INTERNATIONAL DESIGN CONFERENCE - DESIGN 2006
Dubrovnik - Croatia, May 15 - 18, 2006.

DESIGN 2006

A CONTRIBUTION TO METHODOLOGY OF ENGINEERING CALCULATIONS

R. Rohatyński and W. Babirecki

Keywords: engineering calculation, algorithms, occurrence matrices

1. Introduction

This paper deals with composing algorithms of engineering calculations based on constraint management. The constraints are depicted in form of a Boolean matrix called an occurence matrix.
The simplicity of occurence matrices makes them particularly suited for computer implementation in contrast to graphs or other ways of representation [Warfield 1973].
In calculation of mechanical elements and assemblies the constraints form sparse matrices, which should be reordered and decomposed. Several researchers have addressed the problem of selecting input variables and solving the resulting set of equations. Steward [Steward 1965] was one of the earliest investigators who analyzed the structure of simultaneous equations. Friedman and Leondes [Friedman and Leondes 1969] presented a constraint theory based on mathematical operations. Serrano and Gossard [Serrano and Gossard 1988] contributed to constraint management in mechanical computer-aided design. Kusiak and Cheng [Kusiak and Cheng 1990] presented algorithms for clustering the matrices for decomposition. Eppinger [Eppinger 1991] applied structural matrices to design task organization. Aggraval et al. [Aggraval et al. 1993] presented constraint management algorithms based on the decomposition of the occurence matrices. Recently Babirecki [Babirecki 2005] proposed similar approach to rationalization engineering calculations based on operations on the occurrence matrices.
An engineering calculation task is defined by a valid set of constraints and input variables such that the remaining unknown variables can be calculated. Constraints mean here generalized relations between variables. The number of theoretically possible tasks depends on the number of constraints and the number of variables. If these numbers are equal to each other then the matrix is square and there is only one possible task. Usually in the original occurrence matrix there are more variables than constraints. Thus, if the matrix is rectangular of (m x n) dimension, with $(n – m) > 0$, then $(n – m)$ variables must be specified as input. For a given number of constraints the number of possible tasks increases very fast with the number of variables. For example, for 5 constraints and 6 variables there are 6 different tasks possible, for 7 variables there are 15 tasks, and for 8 variables there exist potentially as many as 56 tasks. In reality, the number of feasible task cases is smaller because certain combinations of variables do not make semantic sense.

2. Association the system of constraints with an occurrence matrix

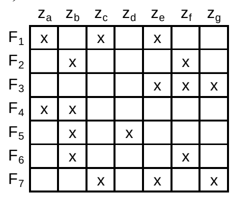
The *occurrence matrix R* for a system of n constraints with v variables is an n x v matrix $[s_{ij}]$ where s_{ij} is a mark if variable v_j appears in relation r_i or is blank otherwise. So the $[s_{ij}]$ matrix maps the set of variables onto the set of relations. The structure of a system of relations takes into consideration only which variables occur in which relations, but not how they appear. In this paper, on the contrary to the cited references, the occurrence matrix may represent any kind of relations: equations, inequalities, tabular data, and graphs, either linear or non-linear ones.

Fig. 1 shows a system of relations and the corresponding structural matrix. Relations are written in a normalized neutral form i.e. without distinction between output (dependent) and input (independent) variables. This is called *the neutral matrix.*

a)

$F_1(z_a, z_c, z_e)=0$
$F_2(z_b, z_f)=0$
$F_3(z_e, z_f, z_g)=0$
$F_4(z_a, z_d)=0$
$F_5(z_b, z_d)=0$
$F_6(z_b, z_f)=0$
$F_7(z_c, z_e, z_g)=0$

b)



	z_a	z_b	z_c	z_d	z_e	z_f	z_g
F_1	x		x		x		
F_2		x				x	
F_3					x	x	x
F_4	x	x					
F_5		x		x			
F_6		x				x	
F_7			x		x		x

Figure 1. a) System of relations b) Corresponding occurrence matrix

Formulation the neutral occurence matrix is the first step. The original matrix as shown in Fig. 1 can be reorder to a more suitable form. This is the subject of the next section.

