Zeitschrift: IABSE reports of the working commissions = Rapports des commissions de travail AIPC = IVBH Berichte der Arbeitskommissionen

Band: 31 (1978)

Rubrik: Session III: Professional responsibility

III.

MAIN SESSIONS

III Session – Professional Responsibility

Session III – Responsabilité professionnelle

III. Sitzung – Fragen der beruflichen Verantwortung

Leere Seite
Blank page
Page vide

IABSE
AIPC
IVBH

COLLOQUIUM on:
"INTERFACE BETWEEN COMPUTING AND DESIGN IN STRUCTURAL ENGINEERING"
August 30, 31 - September 1, 1978 - ISMES - BERGAMO (ITALY)

Professional Responsibility of Engineers

Responsabilité professionnelle des ingénieurs utilisant les moyens informatiques

Verantwortung Der Ingenieur Die Datenverabeitungsmittel Benützen

G. DEPREZ

Docteur en Sciences Appliqées - University of Liège
Director of CEPOC
Liège - Belgium

Summary

As it is practically impossible to check a computer program and to garanty the accuracy of the results only through computation it is nowadays generally admitted that the Consulting engineering overtakes the complete responsibility of the structures for which he ordered computer calculation, and the complete responsibility is a responsibility of results. The calculation center has the duty of performing the calculation with greatest care and to inform the consulting engineers of the limits of the used means (duty of means).

Résumé

Compte tenu de l' impossibilité pratique de vérifier un programme de calcul sur ordinateur et de garantir la précision des résultats par moyen informatique seul, il est actuellement admis dans la plupart des pays que l'ingénieur conseil supporte l' entière responsabilité des ouvrages pour lesquels il fait effectuer des calculs sur ordinateur et qu'il a l' entière responsabilité de la vérification des résultats. Sa responsabilité est une responsabilité de résultats. Le centre de calcul a l'obligation d' effectuer les traitements avec le plus grand soin et d' informer le bureau d' étude des limites des moyens employés (obligation de moyen).

Zusammenfassung

Da es praktisch unmöglich ist eine Berechnung auf Komputer so wie die genauich keit der Ergebnisse allein durch Datenverarbeitung zu überprufen ist es zur Zeit in den meisten Länden allgemein angenommen dass der beratender Ingenieur die ganze Verantwortung der Bauten für die er komputerrechnungen ausführen lässt trägt und dass er die ganze Verantwortung der Uberprüfung der Ergebnisse übernimmt. Seine Verantwortung ist eine Verantwortung der Ergebnisse. Das Rechen institut hat die Verpflichtung die Datenveravarheitung mit der grössten Sorgfalt auszuführen und dem Ingenieurbüro die Grenzen der gebrauchten Mittel mitzuteilen. (Verpflichtung der Mittel).

1. PRELIMINARY NOTE

Present consideration can be applied to problems met in complex scientific and technical applications. They cannot be extended to accounting and management problems.

Following points will be developed
- purpose of a computer run and its implacations,
- control of results after a computer run,
- tasks of a consulting bureau and computer center seen as distinct or common operations,
- respective responsibilities flowing from the performed tasks.

2. COMPUTER RUN

2.1. Computer and program

A computer is a tool having the capability to perform very quickly a large number of computations and data processings according to a previously coded procedure stored on suitable support. Every coded and stored procedure is called a program.

Therefore :

- a computer can only help if it is given a program defining the flow of operations to be performed to solve the submitted problem ;
- solution of a problem by means of a program is performed without outside intervention ; all decisions to be made must have been foreseen and coded within the program which will make use of the given data as sole information source.

Programs constitute the software accompanying the hardware, i. e. the computer and its peripheral equipment.

2.2. Programs and purpose of a computer run

The writing of a program requires the coding of a calculation procedure based on methods and calculation rules either existing or invented by researchers.

No outside interference being possible during a program run, this procedure must be autonomous ; only information planned during coding can and must be provided as data.

The defined procedure must be coded and stored on informatic support.

In practice, limits imposed by the present state of science, synthesis capability of human mind and hardware contraints prevent the writing of a general purpose program to compute any structure while taking into account all possible types of behaviour.

Except for a few standard structures to be analyzed by a standard method, only program computing some types of behaviour for some parts of structures do exist.

2.3. Correctness of a computer run

The complexity of the procedures to be used to solve these problem leads to programs containing from a few hundreds to several tens of thousands of statements.

No exact method enables one to check the correctness of these programs and the only available checking method consists in testing the program in the maximum number of cases which can be verified by other means in order to acquire a presumption of correctness becoming better and better and tending towards certainty.

Besides, fundamental methods upon which programs are based concern ideal behaviours for which an exact or approximate solution can be found by means of a mathematical theory. If one takes into account the fact that most available methods give approximate solutions and that, even if a correct solution exists, it can only be approximated because of the limitations on computer arithmetic, one must be aware that the computed solution will never be exact. The accuracy will depend upon the method being used and the order of magnitude of numerical values dealt with.

On the other hand, the use of a computer program implies that

- the structure to be studied must be fitted to the ideal model for which calculation method is available and the program exists.

- the set of data must be prepared for this model respecting the order and presentation requested by the program : they must be recorded on the appropriate support. This tasks leads to measuring, writing, coding and verifying numbers of data which can reach in the thousands. In spite of all safety measures, it is acknowledged that a very small percentage of errors may remain after performing these tasks ; provided they have significant effects, they will be detected because of their consequences on the results of the computer run.

Finally, one cannot forget that computers and their peripherals are subjects to accidents and breakdown as any piece of equipment. Safeties built in by manufacturers are such that most of these are announced to the user or stop execution.

2.4. Constraints imposed on the computer user

The lack of an exact method to check a program, the use of approximate calculation methods, the limits on arithmetic accuracy in the computer, the hypotheses required by the idealization, the risk of undetected errors during data preparation oblige to state that a computer analysis can never be considered as an exact analysis of a structure.

The user of a computer run has the duty to check that the results he receives are satisfactory for the intended purpose at a given point of a consulting job

Several methods exist to verify that. They differ according to the type of used procedure ; their underlying principles are recalled in paragraph 3.

After this realistic and unremitting presentation of the risks involved in a computer run, it is useful to compare them with respect to traditional calculation.

First a computer analysis enables one to find a solution to the analysis of different types of structures the behaviour of which was unmanageable by hand calculations. Second, when used for solutions where it competes with traditional means of calculation a computer analysis is much more efficient, accurate and less error prone than the former methods. This is of course what gives justification to its use. However its solution are more global and error can lead to more severe consequences. This entails the necessity of careful checks which generally happen to be simpler than hand calculations when available verification means are taken into account.

Possible verification means depend on the type of analysis ; they are within the grasp of every engineer. The underlying principles are recalled herebelow.

3. CHECK OF RESULTS GIVEN BY THE COMPUTER

3.1. Preliminary remark

Within the frame of this short note, it cannot be considered that well defined directives be given to the users concerning the results of such or such problem. This could be the purpose of further works of the task group.

One restricts the present paper to hints about :

- the presumptions to assess the quality of informatic means ;

- the verification principles

- the necessary informations to verify results

3.2. Presumptions concerning the quality of informatic means

These can take place at the level of
- the bureau or firm in charge of the analyses

- what is its repute ?
- what is its main sector of activity ?
- does it frequently solve this type of problem ?
- what help does it offer in data preparation and checking of results ?

- the program being used :

- who wrote it ?
- what method is it based upon ?
- is it frequently used ?
- for what kind of problem is it used ?
- on what computer is it processed ?

3.3. Verification principles

The implementation of these principles implies the knowledge of the analyzed behaviour and the method followed in the program.

It also requires an examination of the results given by the computer and their physical interpretation in structural terms.

All these verifications must not be simultaneaously performed ; it is the designer's responsibility to assess the risk of omitting some.

1st principle :
Verify normal excution of the program and fitness of hardware.

2nd principle :
Check agreement between
-the analyzed structure and its idealization,
-the idealization and the data,
-the data and the results.

3rd principle :
Verify physical soundness of the solution
This verification can bear upon, among others :
- static equilibrium,
- continuity or predictable discontinuities of internal forces,
- continuity or predictable discontinuities of displacements at different point of the structure.

4th principle :
Verify numerical results
- by comparison with previously obtained results for similar structures (experience),
- by comparison with a preliminary design or approximate results obtained by traditional means,
- by spot checks,
- by comparison with another analysis perfored by means of another computer program.

3.4. Data needed for the verification

For the verification of a computation to be possible, it is required to dispose of :
- the drawings of the structures
- the idealization scheme of the structure, including the supporting layout and the loading,
- information about the type of behaviour analysed by the program,
- a description of the analysis method used,
- information on how to interpret the computer listing :
  - meaning of symbols,
  - numerical formats,
  - units,
  - sign conventions,
- the listing of the treatment, which must contain :
  -the full set of input data,
  -the requested results,
  -intermediate results enabling the verifications mentioned in 3.3.
- an interpreted output, where only the useful values will appear, eventually composed by manual evaluation

While this is not mandatory, it is also very useful to dispose of automatic graphical output. Such drawings give the results under a synthetic form very convenient for verification. In particular , the idealised layout as introduced into the computer, force distribution diagrams and drawings of deformed shapes allow safe and quick checks which are tedious if the drawings must be handmade.

4. RESPECTIVE MISSIONS OF THE CONSULTING FIRM AND THE DATA PROCESSING CENTER

4.1. Preliminary remark

There are presently in the different countries represented in the IABSE organizations of various form and by laws which perform studies or computations using date processing. The functions they perform are sometimes difficult to compare.

In order to allow a comparison of these activities, it is necessary to tackle first a basic case where the analysis and data processing aspects are clearly distinct and then to examine how such activities may interfere within organizations performing both activities simultaneously, This method also allows to define the tasks of the different branches of a given organization.
The basic case will be that of a consulting firm using the services of an external computing center.

4.2. Mission of the Consulting bureau

The mission of a consulting bureau consists in the elaboration of a project on account of the customer and the control of the realization. The elaboration of the project involves design, anlaysis, calculation, drawing...
which result in the final plans used for execution.

The project progresses by successive steps ; preliminary studies, draft, project. it is during these steps that the project takes shape, the calculations become more refined and the different types of behaviour are examined.

The justification of the chosen solution and the set of calculations are recorded in the report established by the Consulting bureau.
This note is eventually submitted to an administration or to a controlling office.

The work of the Consulting bureau is rewarded on the basis of a percentage of the cost of the structure.

4.3. Mission of the data processing Center

A data processing Center writes programs or implements existing programs on a computer. The Center makes the program maintenance, brings the improvements needed by the accumulated experience or the evolution of the hardware, writes the user's manuals, runs the programs, records the eventual malfunctions, corrects the programs accordingly when possible and eventually brings some help to the user for the preparation of input data and the interpretation of results.

At the data processing Center level, the calcultation of a structure consists in running the program or programs selected by the Consulting bureau to analyse some behaviour, and eventually, in helping in the preparation of input and the interpretation of output.

The results of the calculation are recorded on listings and drawings produced by the computer's peripheral devices, and transmitted to the Consulting bureau.

The work of the data processing Center is rewarded on the basis of the resources used : use of the computer, of peripheral units, preparation of input, analysis of output. Such costs are independent of the cost of the structure and depend

only on the importance of the data processing Center resources involved.

4.4. Mixed missions

Some organizations perform mixed missions which are generally of on of the following forms :

a)Consulting bureau with its own Computer Center. Its is obvious that in this case, the main function is that of Consulting bureau while the data processing branch is just a tool.

b) Bureau specialised in computer-aided studies, limiting its activity to some types or parts of structures, and working as a contractor for other Consulting bureau.
What distinguishes this type of bureau from a data processing Center is that the study of the structure is entirely made by the bureau. The bureau is enti - rely free of the means to use to perform the study ; it must also select the behaviours to analyze and the assumptions to make. It acts in fact as a sub-contractor.

c) Society of one of the previous types a) or b), offering data processing services in parallel with its main activity. This case occurs for instance when a Consulting bureau puts at the disposition of another bureau its own programs without participating to the study. This may be an exceptional, occasional or current activity and it is necessary that no confusion may exist concerning the performed mission.

5. RESPONSIBILITIES OF THE CONSULTING BUREAU AND THE DATA PROCESSING CENTER

5.1. Responsibility attached to the mission of the consultant

Rewarded on the basis of a percentage of the cost of the structure, this mission requires that the adequate means be activated to complete it.
In particular, the Consulting bureau must : select the behaviours to be analysed by computer, select the data processing Center according to its possi-bilities, decide on the expenses which may be allocated to data processing, fix the assumptions, chooses an idealization, interpret the computer results. Being alone able to judge of the adequacy of the modelization, of input and output data, it is its responsibility to check the results coming from the Computer and, if needed, to refuse them. It must have the necessary means to perform this check and it belongs to the bureau to require them from the data processing Center.

5.2. Responsibility attached to the mission of the data processing Center

Rewarded on the basis of the cost of the prestations, the data processing Center must take any measure required by their good execution. This implies the obligation for the Center : to check the error-free execution of the program by a sufficient number of tests ; to bring the outmost care to the preparation of input data, to make sure that the hardware is in perfect working condition and that the computation was executed normally, to place at the disposition of the customers all information needed for checking and interpreting the results, to rerun any processing where an anomaly imputable to the Center might be detected, to respect its engagements about delay and prices, to inform clearly the Consulting bureau on the nature of its prestations.

It is important to precise that a program in perfect working condition may give useless results for some given numerical values. It may therefore be impossible to the data processing Center to furnish useful results to the Consulting bureau and it is important that the retribution process in exceptional cases be defined in advance.
In the absence of precisions on that point, it is normal to admit that the Consulting bureau might refuse to make any payment for the dubious processing.

