超小型電算機による構造解析

OHCHI, Yozo / 大地, 羊三

(出版者 / Publisher)
法政大学工学部
(雑誌名 / Journal or Publication Title)
Bulletin of the Faculty of Engineering, Hosei University / 法政大学工学部研究集報
(巻 / Volume)
15
(開始ページ / Start Page)
139
(終了ページ / End Page)
155
(発行年 / Year)
1979-03
(URL)
https://doi.org/10.15002/00004146

超小型電算機による構造解析

大 地 羊 三*

Structural Analysis by Micro Computer

Yozo Ohchi

Abstract

We cannot help wondering at the recent rapid progress of micro computer techniques, and it is not a dream to have a desk computer with some Mega bits memories. Then we attempted to examine whether a existing micro computer is able to endure or not the large calculation of the structural analysis.

A program named KOHZO which was developed in our loboratory and is put to general use in the computer center of our college, is rewritten and reform as to fit for the conversational type. The micro computer which was introduced into our laboratory has 64 kbits memories and two 80k bits floppy diskets, and it is available that a Extended BASIC language provided with double pressision arithmetics. The results of numerical tests are shown in fig. 14 and 15, and this is not the sufficient ones. The time of calculation is about 90 times over the available FACOM 230/45 S computer in our college and the structure having less than about 180 joints is only able to analysed. However, it seams that this difficulty will be ouercome if we use the latest announced floppy disket which has 4 times memory size and 2 times speed than ours.

§1. 緒　　言

過去数年来，電算機のハードウェアの進歩にはめざましいものがありCPU，メモリー共に極端に小型化され，5～6年前まで大型機と考えられていた電算機と同程度の能力を持つものが机上にのるような時代も夢ではなくなった。価格も100～200万円あれば，現在普通に行われている規模の構造解析が可能な容量を確保できる。一説によれば，2年后には能力は倍増し価格は半減するであらうとも云われている。割引いて考えても，これは容易ならぬ事態である。しかし，ソフトウェアは人間の頭脳を必要とするので，ハードウェアのこのような進歩にはついていけない。この意味で大型機のレベルアップには限度があるように思われる。ハードウェアの能力は2倍，3倍にあげることができても，それを駆使するソフトウェア，とくに万人向けのOSを開発する

* 土木工学科

点で手づまりを生ずるからである。そして今後のハードウェアの技術は小型，超小型の機種に吸収され，数年後には，現在中型機といわれている程度の能力を持った超小型機が机上に乗るようになるであらうし，価格も1人の人件費を出ないようになると考えられる。小型の機械を作ることでは，日本人は世界で最もすぐれている。自動車，ラジオ，時計等を見ればこの事実は理解できるであろう。電算機部門も例外ではなく，世界で最も精巧な超小型電算機は，日本でできるものと確信している。この場合，ソフトウェアも日本人の頭脳によることが望ましい。

私共の研程室では，約3年の開発期間を経て，研究用を目的とする構造解析のプログラム「KOHZO」を完成し，昭和48年から実用に供している[3)]。これは一つのサブルーチンを追加するだけで新しい要素を含む構造物の解析を可能にするものであり，数時間の指導で学生にも利用できるように考案してある。現在世間で使用されている構造解析用のプログラムは，STRESS, STRUDL, NASTRAN, ASKA 等数が多い。また，多くの企業でそれぞれ独自のプログラムを開発している。しかしこれらはいずれも大型機のゆきとどいたI/Oと大容量の記憶装置に支えられたものである。さらにその内容を分析してみると，プログラムの全体が有機的に結合していて，使用者側ではその内容を変更したり新しい機能をつけることが困難なものと，サブルーチン群だけが用意されていて，使用者側でコントロール・プログラムを作らないと作動しないものの両極端がある。これらに対して，私共が開発したシステム「KOHZO」は，サブルーチンを並列にならべている点では後者に似ているが，別に全体をコントロールするプログラムが用意されていて，内容をくわしく知らなくても計算ができるし，新しいサブルーチンを追加することも容易である。しかし，私共のプログラムは開発費用が桁ちがいに少ないので，入出力に対して複雑なサービスはできない。研究者にとってはこれで十分であろう。

また，世間で使用されているシステムは，デイスクまたは磁気テープベースであり，データは一度外部記憶装置に入れてから計算を実行するようになっているし，メモリーもかなり贅沢に使っている。超大型の構造解析を対象にしているのであるから当然かも知れないが，このために比較的小さな構造物の計算速度が犠牲にされている。システム「KOHZO」では，コア内で処理できる計算には外部記憶装置を使わないようにし，コア内で処理できないときに初めて外部記憶装置を使用することにしてある。しかも，メモリーは出来るだけ節約するように設計されているため，かなり大きな構造物までコア内で処理できるし，計算速度も早い。たとえば，本四公団の三経間連続トラスをコア内で処理したこともあるし，1500 m 級の吊橋でも節点を 50 m 間隔にすればコア内で処理できる。また，コア内におさまる構造物であれば，計算速度も IBM 1130 で比較して STRESS の3倍，UNIVAC 1106 で比較して NASTRAN の2倍強の実積を持っている。超小型電算機が普及すると，構造解析のプログラムを利用する技術者の数が急増するであろう。このようになるとプログラムの使い易さや拡張の可能性が問われるであろうし，電算機が小型であればあるほどメモリーの節約と速度の向上が要求されるであろう。

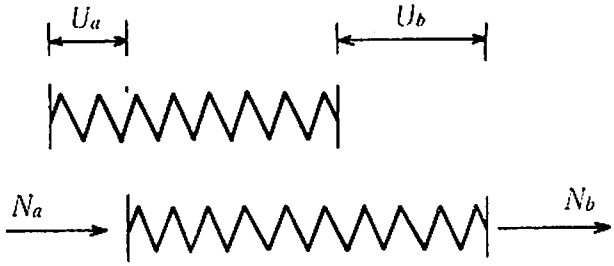
そこで昭和53年度の科学研究費補助金を受けて，マイクロコンピュータ（超小型電算機）を購入し，上記のシステム「KOHZO」を対話型に変更して数値実験を行ったので報告する。使用した言語はBACICである。一部には，FORTRANのコンパイラーがマイクロ・コンピュータで稼動するまで待った方がよいという姿勢もあるが，FORTRANは大型機を対象にし，バッチ処理を目的とした言語であり，対話形式を主体としたマイクロ・コンピュータに適した言語だとは考えられない。