3. The analysis and decomposition of occurrence matrices

The next step in the algorithm design is to rearrange the matrix to a set of sub-matrices. The process of decomposition begins with selecting any row of the matrix and drawing vertical lines through each mark in this row. Then rows with marks appearing in the vertical lines are drawing and so forth until no once-crossed marks are left. Now the twice-crossed marks are rearranged to form the sub-matrix (block). Next, all the twice-crossed marks are removed and the procedure is repeated until no marked cells are left. An example of the procedure is shown in Fig. 2.

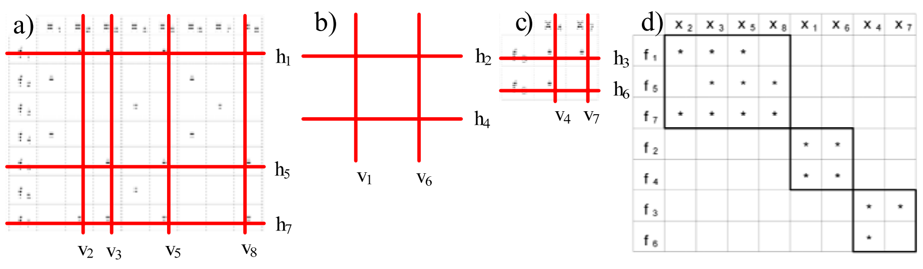


Figure 2. Decomposition of the occurence matrix: a) b) and c) - subsequent iterations, d) -resulting matrix decomposed into three independent blocks

It is not always possible to decompose a matrix into a set of independent blocks. In general, three structures can appear as shown in Fig. 3.

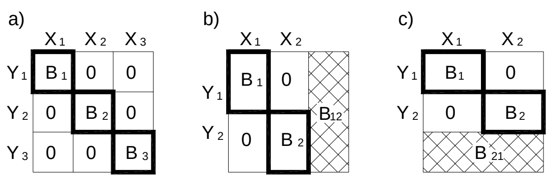


Figure 3. Three possible forms of the occurence matrix after reordering: a) the matrix fully decoupled into independent blocks, b) the matrix partially decoupled into blocks with overlapping variables, c) the matrix partially decoupled into blocks with overlapping constraints

The three blocks in Fig. 3a represent three independent groups of constraints which can be considered and solved simultaneously. The matrix shown in Fig. 3b is divided into two sub-matrices linked by the overlapping variables shown in the right-hand block. Because of these variables the sub-matrices are interdependent and can not be considered simultaneously unless an additional action is provided.

The matrix shown in Fig. 3c has also two independent blocks of constraints but there are also constraints out of the blocks shown at the bottom of the matrix. The solution of these blocks depends on the solution of the bottom constraints. To deal with overlapping constraints a semantic insight is required into the nature of the constraints.

For not fully decomposable matrices like in Figs. 3b and c a number of actions can be taken. The simplest one is to identify the overlapping variables or overlapping constraints and remove them from the matrix. For example, one can remove from the matrix the variable that has the greatest value of the dependence index *id*. This index is calculated for a chosen variable as a number of variables appearing in constraints which contain the variable. After having *id* calculated for all the variables, the variable of the highest index is removed out of the matrix and then the decomposition process proceeds. It means that the removed variable should become an input variable, i.e. the independent one. Kusiak and Cheng proposed a branch-and-bound algorithm to deal with such kind of matrices.

4. The derivation of calculation algorithms from occurrence matrices

The search for a calculation algorithm from the occurence matrix begins with the determination of input variables. It changes the neutral matrix into the directed one. The input variables can be introduced one at a time or at once. It is convenient to separate the specified input variables from the original matrix as it is shown in Fig. 6. When the inputs are determined one after the other the occurrence matrix must be updated step by step by removing columns with the specified variables. After each operation the resulting matrix should be analyzed and reordered. If all input variables are introduced at once then these intermediate operations are not performed. The first alternative may give better insight into systemic relations of the matrix but in the real use designers often prefer the second one. In the following only this approach is presented.

