Dynamic sparse matrix code as an automatic approach to macromodeling

DYNAMIC SPARSE MATRIX CODE AS AN AUTOMATIC APPROACH TO MACROMODELING

P.M. TROUBORST

DYNAMIC SPARSE MATRIX CODE AS AN AUTOMATIC APPROACH TO MACROMODELING

DYNAMIC SPARSE MATRIX CODE AS AN AUTOMATIC APPROACH TO MACROMODELING

PROEFSCHRIFT

TER VERKRIJGING VAN DE GRAAD VAN DOCTOR IN DE TECHNISCHE WETENSCHAPPEN AAN DE TECHNISCHE HOGESCHOOL EINDHOVEN, OP GEZAG VAN DE RECTOR MAGNIFICUS, PROF. IR. J. ERKELENS, VOOR EEN COMMISSIE AANGEWEZEN DOOR HET COLLEGE VAN DEKANEN IN HET OPENBAAR TE VERDEDIGEN OP DINSDAG 9 JUNI 1981 TE 16.00 UUR

DOOR

PIETER MARINUS TROUBORST

GEBOREN TE GOUDA

DISSERTATIE DRUKKERIJ
wibro
HELMOND. TELEFOON 04920-23981

DIT PROEFSCHRIFT IS GOEDGEKEURD
DOOR DE PROMOTOREN

Prof. Dr.-Ing.J.Jess

en

Prof. Dr.-Ing. O.E.Herrmann

CONTENTS

INTRODUCTION

The growing complexity of electrical circuits increases the computational effort to simulate them. Often it is a task to keep the simulation time within acceptable bounds. The application of different levels of simulation can be considered as one approach to cope with this problem. While the simulation at system level is very global, the simulation becomes more and more detailed going via register transfer level, logic level and gate level to circuit level.

On the other hand successful attempts have been made in various ways to obtain efficient methods for the detailed simulation at circuit level. Many approaches exploit the sparsity of the circuit equations [I.1]. All such methods have the common property that a reduction of the computation time is achieved while preserving the accuracy of the simulation results.

Another method is the exploitation of the latency in electrical circuits [I.2]. Then we avoid computing new values for the response variables of a subcircuit if all excitating variables of that sub-circuit do not change during the preceding time step.

Finally we mention macromodeling as a method to decrease the simulation time. A "model" of a circuit in the most general sense is an algorithm to compute the response of an electrical circuit given certain excitations. In the literature of electrical engineering a model is often specified by a circuit diagram comprising standard symbols for circuit elements. It is then implicitly understood that a standard algorithm is applied to compute the response given certain excitations. Given a model of an electrical circuit a "macromodel" of that same circuit is obtained by deleting instructions and internal variables from the model. Again in the literature models are often specified by simplifying the associated circuit diagrams in a certain way. Obviously macromodeling implies a decrease of the accuracy.

The extensive literature on macromodeling is mainly concerned with macromodels for particular circuits (operational amplifiers, logic gates) and with indications how to design macromodels [I.3]. A different approach is the interactive method proposed by Spence [I.4]. In this approach a model is simplified by deleting elements in the model subject to some constraints. All these methods are dominantly "ad hoc"

The macromodeling approach we study in this thesis differs in two points from previous approaches. Firstly with this approach a macromodel can be obtained completely automatically by a computer. Secondly the simplifications concerning the computation of the response variables are dynamical and are made dependent on the values of appropriate variables. Thus we are able to keep the impact of any possible simplification on the response under control. We may admit simplifications only to the extend that the loss of accuracy is guaranteed to stay within specified limits.

The approach starts from a common description of some subcircuit as used in the transient analysis of nonlinear circuits. We assume that Newton iteration is applied and exploit the fact that the Jacobian of the circuit equations contains constant and variable coefficients. The latter arise for instance from nonlinear equations. The significance of the computations involving some coefficient depends strongly on the value of that coefficient. For the variable coefficients thresholds are computed such that it can be determined whether computations involving such a coefficient are significant. If these computations are not significant then they can be skipped by a novel method called "pivotstep skipping". The thresholds together with the partitioning of the computation for a particular circuit constitute a macromodel of that circuit.

Roughly the thesis consists of two parts. One part, chapters 2 and 3, is concerned with appropriate orderings of the variables and the equations describing the circuit. The second part, chapters 4 to 7, describes the actual method of macromodeling by the skipping of operations.

We use the aspects of circuit simulation discussed in chapter 1 as a reference in the remainder of the thesis. In chapter 2 we present the theoretical basis for an algorithm to determine a "bordered lower triangular form" of a matrix. In chapter 3 we describe the algorithm and give an analysis of its time complexity. We illustrate some properties of the algorithm by examples and present some results.

In chapter 4 we develop the theoretical basis for pivotstep skipping. We indicate how the thresholds can be computed and derive upper bounds for the error of the computed solution under the application of pivotstep skipping.

In chapter 5 we discuss briefly three different implementations of the L\U decomposition of a sparse matrix. The two most diverging methods, the "compiled code approach" and a "linked list approach", are taken to indicate an implementation of pivotstep skipping. The speed up obtained by pivotstep skipping is analysed.

In chapter 6 we analyse four possible ways to organize the fore- and backsubstitution in the case pivotstep skipping is applied. We show how function evaluations can be avoided while detecting the pivotsteps to be skipped. Attention is paid to the relation between the convergence achieved with the method, the obtained accuracy and the number of pivotsteps being skipped.

In chapter 7 we show how pivotstep skipping can be applied if special nonstandard elements are present in the circuit.

[I.1] I.S. Duff, "A survey of sparse matrix research", Proceedings of the IEEE, Vol. 65, pp. 500-535 (1977).

[I.2] N.B. Rabbat, H.Y. Hsieh, "A latent macromodular approach to large-scale sparse networks", IEEE Trans. Circ. Systems, Vol. CAS-23, pp. 745-752 (1976).

[I.3] A.E. Ruehli, R.B. Rabbat, H.Y. Hsieh, "Macromodelling - an approach for analysing large-scale circuits", Computer-Aided Design, Vol. 10, pp. 121-129 (1978).

[I.4] R. Spence, T. Neumann, "On model simplification", IEEE Proc. Int. Symp. Circ. Systems, 1978, pp. 350-353.

1. SOME ASPECTS OF CIRCUIT SIMULATION

1.1. Transient analysis of nonlinear circuits

A time dependent nonlinear circuit can be described by a set of nonlinear differential equations. Transient analysis implies the solution of the equations for a set of time-points $t_0, t_1, t_2, \ldots, t_N$. Integration methods can be applied to cope with the differential equations. Then integration formulas replace the time derivatives, and a set of nonlinear equations for each time-point remains. With the vector of variables $x = (x_1, x_2, \ldots, x_n)^T$ and the differentiable vector functions $s_i(x)$, $i = 1, 2, \ldots, n$, a set of nonlinear equations is described by:

$$s(x) = (s_1(x), s_2(x), \ldots, s_n(x))^T = 0 \tag{1.1}$$

Newton iteration can be applied to determine a solution of the equations. Let x^ι denote the ι^{th} iterate for $\iota \geq 1$ while x^0 is some estimate of the solution. The Jacobian of $s(x)$ for $x = x^\iota$ is denoted by the matrix $A^\iota \triangleq A(x^\iota)$:

$$a_{ij}^\iota \triangleq a_{ij}(x^\iota) \triangleq \left.\frac{\partial s_i(x)}{\partial x_j}\right|_{x=x^\iota} \tag{1.2}$$

Using x^0 as initial value and assuming that the Jacobian is nonsingular Newton iteration computes the $\iota+1^{th}$ iterate ($\iota \geq 0$) according to:

$$x^{\iota+1} = x^\iota - (A^\iota)^{-1} \cdot s(x^\iota) \tag{1.3}$$

Actually the Jacobian is not inverted, but the vector z^ι is solved from the matrix equation:

$$A^\iota z^\iota = -s(x^\iota) \tag{1.4}$$

by the application of L\U-decomposition to A^ι. Then the Jacobian A^ι is decomposed into a lower triangular matrix L^ι and an upper triangular matrix U^ι such that $L^\iota U^\iota = A^\iota$ holds. Because L^ι is triangular, the intermediate vector y^ι can be solved easily from:

$$L^\iota y^\iota = -s(x^\iota)$$

by a process called "foresubstitution". Next z^ι is computed in essentially the same way from

$$U^l z^l = y^l$$

by a process called "backsubstitution". With the introduction of the residual vector r^l defined by:

$$r^l \triangleq -s(x^l) \tag{1.5}$$

one Newton iteration can be summarized as follows:

1) evaluate $A^l = A(x^l)$
2) compute L^l and U^l
3) solve y^l from $L^l y^l = r^l$
4) solve z^l from $U^l z^l = y^l$
5) compute x^{l+1} according to $x^{l+1} = x^l + z^l$
6) evaluate $r^{l+1} = -s(x^{l+1})$

The L\U-decomposition of a matrix A can be obtained by means of Gaussian elimination. Initially the coefficients of L and U are set equal to the corresponding coefficients of A:
$l_{ij} \leftarrow a_{ij}$ for $1 \le j \le i \le n$ and $u_{ij} \leftarrow a_{ij}$ for $1 \le i < j \le n$.
The subsequent Gaussian elimination is composed of n-1 steps, called "pivotsteps". The k^{th} pivotstep involves the operations:

$$u_{kj} \leftarrow u_{kj}/l_{kk} \qquad \text{for} \quad k < j \le n \tag{1.6}$$

$$u_{ij} \leftarrow u_{ij} - l_{ik}u_{kj} \qquad \text{for} \quad k < i < j \le n \tag{1.7}$$

$$l_{ij} \leftarrow l_{ij} - l_{ik}u_{kj} \qquad \text{for} \quad k < j \le i \le n \tag{1.8}$$

l_{kk} is called the k^{th} pivot. The execution of (1.7) or (1.8) for particular values of k, i and j will be called the "updating" of u_{ij} or l_{ij} respectively. The product $l_{ik}u_{kj}$ is called an "update" of u_{ij} or l_{ij}. Finally the diagonal coefficients of U, which are still zero, are set equal to one: $u_{ii} \leftarrow 1$ for $1 \le i \le n$.

A possibility to speed up the transient analysis exists in the deletion of operations in the computation of the L\U-decomposition if they have little influence on the result, i.e. the computed solution. For instance if the update $l_{ik}u_{kj}$ is small compared with l_{ij} or u_{ij} we may omit the updating of l_{ij} or u_{ij}. An important factor is the value of the pivot. Generally if $|l_{kk}|$ is large then u_{kj} becomes small and the update $l_{ik}u_{kj}$ can be expected to be small as well. The size of all updates in a pivotstep depends highly on the size of the pivot. If the absolute value of the pivot is large enough we can consider the skipping of the complete pivotstep in order to compute an approximate solution in a faster way. This approach will be called "pivotstep skipping".

Since DC analysis is essentially identical to the transient analysis for one time-point, much of the foregoing covers the DC analysis case as well.

1.2. Variability type

A very general way to describe an electrical circuit is the tableau approach [1.1]. The approach is based on "a tableau which includes all network information in a nonreduced form". The equations arising from the Kirchhoff current and voltage laws are directly put into the tableau. Further the tableau contains branch constitutive relations which may be nonlinear or time dependent. If the derivative operator d/dt is discretized a set of equations as in eq.(1.1) is obtained. The Jacobian of the set is said to be the "tableau matrix".

The authors of [1.1] distinguish several types of coefficients in the Jacobian, see table 1.1. For instance the coefficients in the Kirchhoff equations are of topological type. The branch constitutive relation of a linear resistor yields a c or p type coefficient, and a capacitor a t type coefficient. Nonlinear equations give x type coefficients. Equivalently types of operations and types of pivot-steps can be distinguished. The type of an operation is equal to the highest (in terms of the numbering) of the types of the operands. The type of a pivotstep is the highest of the types of operations it includes.

Hachtel et al. [1.1] exploit the types of operations to determine an optimal pivot order. The algorithm OPTORD which they propose,

TABLE 1.1.

type 1	coefficients which are +1	called topological type
type 2	coefficients which are -1	
type 3	coefficients which never change, called c type	
type 4	coefficients which change with design parameters: p type	
type 5	coefficients which change with time, called t type	
type 6	coefficients which change with the unknown, called x type	

tends to order pivots associated with low type of pivotsteps before pivots associated with high type of pivotsteps. During a transient analysis not all types of pivotsteps need be executed for each L\U-decomposition. Pivotsteps with a type not exceeding four must be executed at most once for the complete transient analysis. t type pivotsteps are repeated for each time-point and x type pivotsteps in each Newton iteration. OPTORD tries particularly to minimize the operations concerned with x and t type pivotsteps.

As pivotstep skipping aims at speeding up the transient analysis it is attractive to skip x and t type pivotsteps because these are executed very often. Hence we will consider the case that all type 1 to 4 pivotsteps are already executed and that pivotstep skipping is applied to a Jacobian inducing mere x and t type pivotsteps. Consequently pivotstep skipping has to take into account the variation of the matrix coefficients.

1.3. The Bordered Lower Triangular form

Jacobian matrices derived from circuit equations are mostly very sparse. For the tableau matrix Hachtel et al. [1.1] mention an average number of about three nonzero coefficients per row in general. The zero-nonzero structure of a matrix can have special forms. One of these is the Bordered Lower Triangular form (BLT form). Let a matrix A be partitioned as follows:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{1.9}$$

where A_{11} and A_{22} are square submatrices. If A_{11} is a lower triangular matrix with nonzero diagonal coefficients then A is said to have a BLT form. The border is constituted by A_{12} and A_{22}. The border width, i.e. the number of columns in A_{12} and A_{22}, is denoted by b while the dimension of the triangular matrix A_{11} is denoted by t. Generally a small border is attractive.

The L\U-decomposition of a BLT matrix has a conspicuous property. Most coefficients of the L and U factors are identical to the corresponding original matrix coefficients. Only the coefficients associated with the border may differ from the coefficients in A_{21} and A_{22}. The property appears from the equations (1.6) to (1.8). Originally the coefficient l_{ij} or u_{ij} is identical to a_{ij}, so initially u_{ij} is zero for all $j \leq t$. If u_{ij} is zero then it becomes nonzero only by a nonzero update. This requires a nonzero coefficient u_{kj} in the same column with $k < i$. Clearly all coefficients u_{1j} with $j \leq t$ remain zero and by induction it follows that all coefficients u_{ij} with $j \leq t$ remain zero. Consequently the execution of (1.8) for $j \leq t$ leaves l_{ij} unaffected, and l_{ij} stays equal to a_{ij}. Mere u_{ii} with $i \leq t$, which are set equal to one, are the trivial exceptions to the above rule. A consequence is that only the coefficients l_{ij} and u_{ij} with $t < j \leq n$ need be computed and stored. Moreover only in the border fill-in coefficients can arise.

Particular attention deserves the fact that the pivots $l_{ii} \equiv a_{ii}$ for $i \leq t$ are not subject to updating. During the computation of the L and U factor these pivots do not become smaller or even zero. The value of the pivot $l_{ii} \equiv a_{ii}$ in a Jacobian matrix is solely determined by the derivative $\frac{\partial s_i(x)}{\partial x_i}$. If a_{ii} is not constant then usually a range of values can be determined which a_{ii} can assume. If x is a vector of circuit variables then the values of the elements of x are finite and moreover for each element a range of values can be determined. So the value of x lies in some bounded domain and this may imply that $\frac{\partial s_i(x)}{\partial x_i}$ is bounded as well. A possible lower bound of $\left|\frac{\partial s_i(x)}{\partial x_i}\right|$ is of particular importance, for it is unattractive from the numerical point of view if $|a_{ii}|$ becomes very small.

In pivotstep skipping the values of the pivots determine whether the associated pivotsteps are executed or not. Because the pivots a_{ii} for $i \leq t$ are not updated, the pivotsteps to be skipped can be established prior to the actual numerical computation of the L and U factor.

The variables x_j with $t < j \leq n$, associated with the border, can be considered as control variables. If the values of these variables are known, the remaining variables, x_j with $1 \leq j \leq t$, can be computed by a relatively simple, foresubstitution-like process, using the original set of equations (1.1). Hence we may require that pivotstep skipping is applied such that the control variables are computed accurately enough. This can be achieved if no pivotsteps associated with pivots in the border, are skipped but only pivotsteps associated with pivots a_{ii} with $1 \leq i \leq t$, if a_{ii} is so large that the i^{th} pivotstep has only little influence on the rest of the L\U-decomposition.

1.4. Bipolar circuits

Willson [1.2] studies the nonlinear equations describing bipolar circuits. He uses the Ebers-Moll transistor model, see figure 1.1. Let v and i be the vector of t diode voltages or currents respectively. The vector f(v) represents the diode functions:
$i = f(v) = [f_1(v_1), f_2(v_2), \ldots, f_t(v_t)]^T$. The description of a bipolar circuit given in [1.2] uses two matrices, A_{21} and A_{22}, and a source vector c, all of dimension t:

$$\begin{aligned} &f(v) - i = 0 \\ &A_{21}v + A_{22}i = c \end{aligned} \tag{1.10}$$

Clearly the Jacobian A of the set of equations has a BLT form with border width t. For A can be partitioned as indicated in eq.(1.9) such that all submatrices have dimension t. Thus A_{11}, the Jacobian of f(v), is a diagonal matrix: $a_{jj} = \frac{df(v_j)}{dv_j}$, $a_{jk} = 0$ for $k \neq j$, $j \leq t$, $k \leq t$, and surely the requirement that A_{11} is lower triangular is fulfilled. By the way A_{12} is the negative of the identity matrix. A feature of this Jacobian is that all coefficients are constants except the diagonal coefficients (pivots!) of A_{11}. The latter coefficients are the derivatives of the nonlinear functions and depend on the values of circuit variables. With respect to pivotstep skipping this Jacobian is attractive as the pivots in the lower triangular submatrix are variable coefficients. So the values of the pivots are appropriate to control the execution of pivotsteps.

The range of the pivot values can be determined with the diode function. If we use the relation $i_k = f(v_k) = I_S(\exp(\frac{v_k}{V_T}) - 1)$, where I_S and V_T are parameters, then the value of the pivot is:

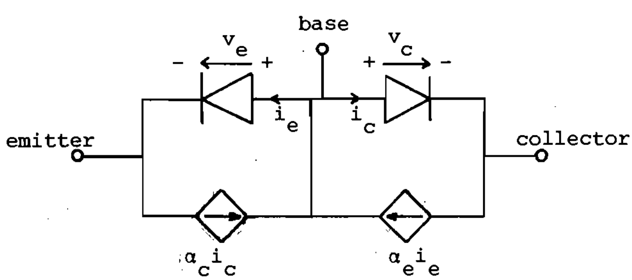

Fig.1.1. The Ebers-Moll model for a NPN-transistor.

$$a_{kk} = \frac{df(v_k)}{dv_k} = \frac{I_S}{V_T}\exp\left(\frac{v_k}{V_T}\right) = \frac{i_k+I_S}{V_T}$$

Although the derivative cannot become exactly zero for finite values of the voltage, it is very small for negative values of the voltage. With $I_S = 10^{-14}$A and $V_T = 25$mV it appears that $|a_{kk}|$ is less than 10^{-12}mho for $v_k \leq 0$V. On the other hand the maximum value of $|a_{kk}|$ is relatively small: if i_k does not exceed the value of 2.5mA then $|a_{kk}|$ satisfies $|a_{kk}| \leq 0.1$mho.

It is more attractive to use the inverse relation $v_k = V_T \ln \frac{i_k+I_S}{I_S}$. If the vectors v and i are interchanged in (1.10) and the inverse relations are applied, a set of equations similar to (1.10) arises. However the new variable pivots are the inverses of the original ones. Hence the domains of the new pivots are given by:

$$|a_{kk}| = \left|\frac{V_T}{i_k+I_S}\right| \geq 10\Omega\ , \quad \text{for } i_k \leq 2.5\text{mA},\quad 1 \leq k \leq t \qquad (1.11)$$

In this way a reasonable lower bound to the absolute value of the pivot is achieved.

The border width of the Jacobian of (1.10) is large: the dimensions of A_{12} and A_{22} are equal to t. Whereas usually a border width which is an order smaller than the dimension of the matrix, can be achieved for circuit equations. However the border width of the Jacobian of (1.10) cannot be expected to be minimal. For instance by reordering of the equations and the variables a set of equations may be obtained such that the Jacobian has a BLT form with a border smaller than t. The reordering may be such that the feature that all coefficients supplied by f(v) are on the diagonal of the triangular matrix, is retained (the property that A_{11} is diagonal need not be retained). Hence one may try to put the equations in a form such that the Jacobian has a BLT form with a minimum border but with the restriction that all variable coefficients are on the diagonal of the triangular submatrix.

1.5. The computation of the residual

Consider a set of nonlinear equations s(x) described by:

$$s(x) = f(x) + \bar{A}x = 0 \qquad (1.12)$$

where $\bar{A}$ is a matrix with constant coefficients and with $\bar{a}_{ii} \equiv 0$ for

$1 \le i \le t$. $f(x)$ is the vector $[f_1(x_1), f_2(x_2),\ldots,f_n(x_n)]^T$ with $f_i(x_i)$ a real function of x_i for $1 \le i \le t$ and $f_i(x_i) \equiv 0$ for $t < i \le n$. Such a set of equations is compatible with a bipolar circuit. Let the matrix D be the Jacobian of $f(x)$: $d_{ii} \doteq \frac{df_i(x_i)}{dx_i}$ for $1 \le i \le t$, d_{ij} is zero otherwise. Then the Jacobian A of $s(x)$ is $A = D + \bar{A}$.

The residual for a set of equations of the form (1.12) can be computed relatively fast during Newton iteration. The definition of the residual $r^l \triangleq -s(x^l)$, suggests that $O(n^2)$ mathematical operations are required to evaluate r^l. For sets of equations of the form (1.12) the residual can be evaluated in $O(n)$ operations. Suppose that λ is some real number and that x^{l+1} is computed according to:

$$x^{l+1} = x^l + \lambda z^l$$

where z^l is the solution of eq.(1.4). λ may represent a damping factor. Then the residual r^{l+1} is:

$$r^{l+1} = -s(x^{l+1}) = -f(x^{l+1}) - \bar{A}x^{l+1} = -f(x^{l+1}) - \bar{A}x^l - \lambda\bar{A}z^l$$

$$= r^l + f(x^l) - f(x^{l+1}) - \lambda\bar{A}z^l$$

Using $\bar{A} = A^l - D^l$ and eq.(1.4) we derive:

$$r^{l+1} = r^l - \lambda r^l + \lambda D^l z^l + f(x^l) - f(x^{l+1}) \qquad (1.13)$$

If we choose $\lambda = 1$ the equation simplifies to:

$$r^{l+1} = D^l z^l + f(x^l) - f(x^{l+1})$$

Alternatively the value of λ can be determined such that r^{l+1} is "minimized" in some sense [1.3]. For instance the norm of r^{l+1} can be minimized. The computation of r^{l+1} according to (1.13) for a given value of λ requires t function evaluations ($f(x^l)$ is already computed during the preceding iteration), t+n multiplications and n+3t additions/subtractions.

If pivotstep skipping is applied then instead of $\hat{z}^l$, the exact solution of eq.(1.4), an approximation z^l is computed. z^l can be considered as the solution of a perturbed set of equations, i.e. perturbations δA^l and δr^l exist such that z^l satisfies:

$$(A^l + \delta A^l)z^l = (r^l + \delta r^l)$$

With the introduction of $\bar{r}^l$, defined by:

$$\bar{r}^l = \delta A^l z^l - \delta r^l \tag{1.14}$$

we write

$$A^l z^l = r^l - \bar{r}^l \tag{1.15}$$

The consequence is that equation (1.13) needs a modification. If in the derivation of an equation for the residual eq.(1.15) is used instead of (1.4) we obtain:

$$r^{l+1} = r^l - \lambda(r^l - \bar{r}^l) + \lambda D^l z^l + f(x^l) - f(x^{l+1})$$

and for $\lambda = 1$:

$$r^{l+1} = \bar{r}^l + D^l z^l + f(x^l) - f(x^{l+1}) \tag{1.16}$$

In addition to the common evaluation of the residual the vector $\bar{r}^l$ has to be determined. In chapter 6 we will show that $\bar{r}^l$ can be determined in O(n) operations for each skipped pivotstep.

[1.1] G.D. Hachtel, R.K. Brayton, F.G. Gustavson, "The sparse tableau approach to network analysis and design", IEEE Trans. Circuit Theory, Vol. CT-18, pp. 101-113, (1971).

[1.2] A.N. Willson, "New theorems on the equations of nonlinear DC transistor networks", Bell Syst. Techn. J., Vol. 49, pp. 1713-1738, (1970).

[1.3] E. Isaacson, H.B. Keller, "Analysis of numerical methods", Wiley & Sons, New York, 1966.

2. THE DETERMINATION OF A BLT FORM

In this chapter we give the theoretical basis for a method to obtain a BLT form. The main objective of such a method is to supply a border which is as small as possible, i.e. it contains a minimum number of columns.

We will refer to graph representations of the zero-nonzero structure of a matrix A. The (undirected) bipartite graph $B(A) = (S, T, E)$ associated with A [2.1] consists of two sets of n vertices, S and T, and a set of edges E defined by $E = \{(s_i,t_j) \mid a_{ij} \neq 0\}$. A set of edges is a "matching" if each vertex in S and T is incident to at most one edge of the matching. A matching is "complete" if it contains n edges. Without ambiguity the associated set of coefficients in A is called a matching too. With appropriate row and column permutations applied to A the coefficients in a matching appear on the main diagonal. Any non-singular matrix has a complete matching. If $B(A)$ has a complete matching then also a directed graph (digraph) $G(A)$ can be associated with A. The digraph $G(A)$ can be obtained from $B(A)$ by

1) directing all edges (s_i,t_j) from s_i to t_j;
2) coalescing any vertices s_i and t_j if (s_i,t_j) is in the matching, while (s_i,t_j) is deleted.

After appropriate row and column permutations applied to A, the diagonal coefficients of A correspond to vertices in $G(A)$ while the edges in $G(A)$ are associated with off-diagonal coefficients.

2.1. Background

The origines of the problem of how to obtain a minimum border lie in the field of graph theory. In 1961 Seshu and Reed [2.2] formulated a problem suggested by Runyan: how to determine in an arbitrary digraph a minimum set of edges which, if removed, leave the resultant graph without any directed cycles. The problem is usually referred to as the "minimum feedback arc set" problem [2.3]. The problem arose from the analysis of systems with feedbacks, for instance switching circuits.

A related problem is the determination of "minimum feedback vertex set" (which, if removed together with their incident edges, leave the resultant graph without any directed cycles). Nathan exploits a feedback vertex set, called by him "principal set of nodes", for the reduction of signal flow graphs [2.4, 2.5]. Recently however it is the notion "essential set" [2.6] that has become very popular. Both problems, the edge and the vertex version, are nonpolynomial complete (NP-complete) [2.7].

The relation between an essential set and a BLT form is indicated by Kevorkian [2.8] and Cheung and Kuh [2.9]. If a set of vertices V in $G(A)$ is an essential set then A can be given a BLT form with border width $|V|$ by appropriate symmetric row and column permutations such that the i^{th} column is in the border if and only if the vertex associated with a_{ii} is in V. Henceforth a set of columns constituting the border of some BLT form of A is called an essential set as well.

The cardinality of a minimum essential set of a graph G is called the index of G. In [2.6, 2.8, 2.9] rules are given for simplifying a graph without changing its index and for the detection of vertices contained in some minimum essential set. Generally these "index preserving" rules do not reduce the graph completely, but a graph may remain to which none of the rules can be applied. The rules can be implemented using $O(n^2)$ operations [2.10]. Hence the application of these rules prior to the use of a non-polynomial algorithm to determine a minimum essential set is advantageous.

An important algorithm is proposed by Smith and Walford [2.11]. They attempt to split some subgraphs from the digraph G such that a minimum essential set for any subgraph is a subset of a minimum essential set for the digraph G. According to the nature of the problem this algorithm and the other algorithms which supply a minimum essential set [2.12, 2.6, 2.8, 2.9] require a number of operations which is not bounded by a polynomial in n.

As the problem is NP-complete heuristic algorithms are proposed which supply a small essential set, hopefully with a cardinality close to minimum, e.g. in [2.13]. The application of breadth-first search to determine an essential set [2.14] is interesting, but much has to be investigated yet. The algorithm proposed in [2.15] works on the bipartite graph $B(A)$ rather than on the digraph $G(A)$. It assumes that $B(A)$ contains a complete matching. However during execution of the algorithm the matching is changed if necessary in order to obtain a minimal essential set. (Whereas an essential set V is called minimum if no smaller essential set exists, V is called minimal if no proper subset of V is essential.) Also the algorithm P^4 [2.16, 2.17] constructs a matching while determining an essential set. In these papers an essential set of matrix columns is called a set of "spikes".However the matching supplied by P^4 may not be complete,even if a complete one exists.This is not required either, as a matching sufficies such that the diagonal of A_{11} in eq.(1.9) is nonzero. Apart from the algorithm in [2.15] the heuristic algorithms supply an essential set which may be not minimal. Then a minimal essential set can be obtained by a method like the one suggested by corollary 2.7 (see section 2.3) and implemented in line 15 of the procedure MES (see section 3.3.1).

The algorithm to be proposed in chapter 3 has in common with the above ones that a matching, not necessarily complete, is constructed during the identification of an essential set. Although the heuristic rule to select essential variables ("take the variable implying the largest train", see section 3.1) differs from previous ones, the main difference with other algorithms is that we exploit the possibility to transform the linear equations in order to obtain a smaller essential set. Consequently we are independent of the way the linear equations are formulated. The equations describing an electrical circuit, particularly the Kirchhoff current and voltage laws, can be written in several equivalent manners. Examples are the tableau approach [2.18] and the modified nodal approach [2.19]. Also the Krichhoff laws can be formulated with respect to some tree. The way of formulating the equations has an impact on the zero-nonzero structure of the coefficient matrix of the circuit and consequently on the size of the minimum essential set for that matrix.

In view of the time complexity of the fore-mentioned algorithms, the exploitation of the "lower block triangular" form (LBT form) of a matrix can be recommended. Let a matrix A consist of submatrices A_{ij}, $1 \leq i \leq m$, $1 \leq j \leq m$. Let A_{ii} be square for $1 \leq i \leq m$ and let A_{ij} be a zero matrix for $j > i$. Then A is said to have a lower block triangular form. The submatrices A_{ii}, $i = 1,\ldots,m$, are called "blocks". The form may be obtained by row and column permutations. If for any permutation matrices P and Q the matrix PAQ has no LBT form, i.e. PAQ consists of exactly one block, then A is called "irreducible". Dulmage and Mendelsohn [2.20] discuss the "canonical decomposition" of a bipartite graph $B(A)$. This decomposition induces a LBT form of A such that the blocks are irreducible.

If a minimum essential set is determined with the restriction that the matching is kept fixed then each minimum essential set is a union of subsets, each being a minimum essential set for an (irreducible) block in the LBT form of the matrix.

To establish an LBT form a complete matching is used [2.21]. The algorithm in [2.22] for constructing a complete matching requires $O(n^{5/2})$ operations. Depth-first search [2.23] can be applied to identify in the digraph associated with the matrix the strongly connected components corresponding to the irreducible blocks. Depth-first search requires $O(n^2)$ operations. Since the algorithms to identify an essential set usually require asymptotically more operations than the algorithms to establish an LBT form, the exploitation of the latter may yield less operations in all (according to the "divide and conquer" principle [2.7], provided smaller blocks are actually found). More for curiosity, however, we remark that the canonical decomposition prior to the identification of an essential set may have an impact on the size of the border. The straightforward application of the algorithm MES, given in chapter 3, may yield a border which is smaller than the union of the minimum borders of the irreducible blocks (e.g. the example in figure 7 in [2.17] has three blocks with four border columns in all, while without decomposition a border of three columns, the columns "2", "1" and "3", can be found).

It should be noted that sticking to a fixed matching means a substantial restriction if we are really looking for the smallest border. Experiments on electrical circuits indicate that a reduction of the border by about 50% can be obtained if we are satisfied with any matching such that the diagonal of A_{11} in eq.(1.9) is nonzero and apply transformations to the linear equations, instead of exploiting the digraph representation of the matrix (see table 3.1).

2.2. Definitions

We consider the set of equations described in eq.(1.1). Let N denote the set of integers $\{1,2,\ldots,n\}$, let S denote the set of functions $S = \{s_i(x) \mid i \in N\}$, and let X be the set of variables $X = \{x_i \mid i \in N\}$. The structure matrix S of s(x) consists of the coefficients s_{ij}, $i \in N$, $j \in N$, defined by:

$s_{ij} = 1$ if $\dfrac{\partial s_i(x)}{\partial x_j} \not\equiv 0$, otherwise $s_{ij} = 0$.
The set of variables on which $s_i(x)$ depends is denoted by $X_i \triangleq X(s_i(x)) \triangleq \{x_j \mid s_{ij} = 1\}$. Equivalently the set of functions dependent on x_j is denoted by $S_j \triangleq S(x_j) \triangleq \{s_i(x) \mid s_{ij} = 1\}$.

A function $s_i(x)$ is called "linear" if the coefficients in the i^{th} row of the Jacobian, $a_{ij} = \dfrac{\partial s_i(x)}{\partial x_j}$ for all j with $x_j \in X_i$, have a typenumber not exceeding 4. Such a function can be noted by

$$s_i(x) = \sum_{x_j \in X_i} a_{ij} x_i - r_i$$

Let L be a subset of N consisting of all indices i such that $s_i(x)$ is linear. The vector function of dimension $|L|$ associated with the linear functions is denoted by $s_L(x)$. The remaining functions are associated with the vector function $s_{\bar{L}}(x)$, where $\bar{L}$ is the complement of L in N. Two vector functions $s(x)$ and $\hat{s}(x)$ are called "equivalent" if 1) $s_{\bar{L}}(x) \equiv \hat{s}_{\bar{L}}(x)$

2) $s_L(x) \equiv \Phi \hat{s}_L(x)$

are satisfied, where Φ is a $|L|$ by $|L|$ nonsingular matrix, called a "transformation" matrix.