§2. 理論の概要

システム「KOHZO」で採用している計算手法は，一般に変形法または有限要素法とよばれているものである。本論文は，これらの理論の内容について議論しようとするものではないので，次節以下の説明を理解して頂くために必要最小限の解説にとどめる。我々があつかっている構造物は，力を受けると変形する弾性体である。しかも，構造物の寸法に比べて変形は微小である，という大前提があるので，作用外力と変形は比例関係にある。

$N = k\delta$

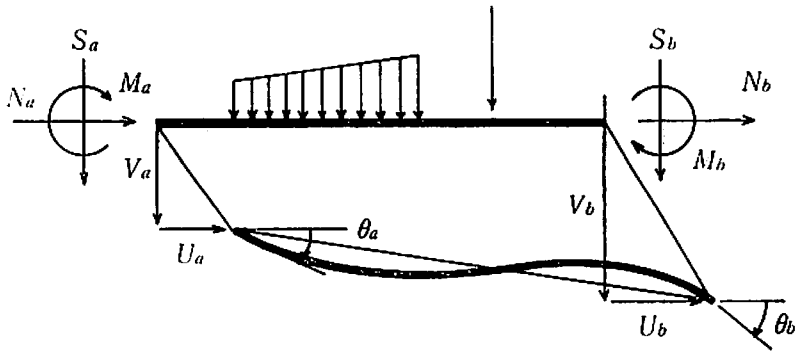
図1



$N_a = ku_a - ku_b$

$N_b = -ku_a + ku_b$

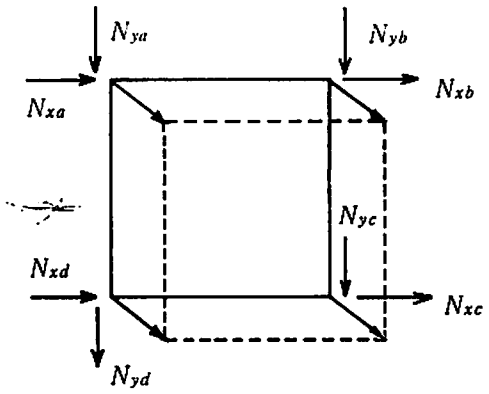
図2



$f_a = k_{aa}u_a + k_{ab}u_b + f_{fa}$

$f_b = k_{ba}u_a + k_{bb}u_b + f_{fb}$

図3



$$f_a = k_{aa}u_a + k_{ab}u_b + k_{ac}u_c + k_{ad}u_d + f_{fa}$$
$$f_b = k_{ba}u_a + k_{bb}u_b + k_{bc}u_c + k_{bd}u_d + f_{fb}$$
$$f_c = k_{ca}u_a + k_{cb}u_b + k_{cc}u_c + k_{cd}u_d + f_{fc}$$
$$f_d = k_{da}u_a + k_{db}u_b + k_{dc}u_c + k_{dd}u_d + f_{fd}$$

図4

最も簡単な構造要素であるばねを例にとって説明しよう。図1に示すように，ばねは力（N）で引張るとδだけ伸びるが，Nとδは比例関係にあり図の下に書いた式が成立つ。ここで比例定数kは，ばね定数とよばれている量である。この式のδを両端の変位u_bとu_aの差で表わし，左側の力の正の方向を逆方向と定義しなおすと，図1の式は図2のように書きかえられる。これが変形法や有限要素法で使われている基本式の最も簡単な形である。

無論，我々が取扱っている構造要素は，図1，2に示すように単純なものばかりではない。図3に示すはり要素の場合は，一点に作用する力は3箇の成分（N，S，M）を持つし，これに伴って変位も3成分（u，v，θ）現れる。また，図4に示す板要素の場合は，力の作用点が4箇あるいはそれ以上となることもある。さらに要素の中間に外力が作用することもある。したが

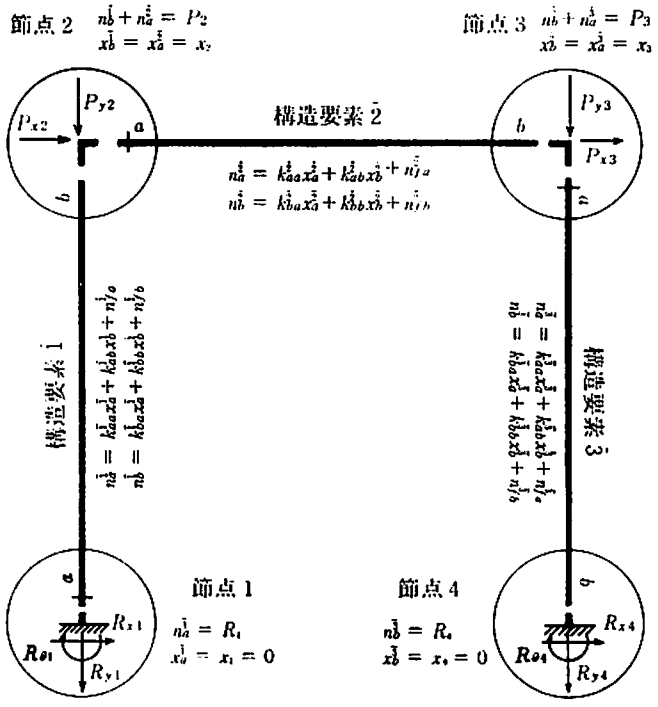


図5

って力および変位は，それぞれの要素の自由度に応じた数の成分を持つベクトルとして表現しなければならない。しかし，この場合でも力ベクトルと変位ベクトルとの間には比例関係があり，それぞれの図に示すような基本式が得られる。なお，それぞれの基本式の最後の項（f_{fa}，f_{fb}…等）は要素の中間に使用する外力に関係するものである。これらの式の具体的な内容については参考文献を参照されたい。

