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

Tatsuo INADA

*1

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 supported by computer, Program development method, Hysteretic damper

1. はじめに

建築構造設計において、コンピ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、コンピ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ピュータプログ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、建築構造設計支援プログラムの要件として、

- ・「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、既存のプログ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)と呼ぶ新しいプログ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プログラムに適用した事例を示すことにより、FACT法を他のプログ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、極めて簡単なプログラム例を対象として考察を進めることにより、既存のプログラム方式が、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) 簡単なプログラム例

ここでは、極めて簡単なプログラム例として、「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」プログラムを記述することを考える。

このプログ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の機能をサブプログラム MENSEK, 第2の機能をサブプログラム 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のようなプログラムが書けるであろう。

しかしこのプログラムは、利用者の意図の所在の如何によっては、きわめて不親切なプログ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.

たプログラムは上記のケース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、もしプログラム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プログ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

とが可能なプログラムとして、図3に示すプログラム2を提示する。プログラム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。

しかし、プログラム2もまた利用者にとって余り親切なプログラムとは言えない一面を持っている。それは、プログラム2の途中で網かけで示した2行であるが、このプログラムではどのような出力を行う場合でも必ずサプログラム 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プログラムを呼ぶ必要の無い場合も少なくない。これは、場合に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、プログラム2は利用者にとって耐え難いプログ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プログラムとしては、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に示すプログラム3を提示する。このプログラム3では、利用者にサプログラムを呼ぶタイミング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、このプログラム3は別の意味で大きな問題を含んでいる。そ

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)
① Rを入力する。
② Aを表示する。
③ Hを入力する。
④ V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)
① Rを入力する。
② Hを入力する。
③ Aを表示する。
④ V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)
① Rを入力する。
② Hを入力する。
③ Vを表示する。
④ A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)
① Rを入力する。
② Hを入力する。
③ Vを表示する。
④ Hを入力する。
⑤ V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)
① Hを入力する。
② Rを入力する。
③ Vを表示する。
④ Rを入力する。
⑤ Vを表示する。

図2

```
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) プログラム1

それは、無駄のないプログラムの起動のタイミングの選定を、プログラム3では利用者の判断に委ねている点である。このことは、このプログ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プログラムの実行のタイミングを意識しておかなければならず、利用者に対しプログラム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、プログラム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) プログラム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、無駄のないプログラムの起動のタイミング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プログ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プログ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プログ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プログラム起動の手順を選択する問題を、既存の手続き型のプログ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プログ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プログラム起動のタイミングは極めて多岐に渡り、通常の手続き型のプログ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、データ数も多く、データ間相互の関係を規定するプログラム行数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、「データの首尾一貫性の保証の問題」と呼ぶことにしよう。「データの首尾一貫性の保証の問題」がテーマとなるプログラムは、現実世界においては多々存在する。「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された既存のプログラム開発方式として、例えばデータベースマネジメントシステムがある。しかし、データベースマネジメントシステムでは、上記の

ようなデータ相互の関係がプログ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。

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、「デ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。

```

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) プログラム3

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) データ関数関連図の作成

この簡単なプログラム例の問題には、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

このような図式表現のことを、FACT法では「データ関数関連図」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、いわば簡条書きのような手軽さをコンピュータプログ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

に順不同にデータの関係を表記することによりプログラミンが可能となれば、プログラムの作成は飛躍的に効率化するはずである。このようなプログラム方式のことを宣言的プログラム方式と呼ぶが、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、プログ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、建築構造設計支援プログラムのような、極めて複雑な論理構造を持つプログ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。

(2) 関数プログラムの作成

FACT法における個々のサブプログラムのことを「関数プログラム」と呼ぶ。FACT法における関数プログラムの記述形式は、上記の簡単なプログラム例の場合について見れば、図7の通りとなる。

(3) FACTメインプログラムの作成

FACT法でプログラミンを行う場合に最後に必要となる重要なプログラムとして、FACTメインプログラムがある。以下に、上記の簡単なプログラム例の場合のFACTメインプログラムの一例を図8に示す。

ここで、FACTメインプログラムの特徴としては、FACTメインプログラムが表現しているのは、プログラムの利用者の入力操作と出力要求のみであり、通常のプログラム方式では必須と考えられるサブプログラムの起動のタイミングについては、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

