Introduction to NETFORM

Mr. Jaap Smit
Twente University of Technology
Department of Electrical Engineering
P.O.Box 217, Enschede
The Netherlands

ABSTRACT.
Netform is an easy to use, interactive, language for symbolic mathematical computation. The well known*, *Formac* package, which uses the normal representation of formula's, forms the basis of the system. This basis is extended by means of:
1° A linear equations solver, which offers two different possibility's to describe a system of linear equations.
2° The description of linear (electrical) networks based on the network structure.
In this paper a more detailed description of the just mentioned extensions is given. It will be shown that symbolic mathematical computation can be a very attractive and powerful instrument for the analysis of electrical network.
At this moment the system is implemented on an IBM 360-50 computer under the MVT operating system and serves 3 2260 display's in a 240k region. Batch processing is possible in order to be independent of the display's. A file mechanism makes off line preparation of a session possible.
For documentation and distribution you may contact the author.

1. The natural input to the system

1.1. Introduction.
While the main goal of this introductionary section is to show how easy NETFORM is to use (and to learn) a sample session is presented.

1.2. Basic FORMAC capability's of NETFORM.
In the following session NETFORM will be demonstrated. The actual computer output is used to demonstrate the system. However to make distinction between the user command (typed in by the user) and the system reply "command" and "reply" are placed in the margin.

```
1: command: PRINT X=EXPAND((A+B)**2);
   reply    : X= 2 B A + A**2 + B**2

2: command: PRINT Y=COEFF(X,A);
   reply    : Y = 2 B

3: command: SET INT;
            PRINT FAC(16)/FAC(15);
   reply    : . . = 16

4: command: LET FNC(F) = A*$(1)**4 + B * $(1)**3 + C * $
            (1)**2 + D*$ (1) +E;
            PRINT F(N +1);
   reply    : . . = E + (N + 1)**2  C + (N +1) D +  (N + 1)**4  A +
            (N + 1) ** 3 B

5: command: PRINT C1=COEFF (# $, N ** 4),
                    C2=COEFF (EXPAND (# $), N ** 4);
   reply    : C1 = 0
            C2 = A
```

Explanation:

1: Application of the multinomial and distributive laws to $(A + B)^2$.
2: Y is set equal of the coefficient in X of A.
3: The quotient of factorial 16 and factorial 15 is calculated. As the equal sign is omitted, the value is assigned to the so called workspace.
4: A function F with independent variable $ (1) is defined.
F(N +1) is printed, and this new value overrides the old value of the workspace.
5: Here we see demonstrated that the COEFF routine works on the upper level of the expression.
$ represents the value of the workspace; F(N + 1).

Most of the NETFORM system presented until now is a direct demonstration of the capability's of FORMAC.
Note that (in 5:) F(N +1) and EXPAND (F (N + 1)) represent different formula's (but equal scalars). If there is a rule which transforms formula's (say E1, E2) into a unique equivalent (say F) then the result (F) is said to be canonical. The (FORMAC) system allows pseudo canonical formula's [1],which may contain canonical subexpressions (elements of the set (F)), but the whole formula is not necessarily canonical. (This is kept under user control).

1.3. The idea's of formula manipulation.
. . . . . the sum of squares.
The goal of this paragraph is to show how easy fairly general results may be obtained with formula manipulation.
An illustrative example is the determination of the generating function

$$\text{of} \quad \sum_{n=1}^{N} n^2$$

If the generating function is called F then the problem is to find a function F with the following property's:

$$n = 1 \rightarrow F(1) = 1 \qquad ①$$
$$n = N - 1 \rightarrow F(N - 1) \qquad ②$$
$$n = N \rightarrow F(N - 1) + N^2 = F(N) \qquad ③$$