全体の解式を作るには図5のようにすればよい。すなわち，与えられた構造物を図2～3で説明した基本式が作れる程度に分割し，それぞれについて基本式を作ったのち節点でつなぎ合せる。つなぎ合せには節点の釣合条件式（各節点でそこに集まる要素端の力が釣合っていなければならないという条件式）と適合条件式（各節点でそこに集る要素端の変位は節点の変位に等しくなければならないという条件式）が用いられる。さきに用意された各構造要素の基本式を節点の釣合条件式に代入し，適合条件を考慮すると節点の数×自由度の次数を持つ連立一次方程式が得られる。図5の場合は次のようになるであろう。

$$\begin{pmatrix} k^{\mathrm{I}}_{aa} & k^{\mathrm{I}}_{ab} & 0 & 0 \\ k^{\mathrm{I}}_{ba} & (k^{\mathrm{I}}_{bb}+k^{\mathrm{II}}_{aa}) & k^{\mathrm{II}}_{ab} & 0 \\ 0 & k^{\mathrm{II}}_{ba} & (k^{\mathrm{II}}_{bb}+k^{\mathrm{III}}_{aa}) & k^{\mathrm{III}}_{ab} \\ 0 & 0 & k^{\mathrm{III}}_{ba} & k^{\mathrm{III}}_{bb} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} - \begin{pmatrix} n^{\mathrm{I}}_{fa} + 0 \\ n^{\mathrm{II}}_{fa} + n^{\mathrm{I}}_{fb} \\ n^{\mathrm{III}}_{fa} + n^{\mathrm{II}}_{fb} \\ 0 + n^{\mathrm{III}}_{fb} \end{pmatrix} \tag{1}$$

なお，図5で基本式の力と変位（fとu）をnとxにと書きかえてあるのは，節点で釣合条件式や適合条件式を作るときに都合のよいように，各要点の力のベクトル（f）および変位ベクトル（u）を座標変換してあるためである。この座標変換によってばね定数のマトリックス（k_{aa}，k_{ab}…等。剛性マトリックスとよんでいる）の内容も変ることは当然であるが，簡単のために同じ記号を使うことにした。式(1)は構造物全体を一種のばねとみなしたとき，節点に作用する力とその点の変位の比例関係を表わすものとみることができる。この連立一次方程式の係数行列（一種のバネ定数マトリックス）のことを構造物全体の剛性行列（ステイフネス・マトリックス）とよんでいる。

式(1)を解けば各節点の変位（x）が求まるし，これを要素の基本式にもどしてやれば各要素に作用する力が求められるわけであるが，式(1)はこのままで解くことができない。左辺の係数行列のデターミナントは0である。これは支点の情報がまだ入っていないため，構造物全体の剛体変形がゆるされているからである。また右辺は反力が含まれている。この困難を取り除くには，支点では変位が0であるという条件を使って解式を変形し，右辺に反力を含んでいる式を取り除く必要がある。この操作を支点の処理とよんでいるが，図5の場合は次のようになる。

$$\begin{pmatrix} (k^{\mathrm{I}}_{bb}+k^{\mathrm{II}}_{aa}) & k^{\mathrm{II}}_{ab} \\ k^{\mathrm{II}}_{ba} & (k^{\mathrm{II}}_{bb}+k^{\mathrm{III}}_{aa}) \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} P_2 \\ P_3 \end{pmatrix} - \begin{pmatrix} n^{\mathrm{II}}_{fa}+n^{\mathrm{I}}_{fb} \\ n^{\mathrm{III}}_{fa}+n^{\mathrm{II}}_{fb} \end{pmatrix} \tag{2}$$

連立一次方程式を解く方法は色々あるが，構造解析の場合は三角化法がよい。構造解析では一

種類の荷重だけで計算をすませることはほとんどない。荷重を色々変えて，それらに対する構造物の応答を調べなければならないことが多い。いいかえると，右辺を色々変えて連立一次方程式を解かなければならないということである。この種の計算に最も適した解法は三角化法であろう。さらに大型構造物の場合は，コア内で連立一次方程式を一度に解くことができなくなり，計算の途中で外部記憶装置に掃出さなければならないことがしばしばある。これに適した方法として，システム「KOHZO」ではウエイブ・フロント法が採用されている。

以上の説明で構造解析のためのプログラムの流れについて予想がついたと思われるが，ここでまとめておこう。

(1)　要素毎の基本式を作るために必要なデータの入力

(2)　全体の剛性行列の作製

(3)　支点に関する情報を用いて支点処理の実行

(4)　剛性行列の三角化

(5)　荷重データの入力

(6)　前進代入，後退代入の手法で連立一次方程式の解を求める。

(7)　要素毎の基本式を用いて力を計算し，結果を出力する。

(8)　必要があれば(5)にもどって計算を繰返す。

以上の各項目はそれぞれサブルーチンにまとめてあるが，細部については4節で説明する。

§3. 計算手法の改良

前節の最後で説明した流れにそってプログラムを作れば，一応の構造解析はできるはずである。事実，世間で一般に使われているシステムはこの流れにそっており，更にそれぞれの特徴を生かすような機能が追加されている。そしてこの機能の追加は電算機のメモリーを増大させる方向でなされるのが常である。しかし，本研究の目的は超小型電算機で稼動するシステムを作ることであるから，世間の方向とは逆に，メモリーを出来るだけ節約しながら機能を向上させるものでなければならない。この目的のために，計算手法にいくつかの改良を加えた。以下にその主なものについて概要を説明する。

(1)　全体の剛性行列を収容する配列の節約

この手法は，本学で稼動しているシステム「KOHZO」でも採用している[3)]が，今回開発したシステムの中核をなすものであるから概略の説明をしておく。前節の説明からも解るように，全体の剛性行列の中は構造要素の自由度と同じ次数の小行列毎にまとまっている（式(1)参照）。しかも，構造物が大型になればなるほど非対角小行列全体の中で零行列でないものの割合が少くなる傾向がある。剛性行列のこの性質を利用して，非対角要素は小行列毎にまとめて零行列でないも

$$\begin{pmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{pmatrix} \rightarrow \begin{pmatrix} E & 0 & 0 & 0 \\ l_{21} & E & 0 & 0 \\ l_{31} & l_{32} & E & 0 \\ l_{41} & l_{42} & l_{43} & E \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & r_{22} & r_{23} & r_{24} \\ 0 & 0 & r_{33} & r_{34} \\ 0 & 0 & 0 & r_{44} \end{pmatrix}$$

