Korean J. of Chem. Eng., **4**(2) (1987) *154-160*

A METHODOLOGY FOR EQUATION-BASED ANALYSIS OF LARGE CHEMICAL PROCESSES USING THE FUNCTIONAL MATRIX

Soo Hyoung CHOI and En Sup YOON

Department of Chemical Engineering, Seoul National University

(Received 30 November 1986 • accepted 6 May 1987)

Abstract—The equation-based approach to process analysis is necessary for efficient optimization of large and complex processes, and involves the problem of solving a large number of equations. To complement this approach, an improved equation-solving system using the functional matrix suggested by Mattione, Meir and Book was developed, and its capacity for the process analysis was demonstrated by case studies.

The equation-solving system developed in this work reads equations, stores them in the functional matrix, rearranges them, and, if they have degrees of freedom, selects design variables which make each partition the easiest to solve. Given the values of the design variables, the system solves the equations as it manipulates the functional matrix.

The developed equation-solving system was proved to be efficient for solving a large number of equations which involve degrees of freedom. Case studies show that the methodology established in this work is an appropriate basis for the equation-based analysis of large chemical processes.

INTRODUCTION

The most promising approach to process analysis is the two tier approach, in which the equation-based approach with simple models and the sequential modular approach with rigorous models are used together. This approach was represented by Rosen [1], Evans [2], et al. The objective of this study is to present a computational basis for the equation-based analysis in the two tier approach.

One of the most important jobs in chemical process design is solving large sets of process-modeling equations. The set of equations to be solved usually consists of both linear and nonlinear equations, and is very sparse when represented in matrix form. The total system of equations, if possible, must be partitioned into subsystems of equations so that the computational difficulty in equation-solving may be minimized.

Algorithms for partitioning systems with no degrees of freedom were given by Steward [3] [4], Sargent and Westerberg [5], Himmelblau [6], Tarjan [7], et al. Results of these algorithms are the same, i.e. they give the complete partitions.

A set of equations modeling a process always contains more variables than equations. It is a very important problem which variables to set to constants, because the configuration of the partitioned set of equations with zero degree of freedom may critically vary with the set of design variables. Algorithms for rearranging equations and selecting design variables were given by Lee, Christensen and Rudd [8], Edie and Westerberg [9], Ramirez and Vestal [10], Book and Ramirez [11] [12], et al.

In previous work, the occurrence matrix in which an element represents just the occurrence of a variable in an equation was being used in order to express the structure of a set of equations. Then, Mattione, Meir and Book [13] presented a type of occurrence matrix, called the functional matrix, in which an element represents not only the occurrence but also the functional form of a variable in an equation. Book and Ramirez [12] made use of the functional matrix in their equation-ordering and variable-grouping algorithm.

After partitioning, the tearing, which is another method for reducing the computational difficulty, can be applied to each partition, so that the solution of subsystems of equations can be iterated. Algorithms for tearing were given by Steward [2], Christensen [14], et al., but not discussed here.

Table 1. Defined functional forms.

Designation	Functional form	Equation form
A	Linear	$C_1x_1 + f(x) = 0$
B	Product	$C_1x_1x_2\cdots x_n + f(x) = 0$
D	Reciprocal	$C_1/(x_1 + C_2) + f(x) = 0$
E	Exponential	$C_1\exp(C_2x_1) + f(x) = 0$
G	Hyperbolic tangent	$C_1\tanh(x_1 + C_2) + f(x) = 0$
H	Hyperbolic sine	$C_1\sinh(x_1 + C_2) + f(x) = 0$
L	Natural logarithm	$C_1\ln(x_1 + C_2) + f(x) = 0$
M	Common logarithm	$C_1\log(x_1 + C_2) + f(x) = 0$
N	Cubic	$C_1(x_1 + C_2)^3 + f(x) = 0$
O	Square root	$C_1x_1 + C_2 + f(x) = 0$
P	Power	$C_1x_1C_2 + f(x) = 0$
R	Fourth order	$C_1(x_1 + C_2)^4 + f(x) = 0$
S	Square	$C_1(x_1 + C_2)^2 + f(x) = 0$
T	Tangent	$C_1\tan(x_1 + C_2) + f(x) = 0$
U	Sine	$C_1\sin(x_1 + C_2) + f(x) = 0$
V	Cosine	$C_1\cos(x_1 + C_2) + f(x) = 0$
W	Hyperbolic cosine	$C_1\cosh(x_1 + C_2) + f(x) = 0$

EQUATION-SOLVING METHODOLOGY

Functional Matrix

An element of the functional matrix represents the function formed by a variable in an equation. The functional form which can appear in an equation is defined as shown in Table 1, where f(**x**) is a function of variables other than explicit ones in the equation. An element which is designated by a letter after 'p' involves multiple roots.

