INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 2, 5–32 (1970)

A FRONTAL SOLUTION PROGRAM FOR FINITE ELEMENT ANALYSIS

BRUCE M. IRONS

University of Wales, Swansea

SUMMARY

The program given here assembles and solves symmetric positive-definite equations as met in finite element applications. The technique is more involved than the standard band-matrix algorithms, but it is more efficient in the important case when two-dimensional or three-dimensional elements have other than corner nodes. Artifices are included to improve efficiency when there are many right hand sides, as in automated design. The organization of the program is described with reference to diagrams, full notation, specimen input data and supplementary comments on the ASA FORTRAN print-out.

INTRODUCTION AND MOTIVATION

The purpose of this text is to advertise the FORTRAN program in Appendix II, whose novel features make it a useful model for future programs. Adding or removing facilities to suit different engineering requirements is a straightforward task given the present documentation, which includes the program notation in Appendix I.

The work was initially motivated by the challenge of optimization of complex continua, say of plastic three-dimensional bodies with large deflection. It is unsatisfactory to spend 1000 times as long in analysing trial structures as in interpreting stresses so as to alter the design, and to analyse many structures which are certain to be distant from the optimum. By linearizing the rates of change of stress and weight with each design change, and using linear programming to find the next trial structure, we can make the two alternating phases of the design cycle more equal and, presumably, reduce their total cost.†

The stresses are differentiated with respect to all possible design changes by a physical argument.[1,2] Consider a region of the boundary subject to design modification, as in Figure 1. The vector RH shows a typical unit design change D_R; thus if $D_R = 0{\cdot}1$, the node R moves to R^* where $RR^* = 0{\cdot}1\,RH$. We should perhaps deal with 50 or more variables D_i. When R moves to R^*, now a small change, an extra region of material RSR^*Q is added to the structure. By considering virtual work, we can find the change in external load (gravity, pressure, buoyancy, etc.) and thus deduce the change in equivalent loads at Q, R and S, the only nodes affected. Further, we can assume that the stress anywhere, e.g. in RSR*, is the same as that nearby on RS, and compute the internal virtual work on the new material, and hence the corresponding internal loads at Q, R and S (see section on *The Re-solution Facility*). Thus we find the vector of equivalent applied loads $\{X\}_R\,\mathrm{d}D_R$ due to a single design change $\mathrm{d}D_R$. Then re-solving the equations with

† Since the problem is non-linear, large design changes must be scaled down to prevent the process from running amok.

Received 10 July 1968
Revised 26 August 1969

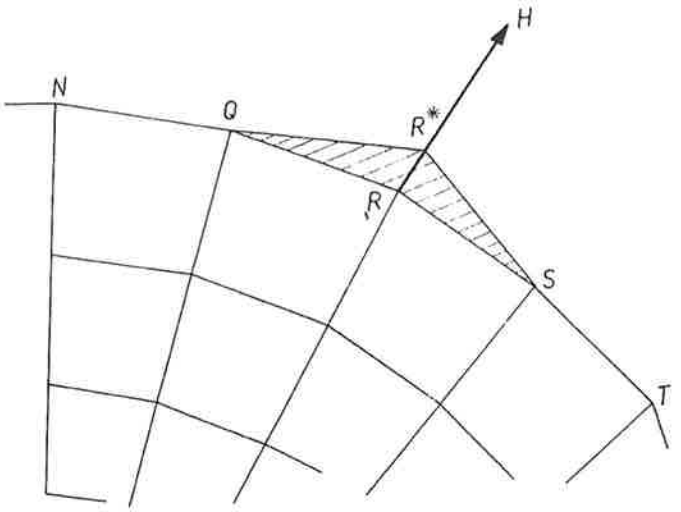


Figure 1. An allowed design change, showing wedges of extra material

pseudo-loads $\{\mathbf{X}\}_R$ gives the stress derivatives $\partial\sigma_j/\partial D_R$ at all points N, Q, R, S, ... If we are constraining a stress at R, we must allow also for the movement of the point R, and this entails another contribution to the derivative $\partial\sigma_R/\partial D_R$, namely the change in stress with constant nodal co-ordinates and deflections. The first structural solution is for, say, N_R load cases, but to calculate the design derivatives the same equations must be re-solved with N_R pseudo-load cases for each design variable D_i, and success depends on doing this economically.

The program has run successfully on the ICT 1905E computer. The problems, designed to test the new facilities, included pathological data, e.g. repeated elements, nodes repeated in one element (see 'Assembly' in Appendix I) and two uncoupled problems introduced as one. The program has also run for an hour computing and checking results from data generated entirely from random numbers except for the fixed values MAXELT = 1,000,000 and NGUIDE = −1: the main incentive to improve the diagnostic facilities (penultimate section) arose in the early stages of this test.

STRATEGIC CONSIDERATIONS

The problem discussed here is solution of the linear algebraic equations

$$\mathbf{Cx} = \mathbf{X} \tag{1}$$

where $\mathbf{C}$ is symmetric and positive-definite. Provided that floating-point arithmetic is used, it is unnecessary to change the order of the equations so as to select the largest pivot, and therefore we can take full advantage of symmetry.[3] When the unknown variables $x_1, x_2, x_3, \ldots, x_N$ are suitably ordered, the coefficient matrix $\mathbf{C}$ is 'banded'; that is, C_{ij} is zero if abs.$(i-j) > n$, where 'n' is the semi-bandwidth. Many workers already have programs which use only $(n+2)(n+3)/2$ words of storage for processing the coefficients, and which require about $\frac{1}{2}n^2 N$ multiplications to solve N equations: see Figure 2. Indeed, no less efficient technique deserves serious attention.

However, a 'frontal' technique sometimes gives even higher efficiency. For as finite element theory developed, workers became interested in elements with more degrees of freedom. Whereas simple elements have corner nodes only, these refined elements often have nodes along their edges, or on the faces of three-dimensional elements. Figure 3(a) shows a typical case, where each node is coupled to eleven others in a rectangular element; the frontal technique uses less

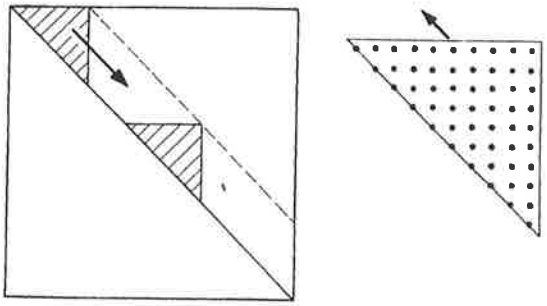
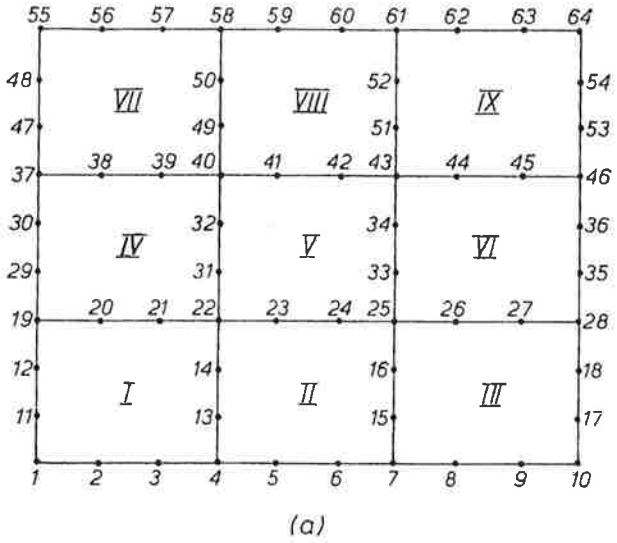
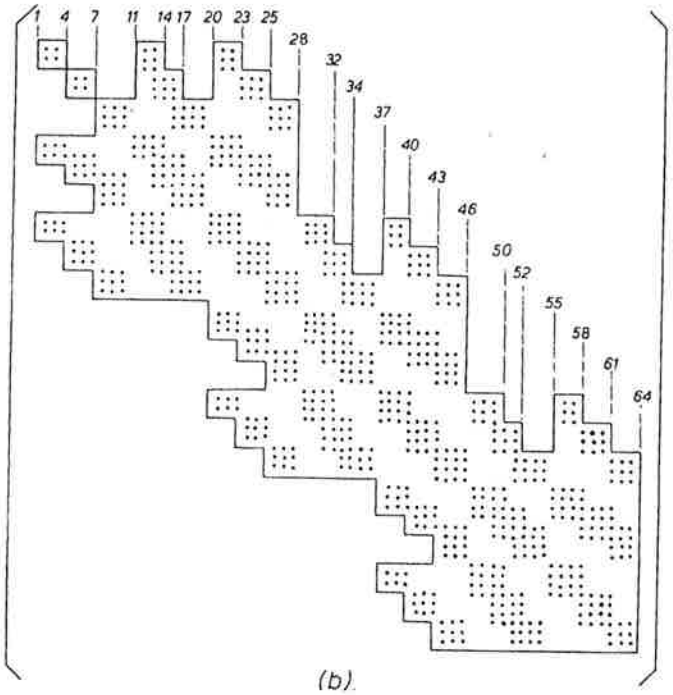


Figure 2. Diagram of normal solution of band equations



(a)



(b)

Figure 3(a) and (b). Finite element assembly giving re-entrant banding, displaying the area of matrix needed for the elimination process

storage and arithmetic here than the best Gaussian band algorithm. Nevertheless, one should debate whether a band algorithm suits one's immediate needs better: for it can be simpler and more straightforward, it spends less time in logic and compilation, and when compiled the program should take less storage—although the frontal solution routines CODEST and ZIPP in Appendix II take only 2400 words of 24 bits in the ICT 1905 computer, by virtue of careful coding. Accordingly, workers with a large- or medium-sized computer and with a heavy commitment to the finite element approach will favour the frontal method, whereas uncommitted workers might start with a good band method. If one's computer is small but one's ambitions are large, the choice is difficult; a case can still be made for the frontal technique, however, as when three-dimensional problems are run on a computer with effectively only 16K 48-bit words.[4]

HOUSEKEEPING WITH BANDED EQUATIONS

In the finite element method, any C_{ij} or X_i in equations (1) is the sum of several contributions from different elements. Each element contributes a square matrix $\mathbf{C}^e$, usually semi-definite, which involves only a few of the perhaps many thousand degrees of freedom. In the elastic case the coefficients of $\mathbf{C}^e$ are calculated by integration:

$$C^e_{ij} = \int c_{klmn}\, \partial\varepsilon_{kl}/\partial x_i .\, \partial\varepsilon_{mn}/\partial x_j .\, \mathrm{d(volume)} \tag{2}$$

where c_{klmn} is the tensor relating stress to strain, so that $\sigma_{kl} = c_{klmn}\,\varepsilon_{mn}$.

If one is presented with a finite element mesh whose nodes—and hence variables—have already been assigned numbers, one knows the semi-bandwidth 'n' from the largest spread of numbers in any element. Thus if the nodes of a particular triangular element are numbered 5, 19 and 37, and if there is one variable at each node, $n \geqslant 37-5 = 32$. Given the bandwidth, an efficient equation solving technique is to retain on core only a triangle of coefficients, which effectively moves diagonally downwards as elimination proceeds: see Figure 2. (The detailed organization is that when the first variable is eliminated using the first equation, all the other coefficients in the triangle are modified, and as they are modified they are shifted diagonally *upwards*; the second equation now takes first place in the coefficients on core, and so on.) Usually the band is sparsely filled, and frequently the bandwidth varies from one position to another, and it is then natural to modify only the coefficients within the $n_l \times n_l$ sub-matrix, where n_l is the temporary semi-bandwidth. But sometimes we meet an inefficiency that does not respond to such treatment, as in the matrix with re-entrant band illustrated in Figure 3(b). The matrix in Figure 4(a) shows diagrammatically how the triangle of coefficients on core is only partially filled with non-zero coefficients in such cases, even during elimination. A scheme as in Figure 4(b), where the columns are strung out as a long vector, gives some improvement.