If V is a set of variables and for some vector function $\tilde{s}(x)$, equivalent to $s(x)$, we have for some i $X(\tilde{s}_i(x))\backslash V = x_j$ then x_j is called a "novice" of V induced by $s(x)$. $\tilde{s}_i(x)$ is called a "novice function" of x_j. We define the notion "partial train" recursively: 1) any set $V \subset X$ is a partial train of itself, 2) if R_V is a partial train of V and x_j is a novice of R_V then $R_V \cup \{x_j\}$ is a partial train of V. The set V is called the "kernel" of the partial train R_V. If a partial train T_V is maximal, i.e. T_V is no proper subset of some other partial train R_V of V, then T_V is called a "train" of V. If T_V is identical to X we say that the kernel V is an "essential set".

Given some partial train R_V a variable $x_j \in R_V$ is called a "follower", while a variable $x_j \in \bar{R}_V$ is called a "nonfollower". A function dependent on followers only is called a "follower function", the remaining functions are called "nonfollower functions". A partial train R_V induces a partitioning of x and $s_L(x)$. $s_L(x)$ can be partitioned into $s_{LF}(x)$, the follower linear functions, and $s_{LN}(x)$, the nonfollower linear functions. x consists of x_F, the followers, and x_N, the nonfollowers. According to this partitioning we write:

$$\begin{bmatrix} s_{LF}(x) \\ s_{LN}(x) \end{bmatrix} \equiv \begin{bmatrix} A_{FF} & 0 \\ A_{NF} & A_{NN} \end{bmatrix} \begin{bmatrix} x_F \\ x_N \end{bmatrix} \tag{2.1}$$

The zero submatrix is implied by the definition of the follower functions.

2.3. Theory

The statements in this section are the basis of the algorithms to be developed. Most of the statements are not fundamentally different from those that could be obtained from graphs. However since we deal with transformations of the linear functions it is no longer adequate to exploit a graph representation of the functions. First we establish the uniqueness of the train of a given kernel and the relation between the train and partial trains of the same kernel.

Lemma 2.1: Let R_V and R_W be partial trains of V and W respectively. $V \subset R_W$ and $R_V \not\subset R_W$ imply that R_V contains a novice of R_W.

Proof: Consider the series of sets $V = R^0, R^1, R^2, \ldots, R^m = R_V$ such that $R^k = R^{k-1} \cup \{x_{j_k}\}$ and x_{j_k} is a novice of R^{k-1} for $k = 1,2,\ldots,m$. By the definition of a partial train such a series exists. $R^0 \subset R_W$ and $R^m \not\subset R_W$ imply that $R^{k-1} \subset R_W$ and $R^k \not\subset R_W$ is satisfied for some k. Then for some i and j we have $R^k \backslash R^{k-1} = \{x_j\} = X(s_i(x)) \backslash R^{k-1} = X(s_i(x)) \backslash R_W$, showing that x_j is a novice of R_W.

Lemma 2.2: If R_V and Q_V are partial trains of V then $R_V \cup Q_V$ is a partial train of V.

Proof: If R_V is a subset of Q_V then the lemma is correct. In the case $R_V \not\subset Q_V$ the relation $V \subset Q_V$ implies by lemma 2.1 that R_V contains a novice x_j of Q_V. By definition $Q_V^1 = Q_V \cup \{x_j\}$ is a partial train of V. Using the same arguments for R_V and Q_V^1 it follows that $R_V \subset Q_V^1$ or R_V contains a novice of Q_V^1. Thus partial trains $Q_V^1, Q_V^2, \ldots,$ can be constructed until for some m we obtain $R_V \subset Q_V^m$. Then $Q_V^m = R_V \cup Q_V$ is a partial train of V.

Theorem 2.3: Each partial train of V is a subset of the unique train of V.

Proof: The uniqueness of the train follows from the application of lemma 2.2 to two trains T_V^1 and T_V^2. Now the theorem follows if lemma 2.2 is applied to the train and a partial train.

Two properties of trains are formulated in the following corollaries.

Corollary 2.4: T_V is a train of V if and only if T_V has no novice.

Proof: Lemma 2.1 and theorem 2.3 prove the "if" part and the "only if" follows by definition.

Corollary 2.5: If T_V and T_W are trains of V and W respectively then $W \subset T_V$ implies $T_W \subset T_V$.

Proof: By contradiction: $T_W \not\subset T_V$ implies by lemma 2.1 that T_W contains a novice of T_V, contradicting that T_V is a train.

The following corollaries give some properties of an essential set.

Corollary 2.6: V is an essential set if it contains an essential set as a subset.

Proof: Let $W \subset V$ be an essential set. By corollary 2.5 the train T_W of W is a subset of the train T_V of V. Since W is essential we have $X = T_W = T_V$ proving that V is essential.

Corollary 2.7: If V is an essential set and $V \backslash \{x_j\}$ is not essential for any $x_j \in V$ then V is a minimal essential set.

Proof: If V is not minimal it contains a proper subset W being essential. Then V contains at least one variable x_j such that $W \subset V \backslash \{x_j\}$. By corollary 2.6 $V \backslash \{x_j\}$ is essential contrary to the supposition.

Next we show how two trains can be joined to one new train.

Theorem 2.8: Let T_V be the train of V and let W be a subset of $X \backslash T_V$. Then the train T_Y of $Y = W \cup T_V$ is identical to $T_{V \cup W}$, the train of $V \cup W$.

Proof: We exploit corollary 2.5. $V \cup W \subset Y$ implies $T_{V \cup W} \subset T_Y$. On the other hand $Y = W \cup T_V \subset T_{V \cup W}$ implies $T_Y \subset T_{V \cup W}$. Hence we have $T_{V \cup W} = T_Y$.

Theorem 2.9: If V is an essential set then a structure matrix $\hat{S}$, associated with a vector function $\hat{s}(x)$ equivalent to

$s(x)$, exists and permutation matrices P and Q exist, such that $P\hat{S}Q$ has a BLT form with border width $|V|$ and the variables in V are associated with the border columns.

Proof: The train of V is identical to X as V is an essential set. So we have the series of sets $V = R^0, R^1, \ldots, R^m = T_V = X$ such that $R^k = R^{k-1} \cup \{x_{j_k}\}$ and x_{j_k} is a novice of R^{k-1} for $k = 1,2,\ldots,m$. If the novice functions $s_{i_k}(x)$ of x_{j_k} for $k = 1,2,\ldots,m$ all are in $s(x)$ then we see immediately that permutation matrices exist such that PSQ, with S the structure matrix of $s(x)$, has the required BLT form. (Q follows from the permutation $j_1, j_2, \ldots, j_m$ completed with an arbitrary ordering of the remaining indices (of the variables in V), P follows from the permutation $i_1, i_2, \ldots, i_m$ completed with an arbitrary ordering of the remaining indices).

If not all novice functions are in $s(x)$ we will show that some equivalent vector function includes all novice functions. We will proof that each partial train R^k, $k = 1,2,\ldots,m$, is associated with some $s^k(x)$, equivalent to $s(x)$, such that all novice equations $s_{i_1}(x), s_{i_2}(x), \ldots, s_{i_k}(x)$ are in $s^k(x)$.

By the definition of a novice R^1 is associated with some $s^1(x)$ containing $s_{i_1}(x)$. For the induction step we assume that R^k is associated with $s^k(x)$ containing $s_{i_1}(x), \ldots, s_{i_k}(x)$. If $s^k(x)$ contains $s_{i_{k+1}}(x)$, the induction step is trivial. Otherwise, as $x_{j_{k+1}}$ is a novice of R^k, some $\tilde{s}(x)$, equivalent to $s^k(x)$, contains a novice function $\tilde{s}_i(x)$ of $x_{j_{k+1}}$. Consider the partitioning of $s^k_L(x)$ and x induced by the partial train R^k of V. Then with eq.(2.1) we have:

$$\tilde{s}_L(x) = \Phi s^k_L(x) = [\Phi_F, \Phi_N] \begin{bmatrix} A_{FF} & 0 \\ A_{NF} & A_{NN} \end{bmatrix} \begin{bmatrix} x_F \\ x_N \end{bmatrix}$$

where the partitioning of Φ is induced by that of $s^k_L(x)$. Hence for the novice function $\tilde{s}_i(x)$ we write:

$$\tilde{s}_i(x) = (\phi_{iF}A_{FF} + \phi_{iN}A_{NF})x_F + \phi_{iN}A_{NN}x_N$$

where $[\phi_{iF}, \phi_{iN}]$ is the i^{th} row of Φ. As $\tilde{s}_i(x)$ is a novice function the vector $\phi_{iN}A_{NN}$ contains only one nonzero coefficient. Apparently the function

$$\hat{s}_i(x) = \phi_{iN}A_{NF}x_F + \phi_{iN}A_{NN}x_N$$

is a novice function too. $\hat{s}_i(x) \not\equiv 0$ implies that ϕ_{iN} contains some nonzero coefficient. Arbitrarily let the ordering of the functions be such that this is the coefficient ϕ_{ii}. Then the transformation matrix $\hat{\Phi}$ being equal to the unit matrix except for the i^{th} row which consists of $[0, \phi_{iN}]$, is nonsingular. The transformation $\hat{\Phi}$ applied to $s_L^k(x)$ leaves all follower functions unaffected while only one nonfollower linear function is replaced by $\hat{s}_i(x)$. In this way $s^{k+1}(x)$ is constructed containing $s_{i_{k+1}}(x) \equiv \hat{s}_i(x)$. By induction $\hat{s}(x) = s^m(x)$ associated with $R^m = T_V$ contains all novice functions.

The proof is constructive in that it gives a detailed description of how to obtain an appropriate equivalent vector function and the appropriate permutation matrices. Another important result is that a novice function, provided it exists, always can be found as a linear combination of the nonfollower (linear) functions. How a novice function can be constructed is suggested by the following theorem.

Theorem 2.10: Let R_V be a partial train of V with respect to $s(x)$ and let the partitioning of $s_L(x)$ induced by R_V be as given in eq.(2.1). If A_{NN} has rank m and contains a m by m unit matrix then R_V has a novice if and only if $s(x)$ includes a novice function.

Proof: The "if" part is trivial. To prove the "only if" part, assume $s(x)$ includes no novice function while some equivalent vector function $\hat{s}(x)$ includes the novice function $\hat{s}_i(x)$. Apparently $\hat{s}_i(x)$ is a linear combination of at least two linear functions of $s(x)$, both depending on some nonfollowers. The unit matrix assures that $\hat{s}_i(x)$ depends on at least two nonfollowers contrary to the supposition.

If $s(x)$ does not satisfy the conditions of the above theorem then we may apply an appropriate transformation to establish an unit matrix in A_{NN}. If a novice equation exists then it will be obtained by this transformation. The transformation matrix can be determined easily if Gauss-Jordan elimination is applied.

[2. 1] A.L. Dulmage, N.S. Mendelsoh, "Graphs and matrices", in Graph Theory and Theoretical Physics, (F. Harary, Ed.), Academic, New York, 1967, pp. 167-227.

[2. 2] S. Seshu, M.B. Reed, "Linear graphs and electrical networks", Addison-Wesley, Reading, Mass., 1961.

[2. 3] D.H. Younger, "Minimum feedback arc sets for a directed graph", IEEE Trans. Circuit Theory, Vol. CT-10, pp. 238-245 (1963).

[2. 4] A. Nathan, "A two-step algorithm for the reduction of signal flow graphs", Proc IRE (Corresp.), Vol. 49, p. 1431 (1961).

[2. 5] A. Nathan, R.K. Even, "The inversion of sparse matrices by a strategy derived from their graphs", Comput. J., Vol. 10, pp. 190-194 (1967).

[2. 6] G. Guardabassi, "A note on minimal essential sets", IEEE Trans. Circuit Theory (Corresp.), Vol. CT-18, pp. 557-560 (1971).

[2. 7] A.V. Aho, J.E. Hopcroft, J.D. Ullman, "Analysis and design of computer algorithms", Addison-Wesley, London, 1974.

[2. 8] A.K. Kevorkian, J. Snoek, "Decomposition in large scale systems: theory and applications in solving large sets of nonlinear simultaneous equations", in Decomposition of large scale problems, (D.M. Himmelblau, Ed.), North Holland, Amsterdam, 1973, pp. 491-515.

[2. 9] L.K. Cheung, E.S. Kuh, "The bordered triangular matrix and minimum essential sets of a digraph", IEEE Trans. Circuits Syst., Vol. CAS-21, pp. 633-639 (1974).

[2.10] A.K. Kevorkian, "General topological results on the construction of a minimum essential set of a directed graph", IEEE Trans. Circuits Syst., Vol. CAS-27, pp. 293-304 (1980).

[2.11] G.W. Smith Jr., R.B. Walford, "The identification of a minimal feedback vertex set of a directed graph", IEEE Trans. Circuits Syst., Vol. CAS-22, pp. 9-15 (1975).

[2.12] A. Lempel, I. Cederbaum, "Minimum feedback arc and vertex sets of a directed graph", IEEE Trans. Circuit Theory, Vol. CT-13, pp. 399-403 (1966).

[2.13] M. Diaz, J.P. Richard, M. Courvoisier, "A note on minimal and quasi-minimal essential sets in complex directed graphs", IEEE Trans. Circuit Theory (Corresp.), Vol. CT-19, pp. 512-513 (1972).

[2.14] N.M. Znotinas, P.H. Roe, "A breadth-first search method for finding minimum essential sets in digraphs", 22nd Midwest Symp. Circuits Syst., 1979, pp. 26-31.

[2.15] A. Sangiovanni-Vincentelli, T.A. Bickart, "Bipartite graphs and an optimal bordered triangular form of a matrix", IEEE Trans. Circuits Syst., Vol. CAS-26, pp. 880-889 (1979).

[2.16] E. Hellerman, D. Rarick, "Reinversion with the preassigned pivot procedure", Math. Programming, Vol. 1, pp. 195-216 (1971).

[2.17] E.Hellerman, D. Rarick, "The partitioned preassigned pivot procedure (P^4)", in Sparse matrices and their applications, (D.J. Rose, R.A. Willoughby, Eds.), Plenum, New York, 1972, pp. 67-76.

[2.18] G.D. Hachtel, R.K. Brayton, F.G. Gustavson, "The sparse tableau approach to network analysis and design", IEEE Trans. Circuit Theory, Vol. CT-18, pp. 101-113 (1971).

[2.19] C.W. Ho, A.E. Ruehli, P.A. Brennan, "The modified nodal approach to network analysis", IEEE Trans. Circuits Syst., Vol. CAS-22, pp. 504-509 (1975).

[2.20] A.L. Dulmage, N.S. Mendelsohn, "Covering of bipartite graphs", Can. J. Math., Vol. 10, pp. 517-534 (1958).

[2.21] A.L. Dulmage, N.S. Mendelsohn, "Two algorithms for bipartite graphs", J. Soc. Indust. Appl. Math., Vol. 11, pp. 183-194 (1963).

[2.22] J.E. Hopcroft, R.M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs", SIAM J. Comput.,Vol. 2, pp. 225-231 (1973).

[2.23] R.E. Tarjan, "Depth-first search and linar graph algorithms", SIAM J. Comput., Vol. 1, pp. 146-160 (1972).

3. THE MINIMAL ESSENTIAL SET ALGORITHM

3.1. Outline of the algorithm

The algorithm MES establishes an essential set in steps. In each step one variable is added to the kernel V. In the k^{th} step the kernel contains k-1 variables and the k^{th} variable is selected from the variables in $X\backslash T_V$. For each variable $x_j \in X\backslash T_V$ the train T_Y of $Y = T_V \cup \{x_j\}$ is determined. Then the variable x_j associated with the train T_Y having the largest cardinality, is selected and added to the kernel. Here we exploit theorem 2.8 which assures that T_Y is identical to $T_{V\cup\{x_j\}}$. By the way it is very easy to apply a more sophisticated criterion to select variables. In one or another way we may compute some figure of merit of T_Y and select the variable x_j on the basis of this figure of merit. For instance the "quality factor" of so called "elementary blocks" as proposed in [3.1] may be used. (An elementary block is a train having a special form.)

The above procedure terminates if $X\backslash T_V$ is the empty set, showing that the kernel V is an essential set. Next corollary 2.7 is exploited to obtain a minimal essential set. For each variable $x_j \in V$ we determine the train of $V\backslash\{x_j\}$ in order to test whether V remains essential if x_j is removed. If so then a smaller essential set is obtained, else x_j has apparently to be retained in the set. Thus MES ends up with a set V satisfying the condition of corollary 2.7 so that V is minimal.

We introduce the procedure TRAIN to determine the train of a given kernel. It starts with the partial train R_V consisting merely of the kernel V. Any novices are detected and added to R_V. If no novice exists then by corollary 2.4 R_V is the train of V. It is easy to determine whether a novice exists as the additional procedure TRANSFORM yields a vector function $\hat{s}(x)$, equivalent to $s(x)$, including a novice function or satisfying the condition of theorem 2.10. TRANSFORM applies Gauss-Jordan elimination in order to establish a unit matrix with a rank equal to that of A_{NN} (see eq.2.1). TRANSFORM is called only if the current vector function includes no novice function.

3.2. The datastructure

In the description of the algorithm we assume that a linked list datastructure represents s(x). The linked list structure contains the nonzero coefficients of the structure matrix S. The coefficients in some column j of S are connected by pointers, forming a single linked list, such that the set $S(x_j)$ is easily accessible. The coefficients in some row i are connected by pointers, establishing a double linked list, in view of having an easy access to $X(s_i(x))$. The latter linked list is doubled because often subsets of $X(s_i(x))$ have to be entered. If Y is a subset of X then $Y(s_i(x)) \triangleq Y \cap X(s_i(x))$ is indicated by omitting, for all j with $x_j \in \bar{Y}$, the coefficients s_{ij} from the list associated with the i^{th} row of S. Further we assume a pointer from each coefficient s_{ij} to the function s_i and one to the variable x_j. For the linear functions the nonzero coefficients of the Jacobian are included in the datastructure such that a_{ij} is stored next to s_{ij}. Pointers from each s_i and x_j to the first coefficient in the list associated with the i^{th} row and the j^{th} column respectively complete the datastructure.

3.3. The description of the algorithm

3.3.1. The procedure MES

The procedure MES is given in Pidgin Algol on p. 26 . It determines a minimal essential set (MES) of variables V for a set of functions S depending on the variables in the set X.

In lines 2 and 3 the train of the empty kernel is determined. The novices of the empty kernel, detected by $|X(s_i)| = 1$ for the novice functions concerned, are gathered in the set W. Next TRAIN (X,W) determines the train T_W of $\bar{X} \cup W = W$. Because the variables in W are novices of ϕ we have $W = R_\phi \subset T_\phi$ and consequently $T_W \subset T_\phi$ by corollary 2.5. The identity $T_W = T_\phi$ follows from corollary 2.4 as $T_W = \tilde{R}_\phi$ has no novice. The detection of novices described above is a feature in many algorithms, e.g. P^4 [3.2] and EL [3.3]. Also a novice can be compared with a so called "compact vertex" [3.4]. The train of the empty kernel is determined for the sake of completeness. In the case that all matrices S, associated with any equivalent

```
  procedure MES(X,S,V)
     begin
1    W ← ϕ; V ← ϕ;
2    for each s ∈ S do if |X(s)| = 1 then W ← W ∪ X(s);
3    T̂ ← TRAIN(X,W); X̂ ← X\T̂; Y ← X̂;
4    for each s ∈ S do Y(s) ← X(s) ∩ Y;
5    while Y ≠ ϕ do
        begin
6       Z ← Y; max ← 0; T̂ ← ϕ;
7       while Z ≠ ϕ do
           begin
8          choose x ∈ Z; T ← TRAIN(Y,x);
9          if |T| > max then
              begin
10            max ← |T|; T̂ ← T; x̂ ← x
              end
11         Z ← Z\T
           end
12      V ← V ∪ {x̂}; Y ← Y\T̂;
13      for each x ∈ T̂ do for each s ∈ S(x) do Y(s) ← Y(s)\{x}
        end
14   for each s ∈ S do X̂(s) ← X(s) ∩ X̂;
15   for each x ∈ V do if TRAIN(X̂,V\{x}) = X̂ then V ← V\{x}
     end MES
```

vector function of s(x), are irreducible, we have $T_{\phi} = \phi$ and may delete lines 2 and 3. Then line 4 can be deleted as well. In the latter linė the datastructure is updated such that the sets of nonfollowers $Y(s_i(x))$ associated with each function $s_i(x)$ are indicated.

In lines 5 to 13 an essential set V is established variable by variable. The train T_V is represented by its complement Y : $\bar{Y} = T_V$. In lines 8 to 10 a variable x is selected. Line 8 determines the train of the kernel $\bar{Y} \cup \{x\} = T_V \cup \{x\}$, which is identical to the train of $V \cup \{x\}$ by theorem 2.8.

Actually TRAIN returns the set $T_{V\cup\{x\}} \cap Y = T_{V\cup\{x\}} \backslash T_V$ i.e. only the extension of the train T_V is determined. The set Z is used to restrict the variables which have to be considered in line 8. By corollary 2.5 any variable $x_j \in T$ does not yield an extension of T_V larger than T, so x_j can be left out of consideration. Hence line 11 removes the complete set T from Z. Line 12 adds the selected variable $\hat{x}$ to the kernel V and removes the associated extension $\hat{T}$ of the train from Y such that the property $\bar{Y} = T_V$ is retained. Consequently, if Y is the empty set in line 5, an essential set V is obtained. Line 15 deletes possibly superfluous variables in the kernel to obtain a minimal essential set.

3.3.2. The procedure TRAIN

The procedure TRAIN is given below. TRAIN(Y,V) determines the extension of some train $T_W = \bar{Y}$ if V is added to the kernel.

The set NOV contains variables which can be added to the train:

```
    procedure TRAIN(Y,V)
       begin
 1     R ← φ; NOV ← V;
 2     while NOV ≠ φ do
          begin
 3           choose x ∈ NOV; NOV ← NOV\{x};
 4           if x ∈ Y then
                begin
 5              Y ← Y\{x}; R ← R ∪ {x};
 6              for each s ∈ S do
                   begin
 7                 Y(s) ← Y(s)\{x};
 8                 if |Y(s)| = 1 then NOV ← NOV ∪ Y(s)
                   end
                end
 9           if NOV = φ then TRANSFORM(Y,NOV)
          end
10        TRAIN ← R; Y ← Y ∪ R;
11        for each x ∈ R do
12           for each s ∈ S(x) do Y(s) ← Y(s) ∪ {x}
       end TRAIN
```

mostly novices but also new variables of the kernel. Such a variable is chosen in line 3 while line 4 tests whether the variable is still in Y. The test is required because the variable may have been a novice by virtue of two or more equations. Line 7 updates the datastructure and line 8 detects possibly new novices.

During the determination of a train each function $s_i(x)$ supplies not more than one novice. For a novice x_j arises only if some variable is removed from $Y(s_i(x))$. As long as $s_i(x)$ is a novice function we have $NOV \neq \phi$ and the current vector function $s(x)$ is not transformed to an equivalent one in line 9. Consequently the novice function $s_i(x))$ is retained until the novice itself is removed from $Y(s_i(x))$. Then we have $|Y(s_i(x))| = 0$ in line 8. From then $s_i(x)$ is left unaffected in any execution of TRANSFORM as we will see. Hence the number of variables put into NOV during the determination of a train is bounded by the number of functions plus the number of variables in the kernel. Because in each execution of the while-loop in lines 2 to 9 one variable is removed from NOV, we conclude that after a finite number of executions NOV is exhausted and the while-loop is terminated.

Before the termination of the while-loop at least once procedure TRANSFORM is called. TRANSFORM may create a novice which is put into NOV such that the execution of the while-loop is continued. Note that the novices supplied by TRANSFORM do not invalidate the above argumentation for the termination of the while-loop. If TRANSFORM creates no novice it ends up with a vector function satisfying the condition of theorem 2.10 showing that no novice exists and the train has been determined. The extension of the train, $R = T_{\bar{Y} \cup V} \cap Y = T_{\bar{Y} \cup V} \backslash \bar{Y}$, is assigned to TRAIN in line 10. Next the datastructure is restored such that the situation before calling TRAIN is recovered.

3.3.3. The procedure TRANSFORM

We will confine ourselves with an informal description of the procedure TRANSFORM as the procedure is almost straightforward. Only some crucial points will be elucidated. TRANSFORM applies Gauss-Jordan elimination to the linear functions $s_L(x)$ in order to establish a unit matrix of maximal rank in A_{NN} (see eq.2.1). The creation of one column of the unit matrix is called an "elimination step". In each elimination step we check whether a novice function arises. The check is constrained to the rows updated in the elimination step. The detection of a novice equation causes interruption of the elimination process after the current elimination step. The novice(s) are put into *NOV* and TRANSFORM is terminated. Otherwise the elimination process is continued until the unit matrix has maximal rank, i.e. the rank of A_{NN}. Obviously the Gauss-Jordan elimination implies a non-singular transformation matrix ϕ assuring that an equivalent vector function is obtained. If still no novice function is created theorem 2.10 assures that no novice exists. Then TRANSFORM is terminated with an empty set *NOV*.

The pivots of the elimination steps are obtained from A_{NN} and thus associated with nonfollowers in columns. Consequently the linear functions dependent on followers only are unaffected by the transformation.

The equivalent vector function s(x) formed by TRANSFORM is saved, for it is much more efficient to proceed with s(x) than with the original vector function $\hat{s}(x)$. For example during the determination of T_ϕ in line 3 of MES a vector function $s_L(x)$ is obtained inducing a unit matrix in A_{NN} (where the partition in eq.2.1 is induced by T_ϕ which is assumed to differ from X). In a next execution of TRANSFORM we have an other partition as the current partial train differs from T_ϕ. Some columns of the unit matrix may be removed from A_{NN} but a number of those columns may still be present. The latter columns can be exploited again if they are combined with appropriate other columns, to be established by elimination steps. If we neglect this point and always start from the original vector function $\hat{s}(x)$, or if we do not exploit the columns of the unit matrix still present, we will perform many elimination steps repeatedly.

3.4. Operationscount

The number of operations required by TRANSFORM is mainly determined by the Gauss-Jordan elimination. If TRANSFORM executes e elimination steps, it requires O(e n$|L|$) operations.

Let $|R|$ denote the cardinality of R at the termination of the procedure TRAIN. Lines 3 to 8 and lines 11 and 12 require O(n$|R|$)

operations. Line 9 requires $O(n|L|^2)$ operations if T_ϕ is determined. In any subsequent execution of TRANSFORM only for the columns disappeared from the unit matrix new columns have to be established. The number of new columns required in one execution of TRAIN does not exceed the cardinality of R. Hence line 9 requires $O(n|R||L|)$ operations, except for the first execution of TRAIN.

Line 3 of procedure MES requires $O(n(|L|+|T_\phi|)|L|)$ operations. The selection of one new variable for the kernel in lines 7 to 11 requires $O(n|\hat{T}||Z| + n|\hat{T}||L||Z|)$ operations. Because the cardinalities of all largest extensions of the train sum to $|\hat{X}|$, lines 5 to 13 require $O(n^2|\hat{X}|(|L|+1))$ operations. Line 15 requires $O(n|\hat{X}||V|(|L|+1))$ operations. Hence the determination of a minimal essential set by MES requires $O(n^4)$ operations, or,if no transformations are applied,$O(n^3)$ operations.(Compare the $O(n^3)$ algorithm in [3.5] and the $O(n^4)$ algorithm in [3.6]).

3.5. Some examples and results

The first example concerns a structure matrix obtained from [3.7]. Kevorkian achieves a BLT form with border width 5. In figure 3.1 a BLT form with two border columns is given. The example shows that indeed sometimes a smaller border can be obtained if the condition of a complete matching is dropped. (No transformations are applied as the types and values of the Jacobian coefficients are unknown.)

The next example is an illustration to line 15 of MES in particular where the minimality of the essential set is established. See figure 3.2. The first variable selected for the kernel is x_1 or x_5. Assume x_1 is taken (x_5 would give the same result). The kernel is completed with two arbitrary variables from the set $\{x_2, x_3, x_4\}$ to an essential set. Figure 3.2 shows the BLT form corresponding to the essential set $\{x_1, x_2, x_3\}$. Line 15 tests whether the subsets $\{x_1, x_2\}$, $\{x_1, x_3\}$ and $\{x_2, x_3\}$ are essential. As only x_1 can be dropped, the last set is minimal essential.

Figure 3.3 shows a circuit which may be derived from a Darlington stage with the (bipolar) transistors in active mode. Assume we describe the current source in the transistor models by $i_C = h_{FE}i_B$: $i_4 = h_{FE}i_3$ and $i_8 = h_{FE}i_7$. Without transformations we find a minimum essential set of two variables, while with the application of transformations we find such a set of one variable. In the latter

	06	11	10	05	14	12	13	16	17	15	25	18	27	20	19	28	21	23	26	22	02	30	29	01	04
01	1																							1	
21	1	1																							
03		1	1																						
02	1		1	1																				1	
06	1				1																			1	1
04	1					1																		1	1
05						1	1																		
08							1	1																	
29								1	1																
07				1			1			1															
24				1	1						1														1
23				1						1		1													
27							1				1		1												
12									1			1		1											
11									1			1			1										
28									1				1			1									
15				1							1			1			1								
13							1								1		1	1							
16															1		1		1						
14															1		1			1					
10												1						1			1				
26																					1	1			
20							1															1	1		
19																1					1		1		
25																		1	1	1					

Fig. 3.1. Example obtained from [3.7], fig.10b, p. 503.

	x_4	x_5	x_1	x_2	x_3
s_1	*		*		
s_2		*		*	*
s_3		*	*	*	*
s_4	*	*	*	*	*
s_5	*	*	*	*	*

	x_5	x_1	x_4	x_2	x_3
s_2	*			*	*
s_3	*	*		*	*
s_1		*	*		
s_4	*	*	*	*	*
s_5	*	*	*	*	*

Fig. 3.2 Procedure MES establishes first the essential set $\{x_1,x_2,x_3\}$ (left) and subsequently, in line 15, the minimal essential set $\{x_2,x_3\}$ (right).

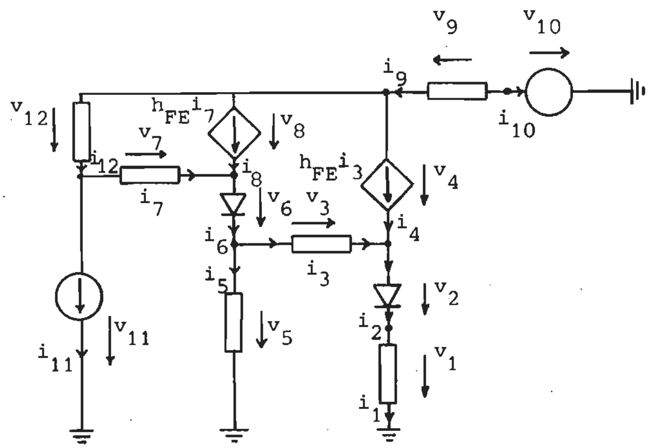

Fig. 3.3. Darlington stage.

case any of the variables v_1, i_1, v_2, i_2, v_3, i_3, i_4 may be the essential variable. E.g. these variables form a partial train of i_4 which can be extended with v_5, i_5, i_6, v_6. Then we have to apply a transformation to the pair of linear functions $i_8 - h_{FE}i_7 = 0$ and $i_8 + i_7 - i_6 = 0$. We may obtain the novice function $(1 + h_{FE})i_7 - i_6 = 0$ and extend the partial train with i_7, i_8, v_7, v_{11}, i_{12} (observe $i_{11} \in T_\phi$), v_{12}, v_8, v_4, i_9, v_9 en i_{10}. Of course the above transformation could have been avoided with a transistor model imposing the equation $i_C = \alpha_E i_E$, i.e. $i_8 = \alpha_E i_6$. Obviously the transformation feature of the algorithm relieves the user from having to be too smart in choosing transistor models. The same applies to the choice of the representation of the Kirchhoff current and voltage equations.

Finally figure 3.4 shows the structure matrix associated with a TTL NAND gate where an Ebers-Moll model is used for the transistors. The essential set obtained by the algorithm MES has appeared to be minimum. The columns of the unit matrix obtained by TRANSFORM are easily recognizable.

