STATIK: A Computer Program for Everyday's Structural Engineering Applications

STATIK: Un programme d'ordinateurs pour les problèmes courants de génie des structures

STATIK: Ein Computerprogramm zur täglichen Anwendung im Bauingenieurwesen

E. ANDERHEGGEN

Professor for Applied Computer Science
Swiss Federal Institute of Technology
Zürich, Switzerland

Summary

Within the context of some of the developments in the field of computer applica tions to structural engineering which took place in Switzerland in the last 15 years, the criteria followed for the design of a new general purpose computer program called STATIK are presented. A number of specific questions concer ning man-machine interface problems are raised which may serve as a basis for further more general discussions.

Résumé

On discute les critères retenus pour le développement d'un nuveau programme d'analyse structurale appelé STATIK, en tenant compte des conditions particu- lières qu'on retrouve en Suisse dans le domaine de l'application des ordinateurs au génie civil. Un certain nombre de questions concernant la relation homme- machine peuvent servir de base à des discussions ultérieures.

Zusammenfassung

Es werden die Kriterien geschildert, die zur Entwicklung eines neuen Computer programmes namens STATIK geführt haben, unter Berücksichtigung der speziel len Bedingungen, die in der Schweiz auf dem Gebiet der Computeranwendungen im Bauingenieurwesen zu finden sind. Eine Reihe Fragen tauchen auf, die als Basis für allegemeinere Diskussionen dienen können.

1. INTRODUCTION

If one considers the way computer programs for structural engineering applications are written and used the following classification seems reasonable:

a. Programs, generally written by non-professional programmers which are used only by their authors or by few very closed associates of their authors. Such programs are often found in the larger firms owning a computer. In fact, although this looks like a great duplication of efforts, it might well make sense to write programs for more or less personal use as the interface problems we are concerned with in this colloquium can then be easily solved in a local, personal way. As an example it is believed that most optimum design programs used today in structural engineering (and there are not many of them) are of this kind, design being too much influenced by personal taste and habits to be left to some not clearly understood black box program written for general use. But of course not everybody can write his own programs.

b. Programs written by professionsl programmers requiring from the user a high level of specific competence generally not found among practising engineers. Most well-known general purpose finite element programs are of this kind, an extreme example being NASTRAN which was written for use in the aerospace industry by highly professional numerical analysts. The use of such programs for civil engineering applications generally requires the professional services of a specialized software firm acting as an interface between the program and its user.

c. Programs written by professional programmers to be used directly by practising structural engineers for whom numerical analysis is only one, and seldom the most important, aspect of their professional activities.

The present paper is only concerned with this last kind of programs where interface problems between automatic computation and everyday's design work are of greatest concern. The criteria to be followed when developing such programs as well as the problems arising by their use are to be discussed. It is felt, however, that a discussion in very general terms would not make much sense. Too many factors greatly varying from place to place would have to be taken into account. Therefore, only some aspects of the developments which took place in Switzerland in the last 15 years and which are closely related to the activities of the writer shall be discussed. It is hoped that such a "case study Switzerland" will give rise to useful discussions and lead to generally valid conclusions.

The following factors certainly had a great influence on the historical developments of computer applications to structural engineering in Switzerland and indeed it would be interesting to compare and discuss the influence that similar factors had in different countries:

a. Swiss structural engineers are not too often confronted with problems where a sophisticated structural analysis would make much sense: Switzerland has practically no aerospace industry; unusual or unusually large structures are rare; rivers flowing near their sources are narrow and therefore not too difficult to span; since over 400 years no major earthquake has occurred; most large dams were designed and built before the advent of computers. However, there are some large reinforced concrete structures for nuclear power plants being designed today.

b. For many years Switzerland had by far the highest per capita cement comsumption of the world which is a good mesure of overall construction activities. Worth mentioning are the national and cantonal road construction programs with hundreds of individually designed, very slender and elegant cast-in-place reinforced concrete posttensioned bridges.

c. The vast majoritiy of Swiss civil engineers confronted with structural design works in relatively small, privately owned consulting offices. This is also a consequence of the Swiss political system as public works (including national highways) are generally managed by the cantons which tend to prefer local consulting firms.

d. Swiss building codes, at least compared with German, are rather liberal allowing considerable freedom in choosing design methods. No state employee checking each single computation exists as this is the case in Germany with the so-called "Prüfingenieure".

e. Switzerland is a rich country with one of the highest computer hardware density of the world. Access to computer facilities is therefore in general easy. As an example the Swiss Federal Institute of Technology in Zurich with some 7200 mostly undergraduate students has main computer facilities worth over 50'000'000 Sfr. not counting many small computers scattered about the institutes.

f. Four years of undergraduate studies are needed to become a civil engineer. At the Swiss Federal Institute of Technology in Zurich out of 12 mandatory and 4 non-mandatory semester courses on statics and structures only one non-mandatory course in the 7th semester deals with computer methods for structural analysis. Graduate studies leading to anything similar to a master's degree do not exist and only some 5

or 6 students reach each year the Ph. D. degree in civil
engineering.