The corresponding NETFORM session may have the following continuation:

```
6: command:  SET EXPAND;
             LET ZERO = F(N) - (F(N - 1) + N ** 2);
             PRINT COEFF (ZERO, N ** 4),
                   COEFF (ZERO, N ** 3),
                   COEFF (ZERO, N ** 2),
                   COEFF (ZERO, N),
                   EVAL (ZERO, N, O),
                   F (1) - 1;
reply    : .. = 0
           ---------
           .. = 4 A
           ------------
           .. = - 6 A + 3 B - 1
           ---------------------------
           .. = 2 C + 4 A - 3 B
           ---------------------------
           .. = - C + D - A + B
           ---------------------------
           .. = C + D + E + A + B - 1
           ---------------------------------------
```

Explanation:

6: The flag EXPAND forces the application of the multinomial and distributive laws to each formula. According to ③ the expression ZERO is formed, which is, due to the EXPAND flag, a polynomial in N (compare to 4:) For succesfull induction ZERO should be zero for all N, which means that the coefficients of the polynomial should be zero. These coefficients are listed in the reply 6 . According to ① F (1) - 1 should be zero, and the result is the last part of the reply. Note that we found 5 equations in 5 unknowns, A, B, C, D and E, the solution of which gives the result of the problem (see 1.4). Due the fact that the mathematical problem can be treated symbolic, it is made fairly easy to apply induction and to make calculations which lead to general results. Even a systematic construction (synthesis) is often easily obtained.

1.4. Formulation of linear equations.

..... the sun of the squares.

In this paragraph it will be shown how a system of simultaneous linear equations may be specified in NETFORM. The theory of the linear equation solver will be discussed in chapter 2.

A continuation of the session might be:

```
7: command:  BEGIN;
             EQU COEFF (ZERO, N **3),     A,
                 COEFF (ZERO, N**2),      B,
                 COEFF (ZERO, N),         C,
                 EVAL (ZERO, N,O),        D,
                 F (1) - 1,               E;
             SOLVE A, B, C, D, E;
             CALC;
reply    : A = O
           ---------
           B = - 8
           ---------
           C = - 12
           ------------
           D = - 4
           ----------
           E = O
           ---------
           .. = - 24
           ------------
8: command:  END;
```

Explanation:

7: A block is opened to allow the formulation of a system of simultaneous linear equations. The EQU statement introduces the equations and unknowns to the system. Note that the equations are referenced as the coefficients of the ZERO polynomial and as F (1) - 1.
The statement SOLVE A, B, C, D, E indicates the system that values must be assigned to A, B, C, D, and E resp.
The CALC statement initiates the calculations, resulting in the above values which are in fact the numerators.
The value of the denominator is assigned to the workspace.
Finally the block is closed.

1.5. Discussion.

In te simple example of the sum of squares it was shown that the solution resulted in a number of integer value's. This suggests that this and probably many other problems closely related to symbolic mathematical computation, may be programmed in (for example) FORTRAN.
As an example we find in [2] a program for symbolic network analysis based on a method indicated in [3] by S Sushu and MB Reed.
The main disadvantage of such, in principle numerical systems is the lack of flexibility. Working with a symbolic mathematical system, as NETFORM, however a generated formula can be interpreted and accessed by the user in an interactive environnement.

2. The linear equations solver

2.1. Introduction.

A linear equations solver should be *fast*, should give *canonial expressions* as output, and should be *flexible*, in order to give optimal results in the (interactive) environment of symbolic mathematical computation. This chapter deals with the philosophy of the linear equations solver, and the ways in which the input language communicates with it. Some applications are given in chapter 3.

2.2. Canonical expressions.

If there is a rule which transforms formula's into a unique equivalent then the resulting formula's are said to be canonical. The rules for transformation are always chosen such that the most primitive result is obtained. Because integers are "more primitive" than rationals, X = 5/7 + 2/7 will result in X = 1. The evaluation of the determinant of a matrix does not necessarily lead to a canonical representation.
Example:

$$Y = \begin{vmatrix} A & A \\ B & B \\ X & D \\ C & X \end{vmatrix} = A(-BX^2 + CBD) - B(-AX^2 + CAD)$$

Expansion of the right hand side results however in the integer result Y = 0.
In general noncanonical expressions are confusing and must be avoided. To increase the comprehensibility of results it is also advisable to avoid algorithms which introduce common factors.
A survey of the methods in use is given in table 1, and the property's of the different wellknown methods are listed. Because sparce matrix expansion is well suited for symbolic mathematical computation, this technique will be discussed in 2.3.