IABSE
A I P C
I V B H

COLLOQUIUM on:
"INTERFACE BETWEEN COMPUTING AND DESIGN IN STRUCTURAL ENGINEERING"
August 30, 31 - September 1, 1978 - ISMES - BERGAMO (ITALY)

Readability of Design Programs

Lisibilite des programmes de calcul

Ablesung von Planungs Programmen

DONALD ALCOCK

MA, MS, MICE, FIStructE, FBCS

Alcock Shearing & Partners

Redhill, Surrey, England

Summary

Computer programs are being used to determine dimensions of structural members and details of reinforcement in engineering structures. Yet it is seldom possible for a design engineer to discover from a user's manual how such a program reaches these decisions for which he, the engineer, is ultimately responsible. This paper proposes a notation called 3 R for describing computer programs in a way that could make their logic intelligible not only to programmers but also to design engineers less familiar with software.

Résumé

Les programmes d'ordinateur sont utilisés pour déterminer les dimensions des éléments de structure et leurs liaisons. Néanmois, un ingénieur d' études peut rarement découvrir dans un manuel d' utilisateur comment un tel programme aboutit aux décisions pour lesquelles il est lui-même responsable en fin de compte. Ce rapport propose donc la notation 3 R pour décrire des programmes d'ordinateur de façon à en rendre la logique intelligible non seulement aux programmeurs mais aussi aux ingénieurs d'études moins familiarisés avec les programmes.

Zusammenfassung

Computer-Programme werden für die Ermittlung von Dimensionen von Bauteilen und Angaben von Verstärkungen im Maschinenbau gebraucht. Es ist jedoch selten für einen Bauingenieur möglich aus einem Anwendungs-Handbuch herauszufinden wie ein solches Programm Entscheidungen trifft, für welche er als Ingenieur letzten Endes verantwortlich ist. Dieser Bericht schlägt ein System vorgenannt 3 R - welches Computer Programme auf solche Weise beschreibt, dass darin enthaltene Logik nich nur dem Programmierer verständlich ist, sondern auch dem Bauingenieur, der weniger mit Software vertraut ist.

1. INTRODUCTION

If a bridge collapses because it was badly designed the consulting engineer is held responsible - whether the faulty calculations were made by incompetent employees or by computer. The legal problem is to prove the design, not the construction, was to blame - but if proof is possible the consulting engineer's insurance company has to pay up.

That is not necessarily the end of the story. Suppose the consulting engineer had based his bad design on the output of some proprietory program offered by a computer bureau? And if the bureau had offered use of the program on behalf of some other company then the bureau, in turn, would seek recompense from that company. It would be difficult because the author would maintain his program had been misapplied (a factor over which he could have no control) and point to the pile of rubble as evidence. Whatever the financial outcome and legal consequences of such a case, the problem posed here is that of a structural engineer injudiciously using results generated by a "black box".

The blackness of such boxes is examined in this paper, and a notation presented by means of which the logic of design programs could be clearly described - thereby reducing the opacity of potentially dangerous black boxes.

2. DESIGN BY COMPUTER

When computers were first used by structural engineers the only ready-made programs were limited in scope to simple analysis. The structural designer would check his results to ensure, for example, that reactions balanced applied loads; then he would work out areas of reinforcement and devise details of structural connections in the traditional way. The structural designer did not need to know much about the inner workings of the programs he used, but things have happened to change the picture. First the advance in analytical techniques (such as finite-element analysis) has made a computer indispensable and manual checking practically impossible; secondly, the computer is now used to decide the dimensions of structural members, details of connections, and precise sizes and arrangements of reinforcement.

2.1 Using Existing Programs

Despite the dangers of allowing a computer program to take this kind of decision it is inevitable that more and more consulting engineers will be compelled to do so. Design by computer is cheaper than traditional methods; failing to take advantage may mean going out of business. But few structural designers have the time or expertise to write their own design programs so most will have no choice but rely on those written by specialists. There will be ever more specialization because junior designers, being directed by their seniors to use existing design programs, will miss experience that would otherwise give them skill and judgement in the design process, hence the abilty to specify their own design programs.

There would be nothing wrong with a specialist writing a design program for other designers to use if only those designers knew precisely how the program reached its decisions, but the evidence is that they do not.

What information can a structural engineer get about a design program? Usually just its user's manual. This should tell him how to specify a problem by preparing data for punched cards or typing at a terminal of a computer. It should also explain how to interpret results produced by the program, and it should explain clearly what engineering assumptions the program makes and by what logic the program selects sizes and dimensions, but this kind of information is often lacking.

2.2 Experiences with some Design Programs

The Design Office Consortium [1], with the author as consultant, recently evaluated some publicly available programs for the design of reinforced concrete beams according to British Standard Code of Practice CP110. All programs had users' manuals which explained clearly enough how to prepare data, but given an identical design problem they produced amazingly different solutions. In a typical cross section the area of steel considered necessary by one program was several times that specified by another.

Except for one program (in which mistakes were found and subsequently corrected by the program's originators) no program seriously defied Code of Practice CP110; the enormous variance was permissible under the code. Yet from reading the users' manuals there were few clues to suggest the solutions would be different; a designer might reasonably have assumed all seven programs would design much the same beam. In other words the users' manuals lacked fundamental information.

2.3 Inadequacy of Information

It is not unknown for design programs to have no users' manuals at all; the designer gets a few rough notes, or perhaps a demonstration at a terminal to show how the program "asks" for everything it wants. More commonly a user's manual exists, but - like those describing the beam design programs mentioned above - it fails to explain fully how the program reaches decisions. Those secrets are concealed in the programmer's documentation which the user is not allowed to see - or which would be unintelligible if seen. Often there is no programmer's documentation either, the secrets lying buried in the programmer's head. But still such programs are used by designers - and structures built according to their results.

How, then, is a structural engineer to discover what a design program does? One answer may be that he can't. If a programmer does not want anyone else to know how his program works then potential users have little hope of finding out - and had best not use his programs because of the dangers described earlier. But if a programmer does want to communicate ideas to his fellow man he will do so; in words, by flow charts, or other means. On the other hand it is not easy for him to do so because of the gulf of expertise between an engineer who specializes in writing design programs and the practical designer who does not.

The next section of this paper introduces the idea of a notation for describing computer programs and designed to help span the gulf referred to above.

3. BIRTH OF A NOTATION

The author's firm was commissioned by the Design Office Consortium to write a computer program for calculating adjustments to fees payable to building contractors as influenced by certain Indices published monthly by the Department of the Environment. This program is called FORPA [2]. The commission was unusual in that the program was to run with minimal alteration on different makes of computer so that FORPA could be locally maintained wherever installed.

The traditional approach to such a problem would be to publish flow charts, specifications and listings. In this case, however, it was decided to devise a notation by which to describe programs generally - then publish a description of FORPA written in this notation together with a realization of FORPA transcribed from the notation into Fortran. This was done, and an identical Fortran realization runs today on several different makes of computer. Currently an APL realization is being transcribed.

The notation was designed to help in reading programs, writing them, and describing their arithmetic processes. Because reading, writing and arithmetic are called (in colloquial English) "the three R's" the notation has been given the name 3R.

3.1 Development of the Notation

Although 3R was devised with a limited aim - to describe the logic of FORPA to programmers and users alike - the notation was felt to have greater potential. Accordingly the Property Services Agency of the Department of the Environment commissioned the author's firm to assist in preparing a proposal [3] for the further development of 3R in cooperation with members of the C.I.B. working party, W52. The aim would be to refine and develop 3R and use it to describe substantial programs in the field of building design, thereby making the logic of decision processes in those programs intelligible to designers as well as programmers.

Concurrently (and from the point of view of software experts rather than building designers) 3R was presented by its designer, Brian Shearing, at a Seminar at Oxford University under the chairmanship of Prof. C. A. R. Hoare. Although some aspects of 3R were found wanting its reception was enthusiastic.

3.2 Relationship with Programming Languages

It is emphasized that 3R is a notation; not another programming language. It is possible to use 3R to describe a program in enough detail for a programmer to transcribe that program into a programming language, and for a potentail user of that program to comprehend its logic. Nevertheless 3R does have things in common with programming languages and may even be thought of as a "common factor" of common languages. For example, 3R has an assignment statement because most languages have assignment statements; 3R is not recursive because not all common langauges are recursive; and so on. Accordingly there is nothing in 3R to deal with machine-dependent details; where these crop up the 3R description has to break into human language.

4. A BRIEF EXPLANATION OF 3R

The 3R notation is simple. Although space forbids full definition, little detail is omitted in the following explanation of the notation as used to describe FORPA [2].

4.1 Overall Structure

A program described in 3R notation is a sequence of lines of text interspersed with blank lines for clarity. A line starting at the left margin is commentary. An indented line is called a "statement" and forms part of the 3R description, but may still include commentary enclosed in curly brackets.

"Words" of the notation are written in capital letters. "Names" - invented by the person describing a program - are written in small letters, several words being allowed in each name.

Logical flow is generally from one statement to the next until the final one, FINISH. But it is possible to parcel groups of statements into named "blocks" and put these anywhere in the text of a program without altering its logical flow - which simply "passes by" the definition of any block encountered. Definition has the form:

```
LET example block BE
   {sequence of statements}
END OF example block
```

Writing the name of such a block in the main program - as though the name were a statement - is called "invoking" a block. It implies logical replacement of the name by the sequence of statements in the block so named. A block may be invoked not only from the main program but also from within another block, and that from within another, and so on indefinitely - provided that no block is invoked recursively as a result.

Although logical flow may be "nested" to any depth as just described there may be no textual nesting of blocks (with consequent privacy of an enclosed block to its enclosing block); in 3R notation all blocks are at the same level. Likewise there is no nesting of loops or conditional statements - the effect of nesting is achieved by writing one or more "blocklets" within a block as explained later. The structure of programs described in 3R notation is constrained to be simple and linear so that a reader has only one level of thought to contend with at a time.

Communication between a block and the invoking piece of program is by arguments or shared variables or both. This is explained later.

4.2 Variables and Assignments

Variables must be declared before being referred to (not necessarily at the beginning of a program or block) and may have their range specified. In the examples below "colour" may take only three scalar values "red", "white" or "blue"; "number of file" may take any integral value from 1 to 99; "total number of files" only the value 99. Character variables have their limit of length specified as illustrated by "name of file" which may not contain more than six characters.

```
VARIABLE colour IS red OR white OR blue
VARIABLE number of file IS 1..99
INVARIABLE total number of files IS 99
VARIABLE name of file IS CHARACTER*6
```

Conventional real and integer variables may also be declared. And variables may be subscripted, in which case the ranges of subscripts must be specified.

```
VARIABLE stiffness matrix [1..300,1..50] IS REAL
VARIABLE list of six titles [1..6] IS CHARACTER*72
```

Variables declared in the main program are accessible to the main program and every block. Variables declared inside a block are private to that block.

Assignment to a variable is indicated by an equals sign. The expression on the right may involve symbols +,-,*,/,↑ (exponentiate by an integral power) in the conventional way. There may be several assignments separated by semicolons on the same line.

```
stiffness matrix [i,j] = -factor*modulus*inertia/(length↑power)
colour = red;   name of file = colour + "man"
```

The operator, +, in character operations denotes concatenation; the name of file above would become "redman".

There are two special operators, DIV and MOD, for use in non-negative integer expressions. These yeild an integral result, and integral remainder, of a division:

```
integral result = numerator DIV denominator
integral remainder = numerator MOD denominator
```

whereas the operator, /, always yields a real result. Otherwise "mixed mode" is not catered for, but special blocks may be assumed which are capable of converting from one mode to another. An example is:

```
x = real from integer (i)
```

When transcoded from 3R into a programming language such a statement would often become the unadorned statement "X = I". But the description of the program in the 3R notation is explicit and assumes no implicit operations.

4.3 Control Statements and Blocklets

An endless loop is denoted by a sequence of statements sandwiched between the words REPEAT and AGAIN. To leave a loop (transfer to the statement immediately following the word AGAIN) one of the statements in the loop may be the word WHILE or UNTIL followed by a Boolean condition.

```
REPEAT
   { optional sequence of statements}
UNTIL i > j    { or WHILE i <= j }
   {optional sequence of statements}
AGAIN
```

There is only one way to describe a choice of logical pathways in 3R notation - and when specifiying any choice all other possibilities must be explicitly catered for. The statement OTHERWISE FAIL is obligatory. An illustration of a choice between two pathways is:

```
IF x < y
   { statements to apply if x < y }
IF x > y
   { statements to apply if x > y }
OTHERWISE FAIL  { in this example x = y implies failure }
```

Separate pathways join again immediately after OTHERWISE FAIL. The null statement, PASS, is used is cases where no statements are needed on a pathway.

The design of this statement is based on Dijkstra's "guarded commands" [4] and chosen in preference to the unsymmetrical and ubiquitous IF..THEN..ELSE. Although it may seem unnecessarily arduous to enumerate every possible result of every condition, doing so has been found salutory - preventing mistakes that would otherwise have crept into programs. And certainly the person who transcribes from a 3R description enjoys the certainty of all cases having been considered.

Execution of the statement, FAIL, implies the "status" of the program becomes invalid. In every program described in 3R notation lies the concept of its current status being valid or invalid. Status starts as valid, but becomes invalid if the logical flow meets the word FAIL or an inconsistency such as a subscript out of range. The idea behind the concept of status is to provide a tidy mechanism for terminating programs in error. By preceding certain statements with the word TEST, and consulting two special Boolean variables VALID and INVALID, status may be tested and the result acted upon.

```
TEST element = vector [i]
IF VALID
   PASS
IF INVALID { etc.}
```

Testing an invalid status revalidates it. It is possible to induce the status to be invalid again by a FAIL statement. Unfortunately there is not enough space to discuss the mechanism of status more fully.

As stated earlier, loops may not be nested - nor may choice. Within a block, however, the effect of nesting may be achieved by naming - hence invoking - an inner structure as a "blocklet", then defining that blocklet. The first such definition is introduced by the word WHERE; subsequent ones by AND WHERE.

```
        REPEAT
           i = i + 1
        UNTIL i > 10
           inner nest
        AGAIN

     WHERE inner nest IS
        j = 0
        REPEAT
           j = j + 1
        UNTIL j > 10
           innermost loop
        AGAIN

     AND WHERE innermost loop IS { etc. }
```

All blocklets are written before the final END OF statement of their enclosing block. Any blocklet may access the variables declared within its enclosing block (as well as those declared in the main program) so there is no concept of "arguments" to blocklets as there is to blocks, as now explained.