The algorithm has been applied to test data concerning vector functions with and without linear functions. In almost all cases the essential set found is even minimum. The execution time of a Fortran

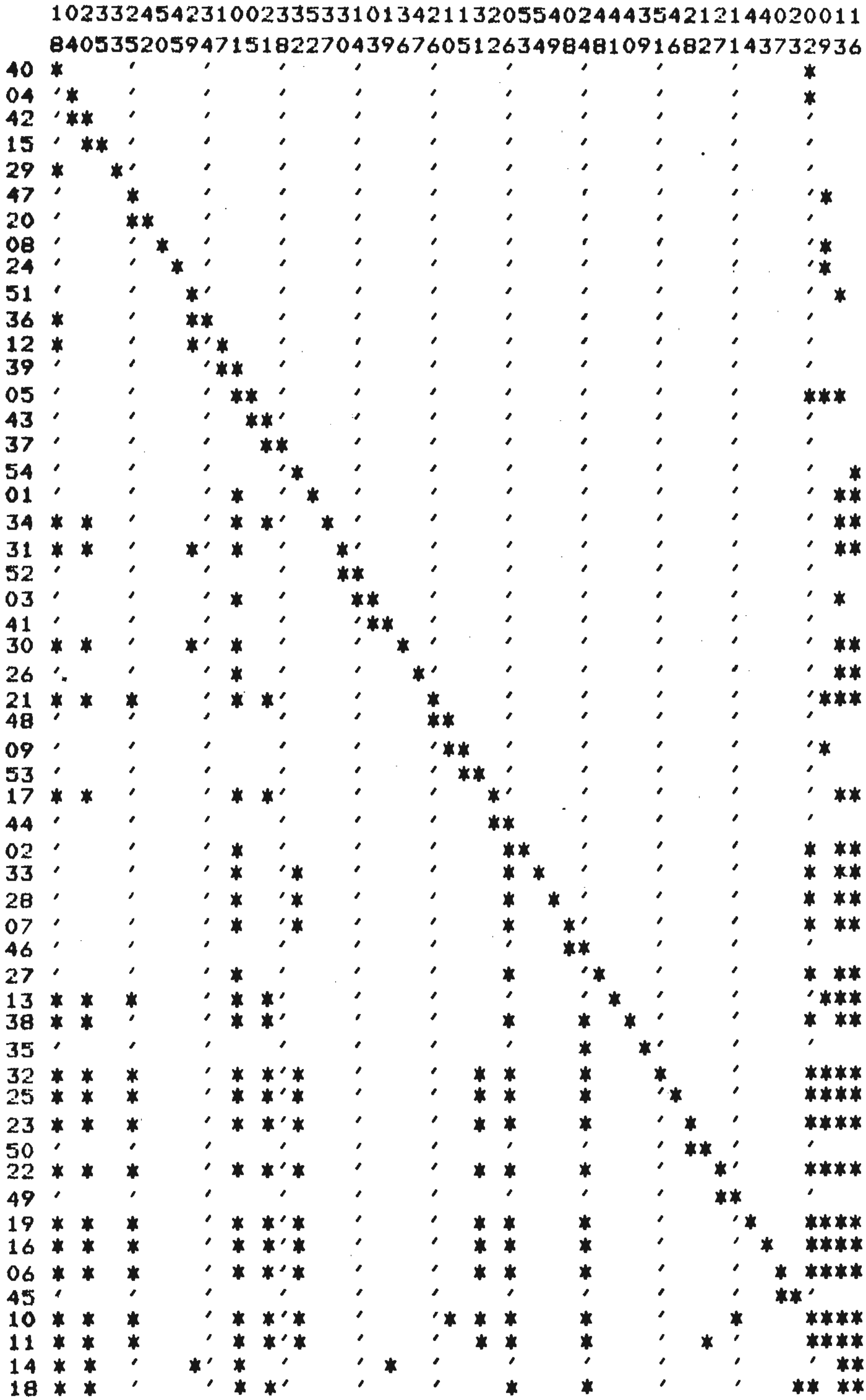

Fig. 3.4. Structure matrix of a TTL NAND gate.

implementation of the algorithm on the minicomputer PDP 11/60 ranges from 60ms. for n=6 to 30.7s. for n=57. As could be expected the execution time depends strongly on the cardinality of the essential set being found.

In the above cases the structure matrix was sparse. An interesting test is the application of the algorithm to vector functions associated with a completely filled structure matrix and including no linear functions. Then the execution time exhibits the $O(n^3)$ property of the algorithm holding for the case that no transformations can be applied. The execution time ranges from 1.2s. for n=10 to 102s. for n=50, what is slightly less than an $O(n^3)$ increase.

The transformation option is applied when dealing with the equations describing electrical circuits. Examples are the TTL NAND gate and the μA 741, an operational amplifier. Interesting is the decrease of the cardinality of the minimal essential set when the restrictions on the applied method are relieved (see table 3.1).

[3.1] C.M. Vidallon, S. Géronimi, P. Dubouix, "An efficient heuristic for the determination of minimal essential variable sets in large sparse systems of network equations", Proc. ICCC 1980, pp. 624-627.

[3.2] E. Hellerman, D. Rarick, "The partitioned preassigned pivot procedure (P^4)", in Sparse Matrices and their Applications, (D.J. Rose, R.A. Willoughby, Eds.), Plenum, New York, 1972.

TABLE 3.1.

cardinality of minimal essential set

	TTL NAND (n=54)	μA 741 (n=126)
fixed complete matching	10	-
no complete matching, no transformation	6 (2.2s.)	14 (24s.)
transformation of Kirchhoff equations	5 (18.0s.)	9 (97s.)
transformation of all linear functions	4 (19.4s.)	8 (97s.)

[3.3] G.J. Sussman, R.M. Stallman, "Heuristic techniques in computer-aided circuit analysis", IEEE Trans. Circuits Syst., Vol. CAS-22, pp. 857-865 (1975).

[3.4] S.L. Hakimi, "On the degrees of vertices of a directed graph", J. Franklin Inst., Vol. 279, pp. 290-308 (1965).

[3.5] A. Sangiovanni-Vincentelli, T.A. Bickart, "Bipartite graphs and an optimal bordered triangular form of a matrix", IEEE Trans. Circuits Syst., Vol. CAS-26, pp. 880-889 (1979).

[3.6] N.M. Znotinas, P.H. Roe, "A breadth-first search method for finding minimum essential sets in digraphs", 22nd Midwest Symp. Circuits Syst., 1979, pp. 26-31.

[3.7] A.K. Kevorkian, J. Snoek, "Decomposition in large scale systems: theory and applications in solving large sets of nonlinear simultaneous equations", in Decomposition of large scale problems. (D.M. Himmelblau, Ed.), North Holland, Amsterdam, 1973, pp. 491-515.

4. PIVOTSTEP SKIPPING

In this and in the following chapters we consider a BLT matrix A. Unless indicated otherwise we assume that A is the Jacobian of a set of equations as given in eq.(1.12) and that A can be partitioned as indicated in eq.(1.9) such that A_{11} is a lower triangular submatrix of dimension t with nonzero diagonal coefficients. It is important to realize that the BLT form implies not only a characteristical pattern of nonzeros but also a pivot order. Many of the definitions introduced here have their counterparts in a general matrix once a pivot scheme is accepted. E.g. a chain satisfying the condition of lemma 4 in [4.1] can be compared to what is called a path in this chapter.

The theory deals with Gauss elimination which is considered as the basic means to compute the Schur complement [4.2] of a BLT matrix. To be more specific the Schur complement is computed by pivotsteps as defined in equations (1.6)-(1.8). Crout elimination does not yield the Schur complement explicitly. However only minor adaptations are necessary to obtain a modification of the Crout scheme such that it generates the Schur complement as well.

4.1. Definitions

We recall that the k^{th} pivotstep involves the operations (compare eq.^{s}.(1.6)-(1.8)):

$$u_{kj} \leftarrow u_{kj}/l_{kk} \qquad \text{for} \quad k < j \le n \tag{4.1}$$

$$u_{ij} \leftarrow u_{ij} - l_{ik}u_{kj} \qquad \text{for} \quad k < i < j \le n \tag{4.2}$$

$$l_{ij} \leftarrow l_{ij} - l_{ik}u_{kj} \qquad \text{for} \quad k < j \le i \le n \tag{4.3}$$

where l_{kk} is the k^{th} pivot. A subset of the operations in (4.2) and (4.3) is obtained if i is kept fixed. Such a subset is called a "pivotsubstep". A pivotsubstep concerns operations on coefficients in one row.

The execution of a pivotstep (or a pivotsubstep) will be skipped if a condition of the form $|l_{kk}| \ge \theta_k$ is satisfied. The right hand side θ_k is called a "threshold".

An ordered set of nonzero coefficients of the BLT matrix A:

$$\{a_{i_0 i_1}, a_{i_1 i_1}, a_{i_1 i_2}, a_{i_2 i_2}, \ldots, a_{i_\ell i_\ell}, a_{i_\ell i_{\ell+1}}\}$$

with $\ell \geq 0$, $i_{k-1} \neq i_k \neq i_{k+1}$ and $i_k \leq t$ for $1 \leq k \leq \ell$, is called a "path" $P_{i_0 i_{\ell+1}}$ of length ℓ. A path is characterized by the pivots it contains: given the pivots (and i_0 and $i_{\ell+1}$) only one choice for the path $P_{i_0 i_{\ell+1}}$ remains (see figure 4.1). A different set of pivots may imply a different path. However not each set of pivots determines a path. Only if the appropriate off-diagonal coefficients are nonzero a path exists. The order of the pivots in a path is always reversed to the pivot order induced by the BLT form.

With a path $P_{i_0 i_{\ell+1}}$ a "term" $\psi_{i_0 i_{\ell+1}}$ is associated, defined by

$$\psi_{i_0 i_{\ell+1}} = (-1)^{\ell} a_{i_0 i_1} \, a^{-1}_{i_1 i_1} \, a_{i_1 i_2} \, a^{-1}_{i_2 i_2} \ldots a^{-1}_{i_\ell i_\ell} \, a_{i_\ell i_{\ell+1}}$$

A term is the product of all off-diagonal coefficients in the path divided by the product of the negatives of all pivots in the path. Note that a term is always nonzero for finite pivot values. The set of all terms ψ_{ij} for some i and j is denoted by Ψ_{ij}. If two terms in

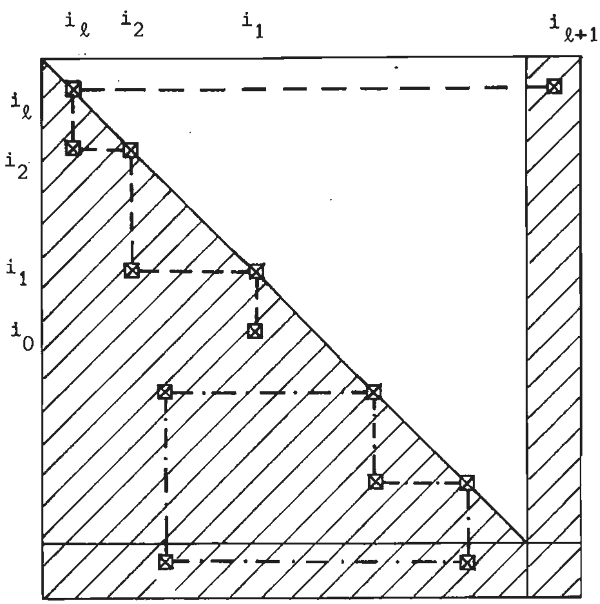

Fig.4.1. A Bordered Lower Triangular matrix containing a path (– – –) and a cycle (—·—·).

Ψ_{ij}, ψ^1_{ij} and ψ^2_{ij}, are related by $|\psi^1_{ij}| \le \Delta|\psi^2_{ij}|$ for some Δ, $0 < \Delta \le 1$, we say that ψ^1_{ij} is "dominated" by ψ^2_{ij} for this value of Δ. ψ^2_{ij} is called a "dominator" of ψ^1_{ij}.

If the length of the path P_{ij} is one or more and a_{ij} is nonzero then $C_{ij} = P_{ij} \cup \{a_{ij}\}$ is called a "cycle". The length of a cycle is the length of its generating path. Figure 4.1 illustrates the definitions.

4.2. The domination principle

Let $\tilde{A}_{22}$ be the Schur complement of A_{11} in A, i.e. $\tilde{A}_{22} = A_{22} - A_{21}A_{11}^{-1}A_{12}$ [4.2]. If $\tilde{a}_{vw}$ is a coefficient of $\tilde{A}_{22}$ and if a path P_{vw} exists then $\tilde{a}_{vw}$ is nonzero except for special values of the coefficients of A [4.1]. The following theorem deals with the value of the coefficients of the Schur complement.

Theorem 4.1: Let $\tilde{A}_{22}$ be the Schur complement of the lower triangular submatrix A_{11} in A. Any coefficient $\tilde{a}_{vw}$ of $\tilde{A}_{22}$ satisfies:

$$\tilde{a}_{vw} = \sum_{\Psi_{vw}} \psi_{vw} \tag{4.4}$$

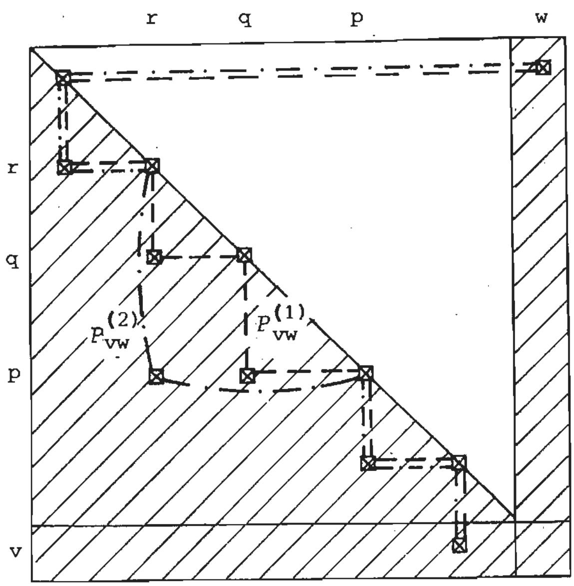

Fig.4.2. Two paths differing in one pivot.

Proof: The proof is by induction on t, the dimension of A_{11}. For t = 0 the theorem is trivial. Now let t = m and let A_{11}^{m-1} denote the submatrix of A_{11} consisting of the first m-1 rows and columns of A_{11}. Let Ψ_{vw}^{m-1} be the set of terms ψ_{vw} associated with a_{vw} or with paths passing through pivots of A_{11}^{m-1} exclusively. If Gauss elimination is applied to A then after m-1 pivot steps the Schur complement of A_{11}^{m-1}, denoted by $\tilde{A}_{22}^{m-1}$, is computed. The induction hypothesis applies to any coefficient $\tilde{a}_{vw}^{m-1}$ of $\tilde{A}_{22}^{m-1}$:

$$\tilde{a}_{vw}^{m-1} = \sum_{\Psi_{vw}^{m-1}} \psi_{vw}$$

With the execution of the m^{th} pivot step the Schur complement $\tilde{A}_{22}^{m} = \tilde{A}_{22}$ can now be computed. Any coefficient $\tilde{a}_{vw}^{m}$ of $\tilde{A}_{22}^{m}$ satisfies:

$$\tilde{a}_{vw}^{m} = \tilde{a}_{vw}^{m-1} - \tilde{a}_{vm}^{m-1} \; (\tilde{a}_{mm}^{m-1})^{-1} \; \tilde{a}_{mw}^{m-1}$$

We have $\tilde{a}_{vm}^{m-1} = a_{vm}$ and $\tilde{a}_{mm}^{m-1} = a_{mm}$ because the m^{th} column is not in the border and thus not subjected to updating. By the induction hypothesis follows:

$$\tilde{a}_{vw}^{m} = \sum_{\Psi_{vw}^{m-1}} \psi_{vw} + \sum_{\Psi_{mw}^{m-1}} (-a_{vm} a_{mm}^{-1} \psi_{mw}) \qquad (4.5)$$

For each term ψ_{mw} the product $-a_{vm} a_{mm}^{-1} \psi_{mw}$ is a term ψ_{vw}. Such a term is associated with a path passing through a_{mm} and possibly some of the first m-1 pivots. Both sums in (4.5) together yield all terms associated with a_{vw} or with paths passing through pivots of A_{11}^{m} exclusively. Hence we have

$$\tilde{a}_{vw}^{m} = \sum_{\Psi_{vw}^{m}} \psi_{vw} \qquad (4.6)$$

Because $\tilde{a}_{vw}^{m}$ equals $\tilde{a}_{vw}$ and Ψ_{vw}^{m} equals Ψ_{vw} (4.6) is identical to (4.4).

From theorem 4.1 we see that any coefficient $\tilde{a}_{vw}$ of the Schur complement is equal to a sum of terms. We may expect that if small terms are omitted, the remaining terms constitute a good approximation of $\tilde{a}_{vw}$. In order to delete small terms we study what we call the "domination principle". The domination principle implies that only dominated terms are deleted: if $|\psi_{vw}^{1}| \leq \Delta|\psi_{vw}^{2}|$ holds then

ψ^1_{vw} can be deleted. In section 4.6 the implications of the principle for the error of the solution are discussed.

4.3. The thresholds

Let the paths P^1_{vw} and P^2_{vw}, associated with ψ^1_{vw} and ψ^2_{vw} differ in one pivot. Arbitrarily let $a_{qq} \in P^1_{vw}$ and $a_{qq} \notin P^2_{vw}$ and let $a_{kk} \in P^1_{vw}$ for all $a_{kk} \in P^2_{vw}$. The the symmetric difference $P^1_{vw} \oplus P^2_{vw}$ appears to be the cycle $C_{pr} = \{a_{pq}, a_{qq}, a_{qr}, a_{pr}\}$ of length one (see figure 4.2). The ratio of the term is:

$$\frac{\psi^1_{vw}}{\psi^2_{vw}} = - \frac{a_{pq} a^{-1}_{qq} a_{qr}}{a_{pr}}$$

ψ^2_{vw} dominates ψ^1_{vw} if the ratio does not exceed Δ or, equivalently, if

$$|a_{qq}| \geq \Delta^{-1} |a_{pq} a^{-1}_{pr} a_{qr}| \tag{4.7}$$

is satisfied. The condition is important not only for the comparison of ψ^1_{vw} and ψ^2_{vw}. Many pairs of paths P^1_{ij}, P^2_{ij} may exist such that $P^1_{ij} \oplus P^2_{ij} = C_{pr}$. Thus if inequality (4.7) holds all terms ψ^1_{ij} associated with paths P^1_{ij} can be deleted.

The common characteristic of the pairs P^1_{ij}, P^2_{ij} is the cycle C_{pr}. There may be more such cycles of length one containing a_{qq}. Exactly, any nonzero a_{pq} in the pivot column combined with any nonzero a_{qr} in the pivot row defines such a cycle if a_{pr} is nonzero as well. (For the moment suppose that the case ($a_{pq} \neq 0 \wedge a_{qr} \neq 0 \wedge a_{pr} = 0$) does not occur). Each such cycle defines a class of pairs of terms. The possible deletion of one of the terms of every pair is controlled by the condition of the defining cycle. All conditions arising from cycles passing through a_{qq} contain the pivot a_{qq} and are of the form (4.7). Hence it is possible that all conditions are satisfied. Then all terms depending on a_{qq} are dominated and can be deleted. This is formally stated in the following lemma.

<u>Lemma 4.2</u>: Let q be given and let for all p such that $a_{pq} \neq 0$ and all r such that $a_{qr} \neq 0$ the coefficient a_{pr} be nonzero and inequality (4.7) be satisfied. Then each term ψ^1_{ij} containing a^{-1}_{qq} has a dominator ψ^2_{ij} individually associated with it.

Proof: We introduce the notation ψ_{ipp} to denote the product $-\psi_{ip}a_{pp}^{-1}$ for $i \neq p$, while $\psi_{iii} \equiv 1$. In the same way ψ_{rrj} denotes $-a_{rr}^{-1}\psi_{rj}$ for $r \neq j$. If ψ_{ij}^{1} contains a_{qq}^{-1} then for some p and r we have $\psi_{ij}^{1} = -\psi_{ipp}a_{pq}a_{qq}^{-1}a_{qr}\psi_{rrj}$. The supposition $a_{pr} \neq 0$ assures the existence of the term $\psi_{ij}^{2} = \psi_{ipp}a_{pr}\psi_{rrj}$. Hence $\psi_{ij}^{1} = -\dfrac{a_{pq}a_{qq}^{-1}a_{qr}}{a_{pr}}\psi_{ij}^{2}$ and from inequality (4.7) follows $|\psi_{ij}^{1}| \leq \Delta|\psi_{ij}^{2}|$. Any other term ψ_{ij}^{3} depending on a_{qq} has a dominator ψ_{ij}^{4} different from ψ_{ij}^{2} by the construction of the dominator.

The foregoing analysis has a meaning for both Gauss and Crout elimination as the deletion of all terms depending on a_{qq} is identical to the skipping of the q^{th} pivotstep in Gauss elimination. Thus the pivotstep is skipped if all conditions of the form (4.7) for fixed q and different p and r are satisfied. Clearly it suffices to test the condition with the largest right-hand side. This condition defines the threshold θ_q of the pivot $l_{qq} \equiv a_{qq}$ (BLT form!):

$$\theta_q = \Delta^{-1} \max_{p,r} \left[a_{pq}a_{pr}^{-1}a_{qr} \right] \tag{4.8}$$

and the q^{th} pivotstep is skipped if $|a_{qq}| \geq \theta_q$ holds.

The effect of skipping a pivotstep is identical to perturbations of appropriate matrix coefficients. For some q let a_{pq} and a_{qr} be nonzero and let a_{pr} be perturbed i.e. $\hat{a}_{pr} = a_{pr} + \delta a_{pr}$. Any set of terms Ψ_{vw} can be partitioned into three subsets $\hat{\Psi}_{vw}$, $\mathring{\Psi}_{vw}$ and $\bar{\Psi}_{vw}$: all terms ψ_{vw} containing the factor $\hat{a}_{pr}$ are in $\hat{\Psi}_{vw}$, all terms ψ_{vw} containing $a_{pq}a_{qq}^{-1}a_{qr}$ are in $\mathring{\Psi}_{vw}$ and the remaining terms ψ_{vw} are in $\bar{\Psi}_{vw}$. The expression for a coefficient in the Schur complement in terms of these subsets is:

$$\tilde{a}_{vw} = \sum_{\Psi_{vw}} \psi_{vw} = \sum_{\bar{\Psi}_{vw}} \psi_{vw} + \sum_{\hat{\Psi}_{vw}} \psi_{vw} + \sum_{\mathring{\Psi}_{vw}} \psi_{vw}$$

Each term in $\hat{\Psi}_{vw}$ is related to a unique term in $\mathring{\Psi}_{vw}$: if $\psi_{vpp}\hat{a}_{pr}\psi_{rrw} \in \hat{\Psi}_{vw}$ then $-\psi_{vpp}a_{pq}a_{qq}^{-1}a_{qr}\psi_{rrw} \in \mathring{\Psi}_{vw}$ and vice versa. If δa_{pr} equals $a_{pq}a_{qq}^{-1}a_{qr}$, the sum of two related terms is $\psi_{vpp}(\hat{a}_{pr} - a_{pq}a_{qq}^{-1}a_{qr})\psi_{rrw} = \psi_{vpp}a_{pr}\psi_{rrw}$. Apparently δa_{pr} causes the cancellation of all terms in $\mathring{\Psi}_{vw}$. Because the foregoing holds for any coefficient of the Schur complement we conclude that the deletion of all terms containing $a_{pq}a_{qq}^{-1}a_{qr}$ is equivalent to the perturbation $\delta a_{pr} = a_{pq}a_{qq}^{-1}a_{qr}$ added to a_{pr}. The deletion of other terms depending on a_{qq} is equivalent to perturbations of other coefficients of A. We see that skipping of the q^{th} pivotstep implies that the Schur complement of the perturbed matrix $(A + \delta A)$ is computed where δA has coefficients $\delta a_{pr} = a_{pq} a_{qq}^{-1} a_{qr}$ for $p \neq q$ and $r \neq q$. The definition of the threshold θ_q (equation (4.8)) assures that for $|a_{qq}| \geq \theta_q$ the perturbations satisfy:

$$|\delta a_{pr}| \leq \Delta |a_{pr}| \tag{4.9}$$

Dependent on the accuracy factor Δ the perturbations are small relative to the original matrix coefficients.

The case $a_{pr} = 0$ given nonzero a_{pq} and a_{qr} is not yet discussed. Then the cycle C_{pr} does not exist and no condition is obtained. But a longer cycle passing through another pivot as well, can be searched. Such a cycle, if present, yields a condition for the product of the concerned pivots. Though this approach is executable, it has severe drawbacks. Firstly the computational complexity to find the cycles increases from $O(n^3)$ to $O(n^4)$ or more, depending on the length of the cycles. Secondly more and more complex conditions control the execution of pivotsteps in this approach.

Another approach is possible if we exploit inequality (4.17) given in section 4.6. The inequality constitutes an upper bound to the error of the solution. (4.17) can be applied if skipping of a pivotstep induces a perturbation matrix δA satisfying $\|\delta A\| \leq \Delta \|A\|$. The condition admits perturbations δa_{pr} in spite of that a_{pr} is zero. E.g. all coefficients of δA may be equal to $\frac{\Delta}{n}\|A\|$. Hence we may require that δa_{pr} satisfies:

$$|\delta a_{pr}| = |a_{pq} a_{qq}^{-1} a_{qr}| \leq \frac{\Delta}{n} \|A\| \tag{4.10}$$

The same condition for pivot a_{qq} would be obtained if the value of a_{pr} was replaced by $\frac{1}{n}\|A\|$ regardless whether a_{pr} is zero or not (compare (4.7)).

The perturbation δa_{pr} satisfying (4.10) has no relation to the size of the coefficient a_{pr}. In order to exploit a possibly large value of a_{pr} we may use a value like $\frac{1}{n}\|A\|$ only if a_{pr} is zero or very small. E.g. for any λ with $0 \leq \lambda \leq 1$ the right-hand side $\Delta\{\frac{\lambda}{n}\|A\| + (1-\lambda)|a_{pr}|\}$ can be used in inequality (4.10). The property $\|A + B\| \leq \|A\| + \|B\|$ implies that the associated perturbation matrix δA satisfies:

$\|\delta A\| \leq \Delta\lambda\|A\| + \Delta(1-\lambda)\|A\| = \Delta\|A\|$

In the latter approach the domination principle is abandoned. The approach is based on the condition $\|\delta A\| \leq \Delta\|A\|$ which is connected with inequality (4.17). However inequality (4.18), which constitutes a tighter bound to the error of the solution, cannot be applied. The following section discusses an approach maintaining the domination principle. The thresholds computed with this approach are such that the skipping of a pivotstep induces a matrix δA with the property that inequality (4.18) can be applied.

4.4. Dependable references

If the absolute values of the pivots are bounded from below the domination principle can be applied without increasing the computational complexity and the disadvantage of more, and more complex, conditions. For that purpose we introduce the notion of a "dependable reference". A nonnegative constant ε_{ij} is a dependable reference if each term ψ^1_{vw} with $v > t$ and $w > t$ containing a term ψ_{ij}, not identical to a_{ij} and satisfying $|\psi_{ij}| \leq \Delta\varepsilon_{ij}$, is dominated by a term ψ^2_{vw} associated with a path P^2_{vw} with the property $P^2_{vw} \cap P_{ij} = \Phi$, where P_{ij} is associated with ψ_{ij}. Dependable references can be used to compute thresholds. For instance if ψ_{pr} is the term $-a_{pq}a^{-1}_{qq}a_{qr}$ then $|\psi_{pr}| \leq \Delta\varepsilon_{pr}$ implies $|a_{qq}| \geq \Delta^{-1}|a_{pq}|.\varepsilon^{-1}_{pr}.|a_{qr}|$. The inequality has the same form as (4.7); ε_{pr} plays the role of a_{pr}. Clearly for $\varepsilon_{pr} = |a_{pr}| > 0$ the inequality becomes indentical to (4.7). Hence we state without proof:

Corollary 4.3: For each i and j $\varepsilon_{ij} = |a_{ij}|$ is a dependable reference.

If a_{ij} is zero the dependable reference obtained from the corollary is useless. The following theorem gives a method to compute positive dependable references from other positive dependable references in an iterative way. First two functions are defined. Let η_q for $q \leq t$ denote the lower bound to the absolute value of pivot a_{qq}: $|a_{qq}| \geq \eta_q > 0$. The functions are:

$$R_{ij} \triangleq \min_{\substack{k \neq i \\ a_{ki} \neq 0}} \left[\frac{\varepsilon_{kj}}{|a_{ki}|} \right] \eta_i \qquad \text{for } i \leq t, \quad R_{ij} \triangleq 0 \quad \text{for } i > t$$

$$C_{ij} \triangleq \min_{\substack{\ell \neq j \\ a_{j\ell} \neq 0}} \left[\frac{\varepsilon_{i\ell}}{|a_{j\ell}|} \right] \eta_j \qquad \text{for } j \leq t, \quad C_{ij} \triangleq 0 \quad \text{for } j > t$$

If for $i \leq t$ a_{ki} is zero for all k then we define $R_{ij} \triangleq \infty$. In the same way we define $C_{ij} \triangleq \infty$ if for $j \leq t$ $a_{j\ell}$ is zero for all ℓ. However practical matrices are usually irreducible and then the cases $R_{ij} = \infty$ and $C_{ij} = \infty$ do not occur. Figure 4.3 illustrates which matrix coefficients and dependable references occur in the functions.

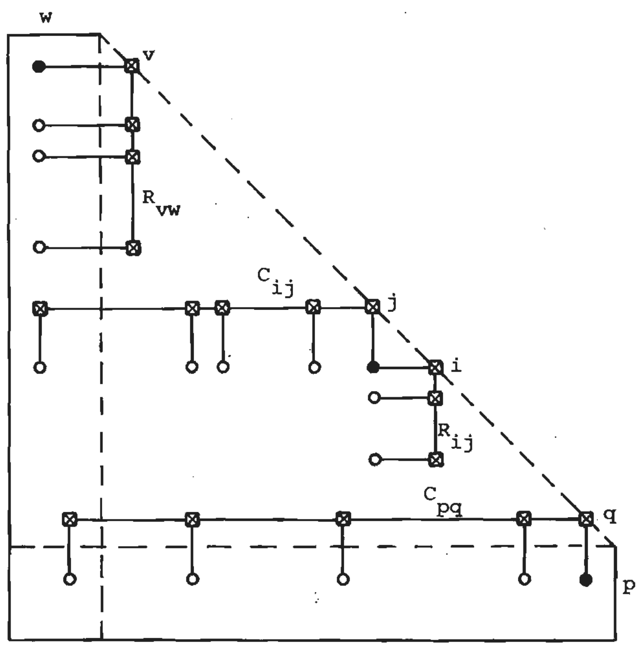

Fig. 4.3. The computation of dependable references according to eq. (4.11).

The border columns are indicated left to the other columns to make the illustration more clear. A ○ *denotes a dependable reference. The dependable references denoted by* ● *are computed using the coefficients and dependable references connected to each other by solid lines. Note that* R_{pq} *and* C_{vw} *are zero (*$p > t$ *and* $w > t$*).*

Theorem 4.4: Let $|a_{qq}| \geq \eta_q > 0$ hold for all $q \leq t$. Let for given i and j all ε_{kj} and $\varepsilon_{i\ell}$ occurring in R_{ij} and C_{ij} be dependable references. Then

$$\varepsilon_{ij} = \max\,[R_{ij}, C_{ij}, |a_{ij}|] \tag{4.11}$$

is a dependable reference.

Proof: We show that both R_{ij} and C_{ij} are dependable references. $|a_{ij}|$ is a dependable reference according to corollary 4.3. If $i > t$ then R_{ij} is zero and $\varepsilon_{ij} = R_{ij} = 0$ is a dependable reference. For $i \leq j \leq t$ no path P_{ij} exists. Hence regardless its value $\varepsilon_{ij} = R_{ij}$ is a dependable reference. In the cases $j < i \leq t$ and $i \leq t < j$ let P^1_{vw} be some path with $v > t$ and $w > t$ and let P^1_{vw} contain some path P_{ij} not indentical to $\{a_{ij}\}$. Then we have for some k (see figure 4.4):

$$P_{kj} = \{a_{ki}, a_{ii}\} \cup P_{ij} \subset P^1_{vw}$$

The inequality $|\psi_{ij}| \leq \Delta R_{ij}$ implies:

$$|\psi_{kj}| = |a_{ki} a_{ii}^{-1} \psi_{ij}| \leq |a_{ki}| \eta_i^{-1} \Delta R_{ij} \leq \Delta\varepsilon_{kj}$$

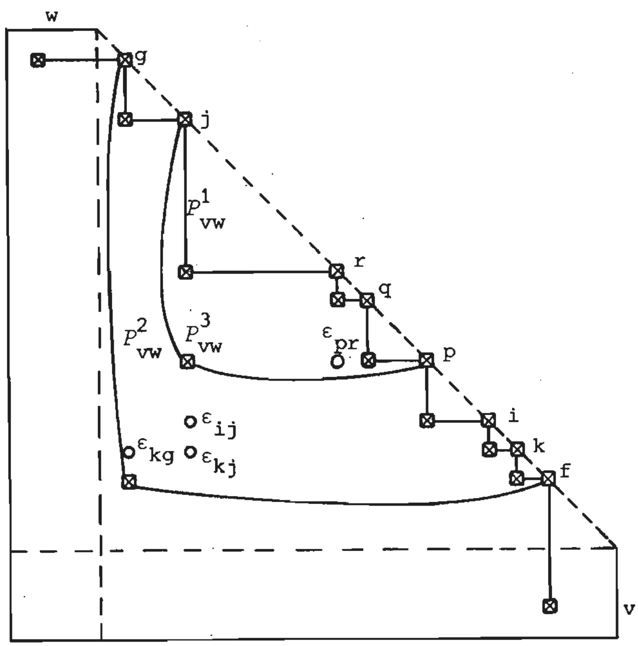

Fig. 4.4. Determination of a dominator of ψ^1_{vw} using dependable references.

ε_{kj} is a dependable reference for it occurs in R_{ij}. Therefore ψ^1_{vw} is dominated by a term ψ^2_{vw} associated with a path P^2_{vw} having the property $P^2_{vw} \cap P_{kj} = \phi$. Consequently $P^2_{vw} \cap P_{ij} = \phi$ holds too and $\varepsilon_{ij} = R_{ij}$ is a dependable reference. The proof that $\varepsilon_{ij} = C_{ij}$ is a dependable reference proceeds in the same way.

The dependable references can be computed in an iterative way by the procedure DEPREF, which we describe here. The procedure EPS(i,j), used in DEPREF, computes ε_{ij} according to (4.11).

```
   procedure DEPREF
   begin
1  for t<i≤n, t<j≤n do EPS(i,j);
2  for m=1 step 1 until t-1 do
      begin
3     for i=t-m+1, t<j≤n do EPS(i,j);
4     for t<i≤n, j=m do EPS(i,j);
5     for i=t-m+j, 1≤j≤m do EPS(i,j)
      end
   end
```

Figure 4.5 shows the order in which the dependable references are computed by DEPREF.

Theorem 4.5: The computation of a dependable reference in DEPREF requires only dependable references computed before in DEPREF.

