Computers in structural design: some general thoughts

VIIa

Computers in Structural Design: Some General Thoughts

Emploi de l'ordinateur dans le dimensionnement des structures: quelques considérations générales

EDV-Anlagen in der Bemessung von Bauwerken: einige allgemeine Bemerkungen

DONALD ALCOCK
Alcock Shearing and Partners
Redhill, Surrey, England

SUMMARY
This paper deals with general problems facing structural designers who would use computers; the opinions expressed are subjective ones. The structural designer should be able to program simple calculations himself. To make the task agreeable he should be able to type; to make it rewarding he should reject facilities that are not Standard. When considering programs developed by others, the designer should refuse to use systems that would usurp his engineering judgement unless he is perfectly clear by what criteria the decisions are made. The possibility of publishing descriptions of such programs in an intelligible form is discussed.

RESUME
Cette communication traite des problèmes généraux qu'affrontent les ingénieurs projeteurs qui voudraient se servir des ordinateurs; les opinions exprimés sont subjectives! L'ingénieur projeteur devrait être capable de programmer lui-même des calculs simples. Il devrait aussi savoir écrire à la machine; il devrait enfin renoncer à tout équipement qui ne soit pas standard. En considérant les programmes préparés par d'autres, le projeteur devrait refuser de se servir de tout programme qui lui enlèverait son indépendance de jugement comme ingénieur – à moins qu'il ne connaisse exactement les critères selon lesquels les décisions importantes ont été prises. La possibilité est également envisagée de publier des descriptions compréhensibles de tels programmes.

ZUSAMMENFASSUNG
Dieser Artikel behandelt allgemeine Probleme, welche bei der Benutzung von EDV-Anlagen durch den entwerfenden Ingenieur entstehen. Die vertretenen Ansichten sind subjektiv. Der entwerfende Ingenieur sollte fähig sein, einfachere Berechnungen selber zu programmieren. Um sich diese Aufgabe zu erleichtern, ist es von Vorteil, wenn er die Maschinenschrift beherrscht. Damit die gestellten Aufgaben wirtschaftlich gelöst werden können, sollten nur standardisierte Anlagen benutzt werden. Um eine fachmännische Beurteilung zu gewährleisten, sollten vom entwerfenden Ingenieur nicht selber entwickelte Programme nur verwendet werden, wenn er gründliche Kenntnisse über die dabei getroffenen Annahmen hat. Die Möglichkeit, Beschreibungen solcher Programme in einer verständlichen Form zu veröffentlichen, wird erörtert.

1. PROBLEMS FACING DESIGNERS

It is difficult to classify computer applications in structural design because of their interdependence. For example one category might be "Small ad-hoc programs" and another "Large general purpose systems", but in such a classification a typical BASIC program could belong in either category, being an entity in itself as well as a set of data for a large BASIC system. Nevertheless a crude classification of some problems of structural design amenable to solution by computer is attempted below to serve as a framework for the paper.

- simple calculations such as can be programmed ad-hoc by a structural designer who is not an expert programmer

- analytical calculations of a kind so frequently encountered by structural designers that it is worth employing programs designed for general use and developed by professional programmers

- complicated analytical problems needing numerical formulation by applied mathematicians and which may or may not be amenable to solution by existing systems such as those for finite-element analysis

- problems of sizing and synthesis often referred to by structural designers as "design"

- specialized tools for automated drafting, word-processing, estimating, scheduling, information retrieval, and other tasks for which proprietory systems are available, some of which include both hardware and software as a package.

This classification excludes the many applications of computers common to all professions such as job costing, payroll, invoicing and so on.

2. SIMPLE CALCULATIONS

Not long ago every structural designer was expected to be proficient in using a slide-rule and electro-mechanical calculator. Nowadays an electronic calculator is cheaper than a good slide-rule and more powerful than a mechanical calculator. Furthermore the cost of the electro-mechanical calculator was higher (in real terms) than that of today's "personal computer" which is able to compile and execute programs written in a popular programming language. In other words the slide-rule is dead and there is no economic reason to prevent a structural designer using a computer.

This discussion refers to "personal computers" but the arguments apply as well to the use of a terminal connected to a large computer.