4.4 Arguments, Input & Output

Additional communication is possible by invoking a block with "arguments". These arguments are interspersed among the words of the block's name to help the reader. The following block:

```
        LET stress at face of member BE
        VARIABLE ARGUMENT a IS REAL
        INVARIABLE ARGUMENT s IS top OR bottom
        INVARIABLE ARGUMENT n IS INTEGER
           { sequence of statements }
        END OF stress at face of member
```

could be invoked from the main program, or from another block, as:

```
        f = stress at (top) face of member (6)
```

where the actual arguments f, top, 6 replace dummy arguments a, s, n respectively. All dummy arguments must be declared either VARIABLE or INVARIABLE as shown.

Arguments may be declared not only in blocks but also in the main program. This is the means of communication between the program being described and its environment. The program's input is declared as a set of invariable arguments; its output as a set of variable arguments.

```
        INVARIABLE ARGUMENT keyboard[1..10000] IS CHARACTER*1
        VARIABLE ARGUMENT disk file[1..1000, 1..1000] IS REAL
        INVARIABLE ARGUMENT punched card[1..10000] IS CHARACTER*80
```

5. AN EXAMPLE 3R PROGRAM

The first program to be described in 3R notation was a simple word-processing program. This was subsequently transcribed into BASIC (one day's effort by Brian Shearing) and is the program by which the photographic masters of this paper were produced. Space does not permit reproduction of the word-processing program itself but the following example taken from an earlier paper [3] illustrates its style of documentation.

Following this example there is a reproduction of the statements of the program with commentary removed (automatically by the word-processing system) then a realization of the program in Fortran and another in BASIC.

5.1 A program for searching, described in 3R

The block of program below is designed to search for a given value in a pre-sorted table of values. If a match is found, the position of the value within the table is to be delivered. If no match is found, the block is to fail.

The following example would set the status of execution INVALID if the value were not fould in table[1]..table[n], but would set j if "value" were found in table[j].

```
TEST j = find (value) in first (n) words of (table)
```

The program to achieve the above example is as follows.

```
LET find in first words of BE
VARIABLE ARGUMENT j IS 1..1000
INVARIABLE ARGUMENT value IS INTEGER
INVARIABLE ARGUMENT n IS 1..1000
INVARIABLE ARGUMENT table[1..1000] IS INTEGER
```

Because the values in the table are sorted the method of "binary searching" can be used whereby the range of values considered is repeatedly halved until a match is found. During the search the range of values to be inspected is table[first]..table[last]. The initial range is the full table.

```
VARIABLE first IS 1..1000
   first = 1
VARIABLE last IS 1..1000
   last = n
```

The main part of the block keeps searching until a match is found.

```
REPEAT
   choose a value for j
UNTIL table[j] = value
   adjust the range
AGAIN

WHERE choose a value for j IS
```

Assuming a fairly regular distribution of values in the table, the position to be inspected from the table is chosen to be that in the middle of the current range.

```
   j = ( first + last ) DIV 2
```

```
AND WHERE adjust the range IS
```

If the value just inspected exceeds the given value then the new range is the lower half of the current range.

```
IF table[j] > value
   { first remains unchanged. }
   last = j - 1
```

If the inspected value is less than the given value then the new range is the upper half of the current range.

```
IF table[j] < value
   first = j + 1
   { last remains unchanged. }
```

The blocklet "adjust the range" cannot be entered if table[j] is equal to the value.

```
OTHERWISE FAIL  { computer failure }
```

Before resuming the main loop, "first" is checked not to have overlapped "last" indicating that the value is not in the table (or that the table is not properly sorted!).

```
IF first <= last
   PASS

OTHERWISE FAIL

END of find in first words of
```

5.2 The 3R Code without comments interspersed

```
LET find in first words of BE
VARIABLE ARGUMENT j IS 1..1000
INVARIABLE ARGUMENT value IS INTEGER
INVARIABLE ARGUMENT n IS 1..1000
INVARIABLE ARGUMENT table[1..1000] IS INTEGER

VARIABLE first IS 1..1000
   first = 1
VARIABLE last IS 1..1000
   last = n

   REPEAT
      choose a value for j
   UNTIL table[j] = value
      adjust the range
   AGAIN


WHERE choose a value for j IS

   j = ( first + last ) DIV 2

AND WHERE adjust the range IS
```

```
      IF table[j] > value
         { first remains unchanged. }
         last = j - 1
      IF table[j] < value
         first = j + 1
         { last remains unchanged. }
      OTHERWISE FAIL  { computer failure }

      IF first <= last
         PASS
      OTHERWISE FAIL

   END OF find in first words of
```

5.3 Transcription from 3R into Fortran.....and BASIC

```
      LOGICAL FUNCTION FIND(J,VALUE,N,TABLE)
C
C RETURNS .TRUE. WITH J SET IF AT TABLE(J)
C RETURNS .FALSE. IF NOT FOUND IN TABLE(1..N).
      INTEGER J, VALUE, N, TABLE(1000)
      INTEGER FIRST, LAST
        FIRST = 1
        LAST = N
10      J = (FIRST + LAST)/2
        IF (TABLE(J) .NE. VALUE) GOTO 20
        FIND = .TRUE.
        GOTO 30
20      IF (TABLE(J) .GT. VALUE) LAST=J-1
        IF (TABLE(J) .LT. VALUE) FIRST=J+1
        IF (FIRST .LE. LAST) GOTO 10
        FIND = .FALSE.
30      RETURN
      END
```

```
1000 DIM T(1000)
1010 REM
1020 REM SET N AND V, THEN GOSUB 1050
1030 REM IF V AT T(J) THEN R = 1,
1040 REM IF V NOT IN T(1..N) THEN R=0
1050 LET F = 1
1060 LET L = N
1070 LET J = INT((F+L)/2)
1080 IF T(J) <> V THEN 1110
1090 LET R = 1
1100 GOTO 1170
1110 IF T(J) <= V THEN 1140
1120 LET L = J - 1
1130 GOTO 1150
1140 LET F = J + 1
1150 IF F <= L THEN 1070
1160 LET R = 0
1170 RETURN
```


REFERENCES

1. Computer Programs for Continuous Beams - CP110, Design Office Consortium, Cambridge, 1978
2. FORPA Computer Program, Formula Price Adjustment for Building Contracts, Design Office Consortium, Cambridge, 1978
3. 3R - A Notation for Describing Computer Programs, Property Services Agency, Department of the Environment, London, 1978
4. Dijkstra, E.W., A Discipline of Programming, Prentice-Hall, 1976

IABSE
AIPC
IVBH

COLLOQUIUM on:
"INTERFACE BETWEEN COMPUTING AND DESIGN IN STRUCTURAL ENGINEERING"
August 30, 31 - September 1, 1978 - ISMES - BERGAMO (ITALY)

Problems Related to the Use of Computers

Problemes relatifs a l'emploi des ordinateurs

Probleme bei der Verwendung von Computers

A. GALLICO

Chief Engineer
ELC-Electroconsult,
Milan, Italy

Summary

In dealing with computers and their versatility, designers have to keep clear in mind the methodology of analysis, the approach to practical solutions, the discrimination of parameters, the discussion of models, the judgement of model limits and results. Factors are sometimes conflicting; available time and cost, users receptiveness, scientific improvement, extent of study and responsibility. Two engineering fields are outlined: structures and hydraulics at the design stage, operation and control of structures during and after construction.

Résumé

Lors de l'utilisation d'ordinateurs les ingénieurs doivent garder à l'esprit les méthodologies analytiques, l'approche de solutions pratiques, la discrimination des paramètres, la discussion des modèles, l'appreciation de leurs limites et leur validité. Les facteurs sont parfois en conflict: temps et coût disponibles, réceptivité des usagers, développement scientifique, importance des études et responsabilità. Deux domaines d' application sont mis en évidence: structures et hydraulique lors de l'étude de projet, fonctionnement et contrôle des ouvrages pendant et après la construction.

Zusammenfassung

Bei der Computeranwendung im Ingenieurwesen sind folgende Faktoren zu beachten: Berechnungsmethode, Weg zur praktischen Lösung, Parametervariation, strukturelle Modelle, Grenzen und Resultate der Modelle. Dies unter den Randbedingungen: Verfügbare Zeit, kosten, Stand der wissenschaftlichen Erkenntnis, Voraussetzungen der Benützer und Verantwortlichkeit. Am Beispiel der Anwendungsgebiete Statik und Hydraulik werden die aufgezeigten Probleme diskutiert.

1. METHODOLOGY

With the advent of computers, designers find themselves confronted by a substantial and rapid evolution of the criteria to be followed in their work.

Such an evolution allows the use of a vast field of techniques of analysis, made up from an almost unlimited capacity of the computers, that in principle contribute to exploration and possibly resolution of problems of engineering practice which are diverse sophisticated and multi-discipline.

Confronted by such a situation, so favourable with regard to the investigative possibilities, the designers, who in the end find themselves in the responsible position for the conception and realization of the project, are called upon to face problems of choice, use and guidance of the analytical tools at their disposal.

To be able to make the best use of the resources and versatility of methods of calculation for a practical result, the following criteria must be taken into account:

- Specify the mechanics of the problem to be resolved and focus the approach to the solution: such a criterion is not strictly bound to the use of modern techniques of calculation but gains importance for the following steps;

- Define the validity of the starting parameters and of the data initially available, in as much as the input is frequently incomplete or partially missing: the initial calculations will give best results in parametric form;

- Discuss the models which will represent the real structures;

- Understand the numerical analyses and their programming, maintaining an almost continuous contact between the computing centre and the designers;

- Judge the limits and validity of the mathematical models and their results: it is during this most critical phase of the process that the reciprocal contribution of the designers-calculator assumes the greatest importance;

- Derive the conclusions and conserve the autonomy of the decisions with regard to the practical utilisation of the analysis.

The observance of such criteria is not easy. Experience teaches in fact that in the development of project activities and use of computers there can arise situations, at times conflicting, which must necessarily be resolved, such as:

- time and funds at disposition, not so much concerning the actual execution and work of the computers but rather in relation to the activity at the interface;

- loss of the physical feeling regarding practical engineering problems;

- receptiveness of those people who in the end must use and put into practice the results from the technical calculations;

- motivation for the search for scientific developement;

- questions of trust and responsibility.

2. FIELDS AND APPLICATION

From the various engineering disciplines where the concept computer-designer is in current use, two fields may be pointed out:

2.1 Structural and Hydraulic Fields in the Project Stage

The application of modern techniques of numerical analysis with the use of computers, is particularly useful in the examination of variables, above all when the initial parameters are not well defined. With the study of the variables the designer has at his disposal the possibility of choice and must keep clear in his mind the possibility of not being in possession of the best definitive solution. Such cases are typically those in which infrastructure and foundations are concerned, frequently lacking sufficient survey information and consequently precise input (for example the mechanical characteristics of rocks and soils).

In superstructures, computers can easily optimise dimensions of civil structures, excepting when they must be drastically modified because for example of later definition of mechanical parts which the structure must receive.

In the purely hydraulic field computers are utilised very effectively for extending records of partial data for example in the calculation of spillway capacity, and the minima for operation of the system with optimisation of the corresponding static hydraulic structures.

2.2 Field of Operation and Control of Structures

In the field of operation and control of the structure modern methods are used to fix the manuals for operation of the works, the static hydraulic and dynamic controls for verifying the changes and foreseeing the trends. The possibilities for use are considerable,

with the condition that all are kept under reasonable control.

At first sight, seen in a unilateral form through the eyes of the responsible designer, it may be said that the benefits of modern techniques of analysis do not have need of proof and demonstration while the drawbacks increase in the proportion in which the methods are confused with the scope, the means with the products, the trust with the self-criticism.

3. ACTUAL EXAMPLES

3.1 Choice of a Type of Dam (Fig. 3.1)

The problem posed was that of planning a large concrete dam with a height varying between 60 and 180 m, and a crest length of 1 500 m, founded on an interbedded foundation of basalt and breccia.

The structure had to be mass-gravity or hollow-gravity. There was a requirement to proceed with the structural decisions and general layout having still limited data on the mechanical characteristics of the foundation, with also incomplete information on the technical and economic aspects of alternatives. The use of a computer was essential to deal with the comparative examination of stresses and deformations of the dam-foundation as a whole derived from different alternative combinations of the variable factors let alone, as well as to obtain the indicative estimates for the works.

The study was conducted on parametric bases not so much to define the best final solution, but to determine the influence of every variable factor, with appropriate alternatives and stated hypotheses. Such variable factors were: type of dam, height of structure, geometry of the excavation, stratification of the rock, homogeneity and discontinuities in the foundation, moduli of deformation, conditions of loading.

The systematic study allowed the problem to be solved without restricting the breadth of enquiry and field of validity of the results.

It is useful to point out that in the course of such analysis numerous classic hypotheses have been ratified, and sometimes also corrected, opening the way to further analyses of optimisation still using computers and afterwards physical models.

3.2 Structural and Hydraulic Study of a Spillway (Fig. 3.2, 3.3)

The question was of studying the most feasible technical and economical solution for passing exceptional floods from a basin used for hydroelectric purposes. The input data constitued two hurricanes

having respectively peak floods equal to 12 000 and 9 000 m^3/sec, with flood volume of 1 000 and 1 500 millions m^3: to the hurricane of greater peak, corresponded the smaller flood volume.

The variables to be considered consequently were the following: the occurrence of the first hurricane or of the second, or of both assuming variable intervals between the first and the second and considering the case of the first followed by the second and viceversa; the volumes impounded for the greatest possible flood routing with the purpose of protecting the area downstream: the levels of the spillways, the type of open-air works and tunnel spillways, the dimensions of the gates, the static and hydraulic structures, the quantities for the works and their cost.