（対係行列）　　$(l_{ji}=r^T{}_j r_{ii}{}^{-1})$

図6

のだけをメモリーに格納することにした。そして各行は，対角小行列を基点とするチェインで結んで前後関係を取るようにしてある。このようにするためにはポインターが必要であるが，小行列毎に1箇のメモリー増ですむのでその数はたかが知れている。また，剛性行列の三角化，前進代入，後退代入も行列の要素単位ではなく，小行列単位で行っている。この場合の計算公式は次のようにすればよい（図6参照）。

$$\begin{aligned} &\text{三角化} \quad r_{ij}=k_{ij}-\sum_{k<i,\,j} r^T_{ki} r_{kk}{}^{-1} r_{kj} \quad (i=1\sim n,\ j=i\sim n) \\ &\text{前進代入} \quad y_i=P_i-\sum_{k<i} r^T_{ki} r_{kk}{}^{-1} y_k \quad (i=1\sim n) \\ &\text{後退代入} \quad x_i=r_{ii}{}^{-1}(y_i-\sum_{k>i} r_{ik} x_k) \quad (i=n\sim 1) \end{aligned} \tag{3}$$

非対角小行列の格納法については，別の角度から4節で説明しなおす。

(2)　支点の処理と不連続点の処理

前節では，構造要素はすべて節点で連続であると仮定した。しかし，この条件を満足しない状態で結合される構造要素もある。たとえば，図7(a)の3節点では左右の構造要素のたわみ角（回転）が連続ではない。このような不連続端を持つ構造要素に対しては，一般には，その構造要素だけに適用できる基本式を別に用意して節点で結合する方法がとられているが，これでは考えられるすべての不連続端を持つ構造要素について別々に基本式を用意しそれに見合うプログラムを作らなければならない。この欠点を取り除くために，不連続点を図7(b)のように処理することにした。すなわち，不連続点と結合している構造要素は，一旦不連点で切断し，残さなければならない連続条件は節点変位間の制約条件として別に考慮しようとするわけである。この種の処理が

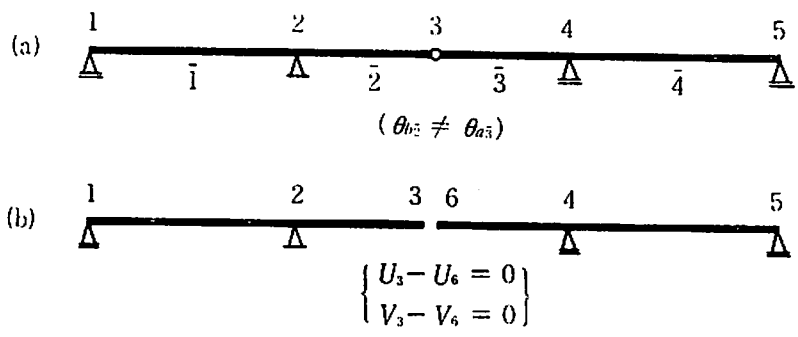


図7

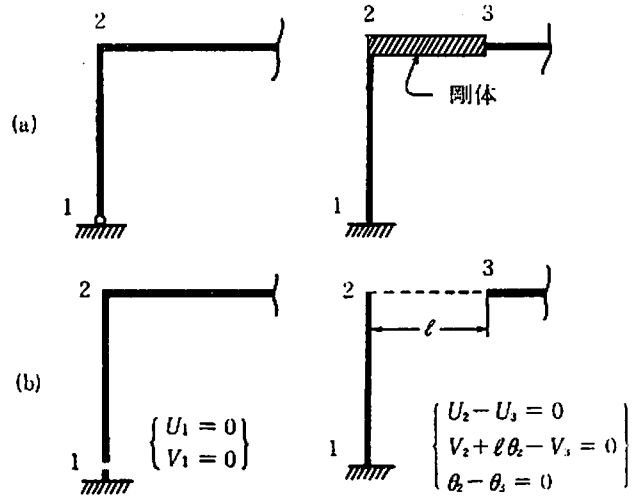


図8

可能であれば，図8(a)に示す支点や剛体をはさむ構造要素も同図(b)の制約条件で対処できる。

構造解析の問題は，マクロに見ると与えられた構造物のポテンシャル・エネルギーを最小にするように節点変位をきめる問題であると解釈できる。節点変位に制約条件がついていれば，その制約条件をつけたポテンシャル・エネルギーを最小にすればよい。節点に作用する荷重のベクトルをP，節点変位のベクトルをU，節点変位間につけられた制約条件を$bU=0$とすると，制約条件をつけたポテンシャル・エネルギーを最小にする問題は次のように書ける。

$$\frac{1}{2}U^TKU-U^TP+\mu^TbU\longrightarrow min \tag{4}$$

ここで，Kは制約条件をつけない構造系の剛性行列，μはラグランジュ乗数を要素とするベクトルである。上式の最初の2項が制約条件のない場合のポテンシャルエネルギーであることは，この2項をUで微分して0とおけば制約条件をつけない構造系の解式が得られることで理解できるであろう。式(4)をU，μで微分して0とおくと，次式が得られる。

$$\left.\begin{aligned} KU+b^T\mu&=P\\ bU\qquad\quad&=0\end{aligned}\right\}\ \text{または}\ \begin{bmatrix}K & b^T\\ b & 0\end{bmatrix}\begin{bmatrix}U\\ \mu\end{bmatrix}=\begin{bmatrix}P\\ 0\end{bmatrix} \tag{5}$$

上式はこのままでは使いにくい。そこで，右側の式の係数行列に含まれている零行列の対角要素を無限小に相当する数$-\varepsilon$でおきかえてからμを消却すると，次に示す制約条件を含めた構造系の解式が得られる。

$$(K+b^T\varepsilon^{-1}b)U=P \tag{6}$$

ここでε^{-1}は無限小の逆数すなわち無限大に相当する数を対角線に並べた対角行列である。式(6)の意味するところは，制約条件をつけない構造系の剛性行列Kを$b^T\varepsilon^{-1}b$で修正すれば，制約条件をつけた構造系の剛性行列が得られるということである。入力データで制約条件を与えてやれば，簡単なプログラムでこの修正はできる。ただし，無限大に相当する数の与え方に問題が残る。余りに大きくすると，Kの要素が桁落ちして良い結果が得られない。この問題に対する対策

