここに提示したような極めて簡単なﾌﾟﾛｸﾞﾗﾑ例においてさえも，利用者の意図に基づく入出力操作に対応したﾌﾟﾛｸﾞﾗﾑ起動のﾀｲﾐﾝｸﾞは極めて多岐に渡り，通常の手続き型のﾌﾟﾛｸﾞﾗﾑ方式の約束に従って場合分けを行い全ての

ｹｰｽを通常の手続き命令として整理し記述することは，ほとんど不可能と考えられるからである。さらに，ﾃﾞｰﾀ数も多く，ﾃﾞｰﾀ間相互の関係を規定するｻﾌﾞﾌﾟﾛｸﾞﾗﾑの数も多岐にわたるような問題を解決するためには，どのような方法が考えられるであろうか。

ここに示したような問題のことを，「ﾃﾞｰﾀの首尾一貫性の保証の問題」と呼ぶことにしよう。「ﾃﾞｰﾀの首尾一貫性の保証の問題」がﾃｰﾏとなるﾌﾟﾛｸﾞﾗﾑは，現実世界においては多々存在する。「ﾃﾞｰﾀの首尾一貫性の保証の問題」を解決するために考案された既存のﾌﾟﾛｸﾞﾗﾑ開発方式として，例えばﾃﾞｰﾀﾍﾞｰｽﾏﾈｼﾞﾒﾝﾄｼｽﾃﾑがある。しかし，ﾃﾞｰﾀﾍﾞｰｽﾏﾈｼﾞﾒﾝﾄｼｽﾃﾑでは，上記のようなﾃﾞｰﾀ相互の関係がﾌﾟﾛｸﾞﾗﾑにより規定されるような自由度を要求する問題に対しては，対応困難であるのが実状である。

```
      REAL R, H, A, V
   10 WRITE(6, 1010)
 1010 FORMAT(1H , '*******************'/
     ¥       1H , '*DATA INPUT  (1)*'/
     ¥       1H , '*DATA OUTPUT (2)*'/
     ¥       1H , '*PROGRAM CALL(3)*'/
     ¥       1H , '*SYUURYOU    (0)*'/
     ¥       1H , '*******************')
      READ(5, *) IOP
      IF(IOP.EQ.0) THEN
          STOP
      ELSEIF(IOP.EQ.1) THEN
          WRITE(6, 1020)
 1020     FORMAT(1H , '**********************'/
     ¥          1H , '*TEIMEN NO HANKEI(1)*'/
     ¥          1H , '*ENCHUU NO TAKASA(2)*'/
     ¥          1H , '**********************')
          READ(5, *) JOP
          IF(JOP.EQ.1) THEN
              READ(5, *) R
              GO TO 10
          ELSEIF(JOP.EQ.2) THEN
              READ(5, *) H
              GO TO 10
          ELSE
              GO TO 10
          ENDIF
      ELSEIF(IOP.EQ.2) THEN
          WRITE(6, 1030)
 1030     FORMAT(1H , '***************************'/
     ¥          1H , '*ENCHUU NO TAISEKI   (1)*'/
     ¥          1H , '*ENCHUU NO TEIMENSEKI(2)*'/
     ¥          1H , '***************************')
          READ(5, *) JOP
          IF(JOP.EQ.1) THEN
              WRITE(6, *) '**V=', V
              GO TO 10
          ELSEIF(JOP.EQ.2) THEN
              WRITE(6, *) '**A=', A
              GO TO 10
          ELSE
              GO TO 10
          ENDIF
      ELSEIF(IOP.EQ.3) THEN
          WRITE(6, 1040)
 1040     FORMAT(1H , '********************'/
     ¥          1H , '*MENSEKI KEISAN(1)*'/
     ¥          1H , '*TAISEKI KEISAN(2)*'/
     ¥          1H , '********************')
          READ(5, *) JOP
          IF(JOP.EQ.1) THEN
              CALL MENSEK(R, A)
              GO TO 10
          ELSEIF(JOP.EQ.2) THEN
              CALL TAISEK(A, H, V)
              GO TO 10
          ELSE
              GO TO 10
          ENDIF
      ELSE
          GO TO 10
      ENDIF
      END
```

図４）ﾌﾟﾛｸﾞﾗﾑ3

このような，「データの首尾一貫性の保証の問題」を解決するために考案したのが，ＦＡＣＴ法なのである。

ａ）ＦＡＣＴ法の記述形式

さてそれでは，上記の簡単な適用事例の問題について，「ﾃﾞｰﾀの首尾一貫性の保証の問題」をＦＡＣＴ法ではどのように解決しているかを見ることにしよう。それでまず，ＦＡＣＴ法を利用して上記の問題を記述すると以下の通りとなる。