Within this framework a small group of research workers and
Ph. D. students headed by the author of this paper has been
active since 1962 at the Swiss Federal Institute of Technology
in Zurich in the field of computer applications to structural
engineering. One of our objectives has always been the develop-
ment of computer programs to be used both for teaching purposes
at our school and for practical applications by consulting
structural engineers.

2. BACKGROUND HISTORY: THE PROGRAM STRESS

In early 1964 a magnetic tape with the source code of the pro-
gram STRESS was sent to us from the Massachusetts Institute
of Technology. After two years of efforts a modified version
of the program was installed on the main computer of our school
and a course for practising engineers was announced. We had,
however, so many inscriptions that a second course in 1967 had
to be held bringing the total attendance to nearly 600 engineers,
a very large number for Switzerland. The programm STRESS has been
used ever since for teaching purposes at both Federal Institutes
of Technology in Zurich and Lausanne and found wide acceptance
among Swiss consulting engineers. It certainly contributed very
much to the spread and to the understanding of computer methods
in structural engineering in Switzerland.

The main reason for this success is due to the fact that STRESS
was the first finite element program specially designed for
civil engineering applications with limited but clearly defined
objectives: it can handle only linear elastic frames and trusses
which is what most structural engineers not only need but also
clearly understand; it does not attempt to design anything which,
even today, would be a rather hopeless objective for a general
purpose program; is is easy to use due to its simple problem-
oriented input language which in many cases allows the prepara-
tion of new inputs just by extrapolation from old ones without
having to study each time the user's manual, a big advantage
specially for sporadic users.

The developments which took place at the MIT after STRESS are
well known: the much publicized ICES project failed to attain
many of the extremely ambitious and clearly utopian objectives
that were set and was finally abandoned at the MIT. ICES, how-
ever, specially in Europe, had some offsprings like the "inte-
grated" (whatever that means) systems GENESIS in England, ITS
in Germany or SYSFAP in Belgium and it would be interesting to
hear some comments on such programs from people directly in-
volved in their development and use. The writer is rather skep-
tical toward such integrated systems, unless the word "integrat-
ed" is used in a restrictive sense meaning a series of programs
for a specific application (like highway design) where the out-
put of a program serves as input for its successor and not in

his original ICES sense meaning a number of programs for total-
ly different applications (like survey, structural analysis
and project management) all working on the same data base
describing the object to be designed and built in very general,
application independent terms.

3. STRESS'S SUCCESSOR: THE PROGRAM STATIK

After STRESS our group was involved in basic finite element re-
search which led, of course, to many programs for personal use
but also to two programs for general use: a program called
PLATTE for linear elastic plate bending analysis where, for
the first time, syntax diagrams for input description, as
explained later, were used as well as extensive graphical
output of results (e.g. contour lines of bending moments enve-
lopes) and a program called FLASH for plate bending, plate
stretching and shall analysis. The program FLASH is today re-
gularly used for teaching purposes and has also become quite
popular among Swiss consulting engineers being efficient and
very easy to use. However, the points we want to make in this
paper are probably best understood if we consider the criteria
which led to the latest and largest of our programs: the pro-
gram STATIK.

Although the STRESS program was a success, we were quite un-
happy with it during many years: it was very inefficient
(which many users never noticed as they never tried to analyse
large structures); it has never been completely error-free
mainly due to the unnecessarily complicated internal data
organization making it extremely difficult to find programming
errors; it could only handle prestressing in a very primitive
way; it had no graphical output, no restart capabilities, etc.
Finally, we decided to write a new state-of-the-art program to
be used directly by Swiss consulting engineers assuming from
them very much the same degree of competence as needed for
STRESS and handling the same kind of everyday's relatively
simple design problems. With a total effort of approximately
5 to 6 man-years the program STATIK was then developed accord-
ing to the following criteria.

When using STATIK no difficulty should arise concerning the
mathematical model used, i.e. only those problems are to be
handled where an exact solution for a clear and well understood
approximation of reality is possible. This requirement excludes
all kind of nonlinear, dynamic and two-or threedimensional prob-
lems where the choice of the element mesh has an influence on
results. STATIK can only handle linear elastic framed structures
under statical loads as well as different kinds of cross section
calculations. Much attention was paid to the load case prestress-
ing. We also tried (and it is not yet clear if it is relly be-
ing used) to implement some simple design procedures for pre-
stressed and non prestressed symmetric reinforced concrete cross
sections.

Input preparation should be very easy also for relatively in-
experienced sporadic users who are only supposed to read a
very short user's manual (48 pages including examples and
appendices) once. This is achieved by using so-called syntax
diagrams for defining and describing in a very concise and
clear way the problem-oriented, free format input language
of the program. Figures 1 to 4 show some of these syntax
diagrams. They are easily understood observing a few simple
rules: the sequence of input data is found by following the
arrows; a thick stroke corresponds to the beginning of an
input statement, a triangle to its end; the first letter of
the upper case words has to be punched on cards or typed on
a terminal as it is; lower case words are identifiers refer-
ring to numerical or nonnumerical problem data to be specified;
what is written between brackets can be left out, etc. The
syntax diagram showing the overall structure of the program
consisting of seven different modules ("QUERSCHNITT PROGRAMM"
to "AUSGABE VON EINFLUSSLINIEN") is given in fig. 1. Fig. 2
shows the syntax diagram of one of these modules ("STRUKTU-
RELLE EINGABE") used for specifying the structural input data
of very general three-dimensional space frames with straight
and curved members. Fig. 3 shows the syntax diagram of the
program module "RESULTAT AUSGABE" used for requesting numeri-
cal and graphical output of results. A similar syntax diagram
of the program FLASH for the input of structural data for
very general continuous plane and space structures is given
in Fig. 4. Syntax diagrams have proved to be an extremely use-
ful tool not only for describing input languages (they are also
becoming a standard tool for describing and defining programming
languages, e.g. the syntax of the new ANSI-standard FORTRAN77
is defined by means of similar so-called railroad diagrams),
but also because they immediately show what a program can do.
Today, we consider them an indispensable feature of all pro-
grams written for general use.