- ・データ関数関連図
- ・関数プログラム
- ・FACTメインプログラム

の3つのプログラムのみを表記することにより、FACT法によるプログラムは完結する。

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、「データの首尾一貫性の保証の問題」を解決しているのでしょうか。そのために、FACT法では、データ変更により生じたデ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メインプログラムから図7の関数プログラムが一切呼ばれていない点である。なぜ、そのようなプログラムが可能かと言えば、FACT法では、図6のデータ関数関連図が、重要なプログ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法ではデータ関数関連図から、デ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、データ関数関連図から関数と入出力デ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、要はデ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。

```
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) 関数プログラム

次に、FACT法ではデ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)と呼ぶソースコードが自動生成される。ここでは、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、データ関数関連定義ファイルで定義された関数と入出力デ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の各プログラムからは、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。

デ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「データ関数関連図」をコンピュータがより扱い易い形に変換した

```
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メインプログラム

MENSEK	R	A
TAISEK	A	V
	H	

図9) デ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

ファイルであり、本質的には「データ関数関連図」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、プログラムライブラリであり、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: データをデータ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、当該データを入力とするルーチンの出力側のデータの無矛盾性チェックフラグを'1'にすると同時に、当該データのフラグを'0'にする。
- ③DBREAD: メインプログラムにおいて、データのデータファイルからの読み込みを行う。内部的にはサブルーチン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グを参照し、入力側デ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: データファイルからデータを入力するプログラム。関数プログラム内からのデータの入力の場合には、当該デ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グとは、各データが下位のデータとの一貫性がとれているか否かのフラグである。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グを介した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)に示した簡潔なプログラム表現とb)に示した矛盾解消機構の導入により、データ間の関係がプログラムにより規定されるような、複雑な「デ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ピ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デル化の一例として、「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、鋼材ダンパで結合し、周辺フレームと芯柱間に生じる相対変位に対しダンパをせん断降伏させることによりエネルギ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デ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、振動解析モデルの形状により、プログラムは大きく変わる。一方②④はモデルの形状が変わっても、プログ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、プログ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、プログ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プログラムを、通常のせん断型多質点系のプログラムを再利用、拡張して作成する場合について述べる。但し、プログ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)解析モデル及び入力データ 芯柱ダンパ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デルを想定する。また、追加が必要な入力データを、図15に示す。

(2)プログラムの実際 FACT法のプログラム記述の最も特徴的な部分である、データ関数関連図を図16に示す。また、図に示された各デ

た、関数の内容を図17に示す。図18に今回作成した関数(サブプログラム)の内、最もステップ数の多いものを一例として示す。

(3)有効性の検討 図18に示したサブプログ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プ数の多いものであるが、それでも20ステップ程度である。今回の機能拡張では、このような規模のサブプログラムを10個作成し、1つの既存のサブプログラムを修正したが、それに必要な手間は、通常のプログラム作成と比較すれば、微々たるものである。このようなプログ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、極めて簡単なプログ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法の既存のプログ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「デ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データの変更の履歴とは無関係に全て実行しな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、個々のサブプログラムの処理に時間を要しサブプログラム毎の部分実行を行いたい場合には、この「データの首尾一貫性の保証」は大きな問題となる。そして、建築構造設計支援プログラムは、この種の制約を

持ったプログラムに相当する。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いえることができる。

第2点としては、「データの首尾一貫性の保証」の問題を解決するにあたり、そのデ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「データ関数関連図」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、プログラムの利用者や維持管理担当者に対し、プログラムの内容を理解するための負担を軽減し、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すりゃき表現に変換することなく、プログ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。「データ関数関連図」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概念に基づくプログラミングを実現した点が、従来のプログ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: コンピ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: 損傷分散型ダンパを設置した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: 芯柱のブロッグ長
DIS: 履歴ダンパの剛性
QIS: 履歴ダンパの降伏強度
IS: 履歴ダンパの接続質点番号

図15) 入力データ一覧

図16) データ関数関連図

データ	データの内容
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 質点変位, 質点力の係数
CSD	剛性 水平変位, 質点力の係数
CDD	マトリクス 水平変位, 水平力の係数
CDR	の 回転変位, 水平力の係数
CRR	係数 回転変位, モーメントの係数
DTS	ダンパ部分の瞬間剛性
SCK	芯柱の部材剛性マトリクス
IS	ダンパの接続質点
DIS	ダンパの剛性データ
JS	ダンパの復元力の種別
AEC	芯柱のヤング係数
AIS	芯柱の断面2次モーメント
ALS	芯柱のブロッグ長

サブルーチン	サブルーチンの機能
MATSET	剛性マトリクスの合成
KSCSET	制震装置の要素剛性マトリクスの縮合
CSSSET	制震装置の要素剛性マトリクスの作成
CSDSET	
CDDSET	
CDRSET	
CRRSET	
KARSET	ダンパ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(サブプログラム)の例

[1997年6月20日原稿受理 1997年9月1日採用決定]