TABLE I PROPERTY / METHOD	canonical expressions generated	common factors introduced	Area of use
1° Gaussion elimination [5]	NO	YES	Floating point Numerical calculations
2° Fraction free methods [6]	NO	NO	Rational (integer) calculations Symbolic math. calculations
3° Sparce matrix expansion	canonical in terms of the matrix elements	NO	Symbolic mathematical calculations

2.3. Sparce matrix expansion.

Sparce matrix expansion uses the recursive definition of a determinant in terms of its minors.

A further explanation of the term *sparce* matrix *expansion* will be given on the basis of NETFORMS implementation. The representation of the matrix is chosen in such a way that a optimal use of zero matrix elements could be made during the calculation of the minors. This is achieved by means of a list structure or skeleton with the following property's.

1) Only nonzero matrix elements are used to construct the skeleton.
2) Each matrix element points to the next element in the same column.
3) Each matrix element points to the first element of the next column.

A possible skeleton might be:

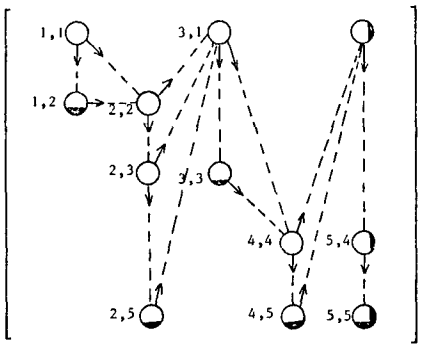

fig. 1

a nonzero matrix element
end of a column
end of a row

As pointed out, the minor of a specific matrix element can easily be found on the skeleton. According to the definition of the determinant one observes that a term of the expansion is represented by a (signed) sequence of minors. As an example one way follow the generation of the first two terms on the skeleton of fig. 1 (see fig. 2 and 3)

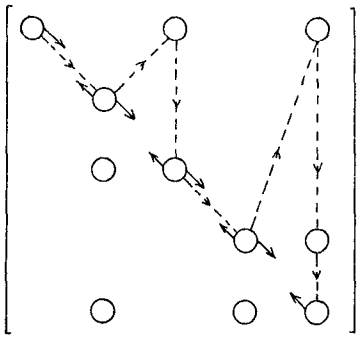

fig. 2

The first complete sequence of minors is represented by the double chain of pointers on the main diagonal. The path followed on the skeleton is dotted in fig. 2. If the sequence of minors is complete then the symbolic matrix elements are linked in order to represent the first term:

$$+ A(1,1) * A(2,2) * A(3,3) * A(4,4) * A(5,5)$$

Not the matrix elements themselves are used to build the formula, but their symbolic value. If the symbolic value is ± 1, then the factor in the term is omitted in the actual formula.

The generation of the second term is now obvious (see fig. 3)

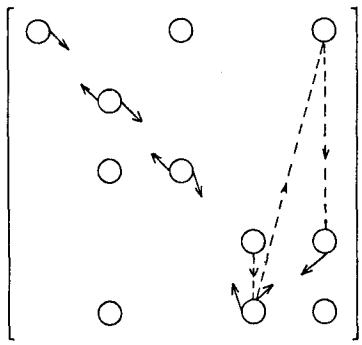

fig. 3

Note that the backwards chain gives the link between the first and the second term. Furthermore it is not necessary ro repeat the whole sign calculation, because a part of the previous calculation may be used.

In order to obtain the desired *expansion* in terms of the value's of the matrix elements the *whole* minor sequence is used to generate the new term.

2.4. Property's and applications of sparce matrix expansion.

To solve an unknown X_i from [A] $\underline{X} = \underline{C}$.