Input echo and numerical output of the program STATIK appear
on numbered pages of standard format (A4) with one or two head-
lines at each page which are generally used for the name and
address of the consulting firm using the program. Printer con-
trol characters are not used as they are not understood by most
terminals. Such details are mentionened here because they are
very welcomed by practising engineers and also because they
complicate programming considerably.

The Program STATIK has extensive graphical output capabilities.
In all cases the drawing area is assumed to be 37 x 27 cm which
corresponds to the screen size of the large storage tube Tek-
tronix terminals. In fact, the STATIK program produces a graphic
file which can be viewed directly on such terminals, a postpro-
cessor being necessary to obtain the same drawing in any size
on a plotter or on any other graphic device. Figures 5 to 8
show some of these drawings.

The program STATIK is written in FORTRAN for a large CDC Cyber
Computer with a conversational remote job entry system (not a
time-sharing system). It is specially designed to be used
from remote terminals connected by telephone. This has the
advantage of requiring only minimal fixed hardware investments
from the user: a cheap alphanumeric terminal is all it is
needed at least to start with. Later, a faster printer, a
graphic terminal possibly with a hard-copy unit or a small
plotter can be added to improve speed and user's comfort.

The program STATIK, if requested, automatically saves all
problem data at the end of the last program module executed.
The computations for a specific object can therefore take
place in any number of subsequent jobs being possible to
change problem data at any time and to execute program modules
in any order. Some kind of interactive "computer aided design"
is therefore possible. It should be noted, however, that in-
teraction between the program and its user takes place at
the level of subsequent, often very short jobs, and not at
the level of each single line of input text as this is the
case when using a time-sharing system. In fact, we think
that for most structural engineering applications time-sharing
would be an unnecessary luxury.

In October 1977, a course on the programs STATIK and FLASH
with an attendance of approximately 250 civil engineers was
held. It is too early to know if the success we had with
STRESS can be repeated. Among our students, however, STATIK
has been intensively used since nearly two years for all kinds
of applications (sometimes without any specific theoretical
background knowledge) becoming quite popular indeed.

Of course, as always when a program has been written, there
are some features of STATIK we are not so happy about today.
We also had some critics.

Some think that university employees should stick to research
work instead of writing commercial programs like STATIK. Our
answer is that the efforts leading to programs like STATIK
are indeed to be considered research work, not in the field
of structural analysis of course, but in the field of computer
science. In fact, we did our best to solve the interface prob-
lem between the computer and its users just like conventional
computer scientist do, for different problems and different
users, when they develop, say, a new programming language.
Our goals, however, are only attained when many peoples use
our programs as an instrument for their professional activi-
ties which is only possible in a commercial environment.

It would be very much in line with our purposes to have STATIK
run on relatively small computers as it could then be made
available to many more users (e.g. a PDP 11/60 would certainly
be powerful enough for most applications). We must admit, how-
ever, that we made a mistake experienced programmers should
not make: in order to improve efficiency we introduced many

machine dependent features greatly impairing program portability.
Although STATIK is completely written in FORTRAN it would cer-
tainly be an extremely long and tedious task to develop today
a general, machine independent version of the program.

Graphical output was planned having in mind Tektronix storage
tube terminals combined with hard-copy units as we thought
such terminals will have a great future being relatively cheap
and easy to connect to a large computer using asyncronous low-
spead (300 to 1200 Baud) data transfer. Today, we are not so
sure about this anymore. In fact, inexpensive small plotters
working in the same way already exist and recent advances in
microcomputer technology may soon make graphical refresh ter-
minals with a high level of built-in intelligence easily avail-
able. However, the advent of new graphic terminals will not
really impair the use of the program STATIK being a simple
task to have the program produce a hardware independent in-
stead of a Tektronix-oriented graphical file. Postprocessors
are then used to produce graphical output on different hard-
ware units (this solution was already implemented in one ver-
sion of the program STATIK running in a large computing center
in Zurich).

Quite contrary to many other programs who use English for in-
put and output (e.g. STRESS), STATIK and FLASH use German.
Rather provincial reasons led us to this choice as we really
did not want our programs to be used from people living too
far away from us, whose problems, habits and level of compe-
tence we do not understand too well. We were also somehow
afraid of the maintenance problems arising when too many
versions of the program are running in different places. Unfor-
tunately, Switzerland is a multi-lingual country, and we al-
ready had to hear the complaints of our collegues at the Swiss
Federal Institute of Technology in Lausanne who would be much
happier with French versions of the programs.