ケース１（Ｒを入力し直ちにＡを確認する。次にＨを入力しＶを表示する）
①Ｒを入力する。
②Ａを表示する。　→　ＭＥＮＳＥＫを起動する。
③Ｈを入力する。
④Ｖを表示する。　→　ＴＡＩＳＥＫを起動する。

ケース２（Ｒ，Ｈを順次入力し，引き続きＡ，Ｖを順次表示する）
①Ｒを入力する。
②Ｈを入力する。
③Ａを表示する。　→　ＭＥＮＳＥＫを起動する。
④Ｖを表示する。　→　ＴＡＩＳＥＫを起動する。

ケース３（Ｒ，Ｈを順次入力した後，Ｖを先に表示しＡを後で表示する）
①Ｒを入力する。
②Ｈを入力する。
③Ｖを表示する。　→　ＭＥＮＳＥＫ，ＴＡＩＳＥＫを順次起動する。
④Ａを表示する。

ケース４（Ｒを固定し，Ｈの変化に伴うＶの変化を見る）
①Ｒを入力する。
②Ｈを入力する。
③Ｖを表示する。　→　ＭＥＮＳＥＫ，ＴＡＩＳＥＫを順次起動する。
④Ｈを入力する。
⑤Ｖを表示する。　→　ＴＡＩＳＥＫを起動する。

ケース５（Ｈを固定し，Ｒの変化に伴うＶの変化を見る）
①Ｈを入力する。
②Ｒを入力する。
③Ｖを表示する。　→　ＭＥＮＳＥＫ，ＴＡＩＳＥＫを順次起動する。
④Ｒを入力する。
⑤Ｖを表示する。　→　ＭＥＮＳＥＫ，ＴＡＩＳＥＫを順次起動する。

図５

（１）ﾃﾞｰﾀ関数関連図の作成

この簡単なﾌﾟﾛｸﾞﾗﾑ例の問題には，以下に示す２つの関係が内在することは明らかであろう。

関係１：MENSEKを起動することにより半径Rより底面積Aを求める。

関係２：TAISEKを起動することにより，底面積Aと高さHより体積Vを求める。

この２つの関係を図的に表現すれば図６ａのように記述することができる。

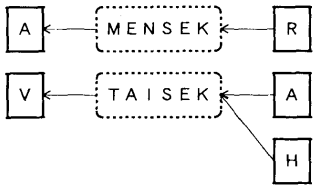

図６ａ

あるいは上記２つの関係を１まとめにすれば，図６ｂのように表現することができる。

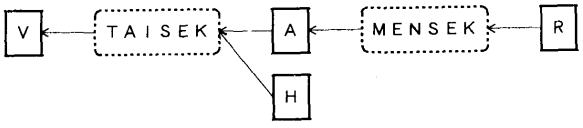

図６ｂ

このよな図式表現のことを，ＦＡＣＴ法では「ﾃﾞｰﾀ関数関連図」と呼ぶ。上記２つの表記法は互いに等価であり，どちらの表現方式を使用したとしても差し支えはないが，それぞれには以下のような特徴を持つ。

まず第１の表記法の特徴は，いわば箇条書きのような手軽さをｺﾝﾋﾟｭｰﾀﾌﾟﾛｸﾞﾗﾑの記述の世界に持ち込んだことである。思いつくまま

に順不同にﾃﾞｰﾀの関係を記述することによりﾌﾟﾛｸﾞﾗﾐﾝｸﾞが可能となれば，ﾌﾟﾛｸﾞﾗﾑの作成は飛躍的に効率化するはずである。このようなﾌﾟﾛｸﾞﾗﾑ方式のことを宣言的ﾌﾟﾛｸﾞﾗﾑ方式と呼ぶが，この方式を実現したことがＦＡＣＴ法の第１の特徴である。

第２の表現形式の特徴としては，ﾌﾟﾛｸﾞﾗﾑの全貌を概略的に把握したいとする立場に対して，極めて有効な表現形式と考えられることである。このことは，建築構造設計支援ﾌﾟﾛｸﾞﾗﾑのような，極めて複雑な論理構造を持つﾌﾟﾛｸﾞﾗﾑの作成に当たっては，その簡明さにおいて極めて有効な表現形式と見ることができるであろう。

```
      SUBROUTINE TAISEK(INAME, IN, ONAME, ON)
      CHARACTER*12 INAME(IN), ONAME(ON)
      REAL A, H, V
      CALL DBGET(INAME(1), A)
      CALL DBGET(INAME(2), H)
      V=A*H
      CALL DBWRIT(ONAME(1), V)
      RETURN
      END

      SUBROUTINE MENSEK(INAME, IN, ONAME, ON)
      CHARACTER*12 INAME(IN), ONAME(ON)
      REAL R, A
      CALL DBGET(INAME(1), R)
      A=3.141592*R**2
      CALL DBWRIT(ONAME(1), A)
      RETURN
      END
```