THE FRONTAL TECHNIQUE

The frontal technique is however more radical.[5,6] Its principles are implied by the Gaussian process itself, for when we eliminate x_s using equation e_s we obtain

$$C^*_{ij} = C_{ij} - \left\{\frac{C_{is}\, C_{sj}}{C_{ss}}\right\} \tag{3a}$$

$$X^*_i = X_i - \left\{\frac{C_{is}\, X_s}{C_{ss}}\right\} \tag{3b}$$

1. C_{ij} or X_i is the sum of the element contributions; but it does not matter in what order the additions and subtractions are made, provided that the terms shown above in curly brackets are

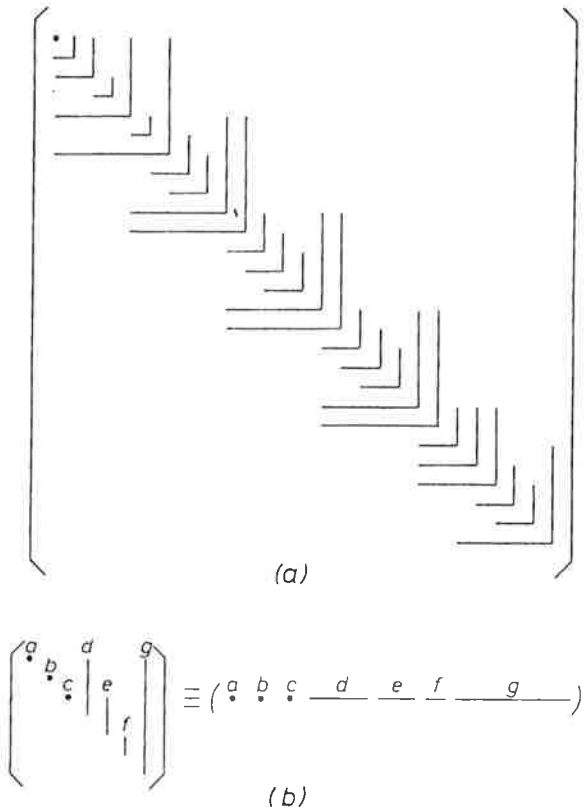


Figure 4(a) and (b). Matrix with re-entrant band and triangle representation

correct: i.e. provided that C_{si} $(= C_{is})$, C_{sj}, C_{ss} and X_s are fully summed. Hence the C_{ij} and X_i need not be fully summed, except those in the equation e_s used to eliminate x_s.

2. If either C_{si} or C_{sj} is zero, $C^*_{ij} = C_{ij}$ from formula (3a). For efficiency, therefore, C_{ij} should be on core only if both C_{si} and C_{sj} are non-zero. The square sub-matrix so defined ('Grandpa' in the program comments, 'G' in the text) relates only to the currently 'active' variables x_f.[6] Thus the subset x_f excludes the x_s already eliminated. Further, consider the equations e_s used in past eliminations and also in the elimination about to be performed; but consider them in their virgin form e^o_s. A considerable number of variables x_k may have zero coefficients in all the equations e^o_s so far encountered, so that the C_{sk} must be zero; thus the 'active' variables x_f must exclude the x_k also. These statements are self-evident for a banded matrix, but they hold generally.

The frontal process alternates between accumulation of element coefficients (assembly), and elimination, so that the virgin equations e^o_s rarely appear, fully summed yet unmodified. But suppose we assemble an element containing variable x_s about to be eliminated, and also containing x_f. Because e^o_s would now have a non-zero coefficient of x_f—for on principle we regard the $\mathbf{C}^e$ as full matrices—we must henceforth count x_f as an active variable. Further, if x_s is ready for elimination there must be no subsequent elements containing x_s. Therefore a variable becomes 'active' on its first appearance and is eliminated immediately after its last.

The subset of active variables x_f continually changes, as in the band algorithm; however, this is no longer achieved by a bodily shift of all the coefficients—which inevitably takes time—but by leaving spaces in 'G'. Thus, when the C^e for element I of Figure 3(a) is added into 'G', variables 1, 2, 3, 11 and 12 are eliminated immediately, leaving null rows and columns. The equations e_s

are stored in a buffer area, ready for tape, as shown in Figure 8, and the C^e for element II is assembled, re-filling the empty rows and columns of 'G'. The process continues thus, and we can assess the storage requirements without for the moment considering the housekeeping in detail. Figures 5 and 3(a) show that just before element VI is introduced, variables 25–28, 33, 34 and 37–43 are active. Element VI then introduces the new active variables 35, 36, 44, 45 and 46. Thus 'G' must accommodate up to 18×18 coefficients, whereas the band algorithm would involve 22 variables because this corresponds to the span of node numbers in each element. We can further observe:

(a) When the frontal technique is more efficient than the band algorithm it is because it discards the active variables in an order different from that in which it picks them up.

(b) The elements are presented in a certain order, which is critical—just as the node numbering is critical in a band algorithm—although again the best order is not always unique. Thus the element number, increasing horizontally in Figure 5, is akin to time t during the assembly–elimination process. We see that x_{40} makes its first appearance in element IV and its last in element VIII. On the other hand, x_{10} makes its first and last appearance in element III.

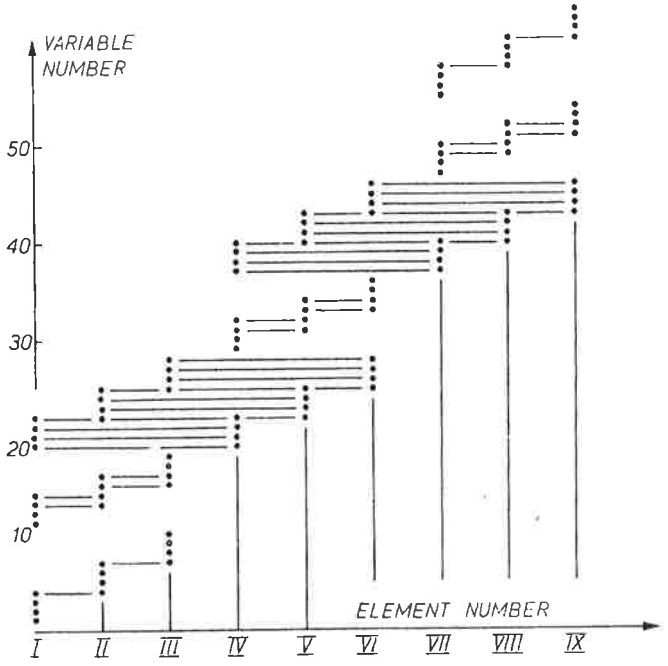


Figure 5. Graphical representation of frontal treatment applied to finite element assembly of Figure 3(a)

(c) Ordering of the variables x is irrelevant to the frontal technique. Indeed, the labels 1 ... 64 are sometimes termed 'nicknames' to emphasize this fact.

(d) If the original mesh is too coarse in some region, as in Figure 6, a soft rubber or a black pen suffices to alter the drawing. Although the frontal data are little changed, a band algorithm may demand extensive re-numbering to preserve a small bandwidth.

(e) In a ring structure, the band algorithm requires an artificial order of node numbers to achieve a small bandwidth, as in Figure 7: but the order of elements for the frontal technique is a natural one. Reference 6 gives less amenable cases.

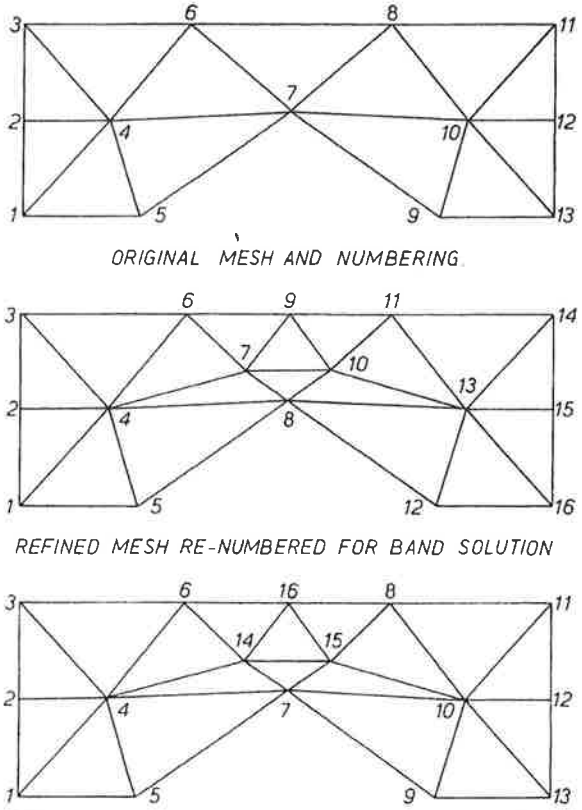


Figure 6. Mesh refinement with band and front solution

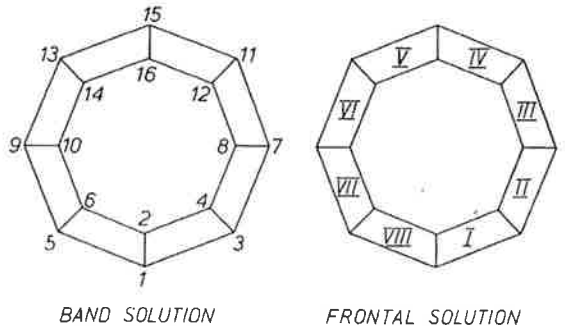


Figure 7. Numbering of nodes and elements in a ring structure

(f) Before element VI is introduced, the active variables are 37, 38, 39, 40, 41, 42, 43, 34, 33, 25, 26, 27 and 28. Figure 3(a) shows that these cut the structure into two independent parts, or 'substructures'. After assembly and reduction of the coefficients arising from element VI, that is, just before element VII is assembled, the active variables are 37 ... 46. These two sets of variables define 'fronts' composed of nodes separating the substructures, and immediately after adding in the coefficients generated by element VI both sets of variables must be active simultaneously. The front advances thus, an element at a time as portrayed in Figure 5, and one can therefore assess the storage requirements directly from the finite element mesh pattern, as in Figure 3(a).

HOUSEKEEPING IN THE FRONTAL TECHNIQUE

The main working area is ELPA, a vector subdivided for dynamic storage as in Figure 8. If the front is large, ELPA must be long; indeed, the size of the problem has often been severely limited by the storage available for ELPA, and the buffer area is sometimes uneconomically small. If the front is small, there may still be some advantage in keeping the maximum dimensions for ELPA, especially if the buffer area is large enough to avoid using tape for the equations.

The flow diagram of Figure 9 shows the pre-program (ending at statement 26 of ZIPP) which assembles elements and eliminates variables in the abstract, using the input lists of variable labels LVABL to represent each element and to update the list MVABL representing 'G'. At statement 20 the element destination vectors LDEST are created; LDEST(7) = 59, for example, would mean that the same variable, say x_{483}, is represented in LVABL(7) as in MVABL(59)—that is, the seventh row and column of the current C^e must accumulate into row and column 59 of 'G'. Meanwhile we count the variables as we eliminate them—see NVABZ in Appendix I; and we record first and last appearances, also in LDEST but in coded form—see KOUNT, NSTRES. We also record maximum sizes for dynamic storage—see MAXPA, LVMAX, NRMAX, NEWRHS, MAXELT, MAXNIC and MAXTAP. Available space is dictated by NIXEND and NELPAZ—see Figure 8.

The pre-program must always execute data checks, which vary from problem to problem, and a few are described in the penultimate section.