4. FURTHER DEVELOPMENTS AND CONCLUSIONS

No further development of our programs FLASH and STATIK are
planned as this would contrast with the objectives we pursued
with them. However, we certainly want to pursue similar objec-
tives in the future for different classes of problems and as-
suming from our users a different level of competence. Graduate
studies leading to something else then a Ph. D. degree shall
be introduced soon at our school which will also have a direct
influence on our activities. In fact, we feel that there is
ample room for computer scientists involved with practical
computer applications to try to span the gap between the real
needs of today's structural engineering and much of the some-
times brilliant but often isolated university research work.

General conclusions shall not be drawn here, the "case history"
presented being rather intended to raise questions than to
anwer them. However, as a kind of summary, some of these ques-
tions shall be given hereinafter.

Does a classification of programs distinguishing programs for
more or less personal use, programs requiring specialists help
and programs to be used directly clarify the situation ? How
should a programmer take into account the way his program will
be used ?

What is the influence of local conditions like those given for
Switzerland on the development of computer applications to
structural engineering in different countries ?

In which circumstances can integrated systems be useful ?

What can be done in order to make the use of a program, i.e. the
preparation of input data and the interpretation of results,
as easy as possible ? In which cases is this of primary impor-
tance ?

How can computer-aided design procedures be helpful in every-
day's structural design work ? What kind of an interaction bet-
ween the engineer responsible for the design and the computer
is needed ? Can automatic optimum design programs be useful
for structural engineering applications ?

What will be the influence of recent advances in computer
technology like the proliferation of minicomputers to be used
both off-line and in-line with a host computer ?

What should civil engineering students learn in order to be
able to use such new instruments properly ?

It is hoped that such questions will help to clarify the inter-
face problems we are concerned with in this colloquium and lead
to useful and more general discussions.

Notice: The User's Manuals of the computerprograms STATIK and
FLASH can be obtained from the Institut für Baustatik
und Konstruktion, ETH-Hönggerberg, 8093 Zürich, Switzer-
land.

PROGRAMM STATIK
GENERELLER ABLAUF

(DATENFILE 'name') (GRAPHISCHES (AUSGABEFILE) 'name')

ERSTE TITELKARTE (60 KOLONNEN WERDEN BERUECKSICHTIGT)

ZWEITE TITELKARTE (60 KOLONNEN WERDEN BERUECKSICHTIGT)

QUERSCHNITTS PROGRAMM

STRUKTURELLE EINGABE $\left[\begin{array}{c}\text{(RAEUMLICHE)}\\ \text{EBENE}\end{array}\right]$ (RAHMEN)

VORSPANNKABEL EINGABE

LASTEN EINGABE

LOESUNG (DES) GLEICHUNGSSYSTEMS

RESULTAT AUSGABE

AUSGABE (VON) EINFLUSSLINIEN

• • • • • siehe detaillierte Eingabeschemas
zu den einzelnen Programmteilen

✷ ENDE

Abkürzung für Integer - Listen:

$\{i\}$ entspricht — i — ($\left[\begin{array}{c}\text{BIS i (SCHRITT n)}\\ \text{SCHRITT n BIS i}\end{array}\right]$)

wobei i = q : Querschnittsnummer   i = v : Vorspannkabelnummer
       = k : Knotennummer   = l : Lastfallnummer
       = s : Stabnummer   = j : Wandelementnummer