図７）関数ﾌﾟﾛｸﾞﾗﾑ

（２）関数ﾌﾟﾛｸﾞﾗﾑの作成

ＦＡＣＴ法における個々のｻﾌﾞﾌﾟﾛｸﾞﾗﾑのことを「関数ﾌﾟﾛｸﾞﾗﾑ」と呼ぶ。ＦＡＣＴ法における関数ﾌﾟﾛｸﾞﾗﾑの記述形式は，上記の簡単なﾌﾟﾛｸﾞﾗﾑ例の場合について見れば，図７の通りとなる。

（３）ＦＡＣＴﾒｲﾝﾌﾟﾛｸﾞﾗﾑの作成

ＦＡＣＴ法でﾌﾟﾛｸﾞﾗﾐﾝｸﾞを行う場合に最後に必要となる重要なﾌﾟﾛｸﾞﾗﾑとして，ＦＡＣＴﾒｲﾝﾌﾟﾛｸﾞﾗﾑがある。以下に，上記の簡単なﾌﾟﾛｸﾞﾗﾑ例の場合のＦＡＣＴﾒｲﾝﾌﾟﾛｸﾞﾗﾑの一例を図８に示す。

ここで，ＦＡＣＴﾒｲﾝﾌﾟﾛｸﾞﾗﾑの特徴としては，ＦＡＣＴﾒｲﾝﾌﾟﾛｸﾞﾗﾑが表現しているのは，ﾌﾟﾛｸﾞﾗﾑの利用者の入力操作と出力要求のみであり，通常のﾌﾟﾛｸﾞﾗﾑ方式では必須と考えられるｻﾌﾞﾌﾟﾛｸﾞﾗﾑの起動のﾀｲﾐﾝｸﾞについては，一切記述していない点が上げられる。

以上をまとめると，上記に示した

・ﾃﾞｰﾀ関数関連図
・関数ﾌﾟﾛｸﾞﾗﾑ
・ＦＡＣＴﾒｲﾝﾌﾟﾛｸﾞﾗﾑ

の３つのﾌﾟﾛｸﾞﾗﾑのみを記述することにより，ＦＡＣＴ法によるﾌﾟﾛｸﾞﾗﾑは完結する。

b）ＦＡＣＴ法の動作原理

しかし，それではＦＡＣＴ法ではどのようにして，「ﾃﾞｰﾀの首尾一貫性の保証の問題」を解決しているのであろうか。そのために，ＦＡＣＴ法では，ﾃﾞｰﾀ変更により生じたﾃﾞｰﾀ構造上の矛盾を解消するという意味での，「矛盾解消機構」と呼ぶ独特の動作原理を導入することにより，この問題を解決している。以下に，「矛盾解消機構」の動作原理を，上記適用例を踏まえて，具体的に説明する。

まず最初に，ａ）に示したＦＡＣＴ法の記述において重要な点としては，図８のＦＡＣＴﾒｲﾝﾌﾟﾛｸﾞﾗﾑから図７の関数ﾌﾟﾛｸﾞﾗﾑが一切呼ばれていない点である。なぜ，そのようなﾌﾟﾛｸﾞﾗﾑが可能かと言えば，ＦＡＣＴ法では，図６のﾃﾞｰﾀ関数関連図が，重要なﾌﾟﾛｸﾞﾗﾑの一部となっているからである。