Cramers rule is used which gives:

$$X_i = \frac{\text{Det } [A]\,i}{\text{Det } [A]}$$

As the value of the denominator is the same in all cases, NETFORM calculates and prints it once. The numerators, Det [A] i, are assigned to X_i.

The result of a calculation is canonical if the matrix element A (i, j) represents a variable, or a product of variables.

In special cases a user may prefer a pseudo canonical formula over a canonical formula.

A pseudo canonical formula may be obtained by means of, for example, the assigment of a (symbolic) sum to a matrix element.

The flexibility of sparce matrix expansion covers many fields, as will be shown. In the area of numerical analysis special routines are used to solve special sets of equations in an efficient way.

A nice example is the three diagonal matrix. The use of specialized routines is avoided by sparce matrix expansion, because it uses the property's of a given problem in an optimum way.

Also the scientist may take advantage of the flexibility of sparce matrix expansion, because he no longer needs to minimize the number of equations, which means that he may directly map his problem into equations. In chapter 3 this will be demonstrated for electrical networks.

To be able to rate the speed of sparce matrix expansion at its true value, it will be compared to its competitors.

If trival cases are omitted (i.e. the calculation of a unit matrix) then the methods listed in Table I are roughly speaking arranged in order of speed but also of generality.

If the Fraction Free method is used for symbolic calculations with the EXPAND option then its speed is lost, and the facility of introduction of non-canonical formula's no longer exists.

Mixed methods, which separate the numerical and symbolical parts of a calculation are supposed to be faster, but, the succes of a mixed method depends strongly on the reduction of the number of terms in the answer. Investigations showed that the number of terms did not decrease too much if about half the number of symbolic variables in the answer was replaced by numerical constants. This is evident, because a term only vanishes if all its factors become nummerical constants. This is the reason why until now no attempt is made to incorporate a mixed method.

As was indicated earlier sparce matrix expansion does not introduce common factors. This does not mean that a user may not introduce common factors. One possible way to factorize a matrix into irreducible parts in terms of its matrix elements is to obtain a block traingular form, see [4].

It is evident that factorization, if possible, increases the comprehensibility of the result and also the speed of computation.
That factorization is not necessarily time consuming can be found from [4] which indicates CPU times in the range of seconds for sets of 112 equations.
Conclusion:
There are still improvements possible to the system, this may be achieved by means of "mixed methods" and "factorization". On the other hand the discussed algorithm was used during 3 years of satisfaction, and was recently updated for the NETFORM system.

2.5. NETFORM-instructions to formulate a set of linear equations: Two examples. In this paragraph the facility's of the input language will be demonstrated by means of a continuation of the session. As an example the following set of linear equations will be solved.

$A * X + B * Y = C\,1$

$C * X + D * Y = C\,2$

In the first, and most elementary, case all (nonzero) matrix elements are listed and the function DET, which is added to the FORMAC system, is used.

```
command: LET  A (1,1) = A,A (2,1) = B,A (0,1) = C 1,
              A (1,2) = C,A (2,2) = D,A (0,2) = C 2;
         PRINT X =  DET (A,2,1),
                Y =  DET (A,2,2),
                DET (A,2);
reply    : X = C1D - BC2
           ----------
           Y = AC1 - C2D
           ----------
           . . = AD - BC
           ----------
```

In the above example X,Y denote the numerator of X,Y. The denominator is assigned to the workspace.
In the second case equations are introduced to the system by de EQU statement and the system builds the sparce matrix. The SOLVE statement indicates which unknowns are to be solved. The equations belong to a specific block denoted by a BEGIN and END statement. This construction makes it in an easy way possible to nest sets of linear equations.

```
command: BEGIN;
         EQU  A * X + B * Y - C1, X,
              C * X + D * Y - C2, Y;
         SOLVE X, Y
         CALC;
reply    : X = CID  - BC2
           ----------
           Y = AC1 - C2D
           ----------
           . . = AD - BC
           ----------
command: END;
```

3. Analysis of networks

3.1. Introduction.
In this chapter an introduction will be given to the analysis of (electrical) networks. An important concept introduced in this chapter is the mapping of the physical network into a set of linear equations without any substitutions. This means that the equations cannot be distinguished from the physical network. The fact that he linear equations solver gives canonical expressions without introduction of common factors makes it possible to relate certain property's of the network to the property's of the solution, obtained by the linear equations solver.