Proof: In line 1 we have $R_{ij} = 0$ and $C_{ij} = 0$ because $i > t$ and $j > t$. Thus no dependable references are required. Consider the execution of lines 3, 4 and 5 for $m = \hat{m}$. In line 3 $j > t$ implies $C_{ij} = 0$. R_{ij} uses ε_{kj} with $i < k \le n$ as a_{ki} is zero for $k < i$ (BLT form). ε_{kj} with $t < k \le n$ is computed in line 1 ($j > t$!). ε_{kj} with $i < k \le t$ is computed in line 3 for $m = \bar{m}$, where $\bar{m}$ follows from $k = t - \bar{m} + 1$. Because of $t - \hat{m} + 1 = i < k = t - \bar{m} + 1 \le t$ we have $1 \le \bar{m} < \hat{m}$. So ε_{kj} with $i < k \le t$ is obtained by a previous execution of line 3. In line 4 we have $R_{ij} = 0$, while C_{ij} uses $\varepsilon_{i\ell}$ with $1 \le \ell < j$ or $t < \ell \le n$. $\varepsilon_{i\ell}$ with

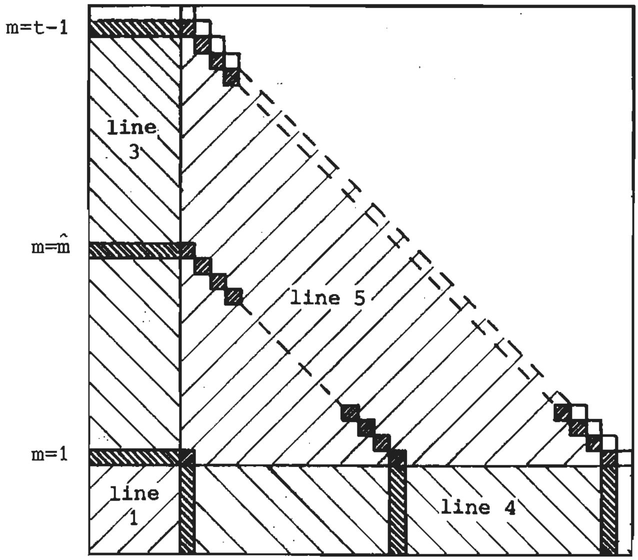

Fig. 4.5. *DEPREF computes the dependable references in the order indicated by m.*

$t < \ell \le n$ is computed in line 1 ($i > t$!), while $\varepsilon_{i\ell}$ with $1 \le \ell < j$ is obtained by a previous execution of line 4, namely for $m = \ell < j = \hat{m}$.

In line 5 we have $1 < i \le t$ and $1 \le j \le \hat{m} < t$. R_{ij} uses ε_{kj} with $i < k \le n$. ε_{kj} with $t < k \le n$ is computed in line 4 for $m = j \le \hat{m}$. ε_{kj} with $i < k \le t$ is computed in line 5 for $m = \bar{m}$ ($j \le \bar{m}$ holds because of $k = t - \bar{m} + j \le t$). Because $t - \hat{m} + j = i < k = t - \bar{m} + j \le t$ implies $1 \le j \le \bar{m} < \hat{m}$, it is assured that ε_{kj} with $i < k \le t$ is obtained by a previous execution of line 5.

C_{ij} uses $\varepsilon_{i\ell}$ with $1 \le \ell < j$ or $t < \ell \le n$. $\varepsilon_{i\ell}$ with $t < \ell \le n$ is computed in line 3 for $m = \bar{m}$. From $i = t - \bar{m} + 1 = t - \hat{m} + j \le t$ we obtain $1 \le \bar{m} = \hat{m} - j + 1 \le \hat{m}$. So $\varepsilon_{i\ell}$ with $t < \ell \le n$ is computed during the latest or a previous execution of line 3. $\varepsilon_{i\ell}$ with $1 \le \ell < j$ is computed in line 5 for $m = \bar{m}$ (for $i = t - \bar{m} + \ell \le t$ implies $\ell \le \bar{m}$). Because of

$i = t - \hat{m} + j = t - \bar{m} + \ell \leq t$ we have $1 \leq \ell \leq \bar{m} = \hat{m} + \ell - j < \hat{m}$. So $\varepsilon_{i\ell}$ with $1 \leq \ell < j$ is obtained by a previous execution of line 5.

Line 1 of DEPREF computes ε_{ij} for $t < i \leq n$ and $t < j \leq n$. Because both R_{ij} and C_{ij} are zero the result is $\varepsilon_{ij} = |a_{ij}|$. If a_{ij} is zero no positive ε_{ij} is computed. Theorem 4.4 does not assure that no positive dependable reference exists in this case. For ε_{ij} computed according to equation (4.11) need not be maximal, i.e. some $\hat{\varepsilon}_{ij}$ with $\hat{\varepsilon}_{ij} > \varepsilon_{ij}$ may exist which is dependable too. However we will show that $\varepsilon_{ij} = |a_{ij}|$ is maximal for $t < i \leq n$ and $t < j \leq n$.

Lemma 4.6: Let P^1_{vw} with $t < v \leq n$ and $t < w \leq n$ contain a path P_{ij}, not identical to $\{a_{ij}\}$, and let $|\psi_{ij}| \leq \Delta\varepsilon_{ij}$ be satisfied. Then the term ψ^1_{vw} has a dominator ψ^2_{vw} such that all pivots in P^2_{vw} are contained in P^1_{vw} as well.

Proof: Let the values of all pivots not contained in P^1_{vw} grow very large while the values of the pivots in P^1_{vw} are constant. Then the value of ψ^1_{vw} is constant and moreover $|\psi_{ij}| \leq \Delta\varepsilon_{ij}$ still holds. On the other hand the value of each term depending on a pivot not contained in P^1_{vw} approaches zero. Hence pivot values exist such that $|\psi^1_{vw}| > \Delta|\psi_{vw}|$ for each term ψ_{vw} depending on a pivot not contained in P^1_{vw}. However $|\psi_{ij}| \leq \Delta\varepsilon_{ij}$ assures that ψ^1_{vw} has a dominator ψ^2_{vw}. Apparently ψ^2_{vw} does not depend on a pivot which is not contained in P^1_{vw}.

Lemma 4.7: The dependable reference $\varepsilon_{vw} = |a_{vw}|$ is maximal in the case $t < v \leq n$ and $t < w \leq n$.

Proof: Any term ψ^1_{vw} satisfying $|\psi^1_{vw}| \leq \Delta\varepsilon_{vw}$, has a dominator ψ^2_{vw} which by the definition of ε_{vw} does not depend on any pivot in P^1_{vw} and which by lemma 4.6 does not depend on any pivot not contained in P^1_{vw}. Then the only possibility is $\psi^2_{vw} \equiv a_{vw}$. Consequently any constant larger than $|a_{vw}|$ cannot be a dependable reference.

From the latter lemma we conclude that no positive dependable reference ε_{vw} exists if a_{vw} is zero in the case $t < v \leq n$ and $t < w \leq n$.

On the other hand consider the case that all elements in A_{22} are nonzero. Then all ε_{ij} computed in line 1 of DEPREF are positive. The other dependable references, computed in lines 3, 4 and 5 use the functions R_{ij} and C_{ij} for $i \le t$ or $j \le t$. At least one of the functions gives a positive result if all dependable references occurring in them are positive. Because after the execution of line 1 all dependable references are positive it follows by induction, in view of theorem 4.5, that all dependable references computed by DEPREF are positive.

Although the case that A_{22} has no zero coefficient occurs very rarely, the foregoing is not unimportant. Let A be a matrix with zero coefficients in A_{22} and let matrix $\bar{A}$ be identical to A except for $\bar{A}_{22}$ which contains no zero coefficient. $\bar{A}$ can be considered as a perturbation of A. Although A may have zero dependable references, from the preceding paragraph it appears that for the perturbed matrix $\bar{A}$ all dependable references are positive.

By the way a computed dependable reference ε_{ij} may be larger than $|a_{ij}|$ sometimes. Then in the computation of the thresholds a, probably, small a_{ij} is replaced by the more appropriate value of ε_{ij}. In appendix A some general statements are made concerning the size of dependable references.

In the following section we will pay attention to the relation between a dominated term and its dominator. If no dependable references need be used then lemma 4.2 shows that there is a one-to-one correspondence between dominated terms and dominators. We will show that this property is lost if dependable references are exploited and we will deal with the consequences of that.

4.5. <u>The determination of a dominator</u>

If the threshold θ_q is computed exploiting dependable references then the terms depending on a_{qq} are dominated for $|a_{qq}| \ge \theta_q$. The construction of a dominator of the term ψ^1_{vw} containing the term $\psi_{pr} = -a_{pq}a^{-1}_{qq}a_{qr}$ is easy if a_{pr} is nonzero (see the proof of lemma 4.2). However if a_{pr} is zero and the dependable reference ε_{pr} is exploited, a dominator is not found immediately. A way to determine a dominator is suggested if we combine the definition of a dependable reference and lemma 4.6. The lemma states that the dominator ψ^2_{vw} of ψ^1_{vw} depends only on pivots in P^1_{vw}. On the other hand we have $P^2_{vw} \cap P_{pr} = \phi$ by the definition of ε_{pr}. Consequently all pivots in P^2_{vw} are in $P^1_{vw} \backslash P_{pr}$. If we succeed in extending P_{pr} to a longer path P_{ij} with the restrictions $P_{pr} \subset P_{ij} \subset P^1_{vw}$ and $P_{ij} \cap P^2_{vw} = \phi$, then the set $P^1_{vw} \backslash P_{ij}$ is smaller than $P^1_{vw} \backslash P_{pr}$. Therefore we may try to make P_{ij} as long as possible, subject to the above restrictions. It will appear that in this way a dominator P^2_{vw} can be determined such that all pivots in $P^1_{vw} \backslash P_{ij}$ are in P^2_{vw} as well.

We say that the path P_{kj} is the "forward extension" of P_{ij} following P_{vw} if P_{kj} satisfies $P_{kj} = \{a_{ki}, a_{ii}\} \cup P_{ij} \subset P_{vw}$. Equivalently the path $P_{i\ell} = P_{ij} \cup \{a_{jj}, a_{j\ell}\} \subset P_{vw}$ is called the "backward extension" of P_{ij} following P_{vw}. Let P_{vw}, with $t < v \leq n$ and $t < w \leq n$, contain the path P_{pr}. Consider the series of paths $P_{pr} = P^0, P^1, P^2, \ldots, P^{\omega} = P_{fg}$ with $\omega \geq 0$. For $\omega \geq 1$ and $0 \leq \sigma \leq \omega-1$ let $P^{\sigma+1}$ be the forward extension of $P^{\sigma} = P_{ij}$ following P_{vw} in the case $\varepsilon_{ij} = R_{ij} > |a_{ij}|$ and let $P^{\sigma+1}$ be the backward extension of $P^{\sigma} = P_{ij}$ following P_{vw} in the case $((\varepsilon_{ij} = C_{ij} > |a_{ij}|) \wedge (C_{ij} > R_{ij}))$. If no next path $P^{\omega+1}$ can be obtained in this way, e.g. if $P^{\omega} = P_{fg}$ and $\varepsilon_{fg} = |a_{fg}|$, or if $P^{\omega} = P_{vw}$, the series is terminated. Clearly the series consists always of a finite number of paths. The last path in the series, $P^{\omega} = P_{fg}$, is called the "final extension" of P_{pr} following P_{vw}.

Lemma 4.8: Let P_{fg} be the final extension of P_{pr} following P_{vw}, with $t < v \leq n$ and $t < w \leq n$. Then $\varepsilon_{fg} = |a_{fg}|$.

Proof: Suppose $\varepsilon_{fg} > |a_{fg}|$ holds. Then we have either $\varepsilon_{fg} = R_{fg} \geq C_{fg}$ or $\varepsilon_{fg} = C_{fg} > R_{fg}$. In the first case $R_{fg} > 0$ implies by the definition of R_{fg} that $f \leq t < v$ holds. Consequently for some k the path P_{kg} is the forward extension of P_{fg} following P_{vw}. This contradicts that P_{fg} is the last path of the series. In the second case we have $g \leq t < w$ because of $C_{fg} > 0$ and the definition of C_{fg}. Then the path $P_{f\ell}$, the backward extension of P_{fg} following P_{vw}, contradicts as well that P_{fg} is the last path of the series. The contradictions imply that the case $\varepsilon_{fg} > |a_{fg}|$ cannot occur and so $\varepsilon_{fg} = |a_{fg}|$.

Note that the final extension P_{fg} is always the first path in the series such that ε_{fg} equals $|a_{fg}|$.

Lemma 4.9: If $|\psi_{ij}| \le \Delta\varepsilon_{ij}$ holds in the case $\varepsilon_{ij} = R_{ij} > |a_{ij}|$ then the term ψ_{kj} associated with the forward extension of P_{ij} following P_{vw}, satisfies $|\psi_{kj}| \le \Delta\varepsilon_{kj}$.

Proof: $\varepsilon_{ij} = R_{ij}$ implies $\varepsilon_{ij} \le \frac{\varepsilon_{kj}}{|a_{ki}|} \eta_i$ by the definition of R_{ij}. Hence the term ψ_{kj} satisfies:

$$|\psi_{kj}| = |a_{ki}a_{ii}^{-1}\psi_{ij}| \le |a_{ki}|\eta_i^{-1}\Delta\varepsilon_{ij} \le \Delta\varepsilon_{kj}$$

Lemma 4.10: If $|\psi_{ij}| \le \Delta\varepsilon_{ij}$ holds in the case $\varepsilon_{ij} = C_{ij} > |a_{ij}|$ then the term $\psi_{i\ell}$ associated with the backward extension of P_{ij} following P_{vw}, satisfies $|\psi_{i\ell}| \le \Delta\varepsilon_{i\ell}$.

The proof proceeds in the same way as the proof of lemma 4.9.

Theorem 4.11: Let $\psi_{ij} \not\equiv a_{ij}$ satisfy $|\psi_{ij}| \le \Delta\varepsilon_{ij}$ and let P_{fg} be the final extension of P_{ij} following P^1_{vw} with $t < v \le n$ and $t < w \le n$. Then $\psi^1_{vw} = \psi^1_{vff}\psi_{fg}\psi^1_{ggw}$ is dominated by $\psi^2_{vw} = \psi^1_{vff}a_{fg}\psi^1_{ggw}$.

Proof: The construction of the final extension and the lemmas 4.9 and 4.10 assure that $0 < |\psi_{fg}| \le \Delta\varepsilon_{fg}$ is satisfied. Lemma 4.8 yields $\varepsilon_{fg} = |a_{fg}|$ which completes the proof.

If $|\psi_{ij}| \le \Delta\varepsilon_{ij}$ assures that ψ^1_{vw} has a dominator then the theorem shows how a dominator ψ^2_{vw} can be determined. We say that ψ^1_{vw} is "assigned" to the dominator ψ^2_{vw}. Figure 4.4 illustrates the determination of a dominator. The series $P^1_{ij}, P^1_{kj}, P^1_{kg}, P^1_{fg}$ yields the final extension P^1_{fg} of P^1_{ij}. So for $|\psi^1_{ij}| \le \Delta\varepsilon_{ij}$ the term $\psi^2_{vw} = \psi^1_{vff}a_{fg}\psi^1_{ggw}$ is a dominator of $\psi^1_{vw} = \psi^1_{vff}\psi^1_{fg}\psi^1_{ggw}$. The series of paths $P^1_{ij}, P^1_{kj}, P^1_{kg}, P^1_{fg}$ can be associated with the series of dependable references $\varepsilon_{ij}, \varepsilon_{kj}, \varepsilon_{kg}, \varepsilon_{fg}$. We say that ψ^1_{ij} is "assigned" to ε_{ij}, ψ^1_{kj} to ε_{kj}, etc.

If $|\psi_{ij}| \le \Delta\varepsilon_{ij}$ holds then the term ψ^1_{vw} is assigned to the dominator ψ^2_{vw} which is determined without ambiguity by the final extension of P_{ij}. Alternatively we may find $|\psi_{pr}| \le \Delta\varepsilon_{pr}$ and determine the final extension of P_{pr}. This may imply that ψ^1_{vw} is assigned to a dominator ψ^3_{vw} not identical to ψ^2_{vw}. For example in figure 4.4 the final extension of P_{pr} is the path P^1_{pj} and $\psi^1_{vw} = \psi^1_{vpp}\psi^1_{pj}\psi^1_{jjw}$ is assigned to $\psi^3_{vw} = \psi^1_{vpp}a_{pj}\psi^1_{jjw}$.

If a pivotstep is skipped a deleted term is assigned to only one dominator. Consider again figure 4.4 and assume $|a_{qq}| \ge \theta_q$ holds.

In order to find the dominator to which ψ^1_{vw} is assigned we take the shortest path in P^1_{vw} containing the concerned pivot a_{qq}. The path is $P_{pr} = \{a_{pq}, a_{qq}, a_{qr}\}$. Because of $|a_{qq}| \geq \theta_q$ we have $|\psi_{pr}| = |a_{pq}a^{-1}_{qq}a_{qr}| \leq \Delta\varepsilon_{pr}$. Hence the final extension of P_{pr} yields the dominator.

In the case of lemma 4.2 we have a one-to-one correspondence between deleted terms and dominators. But if dependable references are exploited two or more terms may be assigned to the same dominator. See figure 4.4 : if the i^{th} pivot exceeds its threshold both ψ^1_{vw} and ψ^3_{vw} are deleted. If the terms associated with the final extension of $P_{kp} = \{a_{ki}, a_{ii}, a_{ip}\}$ in respectively P^1_{vw} and P^3_{vw} are assigned to the same dependable reference $\varepsilon_{fg} = |a_{fg}|$ then ψ^1_{vw} and ψ^3_{vw} are assigned to the same dominator ψ^2_{vw}.

In section 4.3 we have demonstrated that the deletion of terms is equivalent to perturbations of appropriate matrix coefficients. It is attractive to perturb only nonzero coefficients. This is possible if it is known to which dominator a term is assigned. Let in figure 4.4 ψ^1_{vw} be assigned to ψ^2_{vw} because ψ^1_{fg} is assigned to $\varepsilon_{fg} = |a_{fg}|$. The deletion of ψ^1_{vw} induces a perturbation of a_{fg}: $\delta a^1_{fg} = -\psi^1_{fg}$.

If ψ^3_{vw} is assigned to ψ^2_{vw} as well, a second perturbation on a_{fg} is required to account for the deletion of ψ^3_{vw}: $\delta a^3_{fg} = -\psi^3_{fg}$. Although both $|\delta a^1_{fg}| \leq \Delta|a_{fg}|$ and $|\delta a^3_{fg}| \leq \Delta|a_{fg}|$ are satisfied the inequality $|\delta a_{fg}| = |\delta a^1_{fg} + \delta a^3_{fg}| \leq \Delta|a_{fg}|$ need not hold. No upperbound on the perturbation δa_{fg} is obtained because the computation of the dependable references according to (4.11) did not account for the possibility that two terms could be assigned to one dominator. In the following a computation of dependable references is studied such that any perturbation δa_{fg} accounting for the deleted terms, satisfies $|\delta a_{fg}| \leq \Delta|a_{fg}|$.

The alternate computation of dependable references uses the number ν_{ij}. ν_{ij} is the number of all nonzero coefficients a_{ik} in the i^{th} row and a_{kj} in the j^{th} column with the restriction $j < k < \min[i, t+1]$ in the case $j \leq t$ and the restriction $k < \min[i, t+1]$ in the case $j > t$. Figure 4.6 illustrates the definition. If ν_{ij} is nonzero ε_{ij} is computed according to:

$$\varepsilon_{ij} = \nu^{-1}_{ij} \max[R_{ij}, C_{ij}, |a_{ij}|] \tag{4.12}$$

The entities ε_{ij} computed by (4.12) are dependable references because

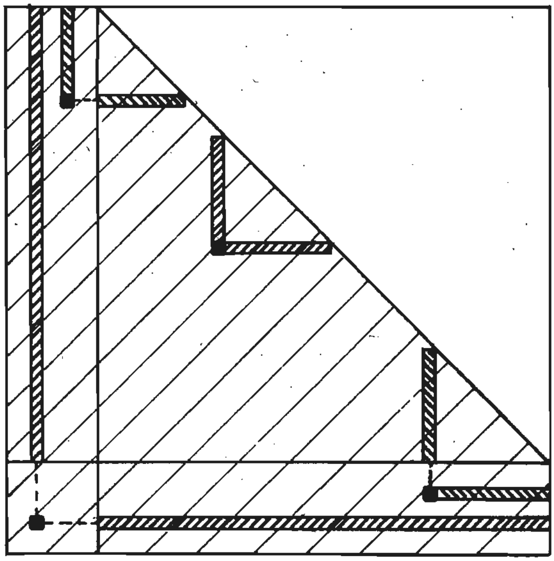

Fig. 4.6. Definition of ν_{ij}.
Black squares denote coefficients a_{ij} *for which the corresponding* ν_{ij} *is illustrated.* ν_{ij} *is the number of nonzeros in the shaded parts of the concerned row and column.*

$\nu_{ij} \geq 1$ if ν_{ij} is nonzero. If ν_{ij} is zero then no term $\psi_{ij} \not\equiv a_{ij}$ exists and consequently no dependable reference ε_{ij} is required.

The determination of the dominator ψ^2_{vw} to which a deleted term ψ^1_{vw} is assigned, proceeds in much the same way. Only the construction of the series of paths P^0, P^1, ..., P^ω yielding the final extension P^ω of P^0 requires a modification. Whether $P^{\sigma+1}$ is the forward or the backward extension of P^σ now depends on the value of $\varepsilon_{ij}\nu_{ij}$ and no longer on the value of ε_{ij}. Analogous to lemma 4.8 we have $\varepsilon_{fg}\nu_{fg} = |a_{fg}|$ for the final extension $P^\omega = P_{fg}$.

<u>Theorem 4.12</u>: Let the threshold θ_q be computed exploiting the dependable references obtained by equation (4.12). Let the q^{th} pivotstep be skipped because of $|a_{qq}| \geq \theta_q$. Let $\bar{\Psi}_{ij}$, for all i and j, denote the set of (deleted) terms assigned to ε_{ij}. Then the deleted terms satisfy

$$\sum_{\bar{\Psi}_{ij}} |\psi_{ij}| \leq \Delta\varepsilon_{ij}\nu_{ij} \qquad (4.13)$$

Proof: First we prove (4.13) for the case $i \le t$ and $j \le t$. The proof is by induction on $m = i-j$. For $m \le 0$ no term $\psi_{i,i-m}$ with $i-m \le t$ exists (BLT form). For $m = 1$ the term $\psi_{i,i-1} = a_{i,i-1}$ is not assigned to $\varepsilon_{i,i-1}$ and $\bar{\Psi}_{i,i-1}$ is empty.

For the induction step suppose (4.13) holds for $i-j < m$. Consider a term $\psi_{ij} \in \bar{\Psi}_{ij}$. Three cases are possible:

1) $\psi_{ij} = -a_{iq}a_{qq}^{-1}a_{qj}$
2) P_{ij} is the forward extension of some path P_{kj} with $k \neq q$: $P_{ij} = \{a_{ik}, a_{kk}\} \cup P_{kj}$
3) P_{ij} is the backward extension of some path $P_{i\ell}$ with $\ell \neq q$: $P_{ij} = P_{i\ell} \cup \{a_{\ell\ell}, a_{\ell j}\}$

Case 2) is depicted in figure 4.7. The case occurs if $\varepsilon_{kj}\nu_{kj} = R_{kj} > |a_{kj}|$ holds. Let $\bar{\bar{\Psi}}_{kj}$ denote the set $\{\psi_{kj} \mid \psi_{kj} \in \bar{\Psi}_{kj} \wedge -a_{ik}a_{kk}^{-1}\psi_{kj} \in \bar{\Psi}_{ij}\}$. As $\bar{\bar{\Psi}}_{kj}$ is a subset of $\bar{\Psi}_{kj}$, and $k - j < m$ holds because of $k < i$, the set $\bar{\bar{\Psi}}_{kj}$ satisfies the induction hypothesis:

$$\sum_{\bar{\bar{\Psi}}_{kj}} |\psi_{kj}| \le \Delta\varepsilon_{kj}\nu_{kj} = \Delta R_{kj}$$

With the definition of R_{kj} follows:

$$\sum_{\bar{\bar{\Psi}}_{kj}} |a_{ik}a_{kk}^{-1}\psi_{kj}| \le |a_{ik}|\ \eta_k^{-1}\Delta R_{kj} \le \Delta\varepsilon_{ij}$$

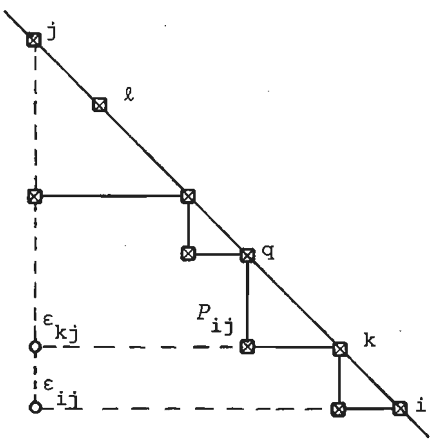

Fig.4.7. Illustration of theorem 4.12.

Equivalently the set

$\bar{\bar{\Psi}}_{i\ell} \triangleq \{\psi_{i\ell} \mid \psi_{i\ell} \in \bar{\Psi}_{i\ell} \wedge -\psi_{i\ell}a_{\ell\ell}^{-1}a_{\ell j} \in \bar{\Psi}_{ij}\}$ satisfies

$$\sum_{\bar{\bar{\Psi}}_{i\ell}} |\psi_{i\ell}a_{\ell\ell}^{-1}a_{\ell j}| \le \Delta c_{i\ell}\eta_{\ell}^{-1} |a_{\ell j}| \le \Delta\varepsilon_{ij}$$

Finally the possible term $-a_{iq}a_{qq}^{-1}a_{qj}$ satisfies $|a_{iq}a_{qq}^{-1}a_{qj}| \le \Delta\varepsilon_{ij}$. Each set $\bar{\bar{\Psi}}_{kj}$ is related to a nonzero coefficient $a_{ik} \not\equiv a_{iq}$ in the i^{th} row, each set $\bar{\bar{\Psi}}_{i\ell}$ to a nonzero coefficient $a_{\ell j} \not\equiv a_{qj}$ in the j^{th} column and the possible term $-a_{iq}a_{qq}^{-1}a_{qj}$ implies two nonzero coefficients, a_{iq} and a_{qj}. The number of these nonzero coefficients is at most equal to ν_{ij}.

Hence

$$\sum_{\bar{\Psi}_{ij}} |\psi_{ij}| = |a_{iq}a_{qq}^{-1}a_{qj}| + \sum_{k}\sum_{\bar{\bar{\Psi}}_{kj}} |a_{ik}a_{kk}^{-1}\psi_{kj}| +$$

$$+ \sum_{\ell}\sum_{\bar{\bar{\Psi}}_{i\ell}} |\psi_{i\ell}a_{\ell\ell}^{-1}a_{\ell j}| \le \Delta\varepsilon_{ij}\nu_{ij}$$

This completes the proof for the case $i \le t$ and $j \le t$. For the cases $i > t$ and $j \le t$, $i \le t$ and $j > t$, $i > t$ and $j > t$ the proof proceeds in the same way.

In the case $|a_{ij}| = \varepsilon_{ij}\nu_{ij}$ all terms in $\bar{\Psi}_{ij}$ are associated with final extensions P_{ij}. Theorem 4.12 implies that the perturbations δa_{ij} accounting for the terms deleted by the skipping of a pivotstep satisfy:

$$|\delta a_{ij}| = |\sum_{\bar{\Psi}_{ij}} \psi_{ij}| \le \sum_{\bar{\Psi}_{ij}} |\psi_{ij}| \le \Delta\varepsilon_{ij}\nu_{ij} = \Delta|a_{ij}| \qquad (4.14)$$

The inequality is similar to inequality (4.9) which is obtained if no dependable references are required.

4.6. Global error analysis

In this section the influence of pivotstep skipping on the solution vector is studied. First the skipping of one, say the q^{th}, pivotstep is considered. In section 4.3 (p. 42) it is shown that skipping of the q^{th} pivotstep implies that the Schur complement of a perturbed matrix $(A + \delta A)$ is computed. The coefficients of δA are:

$\delta a_{pr} = a_{pq}a_{qq}^{-1}a_{qr}$ for $p \ne q$ and $r \ne q$.

The consequence is that the solution vector z has a perturbation δz, such that $(A + \delta A)(z + \delta z) = Az$. If the coefficients of δA are small, then usually the elements of δz are small too. Often bounds for the norm of the error δz are used. If $\|\delta A\| \cdot \|A^{-1}\|$ is less than one, a well-known bound can be applied [4.3]

$$\frac{\|\delta z\|}{\|z\|} \leq \frac{\|A^{-1}\| \cdot \|\delta A\|}{1-\|A^{-1}\| \cdot \|\delta A\|} \tag{4.15}$$

A tighter bound is given in [4.4]. Let $|A|$ denote the matrix consisting of the absolute values of the coefficients of A. If $\| |A^{-1}| |\delta A| \|$ is less than one, then we have:

$$\frac{\|\delta z\|}{\|z\|} \leq \frac{\| |A^{-1}| \cdot |\delta A| |z| \|}{(1-\| |A^{-1}| \cdot |\delta A| \|) \|z\|} \tag{4.16}$$

These inequalities require bounds for $\|\delta A\|$ respectively $|\delta A|$.

First consider the case a_{pr} is nonzero if both a_{pq} and a_{qr} are nonzero. Then $|a_{qq}| \geq \theta_q$ implies (compare equation (4.9))

$$|\delta a_{pr}| = |a_{pq} a_{qq}^{-1} a_{qr}| \leq \Delta |a_{pr}|$$

So $|\delta A|$ is bounded by: $|\delta A| \leq \Delta |A|$, where "$\leq$" is applied coefficientwise. Clearly $\|\delta A\| \leq \Delta \|A\|$ is satisfied as well. If these bounds for $\|\delta A\|$ and $|\delta A|$ are used in (4.15) and (4.16) the inequalities (4.17) and (4.18) can be obtained:

$$\frac{\|\delta z\|}{\|z\|} \leq \frac{\Delta \|A^{-1}\| \cdot \|A\|}{1-\Delta \|A^{-1}\| \cdot \|A\|} \tag{4.17}$$

$$\frac{\|\delta z\|}{\|z\|} \leq \frac{\Delta \| |A^{-1}| |A| |z| \|}{(1-\Delta \| |A^{-1}| |A| \|) \|z\|} \leq \frac{\Delta \| |A^{-1}| |A| \|}{1-\Delta \| |A^{-1}| |A| \|} \tag{4.18}$$

The product $\|A^{-1}\| \cdot \|A\|$ is often called the condition number of the matrix A.

In the case that a_{pr} is zero although both a_{pq} and a_{qr} are nonzero, dependable references which are computed according to (4.12) may be applied. In the preceding section it is shown that the perturbations δa_{ij} accounting for the deleted terms satisfy (4.14):

$$|\delta a_{ij}| \leq \Delta |a_{ij}|$$

Clearly $|\delta A| \leq \Delta |A|$ and $\|\delta A\| \leq \Delta \|A\|$ follow and the inequalities (4.17) and (4.18) apply to this case as well.

If more than one pivotstep, say m pivotsteps, are skipped the same bounds (4.17) and (4.18) can be used if Δ is replaced by $(1+\Delta)^m -1$. For skipping of one pivotstep implies that the Schur complement of the matrix $A + \delta A^1$ is computed, while $|A + \delta A^1| \leq (1+\Delta)|A|$ holds. Skipping of a second pivotstep implies that the Schur complement of

$A + \delta A^1 + \delta A^2$ is obtained, with $|\delta A^2| \le \Delta |A + \delta A^1| \le \Delta(1+\Delta)|A|$. Hence $|A + \delta A^1 + \delta A^2| \le (1+\Delta)^2 |A|$ holds. Finally if m pivotsteps are skipped we have

$$|\delta A^m| \le \Delta |A + \delta A^1 + \ldots + \delta A^{m-1}| \le \Delta(1+\Delta)^{m-1}|A|$$

and the sum of all perturbations satisfy:

$$|\sum_{i=1}^{m} \delta A^i| \le \sum_{i=1}^{m} |\delta A^i| \le \sum_{i=1}^{m} \Delta(1+\Delta)^{i-1}|A| = \{(1+\Delta)^m - 1\}|A|$$

The factor $\{(1+\Delta)^m - 1\}$ is for small Δ almost equal to $m\Delta$.

The meaning of the bounds (4.17) and (4.18) is not primarily that the error δz can be estimated. To compute such an estimate requires usually relatively much time because the inverse of A occurs in these formulas. Genearlly A^{-1} is not explicitly available, e.g. it is not available if L\U-decomposition is applied. Also estimating the condition number without explicitly using A^{-1}, as is suggested in [4.5], may require too much time. The point is that (4.17) and (4.18) show that the error δz can be made arbitrarily small if Δ becomes small enough. Pivotstep skipping is in principle a good method to compute an approximate solution; only the value of Δ has to be chosen appropriately.

4.7. Pivotsubstep skipping

In the foregoing we supposed continually that pivotsteps are skipped as a whole. But pivotsubsteps can be skipped as well. If the conditions induced by cycles of length one passing through a pivot, say a_{qq}, and a special coefficient in the pivot column, say a_{pq}, are collected, a subset of conditions of the form (4.7) is obtained. If all conditions of this subset are satisfied all terms containing the product $a_{pq}a_{qq}^{-1}$ can be deleted. For Gauss elimination this implies that in the q^{th} pivotstep the pivotsubstep associated with a_{pq} can be skipped. Let θ_{pq} be the threshold of this pivotsubstep. From (4.7) and (4.8) we obtain:

$$\theta_{pq} = \Delta^{-1} |a_{pq}| \max_r[|a_{pr}^{-1}a_{qr}|] \le \theta_q$$

Apparently the threshold of a pivotsubstep can be smaller than the threshold of the pivotstep. Thus pivotsubsteps can be skipped sometimes if not the whole pivotstep can be skipped. Pivotsubsteps with approximately the same thresholds may be joined to be controlled by one threshold. Note that this partition of a pivotstep in smaller steps is a simple byproduct of the computation of the thresholds for the pivotsteps. However, the experience is that this refinement only incidentally pays.

Another approach to the control of pivotsubstep skipping gives better results. Consider the operation $u_{ij} \leftarrow u_{ij} - l_{ik}u_{kj}$ in the k^{th} pivotstep. Assume that the k^{th} pivotstep has to be executed ($a_{kk} < \theta_k$), while the i^{th} pivotstep can be skipped ($a_{ii} \geq \theta_i$). Then the execution of the above operation is useless because the result will never be used in the process to compute the Schur complement. Moreover the execution of the whole pivotsubstep associated with a_{ik} is superfluous. If before the execution of the pivotsubstep pivot a_{ii} is tested and $|a_{ii}| \geq \theta_i$ holds, the pivotsubstep can be skipped. In what follows the name "pivotsubstep skipping", abbreviated to PSSS, refers to the latter approach. The approach which skips only pivotsteps as a whole, will be called "pivotstep skipping", abbreviated to PSS. Note that PSSS does not introduce extra errors compared with PSS.