具体的に、データ関数関連図の働きを示せば，以下の通りとなる。

ＦＡＣＴ法ではﾃﾞｰﾀ関数関連図から、ﾃﾞｰﾀ関数関連定義ﾌｧｲﾙ（図９）を自動的に生成する。このﾌｧｲﾙの作り方は、ﾃﾞｰﾀ関数関連図から関数と入出力ﾃﾞｰﾀの組合せを順次拾いだし記入すれば一義的に作成可能である。組合せの記入の順序は自由であり、要はﾃﾞｰﾀ関数関連図で表現されている組合せがすべて整っていればよいことになる。

```
      REAL R, H, A, V
      CALL FILOPN
   10 WRITE(6, 1010)
 1010 FORMAT(1H , '*****************'/
     ¥       1H , '*DATA INPUT  (1)*'/
     ¥       1H , '*DATA OUTPUT(2)*'/
     ¥       1H , '*SYUURYOU    (0)*'/
     ¥       1H , '*****************')
      READ(5, *) IOP
      IF(IOP.EQ.0) THEN
          FILCLS
          STOP
      ELSEIF(IOP.EQ.1) THEN
          WRITE(6, 1020)
 1020   FORMAT(1H , '**********************'/
     ¥         1H , '*TEIMEN NO HANKEI (1)*'/
     ¥         1H , '*ENCHUU NO TAKASA (2)*'/
     ¥         1H , '**********************')
          READ(5, *) JOP
          IF(JOP.EQ.1) THEN
              READ(5, *) R
              CALL DBWRIT('R', R)
              GO TO 10
          ELSEIF(JOP.EQ.2) THEN
              READ(5, *) H
              CALL DBWRIT('H', H)
              GO TO 10
          ELSE
              GO TO 10
          ENDIF
      ELSEIF(IOP.EQ.2) THEN
          WRITE(6, 1030)
 1030   FORMAT(1H , '**************************'/
     ¥         1H , '*ENCHUU NO TAISEKI   (1)*'/
     ¥         1H , '*ENCHUU NO TEIMENSEKI(2)*'/
     ¥         1H , '**************************')
          READ(5, *) KOP
          IF(KOP.EQ.1) THEN
              CALL DBREAD('V', V)
              WRITE(6, *) '**V=', V
              GO TO 10
          ELSEIF(KOP.EQ.2) THEN
              CALL DBREAD('A', A)
              WRITE(6, *) '**A=', A
              GO TO 10
          ELSE
              GO TO 10
          ENDIF
      ELSE
          GO TO 10
      ENDIF
      END
```

図８）FACTﾒｲﾝﾌﾟﾛｸﾞﾗﾑ

次に，ＦＡＣＴ法ではﾃﾞｰﾀ関数関連定義ﾌｧｲﾙに基づいて，関数起動ﾙｰﾁﾝ（図１０）と呼ぶｿｰｽｺｰﾄﾞが自動生成される。この中では、MENSEK, TAISEKが共にCALLされている。ここに，関数起動ﾙｰﾁﾝFNSETの引数ＩＲは、ﾃﾞｰﾀ関数関連定義ﾌｧｲﾙで定義された関数と入出力ﾃﾞｰﾀの組合せの通し番号である。しかし、この関数起動ﾙｰﾁﾝFNSETもまた、図７，図８の各ﾌﾟﾛｸﾞﾗﾑからは、CALLされていない。従って、この問題を解決する為には、もう１つの仕掛けが必要である。

```
MENSEK    R              A
TAISEK    A              V
          H
```

図９）ﾃﾞｰﾀ関数関連定義ﾌｧｲﾙ

```
      SUBROUTINE FNSET(IR)
      CHARACTER*12 INAME(200), ONAME(200)
      IF(IR.EQ.1) THEN
          INAME(1)='R'
          ONAME(1)='A'
          CALL MENSEK(INAME, 1, ONAME, 1)
      ELSEIF(IR.EQ.2) THEN
          INAME(1)='A'
          INAME(2)='H'
          ONAME(1)='V'
          CALL TAISEK(INAME, 2, ONAME, 1)
      ELSE
          STOP '**ERROR**'
      ENDIF
      RETURN
      END
```

図１０）関数起動ﾙｰﾁﾝ

ﾃﾞｰﾀ関数関連定義ファイルからは、もう一つ、「ﾈｯﾄﾜｰｸ構造定義ﾌｧｲﾙ」が自動生成される。「ﾈｯﾄﾜｰｸ構造定義ﾌｧｲﾙ」は「ﾃﾞｰﾀ関数関連図」をｺﾝﾋﾟｭｰﾀがより扱い易い形に変換したﾌｧｲﾙであり，本質的には「ﾃﾞｰﾀ関数関連図」と等価なものであるが，このﾌｧｲﾙを利用する事により上記の問題はすべて解決される事にな

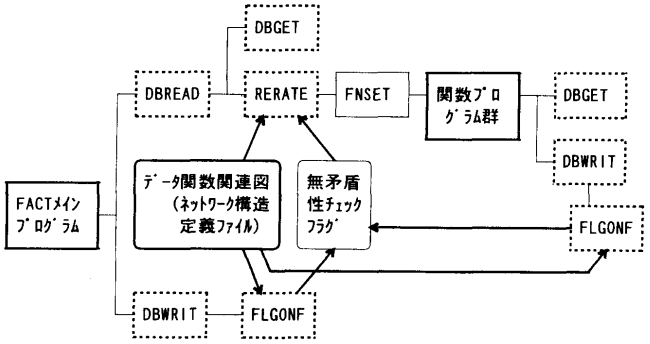

図１１）矛盾解消機構

る。その最後の仕掛けを図１１に示す。

図１１で，太い点線で表現されているのが「FACT-LIB」と呼ばれる，ﾌﾟﾛｸﾞﾗﾑﾗｲﾌﾞﾗﾘであり，ＦＡＣＴ法独特の矛盾解消機構を動作させるための仕掛けである。