3.2. The description of an electrical network via a system of linear equations.
A simple network will be taken as an example to demonstrate the description of an electrical network via a system of linear equations (see fig. 4)
In 3.4. the same network will be taken to demonstrate the description of an electrical network via the network structure.

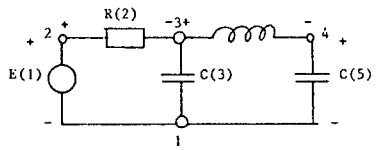

fig. 4

For convenience the nodes and elements are numbered. The following commands describe the network. (The voltage across C(5) is solved).

```
command: BEGIN;
         * ELEMENT EQUATIONS;
         EQU  U (1) - E (1),                  U (1),
              U (2) - I (2) * R (2),          U (2),
              U (3)  * S * C (3) - I (3),     U (3),
              U (4) - I (4) * S * L (4),      U (4),
              U (5) * S * C (5) - I (5),      U (5),
         * CURRENT LAW;
         EQU + I (1) - I (3) - I (5),         I (1),
               I (2) - I (3) - I (5),         I (2),
               I (4) - I (5),                 I (3);
         * VOLTAGE LAW;
         EQU - U (1) - U (2) - U (3),         I (4),
             - U (1) - U (2) - U (4) - U (5), I (5);
         SOLVE  U (5);
         CALC;
reply    : U (5) = - E (1)
           . . = R (2) S C (5) + L (4) S² C (5) + R (2) L (4) C (3) S³
           ----------------------------------------------------------
           C (5) + R (2) C (3) S + 1
           -------------------------
command: END;
```

Explanation:
The element equations describe the behaviour of the elements S stands for the Laplace operator.
The current law describes the fact that the sum of currents through a closed surface is zero.
The voltage law describes the fact that the sum of voltages in a circuit of the network is zero.
The unknowns (in the EQU statements) are listed in an arbitrary order.

3.3. Definition of network elements.
Observation of the preceding example shows that the element equations are characterized by the type of element and the branch number (#BR). NETFORM allows the user to define its own elements. Once defined an element may be referenced by its name and corresponding node pair, as will be shown in 3.4 For convenience the definitions of the elements of an electrical network are already known by the system. Furthermore they are avialable from one of the systems files.
The predefined elements are:

```
command: > DEFN
         *  INPUT FOR THE BATCH VERSION
         *  DEFINE ELEMENTS FOR ELECTRICAL NETWORKS;
         DEFINE;
         U      U (#BR) - E (#BR);
         AM     U (#BR);
         L      U (#BR) - I (#BR) * S * L (#BR);
         I/L    U (#BR) / L (#BR) - I (#BR) * S;
         R      U (#BR) - I (#BR) * R (#BR);
         I/R    U (#BR) / R (#BR) - I (#BR);
         Z      U (#BR) - I (#BR) * Z (#BR);
         I/Z    U (#BR) / Z (#BR) - I (#BR);
         Y1     U (#BR) * Y (#BR) - I (#BR);
         I/Y    U (#BR) - I (#BR) / Y (#BR);
         G      U (#BR) * G (#BR) - I (#BR);
         I/G    U (#BR) - I ( #BR) / G (#BR);
         C      U (#BR) * S * C (#BR) - I (#BR);
         I/C    U (#BR) * S - I (#BR) / C (#BR);
         VM     I (#BR);
         I      I (#BR) - J (#BR);
         END;
         < END . . .
```

Explanation:
> DEFN informs the system that input should come now from file DEFN.
In the DEFINE block the element equations are defined. One element is always known by the system and is called PORT. The element PORT is the only element which introduces no equation to the system.
Related to each element are the unknowns U (#BR) and I (#BR).