[4.1] D.J. Rose, R.E. Tarjan, G.S. Lueker, "Algorithmic aspects of vertex elimination on graphs", SIAM Journal on Computing, Vol. 5, pp. 266-283 (1976).

[4.2] R.W. Cottle, "Manifestations of the Schur complement", Linear algebra and its applications, Vol. 8, pp. 189-211 (1974).

[4.3] J.H. Wilkinson, "Rounding errors in algebraic processes", Prentice-Hall, Englewood Cliffs, N.J., 1963.

[4.4] R.D. Skeel, "Scaling for numerical stability in Gaussian elimination", Journal of the ACM, Vol. 26, pp. 494-526 (1979).

[4.5] A.K. Cline, C.B. Moler, G.W. Stewart, J.H. Wilkinson, "An estimate for the condition number of a matrix", SIAM Journal on Numerical Analysis, Vol. 16, pp. 368-375 (1979).

5. IMPLEMENTATION OF PIVOTSTEP SKIPPING

5.1. Comparison of different methods

Mainly there are three methods of computing the solution of Az = r They are called compiled code, interpretable code and looping indexed approach respectively. In [5.1] and [5.2] these approaches are discussed. Moreover [5.1] gives many references to these methods.

In the compiled code (or machine code) approach the structure of the initial matrix A and its L and U factor are analysed. Then a loop-free list of (machine) instructions is generated concerning all nontrivial operations in the computation of the L\U-decomposition. Gustavson et al.[5.3] report a program GNSO which generates a linear list of FORTRAN statements. The main advantage of this approach is that it is extremely fast because it does no testing or branching and every variable is addressed directly. Secondly with this approach it is very easy to handle different variability types [5.4] or overwrite parts of data which are no longer needed. The main disadvantage is that the compiled code can be very long.

The interpretable (or interpretative) code approach generates instead of a list of instructions, a list of operation codes with addresses of the corresponding operands. In the execution stage an interpreter executes the operations coded in this list and uses the addresses to retrieve the operands and to store the result. This approach is slower than the compiled code (according to [5.5] it can be 5 times slower) but the generated code is shorter.

The looping indexed approach (or derived indexing, or row ordered elimination) generates along with the numerical values of the matrix coefficients some pointer arrays which represent the structure of the matrix. These pointers are used during the numeric factorization to avoid operations with zeros. This approach is the slowest of the three because of the indirect addressing involved. But it requires much less storage than the other two.

For the implementation of pivotstep skipping (PSS) or pivotsubstep skipping (PSSS) it is desired to have a more flexible datastructure than is used by the looping indexed approach. The linked list datastructure which will be described in this chapter offers this flexibility. Although some more pointer arrays are used by the linked list approach, it is closely related to the looping indexed approach. Therefore the latter approach will not be considered here.

It can be shown that Crout elimination is less time consuming than Gauss elimination because the number of data store operations is different. Yet we will only deal with the implementation of Gauss elimination. This is because the implementation of Crout elimination is less simple, mainly because Crout elimination uses two kinds of elimination steps to compute the Schur complement. Whereas the coefficients of U_{12} (corresponding to A_{12}) can be computed along the usual Crout scheme, the updates added to the coefficients of the Schur complement $\tilde{A}_{22}$ have to be assembled along slightly different lines. However no fundamentally new difficulties arise in the implementation of PSS or PSSS in Crout elimination. Qualitatively the execution time of these implementations depends in the same way on the number of skipped pivot steps for both Gauss and Crout elimination. Therefore it suffices to deal with Gauss elimination.

Appendix B contains implementations of Gauss elimination for each of the three approaches. A comparison is made of the approaches, both for time costs and storage requirements. The results of the comparison fit in with the description of the approaches, however the differences appear not to be so pronounced. In the following no attention will be paid to the interpretable code approach. The adaptation of this approach to account for PSS or PSSS is the same as for the compiled code approach. The obtained results lie between both other approaches and depend in the same way on the number of skipped pivotsteps.

Thus in this chapter implementations of PSS and PSSS are given using the compiled code and the linked list approach. To adapt these approaches such that pivot(sub)step skipping can be applied some extra work need be done. E.g. in the compiled code approach test- and branch-instructions are introduced. These instructions have the intention to avoid the execution of other instructions. The overall result depends on the relative costs (in time) of the instructions. Therefore the implementations are analysed in some detail. It is not the aim to give the best possible implementation or to guarantee some execution time but to show that such an analysis is possible and useful.

The implementations are in MACRO-11. The execution times of the instructions are given in units of 170 nsec. The time specifications are obtained from [5.6]. The storage requirements of the instructions are given in bytes. In the implementations Rϕ - R3 denote registers of the CPU, and Fϕ - F2 denote registers of the floating point processor.

5.2. Definitions of parameters

In this paragraph the parameters are defined which characterize a given BLT matrix and which will be used in this chapter.

n, t and b are already defined in chapter 1.

ρ denotes the mean value of the number of coefficients in a row. κ denotes the same for a column. The indices of ρ and κ indicate to which part of the matrix A they are related. E.g. ρ_{12} is the mean value of the number of coefficients in the rows of A_{12}. κ_{*1} is the mean value of the number of coefficients in the columns of $[A_{11}^T A_{21}^T]^T$, $\kappa_{*1} = \kappa_{11} + \kappa_{21}$. (The pivots are not included in ρ_{11} and κ_{11}).

A pivot is called active if its pivotstep has to be executed, otherwise it is called passive.

π is the pivot activity i.e. the number of pivotsteps to be executed relative to the total number of pivotsteps.

μ is the pivot variability i.e. the number of pivots passing their thresholds going from one Newton iteration to the next, relative to the total number of pivots. $\mu = \mu_a + \mu_p$ where μ_a corresponds to the pivots becoming active and μ_p corresponds to the pivots becoming passive.

5.3. Compiled code approach

In the PSSS method the same pivot may be tested several times because it may be also tested during pivotsteps corresponding to other pivots. Then it is better to execute the relative expensive floating point tests in a preprocessing phase and store the results of these tests for later reference. A typical part of the instructions of the preprocessing phase, called PREPRO, is given on page 62.

PREPRO

	instruction		*time*	*storage*	*comment*
1	LDF PIVOT1,Fϕ		13	4	load Fϕ with pivot value;
2	ABSF Fϕ	t	4	2	take absolute value of Fϕ;
3	CMPF #thrshd1,Fϕ		9	4	compare threshold with pivot;
4	CFCC		12	2	transfer result of comparison to CPU;
5	SXT PIVTST1		7(?)	4	store result of comparison into PIVTST1;

COMSUBS

	instruction				*time*	*storage*	*comment*
	PIVOT1:						
1	TST PIVTST1			t	9	4	test whether pivotstep 1 can be skipped;
2	BGE EXECUTE				5	4	go eventually to line 4 to execute pivotstep;
3	JMP PIVOT2			$t(1-\pi)$	8	4	jump to pivotstep 2;
	EXECUTE:						
4	LDF PIVOT1,Fϕ				13	4	load Fϕ with pivot value;
5	LDF COEF1,F1				13	4	load F1 with border coefficient;
6	DIVF Fϕ,F1				38	2	divide border coefficient by pivot;
7	STF F1,COEF1				23	4	store quotient;
8	NEGF F1			$t\pi$	4	2	negate quotient;
9	TST PIVTSTk	κ_{11}			9	4	test whether pivotstep k need be executed;
10	BGE NEXT		ρ_{12}		5	4	skip eventually pivotsubstep and go to line 15;
11	LDF F1,F2				4	2	copy (-quotient) into F2;
12	MULF COEF2,F2	$\kappa_{11}\pi$			13	4	multiply F2 by coefficient from pivot column;
13	ADDF COEF3,F2	+			13	4	add value of coefficient in border to product;
14	STF F2,COEF3	κ_{21}			23	4	store sum;
	NEXT:						

The test of one pivot is described. The total code for the preprocessing phase is obtained by repeating these instructions t times, once for each pivot. This is indicated by the arrow with label t.

A typical part of the instructions for the actual L\U-decomposition is given in the list COMSUBS. Lines 1-3 contain the test concerning the pivotstep in lines 4-14. In lines 5-7 one border coefficient is divided by the pivot (implementing equation (4.1)). Lines 9 and 10 contain the test for the pivotsubstep in lines 11-14. In lines 11-14 one coefficient in the border is updated (according to equation (4.2) or (4.3)).

To obtain the total code, parts of the code in list COMSUBS need be repeated. Lines 9-14 need be repeated for each nonzero coefficient in the column of A_{11} (κ_{11} times). Lines 11-14 need be repeated κ_{21} times for each nonzero in the column of A_{21} (no test is applied because no row of A_{21} corresponds to a skipped pivotstep). The code thus obtained together with lines 5-8 is repeated ρ_{12} times for each nonzero in the row of A_{12}. Note the nested structure of the repetition.

Going on in this way with repeating t times the parts indicated by t, once for each pivot, the total code is obtained. The factors π and $1-\pi$ are unimportant as far as the repetition of parts of the codes is concerned. They indicate how many of such repeated parts actually are executed when PSSS is applied.

Using the time specifications of the preprocessing phase and of COMSUBS the total execution time can be computed. The execution time of the preprocessing phase is

$$T_{PREPRO} = 45t$$

The execution time of the actual code is:

$$T_{COMSUBS} = 14t + 8t(1-\pi) + t\pi\{13+\rho_{12}(78+14\kappa_{11}+53[\kappa_{11}\pi+\kappa_{21}])\}$$

The total time used by PSSS is the sum of both:

$$T_{CCS} = T_{PREPRO} + T_{COMSUBS}$$

If COMSUBS is simplified an implementation of PSS, COMSKIP, can be obtained. Because the test of a pivot is used only once these tests are included in COMSKIP. For the execution time we obtain:

$$T_{CCP} = T_{COMSKIP} = 47t + 8t(1-\pi) + \pi t \rho_{12}(78+53\kappa_{*1})$$

To evaluate these execution times they are compared with the execution time of an implementation without tests and skipping no pivot(sub)steps. In appendix B this time, T_{CC}, is computed. Figure 5.1 shows plots of T_{CCS}/T_{CC} and T_{CCP}/T_{CC} as a function of π for an ECL OR/NOR gate and the operational amplifier µA 709.

COMSKIP

	instruction					*time*	*storage*
	PIVOT1						
1	LDF PIVOT1,Fϕ					13	4
2	LDF Fϕ,F1					4	2
3	ABSF F1				t	4	2
4	CMPF #thrshd,F1					9	4
5	CFCC					12	2
6	BGT EXECUTE					5	4
7	JMP PIVOT2				$t(1-\pi)$	8	4
	EXECUTE:						
8	LDF COEF1,F1					13	4
9	DIVF Fϕ,F1					38	2
10	STF F1,COEF1					23	4
11	NEGF F1				$t\pi$	4	2
12	LDF F1,F2			ρ_{12}		4	2
13	MULF COEF2,F2		κ_{*1}			13	4
14	ADDF COEF3,F2					13	4
15	STF F2,COEF3					23	4

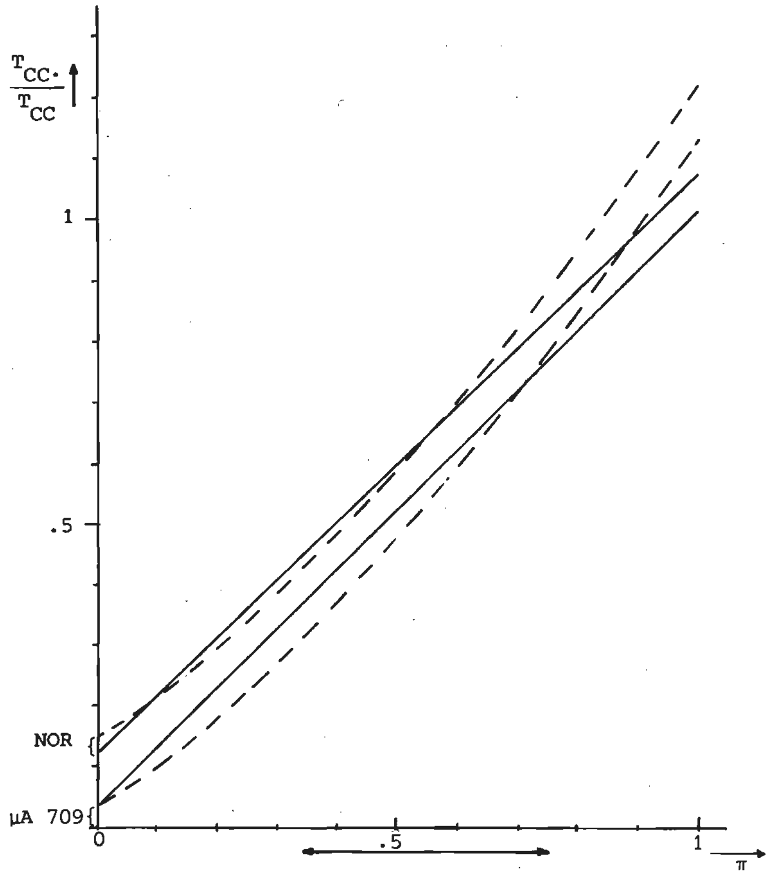

Fig. 5.1. Execution times of pivot(sub)step skipping applying the compiled code approach. The solid lines correspond to T_{CCP} (PSS), the dashed lines to T_{CCS} (PSSS).

5.4. Linked list approach

The drawback of the compiled code approach is the size of the generated code which varies from several kilobytes for a digital gate to decades of kilobytes for larger circuits (operational amplifier, flip-flop etc.). If this is unacceptable, the linked list approach may be attractive. Then the quantity of stored data apart from the numerical values, is determined by the pointer arrays and the

the little program to execute the L\U-decomposition.

Pivot(sub)step skipping can be implemented in the linked list approach in two ways. The first way is roughly the same as in the compiled code approach. During the actual L\U-decomposition the pivots are tested and pivot(sub)steps are skipped according to the outcome of the test. In the case PSSS is applied the pivots can be tested in a preprocessing phase and the results of the tests can be stored for use in the actual L\U-decomposition.

The second way is to modify the linked lists during a preprocessing phase (the "adjustment" phase) such that during the execution of the L\U-decomposition the insignificant pivot(sub)steps are not executed. Then it is important how often the lists have to be adjusted. If in successive Newton iterations the datastructure has to be modified considerably, the adjustment phase will constitute a significant load. Therefore the pivot variability is important. From experiments it appears that the mean value of the pivot variability during an iteration process is very low. By the transition from the estimate to the first iteration μ is about .5 and decreases in a few steps to the low value of about .1. As it will appear this means that the adjustment phase does not waste the time to be gained by applying PSS or PSSS in the execution phase.

With the implementation given in section 5.3 in mind, it is straightforward to adapt the program LINKLIST for implementing PSS or PSSS in the first way. This program executes the actual L\U-decomposition and will be described at the end of this section. In this section only the second way of implementing PSS and PSSS is discussed.

First the datastructure for the case that no pivot(sub)step skipping is applied, will be given. Then it is discussed which pointer arrays have to be modified to account for PSS or PSSS, and which arrays have to be added to do the adjustments conveniently. It is supposed that a fast execution of the L\U-decomposition is the main objective. Although no storage should be wasted, an extension of the datastructure will be allowed here if execution time can be saved substantially. For instance searching should be avoided unless this requires a disproportionate amount of storage. Further it is assumed that the datastructure includes the fill-ins, and that a copy of the border of A is saved so that the coefficients of L and U can be stored in the same locations as the coefficients of A.

During the execution of a pivotstep coefficients in the pivot row ("border coefficients") and column ("pivotsubstep coefficients") are used. The arrays RWNEXT and CLNEXT make these coefficients easily accessible (see table 5.1 for the definitions, figure 5.2 illustrates the datastructure).

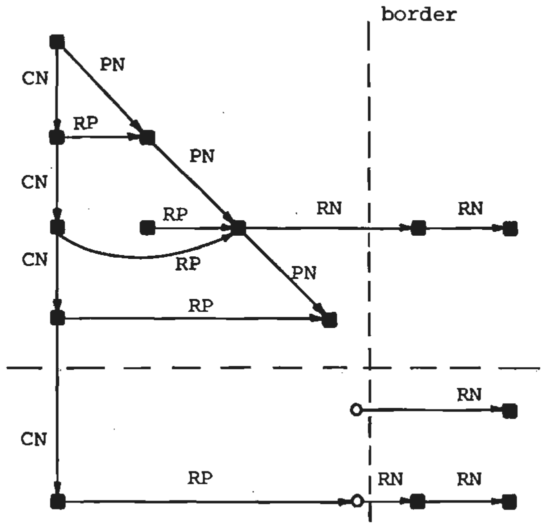

Fig. 5.2. The datastructure for PSS.
A part of a BLT matrix and the pointers connecting the nonzeros are shown. A ■ indicates a nonzero, a o denotes a fictitious pivot. CN = CLNEXT, PN = PVNEXT, RN = RWNEXT, RP = ROWPIV.

TABLE 5.1.

CLNEXT:	points from a coefficient to the next coefficient below it in the same column
COLUMN:	contains the column number of the coefficient
PVNEXT:	points from a pivot to the pivot in the next row
RWNEXT:	points from a coefficient to the next coefficient right of it in the same row
ROWPIV:	points from a coefficient to the pivot in the same row

Now the division of the border coefficients by the pivot is straightforward. During the execution of a pivotsubstep border coefficients in the same row as the pivotstep coefficient have to be accessed. Therefore the pointer ROWPIV is convenient. It points from the pivotsubstep coefficient to the pivot in the same row. From this pivot the border coefficients can be accessed by RWNEXT. Since the last border rows, the rows of A_{22}, have not pivots assigned to them, a "fictitious pivot" is introduced for each row of A_{22}. Such a pivot does not correspond to an actual matrix coefficient but is only starting-point of RWNEXT which connects the coefficients in the concerning row of A_{22}.

If the product of a pivotsubstep coefficient and a border coefficient in the pivot row has been formed, it has to be subtracted from another border coefficient in the row associated with the pivotsubstep coefficient. This row may contain more border coefficients than the pivot row (the reverse is not possible because fill-ins are already included in the datastructure). Thus in the row of the pivotsubstep coefficient sometimes the correct border coefficient has to be searched for. The coefficient is in the same column as the border coefficient in the pivot row. This asks for an array COLUMN which contains the column number of a coefficient. The array PVNEXT pointing from one pivot to the next, completes the datastructure. The termination of lists is indicated by a value less than or equal to zero. For instance, if x is the last coefficient in a row then RWNEXT(x)$\leq$0. The program LINKLIST can be applied to this datastructure if the names of the arrays CLNEXT and PVNEXT are changed into CNXTAC and PNXTAC respectively.

For the implementation of pivotstep skipping only an array connecting the pivots need be modified. Because in the adjustment all pivots need be accessed for testing a new "variable" array is introduced, PNXTAC, which points from a pivot to the next active pivot (see table 5.2 and figure 5.3). To prevent that PNXTAC eventually becomes empty a fictitious "zeroth" pivot is assumed which is always active. The program ADSKIP forms PNXTAC such that PSS will be executed. ADSKIP tests the pivots in the order recorded in PVNEXT. In register R6 the address of the last preceding active pivot is saved. This address is used to adjust PNXTAC of that pivot if a new active pivot is detected.

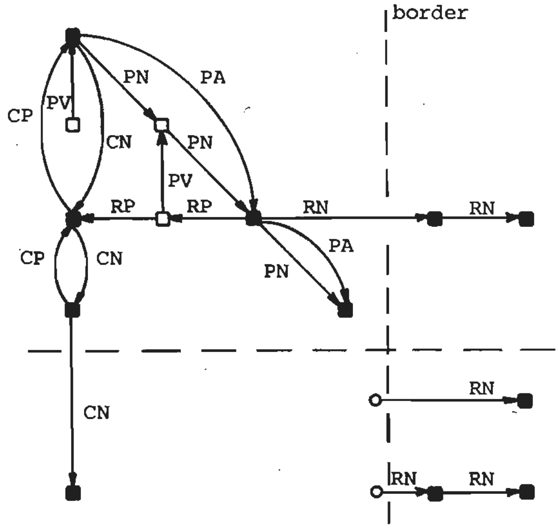

Fig. 5.3. *The datastructure for PSSS.*
A straight line denotes a fixed pointer, a curved line denotes a variable pointer (eventually adjusted). A ■ indicates a nonzero in a row and a column of an active pivot. A □ indicates a nonzero in a row or a column of a passive pivot. A o denotes a fictitious pivot. CN = CNXTAC, CP = CPRCAC, PA = PNXTAC, PN = PVNEXT, PV = COLPIV, RN = RWNEXT. (Not all pointers COLPIV are shown).

TABLE 5.2.

(Contains only arrays not yet defined in table 5.1.)

CNXTAC:	points from a coefficient associated with an active pivotsubstep to another coefficient in the same column being as well associated with an active pivotsubstep
CPRCAC:	same definition as CNXTAC but points in the reversed direction
COLPIV:	points from a coefficient to the pivot in the same column
PIVACT:	contains activity status of pivot (0 = active, -1 = passive)
PNXTAC:	points from an active pivot to the next active pivot
RWPREC:	points from a coefficient to the first nonzero coefficient left of it in the same row

ADSKIP

```
                                   time storage comment
NEXT PIVOT:
MOV   PVNEXT(Rϕ),Rϕ              8      4
BLE   END                        5      2

LDF   COEF(Rϕ),Fϕ               13      4
ABSF  Fϕ                         4      2
CMPF  THRESH(Rϕ),Fϕ          t  13      4
CFCC                            12      2
BLT   NEXT PIVOT                 5      2    if pivot is passive go on with
                                              next one;

MOV   Rϕ,PNXTAC(R6)              7      4    active pivot: connnect it to
                                              preceding active pivot;
MOV   Rϕ,R6                π     2      2    put address of active pivot
                                              into R6;
BR    NEXT PIVOT                 5      2    go on with next pivot;

END:
MOV   #ϕ,PNXTAC(R6)             11      4    zero value in PNXTAC of last
                                              active pivot indicates
                                              termination of list;
```

The execution time of ADSKIP is

$$T_{AP} = t(60+14\pi) + 11$$

The implementation of pivotsubstep skipping requires the adjustment of CLNEXT as well. Because from now on this array contains only coefficients associated to active pivotsubsteps the name CNXTAC will be used instead of CLNEXT. If a pivot becomes passive the pivotsubsteps corresponding to coefficients in the pivot row can be skipped. These coefficients can be accessed by RWPREC (see table 5.2 and figure 5.3). Such a coefficient has to be thrown out of the list CNXTAC so that the coefficient pointing to it has to be accessed. Therefore array CPRCAC is introduced, which together with CNXTAC forms a double linked list.

If a pivot becomes active then RWPREC is used to access the coefficients in the pivotrow which have to be inserted in the arrays CNXTAC and CPRCAC. Because pivotsubsteps can be executed in any order within their pivotstep, a pivotsubstep coefficient can be inserted at any arbitrary position in the list. Here the choice is made to insert such a coefficient just after the pivot in its column, because the pivot is always present in the list. To access the pivot easily the pointer COLPIV is used which points from a coefficient to the pivot in the same column. This datastructure is shown in figure 5.3; the definitions of the arrays are listed in tables 5.1 and 5.2.

The program ADSUBS executes the adjustment of this datastructure. Like in ADSKIP the pivots are tested in the order recorded in PVNEXT and the address of the last preceding active pivot is saved in register R6. Unlike ADSKIP the program ADSUBS executes only real modifications of the datastructure. Therefore an array PIVACT is used which contains the activity status of the pivots. Only if the activity status of a pivot is changed the datastructure is adjusted.

In lines 1-7 of ADSUBS a pivot value is loaded and tested. In lines 8-14 the new pivot activity is compared with the old one. Lines 15-28 are executed if a pivot has become active. In lines 15-19 the pivot itself is inserted into the list PNXTAC. In lines 20-27 a coefficient of the pivot row is inserted into CNXTAC and CPRCAC of the right column. Lines 29-38 are executed of a pivot has become passive. In lines 29-31 the pivot itself is omitted. In lines 32-37 a coefficient of the pivot row is omitted.

The parameters ρ_{11}, μ_a, μ_p, t etc. indicate how often the corresponding parts are executed if the datastructure is adjusted once. The execution time of ADSUBS is:

$$T_{AS} = t\{73 + 5(1-\mu_p) + 7(1-\mu) + 7(1-\mu)\pi + \mu_a(42+57\rho_{11}) + \mu_p(33+43\rho_{11})\} =$$

$$= t\{85 + 7(1-\mu)\pi + \mu_a(35+57\rho_{11}) + \mu_p(21+43\rho_{11})\}$$

Over a large number of iteration steps the mean value of μ_a must be almost equal to the mean value of μ_p. Supposing $\mu_a = \mu_p = \frac{1}{2}\mu$ the formula for T_{AS} simplifies to

$$T_{AS} = t\{85 + 7(1-\mu)\pi + \mu(28+50\rho_{11}\} \qquad (5.1)$$

For $\pi = \frac{4}{7}$ we get

$$T_{AS} = 89t\left(1 + \frac{24+50\rho_{11}}{89}\mu\right)$$

```
   ADSUBS
                                                        time storage comment
   NEXT PIVOT:
 1 MOV  PVNEXT(Rϕ),Rϕ                                    8     4   put address of next pivot into Rϕ;
 2 BLE  END                                              5     2   if address not positive then jump to END;

 3 LDF  COEF(Rϕ),Fϕ                                     13     4   put pivot value into Fϕ;
 4 ABSF Fϕ                                               4     2   take absolute value of Fϕ;
 5 CMPF THRESH(Rϕ),Fϕ                                   13     4   compare threshold with pivot;
 6 CFCC                                                 12     2   transfer result of comparison to CPU;
 7 SXT  R1                                               2(?)  2   store result of comparison into R1;

 8 CMP  R1,PIVACT(Rϕ)                                   11     4   compare R1 with old pivot activity;
 9 BLT  DELETE PIVOT                                     5     2   R1< : delete pivotstep;
10 BGT  INSERT PIVOT        1-μp                         5     2   R1> : insert pivotstep;
11 TST  R1                                               2     2   pivot activity is same as before: test R1;
12 BLT  NEXT PIVOT          1-μ                          5     2   inactive pivot: jump to line 1;
13 MOV  Rϕ,R6                                            2     2   active pivot: update R6;
14 BR   NEXT PIVOT          (1-μ)π                       5     2   jump to line 1;
```

```
   INSERT PIVOT:
15 MOV  R1,PIVACT(Rϕ)              7    4   update pivot activity;
16 MOV  PNXTAC(R6),PNXTAC(Rϕ)     13    6   connect pivot 'Rϕ' with successor of 'R6';
17 MOV  Rϕ,PNXTAC(R6)              7    4   connect 'R6' with 'Rϕ';
18 MOV  Rϕ,R6                      2    2   update R6;
19 MOV  RWPREC(Rϕ),R1              8    4   put address of coefficient of pivot row into R1;

   NEXT INSERTION;
20 MOV  COLPIV(R1),R2              8    4   put address of pivot in column coefficient into R2;
21 MOV  CNXTAC(R2),R3              8    4   put address of successor of this pivot into R3;
22 MOV  R1,CNXTAC(R2)              7    4   connect pivot with coefficient;
23 MOV  R1,CPRCAC(R3)              7    4   connect successor with coefficient;
24 MOV  R2,CPRCAC(R1)              7    4   connect coefficient with pivot;
25 MOV  R3,CNXTAC(R1)              7    4   connect coefficient with successor;
26 MOV  RWPREC(R1),R1              8    4   put address of next coefficient of row into R1;
27 BGT  NEXT INSERTION             5    2   jump to line 20 if address is positive;

28 BR   NEXT PIVOT                 5    2   jump to line 1;

   DELETE PIVOT:
29 MOV  R1,PIVACT(Rϕ)              7    4   update pivot activity;
30 MOV  PNXTAC(Rϕ),PNXTAC(R6)     13    6   connect 'R6' with successor of 'Rϕ';
31 MOV  RWPREC(Rϕ),R1              8    4   put address of coefficient of pivot row into R1;

   NEXT DELETION:
32 MOV  CPRCAC(R1),R2              8    4   put address of predecessor into R2;
33 MOV  CNXTAC(R1),R3              8    4   put address of successor into R3;
34 MOV  R3,CNXTAC(R2)              7    4   connect predecessor with successor;
35 MOV  R2,CPRCAC(R3)              7    4   connect successor with predecessor;
36 MOV  RWPREC(R1),R1              8    4   put address of next coefficient of row into R1;
37 BGT  NEXT DELETION              5    2   jump to line 32 if address is positive;

38 BR   NEXT PIVOT                 5    2   jump to line 1;
                                      ---
                                      126
   END:
```

Labels on arrows beside the listing: t (lines 15–38), μ_a (lines 15–28), ρ_{11} (lines 20–27), μ_p (lines 29–38), ρ_{11} (lines 32–37)

```
LINKLIST
                                          time storage comment
   NEXT PIVOT:
 1 MOV  PNXTAC(Rϕ),Rϕ                        8    4   put address of next pivot into Rϕ;
 2 BLE  END                                  5    2   if address not positive then jump to END;
 3 LDF  COEF(Rϕ),Fϕ                         13    4   put pivot value into Fϕ;
 4 MOV  RWNEXT(Rϕ),R1                        8    4   put address of first border coefficient into R1;
 5 MOV  R1,R2                                2    2   copy address into R2;
   NEXT DIVISION:
 6 LDF  COEF(R2),F1                         13    4   put value of border coefficient into F1;
 7 DIVF Fϕ,F1                               38    2   divide border coefficient by pivot;
 8 STF  F1,COEF(R2)                         23    4   store quotient;
 9 MOV  RWNEXT(R2),R2                        8    4   put address of next border coefficient into R1;
10 BGT  NEXT DIVISION                        5    2   if address positive jump to line 6;
11 MOV  CNXTAC(Rϕ),R2                        8    4   put address of coefficient in pivot column into R2;
   NEXT ROW:
12 LDF  COEF(R2),Fϕ                         13    4   put value of coefficient in pivot column into Fϕ;
13 NEGF Fϕ                                   4    2   take negative value;
14 MOV  ROWPIV(R2),R3                        8    4   put address of pivot in coefficient row into R3;
15 MOV  R1,R4                                2    2   copy border coefficient address into R4;
   NEXT BORDERCOEF:
16 MOV  RWNEXT(R3),R3                        8    4   put address of border coefficient in coefficient row
                                                                                                  into R3;
17 CMP  COLUMN(R3),COLUMN(R4)               16    6   compare column numbers of both border coefficients;
18 BNE  NEXT BORDERCOEF                      5    2   if not in same column jump to line 16;
19 LDF  Fϕ,F1                                4    2   copy value of coefficient into F1;
20 MULF COEF(R4),F1                         13    4   multiply F1 by border coefficient in pivot row;
21 ADDF COEF(R3),F1                         13    4   add border coefficient in coefficient row;
22 STF  F1,COEF(R3)                         23    4   store result;
23 MOV  RWNEXT(R4),R4                        8    4   put address of next border coefficient in pivot row
                                                                                                  into R4;
24 BGT  NEXT BORDERCOEF                      5    2   if address positive jump to line 16;
25 MOV  CNXTAC(R2),R2                        8    4   put address of next coefficient in pivot column into R2;
26 BGT  NEXT ROW                             5    2   if address positive jump to line 12;
27 BR   NEXT PIVOT                           5    2   jump to line 1;
   END:
```

The factor $C = \frac{24+50\rho_{11}}{89}$ determining the influence of μ is listed for a few circuits in table 5.3. (Note that $\rho_{11} \equiv \kappa_{11}$). The table contains also the execution time T_{AS} for $\mu = 0.1$ (the mean value of μ) and T_{AS} in percents of the execution time T_{LL} of LINKLIST for $\pi = 1$. Finally the execution times of ADSKIP, T_{AP} and T_{AP}/T_{LL} (both for $\pi = 1$) are given. From this table we see that the preprocessing phase uses only a small amount of time compared with the time to compute the L\U-decomposition (compare with table B.3 of the appendix).

The program LINKLIST executes the actual L\U-decomposition using the datastructure, eventually adjusted by ADSKIP or ADSUBS. Parameters ϕ, ρ_{12}, $\tilde{\kappa}_{*1}$ and $\tilde{t}$ indicate how often the corresponding parts are executed. $\tilde{t} = \pi t$ for both PSS and PSSS. $\tilde{\kappa}_{*1} = \kappa_{*1}$ for PSS and $\tilde{\kappa}_{1} = \pi\kappa_{11} + \kappa_{21}$ for PSSS.

In lines 1-5 the pivot value is loaded and some initializations are executed. The coefficients in the border part of the pivot row are divided by the pivot in lines 6-10. In line 12 a coefficient in the pivot column is loaded. The row determined by this coefficient is referred to as "coefficient row". In lines 16-18 a border coefficient in the coefficient row, occurring in the same column as the border coefficient in the pivot row is determined. This search and so the number of times ϕ these lines are executed depend on the sparsity structure. In lines 19 to 22 the new value of the border coefficient in the coefficient row is computed. Lines 23-24 and 25-26 respectively cause that the whole border row and the whole pivot column are passed through.

TABLE 5.3.