BACK-SUBSTITUTION AND OUTPUT BY ELEMENTS

Each equation e_s for back-substitution is accompanied on tape by the current size of 'G', i.e. KURPA, and the label NIC of the variable being eliminated, with its position LDES in the list of currently active variables MVABL. Thus in the back-substitution phase the coefficients of equation e_s carry with them information on how long it is and how it is to be solved. The housekeeping remains simple, because when each variable is calculated it is placed in position LDES in the 'running variable' vector. At any time, therefore, the composition of this vector reflects the composition of 'G' at the corresponding time during the elimination; thus, in back-substituting, we can take a direct scalar product of the coefficients of e_s into the running variable vector. Further, if we can choose the correct time during back-substitution, we can use the element destination vector LDEST to select the solutions for a complete element. The correct time is when LPREQ = NEQ (*quod vide*) and this option (NTIREX = 0) is known as 'output by elements'. The alternative, 'entire output' (NTIREX = 1) simply places the computed variable in position NIC in the vector ELPA. Several right hand sides can be processed simultaneously, and the results appear consecutively in ELPA, the first as the first block, etc. as in Figure 8. With output by elements, the results for the last element appear first, to be overwritten by those for the last but one, and so on in reversed order.

THE RE-SOLUTION FACILITY

The program is designed to re-solve without repeating the assembly and reduction of the matrix coefficients. In this mode it can accept many more right hand sides than in the original solution, because the dynamic storage now takes advantage of the absence of left hand sides. This is possible because the coefficients required in formula (2b) can be retrieved from the existing e_s; but their right hand sides must be updated. The element data also comprises right hand sides only. Output by elements (NTIREX = 0) is particularly efficient in this mode, because storage of output data in ELPA now becomes critical. For example, stresses may be calculated at nodes as each element's solutions appear; further, if average stresses at nodes are required the totals may

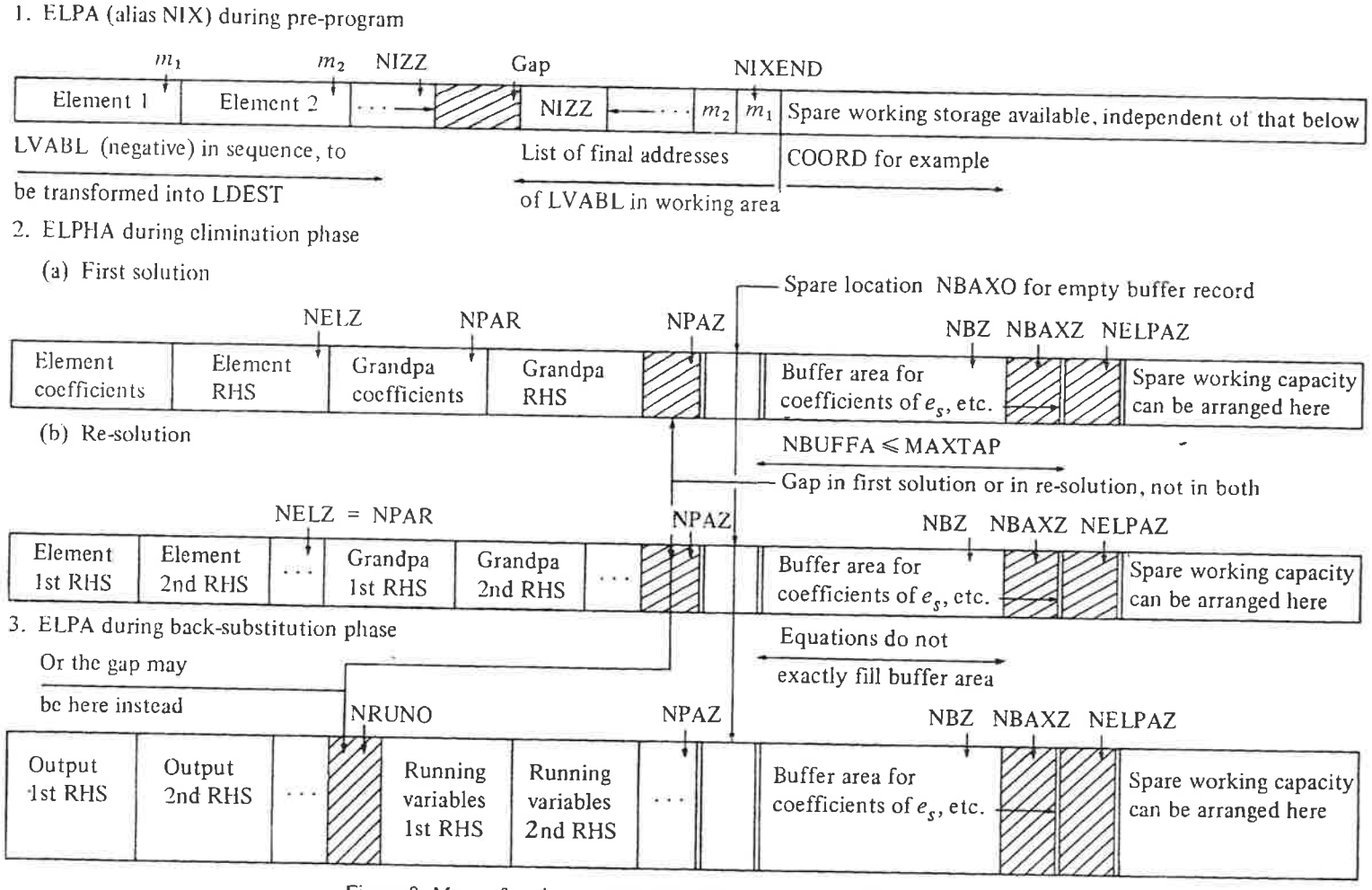


Figure 8. Maps of various storage allocations used in working area ELPA

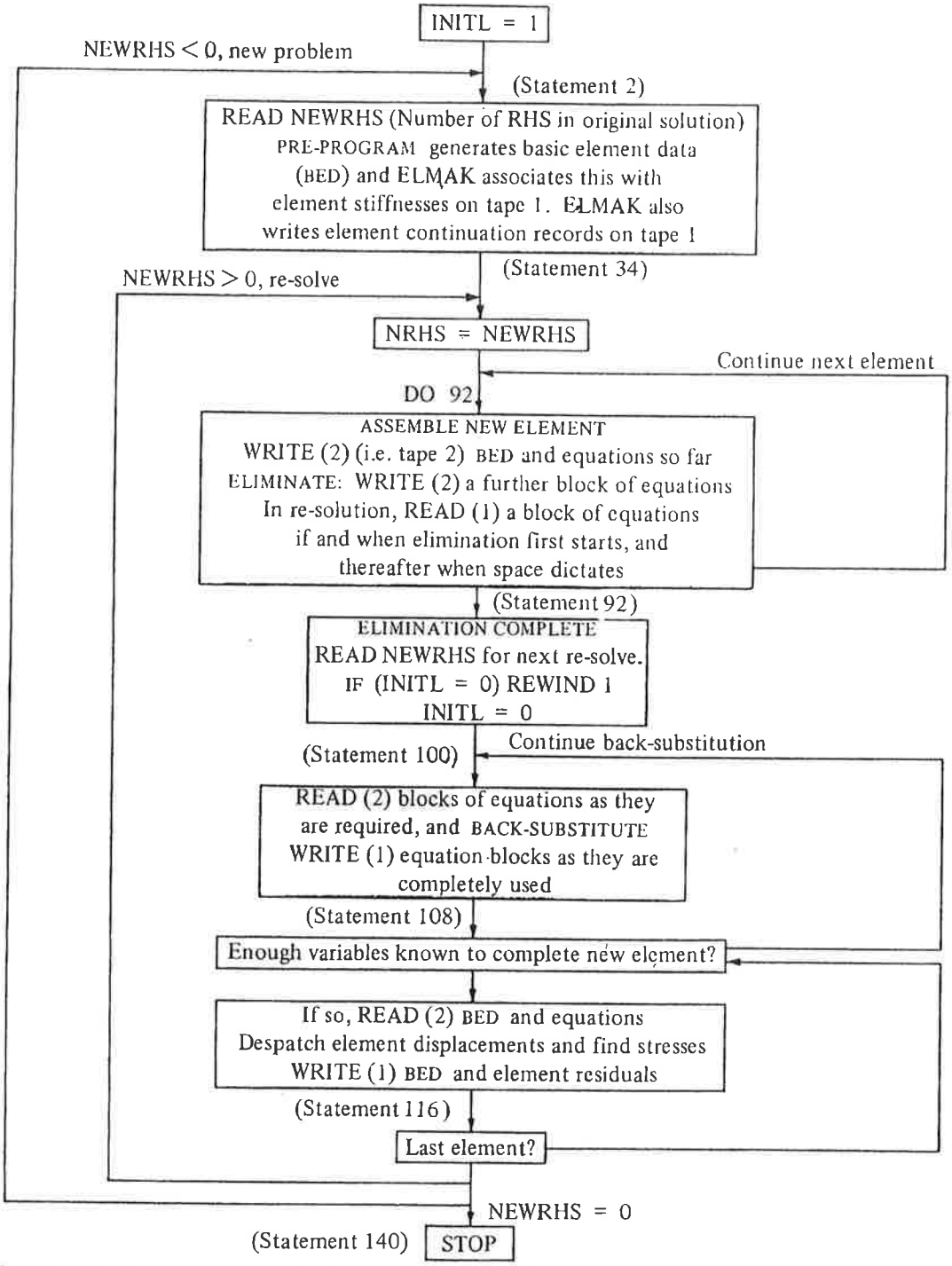


Figure 9. Flow diagram showing main events in the program in Appendix II. Note: when INITL = 0, each READ is preceded and followed by BACKSPACE

be accumulated into another running vector, of stresses this time, and averaged immediately the last stress at a particular node appears. The program not only specifies the correct time, but also provides NSTRES, the number of stresses to be averaged.

To mathematicians the best known algorithm using re-solution is that of iterative refinement for assessing ill conditioning, the new right hand sides being the residuals R calculated from the given equations using double-precision accumulation (Reference 3; and Reference 7, p. 255). In the present context, it would be natural to calculate the residuals directly from the element matrices, $R^e = X^e - C^e x^e$. An equally direct technique is to integrate over the elements. Thus the internal nodal forces are those due to the stresses σ_{ij}, and the external forces X are also calculated by considering virtual work.

$$R_k = X_k - \int \sigma_{ij}\, \partial\varepsilon_{ij}/\partial x_k \,.\, \mathrm{d}(\text{volume}) \tag{4}$$

This involves calculating a vector for each element, usually by numerical integration. Equation (4) has a wider connotation, because it also defines the residuals calculated in order to solve non-linear structural problems by iterative procedures. In such cases the σ_{ij} and $\mathrm{d}\varepsilon_{ij}$ are defined and calculated by more involved and diverse procedures than in equation (2). Physically, the R represent 'residual' forces which, together with the actual forces, would give the current displacements x. Frequently the small displacement linear response of the structure to these forces R gives good correction terms, especially if an accelerator is used. It is convenient to find the contributions R^e for each element as soon as its displacements are available, just as the pseudo-loads mentioned in the first section must be calculated immediately from the output by elements. As presented, ZIPP does this, and the wider strategic implication is that in all iterative processes we should seek ways of exploiting the facility for processing a very large number of right hand sides in the second and subsequent passes. However, we should note that the process is not ideal for the third, at present more usual, application in which many prescribed load cases are applied to a linear structure. Here all the loads are known at outset, and a subset of the given loading cases is processed in each re-solution. Although output by elements remains cheaper, the engineers concerned are unlikely to be pleased with this manner of presentation.

FRAGMENTED ELEMENTS AND SUBSTRUCTURES

Given a sufficiently flexible filing system, the frontal technique can deal with substructures in a general way, so that any structure can become an 'element' of a larger configuration. Although this is not yet feasible, the program can already deal efficiently with very large elements; to introduce C^e and the associated right hand sides as a long vector in ELPA would waste core storage, but the facility exists—as demonstrated in Appendix III—for fragmenting this vector into shorter tape records, of prescribed maximum length MAXELT.

ASSIMILATION OF INPUT IN THE PRE-PROGRAM

We now describe how the program's activities appear to the computer peripherals, and continuous reference to Appendix III and Figure 9 is recommended. Preliminary input data are read below statement 2 of ZIPP (Appendix II):

NEWRHS, which has specialized meanings; see *Notation*, Appendix I.