Fig. 1: Syntax diagram showing the overall structure of the program
STATIK consisting of seven program modules callable in any
order after specifying problem files and output headlines.
The syntax of integer input lists used in subsequent dia-
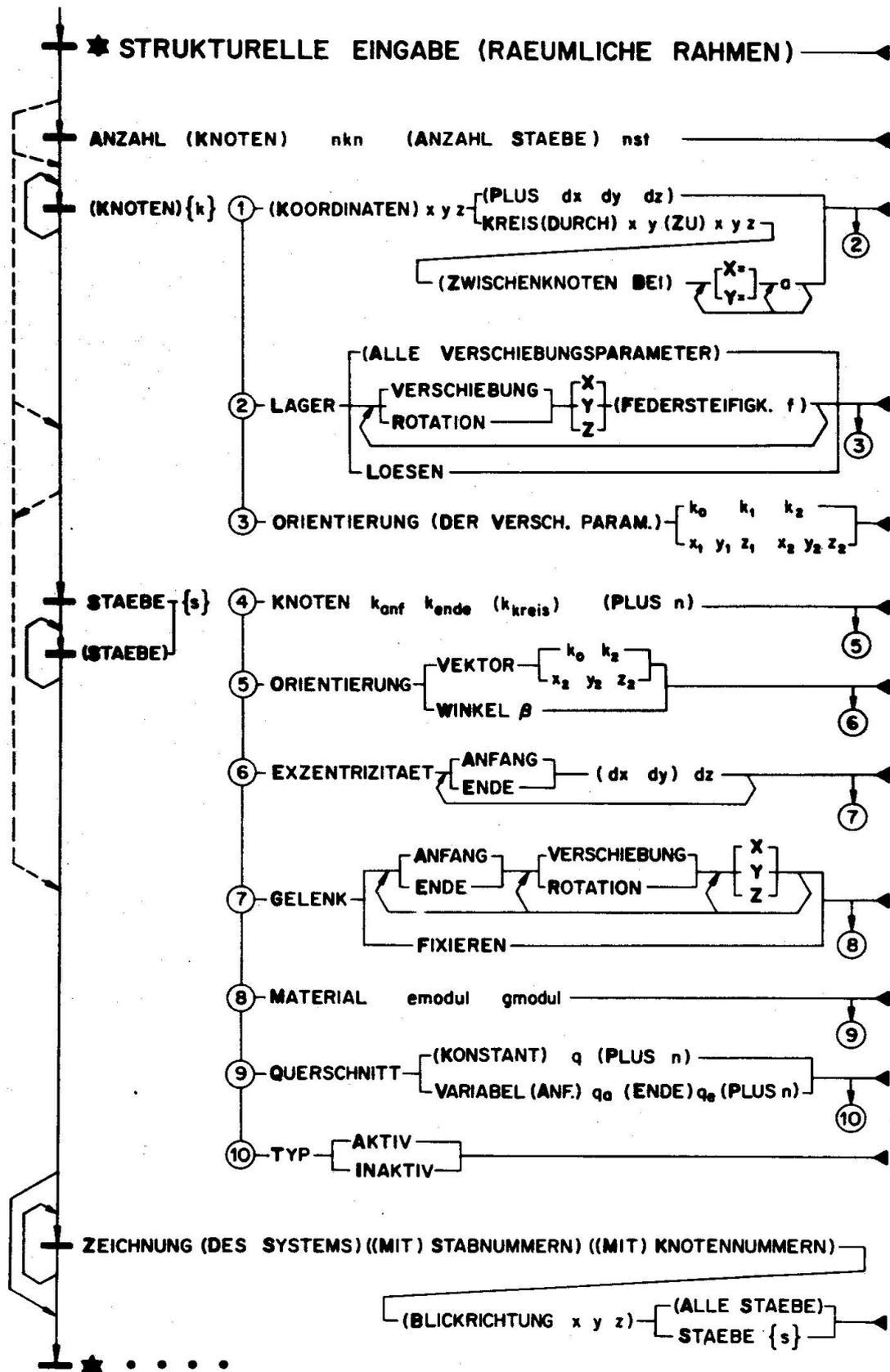grams is also shown.

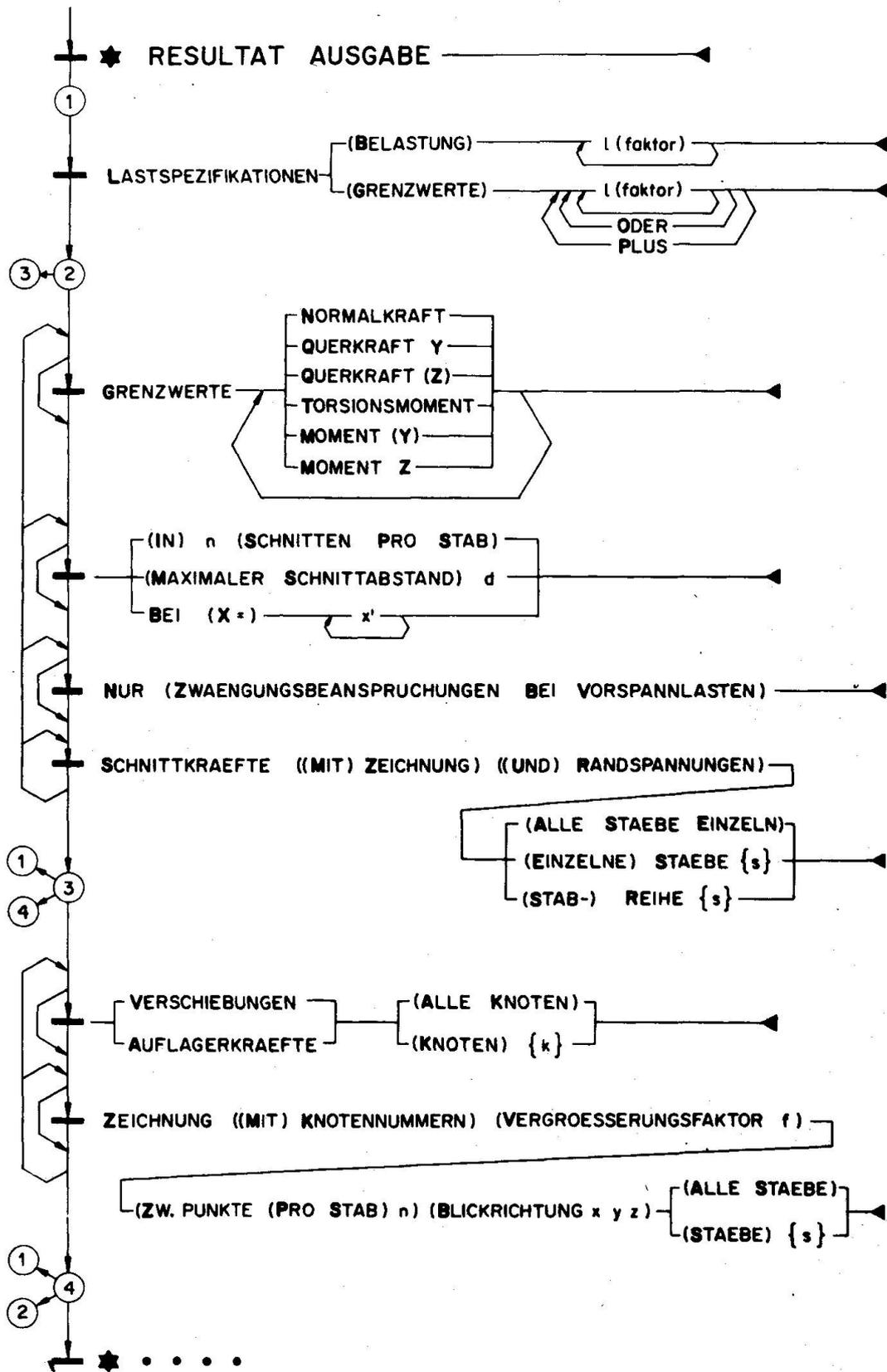Fig. 2: Syntax diagram of the program STATIK for the input of structural data for space frames.