	C	T_{AS}	T_{AS}/T_{LL}	T_{AP}	T_{AP}/T_{LL}
TTL NAND	1.48	1330	5.6	973	4.1
ECL NOR	1.25	1200	10.0	899	7.5
ECL Flip-flop	2.39	3310	2.5	2230	1.7
μA 709	2.90	3330	2.6	2160	1.7
μA 741	2.50	3890	3.0	2600	2.0

The execution time of LINKLIST is:

$$T_{LL} = \pi t\{49 + 87\rho_{12} + \tilde{\kappa}_{*1}(40+\rho_{12}[66+29\phi])\} = T_{LL}(\tilde{\kappa}_{*1})$$

For PSSS we get

$$T_{LLS} = T_{AS} + T_{LL}(\pi\kappa_{11}+\kappa_{12})$$

and for PSS

$$T_{LLP} = T_{AP} + T_{LL}(\kappa_{*1})$$

These execution times are compared with the reference time $T_{LL}(\pi=1)$. Using equation (5.1) for T_{AS} and setting $\mu = 0.1$ the values $T_{LLS}/T_{LL}(\pi=1)$ and $T_{LLP}/T_{LL}(\pi=1)$ are plotted in figure 5.4 for an ECL OR/NOR gate and the operational amplifier µA 709. The plots of the

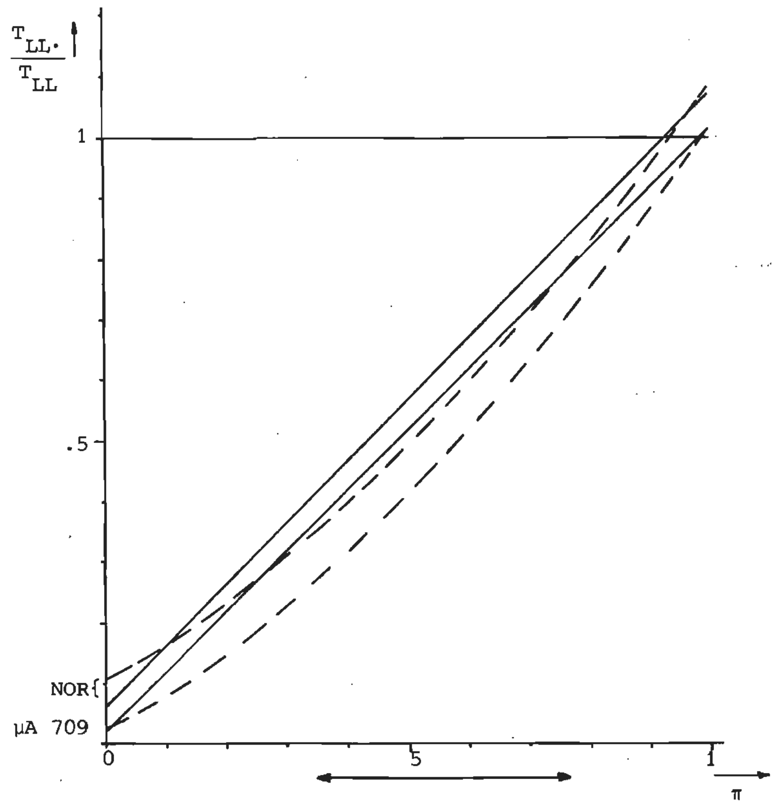

Fig. 5.4. Execution times of pivot(sub)step skipping applying the linked list approach. The solid lines correspond to T_{LLP} (PSS), the dashed lines to T_{LLS} (PSSS).

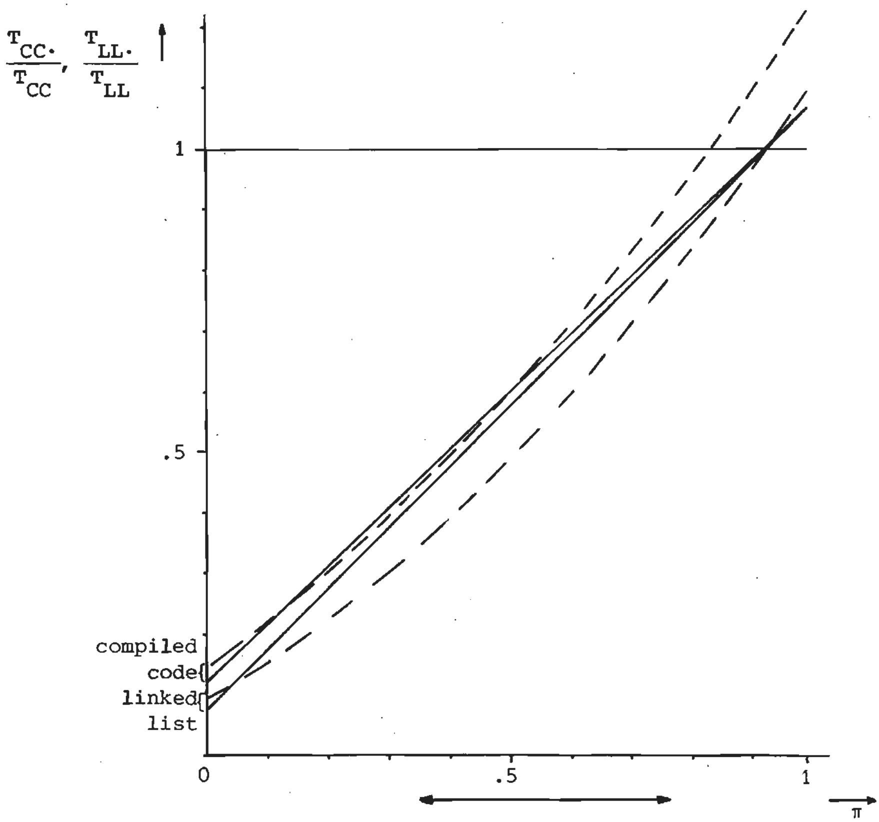

Fig.5.5. Comparison of PSS and PSSS in the complied code approach and in the linked list approach for an ECL OR/NOR gate.

linked list approach show much resemblance to those of the compiled code approach (figure 5.1). The dependence of the execution times on π is basically the same for both approaches. If the linked list approach is applied PSSS is obviously more efficient than PSS, while for the compiled code approach it depends on π whether PSS or PSSS is better. This is the most striking difference between the both approaches as can be seen in fig.5.5. Two reasons for this difference can be given. Firstly the linked list approach is about two times slower than the compiled code approach. Therefore the extra overhead cost (in time) for PSSS compared with PSS is relatively smaller for the linked list approach. Secondly in the linked list approach only for pivots for which the activity is changed, the linked lists are modified. In this way the low pivot variability is exploited by limiting the overhead operations.

5.5. Results

In usual bipolar circuits pivot activities from .34 to .76 are observed during DC analysis (this range is indicated in the figures 5.1, 5.4 and 5.5 by an arrow). In table 5.4 some results are listed. The first line of this table contains the relative number of operations remaining after executing low type pivotsteps (type 1 to 4, see table 1.1). The second line contains the smallest and the largest value of pivot activity observed for the circuits. Lines 3 to 6 concern the comparison of two cases. In the first case PSS or PSSS is applied in one Newton iteration to the remaining pivotsteps (of type 6; type 5 pivotsteps are not present: DC analysis). In the second case all types of pivotsteps are executed in a Newton iteration and neither PSS nor PSSS is applied. The time spent in the first case relative to the time spent in the second case is listed in lines 3 to 6. It is accounted for that the computation of a solution to the Schur complement remains the same.

TABLE 5.4.

		NAND		NOR		flip flop		μA 709		μA 741	
type 6 operations		.465		.450		.570		.517		.667	
π_{small}, π_{large}		.415	.692	.344	.722	.520	.657	.519	.760	.607	.736
compiled code	PSS	.311	.390	.305	.399	.444	.483	.378	.450	.547	.590
	PSSS	.304	.394	.301	.410	.435	.480	.368	.454	.539	.593
linked list	PSS	.305	.387	.294	.380	.441	.480	.376	.450	.546	.590
	PSSS	.285	.369	.279	.396	.414	.455	.345	.427	.510	.563

Several suppositions are made concerning the operations which are not analysed in detail in the preceding (fore- and backsubstitution, handling of the Schur complement). It is supposed that the Schur complement is a full matrix. Actually this is not the case. The largest train method determining a BLT-form tends to given the Schur complement a BLT-form as well. So a solution to the Schur complement can be computed faster than is supposed here. This means that the values in the table are slightly pessimistic. The second supposition is that a multiplication and a division use equal time. Other operations (data-handling, additions) are neglected. This will cause no severe error because the numbers of these operations are proportional to numbers of multiplications and divisions. (E.g. in the L\U-decomposition the number of additions is equal to the number of multiplications).

From the table it can be concluded that a speed-up of a factor 1.7 always can be achieved. The largest speed-up is 3.6.

The time spent on the evaluation of nonlinear functions is not comprised in the preceding analysis. It will be shown in section 6.6 that the function evaluation corresponding to a skipped pivot can be avoided. Then the number of function evaluations is proportional to the pivot activity. The values listed in table 5.4 become closer to the pivot activity if the function evaluations are taken into account. Usually the function evaluations consume most of the analysis time of a circuit: a typical value is 80% of the analysis time. Thus the pivot activity is the most important value in table 5.4.

[5.1] I.S. Duff, "A survey of sparse matrix research", Proceedings of the IEEE, Vol. 65, pp. 500-535 (1977).

[5.2] B. Dembart, A.M. Erisman, "Hybrid sparse-matrix methods", IEEE Transactions on Circuit Theory, Vol. CT-20, pp. 641-649 (1973).

[5.3] F.G. Gustavson, W. Liniger, R. Willoughby, "Symbolic generation of an optimal Crout algorithm for sparse systems of linear equations", Journal of the ACM, Vol. 17, pp. 87-109 (1970).

[5.4] G.D. Hachtel, R.K. Brayton, F.G. Gustavson, "The sparse tableau approach to network analysis and design", IEEE Transactions on Circuit Theory, Vol. CT-18, pp. 101-113 (1971).

[5.5] H.B. Lee, "An implementation of Gaussian elimination for sparse systems of linear equations", in Sparse Matrix Proceedings (R.A. Willoughby, Ed.), Yorktown Heights, N.Y., IBM report RA1 (#11707), pp. 75-83 (1969).

[5.6] PDP 11/60 processor handbook, Digital Equipment Corporation (1977).

6. THE FORESUBSTITUTION AND BACKSUBSTITUTION

In the foregoing the skipping of pivotsteps was studied. In this section attention will be paid to the skipping of parts of the fore- and backsubstitution. Like the L\U-decomposition the fore- and backsubstitution can be decomposed into steps. The q^{th} fore-substitution step consists of the operations:

$$y_q \leftarrow y_q \;/\; l_{qq}$$

$$y_i \leftarrow y_i - l_{iq} y_q \qquad q < i \le n$$

The q^{th} backsubstitution step consists of:

$$z_q \leftarrow y_q$$

$$z_q \leftarrow z_q - u_{qj} z_j \qquad \max\,[t,q] < j \le n$$

(Because of the BLT form u_{qj} is zero for $q < j \le t$).

The skipping of steps in the L\U-decomposition and the fore- and backsubstitution can be done in several ways. E.g. the q^{th} pivotstep may be skipped while the q^{th} foresubstitution step is executed. Four useful options, called SKIP I, SKIP II etc., will be considered in the following sections. Finally, in connection with SKIP I-IV, some refinements to the computation of x^ι, the ι^{th} Newton iterate, and the control of the pivot activity are discussed.

In this chapter we will find it convenient to use the value γ_q defined by:

$$\gamma_q = \gamma_q(x) = -\sum_{j \ne q} a_{qj} x_j$$

Now the q^{th} equation of eq.(1.12) is written as:

$$f_q(x_q) = \gamma_q$$

With use of $\gamma_q^\iota \triangleq \gamma_q(x^\iota)$ two other equalities can be written:

$$\sum_{j \ne q} a_{qj} z_j^\iota = \sum_{j \ne q} a_{qj} (x_j^{\iota+1} - x_j^\iota) = \gamma_q^\iota - \gamma_q^{\iota+1} \tag{6.1}$$

$$r_q^\iota = -\sum_{j \ne q} a_{qj} x_j^\iota - f_q(x_q^\iota) = \gamma_q^\iota - f_q(x_q^\iota) \tag{6.2}$$

For brevity the index ι will be deleted in the following if no ambiguity is possible.

In this chapter we focus only to the q^{th} pivotstep and the q^{th} fore- and backsubstitution step, with $q \le t$. We use A_+ to denote the matrix obtained from A by deleting the q^{th} row and q^{th} column. In the same way z_+ and r_+ denote the vectors obtained from z and r by deleting the q^{th} element. Finally we use δA defined by

$$\begin{aligned} \delta a_{ij} &= a_{iq} a_{qq}^{-1} a_{qj} \qquad \text{for } i > q, \quad j \neq q \\ \delta a_{ij} &= 0 \qquad \text{otherwise} \end{aligned} \tag{6.3}$$

6.1. SKIP I

First we consider the case that the q^{th} pivotstep, fore- and backsubstitution step all are skipped. The case corresponds to the deletion of both the q^{th} row and column of A and the deletion of the q^{th} element of both z and r. So we may formulate SKIP I as follows:

SKIP I: compute z according to:

$$\begin{aligned} A_+ z_+ &= r_+ \\ z_q &= 0 \end{aligned} \tag{6.4}$$

The solution z of these equations satisfies also $Az = r + \delta r$, with $\delta r_i = 0$ for $i \neq q$, and $\delta r_q = (\sum_{j \neq q} a_{qj} z_j) - r_q$. Apparently only the q^{th} element of $\bar{r}$, defined by (1.14), is nonzero. The computation of $\bar{r}_q = -\delta r_q$ can be compared with a backsubstitution step.

The consequences of SKIP I become most obvious if Newton iteration is applied to a set of linear equations. Then the common Newton iteration computes the solution in one iteration:

$$\begin{aligned} A^0 z^0 &= r^0 \\ x^1 &= x^0 + z^0 = z^0 \\ r^1 &= r^0 - A^0 x^1 = 0 \end{aligned}$$

If SKIP I is taken then the residual r^1 after the first iteration is nonzero:

$$\begin{aligned} A_+^0 z_+^0 &= r_+^0 \\ z_q^0 &= 0 \\ r^1 &= \bar{r}^1 = -\delta r^0 \quad \text{with } \delta r_i^0 = 0 \text{ for } i \neq q, \text{ and } \delta r_q^0 = (\sum_{j \neq q} a_{qj} z_j^0) - r_q^0 \end{aligned}$$

A second iteration may be applied to obtain a better approximation:

$$\begin{aligned} A_+^0 z_+^1 &= r_+^1 = -\delta r_+^0 = 0 \\ z_q^1 &= 0 \end{aligned}$$

The solution $z^1 = 0$ implies that no improvement is obtained. For nonlinear equations one may expect that, if the Newton iteration converges, an approximate solution is obtained such that the residual is arbitrarily small except for the q^{th} element.

Although no decrease of the q^{th} residual element can be expected, with some prudence, it is still possible to apply SKIP I and have some advantage. Consider the approximate solution x^l of the set of nonlinear equations $s(x) = f(x) + \bar{A}x = 0$. Assume that the residual elements r_i^l are zero for $i \neq q$ and that $|a_{qq}^l|$ is large. Let $\tilde{x}^{l+1}$ differ from x^l only for the q^{th} element:

$\tilde{x}_q^{l+1}$ is such that $\tilde{r}_q^{l+1} = -s_q(\tilde{x}^{l+1})$ equals zero. Probably $\tilde{x}_q^{l+1}$ is close to x_q^l because of

$$|a_{qq}^l\ (\tilde{x}_q^{l+1} - x_q^l)| \approx |s_q(\tilde{x}^{l+1}) - s_q(x^l)| = |r_q^l|$$

as $|a_{qq}^l|$ is large. The elements of the residual $\tilde{r}^{l+1} = -s(x^{l+1})$ are likely to be small:

$$\tilde{r}_i^{l+1} = -a_{iq}(\tilde{x}_q^{l+1} - x_q^l) \qquad \text{for } i \neq q$$

The residual achieved at any instant can be seen as a measure of the accuracy of the approximate solution obtained. Therefore one may formulate the desired accuracy in terms of upper bounds on the elements of the residual. Let $\Delta\tilde{\varepsilon}_i$ represent these upper bounds, where $\tilde{\varepsilon}_i$ is some positive number. $\tilde{\varepsilon}_i$ plays the same role as a dependable reference. However $\tilde{\varepsilon}_i$ is not the result of the application of the domination principle. Hence we call $\tilde{\varepsilon}_i$ a "pseudo" dependable reference. Thus a solution $\tilde{x}$ is accepted if $|\tilde{r}_i| = |s_i(\tilde{x})| \leq \Delta\tilde{\varepsilon}_i$ holds for all i. The elements of the residual $\tilde{r}^{l+1}$ satisfy these inequalities if $\tilde{x}_q^{l+1}$ is such that

$$|a_{iq}\ (\tilde{x}_q^{l+1} - x_q^l)| \leq \Delta\tilde{\varepsilon}_i \qquad \text{for } i \neq q$$

Apparently

$|\tilde{x}_q^{l+1} - x_q^l|$ has to be smaller than some threshold χ_q, defined by:

$$\chi_q = \Delta \min_{\substack{i \neq q \\ a_{iq} \neq 0}} \left[\frac{\tilde{\varepsilon}_i}{|a_{iq}|}\right]$$

The foregoing implies that SKIP I can be applied as long as $|\tilde{x}_q^{l+1} - x_q^l| \leq \chi_q$ is satisfied. For intermediate iterations the computed values of $\tilde{x}_q^{l+1}$ are used to verify whether $\tilde{x}_q^{l+1}$ is close enough to x_q^l. However, there is no need to accept such an entity as the new value of x_q, denoted by x_q^{l+1}. The variable x_q may be kept constant up to the last iteration. Only then x_q has to be updated to satisfy the q^{th} equation and to obtain an acceptable small residual. This procedure saves much of the work to compute $\bar{r}$ (according to (1.14)) because $\bar{r}_i$ is zero, for $i \neq q$, after intermediate iterations while after the last iteration $\bar{r}_i$, for $i \neq q$, need not be computed at all.

Actually the Newton iteration will not obtain a final solution x^l such that r_i^l is exactly zero, for $i \neq q$. This is no difficulty if r_i^l and χ_q (bounding $|\tilde{x}_q^{l+1} - x_q^l|$) are such that the induced residual element $(r_i^l + \tilde{r}_i^{l+1})$ is small enough for all $i \neq q$.

6.2. SKIP II

In order to obtain a decrease of the q^{th} element of the residual too, z_q may be computed such that the q^{th} equation is satisfied:

SKIP II: compute z according to:

$$A_+ z_+ = r_+ \tag{6.4}$$

$$z_q = a_{qq}^{-1}(r_q - \sum_{j \neq q} a_{qj} z_j) = (\gamma_q^{l+1} - f_q(x_q^l))/a_{qq}^l \tag{6.5}$$

(The latter equality follows if equations (6.1) and (6.2) are exploited.)

An equivalent set of equations uses δA defined in (6.3) and a vector δr defined by

$$\delta r_i = a_{iq} a_{qq}^{-1} r_q \quad \text{for } i > q, \quad \delta r_i = 0 \quad \text{for } i \leq q \tag{6.6}$$

The equations (6.4) and (6.5) together are equivalent to:

$$(A + \delta A)z = (r + \delta r) \tag{6.7}$$

(The q^{th} equation in (6.7) is identical to (6.5). If (6.5) is used to eliminate z_q in the i^{th} equation in (6.7) the i^{th} equation in (6.4) results.)

If SKIP II is taken to solve a set of linear equations then the norm of the residual decreases in each iteration if δA and δr are small enough. This appears from the following lemma.

Lemma 6.1: Let Newton iteration according to SKIP II be applied to solve $Az = r^0$.
If Δ is such that $\Delta \| |A| . |A^{-1}| \| < 1$ and if $|\delta A| \leq \Delta |A|$ and $\|\delta r\| \leq \Delta \|r\|$ are satisfied for this value of Δ then:

$$\frac{\|r^{l+1}\|}{\|r^{l}\|} \le \Delta \frac{1+\|\,|A|\cdot|A^{-1}|\,\|}{1-\Delta\|\,|A|\ |A^{-1}|\,\|} \tag{6.8}$$

Proof: Because $Az = r^0$ is a set of linear equations, the expression for the residual r^{l+1}, given in (1.16), reduces to: $r^{l+1} = \bar{r}^{l}$. Exploiting (1.14) we obtain:

$$|r^{l+1}| = |\bar{r}^{l}| = |\delta A z^{l} - \delta r^{l}| \le |\delta A|\,|z^{l}| + |\delta r^{l}| \tag{6.9}$$

Equation (6.7) gives an expression for z^{l}:

$$z^{l} = A^{-1}(r^{l} + \delta r^{l} - \delta A z^{l}) = A^{-1}(r^{l} - r^{l+1})$$

If this expression is substituted in (6.9) we obtain:

$$|r^{l+1}| \le |\delta A|\,|A^{-1}|\,|r^{l} - r^{l+1}| + |\delta r^{l}|$$

Hence:

$$\begin{aligned}\|r^{l+1}\| &\le \|\,|\delta r^{l}| + |\delta A|\,|A^{-1}|\,|r^{l}|\,\| + \\ &\quad \|\,|\delta A|\,|A^{-1}|\,\|\cdot\|r^{l+1}\| \\ &\le \Delta\|r^{l}\| + \Delta\|\,|A|\cdot|A^{-1}|\,\|\cdot\|r^{l}\| + \\ &\quad \Delta\|\,|A|\cdot|A^{-1}|\,\|\cdot\|r^{l+1}\|\end{aligned}$$

Now inequality (6.8) follows because of $\Delta\|\,|A|\cdot|A^{-1}|\,\| < 1$

The residual decreases if the right-hand side in (6.8) is less than one, that is if Δ is such that

$$\Delta(1 + 2\,\|\,|A|\cdot|A^{-1}|\,\|) < 1$$

is satisfied. If SKIP II is taken for nonlinear equations and the iteration converges, an arbitrarily accurate solution can be obtained.

The computation of z_q according to (6.5) is almost identical to the computation of δr_q in SKIP I. Yet SKIP II implies more operations than SKIP I because the vector $\bar{r} = \delta Az - \delta r$ has to be determined. Note that the i^{th} element of $\bar{r}$ can be computed easily with z_q:

$$\bar{r}_i = -\delta r_i + \sum_j \delta a_{ij} z_j = -a_{iq} a_{qq}^{-1}(r_q - \sum_{j\neq q} a_{qj} z_j) = -a_{iq} z_q$$

The computation of $\bar{r}$ is almost identical to the q^{th} foresubstitution step.

A condition of lemma 6.1 is that $\|\delta r\| \le \Delta\|r\|$ holds. However the

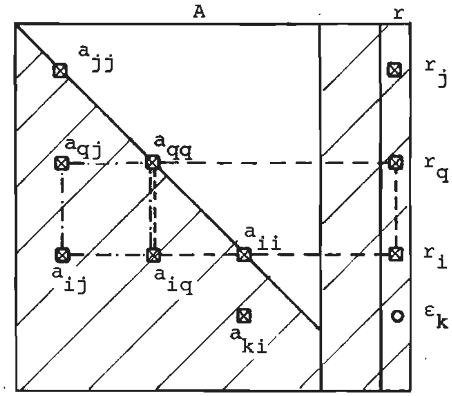

Fig.6.1. The cycles occurring in the computation of $\bar{\theta}_q$.

threshold θ_q, proposed in chapter 4, only assures that $|\delta A| \le \Delta|A|$ and $\|\delta A\| \le \Delta\|A\|$ are satisfied if the absolute value of a_{qq} exceeds θ_q.

The computation of a threshold $\hat{\theta}_q$ such that the induced perturbation δr is small enough for $|a_{qq}| \ge \hat{\theta}_q$, proceeds in much the same way as the computation of θ_q. The residual r can be considered as the n+1th column of the matrix A, see figure 6.1. Beside the common cycles within A, the cycles passing through the residual also supply conditions for the value of the pivot. For instance the cycle $\{a_{iq}, a_{qq}, r_q, r_i\}$ yields the condition

$$|a_{iq}a_{qq}^{-1}r_q| \le \Delta|r_i| \tag{6.10}$$

Consequently $\hat{\theta}_q$ is:

$$\hat{\theta}_q = \max\left[\theta_q, \Delta^{-1}|r_q| \max_{i\neq q}\left[\left|\frac{a_{iq}}{r_i}\right|\right]\right] \tag{6.11}$$

if all elements r_i of the residual are nonzero. If an element of the residual is zero then a dependable reference may be used. A vector ε of dependable references may be computed according to:

$$\varepsilon_i = \max\,[R_i, |r_i|]$$

$$R_i = \min_{\substack{k\neq i\\ a_{ki}\neq 0}}\left[\frac{\varepsilon_k}{|a_{ki}|}\right]\eta_i \qquad \text{for } i \le t, \quad R_i \triangleq 0 \quad \text{for} \quad i > t.$$

(Consider the residual as the $n+1^{th}$ column of A and compare the formulas above with the computation of ε_{ij} according to eq.(4.11).)

In the foregoing no attention is paid to the fact that the values of the residual vector will be different in each iteration. This would imply that the thresholds $\hat{\theta}_q$ and the dependable references ε_i have to be computed for each iteration again. However the time spent on this computation may outdo the time gained by skipping some pivotsteps. It is desired to compute the values θ_q and ε_i once and for all.

Consider the computation of $\hat{\theta}_q$ for a residual r with $r_q = 1$ and $r_i = 0$ for $i \neq q$, using a vector $\tilde{\varepsilon}$ of pseudo dependable references all being equal to one. By eq.(6.11) we have for $\hat{\theta}_q$:

$$\hat{\theta}_q \geq \Delta^{-1} |r_q| \max_{i \neq q} \left[\frac{|a_{iq}|}{\tilde{\varepsilon}_i} \right] = \Delta^{-1} \max_{i \neq q} [|a_{iq}|]$$

Now let r be an arbitrary residual and let SKIP II be applied for $|a_{qq}| \geq \hat{\theta}_q$. The induced perturbation δr has elements δr_i satisfying:

$$|\delta r_i| = |a_{iq} a_{qq}^{-1} r_q| \leq |a_{iq}| . \hat{\theta}_q^{-1} . |r_q| \leq \Delta |r_q|$$

Consequently we have for $\|\delta r\|$ the inequality:

$$\|\delta r\| \leq \Delta |r_q| \leq \Delta \|r\|$$

The threshold $\hat{\theta}_q$ computed in this way is independent of the residual and can be used in each iteration.

6.3. SKIP III

A third possibility is to skip the q^{th} pivotstep and back-substitution step but to execute the q^{th} foresubstitution step.

SKIP III: compute z according to:

$$\begin{aligned} z_q &= a_{qq}^{-1} r_q \\ \tilde{r}_i &= r_i - a_{iq} z_q \qquad \text{for } i \neq q \\ A_+ z_+ &= \tilde{r}_+ \end{aligned}$$

The first two equations represent the q^{th} foresubstitution step. The three equations are equivalent to the set of equations

$$Az = r + \delta r \quad \text{with } \delta r_q = \sum_{j \neq q} a_{qj} z_j \quad \text{and} \quad \delta r_i = o \quad \text{for } i \neq q.$$

This appears if the substitution $z_q = a_{qq}^{-1} r_q$ is performed in the latter set of equations.

Suppose SKIP III is taken to solve $Az = r$. The following lemma supplies a sufficient condition for convergence. It is noteworthy that this condition is formulated in terms of δA instead of δr.

Lemma 6.2: Let Newton iteration according to SKIP III be applied to solve $Az = r^0$. Let δA be defined by (6.3). Then we have for nonsingular A_+

$$\frac{\| z_+^{\iota+1} \|}{\| z_+^{\iota} \|} \leq \| (A_+)^{-1} \delta A_+ \|$$

Proof: Consider the residual $r^{\iota+1}$ after the $\iota+1^{th}$ iteration:

$$r^{\iota+1} = \bar{r}^{\iota} = -\delta r^{\iota}$$

with $\delta r_q^{\iota} = \sum_{j \neq q} a_{qj} z_j^{\iota}$ and $\delta r_i^{\iota} = 0$ for $i \neq q$. In the $\iota+2^{th}$ iteration $\tilde{r}^{\iota+1}$ is computed according to:

$$\tilde{r}_i^{\iota+1} = r_i^{\iota+1} - a_{iq} z_q^{\iota+1} = -a_{iq} a_{qq}^{-1} r_q^{\iota+1} =$$

$$= a_{iq} a_{qq}^{-1} \sum_{j \neq q} a_{qj} z_j^{\iota} \quad \text{for } i \neq q$$

Using δA_+ we may write:

$$\tilde{r}_+^{\iota+1} = \delta A_+ z_+^{\iota}$$

Now the lemma follows from the observation

$$z_+^{\iota+1} = (A_+)^{-1} \tilde{r}_+^{\iota+1} = (A_+)^{-1} \delta A_+ z_+^{\iota}$$

The iteration according to SKIP III converges if $\| (A_+)^{-1} \delta A_+ \| < 1$, for then the norm of z_+ decreases. Consequently $|z_q|$ decreases because of $|z_q^{\iota+1}| \leq \theta_q^{-1} \|A_+\| \cdot \|z_+^{\iota}\|$. For instance convergence is achieved if Δ has a value such that $\Delta \| (A_+)^{-1} \| \cdot \|A_+\| < 1$ holds, and at the same time $\| \delta A_+ \| \leq \Delta \|A_+\|$ for this value of Δ is satisfied.

SKIP III requires the same number of operations as SKIP II. For the computation of δr_q^{ι} is comparable with the evaluation of the sum in (6.5) and the determination of $\tilde{r}_+^{\iota+1}$ is comparable with the determination of the residual $\bar{r}^{\iota}$ if SKIP II is applied.

6.4. SKIP IV

The last option considered here is almost the same as SKIP III. The difference is that z_q is solved such that the q^{th} equation is satisfied:

SKIP IV: compute z according to

$$\tilde{r}_i = r_i - a_{iq}a_{qq}^{-1}r_q \quad \text{for } i \neq q$$

$$A_+z_+ = \tilde{r}_+$$

$$z_q = a_{qq}^{-1}(r_q - \sum_{j\neq q} a_{qj}z_j) = (\gamma_q^{\iota+1} - f_q(x_q^\iota))/a_{qq}^\iota \quad (6.5)$$

These equations are equivalent to $(A + \delta A)z = r$ with δA defined by (6.3).

The following lemma supplies a sufficient condition for convergence if SKIP IV is applied to solve a linear set of equations.

Lemma 6.3: Let Newton iteration according to SKIP IV be applied to solve $Az = r^0$. Let δA be defined by 6.3. If Δ is such that $\Delta\| |A|\cdot|A^{-1}| \| < 1$ holds and if δA satisfies $|\delta A| \leq \Delta|A|$ for this value of Δ then:

$$\frac{\|r^{\iota+1}\|}{\|r^\iota\|} \leq \frac{\Delta\| |A|\cdot|A^{-1}| \|}{1-\Delta\| |A|\cdot|A^{-1}| \|} \quad (6.12)$$

Proof: The residual after the $\iota+1^{th}$ iteration is: $r^{\iota+1} = \bar{r}^\iota = \delta A z^\iota$. From $(A + \delta A)z = r$ follows that $z^\iota = A^{-1}(r^\iota - \delta A z^\iota) = A^{-1}(r^\iota - r^{\iota+1})$. Hence $r^{\iota+1} = \delta A\cdot A^{-1}(r^\iota - r^{\iota+1})$ and consequently:

$$\|r^{\iota+1}\| \leq \Delta\| |A|\cdot|A^{-1}| \|.\|r^\iota\| + \Delta\| |A|\cdot|A^{-1}| \|.\|r^{\iota+1}\|.$$

Because $\Delta\| |A|\cdot|A^{-1}| \|$ is less than one, inequality (6.12) follows.

The right-hand side of (6.12) is less than one if Δ satisfies $2\Delta\| |A|\cdot|A^{-1}| \| < 1$.

The difference between SKIP IV and SKIP II is that SKIP IV implies the execution of the q^{th} foresubstitution step. Yet SKIP IV can be executed with practically the same number of operations as SKIP II because the computation of $\bar{r}^\iota$ and the q^{th} foresubstitution step in the $\iota+2^{th}$ iteration can be combined. The i^{th} element of $\bar{r}^\iota = \delta A^\iota z^\iota$ is $\bar{r}_i^\iota = a_{iq}(a_{qq}^\iota)^{-1} \sum_{j\neq q} a_{qj}z_j^\iota$ for $i \neq q$. Using equation (6.5) for z_q^ι we may write $\bar{r}_i^\iota = a_{iq}\{(a_{qq}^\iota)^{-1}r_q^\iota - z_q^\iota\}$. If we take eq.(1.16) for $r^{\iota+1}$ then for the residual $\tilde{r}^{\iota+1}$ after the q^{th} foresubstitution step in the $\iota+2^{th}$ iteration we obtain

$$\tilde{r}_i^{\iota+1} = r_i^{\iota+1} - a_{iq}(a_{qq}^{\iota+1})^{-1} r_q^{\iota+1} = \bar{r}_i^\iota + a_{ii}^\iota z_i^\iota + f_i(x_i^\iota) - f_i(x_i^{\iota+1}) - a_{iq}(a_{qq}^{\iota+1})^{-1} r_q^{\iota+1}$$

$$\tilde{r}_i^{l+1} = a_{iq} \left\{ (a_{qq}^l)^{-1} r_q^l - z_q^l - (a_{qq}^{l+1})^{-1} r_q^{l+1} \right\} + a_{ii}^l z_i^l - f_i(x_i^{l+1}) + f_i(x_i^l)$$

The first term in the right-hand side is the combination of the computation of $\bar{r}^l$ and the q^{th} foresubstitution step.

The latter three possibilities, SKIP II, III and IV, all save practically the same number of operations. The most important difference is the sufficient condition for convergence. The loosest condition is obtained for SKIP III, the hardest for SKIP II. Moreover the condition $\|\delta r\| \leq \Delta \|r\|$ has to be satisfied for SKIP II.

6.5. An alternative to the backsubstitution

In the foregoing sections we saw already some alternative computations of x_q^{l+1} in the case that the q^{th} pivotstep could be skipped. SKIP II and IV are two examples which compute z_q^l not by backsubstitution but directly from the q^{th} linearized equation. In connection with SKIP I we discussed the computation of $\tilde{x}_q^{l+1}$ such that the q^{th} nonlinear equation is satisfied:

$$\tilde{x}_q^{l+1} = f_q^{-1}(\gamma_q(x^{l+1})) \tag{6.13}$$

The latter computation of $\tilde{x}_q^{l+1}$ is more adventageous than the computation by common backsubstitution in the case that the value of a_{qq}^l is large. Generally a large value of a_{qq}^l implies that u_{qj}^l, $t + 1 \leq j \leq n$, and y_q^l are small and consequently z_q^l is small. Then x_q^l gets only a small update and the new value x_q^{l+1} is only slightly better than x_q^l. It may take a lot of iterations before a sufficiently accurate solution is obtained, see appendix C. The application of (6.13) may save most of those iterations.