「FACT-LIB」は，以下のﾙｰﾁﾝ群より構成される。

①DBWRIT：ﾃﾞｰﾀをﾃﾞｰﾀﾍﾞｰｽﾌｧｲﾙに書き込むと同時に，FLGONFを起動する。

②FLGONF：ﾈｯﾄﾜｰｸ構造定義ﾌｧｲﾙを参照し，当該ﾃﾞｰﾀを入力とするﾙｰﾁﾝの出力側のﾃﾞｰﾀの無矛盾性ﾁｪｯｸﾌﾗｸﾞを'1'にすると同時に，当該ﾃﾞｰﾀのフラグを'0'にする。

③DBREAD：ﾒｲﾝﾌﾟﾛｸﾞﾗﾑにおいて，ﾃﾞｰﾀのﾃﾞｰﾀﾍﾞｰｽﾌｧｲﾙからの読み込みを行う。内部的にはｻﾌﾞﾙｰﾁﾝRERATEとDBGETがCALLされる。

④RERATE：ﾈｯﾄﾜｰｸ構造定義ﾌｧｲﾙ及び無矛盾性ﾁｪｯｸﾌﾗｸﾞを参照し，入力側ﾃﾞｰﾀの一貫性がとれている関数を検出し順次起動をかける。

⑤DBGET：ﾃﾞｰﾀﾌｧｲﾙからﾃﾞｰﾀを入力するﾌﾟﾛｸﾞﾗﾑ。関数ﾌﾟﾛｸﾞﾗﾑ内からのﾃﾞｰﾀの入力の場合には，当該ﾃﾞｰﾀの一貫性は必ずとれていることになるので，このﾙｰﾁﾝを使用する。

ここに，無矛盾性ﾁｪｯｸﾌﾗｸﾞとは，各ﾃﾞｰﾀが下位のﾃﾞｰﾀとの一貫性がとれているか否かのﾌﾗｸﾞである。矛盾なしを(0)とし，矛盾有りを(1)とする。

以上，ﾈｯﾄﾜｰｸ構造定義ﾌｧｲﾙと無矛盾性ﾁｪｯｸﾌﾗｸﾞを介したFLGONFとRERATEのやりとりにより，利用者の意図に基づく多様な入出力操作に呼応した，無駄のない関数の起動を実現する方式のことを，矛盾解消機構と呼ぶ。そして，ａ）に示した簡潔なﾌﾟﾛｸﾞﾗﾑ表現とｂ）に示した矛盾解消機構の導入により，ﾃﾞｰﾀ間の関係がﾌﾟﾛｸﾞﾗﾑにより規定されるような，複雑な「ﾃﾞｰﾀの首尾一貫性の保証の問題」を極めて簡明な方法で解決しているのが，ＦＡＣＴ法の最大の特徴と見ることができるのである。

４．ＦＡＣＴ法の振動応答解析分野への適用

高度な耐震性能要求を実現するためには，様々な構造ｼｽﾃﾑが考案されることが有効と考えられる。その際，考案された構造ｼｽﾃﾑの，妥当性の評価検証を行うためには，様々なモデルに迅速かつ柔軟に対応可能なｺﾝﾋﾟｭｰﾀ支援の方法が準備されていることが，是非とも必要と考えられる。それでここでは，そのような特殊なﾓﾃﾞﾙ化の一例として，「ＦＡＣＴ法」を，筆者らが最近開発した制震装置の評価検証に適用した場合の事例を示す。