With the use of the computers as well, in relatively simple programmes of calculation the alternatives were examined and the practical solutions optimised satisfactorily. Notwithstanding the noteworthy mass of work computerised and translated into drawings for the project, this was not sufficient; in fact the problems of erosion downstream of the structure were not considered and the designer had not paid sufficient attention to the possibility that some of the gates might not be operating producing unsatisfactory consequences hydraulically and statically on the structures theoretically optimised.

The example quoted here is typical of the possibility of forgetting particular points in the course of analysis, confronted by the need to deal with problems of practical engineering: the responsibility is certainly not in the use of computers but in not giving to the computers the whole of the problem to be resolved.

3.3 Operation of a Hydroelectric System

The question was that of a system of power plants in series where for energy motives the reservoirs had to be generally maintained at the highest levels possible within the limits of safety.

The despatching operation centre had constructed a model evidently based on the optimisation of the production of energy and removing autonomy for decision from the local control.

During the period of heavy precipitation the local operators obeying their instructions were opening the gates of the spillway following the orders given to them from the centralised model. In a period of heavy rainfall, however not exceptional, the operators had advised the despatching centre of the opportunity for lowering rapidly the impounded levels to progressively prepare the hydraulic works for absorbing the incoming water volume without danger. The despatching centre had not accepted the suggestion and had not ordered in time the opening of the gates in view of the immi-

nent danger of overtopping.

When finally the decision was taken, the situation had reached the point of catastrophe, the gates could not be opened completely and in time, the levels of the reservoirs went over the allowable level, and overtopped two earth embankments which were almost completely destroyed.

This case is cited as an example of loss of flexibility in the use of a predetermined model considered incorrectly to be perfect.

4. CONCLUSIONS

The following main conclusions are drawn:

- input data are often incomplete or with parts missing, particularly concerning infrastructure - foundation characteristics and therefore it may be wrong to completely rely on the computed results;

- models are never perfect nor complete and therefore problems should be analysed parametrically in order to derive the rational choice of solution;

- although designers and users should autonomously make the final decisions, computing centres and their programmers should be ready to contribute to the interpretation and understanding of the interface steps and of the limits of applicability of the algorithms used.

Annexes: Fig. 3.1, 3.2, 3.3.

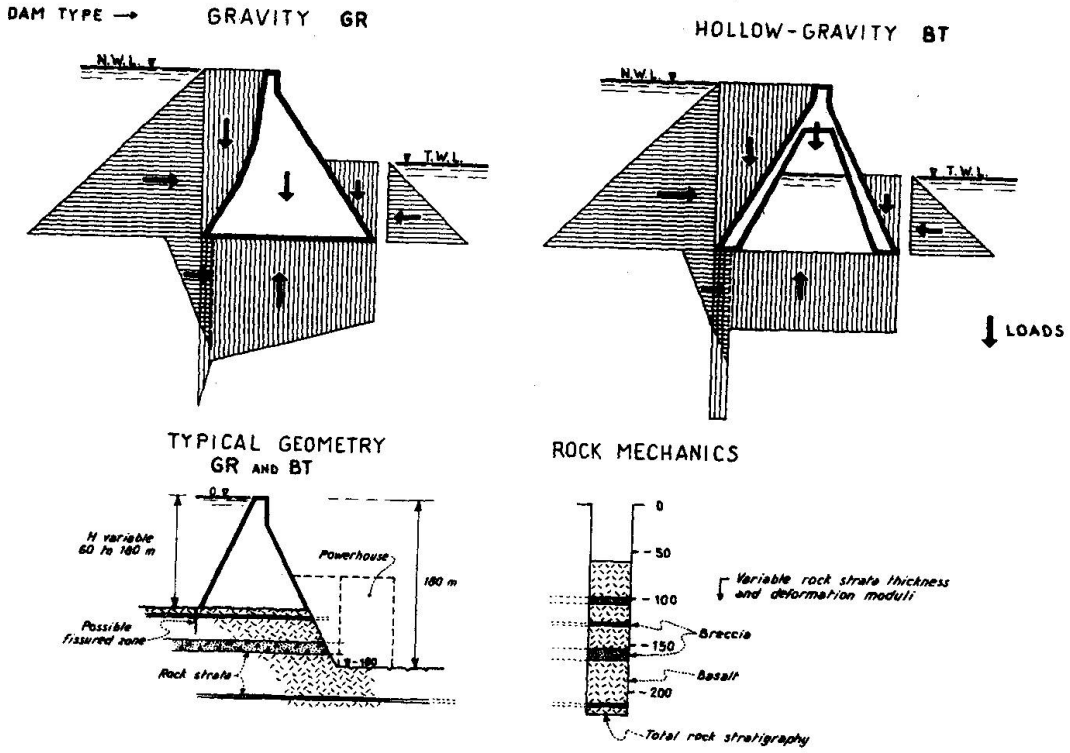

Fig. 3.1 - Investigation on dam-foundation complex

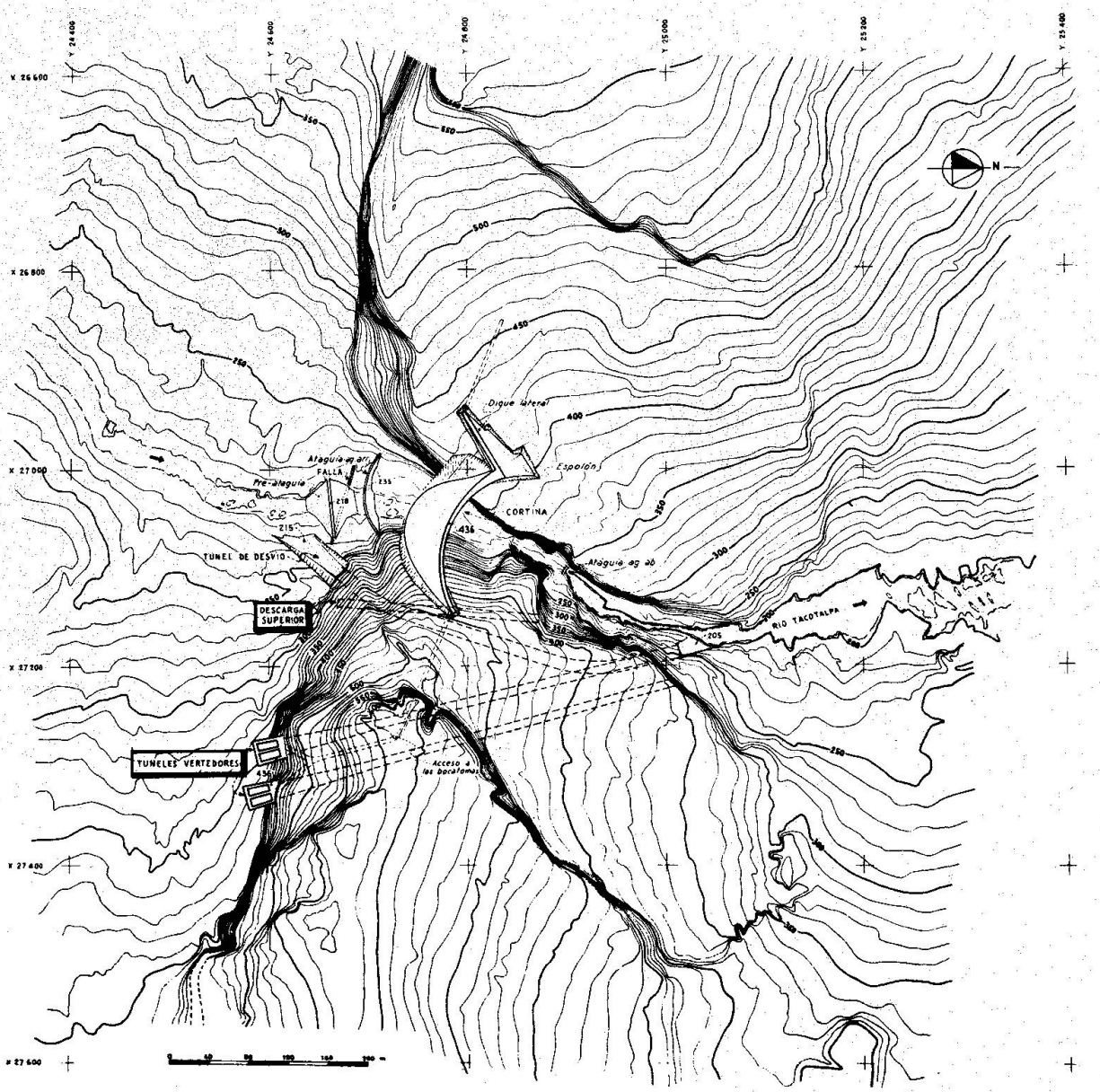

Fig. 3.2 - Study of dam-spillway alternatives. Layout

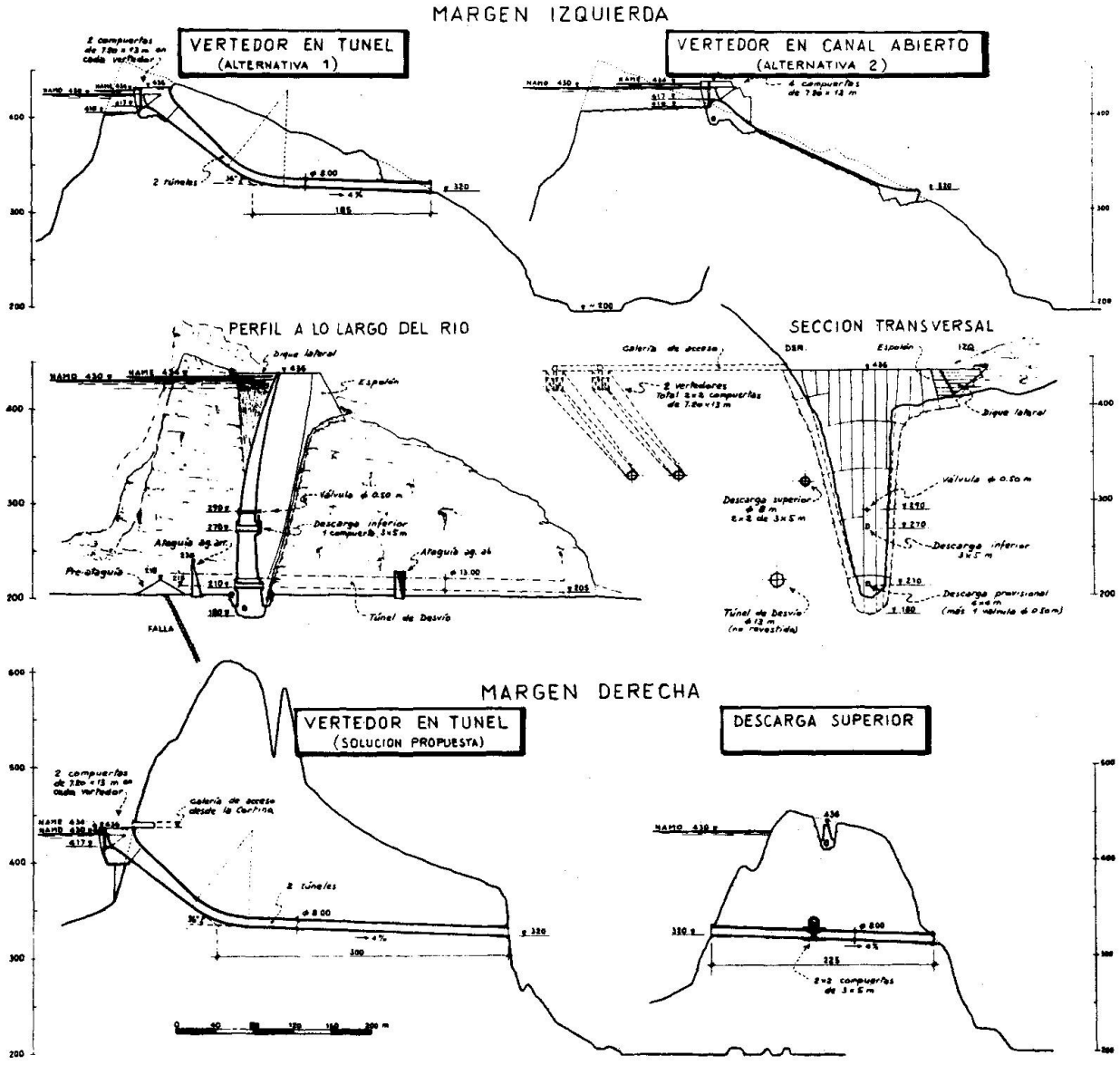

Fig. 3.3 - Study of dam-spillway alternatives. Sections

Leere Seite
Blank page
Page vide

IABSE
A I P C
I V B H

COLLOQUIUM on:
"INTERFACE BETWEEN COMPUTING AND DESIGN IN STRUCTURAL ENGINEERING"
August 30, 31 - September 1, 1978 - ISMES - BERGAMO (ITALY)

The Responsibility for Electronic Calculations

La responsabilité pour les calculs exécutés par l'ordinateur

Die Verantwortung für elektronische Rechnungen

P. KLEMENT

Dip. - Eng. Dr. tech., o. Univ. Professor
Technical University of Graz
Graz, Austria

Summary

The designer is full responsible for electronic calculations as he usually was for manual calculations. He has to check the results of the calculation which is possible due to equilibrium and compatibility costs. As even for tested programs sometimes hidden errors exist testing and checking of the program itself is not enough for judgment of the correctness of a calculation.

Résumé

Le projeteur est aussi bien responsable des calculs exécutés par l'ordinateur que par lui-même. Le contrôle d'un calcul doit inclure les conditions d'équilibre et de déformation. Les programmes éprouvés peuvent toujours contenir des erreurs cachées; il n'est donc pas suffisant de se contenter d' un test, ou d'un contrôle du programme pour une appréciation définitive du calcul.

Zusammenfassung

Die Verantwortung für elektronische Rechnungen liegt wie für manuelle Rechnungen beim Entwerfer. Die Uberprüfung einer Rechnung muss die Ergebnisse erfassen, was mit Gleichgewichts - und Verformungs - kontrollen fast immer möglich ist. Da auch bei getesteten Programmen versteckte Fehler existieren können, kann der Programmtest und die Programmüberprüfung nicht für die endgültige Beurteilung der Berechnung ausreichen.