も講じてあるが長くなるので省略する。参考文献[5]を参照されたい。

(3) 影響線の処理

構造解析においては，色々な荷重に対してその応答を計算しなければならないことは前節で説明した。この問題に対する対策として，単位荷重を色々な位置に移動させたときの応答を前もって計算しておき，荷重の形がきめられたとき，先に計算しておいた単位荷重に対する応答を係数倍しながら重ね合せて，所期の荷重に対する応答を求めようとする方法がある。この単位荷重を移動させながら求めた応答を図示したものが影響線と呼ばれるものである。影響線を求める操作は，数値解析的にみると，解式の右辺すなわち荷重項を〔1，0，…0〕，〔0，1，0…0〕，……と変えながら解くことと同等であるから，剛性行列の逆行列を求めればよいことになる。しかし，逆行列を求める手法をそのままの形で使うと大容量のメモリーが必要になって，現在の目的にそわない。また，一般にはミューラー・ブルスラウの原理を用いることもよく行なわれているが，この手法を用いると一本の影響線を計算する毎に，構造系を改造しなければならないので，多数の影響線が必要なときには計算時間に問題が出る。私共の研究室では，数年前に，構造系を改造しないで一組の荷重系から影響線を計算する手法を開発した。[4] くわしい説明は本論文の目的からそれるので省略するが，この一組の荷重系は図２～４で説明した基本式の係数から作られるものである。今回作製したシステムでは数箇所で構造要素の基本式を参照しているので，その係数を取り出すことは容易な仕事であるが，まだ自動的に計算するプログラムは用意していない。現在の所は，手で計算して荷重データとして入力してやらなければならないが，構造解析を多小理解している者であれば造作なく計算できるはずである。

(4) 分布荷重の処理

図３～４に示した式の右辺の最後の項（f_f）が，分布荷重をこれと等価な節点荷重に変換したベクトルである。任意の分布荷重を対象にすると，この項の計算には数値積分が必要になる。これは厄介なことなので，一般には部材の中間にも多数の節点をもうけて分布荷重を扱わないような方策が取られている。しかし，本システムではメモリーと計算時間を節約するために，節点の間隔はできるだけ長くし，積極的に分布荷重を取り入れる必要がある。特別な形の分布荷重（区間等分布荷重，三角荷重，区間集中荷重等）については，表にまとめて参考文献[1]にのせてあるので参照されたい。将来は，本システムとは別に，任意の分布荷重と等価な節点荷重を計算するプログラムを作成するつもりでいる。

§4. システムの概要

今回開発したシステムは，仮に「BKOHZO」と名付けることにした。これで本学で稼動して

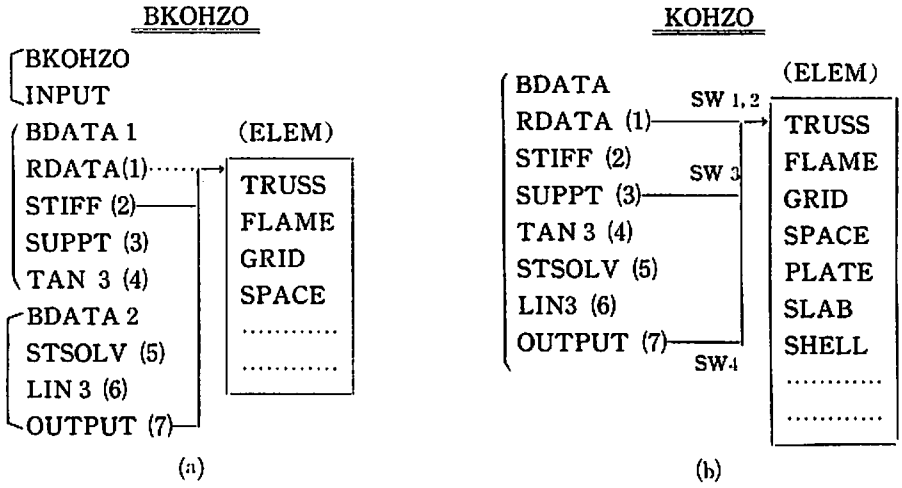


図9

いるシステム「KOHZO」と区別し，BASIC言語で書かれた構造解析用のプログラムという意味を持たせたつもりである。

図9(a)に「BKOHZO」のサブプログラム，(b)に「KOHZO」のサブプログラムがあげてあるが，両者の差はBDATAをBDATA 1とBDATA 2に分けた点とサブプログラムBKOHZOとINPUTが追加された点である。なおサブプログラム名のあとにかっこをつけて書いてある番号は，2節の最後で説明した計算の流れにつけた番号と対応させてある。これでそれぞれのサブプログラムの内容がうかがわれるであろう。構造物が大きくなると，それをコア内で一度に処理することが困難になる。こうような場合には，構造物をいくつかのブロックに分割し，ブロック単位でデータをコア内に読込みながら計算を進めなければならない。システム「KOHZO」では，このブロック間の継ぎ目を処理するサブプログラムとしてBDATAが用意されている。また，我々が取り扱う構造要素には色々な種類があり，それぞれ異った基本式を用意しなければならない。(ELEM) と書いた枠の中にあるサブルーチン群がそれぞれ一つの構造要素の基本式を計算するルーチンであって，これらを必要とするサブプログラムからスイッチをつけて呼ばれるようにしてある。したがって，新らしい構造要素を追加したければ，その構造要素の基本式を計算するサブルーチンを作って枠内に登録するだけで計算ができる仕組となっている。

バッチ処理で大型電算機を使う場合であればこれで十分である。カードにデータをパンチして電算機に投入してやれば，計算結果がラインプリンターに印刷されるであろう。しかし超小型電算機の場合はこれではすまされない。計算時間が桁ちがいに長いので，一組の構造物の計算を初めから終りまで一貫して実行するよりも，全体を数段階に分けて間をおいて計算する方が都合がよいこともあるし，実際的でもあろう。そこでシステムを次の3段階に分けて計算できるように変更した。

(1) 入力データの作成 (INPUT)

(2) データの読込みから三角化まで（BDATA 1～TAN 3）

(3) 荷重項のセットから結果の印刷まで（BDATA 2～OUTPUT）

そして，各段階の結果はフロッピー・デイスク内に用意したファイルに格納し，次の段階で利用できるようにしてある。超小型電算機の鍵盤からデータを入力する場合，文字の打ちちがい，行の削除・追加，順序の入れ替え等色々な事態の起ることが予想される。このためにテキスト・エデイタに似た機能を持つサブプログラムINPUTを用意した。また，剛性行列の作製から三角実化までと，その後の前進代入，後退代入の計算は完全に分けることのできるものであり，この事は構造物をブロックに分割した場合でもかわりがない。そこでブロック間の継ぎ目を処理するサブプログラムBDATAを三角化までとその後の処理に分離し，それぞれBDATA 1とBDATA 2とした。したがって，入力データを作製したあとで電算機からはなれ，数日後に三角化までの計算をさせることもできるし，結果がファイルに保存されている限り，１ケ月後でも２ケ月後でも荷重項を変えて(3)だけの計算をさせることも可能である。