There are many and varied personal computers on the market but most have two things in common: the ability to process programs written in BASIC (or a language similar to it) and a typewriter keyboard by which to convey such programs to the machine. There is universal enthusiasm for teaching children and older students how to write programs in BASIC but apparent reluctance (at least in the author's country) to face the problem of communicating with the machine physically.

Typewriter keyboards have had the traditional QWERTY layout since about 1890. For nearly a century millions of people have found such a keyboard efficiently suited to transcribing thoughts to paper using ten figures acting under the sense of touch. Nowadays the same keyboard is used to convey programs to

computers. This would suggest that anyone who wanted to communicate with a computer would first learn touch typing, for although journalists may pick up amazing speed and accuracy using only three fingers, no secretarial college would impose such acrobatics on trainee typists. Yet the obvious advantage of touch typing seems to be ignored among computer users. Worse! Computer codes are devised which permit abbreviations of words *so as to save on typing!* No trained typist would find LIS DAT, PRG FIL easier or quicker to type than LIST DATA, PURGE FILE, nor would anyone trying to read and understand what had been typed enjoy the cryptic version.

The universal language of the personal computer is BASIC, originally devised as a simple code for traching the elements of programming to beginners. It proved so popular that it has been imitated and adapted in various ways by different computer manufacturers and others. Now the response to the statement "My program is written in BASIC" is likely to be "*What* BASIC?". Extensions (some of them wild) are being made at a rate outstripping the formulation of Standards.

It is useless to bemoan the inadequacies of BASIC: the impossibility of structured forms, dependence on a fixed set of global variables, absence of integers, *etc.* because BASIC is in popular demand and there is no way (in some countries at least) to impose a "better" language on the populace.

Faced with programming in BASIC the structural designer would do well to apply self discipline. He should read the relevant Standard (ECMA, ANSI, or whatever) and go through the BASIC Manual for his own computer, ruthlessly crossing out non-standard facilities however convenient or exciting they may seem. His reward comes when he finds he can sell his programs to someone who uses a different make of computer, or when he has to use a different computer himself.

At the time of writing, the approach advocated above is not fully realizable because no Standard has yet covered such facilities in BASIC as the MAT statements, yet these are implemented with reasonable consistency in many versions of the language. The author has tried elsewhere to summarize such "de-facto" standards so as to encourage "portable" programming [1].

3. ANALYTICAL CALCULATIONS

A structural designer need no longer *be* a computer to determine the elastic distribution of moments in a frame. To initiate such an analysis he has only to write or type data in the form described by a user's manual, then let a standard program do the rest. Because of the demand for skeletal and finite-element analysis, programs and suites of programs have been developed by teams of computer-minded engineers and professional programmers, and offered to designers on a leasing or royalty basis.

Some of these programs are sound and reasonably free of errors; others are notoriously unpredictable in behaviour especially when transferred to new computers or when there are changes to an operating system.

A structural designer who uses such a program is seldom permitted to study its internal documentation, let alone make changes or extensions. The supplier argues that a complicated piece of software has to be "maintained" by a team of specialists who must be the only people allowed to look inside. Yet the conditions (which a potential user has to accept) stipulate that the supplier of the program takes no responsibility for damages caused as a consequence of wrong results.

This apparent dilemma is not as bad as it might seem as long as a program is *analytical*, in which case its results are essentially right or wrong and may be compared with those produced by a competitor's program. Confidence is thus gained or lost as the case may be. Nevertheless it would be better if structural designers were permitted to know more about the programs they use, and in the author's opinion it becomes vital when programs are used for synthesis. This is discussed later.

4. COMPLICATED ANALYTICAL PROBLEMS

Among academics the advent of the computer stimulated the formulation of engineering problems in matrix and other numerical terms; the necessary "number crunching" being no longer impractical. Many such problems are of interest to structural designers, but few structural designers have the mathematical expertise to formulate solutions themselves. Accordingly they consult experts at Universities and research establishments where large computer installations are often used.

If a problem is difficult to formulate mathematically it does not follow that it must be difficult to program. An abstruse problem resolved as a sequence of matrix operations might be programmed very easily. On the other hand a more simple formulation may demand the expertise of professional programmers before a practical program can be realized. A solution formulated mathematically becomes a *computing* problem when the required volume of input data or intermediate data is large, when iterative techniques are employed, when numerical accuracy is critical, and so on.