3.4. The description of an electrical network via its structure. If all elements are described by type and place, then the whole network is given. This implies that the equations of the current law and the voltage law may be obtained from the network structure, which means a further simplification in the notation.
Now the network of fig. 4 may be described by:

```
command: RESTART;
reply    : TOTAL TIME;
command: *THE SAME PROBLEM AS ABOVE;
         BEGIN;
         U (1,2);
         R (2,3);
         C (3,1);
         L (3,4);
         C (4,1);
         SOLVE U (5);
         CALC;
reply    : U (5) = - E (1)
           ----------------
           . . = R (2) C (3) S + R (2) C (5) S + C (5) L (4) S² + R (2)
           ------------------------------------------------------------
           C (5) L (4) C (3) S³ + 1.
           ------------------------
command: END;
```

Note: The equations generated by the system are exactly the same as those introduced in 3.2.

3.5. Discussion.
In the preceding paragraphs the analysis of an electrical network was explained. In order to rate the symbolic mathematical computation at its true value some of the relations between the generated answer and the network under study will be investigated.

3.5.1. The uniqueness of a solution.
Although it was stated sparce matrix expansion is canonic, and the Cramers rule does not introduce common factors, still the solution of a given problem is determined within, so called, associates.
As an example:
The admittance of two resistors in parallel may be written:

$$\frac{\frac{1}{R(1)}+\frac{1}{R(2)}}{\frac{1}{1}} \quad \text{or} \quad \frac{R(2)+R(1)}{R(2)\,R(1)} \quad \text{or} \quad \frac{1+\frac{R(2)}{R(1)}}{R(1)} \quad \text{or} \quad \frac{1+\frac{R(2)}{R(1)}}{R(2)}$$

while the corresponding entry's in the matrix are:

$$\frac{1}{R(1)},\ \frac{1}{R(2)}, \text{ or: } R(1), R(2) \text{ or: } R(1), 1/R(2) \text{ or: } 1/R(1), R(2)$$

Note that there are no common factors in all four cases.
Note that the last three formula's may be derived from the first by multiplication of numerator *and* denominator by R (1) and/or R (2). If the characteristic equation Det (A) = 0 is taken as a test for singularity then the result is different in all four cases. It is however possible to give a meaning to all those different results. If all entry's of the matrix A are restricted to $(-\infty, +\infty)$, which prevents the numerator to become infinite, then singularity may indeed be tested with the characteristic equations, and all four answers belong to different problems.
Example:
In electrical networks one uses as entry's S * C and S * L.
If the entry's are restricted to $(-\infty, +\infty)$ then the characteristic equation has a unique meaning and the roots of the equation lie in the finite S plane.

3.5.2. Examples of singularity.
Loops of voltage sources and cutsets of current sources give singular solutions.
Let us use NETFORMS notation to prove this statement is the case of 3 voltage sources.

```
EQU U (1)          - E (1), U (1),
      U (2)        - E (2), U (2),
        U (3)      - E (3), U (3),
U (1) + U (2) + U (3)    ,. . . . . .;
```

From the example it can be seen that there are four equations in three unknowns, or if one unknown is added, that there is a column of zero's in A. This results in the singularity (Det (A) = 0).
Even if other equations are added, zero is still a factor in the result as may be shown by means of block triangularization. The same proof holds for cut sets of current sources if voltages are replaced by currents. In a similar way many other examples of singularity may be given.
Note that the uniqueness of the solution implies that if the network is singular, this will be represented by Det (A) =0, and *not* by any other expression which is equivalent to the integer O.

3.5.3. The degree of the characteristic equation.
The maximum degree, with respect to the Laplace operator S, of the characteristic equation is equal to the number of elements, which bear the operator in the element equation.
An interesting point is that the actual degree is *always* given by the NETFORM system. This is a consequence of the fact that the expressions are canonical and that no common factors are introduced.
As an example the steps in automatic simplification, which derate the degree are given:
The characteristic equation is $A S^P + B S^{P-1} + \ldots\ldots\ldots$,
A is an canonical expression equivalent to 0.

$$A S^n + B S^{n-1} + \ldots\ldots$$
$$\downarrow$$
$$0.S^n$$
$$\downarrow$$
$$0 + B S^{n-1}$$
$$\downarrow$$
$$B S^{n-1} + \ldots\ldots$$

Note that rounded numbers give non canonical formula's, because rounding errors may affect the automatic simplification.