上記のどの段階の計算をするかといった指示は，サブプログラムBKOHZOで行うようにしてある。このサブプログラムは計算の最初に読み込まれるものであるが，この中では，使用する構造要素の指示，計算の種類の指示（データの入力か，(2)段階の計算か，(3)段階の計算か，あるいは両者を通した計算かといった指示），計算結果を出力する装置の指示（デイスプレイに出力するのか，プリンターか，あるはフロッピー・ディスク内のファイルに格納するのかといた指示），チェック・プリントの指示等が対話型式で行なわれるようになっている。これらの指示をすませたあとは，新らしいデータ（主として荷重データ）の入力が必要とならない限り，自動的に計算を進める。ただし，計算の途中結果をデイスプレイに出力するように指示してある場合は，その指示にしたがってデイスプレイに出力した直後に鍵盤からの入力まちになることは当然である。

次にメモリーの使用法であるが，BASIC言語では，変数名はアルファベット一文字か，これに１～９の数字をつけたものしかゆるされない。したがって，計算の流れを制御するスイッチ類，各種デイメンジョンおよび各種ポインター等は，その意味が解る程度の変数名で表わすわけにはいかない。これがBASICの欠点の一つ（？）であると云われている。この困難は，次に示す５種類の配列を定義することによってさけることにした。FORTRANのラベルド・コモンの内容を配列要素にしたと見ると理解し易いであろう。

A(20)：計算の流れを制御するスイッチ類と全体に共通な定数

B(20)：構造形や荷重データを定義するために必要な各種デイメンジョン。

C(20)：配列DとS＃の内容を区分するために必要なポインター類。

D(約350)：入力データを格納する配列。一部作業領域に使っている。

S＃(約1000)：全体の剛性行列と荷重ベクトルを格納する配列。

最後の配列S＃の＃記号は，著者が使用したBASIC言語に独自のもので，倍精度の定数，変

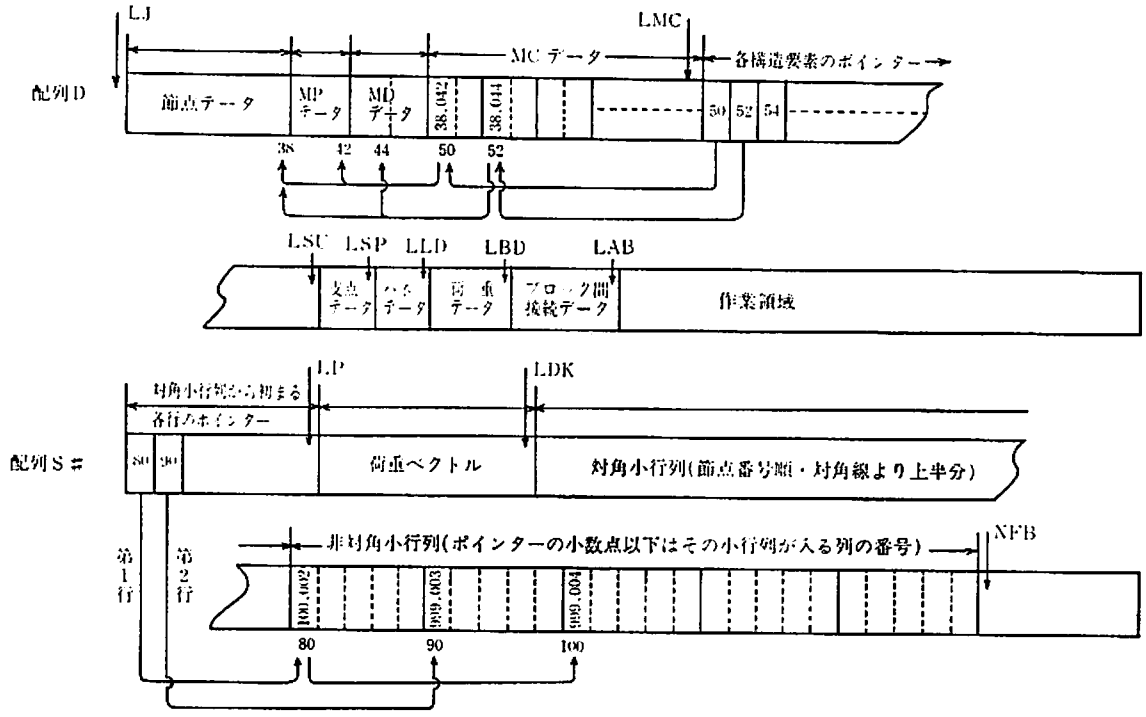


図 10

数および配列を指定するものである。配列A，B，Cのくわしい内容は附表1に示しておいた。また配列D，S# の内容は図10に示すようになっている。図の上側に書いてあるLJ，LMC，…，LP，…等は，配列Cに含まれるポインターと対応するものであり，各ポインターにはそれぞれのデータの先頭番地より一つ若い番地が格納されている。なお，メモリーを圧縮するため，配列Dの中で構造要素のデータはかなり複雑な形で格納されいる。すなわち，LMCに続く番地には各構造要素データの先頭番地をさすポインターが入っており，さらに各構造要素データの先頭番地にはそれぞれの物理定数（MPデータ）および断面定数（MDデータ）が格納されれている番地（くわしくはその一つ前）をさすポインターがおさめられている。また，配列S# の最初の部分には，3節(1)で説明した対角小行列から初まる各行のポインターの初期値が入っており，図の矢印で示す位置にそれぞれの行の非対角小行列が，次に続く小行列の位置を示すポインターを先頭にして格納されている。なおポインターの小数点以下はその非対角小行列が入る列の番号を示すものである。