NTIREX, which is 1 or zero; see section on *Back-substitution and output*, under 'output by elements'.

NRMAX. In the first solution process, we shall introduce the number of right hand sides as NEWRHS. There will be space for many more right hand sides in the 're-solution'; but to ensure that dynamic storage is not violated, we must guarantee at the outset not to submit more than

NRMAX right hand sides in any re-solution. The program allocates storage on this basis and generates an error message if ELPA is too short (for example 3000 words are allocated in Appendix II) or if NRMAX is violated. Furthermore, as the coefficients of each equation enter the buffer area, space is left for NRMAX right hand sides.

MAXTAP. It may happen that ELPA is much longer than necessary. The equations e_s are normally written on tape in long records, to save time; but it was once argued that, because worn tapes tended to waste time by momentarily failing to record, and automatically back-spacing for another attempt, less time would be lost if the record lengths were limited. MAXTAP = 1000 would limit the buffer length NBUFFA to 1,000 words or less. Tape performance has now improved, but this facility will continue to be useful in restricting the records to be nearly an integral number of disc or drum tracks in length.

MAXELT. The maximum length of an element segment—see section on *Fragmented Elements and Substructures.*

Next the program reads below statement 4 what variables feature in each element in turn: see KUREL and LVABL in Appendix I. A final extra 'element' with KUREL = 0 terminates this reading process, so that the program can count the incoming elements—see NELEM, NELEMZ. As the program receives LVABL it extracts the element co-ordinates ELCOR and checks their credibility: we assume here that COORD is available while the pre-program runs. It interprets the LVABL, creates LPREQ below statement 12 and LDEST at statement 20. In the strategy adopted here (case 1 below) the 'basic element data' (BED), comprising KUREL, LPREQ, LVABL, LDEST and ELCOR for each element, is written on tape 1 together with the element stiffnesses and right hand sides when ELMAK is called at the end of the pre-program. The initial element properties must be in the following form for a 3×3 element with two right hand sides:

$$12, [C_{11}, C_{12}, C_{22}, C_{13}, C_{23}, C_{33}, X_{11}, X_{21}, X_{31}, X_{12}, X_{22}, X_{32}]$$

where LZ = 12 (*quod vide*, Appendix I) is the length of the segment. But the element properties for re-solution with KUREL = 3 and NRHS = 4 will be

$$12, [X_{11}, X_{21}, X_{31}, X_{12}, X_{22}, X_{32}, X_{13}, X_{23}, X_{33}, X_{14}, X_{24}, X_{34}]$$

For brevity, such element properties will be called C^e in both cases.

TAPE OPERATIONS DURING THE FRONTAL ELIMINATION

Several strategies should be considered in creating the initial element properties:

1. As early as possible—see previous section; we create C^e for an element as soon as we have its LDEST, having completed as much checking as possible. However, MAXPA remains unchecked, and could cause failure after the C^e have been calculated—an expensive contingency.

2. As late as possible—that is, just prior to assembly. Tape 1 now contains only the BED, and the operations are: create C^I, assemble C^I into 'G', eliminate, create C^{II}, assemble C^{II}, ... This is the cheapest method; the disadvantage is that any arrays used by ELMAK would take storage from ELPA—but intelligently written element routines are usually short.

3. As modular as possible, so that programmers can work independently. The pre-program writes the BED only on tape 2, and a new loop between the pre-program and the elimination retrieves the BED and creates tape 1 as before. There is an extra tape operation, but the principle involved is a popular one.

Just before elimination commences, whether for initial solution or re-solution, dynamic storage is allocated: NBAX0, the start of the buffer area for the equations e_s, is inviolate after the initial allocation, and cannot be altered for re-solutions. During elimination equations e_s are produced, and are written on tape 2 together with the BED—see Figure 9. Every tape-reading operation after

initial element assembly must access the records in reversed order, hence the action BACKSPACE–READ–BACKSPACE. With backward reading tapes this costly artifice can be avoided, but for most users it remains the one ugly feature of Gaussian elimination with backing storage. It is not peculiar to the frontal technique.

TAPE OPERATIONS IN BACK-SUBSTITUTION AND IN RE-SOLUTION

With elimination completed, NEWRHS is read for the next solution; see Appendix I. For back-substitution the e_s are required in reverse order, and output by elements is not possible unless the BED is available when the values of x are calculated. From the first back-substitution onwards, the BED and the e_s are transferred alternately from tape 1 to tape 2 and back again. This is necessary because we cannot update tape records without re-writing them completely.

Just as there were three possible strategies in calculating the initial element data (see previous section) so the right hand sides for re-solution may be calculated as early as possible, as late as possible, or in a separate loop to achieve modularity. The present program is consistent in that the C^e are always calculated at the earliest opportunity. Thus with output by elements the residuals R for the element may be calculated and written on tape immediately, for example. This is possible below statement 112 of ZIPP (Appendix II), but no provision is made for calculating new right hand sides with entire output.

A simple example will show the form of the later tape records and explain an unexpected difficulty. Consider a three element problem, element I (x_1 only), element II (x_1, x_2) and element III (x_2, x_3), and assume that the buffer area has room for two equations. We now write the tape records in shorthand form, separated by commas.

Tape 1 for initial solution:

$$(\text{BED})^{\text{I}}\,\mathbf{C}^{\text{I}}, (\text{BED})^{\text{II}}\,\mathbf{C}^{\text{II}}, (\text{BED})^{\text{III}}\,\mathbf{C}^{\text{III}}$$

Tape 2 for back substitution; the records are read from this tape in reverse order:

$$(\text{BED})^{\text{I}}\,e_0, (\text{BED})^{\text{II}}\,e_1, (\text{BED})^{\text{III}}\,e_2\,e_3$$

Here e_0 denotes a null block, ELPA(NBAX0) only.

Tape 1 for re-solution; the records are written on this tape in reverse order:

$$(\text{BED})^{\text{I}}\,\mathbf{C}^{\text{I}}, (\text{BED})^{\text{II}}\,\mathbf{C}^{\text{II}}, e_1, (\text{BED})^{\text{III}}\,\mathbf{C}^{\text{III}}, e_2\,e_3$$

The difficulty arises because normally the signal to write the buffer to tape 2 and to read another buffer from tape 1 is given when the buffer is full. Thus, for example, when the last record, $e_2\,e_3$, is to be read during the subsequent elimination, e_1 has been re-processed and written on tape 2 as $(\text{BED})^{II}\,e_1$, but only half the buffer area has been used. We must therefore make the computer read $e_2\,e_3$ 'prematurely'.

To resolve this difficulty we put NEW = 1 after each element is assembled. With NEW = 0, the normal condition, the program reads fresh equations only when the present block is exhausted. But with NEW = 1, an instruction to eliminate x_s automatically calls a new block of equations, whereupon NEW reverts to zero until the next element is assembled.

OUTPUT AND ERROR DIAGNOSTICS

With 'entire' output, the value of x_s due to right hand side NR is found in

$$\text{ELPA}\{s+(\text{NR}-1)\times\text{MAXNIC}\}.$$

With 'output by elements' (NTIREX = 0) the 'x' becomes available below statement 110, in ELPA{KL+(NR−1)×KUREL}, and stresses may then be calculated. If the stresses are to be

averaged at nodes, we accumulate them in another 'running vector', in a position corresponding to LDES, just as nodes correspond to variables (say 1 node to 3 variables). Above statement 116, the program later presents us with the position LDES, and an integer NSTRES which is the number of contributions to be averaged.

The error diagnostics print JWHERE etc. to identify the fault:

JWHERE = 1, NELPAZ, LVEND, MVEND, NIXEND, NEWRHS, NRMAX, MAXTAP or MAXELT is zero or negative, or NTIREX is neither 0 nor 1.

JWHERE = 2, KUREL is negative.

JWHERE = 3, increase LVEND, the dimension of LVABL and LDEST.

JWHERE = 4, the element co-ordinates, or some other input element property, appears wrong.

JWHERE = 5, NIX, NIXEND too small for the pre-program.

JWHERE = 6, a nickname is zero or negative.

JWHERE = 7, MVEND, the dimension of MVABL, is too small.

JWHERE = 8, the guaranteed NRMAX has been exceeded.

JWHERE = 9, NELPAZ, the dimension of ELPA, is too small.

JWHERE = 10, an initial element segment is too long, or $LZ \leqslant 0$.

JWHERE = 11, a continuation element segment is too long, or $LZ \leqslant 0$.

JWHERE = 12, the total length of C^e is wrong.

JWHERE = 13, a zero pivot: either a 'flying structure', i.e. one that is inadequately earthed, or a mechanism.

JWHERE = 14, CRIT $> 10^8$, fatal ill-conditioning, as for JWHERE = 13 for practical purposes: see Appendix I for CRIT.

These errors stop the computer, but we can call any further diagnostics the computer provides; see A in Appendix I.

JWHERE = 15, CRIT $> 10^4$, suspected ill-conditioning, or pivot is negative. A warning is printed but the computer does not stop.

DISCARDED FACILITIES

A program must be judged by practical criteria: it can be too complicated or too expensive for its purpose. A facility is an encumbrance if it is not used, and different engineering groups have different requirements. The following facilities were considered but were not included in the present program:

1. Assembly of a $p \times q$ array of coefficients for Lagrange multipliers, accompanied by LVABL of length $p+q$. This now requires the full triangle of coefficients, mostly zero, containing the $p \times q$ rectangle. Indeed, the program has no special facilities for the various types of constraint distinguished in Appendix IV. Most programmers include type 2(b) in the equation solving package, but the author prefers a simple package which places all constraints under the direct control of the engineer.

2. An additional symbol NODVZ = variables per node. Some good elements have NODVZ varying from node to node, so that one nickname per node would not suffice. If NODVZ is constant, the technique now envisaged is to manipulate the nodal nicknames in the pre-program, and to expand the LDEST etc. before entering the frontal elimination.

3. An energy variance criterion of roundoff error, sensitive to right hand sides, as in Reference 7. This was complicated and the calculation of total energy for comparison used excessive storage.

4. Storage facilities in ELPA for several element matrices simultaneously, as in a matrix scheme. Originally a labour saver when element data was manipulated by hand, this has been little used.

5. Input-defined elimination order. This was an encumbrance. We can still disturb the order by introducing single-variable null element matrices.

6. A device which, in the event of a fatal data error, causes the program to skip all the remaining data for the problem and to proceed to the first card of the next problem. This is a desirable feature, but belongs more properly to the realm of system planning than of equation solving.

ACKNOWLEDGEMENT

The author wishes to thank Dr. D. W. Martin, of the National Physical Laboratory, for his generous help in the presentation of this article.

APPENDIX I

Notation

A occurs at the end of ZIPP (Appendix II) as $A = 1{\cdot}0/0{\cdot}0$. This is inadmissible, and causes data errors to give trace or dump. See penultimate section.

'Active' variables x_f are defined in the section on *The Frontal Technique*, para 2.

'Assembly' of an element means accumulation of its coefficients into Grandpa, as in statement 60 of ZIPP. The possibility exists of an LVABL containing two or more identical nicknames, and if LHS = 1, IG = KG and IL ≠ KL, then the contribution ELPA(L) must be doubled, as in statement 54.

BED, or 'basic element data'; see section on *Assimilation of Input in the Pre-program*, also Figure 9.

C in the algebraic discussions is the square matrix in $Cx = X$.

C^e is the element contributions to the left hand side and/or the right hand side of the equations.

CODEST in Appendix II is a sub-routine to interpret **cod**ed **dest**inations.

CONST occurs in the elimination and in the back-substitution innermost loops; see statements 82 and 104 of ZIPP.

COORD is a list of all the nodal **co-ord**inates in the structure. It is included in the COMMON and EQUIVALENCE statements to show how the storage of ELPA might be borrowed elsewhere, e.g. in the routine that calculates the element properties; see Figure 8.