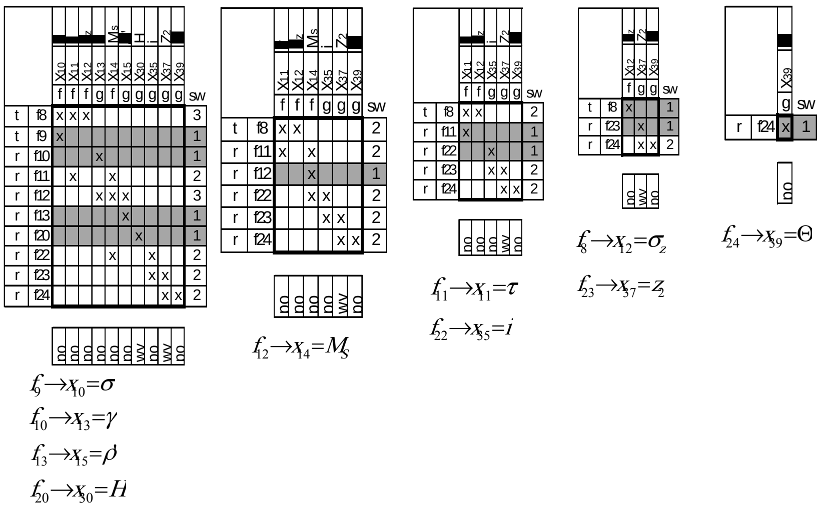

$f_9 \rightarrow x_{10} = \sigma$
$f_{10} \rightarrow x_{13} = \gamma$
$f_{13} \rightarrow x_{15} = \rho$
$f_{20} \rightarrow x_{30} = H$
$f_{12} \rightarrow x_{14} = M_s$
$f_{11} \rightarrow x_{11} = \tau$
$f_{22} \rightarrow x_{35} = i$
$f_8 \rightarrow x_{12} = \sigma_z$
$f_{23} \rightarrow x_{37} = z_2$
$f_{24} \rightarrow x_{39} = \Theta$
sw

Figure 4. Process of constructing of an calculation algorithm

If the occurrence matrix has been fully decomposed into blocks then for each block exists a separate algorithm of calculations. The process of algorithm construction starts with summing the entries in each row of the matrix to form a column for the row indexes *sw*. The row index indicates how many dependent (output) variables must be find in order to satisfy the constraint in a row. If the row index is equal to 1 then the constraint can be solved independently. Solving this constraint means that one operation of the being constructed algorithm has been determined. Then the occurrence matrix is updated and the procedure continues. If there is no one row with only one entry, this implies that a procedure of identifying simultaneous equations should be performed. This procedure continues until the *sw* column is identically equal to 0 i.e. there are no more dependent variables remaining, which means that the algorithm is complete.

Inequalities require a special attention. They can not be used for identification of the value of a particular dependent variable. They must be checked as soon as possible or convert into equalities.
In an example shown in Fig. 4 the original matrix and all the subsequent ones contain rows with at least one dependent variable, which makes the algorithm easy to find. In the original matrix there are four rows with *sw* index equal to one. After crossing out the rows with these constraints and the column with just calculated variables the subsequent matrix appears, in which there are six constraints and one among them has again one unknown variable. Thus the next matrix results and the operation repeats. The inscriptions in the Fig. 4 denote constraints that are solved and resulting variables. The resulting algorithm can be drawn in any useful form e.g. a graph, a flow-chart or other. This example was taken from a jackscrew calculation [Babirecki 2005].

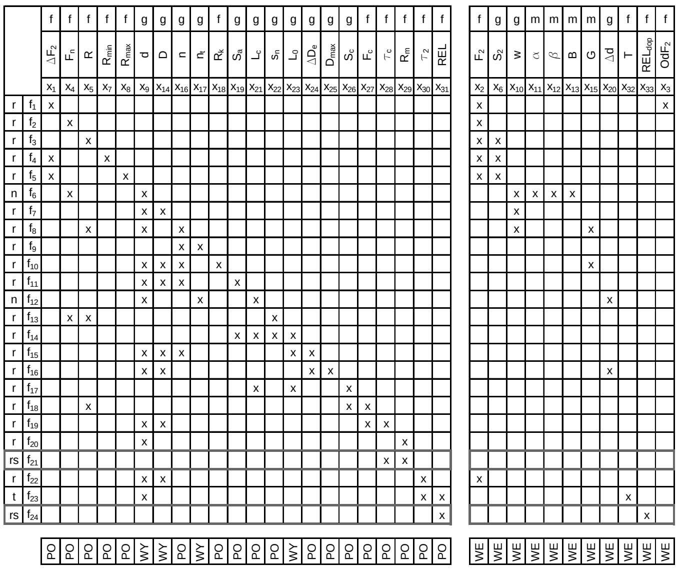
5. Example of application: setting up an algorithm for helical spring calculation