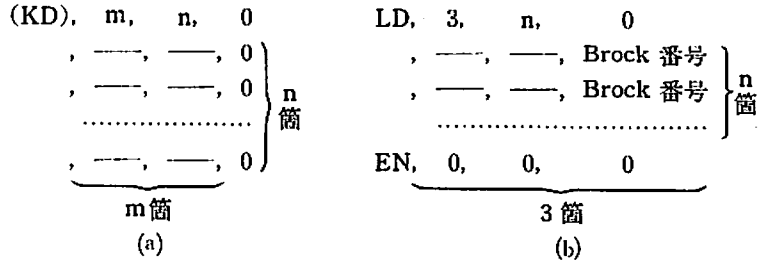


図 11

最後に入力データについて一言しておこう。今後改良の余地が十分にあると考えているが，現時点でサブプログラムINPUTから入力するデータはすべて図11(a)の形に統一してある。ただし，荷重データだけは複数個あることが予想されるので図11(b)の形で次々と追加することをゆるしている。また，すべての荷重データが処理されたあとで，手動で荷重データを入力することもできるように配慮してある。入力データの種類およびその内容は，附表2を参表されたい。

§5. 数値実験の結果

本研究で使用したマイクロ・コンピュータは，64Kバイトのコア・メモリーに鍵盤，ディスプレイ，40桁のジャーナル・プリンタおよび80Kバイトのフロッピー・ディスク2台をつけたものであり，CPUはザイログ社のZ80である。マイクロ・コンピュータでは1バイトを8ビットで構成しているので，大型機と比較するときはメモリーを1/2して見なければならないが，それでも著者が10数年前に使用していた電算機より容量が大きい。この程度の電算機が机上に乗る時代が来ている。使用した言語は，倍精度演算ができるExtended BASICである。図12にこの言語で計算したときの計算速度をあげておいた。これは1000回の繰返計算の平均である。倍精度の割算が目立っておそいのが気になるが，手元において使える気安さを考えれば我慢できな

Extended BASICによる計算時間（単位 *ms*）

計　算　式	加算(+)	減算(−)	乗算(*)	割算(／)	代入(=)	備　考
A=1111.OP.1234	7	7	8	10	5	単精度
A=S.OP.R	7	7	9	10	6	〃
A=S(I).OP.R(I)	11	11	12	13	6	〃
A#=1111#.OP1234#	7	7	13	22	6	倍精度
A#=S#.OP.R#	7	7	15	33	6	〃
A#=S#(I).OP.R#(I)	11	11	19	37	7	〃

図12

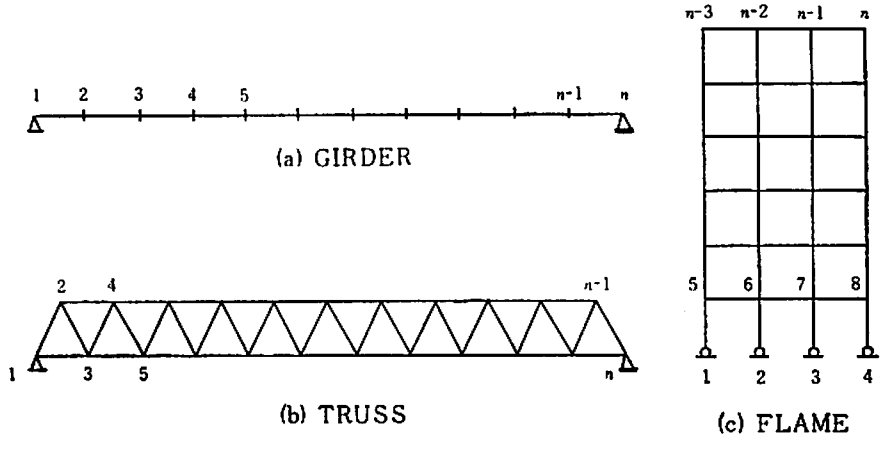


図13

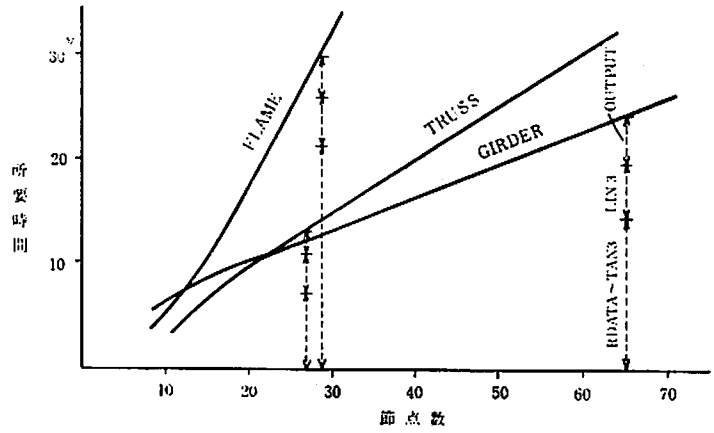


図14

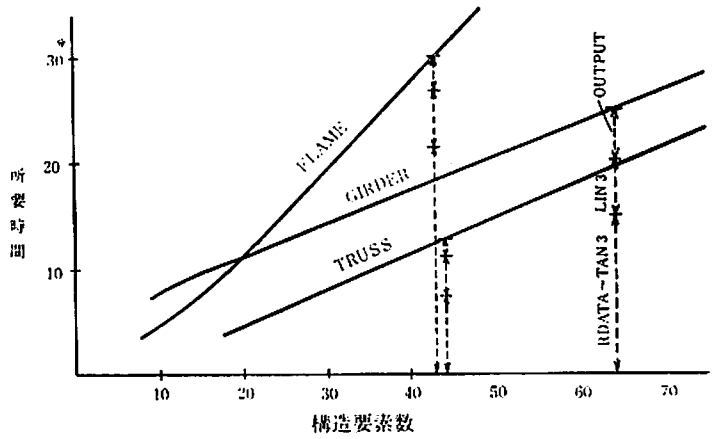


図15

い値ではない。近日中に速度を倍以上にあげる計画もあるとのことである。

今回の数値実験に使ったモデルは，図13に示す3種類である。構造要素の数が50~60程度であれば，ブロックに分割しないでも計算できるが，今回は10数節点毎にブロックに分割して数値実験を行った。計算時間の中にしめるディスク・オペレーションの割合がかなり大きいので，ブロックに分割しても，分割しない場合の2倍，3倍の時間がかかることはないと判断したからである。実験の結果は図14のようになった。この図は横軸を節点数，縦軸を所要時間に取って図示したものであるが，図13の3種類のモデルの間にかなり異った傾向が見られる。これは横軸に節点数を取ったためである。構造解析のかなりの部分の計算は，構造要素の数に比例しているので図15のように横軸に構造要素の数を取る方がよい。しかし，構造物の規模を比較する場合に，一般には節点数を用いることがよく行なわれているので，図14もそれなりに意味がある。なお，図中の縦の点線は，全体剛性行列の三角化までと前進代入・後退代入および計算結合の印刷に要する時間をそれぞれ分けて示したものである。FLAMEの場合は，全体剛性行列が複雑なので，これを三角化するために要する時間も他に比べて長くなっている。