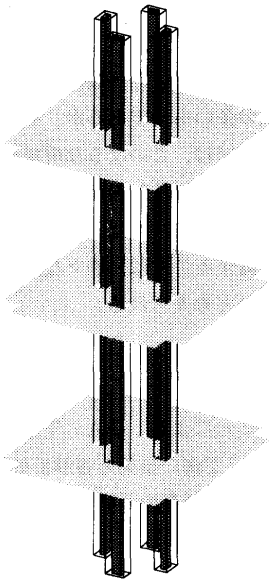

図１２）制震構造ｼｽﾃﾑ概要

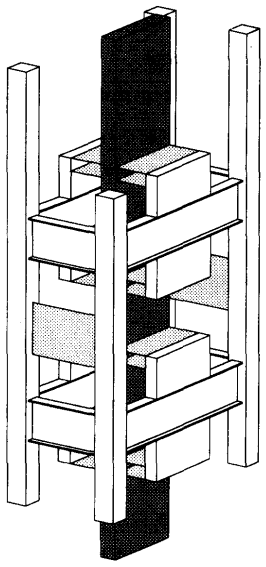

図１３）芯柱型制震装置概要

ａ）制震装置の概要

ここに事例として示す制震装置は，図１２のようなものである。即ち，建物全層にわたり，４本の柱とそれを繋ぐ大梁で構成される，貫通ｼｬﾌﾄを設け，ｼｬﾌﾄ内にそれぞれ１本の曲げ柱（芯柱と呼ぶ）を貫通させる。個々の芯柱においては，図１３に見られるように，芯柱を挟んで各階大梁上下の水平面内に設置した，鋼材ﾀﾞﾝﾊﾟｰﾊﾟﾈﾙで結合し，周辺ﾌﾚｰﾑと芯柱間に生じる相対変位に対しﾀﾞﾝﾊﾟﾊﾟﾈﾙをせん断降伏させることによりｴﾈﾙｷﾞｰ吸収させる制震装置である。

ｂ）質点モデルによる振動応答解析の特徴

超高層建物の設計等で多様される質点ﾓﾃﾞﾙによる振動応答解析の処理の流れは，概ね以下の４つに区分される。

①部材要素の復元力から，質点系の復元力を算出する部分

②質点系の復元力から，数値積分法を適用することにより，時刻歴の質点変位を求める部分

③時刻歴の質点変位から，部材要素の応力を算出する部分

④部材の応力と変形から，部材要素の復元力を求める部分

この内，①③は，振動解析ﾓﾃﾞﾙの形状により，ﾌﾟﾛｸﾞﾗﾑは大きく変わる。一方②④はﾓﾃﾞﾙの形状が変わっても，ﾌﾟﾛｸﾞﾗﾑは殆ど変わらない，定型的な特徴を持つ部分である。従って，ﾌﾟﾛｸﾞﾗﾑが大きく変わることが予想される①③に対して，ＦＡＣＴ法を適用することにより，ﾌﾟﾛｸﾞﾗﾑ作成の効率化を図ることが，特に有効と考えられる。

ｃ）ＦＡＣＴ法の適用

ここでは，ａで示した制震装置の性状を評価するためのﾌﾟﾛｸﾞﾗﾑを，通常のせん断型多質点系のﾌﾟﾛｸﾞﾗﾑを再利用，拡張して作成する場合について述べる。但し，ﾌﾟﾛｸﾞﾗﾑの拡張箇所を全て示すことは紙面上の制約があるので，ここでは特に上記ｂの①の部分の表記がどのように行われるかについて詳述する。

(1)解析ﾓﾃﾞﾙ及び入力ﾃﾞｰﾀ　芯柱ﾀﾞﾝﾊﾟｰの性状を的確に把握する為には，芯柱と周辺ﾌﾚｰﾑの間に生じる相対変形を正確に評価することが重要である。そのために，図１４に示すような解析ﾓﾃﾞﾙを想定する。また，追加が必要な入力ﾃﾞｰﾀを，図１５に示す。

(2)ﾌﾟﾛｸﾞﾗﾑの実際　ＦＡＣＴ法のﾌﾟﾛｸﾞﾗﾑ記述の最も特徴的な部分である，ﾃﾞｰﾀ関数関連図を図１６に示す。また，図に示された各ﾃﾞ

-タ，関数の内容を図１７に示す。図１８に今回作成した関数（ｻﾌﾞﾌﾟﾛｸﾞﾗﾑ）の内，最もｽﾃｯﾌﾟ数の多いものを一例として示す。

(3)有効性の検討　図１８に示したｻﾌﾞﾌﾟﾛｸﾞﾗﾑは今回機能拡張のために作成したものの中でも最もｽﾃｯﾌﾟ数の多いものであるが，それでも２０ｽﾃｯﾌﾟ程度である。今回の機能拡張では，このような規模のｻﾌﾞﾌﾟﾛｸﾞﾗﾑを１０個作成し，１つの既存のｻﾌﾞﾌﾟﾛｸﾞﾗﾑを修正したが，それに必要な手間は，通常のﾌﾟﾛｸﾞﾗﾑ作成と比較すれば，微々たるものである。このようなﾌﾟﾛｸﾞﾗﾑ方式を準備することは，様々な構造ｼｽﾃﾑの開発とその性能検証のために，有効と考えられる。