For helical springs used in ordinary applications 24 constraints of various kind have been selected [Branowski 1997], as shown in Table 1. In these constraints 33 variables appear. If we assume that the constraints specified in Table 1 represent the basic body of formal knowledge of helical compression springs then they can be used for solving any problem of the spring analysis and/or synthesis. The variety of problems that may occur in practice is enormous.
The relevant variables for the helical spring are:
B - spring coefficient (usually *B* = 0,5 for compression springs); *D* - mean coil diameter; D_{emax} - maximal outside spring diameter; ΔD_e - diameter enlargement of compressed spring; F_c - theoretical load of blocked spring; F_2 - spring load; ΔF_2 - load deviation; *G* - modulus of elasticity; L_c - length of the blocked spring; L_0 - free spring length; OdF_2 - relative load deviation; *R* - spring rate constant; R_k - corrected stiffness (averaged); R_m - tensile strength; REL_{dop} - allowable relaxation index; S_a - minimal clearance between coils (total); *T* - spring working temperature; *d* - wire diameter; Δd - wire diameter deviation; *n* - number of active coils; n_t - total number of coils; s_c - blocking deflection; s_2 - spring deflection; *w* - spring index; α, β - material constants; τ_2 - shearing stress; τ_c - shear stress of the blocked spring.
The occurrence matrix shown in Fig. 5 encompasses the constraints in rows in original order. The first left column in the matrix denotes a constraint type; *r* indicates equation, *n* – inequality, *t* – a constraint in form of table or chart. The second left column shows constraint numbers. The second upper row designates symbols of variables which occur in the constraints, while the first one denotes the type of a variable; *f*- indicates the functional variable, *g* – geometric, *m* – material. x_1, x_2,…. x_{33} are symbols for relevant variables. If a variable occurs in a constraint then the relevant cell is marked.

		f	f	f	f	f	g	f	f	g	g	m	m	m	g	m	g	g	f	g	g	g	g	g	g	g	g	f	f	f	f	f	f	f
		ΔF_2	F_2	OdF_2	F_n	R	S_2	R_{min}	R_{max}	d	w	α	β	B	D	G	n	n_t	R_k	S_a	Δd	L_c	S_n	L_0	ΔD_e	D_{max}	S_c	F_c	τ_c	R_m	τ_2	REL	T	REL_{dop}
		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}	x_{28}	x_{29}	x_{30}	x_{31}	x_{32}	x_{33}
r	f_1	x	x	x																														
r	f_2		x		x																													
r	f_3		x			x	x																											
r	f_4	x	x				x	x																										
r	f_5	x	x				x		x																									
n	f_6				x					x	x	x	x	x																				
r	f_7									x	x				x																			
r	f_8					x				x	x					x	x																	
r	f_9																x	x																
r	f_{10}									x					x	x	x		x															
r	f_{11}									x					x		x			x														
n	f_{12}									x								x			x	x												
r	f_{13}				x	x																	x											
r	f_{14}																			x		x	x	x										
r	f_{15}									x					x		x							x	x									
r	f_{16}									x					x						x				x	x								
r	f_{17}																					x		x			x							
r	f_{18}					x																					x	x						
r	f_{19}									x					x													x	x					
r	f_{20}									x																				x				
n	f_{21}																												x	x				
r	f_{22}		x							x					x																x			
t	f_{23}									x																					x	x	x	
n	f_{24}																															x		x

Figure 5. Neutral occurrence matrix for helical spring

The original neutral matrix could be ordered and blocked in order to decompose it to smaller sub-matrices or this operation can be executed after removing the input variables.