3.5.4. The state variables.
A maximum set of linear independent initial conditions of an electrical network is often called a set of state variables
Often used candidates for state variables are: capacitor voltages and inductor currents. Procedures to find a set of state variables are complicated in nature. It will now be demonstrated that the desired information is in fact imbedded in the characteristic equation.
The element equation for the capacitor is:

```
U (#BR) * S * C (#BR) - I (#BR);
```

The element equation for the voltage source which represents the initial condition is:

```
U (#BR)  - E (#BR);
```

If the characteristic equation contains C (#BR) then it may be written:

```
Minor 1 * C (#BR) + Minor 2 * 1
```

If now instead of the element equation for the capacitor, the element equation for the initial condition was used, then the characteristic equation would be: Minor 1 * 1 +Minor 2 * 0 ≡ Minor 1
Proceeding in this way a maximum set of initial conditions may be found.
Note that the inital conditions are linear independent, because the corresponding network is non singular.
Some examples:
1) In the network of 3.4 3 state variables can be found, as indicated by the term of highest degree:
   S^3 C (3) L (4) C (5)
   The state variables are:
   the voltage across C (3) and C (5) the current through L (4)
2) In a loop of 3 capacitors only two state variables may be found. It is impossible to find three state variables because if all capacitors are replaced by voltage sources, then the network which is obtained is singular (see 3.5.2). So the degree of the characteristic equation is reduced to 2. As each combination of two voltages gives a set of state variables, it follows that the coëfficient of S^2 is of the form:
   (C (1) C (2) + C (2) C (3) + C (3) C (1))
   Note: The positive signs are not explained by the above discussion.

CONTENTS:

LITTERATURE:

1 Tobey, R.G. - Bobrow, R.J. - Zilles, S.
Automatic Simplification in FORMAC,
Proc. of fall joint computer conference,
Spartan books vol. 27 nov. 1965

PL/1 - FORMAC Symbolic Mathematics Interpreter 360D - 03.3. 004, 'IBM manual'.

Bobrow, D.G.
Symbol Manipulation languages and techniques,
Proc. of ifip working conference on symbol manipulation languages.

2 J. Mucha.
Zur nummerischen und nichtnummerischen Empfindlichkeits analyse linearer elektrische Netzwerke, Ph. D. thesis Aachen.

3 S. Seshu and M.B. Reed.
Linear graphs and elektrical networks,
Addison-Wesley 61-5309.

4 A.K. Kevorkian and J. Snoek.
Theory and applications of structural analysis in disjointing and constructing hierarchical systems, Shell report.

5 D.K. Faddejew / W.H. Faddejewa.
Numerische methoden der linearen algebra,
VEB deutscher verlag der wissenschaften, Berlin 1964.

6 J. Lipson.
Symbolic methods for the computer solution of linear equations with applications to flowgraphs,
Proceedings of the 1968 summer institute on symbolic mathematical computation, Robert G. Tobey, editor.

ALTRAN Programs for SIGSAM Problem #6

(Continued from page 13)

IV. Remarks

In an attempt to understand the astonishing growth in the CPU time required for higher orders, we gathered some timing statistics from the routines internal to ALTRAN. Surprisingly, 75% of the CPU time is spent in the polynomial "divide-if-divisible" operation, while only 13% is spent in polynomial gcd computations. This suggests that substantial improvements in performance can be gained by using a faster algorithm for this operation (see [2]).

References

1. D. P. McCarthy, Problem #6 - The Kiang Problem, SIGSAM Bulletin No. 28, (December 1973), pg. 12.
2. S. C. Johnson, Sparse Polynomial Arithmetic, in preparation.