CRIT is a refined diagonal decay **crit**erion of roundoff damage, which has proved successful although not strictly defensible[8]. It is created and assessed during elimination, and the associated output NIC shows which pivot caused the failure, thus helping the engineer to understand it physically and perhaps to correct it.

e_s denotes equation **s** as used to eliminate x_s.

EL contains **el**ement segments: used only for demonstration in ELMAK.

ELCOR contains the nodal **co-or**dinates for the particular **el**ement. These would form part of various tape records.

ELMAK in Appendix II is a routine that **mak**es **el**ement data. Here the data C^e is read from cards, but in a real application it would be created in ELMAK from ELCOR, etc.

ELPA is the long working vector, comprising **el**ement contributions, Grand**pa**, etc. The dynamic storage maps are in Figure 8.

G, or Grandpa, is a sub-vector of ELPA which contains the coefficients of the assembled equations and the associated right hand sides.

IB0 preserves a starting value of IBA.

IBA: a coefficient in the equation for **b**ack-substitution is ELPA(IBA).

IBAR: the mth right hand side in the equation is ELPA(IBAR + m).

IBDIAG: the **diag**onal coefficient in the equation is ELPA(IBDIAG).

INITL is 1 for the **init**ial solution, zero for any re-solution. It becomes zero just before the first back-substitution.

IRHS is 1 when a **r**ight **h**and **s**ide is being processed; IRHS = 1 − LHS, or LHSRHS − 1.

JWHERE takes values 1 to 12 to identify input errors. See penultimate section.

KL = 1 to KUREL covers the variables of the **cur**rent **el**ement.

KOUNT **counts** the number of appearances of a variable, in thousands, plus 1000.

KUREL is the number of variables in the **cur**rent **el**ement.

KURPA is the **cur**rent size of Grand**pa**, i.e. the number of 'active' variables; see section on *The Frontal Technique*, para 2; or the length of an equation e_s.

L = 1 to LZ: an **el**ement contribution is ELPA(L).

LAS, LAST: the **last** appearance of a variable in the pre-program is in NIX(LAST). A provisional value of LAST is LAS.

LCUREQ is the number of variables already eliminated when the **cur**rent **el**ement appears, i.e. the number of stored **eq**uations. See NVABZ.

LDES, LDEST. The **el**ement **des**tinations are in LDEST, and LDES is the decoded version of LDEST(KL). See CODEST in Appendix II.

LHSRHS = 1 or 2. Assembly and elimination are compressed, so that the same innermost loop suffices for both LHS and RHS.

LHS = 1 means that the LHS is being processed, LHS = 0 the RHS: see IRHS.

LPREQ is the number of stored **eq**uations when the **pre**vious **el**ement was assembled. Used for output by elements; see NVABZ, NEQ.

LVABL is the vector giving labels of **el**ement **variab**les.

LVEND is the dimension of LVABL.

LVMAX is the **max**imum number of **v**ariables per **el**ement (degrees of freedom) actually encountered. There are many vectors LVABL, one for each element, and the length of the longest LVABL is LVMAX.

LZ: the current **el**ement segment extends from ELPA(1) to ELPA(LZ).

MAXELT is the guaranteed **max**imum length of any **el**ement segment.

MAXNIC is the **max**imum element label (**nic**kname) encountered.

MAXPA is the **max**imum size of Grand**pa** required, in terms of the number of variables.

MAXTAP limits the buffer length used; see section on *Assimilation of Input in the Pre-program.*

MG = MG0 + L to MGZ is the counter used in the elimination and the back-substitution innermost loops. See CONST.

MVABL is the list of 'active' **variab**les, i.e. those in Grand**pa** or in the running variables used in the back-substitution; see sections on *The Frontal Technique* and on *Housekeeping in the Frontal Technique.*

MVEND is the dimension of MVABL.

N1—see NEW.

NBAX0 + 1 to NBAXZ is the range of the buffer area in ELPA reserved for equations e_s for **back**-substitution; see Figure 8.

NBUFFA = NBAXZ − NBAX0 is the number of words in the **buffer**.

NBZ is the last word currently used in the **buffer**.

NDELT: The innermost loops increment two addresses in ELPA, separated by NDELT. See MG.

NDEQN is the position in ELPA of the **end** term of the **eq**uation.

NDIAG is the position in ELPA of the **dia**gonal coefficient in Grandpa.

NELEM = 1 to NELEMZ counts the **elem**ents.

NELPAZ is the effective dimension of ELPA. See Figure 8.

NEQ counts the **eq**uations backwards during back-substitution, to compare with LPREQ for output by elements. See section on *Back-substitution and Output by Elements.*

NEW = N1 to NZ in the pre-program ranges over an element in NIX.

NEW = 1 when an element has just been assembled, becoming zero later. See section on *Tape Operations in Back-substitution and in Re-solution.*

NEWRHS is typically the number of right hand sides in the next re-solution; but the number of initial right hand sides is first introduced as NEWRHS. It also takes the code meanings: NEWRHS = 0, no more problems; and NEWRHS < 0, another problem to follow. See Appendix II. NEWRHS is read before the back-substitution because tape writing operations can thus be avoided in the last solution.

NFUNC is a **func**tion defined in ZIPP giving the position in the equivalent vector of term (I, J) in an upper triangular matrix.

NGUIDE; see Appendix III; it **guides** ELMAK.

NIC is a label for a variable, a **nic**kname, always a positive integer. With output by elements, the NIC may be large numbers. This may be useful when the NIC are created automatically.

NIX, a vector using the same storage area as (equivalenced to) ELPA, is the main working area for the pre-program; see section on *Housekeeping in The Frontal Technique* and Figure 8. It starts as a list of values −NIC for successive elements.

NIXEND is the usable length of NIX, the remainder of ELPA being allocated for any additional arrays; see Figure 8. The program is dimensioned for a computer that uses two integer words for a floating-point number.

NIZZ gives the last label as NIX(NIZZ).

NODVZ is the **v**ariables per **node**; see last section.

NPAR: the term m in the right hand side NR associated with Grand**pa** is

$$\text{ELPA}\{\text{NPAR} + \text{MAXPA}*(\text{NR} - 1) + m\}.$$

Thus NPAR is the location preceding the assembled right hand sides.

NPAZ is the last available location for Grand**pa**; see Figure 8.

NRHS is the current number of **r**ight **h**and **s**ides, and differs from NEWRHS during the back-substitution.

NRMAX is the **max**imum **n**umber of **r**ight hand sides, guaranteed at outset for each problem. See section on *Assimilation of Input in the Pre-program.*

NRUN is the address of a **run**ning variable, ELPA(NRUN).

NRUN0 is the address preceding the **run**ning variables; see Figure 8.

NSTRES is the number of **stress**es to be averaged, alias the number of appearances of a variable. See comments in CODEST, Appendix II, also penultimate section and section on *The Re-solution Facility.*

NTIREX is 1 or 0 depending upon whether all the variables x are presented together (**entire** output), or one element at a time. See penultimate section and section on *Back-substitution and output by Elements.*

NVABZ becomes the total number of **variab**les. It is found by counting the variables as they are eliminated in the pre-program; see section on *Housekeeping in the Frontal Technique.* Before the variables in an element have been eliminated NVABV gives LCUREQ for that element. The value of LCUREQ for the previous element is known as LPREQ.

NZ: see NEW.

p are the **p**rescribed deflections or other variables x.

R are the **R**eaction forces to earth, or otherwise the **R**esiduals caused by prescribed values $x_i = p_i$. When an equation is not satisfied exactly, R_i denotes the discrepancy between the left and right hand sides.

'Running variables'; see section on *Back-substitution and Output by Elements* and Figure 8.

x are the deflections or other unknown variables.

X are the externally applied loads, or other right hand sides.
ZIPP in Appendix II is the main routine in the frontal solution program.
λ are Lagrange multipliers. See last section and Appendix IV.