Important is that equation (6.13) implies that $\tilde{x}_q^{l+1}$ is computed from the function value $f_q(\tilde{x}_q^{l+1}) = \gamma_q^{l+1}$ applying the inverse function f_q^{-1}. The computation is executed if the pivot is passive. The common way is to compute x_q^{l+1} according to

$$x_q^{l+1} = x_q^l + z_q^l \tag{6.14}$$

and to determine the function value by evaluating $f_q(x_q^{l+1})$. The latter computation is performed if the pivot is active. Then we can consider x_q as the "controlling variable": the iteration computes

x_q^{l+1} first and $f_q(x_q^{l+1})$ is derived thereafter. If we apply (6.13) then $\gamma_q(x^{l+1}) = f_q(x_q^{l+1})$ is the controlling variable which is computed first and x_q^{l+1} is computed afterwards. Note that both x_q and $f_q(x_q)$ are circuit variables: usually x_q is the current of a circuit element and $f_q(x_q)$ is the voltage of the same element or the other way round. A controlling variable is computed with the intention to satisfy all equations in the set. The other variable associated with it is computed to satisfy only the equation describing the relation of two variables (voltage and current) of a circuit element.

The general experience is that it depends on the value of the derivative of $f_q(x_q)$ which of the variables x_q or $f_q(x_q)$ should be used as controlling variable. If $f_q(x_q^{l+1}) - f_q(x_q^l)$ is small relative to $x_q^{l+1} - x_q^l$ then x_q should be the controlling variable. For then the residual r^l, which mainly arises because of the nonlinearity of the equations, is expected to be small. This case is likely to happen if $a_{qq}(x_q) = f_q'(x_q)$ is small.

Because active pivots are small (anyhow they are smaller than their thresholds) the fact that normally x_q is the controlling variable, is favourable for active pivots. For passive pivots we want $f_q(x_q)$ as controlling variable because a passive pivot is relatively large (larger than its threshold). For SKIP I the computation of $f_q(\tilde{x}_q^l)$ and subsequently $\tilde{x}_q^l$ according to (6.13) was already proposed. SKIP II and SKIP IV can be adapted easily of equation (6.5) is replaced by equation (6.13). If SKIP III is adapted in the same way it looses its typical character and becomes equal to SKIP IV.

6.6. Some refinements of the control of the pivot activity

If equation (6.13) is taken to compute $\tilde{x}_q^{l+1}$ a problem arises concerning the control of the pivot activity. For the difference $\tilde{z}_q^l \triangleq \tilde{x}_q^{l+1} - x_q^l$ may be so large that the perturbations δA and δr induced by the skipping of the q^{th} pivotstep and fore- and back-substitution step become too large.

Firstly consider the computation of x_q^{l+1} according to (6.14) while z_q^l is computed according to (6.5):

$$z_q^l = (\gamma_q^{l+1} - f_q(x_q^l))/a_{qq}^l \tag{6.5}$$

Note that z_q^l and the elements of the perturbations δA and δr, defined by (6.3) and (6.6), which are induced by SKIP II and SKIP IV, all are proportional to $(a_{qq}^{l})^{-1}$.

Secondly consider the computation of $\tilde{x}_q^{l+1}$ according to (6.13). Arbitrarily let $\tilde{x}_q^{l+1}$ be larger than x_q^{l}. From the mean value theorem it follows that the domain $[x_q^{l},\tilde{x}_q^{l+1}]$ contains a value ξ such that $\tilde{z}_q^{l}$ satisfies:

$$\tilde{z}_q^{l} = \tilde{x}_q^{l+1} - x_q^{l} = (f_q(\tilde{x}_q^{l+1}) - f_q(x_q^{l}))/a_{qq}(\xi) \tag{6.15}$$

If we compare equations (6.15) and (6.5) and consider that γ_q^{l+1} is equal to $f_q(\tilde{x}_q^{l+1})$ we see that $\tilde{z}_q^{l}$ would be equal to z_q^{l} if $a_{qq}(\xi)$ would be equal to a_{qq}^{l}. Consequently the replacement of equation (6.5) by (6.13) is identical to the replacement of a_{qq}^{l} by $a_{qq}(\xi)$. The point is whether $|a_{qq}(\xi)| \geq \hat{\theta}_q$ holds or not. If it does not hold we may try to determine a value $\approx{x}_{q+1}$, contained in $[x_q^{l},\tilde{x}_q^{l+1}]$, such that

$$\approx{x}_q^{l+1} - x_q^{l} = (f_q(\approx{x}_q^{l+1}) - f_q(x_q^{l}))/\hat{\theta}_q$$

An alternative way is to take the value $\bar{x}_q^{l+1}$ being as close to $\tilde{x}_q^{l+1}$ as possible with the restriction that $|a_{qq}(x_q)| \geq \hat{\theta}_q$ holds for all x_q in the domain $[x_q^{l},\bar{x}_q^{l+1}]$. The latter possibility is safer because the value of a_{qq} is bounded in the domain $[x_q^{l},\bar{x}_q^{l+1}]$ while it may achieve any arbitrary value in the domain $[x_q^{l},\approx{x}_q^{l+1}]$.

To obtain the value $\bar{x}_q^{l+1}$ it is convenient to exploit the domain Ξ_q, defined by:

$$\Xi_q \triangleq \{x_q \in R \mid |a_{qq}(x_q)| \geq \hat{\theta}_q\}$$

If Ξ_q is not connected it consists of a set of connected components Ξ_q^{i} : $\Xi_q = \bigcup_i \Xi_q^{i}$. The determination of $\bar{x}_q^{l+1}$ proceeds as follows. Determine the domain Ξ_q^{i} containing x_q^{l} (such a domain exists for the pivot is passive). If $\tilde{x}_q^{l+1}$ is contained in Ξ_q^{i} as well we take $x_q^{l+1} = \tilde{x}_q^{l+1}$. Otherwise $\bar{x}_q^{l+1}$ is the boundary value of Ξ_q^{i} being between x_q^{l} and $\tilde{x}_q^{l+1}$, and we take $x_q^{l+1} = \bar{x}_q^{l+1}$. Note that, if the test "$x_q^{l+1} \in \Xi_q$" supersedes the test "$|a_{qq}(x_q^{l+1})| \geq \hat{\theta}_q$", the evaluation of the value of the pivot can be saved in SKIP I and SKIP II. Often Ξ_q consists of only one connected component and has only one finite boundary value (e.g. if f_q is the exponential diode characteristic). Then the test "$x_q^{l+1} \in \Xi_q$" is as simple as the test "$|a_{qq}(x_q^{l+1})| \geq \hat{\theta}_q$".

If we apply SKIP I even the evaluation of x_q^{l+1} can be saved if we inspect the value $f_q(x_q^{l+1})$ instead of x_q^{l+1} or a_{qq}^{l+1}. Let Γ_q be defined by:

$\Gamma_q = \{f_q(x_q) \in R \mid x_q \in \Xi_q\}$.

A requirement for the application of SKIP I is that $|\tilde{x}_q^{\iota+1} - x_q^{\iota}| \leq \chi_q$. Therefore we may partition Ξ_q and Γ_q in connected components Ξ_q^i and Γ_q^i with two constraints. Firstly the size of Ξ_q^i, $|\Xi_q^i|$, is not larger than χ_q for each i and secondly each Ξ_q^i is related to a unique Γ_q^i such that $x_q \in \Xi_q^i$ implies $f_q(x_q) \in \Gamma_q^i$ and vice versa. The constraints imply that if $f_q(x_q^{\iota})$ and $f_q(\tilde{x}_q^{\iota+1}) = \gamma_q^{\iota+1}$ are in the same domain Γ_q^i then the pivot stays passive and SKIP I can be chosen. If $f_q(x_q^{\iota})$ and $f_q(\tilde{x}_q^{\iota+1})$ are in different but adjacent domains then SKIP II (with or without the modification proposed in the preceding section) can be chosen. Note that we easily may transfer from SKIP I to SKIP II only by computing $x_q^{\iota+1}$ (according to either (6.14) or (6.13)). If $f_q(\tilde{x}_q^{\iota+1})$ is not contained in any domain Γ_q^i then we may determine $x_q^{\iota+1}$ in almost the same way as before. The difference is that $x_q^{\iota+1}$ may be the boundary value of a domain Ξ_q^i which is not identical to the domain Ξ_q^k containing x_q^{ι}, but then Ξ_q^i and Ξ_q^k are connected by a series of adjacent domains Ξ_q^j.

The advantage of the test of the value of $f_q(x_q^{\iota+1})$ is that $x_q^{\iota+1}$ need not be computed if $f_q(x_q^{\iota+1})$ is in the same domain as $f_q(x_q^{\iota})$.

Even if χ_q is small a domain Γ_q^i may be very large. If $a_{qq} = f_q'(x_q)$ is very large for all x_q in the small domain Ξ_q^i then the corresponding domain Γ_q^i is comparatively much larger. This is for instance the case if f_q is the inverse of the exponential diode characteristic. Then a_{qq} may become extremely large. Yet the constraint $|\Xi_q^i| \leq \chi_q$ may imply that the partitioning of Ξ_q yields series of many adjacent domains Ξ_q^i. Then it is laborious to determine the domain Ξ_q^i containing $f_q(\tilde{x}_q^{\iota+1})$. Because a domain Ξ_q^i in such a series generally is small, it is unlikely that $f_q(x_q)$ and $f_q(\tilde{x}_q^{\iota+1})$ are in that same domain Ξ_q^i. Mostly, if $f_q(x_q^{\iota})$ is in a small domain, SKIP I cannot be applied because $f_q(\tilde{x}_q^{\iota+1})$ is in another domain. Therefore, without a large drawback, a series of ajacent domains Ξ_q^i can be united to one domain Ξ_q^j. If $f_q(x_q^{\iota})$ and $f_q(\tilde{x}_q^{\iota+1})$ are in the same domain Ξ_q^j SKIP I is not taken but for instance SKIP II. In this way the number of domains can be reduced considerably. Again consider the logarithmic function being the inverse of the characteristic of a bipolar diode: $f(x_q) = V_T \ln (\frac{x_q+I_S}{I_S})$ and $a_{qq} = f_q'(x_q) = \frac{V_T}{x_q+I_S}$ while $x_q + I_S$ is always positive. The domain Ξ_q is

$$\Xi_q = \{x_q \mid -I_S < x_q \le \frac{V_T}{\hat{\theta}_q} - I_S\}$$

Ξ_q can be partitioned in Ξ_q^1 and Ξ_q^2 in the case $\chi_q < \frac{V_T}{\hat{\theta}_q}$:

$$\Xi_q^1 = \{x_q \mid -I_S < x_q \le \chi_q - I_S\}$$

$$\Xi_q^2 = \{x_q \mid \chi_q - I_S < x_q \le \frac{V_T}{\hat{\theta}_q} - I_S\}$$

The corresponding Γ_q^1 and $\bar{\Gamma}_q^2$ are

$$\Gamma_q^1 = \{f_q(x_q) \mid -\infty < f_q(x_q) \le V_T \ln (\frac{\chi_q}{I_S})\}$$

$$\bar{\Gamma}_q^2 = \{f_q(x_q) \mid V_T \ln (\frac{\chi_q}{I_S}) < f_q(x_q) \le V_T \ln (\frac{V_T}{\hat{\theta}_q I_S})\}$$

6.7. Convergence, accuracy and pivot activity

Although the convergence of the Newton iteration has not been investigated exhaustively if pivotstep skipping is applied, some experiments have been done. PSS and PSSS have been used in the computation of the DC solution of some bipolar circuits. This is one of the hardest tasks in circuit analysis. The exponential characteristic of a bipolar transistor is highly nonlinear, i.e. the derivative $\frac{di}{dv}$ has a large range. It varies from almost zero (theoretically $\frac{di}{dv}$ is less than 10^{-12} mho for $v < 0$ Volt) to the order of 10^{-1} mho. Such a derivative may become much larger or much smaller than the mean conductivity in these circuits which is of the order of 10^{-5} to 10^{-3} mho. The variation of these derivatives is mostly harmful to the convergence of the iteration. Therefore a good estimate of the solution such that the derivatives do not differ too much from the values they achieve for the real solution, is advantageous. But in DC analysis mostly no good estimate is available, in contrast to transient analysis where the solution for some timepoint constitutes a good estimate for the solution for the next timepoint.

In the experiments pseudo dependable references were applied instead of real dependable references. For the Jacobian

$\tilde{\varepsilon}_{ij} = 1 \approx \frac{\|A\|}{n}$ was used (compare section 4.3).

For with a simple scaling (i.e. express the currents in mA) most matrix coefficients become of the order of one. Note that the choice of ε_{ij} is not so critical because a too large value of ε_{ij} can always be corrected by choosing the accuracy factor Δ smaller. The pseudo dependable references associated with the residual were chosen equal to one also (compare section 6.1). From the options discussed in this chapter, SKIP I was selected, which is the only option which does not assure that the residual will decrease continuously. The conclusion is that the conditions for convergence were unfavourable in the experiments. Table 6.1 shows the number of iterations observed in the experiments.

The rate of convergence appeared to vary a great deal. Firstly the rate depended on the circuit, with the tendency that larger circuits had a slower rate of convergence. Secondly the rate depended on the input signals or, equivalently, on the solution to be computed. Finally it depended on the accuracy of the computation, i.e. on the factor Δ.

The dependence on Δ was obscure. With a low accuracy the convergence was sometimes better than for a high accuracy. An explanation may be that with a low accuracy an approximate solution is acceptable. (Note that simply continuing the iteration may not yield a better solution if SKIP I is applied.) A more accurate solution may require more iterations.

In other cases the convergence was sometimes better for a high accuracy than for a low accuracy. Anyhow if the factor Δ became too large the Newton iteration did not converge. Mostly no convergence was obtained for $\Delta = 1$. Some circuits (ECL OR/NOR gate, TTL NAND gate) still exhibited convergence if Δ was slightly larger than one, although this value of Δ is not justified theoretically.

TABLE 6.1.

Typical number of iterations

(20) and (45) are incidentally observed

	TTL NAND	ECL OR/NOR	ECL flip-flop	μA 709	μA 741
number of iterations	5-10	6-9	5-10 (20)	20 (45)	35-50

For two circuits showing convergence for all values of Δ not exceeding 2, the ECL OR/NOR gate and the TTL NAND gate, the mean pivot activity during the iteration process is depicted as a function of the accuracy factor Δ in figure 6.3. The circuits are shown in figure 6.2. As it could be expected, the pivot activity increases with decreasing Δ. It is noteworthy that the pivot activity π exhibits some saturation far before π achieves one, its maximum value. Apparently during the iteration a number of diodes is strongly reverse biased. The consequence is that the range of the pivot activity is restricted for $\Delta > 10^{-18}$.

However if Δ becomes small enough then the value of π becomes one. In the experiments the value of Δ such that π became one depended

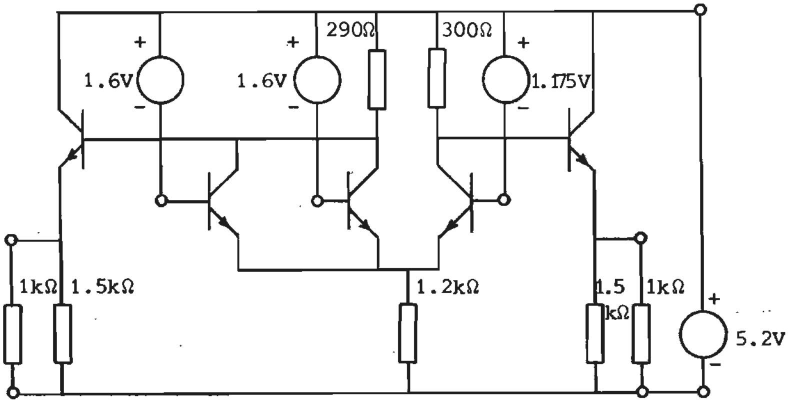

Fig.6.2.a. An ECL OR/NOR gate.
The voltage sources represent the excitations used in the experiments.

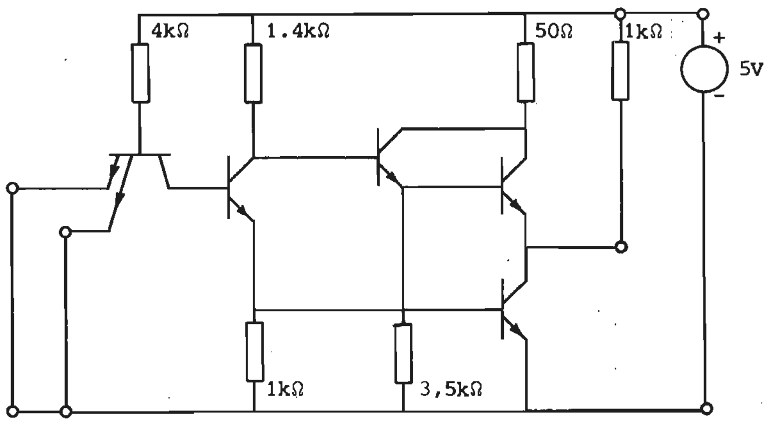

Fig.6.2.b. A TTL NAND gate.

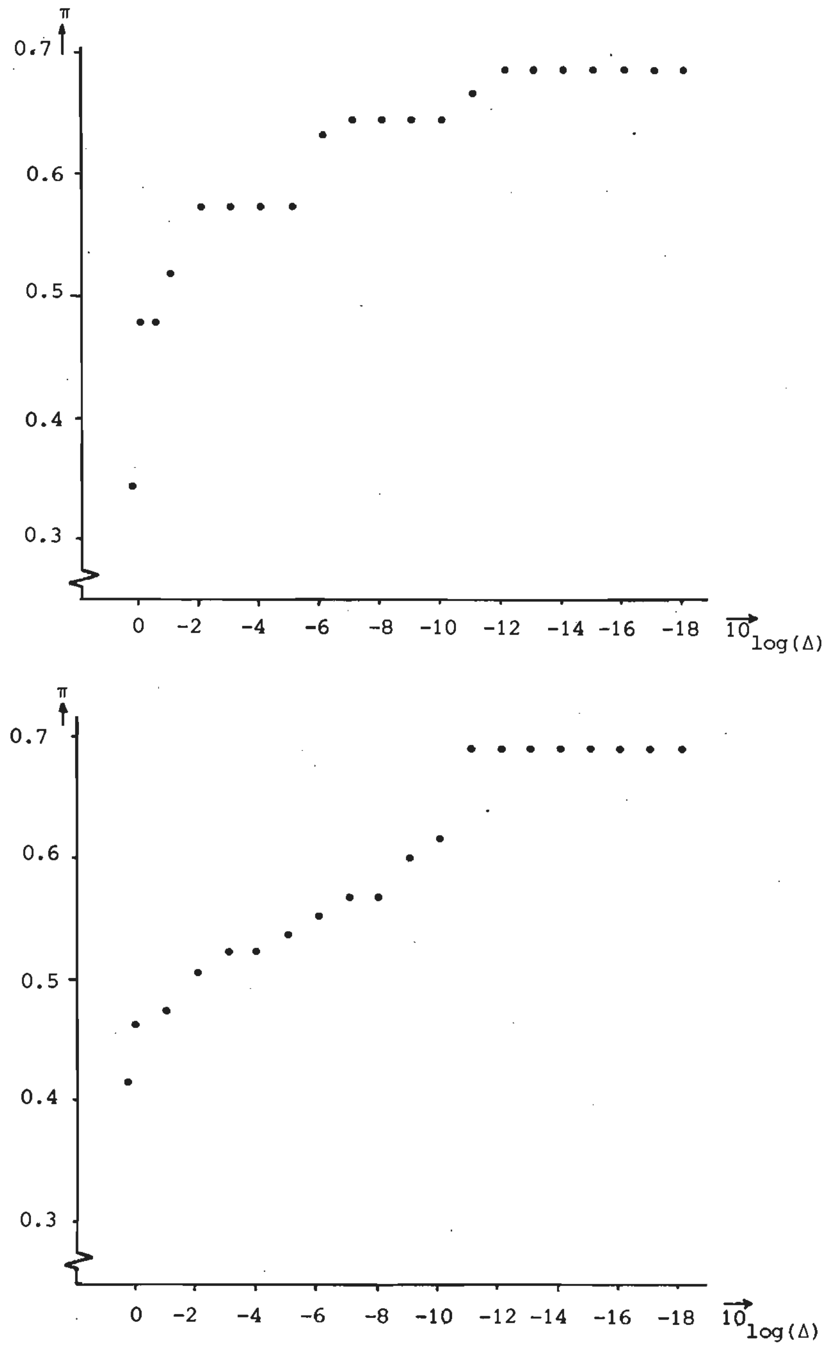

Fig.6.3. The pivot activity as a function of the accuracy for an ECL OR/NOR gate (top) and the TTL NAND gate (bottom).

on properties of the software and the machine used. Hence this phenomenon is not indicated in figure 6.3.

For usual analysis of these circuits a value of Δ not smaller than 10^{-2} or 10^{-3} yields a sufficiently accurate solution. The norm of the error vector of the solution and the norm of the residual are then in the same order of magnitude as Δ. For values of Δ larger than 10^{-3} the dependence of the pivot activity on the accuracy factor Δ is the largest. Although the variation of the execution time achieved by varying Δ is not so large, it is clearly perceptible. If Δ varies from 2 to 10^{-2} then for the ECL OR/NOR gate the relative execution time of the linked list method using PSSS varies from .34 to .57. This can be concluded from figures 6.3 and 5.4. For the TTL NAND gate this variation appears to be smaller. The relative execution time varies from .38 to .49.

7. SOME SPECIAL CIRCUIT ELEMENTS

In this chapter we will comment the application of pivotstep skipping in some particular cases which differ from the general case considered in the preceding chapters. Firstly we focus to reactive elements by considering the equations describing a capacitor. Next we show how field effect transistors can be modelled such that a set of equations of the form (1.10) arises. Finally we pay attention to the application of pivotstep skipping to a set of nonlinear equations which cannot be cast into the form (1.10).

7.1. Capacitors

When reactive elements like capacitors and inductors are present in the circuit, differential equations arise. For the numerical solution of these equations integration formulas are introduced in order to eliminate differential terms. The ordering of such equations in a BLT matrix and the application of pivotstep skipping deserves some attention. This section is restricted to the application of linear multistep formulas, a class of integration formulas commonly used for the transient analysis of electronic circuits [7.1, 7.2]. The Runge-Kutta method for instance, seems to be less attractive. Generally the application of a k stage Runge-Kutta process [7.3] to a set of n differential equations requires the solution of a set of kn equations for each time-point. Moreover it is likely that the number of essential variables increases considerably for implicit Runge-Kutta methods.

Here we consider capacitors exclusively. Inductors can be dealt with in much the same way. Let q denote the charge on a capacitor, v denote its voltage, i its current and t the time. We suppose that a nonlinear capacitor can be described by the equations:

$$q = f(v) \tag{7.1}$$

$$i = \frac{dq}{dt}$$

The latter equation is used to eliminate the time derivative $\frac{dq}{dt}$. Let h^τ denote the τ^{th} time-step, i.e. $h^\tau = t^\tau - t^{\tau-1}$, and let q^τ, v^τ, i^τ denote the values of q, v and i respectively for $t = t^\tau$. We write a linear multistep formula:

$$q^{\tau} = \alpha_1 q^{\tau-1} + \alpha_2 q^{\tau-2} + \ldots + \alpha_k q^{\tau-k} +$$

$$+ h_{\tau}(\beta_0 i^{\tau} + \beta_1 i^{\tau-1} + \ldots + \beta_k i^{\tau-k})$$

If β_0 equals zero the formula is called explicit, otherwise implicit. We introduce ζ^{τ} which is identical to the part of the formula concerning only values of q and i at preceding time-points $t^{\tau-1}, \ldots, t^{\tau-k}$:

$$\zeta^{\tau} = \alpha_1 q^{\tau-1} + \ldots + \alpha_k q^{\tau-k} + h_{\tau}(\beta_1 i^{\tau-1} + \ldots + \beta_k i^{\tau-k})$$

Hence:

$$q^{\tau} - h_{\tau}\beta_0 i^{\tau} = \zeta^{\tau} \tag{7.2}$$

ζ^{τ} can be considered as a source value for the circuit at $t = t^{\tau}$.

If the linear multistep formula is explicit the Jacobian becomes singular if the differential capacitance $\frac{dq}{dv} = f'(v)$ becomes zero. For $\frac{dq}{dv} = 0$ implies that the row of the Jacobian associated with equation (7.1) contains only one nonzero coefficient. In the same way $\beta_0 = 0$ implies that the row associated with (7.2) has only one nonzero coefficient. Both coefficients are in the column associated with q^{τ} and consequently the Jacobian is singular. Therefore an implicit formula has to be applied if $\frac{dq}{dv}$ may become zero or very small. By the way implicit methods are preferable even if $\frac{dq}{dv}$ is large [7.1].

If $\frac{dq}{dv}$ is excluded to be pivot because its value may become too small, the ordering of (7.1) and (7.2) depicted in figure 7.1 is obtained. The pivot $\frac{dv}{dq}$ will never become zero, for this would mean

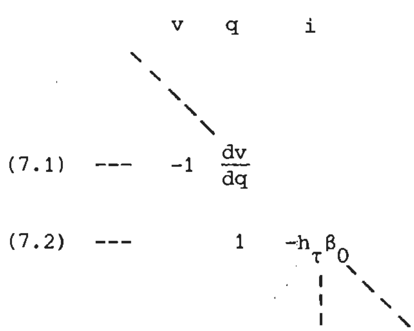

Fig.7.1.

that the capacitance becomes infinitely large. Realistic values for the parasitic nonlinear capacitances of usual semiconductor devices are of the order of pikofahrads. For these capacitances the value $10^{11}F^{-1}$ often suffices as a lower bound to $\frac{dv}{dq}$. For this size of the capacitance it is better to scale q down to for instance pikocoulombs. Then the lower bound becomes $10^{-1}pF^{-1}$. If this is followed by a rowscaling of (7.2) the size of the very small coefficients in this row (h_τ may be of the order of nanoseconds) becomes closer to one as well. The condition $\beta_0 \neq 0$ assures that the pivot $-h_\tau\beta_0$ does not become too small because extreme small values of the time-step are avoided with a good integration method.

7.2. Field effect transistors

A set of equations of the form given in (1.10) is attractive because its Jacobian only has the pivots of the triangular submatrix as variable coefficients. It is straightforward to obtain such a set if in bipolar circuits the Ebers-Moll transistor model is chosen (see section 1.4). Such a set can be obtained also for circuits with field effect transistors but some explanation may be useful.

First a MOS transistor [7.4] is considered. Its circuit model is given in figure 7.2. The usual form of the equations using a parameter K_G and the threshold voltage V_T, is:

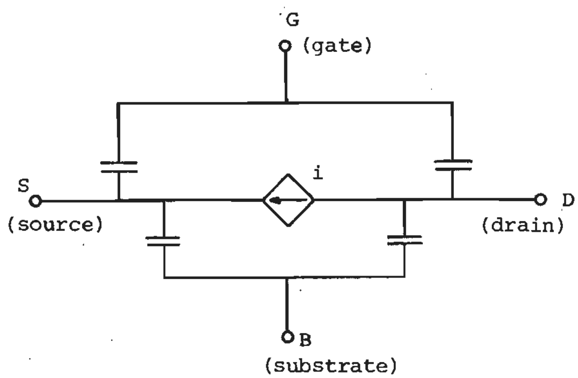

Fig. 7.2. Model of MOS transistor (n-channel).

$$i = K_G[2(v_{GS} - V_T)v_{DS} - v_{DS}^2] \qquad 0 \le v_{DS} < v_{GS} - V_T \tag{7.3}$$

$$i = K_G(v_{GS} - V_T)^2 \qquad 0 < v_{GS} - V_T \le v_{DS} \tag{7.4}$$

$$i = 0 \qquad v_{GS} - V_T \le 0 \le v_{DS} \tag{7.5}$$

Equation (7.3) yields two variable coefficients in one row of the Jacobian. With the help of the substitution $v_{DS} = v_{GS} - v_{GD}$ in equation (7.3) the equations (7.3) to (7.5) can be cast into the form:

$$\left.\begin{array}{lll} i = i_S - i_D & & \\ i_S = K_G(v_{GS} - V_T)^2 & v_{GS} > V_T, & \text{otherwise } i_S = 0 \\ i_D = K_G(v_{GD} - V_T)^2 & v_{GD} > V_T, & \text{otherwise } i_D = 0 \end{array}\right| \tag{7.6}$$

The equations supply at most one variable coefficient per row of the Jacobian, just as in the bipolar case. Moreover with eq.(7.6) an Ebers-Moll-like model can be drawn, see figure 7.3. To satisfy eq.(7.6) α_D and α_S both have to be chosen equal to one. Willson [7.5] excludes this case, although it does not prevent to derive a set of equations simular to the one given in eq.(1.10).(Note that $\alpha_S \equiv 1$ and $\alpha_D \equiv 1$ imply that the current through the gate connection is zero as it ought to be.)

The disadvantage of (7.6) is that the current i is the difference of two terms. If both are equal, i.e. $v_{DS} = 0$, then i should be zero. However, because of numerical inaccuracy the result may differ from zero. An estimate of this error can be made if the ranges of i_S and i_D are oonsidered. The range of i_S is equal to the range of i for

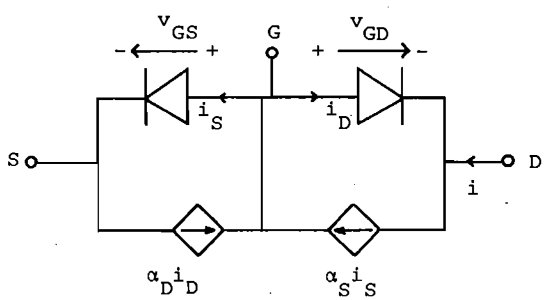

Fig. 7.3. Ebers-Moll-like model for a MOS transistor (n-channel).

$v_{GD} = V_T$. For usual gate voltages i never becomes larger than a few milli-amperes. Anyhow it is far less than 100mA. Such a value may easily be computed with an accuracy of 10nA, even with single precision on a small computer. If both i_S and i_D have this inaccuracy then the error in i is less than 20nA. Such an error is usually quite acceptable. Note that such an error in i appears only for $v_{GS} \approx v_{GD} > V_T$, i.e. when the transistor is conducting. For $v_{GS} \le V_T$ and $v_{GD} \le V_T$ the current i = 0 has no inaccuracy because both i_S and i_D are exactly zero.

Another important point is the range of the derivatives. From (7.6) it appears that the derivative $\frac{di_S}{dv_{GS}}$ becomes zero for $v_{GS} = V_T$. Its inverse, the derivative $\frac{dv_{GS}}{di_S}$, is bounded from below for usual values of the variables.

$$\frac{dv_{GS}}{di_S} = \frac{1}{2K_G(v_{GS} - V_T)} = \frac{1}{2\sqrt{K_G i_S}}$$

For $K_G = 2\text{mA/V}^2$ and $i_S = 50\text{mA}$ (both are large values) the derivative is 0.05V/mA. Often this value can be used as a lower bound to the value of $\frac{dv_{GS}}{di_S}$. The same holds for the derivative $\frac{dv_{GD}}{di_D}$. Whereas these derivatives recommend themselves as pivots, the derivatives $\frac{di_S}{dv_{GS}}$ and $\frac{di_D}{dv_{GD}}$ should be excluded to be pivot.

The following formula, which may supersede equation (7.3), takes into account the influence of the substrate ("B"). It contains a parameter K_B and the fermivoltage V_F:

$$\begin{aligned} i = {} & K_G[2(v_{GS} - V_T)v_{DS} - v_{DS}^2] - \\ & - K_B[(v_{DS} + v_{SB} + 2V_F)^{3/2} - (v_{SB} + 2V_F)^{3/2}] \end{aligned} \tag{7.7}$$

The equation holds for $0 \le v_{DS} \le v_{GS} - V_T$, $v_{DS} + v_{SB} + V_F \ge 0$ and $v_{SB} + 2V_F \ge 0$. In the associated row of the Jacobian it supplies three variable coefficients. However, like equation (7.3), equation (7.7) can be cast into a form which is more attractive for pivotstep skipping. The appropriate substitutions are $v_{DS} = v_{GS} - v_{GD}$ and $v_{DS} + v_{SB} = v_{DB}$:

$$
\left.\begin{array}{lll}
i = i_S - i_D + i_{SB} - i_{DB} & & \\
i_S = K_G(v_{GS} - V_T)^2 & v_{GS} > V_T, \text{ otherwise } i_S & = 0 \\
i_D = K_G(v_{GD} - V_T)^2 & v_{GD} > V_T, \text{ otherwise } i_D & = 0 \\
i_{SB} = K_B(v_{SB} + 2V_F)^{3/2} & v_{SB} > -2V_F, \text{ otherwise } i_{SB} & = 0 \\
i_{DB} = K_B(v_{DB} + 2V_F)^{3/2} & v_{DB} > -2V_F, \text{ otherwise } i_{DB} & = 0
\end{array}\right\} \quad (7.8)
$$

Equations (7.8) induce two Ebers-Moll models in parallel, as given in figure 7.4. Thus according to [7.5] a circuit can be obtained which can be described by an equation set of the form (1.10).

Equations (7.8) suggest that the inaccuracy of i for $v_{DS} = 0$ is doubled because i consists of four terms now. In fact the inaccuracy is hardly increased because the terms i_{SB} and i_{DB} are small compared to i_S and i_D. The parameter K_G is two orders or more larger than K_B.