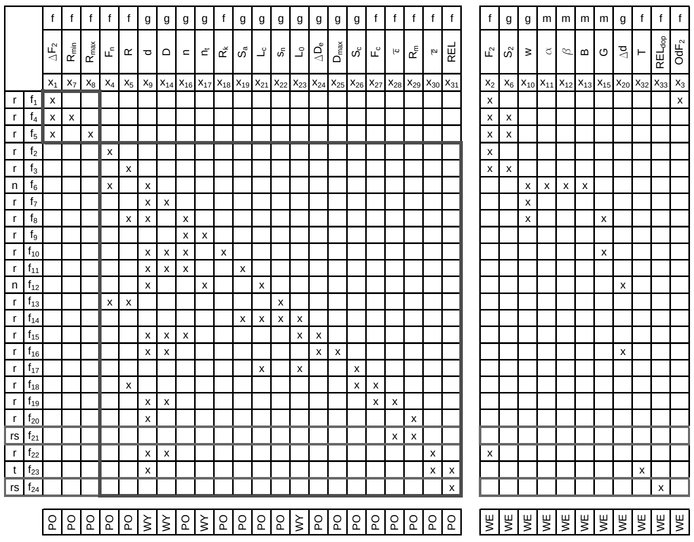


		f	f	f	f	f	g	g	g	g	f	g	g	g	g	g	g	g	f	f	f	f	f	f	g	g	m	m	m	m	g	f	f	f
		ΔF_2	F_n	R	R_{min}	R_{max}	d	D	n	n_t	R_k	S_a	L_c	S_n	L_0	ΔD_e	D_{max}	S_c	F_c	τ_c	R_m	τ_2	REL	F_2	S_2	w	α	β	B	G	Δd	T	REL_{dop}	OdF_2
		x_1	x_4	x_5	x_7	x_8	x_9	x_{14}	x_{16}	x_{17}	x_{18}	x_{19}	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}	x_{28}	x_{29}	x_{30}	x_{31}	x_2	x_6	x_{10}	x_{11}	x_{12}	x_{13}	x_{15}	x_{20}	x_{32}	x_{33}	x_3
r	f_1	x																						x										x
r	f_2		x																					x										
r	f_3			x																				x	x									
r	f_4	x			x																			x	x									
r	f_5	x				x																		x	x									
n	f_6		x				x																			x	x	x	x					
r	f_7						x	x																		x								
r	f_8			x			x		x																	x				x				
r	f_9								x	x																								
r	f_{10}						x	x	x		x																			x				
r	f_{11}						x	x	x			x																						
n	f_{12}						x			x			x																		x			
r	f_{13}		x	x										x																				
r	f_{14}											x	x	x	x																			
r	f_{15}						x	x	x						x	x																		
r	f_{16}						x	x								x	x														x			
r	f_{17}												x		x			x																
r	f_{18}			x														x	x															
r	f_{19}						x	x											x	x														
r	f_{20}						x														x													
rs	f_{21}																			x	x													
r	f_{22}						x	x														x		x										
t	f_{23}						x															x	x									x		
rs	f_{24}																						x										x	
		PO	PO	PO	PO	PO	WY	WY	PO	WY	PO	PO	PO	PO	WY	PO	PO	PO	PO	PO	PO	PO	PO	WE	WE	WE	WE	WE	WE	WE	WE	WE	WE	WE

Figure 6. The directed occurrence matrix for a set of determined variables shown on the right

An example of the directed matrix is shown in Fig. 6 where input variables are separated on the right. These are named as WE. The task consists in finding out the wire diameter d, total number of coils n_t, mean coil diameter D, and free spring length L_0. These are the main output variables named as WY. All other variables which should also be evaluated are regarded as intermediate and named as PO.

Now in the occurrence matrix there are only 22 variables to evaluate but still 24 constraints with four inequalities amongst them. Since inequalities can not be used for calculation of variables then two of them should be turned into equations. The other two inequalities, designated *rs*, must be satisfied as they are.