When designing buildings it was the usual way that the responsibility for the calculation was at the designer, even when some detail calculation was done by auxiliary designer. The man in charge of the job was able to overlook the whole calculation and therefore no difficulties in checking such calculation arose. By simple formulas it could be checked that all assumptions are reasonable and when usual methods were used detail checking of figures could be left to auxiliary personal.

In some way methods of checking calculations have changed since computers are used and new problems arise about the responsibility for the total design. Due to my experience the designer should be responsible for the calculation. He must find a way to check the calculation of the computer to be able to take the responsibility. For further remarks there is no special difference if the computer center belongs to the same firm as the designer, or if the computer center is an own firm and is calculating on a commercial base.

Some of the remarks which follow will be valued not only for checking calculations of a computer but also for checking of programs during their developement.

All input data should be systematically printed to give the possibility for calculating the same problems manually or with other programs. Additionally, preferably intermediate results should be printed to allow checks on random sampling such figures for a equilibrium or compatibility tests. Sometimes, for well known programs, such checks are enough to proof the credibility of the results. Usually such controll calculations are even simple, when the program uses a very complicated procedure. As an example, even the results of a large system of equations can be checked by multiplication of one single line of this system. The total equilibrium of a complicated structure calculated by finite elements usually is also not difficult to check. Naturally some labour has to be invested.

Praxis shows that a too strict demand for such checks would especially hinder the developement of new procedures. The following way gives as well for the computer center as for the designer a good economy on their cooperation.

Naturally some confidence and reliance between both of them is needed. The progammer has to give a very exact description of the methods he uses and he must show the limits of these methods and has to show the sensitivity against defaults and small changes in input datas. Programs should be safe in an economic way against failure in use, when this seems possible. Today, the security against machine failures is nearly 100%. The printed calculation should give, additional to input data and intermediate results, the final results and also dimensions if the program is not designed to work dimensionless.

As usually such calculations need to be copied very often calculation sheets should be signed in a way that they cannot be confused with calculations of similar kind or even with nearly the same input data at the same day. A very good way is to print the date and time of the beginning of the calculation on every sheet.

Everybody who has to do with computer calculation will remember the case, that programs, which have been tested some years ago and which seemed to be correct due to the experience of hundreds of calculations, show bad results after years when input data are in a special constellation. The mistake is due to the fact that there is a large number of logical ways through the program and it is impossible to test all of them during testing a program. Nobody should therefore make the computer center or the programmer too much responsible for the consequences of such wrong calculations. A computer center working on a commercial base should deliver a new calculation without additional costs if the mistake is due to a wrong program. But the responsibility for technical consequences must be with the designer.

We must therefore insist on the fact that testing a program does not give us the security that the calculations are correct. There must be at least some check of the results to be safe.

As said before, for a lot of problems the printing of intermediate results allows a manual check at least on a random base. For more complicated problems as there are Eigen-value problems the printing of the matrix and the printing not only of the Eigen-value but also of the Eigen-solution were a valuable help. Even programs which are

proofed since years can give wrong results due to changing in the compilers for new installed machines.

When there are calculations of extremely large extension the responsible designer will be compelled to check such calculations with tests or in comparison with results of another program. As there are sometimes - especially for difficult problems - no other programs available a second calculation with the same program but with the input data made according to the drawings by another person will give safety against errors.

When making compatibility tests for the deflections it should be noted that for manual calculations the virtual forces can act at any chosen statically determinate system and therefore such a check is very easily done.

Most of these ideas have been fixed in "Allgemeine Richtlinien für die Vorlage und Prüfung von statischen Berechnungen mit programmgesteuerten Rechenanlagen (elektronische Berechnungen)" which have been edited by the Österreichischen Stahlbauverband in 1967. These recommendationes give at least a good guidance for checking of electronic calculations and should be published to experts.

IABSE
A I P C
I V B H

COLLOQUIUM on:
"INTERFACE BETWEEN COMPUTING AND DESIGN IN STRUCTURAL ENGINEERING"
August 30, 31 - September 1, 1978 - ISMES - BERGAMO (ITALY)

The Influence of the Computer on the Professional Ethics

L'influence de l'ordinateur sur l'ethique de la profession

Einfluss des Computers auf das Berufsbild

I. UHERKOVICH

Head of the Design Office
Losinger Ltd - VSL International
Berne, Switzerland

Summary

The use of calculation programmes developed by third parties entails a serious encroachment on the ethics of the engineering profession. Complex calculations cannot be verified by the ordinary practising engineer. The lack of effective possibilities of checking leads automatically to the question of the legal responsibility for the results and the ethical justification of the use of such calculation. The consequences of this transformation cannot be ignored by the engineer profession.

Résumé

L'utilisation de programmes d' ordinateur développés par des tiers engendre de sérieux empiètements sur l'éthique de la profession d'ingénieur en raison de l'impossibilité pour l'ingénieur praticien de vérifier par lui-même l' exactitude des résultats de calculs complexes. Cette situation met indubitablement en question la responsabilité juridique des résultats ainsi que la justification éthique de l'utilisation de tels calculs. L'ingénieur ne peut se permettre d'ignorer les conséquences de cette évolution.

Zusammenfassung

Die Anwendung von auswärts hergestellten Berechnungsprogrammen bedeutet einen schweren Eingriff in die Ethik des Ingenieurberufes. Komplexe Berechnungen sind für einen normalen Ingenieur nicht übersehbar. Mangels richtiger Kontrollmöglickeiten stellt sich automatisch die Frage nach der rechtlichen Verantwortung für das Resultat der programmierten Berechnung und der ethischen Verantwortbarkeit der Anwendung solcher Berechnungsmethoden. Die eingeleitete Wandlung des Berufsbildes bringt Konsequenzen mit sich, welche unbedingt zu berücksichtigen sind.

1. INTRODUCTION

To call the invention of the computer a revolution has become a cliché. In most cases it is used in connection with the machine, whereas the revolution is actually somewhere else.

Up to now every university graduate is proud of the fact, that he comprehends the complexities of his work and not barely the working methods themselves. The method is of course already clear to him based "on his higher level of education". Thus a proudly defended line exists today between engineers graduated from universities and those from engineer colleges.

Very often one hears the statement that a university-engineer does not have to know the formulae by heart nor to look them up in a book as he can develop them himself.

I have no intention to talk about the differences between good and bad engineers. What interests me in this connection is the professional image and the professional work in general. I therefore ask myself the question: is the above criterion still valid today after the introduction of computer-programmed calculations? Did not the computer degrade the engineer to a technician? I would try to analyze this question in a sort of question-and-answer-game.

2. WHAT IS THE INFLUENCE OF USING PROGRAMMED CALCULATIONS ON THE ACTIVITIES OF A CONSULTING OFFICE ?

Outsiders will of course immediately state that the productivity increases. To this I would like to make the following comments from my own experience: Recently our office designed a bridge which was similar in size and concept to one which I had designed some 17 years ago together with some colleagues. Adding the time spent (engineers and draftsmen) for the later bridge I came to approximately 10'000 working hours. A comparison made with the earlier bridge revealed that only some 6'000 hours were spent on that project. Despite the fact that at that time we only had an electronic calculator and that for the later project we used the computer facility practically to its fullest extent (35 kg of print-out will prove this) we required today almost twice as much time to achieve the same. Increased productivity? Hardly!

From my experience the effect of the computer is not in the quantitative area even if it is true that the computer delivers many more figures than has it been analyzed earlier.

In the earlier days the personnel structure of a consulting office was quite manifold. The engineer in charge knew exactly who could perform how much. Even though everybody could present similar graduation certificates, not everybody was given the same type of work.

Today it is much more difficult to know the real limits of theoretical capacity of an engineer. All or almost all will analyze structural problems in the same way: with the help of a computer programme independent of the fact

whether without computer they would be able to solve the problem or not. To say it a bit exalted: with a computer programme we have created a means which will allow also the incompetent to give the impression of being competent.

HOW CAN ONE TRUST THE RESULTS OF COMPUTER CALCULATIONS ?

The machine can not be wrong is an argument which very often leads to an almost superstitious reverence of the print-outs from a computer. Despite that we almost daily come across errors "made by the computer"; starting with wrong results from sports events to the erroneous reminders of invoices which have been paid long ago. And in engineering it is no different.

It is clear that the machine rarely makes errors and once it happens it will be clearly visible. The sources of errors are however numerous. They can range from wrong interpretation of the instructions, erroneous selection of the rheological model, syntax errors etc. to the wrong interpretations of the results. Many of these possibilities of errors also exist with traditional calculations. They are however detected much easier.

To check an output of several hundred pages would actually destroy the advantages that result from computer calculations. Certainly there is also the possibility of making intelligent checks but where there are doubts about the intelligence of the checker, there will be little reliability in the figures of an output. In addition there are two more important considerations:

- Most checking methods require the command of traditional calculation methods and are therefore the result of the so-called old school.

  The young generation has however a completely different approach to the problems. They master the computer much better; however, rarely do they have the experience of the traditional design methods which would give them the possibility to do a quick check.

- The automatic interlinked calculations with a highly complicated model and high accuracy requirements which I have tried to show in my paper "Possibilities and problems in connection with construction stage analysis..." will give very few if any possibilities to check the results with the required accuracy. Even the most experienced engineer will here only be in a position to detect major errors.

WHO IS RESPONSIBLE FOR THE CORRECT FUNCTIONING OF THE PROGRAMMES ?

This is a question for a lawyer rather than for an engineer.

You all know the famous sentence below the heading of almost any computer programme manual: "this programme has been tested to the best of our knowledge, any responsibility in connection with the use of this programme must however be declined".

In other words you are buying a cook-book but you will have to convince yourself about the edibility of the meals cooked according to the listed recipes.

With these culinary specialities it will be rather simple, although not without cost, if you only have to cook the meal once to verify the result.

The body of a programme, however, contains such a number of ramifications that to examine them all by the user is out of the question. Certainly when we make calculations by hand, we have already trusted the theories we learnt from a teacher at school or found somewhere in a technical newspaper. The author of a paper does not bear the legal consequences of the application of his theory either, here only the possibility of a large control is given to the professional readers. A wrong statement in a professional article rarely remains unobjected.

Programmes on the other hand are "top secrets" - hardly anybody published his list. There are not only technical but mainly economical obstacles to this. The possibility of recuperation of the generally important costs would practically be nil from the moment of publication. Unfortunately there is no institute in the world to my knowledge which would check the correct functioning of foreign programmes and which would confirm the result of such a check by its seal. Therefore nothing else remains than to use programmes for which nobody takes any responsibility (apart from proper developments which, however, might be exceptions rather than the rule).

IS A SPECIALIZATION BETWEEN THE DESIGNER AND THE EXCLUSIVELY ANALYTICALLY WORKING MATHEMATICIAN THE WORKING PROCEDURE OF THE FUTURE ?

In the early use of computers for engineering calculations nobody thought about the question of responsibility. The computing plant was inaccessible for most of the offices for cost and handling reasons. This situation led to a solution which still is considered the best by many people: a division of the duties between design offices and computing centres offering a full computing service.

The development has partly reversed this solution - firstly because the "hardware" has become so cheap that a powerful machine today is even within the reach of investments of a smaller office, and secondly, because the division between the two different offices is quite disadvantageous, especially when flexibility is concerned. Nevertheless, the danger of specialization is not overcome, it has only been displaced from outside, i.e. from the institutions, to inside, into the office. This is due to the fact that in offices that own a computing plant, the computer is not accessible to all engineers. In such offices there are engineers who know how and what to calculate - but they have no influence on the utilization of the results, and on the other hand, there are others who are designing, but from whom the original duty of an engineer - mastering of the forces flowing in a structure and forming the structure adequately for this force flow - slowly escapes.

What has already happened half a century ago with buildings, the division of the profession into architects and engineers is presently going on in other fields of civil engineering.

Although we may deplore such a development with nostalgic regret, we can hardly stop it.

CONCLUSION

As a summary of the analysis in this contribution we can say the following on the development of the engineering profession after the introduction of programmed computation:

- The larger volume of calculations caused by outer circumstances (better knowledge of the material properties, perfectionistic codes, complicated construction methods a.s.o.) renders the use of computers compulsory.

- The existing possibility for analysing a structure without mastering the theoretical bases of such an analysis, coupled with the difficulty to control computerized calculation, leads to well founded doubts about the acceptability of such a situation.

- The division of programme manufacturing and the calculation itself generates serious questions about the legal responsibility for the effects of faulty programmes.

- The specific claims of the direct users of computers lead to a further specialisation of the engineering profession.

It is the duty of schools, engineering societies and organisations to consider these facts in order to remove the negative effects of the computer revolution and in order not to let the engineering profession drop to a narrow-minded level.

New steps are therefore required:

- at school
- in the transmission of new findings
- in the proofing of programme qualities
- in the check of the calculations.

Leere Seite
Blank page
Page vide

IABSE
AIPC
IVBH

COLLOQUIUM on:
"INTERFACE BETWEEN COMPUTING AND DESIGN IN STRUCTURAL ENGINEERING"
August 30, 31 - September 1, 1978 - ISMES - BERGAMO (ITALY)

Educational and Professional Implications of Reliability Assesment in Computerized Structural Analysis

Conséquences, sur l'éducation et la profesion, de la détermination de la fiabilité dans le dimensionnement de structures à l'aide de l'ordinateur

Einfluss der Zuverlässigkeits-Schätzung bei Computerberechnung von Tragwerken auf die Ausbildung und die Berufstätigkeit

G. MAIER
Prof. of. Dept. Struct. Eng.
Technical University (Politecnico)
Milan, Italy

A. PEANO
Dr. Dept. Struct. Eng.
Technical University (Politecnico)
Milan, Italy

Summary
The continuing growth of computerized structural analysis and its increasing impact on engineering practice raise the problem of assessing and improving the reliability of its results. Some aspects of this problem are briefly discussed here, precisely: verification and qualification of structural software; certification of program users and enhancement of their professional standars through various types of educational processes.