There is no reason to suppose that an academic who is an expert at formulating engineering problems in mathematical and numerical terms is *ipso facto* capable of writing foolproof computer programs. The sad experiences of computer bureaux when trying to adapt "University" programs to practical use testify to an unfortunate gulf between the disciplines of mathematical formulation and sound programming.

Computer science has not long been recognised in academic circles as a worthy discipline in itself, and many structural designers are not aware that it has much to offer their profession. In the present state of the art the structural designer should appreciate there is more to the numerical solution of structural problems than mathematics and self-taught fluency in Fortran. But there is nothing he can do to ensure that a program originating from a University or research establishment will be subjected to any form of quality control. Some programs are developed with the very highest degree of professional competence, but some are not.

5. SIZING AND SYNTHESIS

Analysis by computers is an essential part of structural design, but sizing and synthesis by computer is having more impact on the profession. It is now commonplace, for example, to let a computer program choose certain dimensions of reinforced concrete structures.

This practice poses the problem of responsibility for disaster. If such a program is developed and used by a structural designer then responsibility for a subsequent structural failure is obviously his. But if he accepts the results of a program developed outside his own organization he is allowing other people to design his structure for him. And it would seem that ever more structural designers are *willing* to take responsibility for other peoples' designs in this way.

Because these programs have to work according to Codes of Practice it might be supposed they are easy to test and evaluate. An evaluation of this kind was indeed attempted by the Design Office Consortium [2] revealing enormous differences of interpretation of a single Code of Practice among the programs tested. From users' manuals it was not possible to know in advance how each program would interpret the Code of Practice. How, then, is a structural designer to know what program to trust? At present he cannot.

There are two ways in which the situation could be improved:

- by trial and evaluation of proprietory programs carried out by some agency with power to award "Seals of Approval"

- by owners of proprietory programs disclosing in comprehensible form the internal descriptions of their programs.

Whereas the first approach can be made to work in specialized areas (for example by the Highway Engineering Computer Branch of the Division of Transport in the United Kingdom) the second approach may have more general potential and is considered in more detail below.

Program development is often undertaken piecemeal; an original version being augmented as demands for extra facilities arise and as new ideas occur to the developer. This style of development is *encouraged* by computer manufacturers and bureaux who provide software tools for interactive editing, selective tracing of execution, and so on. Although such facilities are difficult for a programmer to resist, the problem with the approach is that a program matures without documentation, and its "listing" might be unintelligible to a potential user even if he were allowed to see it. By adopting the more responsible approach advocated below, documentation automatically precedes testing and is, furthermore, intelligible to a potential user.

To be intelligible a program should be described in a notation less cryptic and less specialized than a computer language but more concise than a natural language. It is not obvious exactly where between these extremes the ideal notation should be pitched but an attempt has been made to define such a notation. It is called 3R and has been used to describe a program for which the internal description has been published [3]. From the published description "realizations" of the program have been written in Fortran, APL, and BASIC. The largest system so far described in 3R has a Fortran realization of some 20,000 statements [4].

Experience with 3R convinces the author that it is both possible and desirable to use such a notation for describing any program for technical application before encoding it in computer language. This approach avoids the problem of programs being developed without documentation - hence avoids the likelihood of their algorithms being intelligible only to their authors (and then only for the duration of the development period).

Even when thorough documentation does exist it may be argued that internal descriptions should not be disclosed outside the author's organization because the secrets are valuable capital to be protected. But against this it can be said that the potential user or buyer is more likely to adopt a program for which the internal description is made available than one for which it is not. Jealously guarding internal documentation may (it is hoped) become a short-sighted marketing policy for program developers. "Open" documentation leads to wider use of a program, so developers stand to recover investment from wider sales at cheaper prices - despite the occasional breach of Copyright that such openness might invite.

6. SPECIALIZED TOOLS

Dependence of structural designers on large installations operated by computer bureaux diminishes steadily as small machines become available. The price of a typical desk-top computer is about the same as that of a family car, and on such a machine a structural designer may run programs covering many (if not most) technical applications in his field.

There are, of course, practical limits to running design-office programs on a small computer; limits on size of program, volume of data, speed of processing. In some cases the designer may still need access to a big computer, perhaps using his small one as an intelligent terminal.