5．まとめ

以上，極めて簡単なﾌﾟﾛｸﾞﾗﾑ例を示すことにより，ＦＡＣＴ法の記述形式と動作原理を説明し，併せてＦＡＣＴ法の振動応答解析分野への適用例を示すことにより，ＦＡＣＴ法の既存のﾌﾟﾛｸﾞﾗﾑ方式に対する優位性を明らかにした。

ＦＡＣＴ法の優位性をまとめると，以下の３点が指摘できる。

まず第１点は，３章で示したような「ﾃﾞｰﾀの首尾一貫性の保証」の問題を極めて簡単な約束により解決している点である。もちろん関係する全ての処理をﾃﾞｰﾀの変更の履歴とは無関係に全て実行しなおすことを前提とすれば，このようなことは問題とはならない。しかし，個々のｻﾌﾞﾌﾟﾛｸﾞﾗﾑの処理に時間を要しｻﾌﾞﾌﾟﾛｸﾞﾗﾑ毎の部分実行を行いたい場合には，この「ﾃﾞｰﾀの首尾一貫性の保証」は大きな問題となる。そして，建築構造設計支援ﾌﾟﾛｸﾞﾗﾑは，この種の制約を持ったﾌﾟﾛｸﾞﾗﾑに相当する。この問題を極めて簡単な約束により解決した点は，ＦＡＣＴ法の大きな優位性と見ることができる。

第２点としては，「ﾃﾞｰﾀの首尾一貫性の保証」の問題を解決するにあたり，そのﾃﾞｰﾀ間の首尾一貫性を保証する「関係」を定義するための表現形式として，簡明さと宣言性に特徴をもつ「ﾃﾞｰﾀ関数関連図」と呼ぶ図式表現を採用した点が上げられる。この表現形式は，ﾌﾟﾛｸﾞﾗﾑの利用者や維持管理担当者に対し，ﾌﾟﾛｸﾞﾗﾑの内容を理解するための負荷を軽減し，多くの利便性を提供することになる。

第３点としては，第２点で指摘した簡明で宣言的な表記法に対応して，宣言的表現のままで手続き的表現に変換することなく，ﾌﾟﾛｸﾞﾗﾑ実行を可能とする「矛盾解消機構」と呼ぶ独特の動作原理を導入した点が上げられる。「ﾃﾞｰﾀ関数関連図」と「矛盾解消機構」を導入することにより，表記と動作の両面において統一的な概念に基づくﾌﾟﾛｸﾞﾗﾐﾝｸﾞを実現した点が，従来のﾌﾟﾛｸﾞﾗﾑ開発手法と本質的に異なるＦＡＣＴ法の特徴である。

参考文献

1)稲田：ｺﾝﾋﾟｭｰﾀによる建築構造設計支援のあり方に関する研究，「（その１）ＦＡＣＴ法の提案」，1996.12，日本建築学会技術報告集第３号
2)原，稲田，小川：性能型構造設計法についての考察，「その５：ｺﾝﾋﾟｭｰﾀｼﾐｭﾚｰｼｮﾝによる設計検証」，1997年9月，日本建築学会大会学術講演梗概集
3)小川，稲田，大垣，金井，川村，：性能型構造設計法についての考察「その７：損傷分散型ﾀﾞﾝﾊﾟｰを設置したＳ造超高層建物の設計」，1997年9月，日本建築学会大会学術講演梗概集

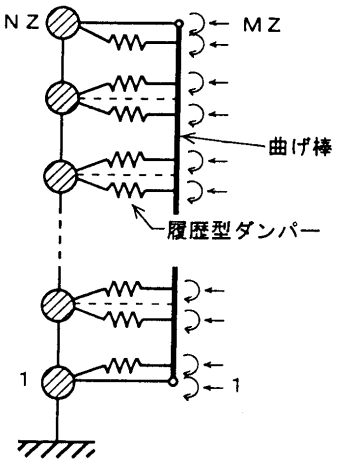

図１４）振動解析モデル

MZ：芯柱接点数
AIS：芯柱の断面２次ﾓｰﾒﾝﾄ
ALS：芯柱のﾌﾞﾛｯｸ長
DIS：履歴ﾀﾞﾝﾊﾟｰの剛性
QIS：履歴ﾀﾞﾝﾊﾟｰの降伏強度
IS：履歴ﾀﾞﾝﾊﾟｰの接続質点番号