一方，本学にあるシステム「KOHZO」で，図13に示すモデルと同じ構造物を計算させてみ

た。計算速度は，本研究の約90倍早いことがわかったが，著者のテストによると，超大型機は別にして世間で一般に使われている電算機は，平均すると本学の電算機より約3割早いように思われるので，大まかにいって，マイクロ・コンピュータで計算すると世間一般で使われている電算機による計算の120倍の時間がかかることになる。マイクロ・コンピュタの速度向上も著しいものがあり，昭和53年10月には，著者が使用したマイクロ・コンピュータよりも，メモリーで4倍強，速度で2倍強の機種がより安い価格で発買され初めた。この事実を考慮すると，世間一般で行なわれている計算の1秒がマイクロ・コンピュータの1分に相当するとみても大過ないように思われる。

§6. 結　　言

本研究は，本学で稼動しているシステム「KOHZO」を，マイクロ・コンピュータに向くように改造して数値実験を行ったものである。改造の要点は，本文を参照されたい。使用した言語は，倍精度演算が可能なExtended BASICである。倍精度演算ができなければ，構造解析は不可能であろう。著者が使用したマイクロ・コンピュータ（64Kバイトのメモリー＋80Kバイトのブロッヒーディスク2台・COUはZ80）では，本学の電算機に比べて90倍の計算時間がかかるが，これでは多小物足りない。しかし，最近発表された容量で4倍強，速度で2倍強のブロッピーディスクを装備すれば，十分実用にたえると思われる。さらに，BASICで書かれたプログラムを機械語にコンパイルする事も考えられているが，これが可能になれば，コア内の計算時間も1/3～1/4に短縮できるはずである。

なお，本論文は文部省科学技術研究補助金の交付を受けて行っている研究の一部であることを付記する。

参考文献

(1) 大地“マトリックス構造解析”　コロナ社（1977）
(2) 大地“有限要素法とその応用”　森北出版（1975）
(3) 大地・山下“マトリックス法による構造解析の汎用プログラムについて”　土木学会年次学術講演会I-（1972）
(4) 山下・久保“変形法プログラムにおける影響線の求め方について”　土木学会年次学術講演会I-26（1975）
(5) 大地“拘束条件の実用的処理”　土木学会年次学術講演会I-20（1978）

附表1　配列A，B，Cの内容

	配列A（20）	配列B（20）	配列C（20）	
1	(IZ)　最大節点数＝999	(NMA) 質量データの数	(LP)　荷重ベクトルのポインター	1
2	(CON 1)　制約条件付計算で使用する定数	(NLD) 荷重データの数	(LDK)　対角小行列のポインター	2
3	(NOC 2)　同上	(NBD) ブロック接続データの数	(NFB)　配列S# の未使用部分のポインター	3
4	(NRP)　制約条件付計算の繰返回数	(M)　節点変位の自由度	(LJ)　節点データのポインター	4
5	(TIME)　計算開始時間	(N)　接続点の数	(LSU)　支点データのポインター	5
6	(KCAL)　計算の種類	(NST) 断面力の数	(LSP)　ばねデータのポインター	6
7	(KEL)　構造要素の種類	(IJC)　サブルーチン(ELEM)内で使用するポインター　MC データのポインター	(LMA)　質量データのポインター	7
8	——	(IJP)　MP データのポインター	(LLD)　荷重データのポインター	8
9	(LR)　入力装置の種類	(IJD)　MD データのポインター	(LBD)　ブロック接続データのポインター	9
10	(LW)　出力装置の種類	——	(LAB)　作業領域のポインター	10
11	(NCHECK)　チェックプリントのスイッチ	(MMD)　M*(M+1)/2	(LMC)　構造要素データのポインター	11
12	——	——	——	12
13	——	(JD)　1組の節点データに含まれるデータ数	——	13
14	——	(NSD)　1組の支点データに含まれるデータの数	——	14
15	——	(NJ)　節点数	——	15
16	(NBR)　最大ブロック数	——	——	16
17	(IBR)　ブロックのカウンター	(NM)　構造要素の数	——	17
18	(MAXNJ)　ブロック間で最大の節点数	(NS)　支点データの数	(LDAEND)　配列Dの未使用部分のポインター	18
19	(ISUB)　計算の進行を制御するスイッチ	(NSPD)　1組のばねデータに含まれるデータ数	(NDA)　配列Dの大きさ	19
20	(IDEND)　入力データの最後のレコード番号	(NSP)　ばねデータの数	(NDK)　配列S# の大きさ	20

註：以上の外に次のスイッチが用意されている。E=（ELEM）へ飛越すときのスイッチ。E1=LIN 3内で繰返計算をするときのスイッチ。E2=配列Dに荷重があるか否かのスイッチ。E3=計算の範囲をきめるスイッチ

附表2　入力データとその内容

(KD)	m	n	データの内容
JT	2～3	節点の数	各節点の座標値
MP	構造要素の種類毎に異る	2～3	構造要素の物理定数
ND	同上	断面の異る構造要素の数	構造要素の断面定数
MC	2又はそれ以上	構造要素の数	MP, MC データのポインター（MP. 0 MCの形）と接続する節点の番号（I. ooJ の形）
SU	1又は3	拘束数	支点の位置と拘束の方向（JT. ooD の形） m＝3 の場合は制約条件（脚註2）
SP	2	ばねの数	ばねの位置と方向（JT. ooD の形）およびばねの大きさ
LD	2	荷重の数	作用位置と方向（JT. ooD の形）および荷重の大きさ
BD	1	次のブロックに接続する節点の数	現在のブロックの節点番号と次のブロックの節点番号（oldJT. newJT の形）
EN	0	0	なし（データの終りを示す）
MA	2～3	節点の数	各節点の質量データ

註1：かっこ内に書かれた小数点付のデータは，小数点以下3桁とする。
註2：データの内容は剛性行列の位置（LI. 00 I, LJ. 00 J の形）とそこに入れる数の大きさ。