		f	f	f	f	f	g	g	g	g	f	g	g	g	g	g	g	g	f	f	f	f	f	f	g	g	m	m	m	m	g	f	f	f
		ΔF_2	R_{min}	R_{max}	F_n	R	d	D	n	n_t	R_k	S_a	L_c	S_n	L_0	ΔD_e	D_{max}	S_c	F_c	τ_c	R_m	τ_2	REL	F_2	S_2	w	α	β	B	G	Δd	T	REL_{dop}	OdF_2
		x_1	x_7	x_8	x_4	x_5	x_9	x_{14}	x_{16}	x_{17}	x_{18}	x_{19}	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}	x_{28}	x_{29}	x_{30}	x_{31}	x_2	x_6	x_{10}	x_{11}	x_{12}	x_{13}	x_{15}	x_{20}	x_{32}	x_{33}	x_3
r	f_1	x																						x										x
r	f_4	x	x																					x	x									
r	f_5	x		x																				x	x									
r	f_2				x																			x										
r	f_3					x																		x	x									
n	f_6				x		x																			x	x	x	x					
r	f_7						x	x																		x								
r	f_8					x	x		x																	x				x				
r	f_9								x	x																								
r	f_{10}						x	x	x		x																			x				
r	f_{11}						x	x	x			x																						
n	f_{12}						x			x			x																		x			
r	f_{13}				x	x								x																				
r	f_{14}											x	x	x	x																			
r	f_{15}						x	x	x						x	x																		
r	f_{16}						x	x								x	x														x			
r	f_{17}												x		x			x																
r	f_{18}					x												x	x															
r	f_{19}						x	x											x	x														
r	f_{20}						x														x													
rs	f_{21}																			x	x													
r	f_{22}						x	x														x		x										
t	f_{23}						x															x	x									x		
rs	f_{24}																						x										x	
		PO	PO	PO	PO	PO	WY	WY	PO	WY	PO	PO	PO	PO	WY	PO	PO	PO	PO	PO	PO	PO	PO	WE	WE	WE	WE	WE	WE	WE	WE	WE	WE	WE

Figure 7. The decomposed occurrence matrix for the helical spring calculation

After re-arranging rows and columns in the matrix a new its form is obtained, consisting of two independent blocks. It is shown in Fig. 7. This matrix has been used to found a rational calculation algorithm. Babirecki has shown that four different algorithms can be set up. One is shown in Table 1.

Table 1. Calculation algorithm for helical spring

Calc. order		Constraints	Evaluated variables
		From relation	Calculate
BLOCK 1	1	$f_1 = f(\Delta F_2, OdF_2, F_2)$	$x_1 = \Delta F_2 = [OdF_2]\frac{F_2}{100}$
	2	$f_4 = f(s_2, F_2, \Delta F_2, R_{\min})$	$x_7 = R_{\min} = \frac{F_2 - \Delta F_2}{s_2}$
	3	$f_5 = f(s_2, F_2, \Delta F_2, R_{\max})$	$x_8 = R_{\max} = \frac{F_2 + \Delta F_2}{s_2}$
BLOCK 2	1	$f_2 = f(F_n, F_2)$	$x_4 = F_n = 1{,}15 F_2$
	2	$f_3 = f(R, s_2, F_2)$	$x_5 = R = \frac{F_2}{s_2}$
	3	$f_{13} = f(s_n, R, F_n)$	$x_{22} = s_n = \frac{F_n}{R}$
	4	$f_6 = f(d, F_n, w, \alpha, \beta, B)$	$x_9 = d \geq \left(\frac{8F_n w}{\pi B \alpha}\right)^{\frac{1}{2-\beta}}$ - turned into equation
	5	$f_7 = f(d, D, w)$	$x_{14} = D = wd$
	6	$f_8 = f(d, G, w, R, n)$	$x_{16} = n = \frac{Gd}{8w^3 R}$
	7	$f_{20} = f(R_m, d)$	$x_{29} = R_m = 0{,}5[(2220 - 820\log d) + (2470 - 90\log d)]$
	8	$f_9 = f(n_t, n)$	$x_{17} = n_t = n + 2$
	9	$f_{10} = f(d, G, D, R_k, n)$	$x_{18} = R_k = \frac{Gd^4}{8D^3 n}$
	10	$f_{11} = f(d, S_a, D, n)$	$x_{19} = S_a = \left(0{,}0015\frac{D^2}{d} + 0{,}1d\right)n$
	11	$f_{22} = f(F_2, D, d, \tau_2)$	$x_{30} = \tau_2 = \frac{8F_2 D}{\pi d^3}$
	12	$f_{23} = f(REL, D, d, \tau_2)$	$x_{31} = REL = f(T, d, \tau_2)$
	13	$f_{24} = f(REL, REL_{dop})$	$REL \leq REL_{dop}$ - used for examination only
	14	$f_{12} = f(d, n_t, \Delta d, L_C)$	$x_{21} = L_C \leq n_t(d + \Delta d)$ - turned into equation
	15	$f_{14} = f(L_0, L_c, S_a, s_n)$	$x_{23} = L_0 = L_c + S_a + s_n$
	16	$f_{15} = f(L_0, d, n, D, \Delta D_e)$	$x_{24} = \Delta D_e = 0{,}1\frac{\left(\frac{L_0 - d}{n}\right)^2 - 0{,}8\left(\frac{L_0 - d}{n}\right)d - 0{,}2d^2}{D}$
	17	$f_{17} = f(L_0, L_c, s_c)$	$x_{26} = s_c = L_0 - L_c$
	18	$f_{16} = f(D_{e\max}, D, \Delta D_e, d)$	$x_{25} = D_{e\max} = D + d + \Delta D_e$
	19	$f_{18} = f(F_c, R, s_c)$	$x_{27} = F_c = Rs_c$
	20	$f_{19} = f(F_c, D, d, \tau_c)$	$x_{28} = \tau_c = \frac{8F_c D}{\pi d^3}$
	21	$f_{21} = f(R_m, \tau_c)$	$\tau_c \leq 0{,}56 R_m$ - used for examination only