Résumé
Le développement continu du dimensionnement de structures à l'aide de l'ordinateur et son influence grandissante sur la profession d' ingénieur soulève le problème de la détermination et de l'amélioration de la fiabilité des résultats. Quelques aspects de cette question sont traités, et en particulier la vérification du logiciel applicable aux structures, la qualification des utilisateurs des programmes et l'élévation de leur niveau professionnel au moyen de divers types de formation théorique et pratique.

Zusammenfassung
Die dauernde Entwicklung der Computerberechnung von Tragwerken und deren steigenden Einfluss auf den Ingenieurberuf wirft das Problem der Zuverlässigkeits-Schätzung und Verbesserung von Resultaten auf. Einige Aspekte dieses Problems werden kurz dargestellt, insbesondere: Prüfung und Beurteilung der Qualität von Programmen. Qualifikation des Programmbenützers und Verbesserung des Berufsniveaus durch verschiedene Arten von Ausbildung.

1. INTRODUCTION

In the last two decades the cost of computer use has been reduced by a factor of ten roughly every five years and such a trend is likely to continue in the near future, along with further developments in computer technology. This circumstance provides sufficient reasons for surmising that computerized analysis will play in engineering of tomorrow even a more important role than it does nowadays. In particular, nonlinear and transient structural problems, until recently almost prohibitive, will be solved routinely in years that lie ahead. The results of such analyses are generally difficult to be interpreted and checked on the basis of engineering judgement and intuition, which are hardly illuminating in the presence of complex or unusual mechanical behaviors; on the other hand, a large number of decisions is bound to be based on results supplied by computers. In view of these facts and prospects, there are grounds to be concerned about the responsible use of computers for structural engineering purposes.

Clearly, there cannot be such thing as absolute reliability assessment of the results of a complex computerized analysis. Errors may be due to bugs in the software, to failures in the hardware, or to inappropriate use of both.Particularly in front of problems which are intractable by hand calculations and hardly accessible to intuition, a combination of inadequate computational tools and inexperienced users may lead to a situation which is undesirable and dangerous.

The scope of this paper is to discuss the actions which could be taken in order to raise the level of confidence of computerized stress analysis.

The narrowest range of possible actions concerns the "verification" of computer programs, i.e. checking that a program is actually capable to do with satisfactory accuracy the calculations for which it was designed. A more ambitious task, referred to here as "qualification",is to ascertain whether the mathematical model adopted as a basis for the analysis represents a sufficiently close description of the mechanical system in the engineering situation considered. However, reliability of computerized structural analysis can be established only by an integrated process which encompasses verification, qualification of computer programs and, finally, a critical appraisal of results, which will be referred to here as "validation".

Even more difficult and perhaps inherently elusive, appears to be any action aiming at an assessment of users' competence, and possibly at its formal certification .

Attention is paid herein primarily to measures apt to enhance the professional standards in the community of structural software users. This discussion, covering both continuing education and normal engineering curricula, inevitably leads to the institutions potentially active in this area and responsible for gathering and registering the results of assessments of merit, both of software and users.

2. VERIFICATION OF STRUCTURAL SOFTWARE

Verification means checking that computer programs do exactly what they are supposed to do. This is a difficult task,as programs for structural analysis may be comprised of several hundred thousands of FORTRAN statements and may be used following a large number of different computational procedures. Moreover, verification includes checking correctness and completeness of the program documentation as well. Programs are seldom documented adequately, in part because documentation is generally regarded as a tedious and distasteful chore to be done at the end of the job. Programming or documentation errors may be detected after many years of satisfactory use.In fact, particular applications may require unusual solution paths,array dimensions never used in earlier applications or non-standard mesh arrangements,thus leading to situations unforeseen by the program developer. Moreover,many kinds of errors produce a limited perturbation of the results and remain undetected until a skillful user faces an application which magnifies their consequences.

It can be said, therefore, that there is no such a thing as absolute verification. On the other hand,a thorough testing would be harodly possible, and in most cases useless, because computer programs become rapidly obsolete due to advances in hardware technology and computational mechanics. Note that the simple addition of a new capability may inadvertently damage existing well-tested procedures. These and otherreasons explain why the certification of computer programs was never attempted, although widely discussed since years, [1] .

In spite of all the above circumstances, verification of computer programs can be successfully carried out provided its scope be limited merely to generate an acceptable level of confidence. By this wording we mean that the uncertainties due to programming errors may be reduced to the same incidence level as those caused by hardware malfunctioning or mistakes in the design ad/or construction process.

An important step toward safer computer programs is the adoption

of a modern programming style that can be called "programming for debugging". The first generation of finite element computer programs has required average debugging costs as high as fifty percent of the total cost of developing an operational program [2]. Therefore it makes sense to plan a debugging strategy in the early stages of designing a new computer program. Being easy to debug should become one of the fundamental requirements of future computer software. Practical implications follow at two different levels.

At the software architecture level, a clear decomposition of the algorithm into functional modules must be sought. The purpose is to make visible the function of each module and the transfer of information. The concept of completely modular structure leads from the usual large general-purpose software systems to the so-called programming systems. The latter are simply a comprehensive set of software modules and data handling tools that can be combined in a very flexible way as needed by each particular application. Since each module can be debugged independently and new features can be added by means of separate modules, the overall software reliability is generally increased by an order of magnitude [2]. The cost of debugging is much reduced because the implications of a single program segment can be understood without taking into account remote parts of the program.

At the level of coding the various modules, "programming for debugging" means that a substantial effort should be spent in making the FORTRAN code readable. In particular, marginal computational benefits should be sacrified in favor of the linearity of the algorithm.

Guidelines for advanced and safe computer programming have been done by many authors and would not be pertinent to this Colloquium. However existing contributions to software engineering are dispersed in the literature and hardly available to many engineers-programmers responsible for the production of structural analysis codes. Since there are no established channels for teaching it, computer programming for structural engineering can still be regarded more as an art than as a science. The importance of this remark should be appreciated in view of current trends of computerized structural analysis.

Due to the increasing availability of less and less expensive and more and more powerful computers, of structural packages and of information on structural analysis techniques, an increasing number of users may try to develop their own proprietary packages. The result may be a mushrooming of poorly tested and highly unreliable software systems. Appropriate actions have to be taken by educational institutions and professional bodies in order to avoid unacceptable consequences. In this respect it would be useful to

establish and make available comprehensive sets of test problems, particularly for nonlinear and dynamic analysis. Program developers should be required to demonstrate successful solution of these benchmark problems before any stress report based on their program can be taken into consideration.

Clearly, the solution of a limited number of benchmark problems is not a substitute for the program verification and may engender a false level of confidence. However, the combination of better programming techniques and of a broad range of test problems certainly provides a significant level of verification.

3. QUALIFICATION OF ANALYSIS PROCEDURES AND VALIDATION OF COMPUTER RESULTS

The analysis of engineering structures rests on assumptions concerning the behaviour of the materials, the structure geometry, the nature of the applied loading and other aspects of physical systems. In any specific application, the use of a computer program is meaningful only provided the underlying assumptions (say for instance: linear elasticity) are shown to be acceptable. When design procedures rely heavily on the computer, the analyst may be induced to make assumptions which are difficult to check. For instance, non-axisymmetric bodies may be modelled axisymmetric in order to reduce the size of the problem. Analogously, significant details may be omitted in order to simplify the mathematical model. Even more difficulties are encountered in nonlinear problems. For instance, stress-strain curves derived from standard uniaxial tests may not be adequate for large strain elastic-plastic analysis. In fact, the assumed stress-strain curves are only averages over the specimen; on the other hand, they must be generalized to multiaxial constitutive laws through suitable hypotheses. Similarly in the presence of geometrical nonlinearities it is possible to perform a bifurcation buckling analysis. However, the practical meaning of the computed bifurcation load is difficult to ascertain unless a much more complex investigation on the imperfection sensitivity of the structure is performed as well [9].

The above remarks show that qualification of analysis procedures is not a simple task. However, there are areas where appropriate actions could and should be taken, in order to increase the reliability of computerised structural analysis.

E.g. in some applications the significance of geometric nonlinearities is difficult to estimate in advance. However it is possible to compute first the linear solution and then have an "a-posteriori" estimate of the magnitude of the nonlinear terms. Computer programs for geometrically nonlinear analysis should be able to furnish this information routively, whenever possible.

A second example concerns nonlinear material behavior. Many programs have a capability for elastic-plastic analysis. However the models of material nonlinearity available in the program may not be appropriate for important areas of application (for instance soils). In the preceding section the need for a comprehensive set of test problems has been pointed out. This set should encompass a wide range of fully solved elastic-plastic benchmark problems, especially devised for verifying the range of applicability and the level of accuracy of the material models available in the program.

Verification of the computer program and qualification of the solution procedure are not sufficient for verifying the consistency and accuracy of the computed results. We will call"validation"the set of checks usually performed "a posteriori" to this scope. There are at least four possible causes of concern. The most impredictable one is certainly malfunctioning of hardware: this may be due,e.g., to faulty integrated circuitry or to errors in the data transmission from a remote computer to the user's terminal. The second cause of concern is the possibility of undetected input errors. A good computer program contains checks of the consistency of the input data and of other quantities computed during the solution process (for instance,non negative diagonal stiffness coefficients). It seems worth encouraging systematic comparison of the efficiency of the diagnostic capabilities in large scale computer programs currently in use. This kind of investigations(though, unfortunately, not appealing to the academic environment) would likely represent a major contribution to safer computerised structural analysis, inasmuch it would stimulate the development of more advanced data checking techniques.

The third cause of concern stems from initial truncation or round-off errors. Nowadays this is not a very common cause of failure, because large computers have a large number of digits. However, the use of minicomputers spreading in small civil engineering design offices,is likely to modify rapidly the situation. E.g. large differences between axial and bending stiffness of frameworks are a common cause of ill-conditioning, which is usually diagnosticated by checking equilibrium at nodes. Program developers who do not automatically provide this check,should be censured. When more complex structures are considered, the diagnostic of ill-conditioning is much more difficult. The computation of the conditioning number is not widely adopted as it is relatively expensive and occasionally very conservative. The fourth and final cause of concern are discretization errors. Particularly in nonlinear ad for three-dimensional problems,the user is forced to limit the number of degrees of freedom to avoid prohibitive costs. It is usually stated that discretization errors can be controlled by mesh refinement. This is certainly true in two-dimensional problems, although many users prefer to adopt from the very beginning a conservatively large de-

grees-of-freedom number in order to avoid a second computation and the attendant delays. On the other hand, mesh refinement of 3-D problems is difficult to apply because of its overwhelming cost.

The above remarks show that control of discretization and numerical error is often based on engineering intuition only. Algorithms for providing automated error control are currently being developed at ISMES [3] . Preliminary but fairly extensive numerical results have been successful. If the possibility of inexpensive "a-posteriori" error controls will be confirmed,it is likely that future computer programs will make use routinely of this new approach.

As a conclusion, validation of computerized structural analyses requires expertise in structural mechanics, numerical analysis and software engineering. This suggests that it might be more appropriate to qualify the user besides the solution method.

In the absence of user's qualification procedures, the solution of complex analysis problems should be checked by a separate computation performed by an objective outside organization. This is already standard practice in the shipping industry.

4. CERTIFICATION OF SOFTWARE USERS

The user of structural software seldom coincides with the program developer, whose competence is indirectly checked by the program verification, or with the engineer, who is responsible for the outcome of the overall design process. The program user has to be responsible for the validity and the accuracy of the calculations he carries out. This responsability may often be legally attributed to a computer service bureau; however,the professional competence of individuals represents the crucial factor anyway. The stress analyst using computers as a normal working tool, is generally required both to know thoroughly the solution methods implemented in his programs,and to have a deep understanding of the physical theories on which those methods rest. Not only is he supposed to be familiar with the use of his programs and computing facilities,but also he needs an integrated knowledge of numerical analysis, programming techniques and structural mechanics, in order to fully exploit his software capabilities and possibly to modify and,occasionally, to further develop his computer codes.

How to check, certificate and enhance the user's competence in the above areas, is a problem which cannot be discussed without due consideration to the environment, both technical and social. In fact, any effective solution to such a problem necessarily involves a variety of ingredients and factors, some of which loosely connected with the sofware users themselves: the analyst's employers, the

customers who pay for the structural analysis or design, governmental agencies in charge of technical supervision and control, software producers, hardware suppliers, professional societies, universities, and finally, to some extent, even the general public.

As far as the environment is concerned, it appears useful, for concreteness and clarity, to refer here exclusively to two distinct national situations. One situation, examplified nowaday by the USA, is characterized by a leading role in technology, a large amount of activities in the specific field, a multeplicity of the above listed organizations, all acting under the pressure of strong competition, in a society with much mobility of manpower and readiness to changes and adjustments. The other reference situation (Italy might be cited as an example) is characterized by still limited, though growing, computerized stress analysis activities and specialist community, little adaptive educational institutions and governmental agencies, professional associations with marginal roles, a centralized and stratified society where traditions, stable aggregations of individual interests and pressing social problems affect the policy making processes in technical areas.

We believe that only in the former environment the professional ethics and competence of structural software users can be guaranteed through formal certification. In fact, an effective licensing program based on (possibly periodical) exams and registration, presumes a strong motivation and an active role on the part of at least three entities: the community of those whose professional status is being certified and, hence, protected; an institution apt to responsibly carry out the whole process in the general interest (preferably an engineering professional association); some legislative body capable to provide the legal framework. An official licensing system (parallel to the Professional Engineer Registration in use since decades) was recently advocated and debated repeatedly in the USA, [1 , 4] ; although still a controversial, issue, as far as we know, the trend is towards the implementation of licensing in a near future.

The aforementioned, far-reaching implications of a reliable certifying process makes it impractical, in the writers' opinion, for environments of the latter type, whereas insufficient information prevent the authors from expressing opinions on other kinds of situations, e.g. in Soviet Union.