Fig. 3: Syntax diagram of the program STATIK for requesting
numerical and graphical output of results.

EINGABESCHEMA ZUM PROGRAMM FLASH

BEGINN
KONTROLLE
NEUE
ZUSAETZLICHE ──(LASTFAELLE) ──▶ (α)
AUSGABE ──▶ (β)
GRENZWERTE ──▶ (γ)

TITEL DES PROBLEMS (MAX. 80 ZEICHEN)

EBENER (SPANNUNGSZUSTAND)
EBENER VERZERRUNGSZUSTAND
PLATTE ── nkn nel net ((SCHEIBENMODELL) i)
SCHALE

(KNOTEN) k (KOORDINATEN) x (y(z)) (PLUS dx (dy (dz)) (KNOTEN) {k})

BOGEN k_1 $\begin{bmatrix} k_2 \\ -k_2 \end{bmatrix}$ ZENTRUM $\begin{bmatrix} x(y(z)) \\ k_3 \end{bmatrix}$ MIT n (ZWISCHENKNOTEN) {k}

MATRIX m n (KNOTEN) k d_m d_n (KOORDINATEN) x (y (z))

PLUS $dx_m(dy_m(dz_m))$ UND $dx_n(dy_n(dz_n))$ (FAKTOREN f_m UND f_n)

★

(ISOTROP) $\begin{bmatrix} emodul\ v \end{bmatrix}$ dicke

ORTHOTROP $\{s_{11}\ s_{22}\ s_{33}\ s_{12}\}$ $\{p_{11}\ p_{22}\ p_{33}\ p_{44}\ p_{55}\ p_{12}\}$ (FEDERKONSTANTE f)

EXZENTRIZITAET e_z

STAB $\begin{bmatrix} emodul\ gmodul \end{bmatrix}$ fx fy $\{fz\ ix\ iy\}$ iz ─(─$\begin{bmatrix}G\\N\end{bmatrix}\begin{bmatrix}G\\N\end{bmatrix}(\begin{bmatrix}G\\N\end{bmatrix}\begin{bmatrix}G\\N\end{bmatrix}(\begin{bmatrix}G\\N\end{bmatrix}\begin{bmatrix}G\\N\end{bmatrix}))$─) TYP{t}

WIE t'

★

(ELEMENT) e (KNOTEN) k_1 k_2 k_3 (k_4) (TYP t) (PLUS d (ELEMENTE) {e})

STABELEMENT e (KNOTEN) k_1 k_2 k' (TYP t) (PLUS d_1 d_2 (ELEMENTE) {e})

MATRIX m n (EL.) e (KN.) k_1 k_2 k_3 k_4 (TYP t) ─(─$\begin{bmatrix} DIFFERENZ\ d_m\ d_n \\ UNTERTEILUNG \begin{bmatrix}1\\2\end{bmatrix} \end{bmatrix}$─)

★

(TYP) t_a $(t_e (d))$ ELEMENTE {e}

★

$\begin{bmatrix}F\\N\end{bmatrix}\begin{bmatrix}F\\N\end{bmatrix}\begin{bmatrix}F\\N\end{bmatrix}(\begin{bmatrix}F\\N\end{bmatrix}\begin{bmatrix}F\\N\end{bmatrix}\begin{bmatrix}F\\N\end{bmatrix})$ (ROTATION $\begin{bmatrix} ELEMENT\ e \\ k_1\ k_2\ k' \\ x_1\ y_1\ z_1\ x_2\ y_2\ z_2 \end{bmatrix}$)─(KNOTEN) {k}
c_1 c_2 c_3 $\{c_4\ c_5\ c_6\}$