All dependent variables can be calculated sequentially. Note that information from the Block 1 is not used in Block 2, which suggests some void in the set of constraints.

The user may want to know which variables are influenced by a certain input e.g. modulus of elasticity *G*. Since *G* appears in f_8 and f_{10} so, according to the algorithm in Table 1, it will affect directly only R_k and *n* which, in turn, appears in f_9, f_{11}, and f_{15}. Thus, n_t and S_a and ΔDe are indirectly dependent on *G* because they depend directly on R_k and *n.* In Fig. 8 it is shown how the chain of dependence proceeds. The final result is that modulus of elasticity *G* has an impact upon D_{emax}, Δde, F_c, L_0, L_c, R_k, S_a, *n*, n_t, s_c, *and* τ_c. It should be noted that this pattern holds only for the algorithm of Table 1.

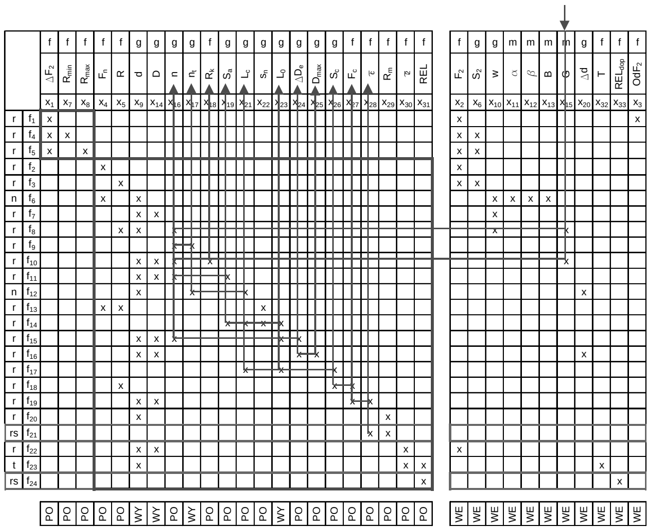


Figure 8. Tracing variables dependent on G

The matrix can also be used for identification which input variables influence certain dependent variable of interest. The way of search for the wire diameter *d* is shown in Fig. 9. It appears that the diameter is dependent on the load F_2, spring index *w*, and the spring material constants *B*, α, β. Like in the former example, this result holds for this particular calculation algorithm only.