5. IMPROVING PROFESSIONAL STANDARDS OF USERS

The environmental conditions which affect the prospect of certification and licensing programs, act in a similar fashion on the potential role of formal university education in enhancing the gene-

ral competence of structural software users. Although slowness of changes is everywere claimed to be a permanent attitude of academic institutions, a variety of independent and diversified engineering schools, competing with each other and actively interacting with the outside world, is clearly a factor in favor of prompt curricula adjustments to emerging needs. Infact, in the former (say American) situation depicted in the preceding section, structural engineering curricula have been significantly reshaped, so that e.g. courses on programming and computer methods, finite element analysis, applied approximation theory, have become normal offerings in most Departments. Moreover, in view of the future growth of large-scale computerized analysis in such sophisticated areas as nonlinear, transient or interdisciplinary problems, the prospects have been envisaged [5] of 7-8 years doctoral curricula without research connotation and special academic institutes for computer applications.

In the latter, less responsive environment mentioned in Sec.4, contributions from formal education to the improvement of stress analysts' competence are bound to be limited and delayed, but by no means negligeable. The reasons and possible remedies cannot be discussed here, for space limitations; some hints and details can be found in [6] [7] .

Very significant contributions to the same purpose, almost independently from environmental conditions, can be provided by continuing education. Training practitioners in new methodologies and disciplines, or updating and "brushing up" their technical and scientific bachgrounds, are educational processes obviously needed in times of rapidly expanding technology and underlying sciences. But the remarkable impetus recently gained in most countries by engineering continuing education and its tremendous potentialities can be explained by its peculiar features, like the following ones (see e.g. [8]): flexibility of contents and teaching methods, due to the extra-curriculum, informal nature of short courses; compatibility of these with professional committments of participants; self-financing; use of new teaching aids, such us videocassettes and CRT's for dissemination of carefully designed courses in the engineering environment; relatively easy interchange of lecturers, experience and documentation at the international level; natural involvement with mutual motivation of universities, research institutes, professional societies, design offices, industries, government agencies.

A measure of the potentialities of continuing education in the stress analysis profession, can be achieved e.g. by considering the important role played by short courses in spreading a knowledge of the finite element method among practitioners, most of which left university before it was formally tought.

At least two peculiar aspects of continuing education for structural software users appear worth mention here. The English wording "I hear, I forget; I see, I remember; I do, I understand" is especially suitable in our context; the implication is that workshops (and the facilities involved) have to be a substancial ingredient of continuing education for software users. The second aspect is related to the difficult issue of users certification: the easiest solution, in all professional and social environments, might consist of a formalized system of granting continuing education credits, based on "ad hoc" designed programs of coordinated and qualified short courses.

6. CONCLUSIONS

In view of its growing impact on engineering practice, computerized structural analysis has been considered herein from the standpoint of possible improvements of its confidence level. Practical prospects of progresses in this direction have been critically examined and found fairly promising.
Some of them (verification, qualification, validation) are of technical nature and concern primarily the structural software. Other kinds of initiatives(licensing, restructuring formal curricula, continuing education) are addressed to the users, have broader and somewhat controversial implications, and may be most beneficial in the long range.

REFERENCES

[1] BERMAN, I. (Editor),"Engineering Computer Software," Proc. Symp. ASME,1971.

[2] SCHREM, E., "Development and Maintenance of Large Finite Element Systems", Symp. on Structural Mechanics Computer Programs, University Press of Virginia, 1974.

[3] PEANO, A and RICCIONI, R., "Automated discretization error control in finite element analysis," Proc. 2nd World Congress on Finite Element Methods, Bournemouth, 1978.

[4] GRIFFIN, D.S."Certification of Structural Software Users", The Software User: Education and Qualification, Krans H. Ed.,ASME 1972.

[5] GALLAGHER, R.H., "Computerized Structural Analysis and Design The Next Twenty Years," Computers and Structures, V.7,1977,pp. 495-501.

[6] VILLAGGIO,P., "Mathematical Education of Engineers and Related Teaching Problems" (in Italian), Notiziario U.M.I.,5 Suppl.No.6 June 1978,pp.23-30.

[7] CERCIGNANI C. and MAIER G., "Teaching Mechanics in Italy" Mech. Research Communications, Vol.5,No.2,pp.105-112.

[8] SENIGER,H.A., FRANCIS,J.R.D., GOLLING,E., MAIER,G., HAVEMANN H.A. and LECAMUS R., "Continuing Education in Engineering-Aachen SEFI Symposium Working Group Reports", European Journal of Engineering Education, Vol.2., 1977,pp.31-45.

[9] ALMROTH, B.O. and BROGAN, F.A.,:"Automated Choice of Procedures in Computerized Structural Analysis", Computers & Structures, Vol.7, pp.335-342, Pergamon Press 1977.

Leere Seite
Blank page
Page vide

IABSE
AIPC
IVBH

COLLOQUIUM on:
"INTERFACE BETWEEN COMPUTING AND DESIGN IN STRUCTURAL ENGINEERING"
August 30, 31 - September 1, 1978 - ISMES - BERGAMO (ITALY)

Computer Aided Checking of Strucuural Safety

Vérification de la securité des structures à l'aide de l'ordinateur

Rechnerunterstûtzte Standsicherheitsprûfung

H. SCHWARZ

Professor, Dr.-Ing.
Fachgebiet Informationsverarbeitung im Bauwesen, TH Darmstadt
Darmstadt, Deutschland

Summary

The procedure of auditing the calculations of inner forces and dimensions of building structures, which is usual in the German Federal Republic, is criti cized. Instead of this procedure it is proposed to check the safety of a structure against failure by determining its bearing load considering the dimensions of its members as they will be constructed. Therefore designing engineers should be allowed to deliver data holders which contain detailed descriptions of all mem bers of a structure in a standardized form instead of giving evidence of its safe- ty by calculation. Auditing engineers should have software at their disposal which enables them to deter:nine the bearing load from these data.

Résumé

La procédure de contrôle officielle des efforts et du dimensionnement des struc tures, utilisée dans la République Fédérale d'Allemagne, est critiquée. Au lieu de cette procédure, on propose de vérifier la stabilité d'une structure en dériv ant ses charges de rupture de documents d'exécution. Par conséquence, on devrait permettre - à la place de la preuve de stabilité - de présenter pour vérification des supports de données qui contiennent la description de tous les éléments de construction dans une forme normalisée. Les bureaux de contrôl doivent disposer d' un système de programmes d' ordinateur, les rendant capables de dériver la charge de rupture sur la base de ces données.

Zusammenfassung

Das ist der Bundesrepublik Deutschland gebräuchliche Verfahren, die Schnittkraftermittlungs und Dimensionierungsberechnungen für Bauwerke amtlich zu prüfen, wird kritisiert. Statt dieses Vorgehens wird vorgeschlagen, die Standsicherheit durch Ermitteln der Traglasten aus den Ausführungsunterlagen zu bestätigen. Es sollte deshalb zulässig sein, an Stelle eines Standsicherheitsnachweises Datenträger zur Prüfung vorzulegen, die ausführungsreife Beschreibung aller Tragglieder in standardsierter Form enthalten. Prüfämter und Prüfingenieure sollten über Software verfügen, mit deren Hilfe sie die Traglasten der so beschriebenen Tragwerke ermitteln können.

In many countries the construction documents of building structures have to be audited before the erection of a building is licensed, in order to guarantee structural safety and thus to protect the public against the danger of a collapse.

Traditionally, together with the request for the building licence evidence of structural safety has to be given in a verificable manner, in order to facilitate the procedure of auditing.

Usually, for this purpose the designer presents the calculation by which he determined the inner forces of the structure and the dimensions of its members. He really could not be forced to do so. Instead of this he rather could determine the ultimate inner forces of each member, and then give evidence that the whole structure is able to bear ν times the working and dead load of the building, where ν is the safety coefficient.

The auditing officials or the consulting engineers to whom the licensing authority might delegate the auditing work have to check whether the evidence is complete and correct.

If, instead of a special evidence of structural safety, the designer presents his calculation of inner forces and dimensions, the auditor inevitably does not only check the results, but also the procedure of the designer's work. Thus his attention will be directed to the numerical correctness of the calculation and diverted from the real structural safety.

For this reason - in the German Federal Republic - more and more instructions for the adequate manner of calculating inner forces and of dimensioning have been incorporated into the official building codes. The designers working conditions have been affected and - at least partially - narrowed by these regulations.

When computers were introduced into structural design work it was generally presumed that furthermore calculations of inner forces and dimensioning should remain to be objects of auditing. In the "Preliminary guidelines for setting up and auditing electronic computations of structural safety" from 1966, which are still valid now, there is prescribed that the author has to indicate the structural model, upon which he bases his calculation, and to describe in detail the computer program which he uses.

Proposales of alternative procedures as "parallel calculation by use of an independant computer program" or "auditing by use of intermediate results" suggest the auditor to deal intensively with the designers assumptions and with the results of his calculation rather than with the designed structure.

Consequently there are continued efforts to standardize the procedures of structural design work in order to facilitate the auditing work. Demands like that for "computer adequate standardization"sometimes are based on the idea the whole procedure of dimensioning, the calculation of the inner forces included, obligatory should be prescribed for important, regular cases.

In three different respects this development appears to be disadvantageous to the individual engineer als well as to the whole profession:

a The creative energy of designing engineers is exhaustet more and more by observing the increasing amount of regulations.[+] They remain fixed to the analytical procedure of dimensioning, though the use of computers should have liberated them from mechanical computing work.

b The amount of documents necessary to give evidence of structural safety and therewith the expenditure of time and labour for auditing work is growing more and more. The difference between the designer's and the auditor's expenditures diminishes, while the charges for auditing calculations remain much lower then those for establishing them.

c It grows more and more difficult to make sure, that the auditing procedure is totally independent from the dimensioning one. Therefore, in spite of the increasing expenditure for auditing, the risk also increases, that serious deficiencies of a structure remain undetected.

These are the reasons for my proposal to disconnect the evidence of structural safety totally from the procedure of dimensioning.

The auditors should have computer programs to their disposal, by which they are able to determine the bearing loads of structures from the dates which they understand immediately from the construction documents. They should use these programs without to know the assumptions the designers had taken while dimensioning the structure.

The designers could be exempted from giving evidence of the safety of their structure in case they deliver to the licensing authority not only the construction documents but also data holders which contain detailed descriptions of all members of the structure in a standardized form.

If these data holders could also be used to produce drawings and other construction documents automatically, the expenditures of the designers as well as those of the auditors and even the total amount of paper which has to be interchanged between designing, auditing and constructing organizations could

[+] Alfred Mehmel warned already in 1965 structural designs would remain sums of observed regulations only.

be decreased.

In favour of the standardization of those data holders all efforts should be canceled, to standardize the procedure of dimensioning structures and proving structural safety. Existing regulations on these procedures should not be obligatory in case designer and auditor agree about the interchange of data holders.

The previous critical remarks and propositions are based on the problems, we actually have in the German Federal Republic. It would be of great interest to learn in the discussion, whether in other countries similar problems exist or not, and if more effective procedures of proving structural safety are practised.

III SESSION

DISCUSSION

September 1, 1978, Morning.

Chairman: BLAUWENDRAAD (Netherlands)

BLAUWENDRAAD - I think now we can open the discussion to your contribu_ tions and comments.

KLEMENT - When you have made programs, you know that not only the tech_ nical aspect of the programs, but also the work which you have to do until the program is tested, gives you the right to say: "this program is mine". If you document the way in which the results are made in a mathematical form, you give what the user needs. You do not need to give full information about the inside working of a program to let a user be able to use it in a perfect way, and not always there is some progress in making programs again.

BLAUWENDRAAD - I think this is for Mr Alcock.

ALCOCK - Of course, I understand this, because my organization lives by sel_ ling software systems. For example, we have a system that is a version of STRESS, and I think there are about thirty copies already sold but we do not protect the internal documentation; we are quite happy for it to go out. To the best of our knowledge, nobody has stolen it yet, and it is just more convenient to pay the knowledge, nobody has stolen it yet, and it i s just more convenient to pay the price of this, for the person who knows its inside and can support it, but at least when they make mistakes, they can see what it is that the pro_ gram is trying to do. Furthermore, they can adapt it to their own purposes and add to it, and so on. One more point on this : I am not, in fact, necessarily pro_ posing that if you write in FORTRAN you will provide the FORTRAN. What I try to present or just to suggest is a notation which is half way between the mathematical description and, say, the FORTRAN realization. We have done it for this program FORPAR, and what you buy when you buy FORPAR is not necessarily the FORTRAN. If we have that, it will cost another sum of money, but what you get is an exact description of what the program attempts to do, and one buyer of this can provide to make his own realization of it. Someone has adapted it for interactive use; some have put it into back use, but the point is that there are many installations on many computers, and this has been done by means of this intermediate notation and not FORTRAN. Thank you.

PFAFFINGER - I would like to make two brief comments about the responsibi_ lity. Today, for almost all the programs, you have an agreement with the de- veloper of the program that is not liable in any respect for anything that might

happen with the program or the results of the program, and the same is true for the data center. If you work on a data center, you expressly have to accept the fact that the data center will not be liable for any program they use there, or any result which might be wrong. So, I would like to stress this point : even if from a little point of view, you are forced to check your results. The second comment I would like to make is this: I strongly support the opinion of Prof. Klement, that it is really not necessary to know every detailed documentation and every detail that are in the program. As a matter of fact, we see the development of something which I would like to call: "The Fortran Industry", and there are people making their living on developing software. Software is much protected from a legal point of view. We have to know the basics of the procedures; we do not have to know all the details, and we are just glad if we are in a position to check what is coming out of the program.

ALCOCK - To explain shortly on the legal side, I do not know the laws: I think that in all the countries they are different, but I know that in Britain I am not allowed to put on the front of my car: "The driver of this car is not responsible for anyone standing in his way".