APPENDIX II

```
C      FRONT SOLUTION, MARK V, WRITTEN BY B.IRONS, UNIVERSITY COLLEGE, SWANSEA.
C
       SUBROUTINE CODEST
C
C      (INTERPRETS CODED ELEMENT DESTINATIONS FROM LDEST.)
C
       COMMON ELPA(3000), LVABL(30), LDEST(30), MVABL(60), ELCOR(3,20),
      1   INITL, NTIREX, NEWRHS, NELEM, NELEMZ, KUREL, LPREQ,
      2   LZ, NELZ, NBAXO, NBZ, KL, LDES, NSTRES
       LDES = LDEST(KL)
       DO 2 NSTRES = 1,1000000          LDES becomes true destination.
       IF(LDES.LT.1000) GO TO 4         NSTRES = N for the first of N appearances.
       LDES = LDES - 1000                      = 0 for last appearance - but see below.
     2 CONTINUE                                = -1 for intermediate appearances.
     4 NSTRES = NSTRES - 2                     = 1 for only appearance - i.e. first
       RETURN                                        and last.
       END

       MASTER ZIPP
C
C     (PRE-FRONT AND FRONT ROUTINES COMBINED.)
C
       DIMENSION NIX(6000), COORD(3,100)
       COMMON ELPA(3000), LVABL(30), LDEST(30), MVABL(60), ELCOR(3,20),
      1   INITL, NTIREX, NEWRHS, NELEM, NELEMZ, KUREL, LPREQ,
      2   LZ, NELZ, NBAXO, NBZ, KL, LDES, NSTRES
       EQUIVALENCE (NIX(1),ELPA(1)), (COORD(1,1),ELPA(2701))
C
       NFUNC(I,J) = I + (J*(J-1))/2     This formula gives the position in the vector of
C                                       term i,j in an upper triangular matrix:  i ≤ j.
       NELPAZ = 3000
       LVEND = 30                       These quantities must be altered
       MVEND = 60                       when the dimension statement
       NIXEND = 5400                    is altered.
     2 INITL = 1
CARD INPUT
       READ(5,900) NEWRHS, NTIREX, NRMAX, MAXTAP, MAXELT as discussed in text.
       WRITE(6,802) NEWRHS              This must not exceed NRMAX.
 802   FORMAT(23H1NEW PROBLEM, INITIALLY,I4,3X,3HRHS//)
       WRITE(6,804) NTIREX, NRMAX, MAXTAP, MAXELT
 804   FORMAT(9H NTIREX =,I2,4X,7HNRMAX =,I2,4X,
      1   8HMAXTAP =,I4,4X,8HMAXELT =,I4/)
       JWHERE = 1                       The first of fifteen error traps.
       IF(NELPAZ.LE.0. OR .LVEND.LE.0. OR .MVEND.LE.0. OR .NIXEND.LE.0.
      1   OR .NEWRHS.LE.0. OR .NRMAX.LE.0. OR .MAXTAP.LE.0.
      2   OR .MAXELT.LE.0. OR .(NTIREX.NE.0. AND .NTIREX.NE.1)) GO TO 130
       LVMAX, NIZZ, MAXNIC, MAXPA, NVABZ, LCUREQ = 0
       DO 4 I = 1,MVEND                 MVABL will be the vector
       MVABL(I) = 0                     of Grandpa's nicknames.
     4 CONTINUE
C
C      PUT ALL ELEMENT NICKNAMES IN LONG VECTOR, NIX.
C
       DO 10 NELEM = 1,1000000          NELEM is element number, KUREL is current
CARD   INPUT                            element size.  LVABL is the list of current
       READ(5,900) KUREL, (LVABL(I), I = 1,KUREL)  element nicknames.  Zero KUREL is
       JWHERE = 2                       the normal end to the reading operation.
       IF(KUREL.LT.0) GO TO 130         Error trap: negative KUREL cannot be intended.
       IF(KUREL.EQ.0) GO TO 12
       IF(KUREL.LE.LVMAX) GO TO 6       LVMAX gives size of largest element matrix.
```

```
      LVMAX = KUREL
      JWHERE = 3
      IF(LVMAX.GT.LVEND) GO TO 130        Dimensions of LVABL, LDEST must be increased.
    6 JWHERE = 4                          This check is presumably in element routine.
C     (CHECK COORDINATES AGAINST NEGATIVE AREAS, REFLEX ANGLES ETC.)
      JWHERE = 5                          Is NIX large enough to contain next element?
      IF(NIZZ+KUREL+NELEM.GT.NIXEND) GO TO 130  NIZZ becomes the sum of all the
      DO 8 I = 1,KUREL                    element degrees of freedom.
      NIC = LVABL(I)
      JWHERE = 6                          Nicknames must be positive integers.
      IF(NIC.LE.0) GO TO 130
      NIZZ = NIZZ + 1
      NIX(NIZZ) = -NIC                    The nicknames are now made negative to
      J = 1                               distinguish them later from destinations.
    7 J = J + 1                           The program gives a warning message if an
      IF(J.GT.KUREL) GO TO 8              element contains a repeated nickname.
      IF(LVABL(J).EQ.NIC) WRITE(6,834) JWHERE, NIC
      GO TO 7
    8 CONTINUE
      NIX(NIXEND+1-NELEM) = NIZZ          Record element divisions, also in NIC.
   10 CONTINUE
   12 NELEMZ = NELEM - 1                  NELEMZ is the total number of elements.
      N1 = 1
      DO 26 NELEM = 1,NELEMZ              NVABZ becomes total number of variables
      LPREQ = LCUREQ                      eliminated: LCUREQ becomes number before
      LCUREQ = NVABZ                      current element, LPREQ before previous
      NZ = NIX(NIXEND+1-NELEM)            element.
      KUREL = NZ - N1 + 1                 We must reconstruct KUREL (current element size)
C
C     FIND NEW NICKNAMES AND USE EXISTING DESTINATIONS.
C
      DO 22 NEW = N1,NZ
      NIC, LDES = NIX(NEW)                This may be either nickname or destination. (A
      IF(NIC.GT.0) GO TO 20               multiple assignment - NIC, LDES take same value)
      IF(MAXNIC+NIC.LT.0) MAXNIC = -NIC   Thus MAXNIC becomes the largest nickname.
      DO 14 LDES = 1,MAXPA                Seek LDES i.e. the destination by looking for
      IF(MVABL(LDES).EQ.0) GO TO 16       a free space in Grandpa.
   14 CONTINUE                            If it reaches CONTINUE a new space is needed,
   16 MVABL(LDES) = NIC                   and automatically LDES becomes MAXPA + 1. We
      IF(LDES.GT.MAXPA) MAXPA = LDES      now put new nickname into Grandpa.
      JWHERE = 7
      IF(MAXPA.GT.MVEND) GO TO 130        Increase dimension of MVABL.
      KOUNT = 1000                        Counts 1000 for each appearance plus one.
C
C     RECORD FIRST, LAST AND INTERMEDIATE APPEARANCES.
C
      DO 18 LAS = NEW,NIZZ                This loop could profitably be written
      IF(NIX(LAS).NE.NIC) GO TO 18        in machine code.
      NIX(LAS) = LDES                     Replace nickname by positive destination.
      KOUNT = KOUNT + 1000
      LAST = LAS                          Becomes position of last appearance in NIX.
   18 CONTINUE
      NIX(LAST) = LDES + 1000             This labels the last appearance.
      LDES, NIX(NEW) = LDES + KOUNT       This labels the first appearance.
   20 LDEST(NEW-N1+1) = LDES              This provides the permanent record.
   22 CONTINUE
      N1 = NEW                            Here N1 becomes the previous NZ plus 1.
```

```
C
C     RECONSTRUCT ELEMENT NICKNAMES AND COUPLE WITH DESTINATION VECTORS,
C
C     AND INITIAL ELEMENT STIFFNESS AND LOAD DATA.
C
      DO 24 KL = 1,KUREL                    Subroutine CODEST interprets the
      CALL CODEST                           coded destinations.
      NIC, LVABL(KL) = -MVABL(LDES)         Here we must reconstruct LVABL.
C     (COLLECT ELEMENT COORDINATES IN ELCOR, USING VALUE OF NIC)
      IF(NSTRES.NE.0. AND .NSTRES.NE.1) GO TO 24
      MVABL(LDES) = 0                       This is now the last appearance of NIC,
      NVABZ = NVABZ + 1                     so we erase the nickname from Grandpa.
   24 CONTINUE                              NVABZ counts the variables.
CARD INPUT, TAPE                            Now complete element tape by computing
      CALL ELMAK                            element stiffnesses and loads.  (For
   26 CONTINUE                              re-solution we create element data during
      REWIND 1                              back-substitution.)  Rewind to use tape.
C
C     PRE-PROGRAM ENDED AND ELEMENT TAPE WRITTEN.
C
C     ESTABLISH STORAGE REQUIREMENTS AND AREA BOUNDARIES.
C
   34 JWHERE = 8                            Have we exceeded NRMAX right-hand-sides?
      NRHS = NEWRHS                         Current number of right-hand-sides.
      IF(NRHS.GT.NRMAX) GO TO 130
      IF(INITL.EQ.0) WRITE(6,806) NRHS      There is no left-hand-side in re-solution.
  806 FORMAT(14H1RE-SOLVE WITH,I4,3X,3HRHS//)
      NELZ = NFUNC(0,LVMAX+1)*INITL + LVMAX*NRHS
      IF(NELZ.GT.MAXELT) NELZ = MAXELT      End of element record in ELPA.
      NPAR = NFUNC(0,MAXPA+1)*INITL + NELZ  Location just preceding Grandpa's RHS.
      IF(INITL.EQ.0) GO TO 36
      NPAZ = (LVMAX+MAXPA)*NRMAX            End of Grandpa fixes start of buffer area for
      IF(NTIREX.NE.0) NPAZ = (MAXNIC+MAXPA)*NRMAX-  equations.  This is inviolate, so
      N = NPAR + MAXPA*(NRHS+1)             we consider the worst re-solution conditions.
      IF(N.GT.NPAZ) NPAZ = N                (Clearly MAXNIC excceds LVMAX.)
      NBAX0 = NPAZ + 1                      NBAXzero just precedes buffer area.
      NBAXZ = NBAX0 + MAXTAP                The end of the equation buffer area.
      IF(NBAXZ.GT.NELPAZ) NBAXZ = NELPAZ
      NBUFFA = NBAXZ - NBAX0                This is the length of buffer actually available.
      JWHERE = 9                            Is it enough? - If not, buy a larger computer.
      IF(NBUFFA.LT.MAXPA+NRMAX+3) GO TO 130
      NRUN0 = NPAZ - NRMAX*MAXPA            NRUNzero just precedes the running variables.
   36 DO 38 I = 1,NBAX0 + INITL*NBUFFA      We must not zero buffer area prior to the
      ELPA(I) = 0.0                         re-solution.
   38 CONTINUE
      KURPA = 0                             Becomes the current size of Grandpa.
C
C     SEEK AND ASSEMBLE NEW ELEMENT.
C
      DO 92 NELEM = 1,NELEMZ                Assemble and reduce for each element in turn.
C-TAPE
      IF(INITL.EQ.0) BACKSPACE 1            In re-solution, elements are in reversed order.
      READ(1) KUREL, LPREQ, (LVABL(I), LDEST(I), I = 1,KUREL),
     1   LZ, (ELPA(I), I = 1,LZ)            Element data with associated equations.
C     (FOLLOWED BY THE ELEMENT COORDINATES IN THE SAME TAPE RECORD.)
      IF(INITL.EQ.0) BACKSPACE 1
      WRITE(2) KUREL, LPREQ, (LVABL(I), LDEST(I), I = 1,KUREL),
     1   IBA, (ELPA(I), I = NBAX0,IBA)      Create tape ready for back-substitution.
C     (FOLLOWED BY THE ELEMENT COORDINATES IN THE SAME TAPE RECORD.)
```

```
      JWHERE = 10                          Is the element stiffness segment too long?
      IF(LZ.GT.NELZ. OR .LZ.LE.0) GO TO 130
      WRITE(6,808) NELEM
  808 FORMAT(/8H ELEMENT,I4/)
      WRITE(6,800) (ELPA(I), I = 1,LZ)
      WRITE(6,810) KUREL, LPREQ, (LVABL(I), I = 1,KUREL)
  810 FORMAT(/8H KUREL =,I4,4X,7HLPREQ=,I4//12H NICKNAMES =,(15I7))
C     WRITE(6,812) (LDEST(I), I = 1,KUREL)
C812  FORMAT(/21H DESTINATION VECTOR =,(15I7))   IBA becomes position in equation.
      IBA = NBAX0                          Because buffer area cleared for new element.
      NEW = 1                              Triggers tape read for first equation needed.
      L = 0
      DO 40 KL = 1,KUREL                   Find destinations ready for element
      CALL CODEST                          assembly, and store them temporarily
      MVABL(LDES) = LVABL(KL)              in LVABL.  We keep nicknames in MVABL.
      LVABL(KL) = LDES
      IF(LDES.GT.KURPA) KURPA = LDES       Grandpa may need enlarging for new intake.
   40 CONTINUE
      DO 66 LHSRHS = 2-INITL, 2            Process LHS and RHS in turn, except in re-solution.
      LHS = 2 - LHSRHS                     LHS is 1 for left-hand-side process.
      IRHS = 1 - LHS                       IRHS is 1 for right-hand-side process.
      DO 64 KL = 1, LHS*KUREL+IRHS*NRHS    Telescope LHS and RHS assembly processes.
      GO TO (42,44), LHSRHS
   42 KG = LVABL(KL)
      MG0 = NFUNC(0,KG) + NELZ             A starting location within Grandpa.
      GO TO 46
   44 MG0 = (KL-1)*MAXPA + NPAR            The same for right-hand-side assembly.
   46 DO 62 IL = 1, LHS*KL+IRHS*KUREL
      IG = LVABL(IL)
      L = L + 1                            Will become position in element segment in ELPA.
   48 CE = ELPA(L)                         This is the element contribution.
      GO TO (50,56), LHSRHS
   50 IF(KG-IG) 52,54,56                   Lower or upper triangle, or diagonal term?  (If
   52 MG = NFUNC(KG,IG) + NELZ             two nicknames are the same in the element, their
      GO TO 58                             off-diagonal contribution to Grandpa's diagonal
   54 IF(KL.NE.IL) CE = CE+CE              must be doubled, as in this statement.)
   56 MG = MG0 + IG
   58 IF(L.LE.LZ) GO TO 60                 Do we need a new tape record for a fragmented
C-TAPE                                     element?  This would entail an element
      IF(INITL.EQ.0) BACKSPACE 1           continuation record.  The elements are in
      READ(1) LZ, (ELPA(I), I = 1,LZ)      reverse order in the re-solution mode.
      IF(INITL.EQ.0) BACKSPACE 1
      WRITE(6,814)
  814 FORMAT(/21H ELEMENT CONTINUATION)
      WRITE(6,800)(ELPA(I), I = 1,LZ)
      JWHERE = 11                          Is the element continuation segment longer
      IF(LZ.GT.NELZ. OR .LZ.LE.0) GO TO 130    than allowed?
      L = 1                                Start again to use the new element segment.
      GO TO 48
   60 ELPA(MG) = ELPA(MG) + CE             At last we assemble the element coefficient.
   62 CONTINUE