★

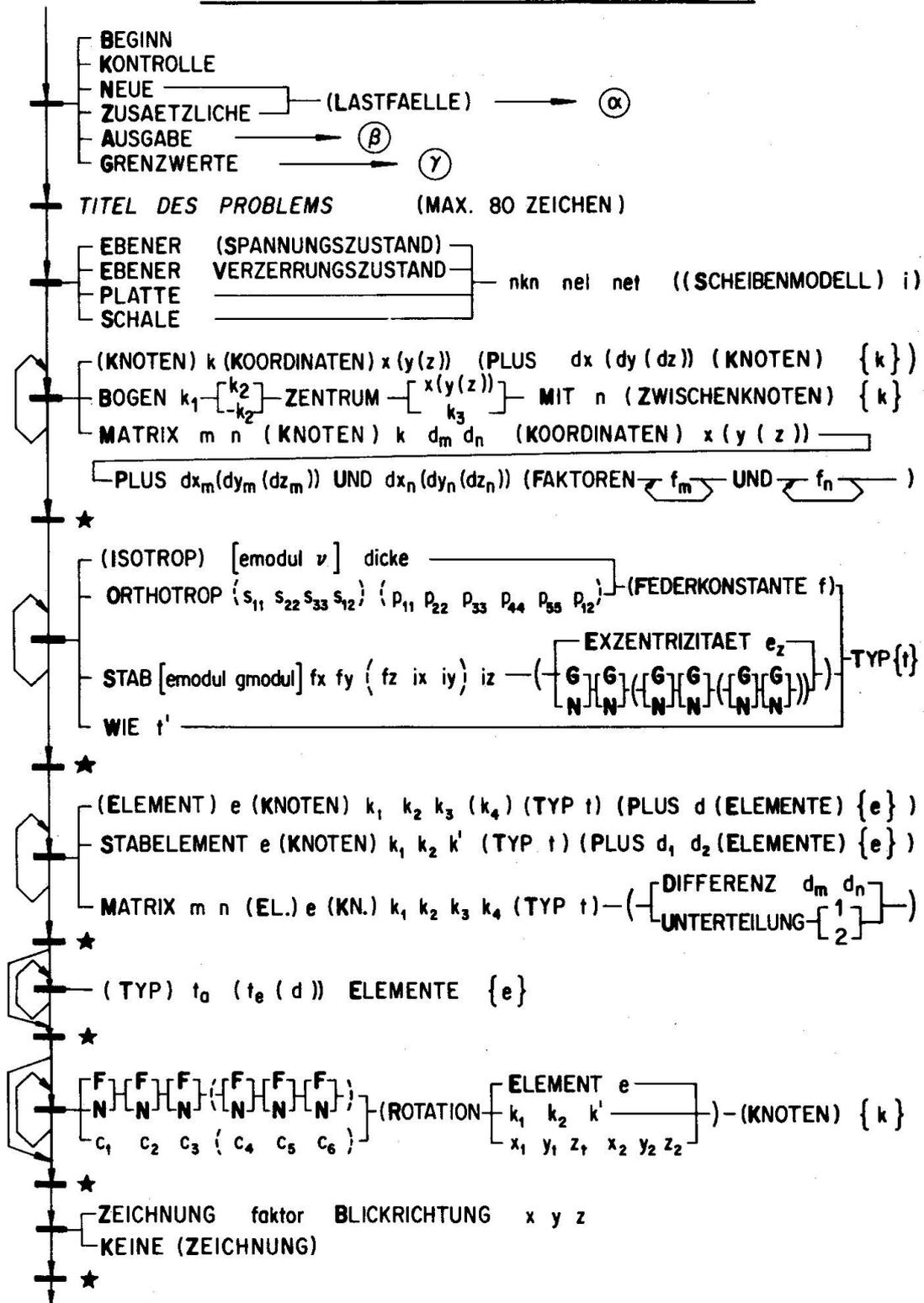ZEICHNUNG faktor BLICKRICHTUNG x y z
KEINE (ZEICHNUNG)