HAAS - I would like to comment on the contribution of Mr Alcock, concerning documentation of programs. I think we have to distinguish between the part which deals with the stress analysis and the part which handles with design forces. I think that the first must not be so well documented and it can remain a black box, whereas the second, which handles with design forces, must be clearer. The design forces are open to interpretation, and it must be said very clearly how to interpret them and how we handle the results, how we get them.

DEPREZ - I would like to comment on Mr Uherkovich' s paper about professional needs. I think the main problem is that the computer permits to all designers to all consulting engineers to think thay can solve any problem.
We can solve it in two ways: first, we can ask the user to have a licence for this, but I think we have another possibility. We must know exactly what we can do, what we can get like guarantee. Now, to precise correctly the responsibility of the computing center or the consulting engineer. Everyone can choose a specialist in the structure he wants to compute, or a consulting engineer specialized in the use of computer in this particular field. For instance, we can say in dynamic problems of large structures, there are no many organizations in Western Europe which are able to do it, and we know that somebody has this kind of problems and goes to the computing center to try to have a solution. If there is no previous experience in this field, I think there are a few chances to reach a good solution. In the ethic of the profession, in my opinion, it is important that the consulting engineer knows that he is unable to solve this problem himself. He must not try to obtain from the computing center the information to solve it, but he should have to go to consulting engineers who are specialized in this field.

TAFFS - Sometimes you treat the computer just like you were treating an engineer or an assistant. When we talk about this possibility, to delegate someone to carry out a part of the design, we do not feel we are giving a part of responsibility for that design to another person. We automatically say: "The subordinate will make a mistake without our guidance and he is perhaps the judge". We can give this subordinate some guidelines and we can ask him to follow them, but we will expect that the subordinate automatically is able to follow to the letter or we will check. If we look upon the computer like a subordinate, I think it can help very much in clarifying which the areas of responsibility are. Another important point of discussion could be the use of a particular system, when you are forced to assume it by the client.

TOMINO - If we are talking about the responsibility, whose is the responsibility in an engineering society ? We are responsible and the designer too, this is obvious. But if we are talking about the computer programs, any computer center has in contact a limited responsibility. It is always said in a contract that if someone uses a program, the computer center is unresponsible at any degree about results. For example, if we give some program to some user, we say in our contract: "We are responsible for the maintenance, but we are not responsible to any extent about results coming from that kind of program". That is our current practice. Then the question which comes up to my mind is again: "Are we talking about avoiding such kind of catastrophe which may occur to the practical engineers ?"

MILSTON - I am very interested in Dr Gallico speaking about a failure which has occurred with some computer design on dam operations. I would like him to amplify about this failure, presumably due to a computer program.

BLAUWENDRAAD - Mr Milston wants to have a 'scandaleuse rubrique', so you are cordially entitled to contribute.

GALLICO - This failure ... is a typical "case history" and very serious. In fact, it involved about 140 millions of dollars of loss. Luckily, no people was injured, but anyway, as far as money is concerned, it was and still is a big affair. This happened in South America, and it was reported in the various papers. It was a system of a hydroelectric power plant. The entire operation of the system was designed by engineers and the input was given according to meteorology, hydrology, operation energy conception, in order to optimize the use of water.
This was translated into a program and centralized in a despatching center which was completely responsible and had the best trust in model idealization of operation. Everything was arranged for several years, as far as I know, because we were called - and we are still called - to give our opinion.
It happened that the local operators were unable , and hadn't the legal authorization or the technical skill, to discuss the orders given by the dispatching center. Nearly one year ago, the inflow was a little bit different from what

they had forecast, and the local operators advised - informed by telephone the dispatching center, that there was something wrong, not matching with the anticipation of the operation. The dispatching center did not discuss that at all. They only said: "You keep silent, obey and do not discuss". But the local operators insisted and it was a matter of three days, two or three days, they insisted saying that the anticipation of the program and the model were not covering this same case. Then, the responsible personnel was convinced that there was something wrong in the model, but it was to late to operate. So, they decided to close a power station and open a discharging canal. The water was too much and overflowed the arch dam, destroying it. At the same time, power went out of service, the flood came to a second dam and destroyed it; then it entered into the power station. Now, this case is dealt on the trial and I cannot give at the present time what will be the result of the analysis, or what happens. I would not like to give the impression that the mistake was on the computer, nor in the program. This case is an example of insufficiency and blind trust in the results which, sometimes, are not covering all possibilities.

TAFFS - Often it is not the computing of the program, of course, which is at fault, but the designer, the human element. One experience we have had, which was very painful, where there was an error in a program used very seldom at that time. If the failure is embodied within the program, some social trouble can result from it. The program we used was well accepted, many of the normal structures for which the program was designed have run successfully through. Something wrong happened in one case: if the program had been well tested, the fault would be detected at the time, but testing a program is something we cannot cope with. The cost for testing a program is out of proportion to the cost of developing the program and the cost of development, we know, is already exorbitant. We have gone up to the point of spending up to 2/3 additional cost. We had another case where we are analyzing a primary structure, a very important structure, and we could not understand the answers coming from the computer. We sent the information to three researh centers, for their advice: in each case we had a reply which did not completely explain how was such a complex inter-relationship within the various parts of the building. Luckily, the project engineer insisted on this investigation and we discovered there was a great mistake in the program.

DUTERTRE - Just about program testing, if you have had a program which you have used for one year or two years, and if you think you have tested when you try to commercialize it, then you have a big surprise, because it is tested in your firm, with people thinking and doing the same way. When the program goes outside and you have fifty persons, fifty different firms doing things in fifty different ways, only in this way we can say the program begins to be really tested, because the best testing procedure is: "The more people use a program, the better the program becomes."

HAAS - Sometimes there is the trend of treating programs like a human being. But people cannot say: "That is your program, that is your baby and you have to look and bother that it works well." Then we say: "Yes, in our opinion the program works well, but if the program does not do completely what you want and makes some mistakes, having or buying it, you must check and take the responsability that it works well".

GALLICO - I am a little worried at this time because I heard there may be an error in the program even more than one time. I don' t agree : so, again, there is a misunderstanding. When in our office our young engineers use the Hewlett Packard, a little computer, I agree completely, because it is manageable, little, let us say they can face quick problems and that is all. When people have a big problem - I mentioned before Kalayaan power station in the Philippines, something very complicate static structure, or the structure shape, underground construction, etc. rock characteristics not well known - we rely completely on the program made by specialists . We keep our responsibility, we take the responsibility of the design because we know that these gentlemen probably can explain better than I what is the responsibility of the designer or the consulting engineer - but we assume that the computing center gives us and develops results using 100% tested programs with no mistakes.

SHIMADA - Luckily you keep your possibility to do a lot of equilibrium checks and you always can do some more check calculations to find the order of the displacements, and so and so.

KRUISMAN - I would like to make a comment on errors. I think it is good to start from the statement that all is right, what has not been proved to be wrong. The only way you can approach what is wrong is to get something like a "common opinion" and that again is a play sitting together and exchanging experience on programs and so on, and then you get from all the new users who check and find the errors in the program; then you get an idea of what Mr Alcock also mentioned: the evaluation of several programs that has, for instance, been done in the Dutch Association. There are several groups, say groups of programs, that evaluate and exchange information about them. I think it is the only way to come to the statement that something wrong is in the program. You never can prove that the program is right, because we do not know what is right.

PFAFFINGER - This philosophical statement is challenging, I think.
I put a thesis and this thesis is: unverified calculation is wrong until it is verified. To my understanding, it is the best approach, because we know by experience that our tools often are insufficient or inadequate and we do not know what is right in many of the date problems that we are solving. For instance, let us consider linear elastic analysis: the basic of this analysis is well-established, and we have enough data to verify a problem. It is a different story if you do highly sophisticated dynamic analyses of something that no-

body has done before. There I agree: we do not know what is right, but in the other cases, we know what is right and I think we have to verify our data assuming they are wrong, the result are wrong, and we have to verify and prove that they are right.

KRUISMAN - Well, you say that the point is that a linear elastic analysis is proved to be right. It was about 15 years ago or 20 years ago that a lot of English planes came down and that was the start of a deep research on mechanics; up to then we did not know that the problem existed. There is a lot of things we do not know, because we never experienced them. This is the reason why I say that unless it is proved to be wrong, it is right. Of course, you can contradict that. Is was a large contradiction, I think, in the country of Prof. Klement at the beginnging of the century, whether you approach the problems from the positive side or the negative side.

DUTERTRE - About right and wrong, I would like to quote a statement from Prof. Newmark in 1965, in a Soil Mechanics symposium. He said : "All the formulas are wrong; however, they are wrong in a consistent manner". The problem with computer is: "they are wrong inconsistently, with no coherence in the way they are wrong".

VOS - The serious face of the question I think that is the point of view of the designer. I should really have looked at this way. Every program, even the most universal programs are written by a first committee who decided on them, considering a particular constructuion in their head. Therefore, I think you should not talk in terms of 'right' and 'wrong', but of a program being fit for certain constructions or for certain problems; and then at a certain moment, when you put a new type of construction in such a program, you can say: the program was not fit for that construction, or the model was not fit for that problem, and then - of course - it was wrong.
I think it is better not to speack about right or wrong.

KLEMENT - You see, we are speaking here about statics, and static is a very reliable thing if you compare it with other parts of the technical calculation. When I was in charge of a computer center, I had a lot to do with tunnel calculations and designing boilers of a size which have never been built before. The mathematical models you do are much more away than models in static problems, but by means of a mathematical model, you get at least a rule from which you can, afterwards, find in what way results are away. Before, people had calculated boilers with two days manual calculation. But, if this grows up to very high dimensions, they found that you must take mathematical models, so the computer gives a possibility to have at least a scale for comparing the results afterwards and to make better mathematic models in the future. Therefore, I think we cannot say "results of a calculation are wrong" if they are not related to their building. I say: "all our calculations are only calculations to get the correct dimensions", but nobody believe at the stresses you have calculated. You will always find that the stresses are very different from your calculated stress.

DUTERTRE - I perfectly agree. What I menat when I got the first sentence by Prof. Newmark can be applied to any model in computing, providing that the program does work right. Now, when I meant the "erratic error" is from a program error, but sure, when you build a model, the model is wrong, it is always wrong but it is consistent, it is a way to move.

BLAUWENDRAAD - There is time for other questions, if you like, Prof. Werner.

WERNER - Something about the errors : I think we are talking now about the program errors - errors in the source code - but I think there are many other possibilities of errors. First, not the errors between the user ' s han book and the code. If you change the code, it is often forgotten to change the corresponding item in the user's handbook; for instance, the description input is often not in direct relation with the code itself. The second, we detected some errors in the standard itself. The standards often are proved by normal applica tions. When you are writing a program, you must take into account a broader area, and you have to program up these standards and often in these standards it is not thought about the various applications, and sometimes there are errors in the standard itself.

BLAUWENDRAAD - I think you are raising a problem on which it is worthwhile to have a colloquium as we had here for three days. You can feel that the recent building codes of practice have not been written with the use of computer in mind. So, you even have to restructure them at all, taking use of the modern possibilities of decision tables and things like that.
I think it is a rather big problem.
Is there any other in the audience who likes to comment on this subject or on another one ?

ADLER - I am busy in Switzerland in an office of consulting engineers. I had to point this, so that you see clearly that I am just a user of software and not a producer. It is 100% clear for me that we must check all programs we get, and it is on our own responsibility, somehow I do not feel it so good. Software men say in a strict way they are not responsible for what they give out. I do not know. I would like to here someone else about this point. I would like to know whether there are some other profession men who dispense themselves from their job or work they sell or not.

UHERKOVICH - I think in many countries this label is not valid, legally not va lid. You can make ten labels " I am not responsible"; in the real case, you will be.

PFAFFINGER - In the contract, you can exclude your liability. If it comes to a case and the lawyers can prove your gross neglecture, you will be liable, that is the case, but usually the lawyers try to exclude everything.

HAAS - I have a question which has something to do with money, because we cannot check the program so that we weed all mistakes out. Such program would become very expensive and almost nobody could afford its use, because

we have not so many applications of our standards that we could consider, let me say, 100 times, than we can put such an effort in our programs and we can check them to a higher degree as we do now.

DEPREZ - I think there are two kinds of responsibilities: one kind of responsibility is that of means and the other is that of results. The consulting engineer has the responsability for results; he must provide a good design. The responsibility for data processing center is that of means; it must be careful using the program and it must supply the engineer with good documentation as well. In the lawyer's courts, there is difference from country to country. In some countries, you can limit your responsibility to results, in others - like in France and Belgium - you can never limit your responsibility to results, but for your responsibility in means you can always do it, because this kind of responsibility does not engage the results themselves. You are responsible only for negligence or not good work. And this is why I think it is important to make engineers and software centre know correctly what is their own responsibility.

BLAUWENDRAAD - If I may interrupt, you have said before that you see no responsibility for the data processing center about the results. But when the user has no knowledge of your program, you said he should go to some advanced consulting engineer. Who tells the way he should do it ? For the everyday practice, I can understand that the user will have knowledge enough and so the data processing center needs no responsibility, but what about a very small number of advanced programs we use ?

ALCOCK - Well, what about the small beam or something ? The problem is almost completely automated. You have to put the overall dimensions of the building and the computer output and the drawing out come . That situation is with us now and it means that we, engineers, are actually using that. He is legally responsible, because he is the engineer. But he may not have witness; his bureau has probably no witness. There is a third part involved and clearly the responsibility concerns people who actually produce software. For this reason, the only hope of ever clearing the matter up is that the inside is exposed to those who are able to read the signs.

VOS - We really must distinguish between liability, which can come to court, and responsibility, which is used in many senses here, also in the sense of engineering ethics, moral responsibility. Thinking of design practice and what does the designer : when you are really in doubt of using a certain program, you only have to use a computer program and just look if the problems you have now, have been solved. I think this may be a good practice.

HAAS - One word on separating the responsibility in legal and moral responsibility. Of course, each software producer should feel responsible in this moral kind of responsibility and should give correct programs as quickly as possible.

BLAUWENDRAAD - Thank you. We would close now.
In private discussions, you can go on.
Thank you very much for your contributions . You made my job very easy .