   64 CONTINUE
   66 CONTINUE
      JWHERE = 12                          But were the correct number of terms assembled?
      IF(L.NE.LZ) GO TO 130
C     WRITE(6,816)
C816  FORMAT(/8H GRANDPA)
C     WRITE(6,800)(ELPA(I), I = NELZ+1,NPAZ)
C
C     ELIMINATE VARIABLE IN POSITION LDES, AND WRITE EQUATION FOR TAPE.
C
```

```
      DO 90 KL = 1,KUREL                    Look through current element variables.
      CALL CODEST                           - Are there any ready for elimination?
      IF(NSTRES.NE.0. AND .NSTRES.NE.1) GO TO 90
   68 NDEQN = IBA + KURPA + NRMAX + 3       The end of the equation for back-substitution.
      IF(NDEQN.LE.NBAXZ. AND .NEW.EQ.0) GO TO 70        - but only if there is room.
C-TAPE
      IF(NEW.EQ.0) WRITE(2) IBA, (ELPA(I), I = NBAXO,IBA)
      IBA = NBAXO                           Start again in new block of equations.
      NEW = 0                               De-trigger if sensitised for tape read.
      IF(INITL.NE.0) GO TO 68               We must now re-calculate NDEQN.
C-TAPE                                      In the re-solution we must read fresh
      BACKSPACE 1                           equations whenever we write the current
      READ(1) NBZ, (ELPA(I), I = NBAXO,NBZ)       block on tape.
      BACKSPACE 1
C     WRITE(6,818)(ELPA(I), I = NBAXO,NBZ)
C818  FORMAT(/18H RAW EQUATIONS ARE/(12F10.5))
      GO TO 68                              IBDIAG will be the position of the diagonal
   70 IBDIAG, NDIAG = IBA + LDES            term in the equation, and NDIAG the position
      IF(INITL.NE.0) NDIAG = NFUNC(0,LDES+1) + NELZ     in Grandpa.  We need NDIAG in
      PIVOT = ELPA(NDIAG)                   the first solution, IBDIAG in the re-solution.
      ELPA(NDIAG) = 0.0
      JWHERE = 13                           A perfect "flying structure" (unearthed) is
      IF(PIVOT.EQ.0.) GO TO 130             rare, but we test for this mistake here.
      MGZ = NELZ                            Will become the end of the innermost loop.
      JGZ = KURPA                           Last column in Grandpa to be processed.
      IBO = IBA                             Preserve initial value to calculate NDELT below.
      IF(INITL.EQ.0) IBA = IBA + KURPA      We only process RHS in re-solution.
      DO 86 LHSRHS = 2-INITL, 2             Process LHS, then process RHS.
      IF(LHSRHS.EQ.2) JGZ = NRHS
      DO 84 JG = 1,JGZ                      Proceed along equation.
      IBA = IBA + 1
      GO TO (72,76), LHSRHS                 This will define start of the innermost loop.
   72 MGO = MGZ
      MGZ = MGO + JG
      IF(LDES.GT.JG) GO TO 74               Position of coefficient in Grandpa for the
      MG = MGO + LDES                       equation for back-substitution.
      GO TO 78
   74 MG = NFUNC(JG,LDES) + NELZ
      GO TO 78
   76 MGO = (JG-1)*MAXPA + NPAR
      MG = MGO + LDES
      MGZ = MGO + KURPA
   78 NDELT = IBO - MGO                     Quantity needed for standardised loop.
      CONST, ELPA(IBA) = ELPA(MG)           Put coefficient into equation.
      IF(CONST.EQ.0.) GO TO 84              Do this to save computer time only.
      CONST = CONST/PIVOT                   Another quantity needed for innermost loop.
      ELPA(MG) = 0.0                        Leave behind zero in Grandpa.  Calculate
      IF(INITL.NE.LHSRHS) GO TO 80          roundoff criterion, in first solution only.
C
C     SIMULTANEOUSLY, CREATE A SIMPLE ROUNDOFF CRITERION.
C
      MG = NPAR + NRHS*MAXPA + JG
      ELPA(MG) = ELPA(MG) + ELPA(MGZ)**2
C
   80 DO 82 I = MGO+1, MGZ                  } Standardised innermost
      ELPA(I) = ELPA(I) - CONST*ELPA(I+NDELT)} loop, obviously suitable
   82 CONTINUE                              } for writing as a routine
C                                           } in the basic machine code.
   84 CONTINUE
   86 CONTINUE
```

```
      ELPA(IBDIAG) = PIVOT                    Replace diagonal coefficient that was made zero.
      IBA = NDEQN                             Prepare for the next equation.
      ELPA(IBA) = KURPA                       Current size of Grandpa, i.e. equation length.
      ELPA(IBA-1) = LDES                      Destination gives position of diagonal term.
      ELPA(IBA-2) = MVABL(LDES)               Nickname of variable now being treated.
      IF(INITL.EQ.0) GO TO 88                 Examine roundoff criterion, first solution only.
      MG = NPAR + NRHS*MAXPA + LDES
      CRIT = SQRT(ELPA(MG))/ABS(PIVOT)        This is the simple roundoff criterion.
      ELPA(MG) = 0.0                          Leave zero in the location reserved for the
      JWHERE = 14                             next accumulation, for another criterion.
      IF(CRIT.GT.1.0E8) GO TO 130             Disaster - flying structure or mechanism.
      JWHERE = 15                             Possible roundoff trouble: this gives a
      IF(CRIT.GT.1.0E4. OR .PIVOT.LT.0.)      warning message to help identify the cause, but
     1   WRITE(6,834) JWHERE, NIC, CRIT, PIVOT  without stopping the computer.
C     WRITE(6,820)
C820  FORMAT(/18H AFTER ELIMINATION)
C     WRITE(6,816)
C     WRITE(6,800)(ELPA(I), I = NELZ+1, NPAZ)
C     WRITE(6,822)
C822  FORMAT(/10H EQUATIONS)
C     WRITE(6,800)(ELPA(I), I = NDEQN-KURPA-NRMAX-2, NDEQN)
   88 MVABL(LDES) = 0                         Delete from list of Grandpa's nicknames.
      IF(MVABL(KURPA).NE.0) GO TO 90          Can we now work with a smaller Grandpa? (New
      KURPA = KURPA - 1                       zero may expose a block of further zeros.)
      IF(KURPA.NE.0) GO TO 88                 This particular safeguard should never be
   90 CONTINUE                                needed: extra safety for foolproof program.
   92 CONTINUE
      DO 94 I = 1,NELZ+NTIREX                 Zero latest element input, so that for
      ELPA(I) = 0.0                           entire output any nickname not used is
   94 CONTINUE                                associated with zero displacements.
C
C     BACK-SUBSTITUTE INTO EQUATIONS, TAKEN IN REVERSE ORDER.
C
C-TAPE
      IF(INITL.NE.0) REWIND 1
      INITL = 0
CARD INPUT                                    We are already effectively in re-solution phase.
      READ(5,900) NEWRHS
      WRITE(6,824) NEWRHS                     Number of right-hand-sides in next re-solution.
 824  FORMAT(//42H BACK-SUBSTITUTION STARTS, AND NEXT NO. OF,
     1   12H RHS WILL BE,I4/)
      NBZ = IBA
      NEQ = NVABZ                             Actual effective end of equation buffer block.
      LPREQ = LCUREQ                          This will count the equations backwards.
      NELEM = NELEMZ                          We set the first value: later values from tape.
  100 IF(IBA.NE.NBAX0) GO TO 102              The program must also count elements backwards.
C-TAPE                                        Do we need another block of equations yet?
      BACKSPACE 2
      READ(2) NBZ, (ELPA(I), I = NBAX0,NBZ)
      BACKSPACE 2
      IBA = NBZ
  102 KURPA = ELPA(IBA)                       We start at the end of the last equation.
      LDES = ELPA(IBA-1)                      This gives us the length of the equation.
      NIC = ELPA(IBA-2)                       This gives the position of the diagonal term.
      IBAR = IBA - NRMAX - 3                  This gives the nickname of the current variable.
      IBA = IBAR - KURPA                      Find where the RHS of the equation starts.
      IBDIAG = IBA + LDES                     Position preceding the start of the equation.
      PIVOT = ELPA(IBDIAG)                    Position in ELPA of the diagonal coefficient.
C     WRITE(6,822)                            This was the pivot in the reduction process.
C     WRITE(6,800)(ELPA(I), I = IBA+1,NBZ)
```

```
      ELPA(IBDIAG) = 0.0                    Set the diagonal zero to simplify the coding.
      DO 106 J = 1,NRHS                     Deal with each right-hand-side in turn.
      MGO = NRUNO + (J-1)*MAXPA             MGzero is the start of the running variables
      MGZ = MGO + KURPA                     for the current right-hand-side.
      CONST = ELPA(IBAR+J)                  Needed for the standardised innermost loop.
      NDELT = IBA - MGO                     Also needed.
C
      DO 104 I = MGO+1, MGZ                 } Innermost loop, suitable for
      CONST = CONST - ELPA(I)*ELPA(I+NDELT) } writing as a special routine
  104 CONTINUE                              } in some basic machine code.
C
C     PLACE ANSWERS AND PREPARE FOR NEW ITERATIVE LOOP.
C
      ANSWER, ELPA(MGO+LDES) = CONST/PIVOT  Place answers in the running variables.
      IF(NTIREX.NE.0) ELPA(NIC) = ANSWER    Also, for entire output, in position NIC.
      NIC = NIC + MAXNIC
  106 CONTINUE
      ELPA(IBDIAG) = PIVOT                  Replace diagonal coefficient which was set
C-TAPE                                      to zero.  In last re-solution NEWRHS is negative
      IF(IBA.EQ.NBAXO. AND .NEWRHS.GT.0)    and it is therefore not worthwhile to write
     1   WRITE(1) NBZ, (ELPA(I), I = NBAXO,NBZ)     tape.
C     WRITE(6,826)(ELPA(I), I = NBAXO,NBZ)
C826  FORMAT(/35H EQUATIONS AS WRITTEN ON TAPE 1 ARE/(12F10.5))
      NEQ = NEQ - 1                         Number of variables yet to be calculated.
  108 IF(NEQ.NE.LPREQ) GO TO 100            Otherwise we shall need a new element.
C-TAPE
      BACKSPACE 2
      READ(2) KUREL, LPREQ, (LVABL(I), LDEST(I), I = 1,KUREL),
     1   NBZ, (ELPA(I), I = NBAXO,NBZ)
C     (FOLLOWED BY THE ELEMENT COORDINATES IN THE SAME RECORD.)
      BACKSPACE 2
      IBA = NBZ                             Start again at the end of the last equation.
      IF(NTIREX.NE.0) GO TO 114
C     (NOW PLACE ANSWERS FOR OUTPUT BY ELEMENTS.)
      DO 112 KL = 1,KUREL
      CALL CODEST                           in order to find element variables in
      NRUN = NRUNO + LDES                   vector of running variables.
      DO 110 L = KL, NRHS*KUREL, KUREL      Position in element output in ELPA.
      ELPA(L) = ELPA(NRUN)
      NRUN = NRUN + MAXPA                   Required position amongst running variables.
  110 CONTINUE
  112 CONTINUE
      WRITE(6,828) NELEM
  828 FORMAT(/17H ANSWERS, ELEMENT,I4/)
      WRITE(6,810) KUREL, LPREQ, (LVABL(I), I = 1,KUREL)
      WRITE(6,800)(ELPA(I), I = 1,KUREL*NRHS)
CARD INPUT, TAPE
  114 CALL ELMAK
      DO 116 KL = 1,KUREL                   Look again through the element variables: is
      CALL CODEST                           it the last appearance of the node in the
      IF(NSTRES.LE.0) GO TO 116             back-substitution, i.e. first in reduction?
      NIC = LVABL(KL)
C     (FIND AVERAGE OF THE (NSTRES) VALUES OF STRESS AT NODE NIC.)
  116 CONTINUE
      NELEM = NELEM - 1                     Pass on to the next element
      IF(NELEM.NE.0) GO TO 108              - if there is one.
      IF(NTIREX.NE.0) WRITE(6,830)(ELPA(I), I = 1,MAXNIC*NRHS)
  830 FORMAT(/17H ENTIRE OUTPUT IS/(7F17.9))
      IF(NEWRHS) 2,140,34                   New problem, finish, or re-solution respectively
```

```
C      ERROR DIAGNOSTICS AND FINISH.
C
  130 WRITE(6,832)
  832 FORMAT(/6H ERROR)
      WRITE(6,834) JWHERE, NIC, CRIT, PIVOT, LZ, NELZ, NELEM, NRHS,
     1   NBUFFA, LVMAX, NIZZ, NELPAZ, LVEND, MVEND, NIXEND
      A = 0.0                              ,    An artifice to produce trace, etc. for an
      A = 1.0/A                                 error detected by the program.
  140 STOP
  800 FORMAT(5X,12F9.5)
  834 FORMAT(/9H JWHERE =,I3,5X,5HNIC =,I4,5X,6HCRIT =,E9.2,3X,
     1   7HPIVOT =,E12.4,3X,4HLZ =,I5,11X,6HNELZ =,I5/
     2   8H NELEM =,I4,5X,6HNRHS =,I3,5X,8HNBUFFA =,I6,4X,
     3   7HLVMAX =,I5,10X,6HNIZZ =,I5,9X,8HNELPAZ =,I5/
     4   8H LVEND =,I4,5X,7HMVEND =,I4,3X,8HNIXEND =,I6)
  900 FORMAT(40I0)
      END

      SUBROUTINE ELMAK  to demonstrate the program with card input.
C
      DIMENSION EL(100)
      COMMON ELPA(3000), LVABL(30), LDEST(30), MVABL(60), ELCOR(3,20),
     1   INITL, NTIREX, NEWRHS, NELEM, NELEMZ, KUREL, LPREQ,
     2   LZ, NELZ, NBAX0, NBZ, KL, LDES, NSTRES
C
C      IN THIS SECTION CALCULATE ELEMENT STRESSES.
C
      IF(NEWRHS.LE.0) RETURN
C     WRITE(6,802)
C802  FORMAT(/41H ELEMENT SEGMENTS AS ORIGINALLY ON TAPE 1)
C
C      NOW CALCULATE ELEMENT RESIDUALS.
C
CARD INPUT
    4 READ(5,900) NGUIDE, LZ, (EL(I), I = 1,LZ)
  900 FORMAT(2I0,40F0.0)
C     WRITE(6,804) NGUIDE, LZ, (EL(I), I = 1,LZ)
C804  FORMAT(/9H NGUIDE =,I2,4X,4HLZ =,I4,4X,5HEL IS//(5X,12F9.5))
      NGUID = IABS(NGUIDE)
      GO TO (1,2), NGUID
C-TAPE
    1 WRITE(1) KUREL, LPREQ, (LVABL(I), LDEST(I), I = 1,KUREL),
     1   LZ, (EL(I), I = 1,LZ)
C     (FOLLOWED BY THE ELEMENT COORDINATES IN THE SAME TAPE RECORD.)
      GO TO 3
C-TAPE
    2 WRITE(1) LZ, (EL(I), I = 1,LZ)
    3 IF(NGUIDE.GT.0) GO TO 4
      RETURN
      END
      FINISH

      DATA FOR SAMPLE PROBLEM.
1  0   2  10  20
3    2 4 5
3    3 4 5
1    3
0    0
-1  9  2. 0. 2. 0. 0. 1.    1. 2. 3.
1    4    3. 2. 4. 0.
-2    5    1. 2.     2. 3. 4.
-1    2     6.    1.
0
```