図１５）入力データ一覧

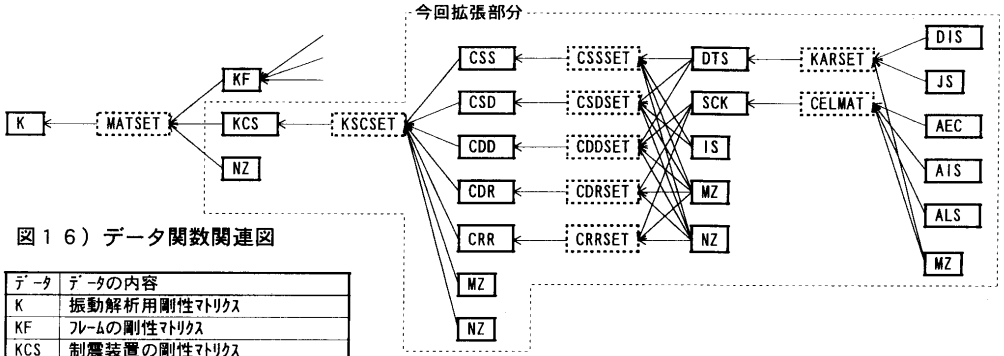

図１６）データ関数関連図

ﾃﾞｰﾀ	ﾃﾞｰﾀの内容
K	振動解析用剛性ﾏﾄﾘｸｽ
KF	ﾌﾚｰﾑの剛性ﾏﾄﾘｸｽ
KCS	制震装置の剛性ﾏﾄﾘｸｽ
NZ	質点数
MZ	芯柱の節点数
CSS	芯柱の剛性ﾏﾄﾘｸｽの係数 質点変位，質点力の係数
CSD	水平変位，水平力の係数
CDD	水平変位，水平力の係数
CDR	回転変位，水平力の係数
CRR	回転変位，ﾓｰﾒﾝﾄの係数
DTS	ﾀﾞﾝﾊﾟｰ部分の瞬間剛性
SCK	芯柱の部材剛性ﾏﾄﾘｸｽ
IS	ﾀﾞﾝﾊﾟｰの接続質点
DIS	ﾀﾞﾝﾊﾟの剛性ﾃﾞｰﾀ
JS	ﾀﾞﾝﾊﾟの復元力の種別
AEC	芯柱のﾔﾝｸﾞ係数
AIS	芯柱の断面２次ﾓｰﾒﾝﾄ
ALS	芯柱のﾌﾞﾛｯｸ長

ｻﾌﾞﾙｰﾁﾝ	ｻﾌﾞﾙｰﾁﾝの機能
MATSET	剛性ﾏﾄﾘｸｽの合成
KCSSET	制震装置の要素剛性ﾏﾄﾘｸｽの縮合
CSSSET CSDSET CDDSET CDRSET CRRSET	制震装置の要素剛性ﾏﾄﾘｸｽの作成
KARSET	ﾀﾞﾝﾊﾟｰ部分の瞬間剛性のｾｯﾄ
CELMAT	芯柱部材剛性ﾏﾄﾘｸｽの作成

図１７）データ，関数の内容

```
SUBROUTINE KSCSET(INAME,NI,ONAME,NO)
COMMON // AAA(5000),BBB(5000), CSS(5000),CSD(5000),
¥  CDD(5000),CDR(5000),CRR(5000),MZ,NZ,KSC(5000)
CHARACTER*12 INAME(NI),ONAME(NO)
CALL DBGET(INAME(1),CSS)
CALL DBGET(INAME(2),CSD)
CALL DBGET(INAME(3),CDD)
CALL DBGET(INAME(4),CDR)
CALL DBGET(INAME(5),CRR)
CALL DBGET(INAME(6),MZ)
CALL DBGET(INAME(7),NZ)
CALL COPY(CRR,AAA,MS)
CALL SWEEP(AAA,MZ,MZ,MZ)
CALL S003(CDR,AAA,BBB,MZ,MZ,MZ)
CALL TENCHI(CDR,AAA,MZ,MZ)
CALL S003(BBB,AAA,CCC,MZ,MZ,MZ)
CALL S002(CCC,CDD,BBB,MZ,MZ)
CALL SWEEP(BBB,MZ,MZ,MZ)
CALL S003(CSD,BBB,AAA,NZ,MZ,MZ)
CALL TENCHI(CSD,BBB,NZ,MZ)
CALL S003(AAA,BBB,CCC,NZ,MZ,NZ)
CALL S001(CSS,CCC,KSC,NZ,NZ)
CALL DBWRIT(ONAME(1),KSC)
RETURN
END
```

図１８）今回作成した関数（ｻﾌﾞﾌﾟﾛｸﾞﾗﾑ）の例

［1997年６月20日原稿受理　1997年９月１日採用決定］