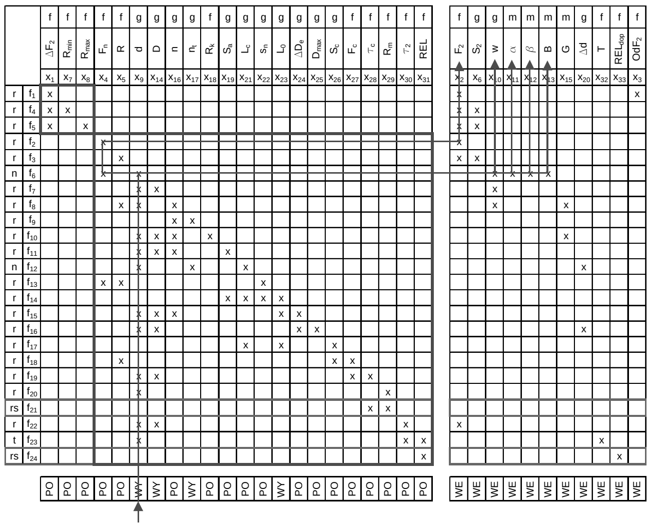


Figure 9. Tracing dependence of d on input variables

6. Conclusions

The occurrence matrix is a convenient tool for storing relationships among variables and constraints and for identifying dependencies among the variables. It was shown that the decomposition procedure enables the user to find the most efficient sequence for computing unknown variables and to have the full insight into the computing process. The method presented in this paper is not limited to dealing with equations only but allows for inequalities, tabular data, and other forms of relations between variables. The user can convert inequalities into equalities or leave them for control only.
The user can define a list of constraints and related variables, and any of the variables can be selected for input at any time. Either inputs or outputs can be respecified in any stage of the calculations.
It can be checked what dependent variables are affected by a specified input variable.
It can be checked how a change in the value of an input variable affects the dependent variables. This feature is beneficial for checking sensitivity against tolerances and other deviations.
It can be checked which input variables affect a specified output variable.
It can be checked which knew variables can be computed after introducing a new input variable.
It can be checked which variables can not be computed when an input variable has been removed.
When a dependent variable is changed to the input it is easy to recognize its effect on the other variables. This case may occur when the user is not entirely satisfied by the computed value of a variable and prefers an exact, e. g. taken from a catalogue, value of this variable.
The presented methodology can be used in design of mechanical systems, particularly in larger problems in which there are complex dependencies between the variables. All body of formalized knowledge of a class of elements, assemblies or machines can be represented by means of the occurence matrices and processed interactively with the user according to a given task. Thanks to the presented techniques the user possess the possibility to plan the calculations in advance and to navigate rationally through constraints and variables. An interactive computer program in DELPHI has been devised in order to exempt the user from laborious operations with matrices.

References

Aggraval, R., et al., "Engineering Constraint Management Based on an Occurrence Approach", Journal of Mechanical Design, Vol. 115, March 1993, pp. 103-109.
Babirecki, W., "Towards rationalization of calculation of machine elements and assemblies by application of dependency matrices" (in Polish), PhD Thesis, Department of Mechanical Engineering, The University of Zielona Góra, 2005.
Branowski, B., "Metal Springs" (in Polish), WNT Warszawa, 1997.
Eppinger, S., "Model-Based Approaches to Managing Concurrent Engineering", ICED Zurich, August 1991.
Friedman, G. J., Leondes, C. T., „Constraint theory. Part I: Fundamentals", IEEE Trans. Syst. SCI & Cyber. Vol. SSC-5, No. 1, 1969, pp. 48-56.
Kusiak, A., Cheng, C. H., „A branch-and-bound algorithm for solving the group technology problem", Annals of Operations Research Vol. 26, 1990, pp. 415-431.
Serrano, D., Gossard, D., "Constraint Management in MCAE" Artificial Intelligence in Engineering Design, J. S. Gero (Ed.), Computational Mechanics Publications, 1988, pp. 361-378.
Steward D. V., "Partitioning and tearing systems of equations", SIAM J. Numer. Anal., Vol. 2, No. B, 1965.
Warfield, D. E., "Binary matrices in system modeling", IEEE Transactions on Systems, Man, and Cybernetics, SMC-3, 1973, pp. 441-449.

Ryszard Rohatyński, Prof. dr hab. eng.
Chair of Production Systems Design
University of Zielona Góra, Department of Management
Podgórna 50, 65-246 Zielona Góra, Poland
Tel.: +48 68 328 2546
Fax.: +48 68 328 2554
Email: r.rohatynski@wz.uz.zgora.pl