APPENDIX III

Specimen data for demonstration ELMAK

NEWRHS = 1, NTIREX = 0, NRMAX = 2, MAXTAP = 10, MAXELT = 20, for first problem.

KUREL	LVABL	
3	2, 4, 5	Element labels
3	3, 4, 5	
1	3	
0	0	Pseudo-element to terminate data

NGUIDE†	LZ	EL (right hand sides bracketed for clarity)
−1	9	2., 0., 2., 0., 0., 1., (1., 2., 3.) One segment only
1	4	3., 2., 4., 0. First element data segment
−2	5	1., 2., (2., 3., 4.) Element continuation segment
−1	2	6., (1.) One segment only

The problem that has now been presented is:

$$2x_2+0x_3+0x_4+0x_5 = 1$$
$$0x_2+9x_3+2x_4+0x_5 = 3$$
$$0x_2+2x_3+6x_4+ x_5 = 5$$
$$0x_2+0x_3+ x_4+3x_5 = 7$$

NEWRHS = 2, for re-solution of first problem

NGUIDE	LZ	EL (right hand sides bracketed for clarity)
−1	2	(4.), (7.) Elements in reverse order
−1	6	(3., 4., 2.), (0., 8., 3.) Penultimate element
2	3	(1., 6., 5.) Continuation segment presented first
−1	3	(4., 7., 2.) 'First' segment, associated with BED on tape

The problem that has now been presented is:

$$2x_2+0x_3+0x_4+0x_5 = 4, 1$$
$$0x_2+9x_3+2x_4+0x_5 = 7, 7$$
$$0x_2+2x_3+6x_4+ x_5 = 11, 14$$
$$0x_2+0x_3+ x_4+3x_5 = 4, 8$$

NEWRHS = −1 now directs program to a second problem.

NEWRHS = 1, NTIREX = 1, NRMAX = 1, MAXTAP = 40, MAXELT = 20, for second problem.

† NGUIDE is only an artifice for demonstrating the program with card input, and its equivalent would not arise in a larger program. If it is negative, the segment it introduces is the last for the element. If it is positive, the segment it introduces does not complete the data for the element, i.e. there is another segment to follow. If it is ± 1, we have an element segment which goes on tape with the basic element data, and must be the first segment to be read from tape for that element. If it is ± 2, we have a continuation record, which goes on tape alone. In the initial solution four segments comprising a single C^e would be introduced by the NGUIDE values 1, 2, 2, −2. In the re-solution mode the record blocks must be written on tape, and therefore presented on cards, in reversed order, and four segments would be introduced by NGUIDE = 2, 2, 2, −1.

KUREL	LVABL	
3	5, 89, 5	One element only: note repeated nickname
0	0	Pseudo-element to terminate data
NGUIDE	LZ	EL (right hand sides bracketed for clarity)
−1	9	2., 2., 4., 1., 2., 2., (1., 2., 3.) One segment only.

The new problem is expressed by the expanded element data

$$\begin{bmatrix} 2, & 2, & 1 \\ 2, & 4, & 2 \\ 1, & 2, & 2 \end{bmatrix} \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix}$$

or, after assembly, in the final form

$$6x_5 + 4x_{89} = 4$$

$$4x_5 + 4x_{89} = 2$$

The coefficient 1 contributes twice: see 'Assembly' in Appendix I for further discussion.

NEWRHS = 0 now means that there are no more problems.

APPENDIX IV

Treatment of different forms of constraint

1. The simplest case is $x_i = 0$.
   (a) This is most easily implemented by deleting x_i from the list of variables.
   (b) In structural work, however, we often require the 'reaction to earth', R_i, defined by

$$C_{i1}x_1 + C_{i2}x_2 + \ldots + C_{ii}x_i + \ldots = X_i + R_i \tag{5}$$

The easiest technique is to add a large number, 10^{20} say, to C_{ii} in equations (1) giving the modified equation

$$C_{i1}x_1 + \ldots + (C_{ii} + 10^{20})x_i + \ldots = X_i \tag{6}$$

Equations (1) remain well-conditioned, and yield x_i of order 10^{-20} and the other x with errors of order 10^{-20}. On subtracting equation (6) from (5) we obtain

$$R_i = -10^{20}x_i$$

2. x_i may have the prescribed non-zero value p_i.
   (a) If R_i is not required, we normally add 10^{20} to C_{ii} and $10^{20}p_i$ to X_i in equations (1).
   (b) If R_i is required, we change the origin of x_i. That is, we replace x_i by $p_i + x_i$, so that x_i becomes zero, and proceed as in 1(b). Writing $p_j = 0$ where x_j is not prescribed, equations (1) become

$$Cx = X - Cp \tag{7}$$

3. If the elements form an interference fit at a given node, equations (7) may be interpreted as contributions from an element, with p_i varying from one element to another. Thus the right hand side contributions are modified by $C^e p^e$. If the node is otherwise free, constraint 1 is not applied.

4. If the nodal reactions between elements or substructures are required, the element contributions to (5) are written

$$C^e x = X^e + R^e \tag{8}$$

so that the vectors $\mathbf{R}^e$ can be found directly.

5. General linear constraints on the stationary value of a quadratic function. Equations (1) may be regarded as minimizing $U = \frac{1}{2}x^T Cx - x^T X$. To implement the constraints $Qx = q$, Lagrange multipliers λ may be introduced, giving the constrained minimum condition of U as the unconstrained stationary condition of $U^* = U + \lambda^T\{Qx - q\}$:

$$\begin{Bmatrix} \partial U^*/\partial x_i \\ \partial U^*/\partial \lambda_j \end{Bmatrix} = \begin{bmatrix} C & Q^T \\ Q & 0 \end{bmatrix} \begin{Bmatrix} x \\ \lambda \end{Bmatrix} - \begin{Bmatrix} X \\ q \end{Bmatrix} = 0 \tag{9}$$

The additional variables $\lambda_j = -\partial U/\partial q_j$ may be valuable by-products.[8] Although the matrix in equations (9) is not positive-definite, symmetric elimination is satisfactory if the x are eliminated before the λ, because the equations reduce to

$$-QC^{-1}Q^T\lambda = q - QC^{-1}X$$

and the matrix on the left is negative-definite.

If for example Q were a full matrix, the banding in (9) would be completely destroyed. However, the frontal technique gives a satisfactory performance with the following algorithm. If λ_j constrains $x_k, x_l, x_m, \ldots$, then λ_j is eliminated immediately after the last of $x_k, x_l, x_m, \ldots$. For when λ_1 is eliminated in (8), the remaining coefficients of the corresponding $x_k, x_l, x_m, \ldots$ in C are already zero, so that the pivotal value when λ_1 is eliminated immediately is the same as it would be if elimination were delayed as in (8). By eliminating λ_1 we have imposed the first constraint, as if by substituting $x_k = Q_{k1}^{-1}(q_l - Q_{l1}x_l - Q_{m1}x_m \ldots)$, so that we may now treat λ_2 as if it were the first. The algorithm thus succeeds. From the section on *The Frontal Technique*, para 1, it follows that the coefficients of the constraint equation need not all be in core simultaneously, but may be introduced piecemeal as the front progresses.

If C is semi-definite but a known submatrix C' is positive-definite, inversion is satisfactory if C' is inverted first, then the partitions involving the λ, and finally the remainder of C. Cases arise frequently but the crucial question of pivoting order escapes comment.

The method of Lagrange multipliers is recommended as both general and tidy, but some engineers regard it as obscure.

REFERENCES

1. C. A. Cornell, H. F. Reinschmidt and J. F. Brotchie, 'A method for the optimum design of structures', Paper presented at *Int. Symp. (electl) digtl. Comput. Struct. Engng*, Newcastle upon Tyne (July 1966).
2. B. M. Irons, Proceedings at above conference, Published by Department of Civil Engineering, University of Newcastle upon Tyne, 1966, p. 82.
3. G. E. Forsythe, 'Today's computational methods of linear algebra', *SIAM Rev.*, **9**, 489–515 (1967).
4. I. Ergatoudis, B. M. Irons and O. C. Zienkiewicz, 'Three-dimensional analysis of arch dams and their foundations', Symposium on Arch Dams, *Instn. civ Engng*, London (1968).
5. B. E. Greene, D. R. Strome and R. C. Weikel, 'Application of the stiffness method to the analysis of shell structures', *Am. Soc. mech. Engrs*, 61-AV-58 (1961).
6. R. J. Melosh and R. M. Bamford, 'Efficient solution of load-deflecting equations', *J. Am. Soc. civ. Engrs (Struct. Div.)* Paper No. 6510, 661–676 (1969).
7. J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, 1965.
8. B. M. Irons, 'Roundoff criteria in direct stiffness solutions', *AIAA Jnl.*, **6**, 7, 1308–1312 (1968).
9. B. M. Irons, 'Lagrange multipliers in structural analysis', *AIAA Jnl.*, **3**, 6, 1172 (1965).