★

Fig. 4: Syntax diagram of the program FLASH for the input of structural data for plates and shells.
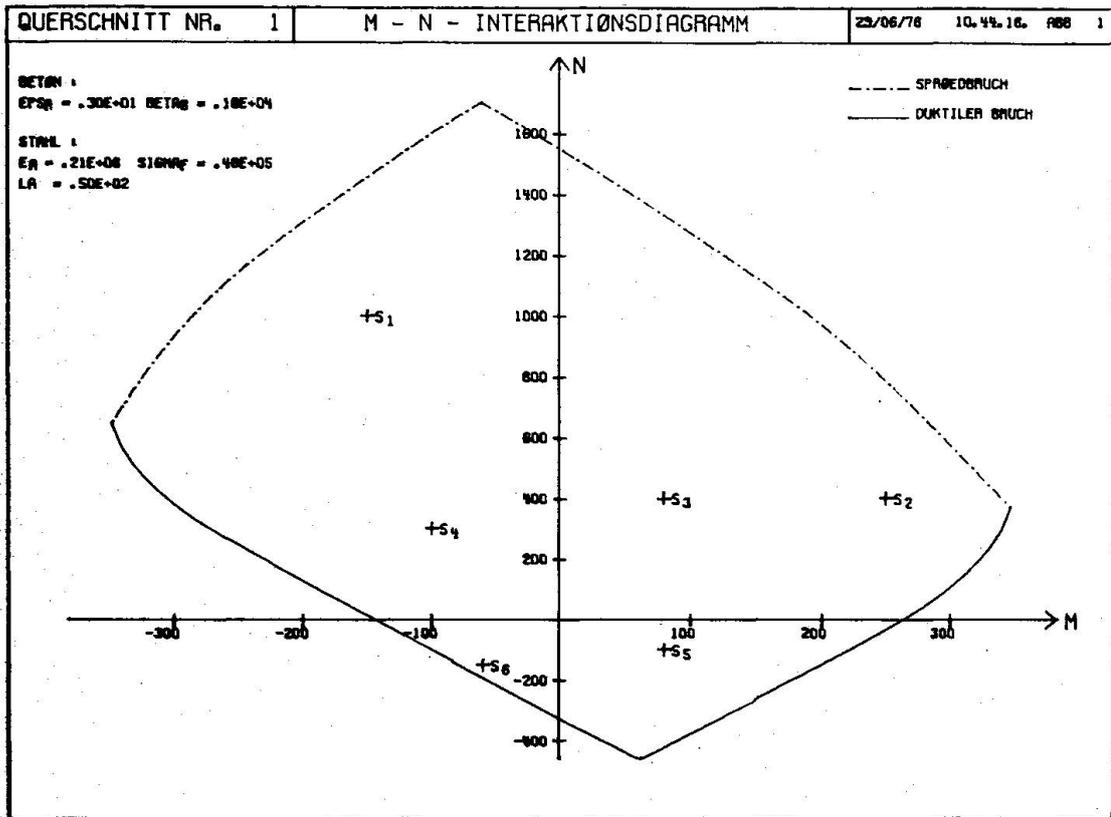
Fig. 5: Graphical output of normal-force bending-moment interaction for a reinforced concrete prestressed cross section.
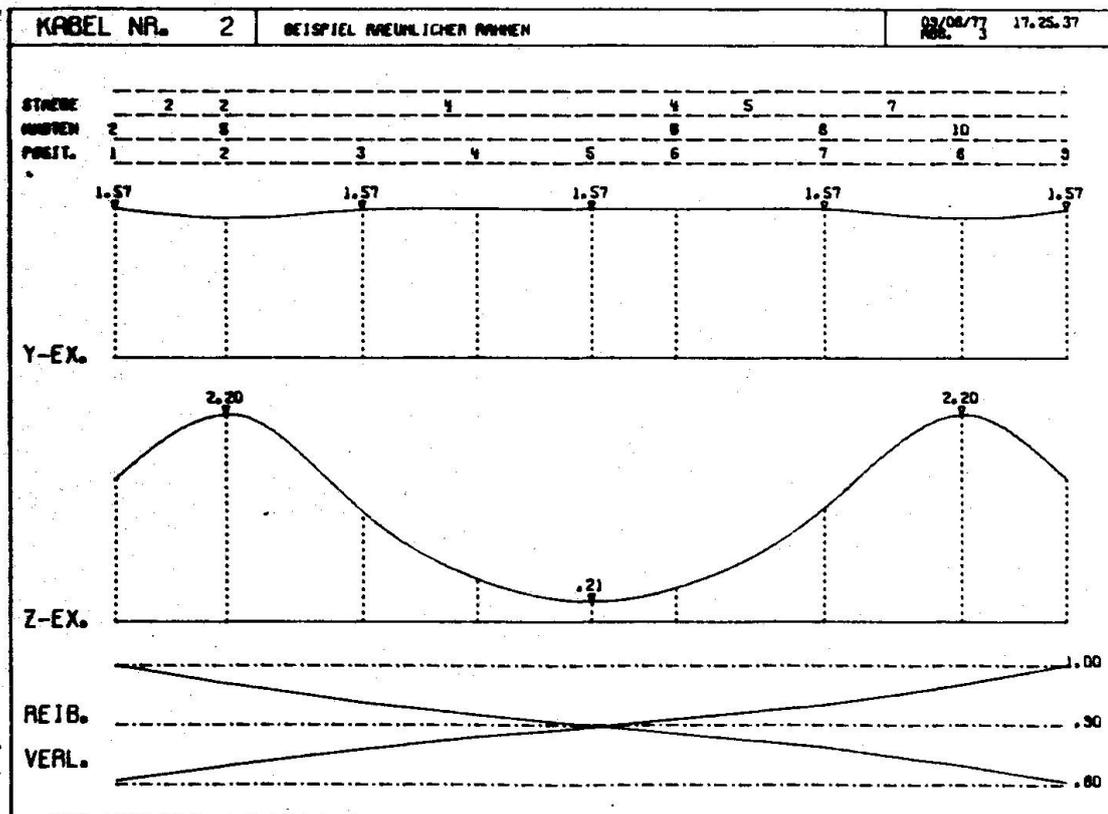

Fig. 6: Graphical output of prestressing cable geometry and friction losses.

Fig. 7: Graphical output of torsional- und bending-moment envelopes.

Fig. 8: Graphical output of the displaced shape of a multistory
         space frame.

Leere Seite

Blank page

Page vide