The derivatives $\frac{dv_{SB}}{di_{SB}}$ and $\frac{dv_{DB}}{di_{DB}}$ are bounded from below for usual values of the substrate voltage. For instance:

$$
\frac{dv_{SB}}{di_{SB}} = \frac{2}{3K_B\sqrt{v_{SB} + 2V_F}} = \frac{2}{3\sqrt[3]{K_B^2 i_{SB}}}
$$

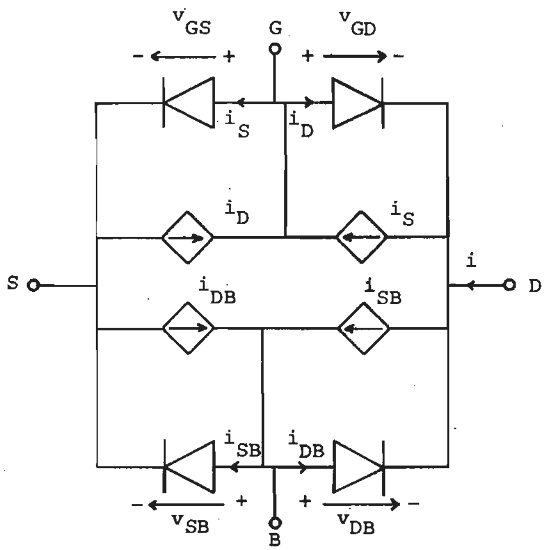

Fig.7.4. Ebers-Moll models for a MOS transistor if the influence of the substrate is taken into account.

With the comparatively large values $K_B = \frac{2}{3}10^{-2}\mathrm{mA/V}^2$ and $v_{SB} + 2V_F = 25V$ this derivative becomes 20V/mA. Often this lower bound suffices.

For junction gate field effect transistors (junction FET) equations of the same form as the last two equations of (7.8) can be derived. In fact the substrate of a MOS transistor can be considered as a junction gate. The parameter K_B is larger for a junction FET because of the construction of the transistor. Therefore the lower bound to the derivative $\frac{dv_{GS}}{di_S}$ of the FET is smaller than $\frac{dv_{SB}}{di_{SB}}$ of the MOS transistor but it is comparable with the lower bound of $\frac{dv_{GS}}{di_S}$ of the MOS transistor.

7.3. General nonlinear functions

Pivotstep skipping is applied if extreme high accuracy is not required. In that case simple transistor models are mostly sufficient. If in special cases an accurate transistor model is required then nonlinear equations depending on more than two variables cannot always be avoided. An example is the case that channel length shortening and mobility reduction for a MOS transistor have to be taken into account [7.4]. This section will show how pivotstep skipping can be applied to such equations.

Let the k^{th} equation, $s_k(x)$, be nonlinear and be dependant on three or more variables. One of the coefficients in the associated row of the Jacobian can be chosen equal to one, let this be the pivot: $a_{kk} \equiv 1$. Then the other coefficients are (compare eq.(1.2)):

$$a_{ki} = \frac{\partial s_k(x)}{\partial x_i} * \left(\frac{\partial s_k(x)}{\partial x_k} \right)^{-1} \qquad \text{for } i \neq k$$

Thus the k^{th} row of the Jacobian implies the equation:

$$a_{k1}z_1 + a_{k2}z_2 + \ldots + a_{k,k-1}z_{k-1} + z_k + a_{k,t+1}z_{t+1} + \ldots$$
$$\ldots + a_{kn}z_n = r_k \tag{7.9}$$

Assume the variables $\bar{z}_i = a_{ki}z_i$, $i = 1,\ldots,k-1,t+1,\ldots,n$, are introduced. Then equation (7.9) is equivalent to the following set of equations:

$$\left.\begin{array}{ll} -z_i + \dfrac{1}{a_{ki}}\,\bar{z}_i = 0 & i = 1,\ldots,k-1,t+1,\ldots,n \\ \displaystyle\sum_{i=1}^{k-1} \bar{z}_i + z_k + \sum_{i=t+1}^{n} \bar{z}_i = r_k & \end{array}\right\} \quad (7.10)$$

The BLT form of the Jacobian can be retained if (7.9) is replaced by (7.10) and the new variables $\bar{z}_i$ are inserted between z_{k-1} and z_k (see figure 7.5 for an example). It appears that all variable coefficients in (7.10) are put on the diagonal and the form attractive for pivotstep skipping is achieved.

The drawback is that the Jacobian is extended. However a further consideration shows that most of the new pivotsteps correspond to pivotsubsteps induced by the original Jacobian. For instance the skipping of the pivotstep associated with the new pivot $\frac{1}{a_{42}}$ in figure 7.5 is identical to the skipping of the pivotsubstep associated with the original coefficient a_{42}: in both cases all terms containing the factor a_{42} are deleted. A similar statement applies to the new pivot $\frac{1}{a_{43}}$. However the pivotstep associated with $\frac{1}{a_{47}}$ does not correspond to a pivotsubstep for the original matrix. Skipping of this pivotstep is identical to the deletion of a_{47} in the original case.

The conclusion is that it is not required to use the extended Jacobian in the actual computation of the solution. The original matrix can be used and pivotsubsteps associated to variable off-diagonal coefficients can be controlled by thresholds. The thresholds

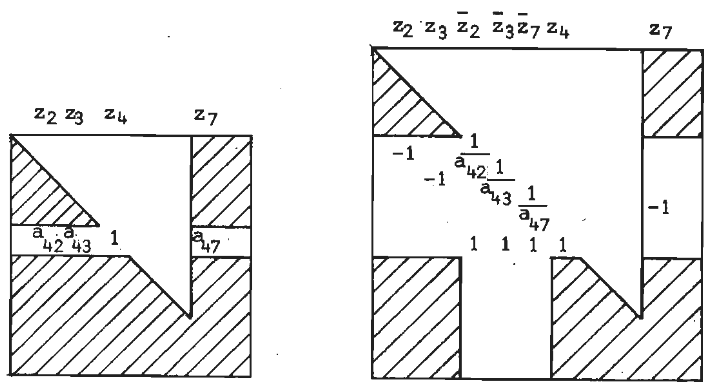

Fig. 7.5. The transformation of a nonlinear equation.

may be computed using the extended Jacobian. Equivalently the original matrix can be used as well. This will be illustrated by the following examples.

The examples concern the case depicted in figure 7.6. The first example is the computation of ε_{qr}. ε_{qr} determines the threshold θ_{qr} applying to the pivot $\frac{1}{a_{qr}}$: $\theta_{qr} = \Delta^{-1}\varepsilon_{qr}^{-1}$.

ε_{qr} is computed from R_{qr} and C_{qr}: $\varepsilon_{qr} = \max[R_{qr}, C_{qr}]$

$$R_{qr} = \min_{\substack{p \neq q \\ a_{pq} \neq 0}} \left[\frac{\varepsilon_{pr}}{|a_{pq}|} \right]$$

$$C_{qr} = \min_{\substack{w \neq r \\ a_{rw} \neq 0}} \left[\frac{\varepsilon_{qw}}{|a_{rw}|} \right] \eta_r$$

In the case $\varepsilon_{qr} = C_{qr}$ the condition $\left|\frac{1}{a_{qr}}\right| \geq \theta_{qr}$ is identical to:

$$|a_{qr}| \leq \theta_{qr}^{-1} = \Delta\varepsilon_{qr} = \Delta\eta_r \min_{\substack{w \neq r \\ a_{rw} \neq 0}} \left[\frac{\varepsilon_{qw}}{|a_{rw}|} \right]$$

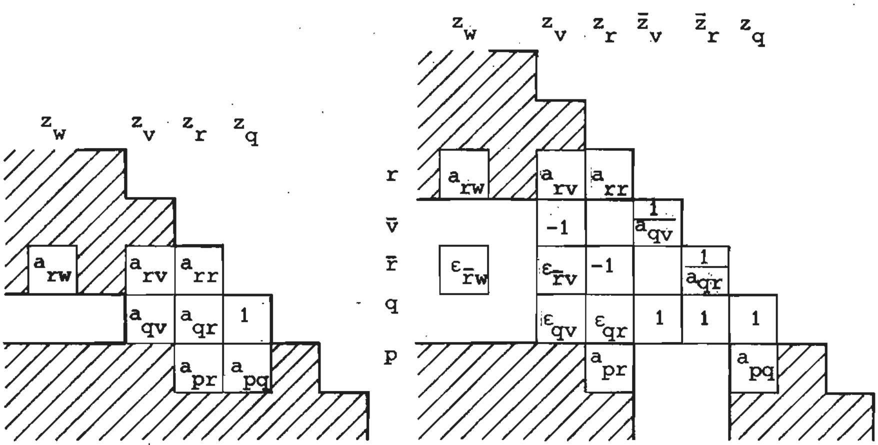

Fig. 7.6. A part of an original Jacobian (left) and the corresponding part of the extended Jacobian (right).

This condition corresponds to the requirement that the terms $\psi_{qw} = -a_{qr}a_{rr}^{-1}a_{rw}$ in the original matrix for all w satisfy $|\psi_{qw}| \le \Delta\varepsilon_{qw}$. Apparently the variable pivot a_{rr} is replaced by its lower bound η_r and the variable coefficient a_{qw} is replaced by the associated dependable reference ε_{qw}.

The case $\varepsilon_{qr} = R_{qr}$ implies that $|a_{qr}| \le \theta_{qr}^{-1}$ is satisfied if the terms $\psi_{pr} = -a_{pq}a_{qq}^{-1}a_{qr} = -a_{pq}a_{qr}$ satisfy $|\psi_{pr}| \le \Delta\varepsilon_{pr}$ for all p.

Another example is the computation of θ_r. The difference with the cases discussed in chapter 4 is that the cycles in the original matrix which are used in the computation of the threshold may contain more than one variable coefficient. An example is the cycle $C_{qv} = \{a_{qr}, a_{rr}, a_{rv}, a_{qv}\}$. Let in the extended Jacobian the columns associated with $\bar{z}_v$ and $\bar{z}_r$ have the numbers $\bar{v}$ and $\bar{r}$ respectively. In the same way the row numbers $\bar{v}$ and $\bar{r}$ are introduced (see figure 7.6). Let θ_r be determined by the condition $|-1 \cdot a_{rr}^{-1} \cdot a_{rv}| \le \Delta\varepsilon_{\bar{r}v}$ emanating from the extended matrix. We have $\varepsilon_{\bar{r}v} = \max[R_{\bar{r}v}, C_{\bar{r}v}]$ and

$$C_{\bar{r}v} = \min_{\substack{w \ne v \\ a_{vw} \ne 0}} \left[\frac{\varepsilon_{\bar{r}w}}{|a_{vw}|}\right]\eta_v$$

and with η_{qr}, the upper bound of a_{qr}

$$R_{\bar{r}v} = \varepsilon_{qv}\eta_{qr}^{-1}$$

In the case $\varepsilon_{\bar{r}v} = R_{\bar{r}v}$ the threshold θ_r is

$$\theta_r = \Delta^{-1}\,\frac{|a_{rv}|}{\varepsilon_{qv}}\,\eta_{qr}$$

The condition $|a_{rr}| \ge \theta_r = \Delta^{-1}\,\dfrac{|a_{rv}|}{\varepsilon_{qv}}\,\eta_{qr}$ corresponds to the condition obtained from the cycle $C_{qv} = \{a_{qr}, a_{rr}, a_{rv}, a_{qv}\}$ in the original matrix. For the variable coefficients the value such that the condition becomes the strongest is assumed: $a_{qv} = 0$ (so ε_{qv} is used) and $a_{qr} = \eta_{qr}$.

In the case $\varepsilon_{\bar{r}v} = C_{\bar{r}v}$ we have $C_{\bar{r}v} = \dfrac{\varepsilon_{\bar{r}w}}{|a_{vw}|}\cdot\eta_v$ for some w. On its turn $\varepsilon_{\bar{r}w}$ may be equal to $C_{\bar{r}w}$ and so be computed from some $\varepsilon_{\bar{r}i}$ in row $\bar{r}$. Because all $a_{\bar{r}i}$ are zero (except for the trivial values i=r and i=$\bar{r}$) sooner or later we will find some $\varepsilon_{\bar{r}j} = R_{\bar{r}j}$. Anyhow $\varepsilon_{\bar{r}j} = R_{\bar{r}j}$ for all j, $t+1 \le j \le n$. Consequently some $i \ge 1$ exists such that

$$\varepsilon_{\bar{r}j} = R_{\bar{r}j} = \varepsilon_{qj}\eta_{qr}^{-1} \tag{7.11}$$

holds for $1 \le j < i$ and $t+1 \le j \le n$. Then for $C_{\bar{r}i}$ follows:

$$C_{\bar{r}i} = \min_{\substack{j\neq i\\ a_{ij}\neq 0}} \left[\frac{\varepsilon_{\bar{r}j}}{|a_{ij}|}\right]\eta_i = \min_{\substack{j\neq i\\ a_{ij}\neq 0}} \left[\frac{\varepsilon_{qj}}{|a_{ij}|}\right]\eta_i\eta_{qr}^{-1} = C_{qi}\eta_{qr}^{-1}$$

Because the definition of ε_{qi} implies

$$R_{\bar{r}i} = \varepsilon_{qi}\eta_{qr}^{-1} \ge C_{qi}\eta_{qr}^{-1} = C_{\bar{r}i}$$

it appears that $\varepsilon_{\bar{r}i} = R_{\bar{r}i}$ and (7.11) holds for j=i too. By induction on i follows $\varepsilon_{\bar{r}v} = R_{\bar{r}v} = \varepsilon_{qv}\eta_{qr}^{-1}$ and this case has already been discussed previously.

If for the variable off-diagonal coefficients the associated upper bounds and dependable references are appropriately exploited, thresholds for these variable coefficients can be derived. Dependable references must be computed for each of these coefficients (eventual lower bounds suffice as well). If such a variable coefficient occurs in a function C_{ij} or R_{ij} the associated upper bound has to be used (it occurs in the function "min" in the denominator). Like the thresholds for the pivots the thresholds for the off-diagonal coefficients are computed from a set of cycles. θ_{qr} is computed from all cycles of length one passing through a_{qr} and a_{qq} ($\equiv 1!$) or all cycles of length one passing through a_{qr} and a_{rr}. If in a cycle which is used in the computation of the threshold of some coefficient another variable coefficient occurs, then for the latter coefficient the dependable reference or the upper bound has to be used in the condition associated with the cycle, such that this condition becomes the strongest.

[7.1] L.W. Nagel, "SPICE2: a computer program to simulate semiconductor circuits", Electronics Res. Lab., Univ. California, Berkeley, Memo ERL-M520, 1975.

[7.2] R.K. Brayton, F.G. Gustavson, G.D. Hachtel, "A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formulas", Proc. IEEE, Vol. 60, pp. 98-108, (1972).

[7.3] J.C. Butcher, "Implicit Runge-Kutta processes", Math. Comp., Vol. 18, pp. 50-64, (1964).

[7.4] R.S.C. Cobbold, "Theory and applications of field-effect transistors", Wiley & Sons, New York, 1970.

[7.5] A.N. Willson, "New theorems on the equations of nonlinear DC transistor networks", Bell Syst. Techn. J., Vol. 49, pp. 1713-1738, (1970).

CONCLUSIONS

Methods are given for the analysis of an electrical circuit in order to obtain a macromodel. The macromodel permits a transient simulation which is faster than the simulation based on the original circuit description.

The methods comprise an algorithm to order the variables and equations of the circuit such that the Jacobian of the equation set assumes a BLT form. The application of linear transformations to the set of linear equations has a substantial significance if we want to obtain a small border.

Pivotstep skipping exploits the BLT form in a natural way. The computation of the thresholds can easily be executed by a computer. The dependable references, used instead of zero matrix coefficients, fit exactly into this computation. An algorithm to obtain dependable references is presented. Hence a macromodel of a circuit can be obtained completely automatically.

The accuracy achieved with pivotstep skipping can be controlled with the value of the factor Δ. As the thresholds all are proportional to Δ^{-1} the accuracy can be adapted rather easily during the actual simulation. The upper bounds to the error of the solution show that this error can be made arbitrarily small by choosing an appropriate value of Δ.

The speed up obtained by two strongly differing implementations of pivotstep skipping is studied. The speed up of around a factor 2 is slightly disappointing. If only type 6 pivotsteps are left in the macromodel the speed up lies between 1.7 and 3.6 for the circuits studied.

The harmful influence of pivotstep skipping on the convergence of the Newton iteration process, which can be expected, seems to be relevant only if Δ becomes of the order of 0.1 to 1. For smaller values of Δ the convergence is sometimes worse and sometimes better than for a computation without pivotstep skipping.

Pivotstep skipping is most appropriate for circuits with bipolar transistors and field effect transistors. As also variable off-diagonal coefficients in the Jacobian can be dealt with, pivotstep skipping is generally applicable.

APPENDIX A

The size of dependable references

The object of this appendix is to get a general idea of the size of the dependable references. Therefore some general suppositions are made concerning the size of matrix coefficients and the size of the lower bounds of the pivot values. It is supposed that:

i) all matrix coefficients have value μ,

ii) all lower bounds to the pivot values have value η.

If μ and η are appropriately chosen, at will a lower bound, upper bound or mean value of the dependable references is obtained.

Now if ε_{ij} is a dependable reference then $\varepsilon_{ij} = \mu$ if $\varepsilon_{ij} = |a_{ij}|$ or $\varepsilon_{ij} = \varepsilon_{kj}\ (\eta/\mu)$ or $\varepsilon_{ij} = \varepsilon_{i\ell}(\eta/\mu)$. The same holds for ε_{kj} and $\varepsilon_{i\ell}$. Hence we have $\varepsilon_{ij} = \mu(\eta/\mu)^m$ for some m. Let the exponent m be called the level of the dependable reference. Dependable references of level m are $(\eta/\mu)^m$ times as small (or large) as original matrix coefficients. The factor (η/μ) and the maximum level of the dependable references are important.

In usual electrical circuits μ is of the order 10^3 to $10^4\Omega$. The order of the lower bounds η depends on the range of the values of the circuit variables. In section 1.4 we computed a lower bound $\eta = 10\Omega$ (eq.(1.11)) for bipolar circuits. Then we have $\eta/\mu = 10^{-2}$ to 10^{-3}. If the current through a diode does not exceed the value of 25 μA then it appears that $\eta = 10^3\Omega$. So if $\mu = 10^3\Omega$ holds as well we obtain $\eta/\mu = 1$. In this case all ε_{ij}, independent of their level, are of the same order as the matrix coefficients. In operational amplifiers the (DC) currents in the input stage are very low. Then it is attractive to make use of this fact to obtain lower bounds to the pivot values which are as high as possible.

The level of the dependable references does not exceed one for small circuits (logical gates) and does not exceed three for larger circuits (operational amplifiers, flip-flops).

If the dependable references are computed according to eq.(4.12) the dependable references are smaller. The numbers ν_{ij} are of the order 10^1 (some extreme values of ν_{ij} are 36 for the μA 709 and 35 for the μA 741). This means that all ε_{ij} are at least one order smaller compared with the case that ε_{ij} is computed according to eq.(4.11). However if the level of ε_{ij} is m then ε_{ij} is (m+1) orders smaller.

APPENDIX B

Comparison of the compiled code, the interpretable code and the linked list approach

For each method an implementation is given in MACRO-11 for a PDP 11/60 computer with a fast floating point processor. The expected execution time and the storage requirements are computed and listed in a table.

Compiled code

A typical part of the list of instructions is COMCODE:

COMCODE

	instruction		*time*	*storage*	*comment*
1	LDF PIVOT,Fϕ		13	4	load Fϕ with pivot value;
2	LDF COEF1,F1		13	4	put border coefficient into F1;
3	DIVF Fϕ,F1		38	2	divide coefficient by pivot;
4	STF F1,COEF1		23	4	store quotient;
5	NEGF F1	t	4	2	negate quotient;
6	LDF F1,F2	ρ_{12}	4	2	copy (-quotient) into F2;
7	MULF COEF2,F2	κ_{*1}	13	4	multiply F2 by coefficient from pivot column;
8	ADDF COEF3,F2		13	4	add value of coefficient in border to product;
9	STF F2,COEF3		23	4	store sum;

(Fϕ,F1,F2 are registers of the floating point processor ; κ_{*1},ρ_{12} and t are defined in section 5.2).

COMCODE contains the loading of the floating point processor with the pivot value (line 1), the division of one border coefficient by the pivot value (lines 2 to 5) and the multiplication of this quotient with one coefficient from the pivot column (lines 6 to 9). To obtain the total code the part indicated by κ_{*1} need be repeated κ_{*1} times (for each coefficient in the pivot column). Further the part indicated by ρ_{12} need be repeated ρ_{12} times (for each coefficient in the border part of the pivot row) and finally the total must be repeated t times (for each pivot outside the border). Note the nested structure of the repetition. The column "time" contains the execution times of the instructions. The times are in units of 170 nsec. The time specifications are obtained from [B.1]. The column "storage" contains the storage requirements of the instructions, these are given in bytes.

From the listing a formula for the total time to execute the pivotsteps can be obtained:

$$T_{CC} = t\{13 + \rho_{12}(78+53\kappa_{*1})\}$$

The formula for the total storage requirement of the code is

$$M_{COMMODE} = t\{4 + \rho_{12}(12+14\kappa_{*1})$$

Besides the numerical values of the matrix coefficients require storage. If one coefficient uses 8 bytes, then the formula for this storage requirement is:

$$M_{NUM} = 8\{t(1+\kappa_{*1}) + b(\kappa_{12}+\rho_{22})\}$$

The total storage requirement is

$$M_{CC} = M_{COMCODE} + M_{NUM}$$

Interpretable code

In this approach the operations are coded in a very uniform way. Each instruction consists of one operation code and a fixed number of addresses. In this appendix three different instructions are used. They are listed in table B.1. Note that two instructions use only one address. The last two instructions use registers of the floating point processor; the contents of such a register must be determined by preceding operations. The instructions can be interpreted by the program INTERPRET.

TABLE B.1.

instruction			*interpretation*
LOAD	ARG1	ARG2	COEF(ARG1) is put into FΦ
DIVIDE	ARG1	ARG2	COEF(ARG1) ← COEF(ARG1)/FΦ
MULTIPLY	ARG1	ARG2	COEF(ARG2) ← COEF(ARG2) - COEF(ARG1)*F1

INTERPRET

	instruction			*time*	*storage*	*comment*
	LOOP:					
1	INC	Rϕ	$t + t\rho_{12} + t\rho_{12}\kappa_{*1}$	2	4	Rϕ points to an instruction in the list;
2	MOV	ARG1(Rϕ),R1		8	4	put ARG1 (alway used) into R1;
3	JMP	OPCODE(Rϕ)		8	4	jump to line 4, 6 or 10 dependent on the operation code;
	LOAD:					
4	LDF	COEF(R1),Fϕ	t	13	4	put value of pivot into Fϕ;
5	BR	LOOP		5	4	go back to line 1;
	DIVIDE:					
6	LDF	COEF(R1),F1	$t\rho_{12}$	13	4	put border coefficient into F1;
7	DIVF	Fϕ,F1		38	2	divide border coefficient by pivot;
8	STF	F1,COEF(R1)		23	4	store quotient;
9	BR	LOOP		5	4	go back to line 1;
	MULTIPLY:					
10	LDF	F1,F2	$t\rho_{12}\kappa_{*1}$	4	2	copy quotient into F2;
11	MULF	COEF(R1),F2		13	4	multiply quotient by coefficient from pivot column;
12	NEGF	F2		4	2	negative product;
13	MOV	ARG2(Rϕ),R1		8	4	put ARG2 into R1;
14	ADDF	COEF(R1),F2		13	4	add border coefficient to (-product);
15	STF	F2,COEF(R1)		23	4	store sum;
16	BR	LOOP		5	4	go back to line 1.

(Rϕ,R1 are registers of the CPU; Fϕ,F1,F2 are registers of the floating point processor)

The coefficients κ_{*1}, ρ_{12} and t indicate how often parts of the program are executed. (Unlike before they do not indicate that instructions of the program are repeated. The whole program INTERPRET consists of the 16 given lines). Lines 1-3, the decoding of the operation code, are executed for each instruction.

The total execution time is

$$T_{IC} = t\{36 + \rho_{12}(97+88\kappa_{*1})\}$$

Each instruction consists of 5 bytes, one byte for the operation code and two bytes for each argument. The storage requirement of the instructions is therefore:

$$M_{INSTR} = 5t\{1 + \rho_{12}(1+\kappa_{*1})\}$$

(For each pivotstep the pivot value is loaded, ρ_{12} divisions and $\kappa_{*1}\rho_{12}$ multiplications are executed.) The interpreting program requires only 56 bytes. The storage requirement for the numerical values is the same as for the compiled code approach, so the total storage requirement is

$$M_{IC} = M_{INSTR} + M_{NUM} + 56$$

<u>Linked list approach</u>

The implementation of this approach is given in section 5.4 (program LINKLIST). The execution time is:

$$T_{LL} = t\{49 + 87\rho_{12} + \kappa_{*1}(40+\rho_{12}[66+29\phi])\}$$

The storage requirement for the pointer arrays is:

PNXTAC:	$2(t+1)$
RWNEXT:	$2t(\rho_{12}+1)$
CNXTAC:	$2t(\kappa_{*1}+1)$
ROWPIV:	$2t\kappa_{*1}$
COLUMN:	$\underline{2t\rho_{12} + 2b\rho_{22}}$

total:

$$M_{POINT} = 2t(2\kappa_{*1}+2\rho_{12}+3) + 2b\rho_{22} + 2$$

The program requires 88 bytes and the numerical values require the same amount of storage as before, so in all we have:

$$M_{LL} = M_{POINT} + M_{NUM} + 88$$

Table B.3 shows the amount of time and storage required by these methods for a few circuits. The parameters of these circuits are shown in table B.2.

TABLE B.2

	n	t	b	ρ_{12}	ρ_{22}	κ_{*1}	κ_{12}	ϕ
TTL NAND	18	13	5	2.54	4.40	4.92	6.60	1.45
ECL OR/NOR	16	12	4	1.92	3.00	3.08	5.75	1.58
ECL flip-flop	41	30	11	3.77	7.91	7.27	10.27	2.54
μA 709	38	29	9	3.69	6.78	9.03	11.89	1.56
μA 741	47	35	12	3.43	6.92	7.66	10.00	1.77

TABLE B.3

	compiled code		interpretable code		linked list	
	T_{CC}	M_{CC}	T_{IC}	M_{IC}	T_{LL}	M_{LL}
TTL NAND	11350	3780	17970	2150	23635	1660
ECL OR/NOR	5710	1990	8900	1260	11990	1010
ECL flip-flop	52700	16600	84300	8460	134720	5350
μA 709	60000	18600	96500	9240	128800	5530
μA 741	58500	18500	93800	9470	130800	6070

From table B.3 it appears that the ratio T_{IC}/T_{CC} hardly depends on the sort of circuit and is not so large as reported in [B.2]. The ratios M_{IC}/M_{CC} and M_{LL}/M_{CC} are smaller for larger circuits.

[B.1] PDP 11/60 processor handbook, Digital Equipment Corporation (1977).

[B.2] H.B. Lee, "An implementation of Gaussian elimination for sparse systems of linear equations", in Sparse Matrix Proceedings (R.A. Willoughby, Ed.), Yorktown Heights, N.Y., IBM report RA1 (#11707), pp. 75-83 (1969).

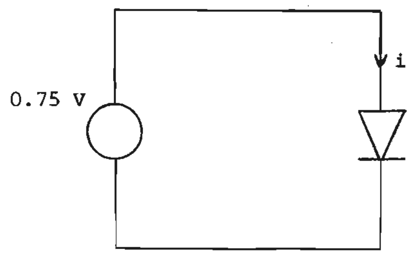

Fig.C.1. Simple circuit to illustrate slow convergence.

TABLE C.1.

step	i + Is	v
0	0.42483543E-29	-1.00000000
1	0.30163313E-27	-0.89343297
2	0.20130186E-25	-0.78841388
3	0.12588725E-23	-0.68502009
4	0.73519167E-22	-0.58333689
5	0.39945521E-20	-0.48345834
6	0.20107912E-18	-0.38548917
7	0.93340061E-17	-0.28954616
8	0.39745920E-15	-0.19576046
9	0.15433508E-13	-0.10428035
10	0.54281520E-12	-0.15274659E-01
11	0.17158924E-10	0.71062960E-01
12	0.48315207E-09	0.15450829
13	0.11991674E-07	0.23479919
14	0.25911649E-06	0.31162584
15	0.48027159E-05	0.38461733
16	0.74995885E-04	0.45332360
17	0.96497627E-03	0.51719034
18	0.99512078E-02	0.57552397
19	0.79401098E-01	0.62744445
20	0.46864292	0.67182767
21	1.9340392	0.70726579
22	5.2400246	0.73218369
23	8.9743395	0.74563473
24	10.541354	0.74965823
25	10.685464	0.74999762
26	10.686483	0.75000006
27	10.686458	0.74999994
28	10.686483	0.75000006
29	10.686458	0.74999994
30	10.686483	0.75000006

Newton iteration applied to an exponential function

Let the diode in figure C.1 be described by the equation:

$$i = I_s\{\exp\left(\frac{v}{V_T}\right) - 1\}$$

v is the diode voltage, i is the diode current; V_T and I_s are parameters. With $V_T = \frac{1}{40}$ V and $I_s = 10^{-12}$ nA we obtain the equation

$$0.75\ \mathrm{V} = v = \frac{1}{40}\ln(1+10^{12}i)$$

Let the solution i be determined by Newton iteration starting with the value i^o corresponding with a reversed biased diode voltage v^o of - 1V. After one iteration only a minor improvement is obtained. Only after more than 20 iterations the approximation may be accurate enough, see table C1. Apparently the use of i as controlling variable has two disadvantages. Firstly the convergence is very slow, secondly accuracy problems arise if i is stored instead of $(i+I_s)$.

NOTATIONS

A	Jacobian matrix
C_{ij}	function to compute a dependable reference
D	diagonal matrix
L	lower matrix of L\U decomposition
R_{ij}	function to compute a dependable reference
T	transpose indicator
U	upper matrix of L\U decomposition
a	coefficient of A
b	border width
d	coefficient of D
h	time step
i	sections 1.4, 6.3, 7.1, 7.2 : vector of currents
l	coefficient of L
n	number of variables, number of equations
q	section 7.1 : charge
r	sections 1.1, 1.5, 4.6, chapter 6 : residual vector
s	vector of vectorfunctions
t	dimension of A_{11}
	section 1.1 : time
u	coefficient of U
v	sections 1.5, 6.7, 7.1, 7.2 : vector of voltages
x	n-vector of variables
y	n-vector, result of foresubstitution
z	n-vector, result of backsubstitution
$\mathcal{B}$	bipartite graph
$\mathcal{C}$	cycle
$\mathcal{G}$	(directed) graph
$\mathcal{L}$	set of indices associated with linear functions
$\mathcal{N}$	set of indices : {1,2, ..., n}
$\mathcal{P}$	path
$\mathcal{Q}$	partial train
$\mathcal{R}$	partial train
$\mathcal{S}$	set of vector functions
$\mathcal{T}$	train
$\mathcal{V}$	kernel
$\mathcal{W}$	kernel

X	set of variables
Y	set of variables
Z	set of variables
Δ	accuracy factor
Φ	transformation matrix
Ψ	set of terms
α	parameter in integration formula
β	parameter in integration formula
γ	$-\sum_{i \neq q} a_{qi} x_i$: linear part of nonlinear function
δ	perturbation indicator
ε	dependable reference
η	lower bound pivot value
θ	threshold
ι	iteration number
κ	mean number of coefficients in a column
μ	pivot variability
ν	number of nonzero coefficients
π	pivot activity
ρ	mean number of coefficients in a row
ψ	term
$\lVert x \rVert$	$\lVert x \rVert_\infty = \max_i \lvert x_i \rvert$
$\lVert A \rVert$	$\lVert A \rVert_\infty = \max_i [\sum_{j=1}^{n} \lvert a_{ij} \rvert]$
$\lvert A \rvert$	the matrix obtained from **A** by taking the absolute value of each coefficient of **A**.
$X \backslash Y$	difference of sets : $\{ x \mid x \in X \wedge x \notin Y \}$
$X \oplus Y$	symmetric difference of sets : $X \backslash Y \cup Y \backslash X$
$[c_1, c_2]$	set of real values: $\{ x \mid c_1 \leq x \leq c_2 \}$

STELLINGEN

bij het proefschrift van

P.M. Trouborst

9 juni 1981

Technische Hogeschool Eindhoven

1. Is het terecht om de mogelijkheid te onderzoeken om de "blokdecompositie" van een matrix te bepalen zonder een "transversal" te gebruiken, de karakterisering van een irreducible matrix die Kevorkian voorstelt is op zich onvoldoende voor een dergelijke methode.

   A.K. Kevorkian, "Graph-theoretic characterization of the matrix property of full irreducibility without using a transversal", J. of Graph Theory, Vol. 3, pp.151-174 (1979).

2. De matrix die Hsieh en Ghausi gebruiken bij het vergelijken van verschillende methoden voor het verkrijgen van een pivot volgorde, is niet representatief. Een structuuranalyse van deze matrix (die sneller is dan de genoemde methoden) kan de optimale pivot volgorde vrijwel geheel leveren. In de twijfelgevallen die de structuuranalyse overlaat, wordt door alle beschouwde methoden de optimale pivot volgorde bepaald.

   H.Y. Shieh, M.S. Ghausi, "On optimal-pivoting algorithms in sparse matrices", IEEE Trans. Circuit Theory (Corresp.), Vol. CT-19, pp. 93-96 (1972).

3. Bij het ontwerpen van een grote elektronische schakeling dient niet alleen de testbaarheid van de schakeling bevorderd te worden, maar ook de mogelijkheid de schakeling binnen acceptabele tijd te simuleren. Voor beide oogmerken is het van belang de schakeling volgens een hiërarchische structuur uit kleinere onderdelen op te bouwen.

4. Het absoluut optimaliseren van kwaliteitsfaktoren van complexe elektronische systemen zoals snelheid, betrouwbaarheid, nauwkeurigheid, dissipatie en produktiekosten, is ondoenlijk.

   R.K. Brayton, R. Spence, "Sensitivity and optimization", Elsevier, Amsterdam, 1980, p. 195.

5. Voor vereenzaming en vervreemding kan een belangrijke oorzaak gezocht worden in de toenemende fysieke onafhankelijkheid van de mensen, welke mogelijk is geworden door de ontwikkeling van de techniek.

   J. Reese, e.a., "Gefahren der informationstechnologischen Entwicklung", Campusverlag, Frankfurt 1979, p. 62 vv.

6. Zij die de gezegden "Abraham zien" of "Van Pontius naar Pilatus sturen" bezigen, geven daarmee niet te kennen te weten waar de klepel hangt.