But because this modern electronic machinery is so small and cheap it may economically be dedicated to particular tasks of which a few were named earlier.

An example of a task of particular interest to structural designers is automated drafting. A typical drafting system has a dedicated computer controlling a digitizer for input and flat-bed or drum plotter for output. It may also have tape drives from which to read data generated by a general-purpose computer.

A system of this kind is sold as a package comprising both hardware and software. And, as was predicted a decade ago, the value of the software content is beginning to overtake the value of the hardware.

The structural designer may look forward to having ever more specialized tools at his disposal, each comprising a dedicated computer running dedicated programs. As long as the specialized tool is not designed to take over any of the tasks to which he should apply skilled judgement as a professional engineer he need not be concerned about the details of the software which controls the specialized tool. If the tool does not work properly it can be sent back to the supplier. But the structural designer would be ill advised to adopt the same attitude towards any specialized tool that might usurp his judgement as an engineer. Before using such a system he should know precisely what criteria and algorithms the machine uses to "design" structures.

7. DETAILING

A task that consumes a lot of time both of the structural designer and his draftsman is "detailing". In reinforced concrete work this includes both the choice of concrete dimensions and the selection of bar sizes and their placement; in steelwork it includes the choice of section sizes and details of connections. Detailing falls into the last two of the author's categories and is therefore discussed separately.

Detailing is heavily prescribed by Codes of Practice and might therefore seem to be a task that could be automated completely, making possible a specialized tool comprising items of hardware such as those used for automated drafting systems. The fact that such a tool does not yet exist seems to indicate that detailing is not as simple as it may at first appear. Certainly the evaluations undertaken by the Design Office Consortium [1] suggest this is so because every program chose a different pattern of reinforcement for the same beam. Codes of Practice - however restrictive they may seem - allow enormous variations in interpretation.

Perhaps our "built environment" will retain more interest if detailing is allowed to remain a minor art, but for those preferring to strive for efficiency

the problem could be resolved if future Codes of Practice were written to include *algorithms* as well as constraints. The algorithms could then be transcribed into computer languages, thus encouraging automated detailing.

The problem of detailing can be circumvented altogether by the use of "system building". No such system is complete today without a computer program with which to plan a building on a grid displayed on a screen. All schedules, drawings, documents necessary for construction emerge untouched by human hand. But it pleases the author to reflect that the slightest unusual requirement such as a skew gable or a step in the level of the ground makes many such a system collapse.

8. IN CONCLUSION

The author sees a widening gulf in the use of computers by structural designers; on one side the development of ever more complicated systems: on the other a phenomenal increase of "do it yourself" programming. Developments on both sides of the gulf have one thing in common; how the computer reaches its results is becoming increasingly obscure to all but those who wrote the programs - and often to them as well.

Structural designers can have little influence on developments in computer hardware or general-purpose software because structural design represents such a small market. But it should be possible for them to keep their house in order by adopting policies and attitudes some of which are discussed in this paper. At risk of being judged both trite and arrogant the author dares to summarize his subjective conclusions as follows:

- a structural designer who intends to use a computer frequently should develop some skill in communicating with the machine via its keyboard

- a designer who writes programs in BASIC should take care to use only the forms and facilities of that language that are Standard (and this applies similarly to Fortran programming)

- a designer who uses a proprietory system for analytical calculations should satisfy himself (by comparisons if necessary) that the chosen system is sound

- a designer should not assume a program is sound simply because it is based on a demonstrably sound methematical model

- a designer should refuse to take responsibility for "designs" generated by a computer program unless he fully understands the processes by which such designs are synthesized.

Finally the author believes it is both possible and desirable to make the working of computer programs comprehensible to potential users; the ideals expressed above are attainable.

REFERENCES

1. Donald Alcock: "Illustrating BASIC", Cambridge University Press, 1977.

2. Design Office Concortium: "Computer programs for continuous beams - CP110", DOC, Guildhall Place, Cambridge, 1978.

3. DOE PSA/LAMSAC/DOC: "FORPA Computer Program", Design Office Consortium, Guildhall Place, Cambridge, 1978.

4. S.I.A. Limited: "NUSTRESS", SIA Ltd., Ebury Gate, London, SW1. (To be published).