The row-oriented column-linked list is used in order to store the functional matrix. Elements in the topmost row are stored contiguously in the memory in the increasing order of their column numbers, and those in the next row, etc. Elements in each column are linked from top to bottom. A record for an element node contains fields for the code representing the functional form and the constants specifying the function, as shown in Fig. 1. Fields for the mark and the Jacobian matrix are used only while the functional matrix is being used for equation-solving.

A type of circular list is used in order to express the product of variables. The code is used as the link. If the value of code of an element is positive, it corresponds to the element number of the multiplied variable. The code of the multiplied element is the element number of the next multiplied variable, etc. The code of the last multiplied element is the element number of the first variable in the product, so the elements are circularly linked. The product is evaluated by tracing the link with the visited nodes being marked.

Row	Column	Column-link	Code	C_1	C_2	Mark	Jacobian

Fig. 1. Contents of an element node.

Functional forms other than the product of variables are represented by nonpositive code values. Fig. 2 shows an example of the functional matrix, which contains the same equations that were used for an example by Mattione, Meier and Book [13].

Equation-Ordering and Partitioning

An equation which contains only one variable can be solved first. Therefore, such an equation is put to the topmost row, and its variable is put to the leftmost column. Then, its row and column are eliminated from the original matrix. The same work is repeated until such an equation is not found.

If a variable appears in only one equation, the equation which contains that variable can be solved after all the other equations are solved. Therefore, such an equation is put to the bottommost row, and such a variable is put to the rightmost column. Then, the row of that equation and the column of that variable are eliminated from the original matrix. The same work is repeated until such a variable is not found.

If there are remaining equations in the original matrix, the equation with the minimum number of variables is put to the topmost empty row [15]. If there are several candidates, the one with which the maximum number of elements are to be eliminated is selected. Then, its original row and columns are eliminated. The same work is repeated for the remaining

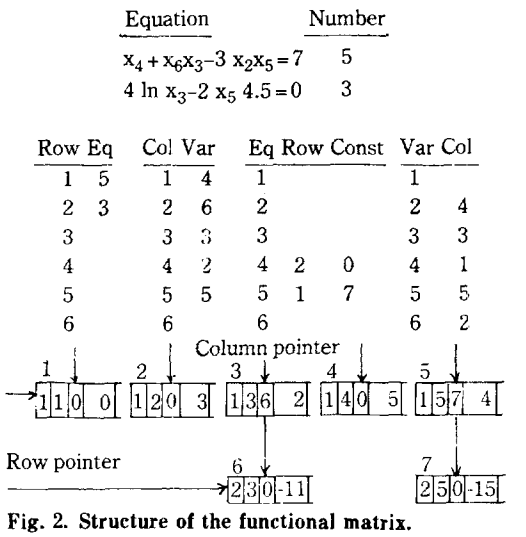

Fig. 2. Structure of the functional matrix.

equations. This is a heuristic method which does not guarantee the most small-sized partitions, and is a simple method which does not involve any serious combinatorial problem.

After all the equations are ordered, the rearranged matrix is partitioned. Conventionally, the partition is defined as the smallest subsystem of equations. Here, contiguous unit partitions are so assembled that they compose a trapezoid or triangular partition as shown in Fig. 5. The size of partition is defined as the number of equations that must be solved simultaneously in the partition.

Design Variable Selection

After partitions are determined, proper design variables are selected so that each partition with degrees of freedom may become the easiest to solve. In previous work using the occurrence matrix, any situation that all the equations can be solved sequentially was the most preferred one. Here, an objective function to be minimized is used, which was used by Book and Ramirez [12] during the equation-ordering. It is as follows.

1. Equations that can be solved sequentially giving a single root vector
2. Linear simultaneous equations
3. Equations that can be solved sequentially giving multiple root vectors
4. Nonlinear simultaneous equations

The equation-solving system selects output variables involving multiple roots in a trapezoid partition for design variables, if that selection gives a set of linear simultaneous equations. If there is a partition of simultaneous equations with some degrees of freedom, the system selects variables in nonlinear functional form for design variables so that the simultaneous equations may become linear.

Numerical Solution

If appropriate values are assigned to the selected design variables, the set of equations can be solved. Because the functional matrix represents the equation set itself, it can be directly used for equation-solving.

An equation which contains only one variable is solved simply by applying the inverse function. The determined variable is immediately changed into a constant, with the functional matrix being accordingly modified. Thus, the equations that can be solved sequentially are solved in such an analytical manner. If an output variable involves multiple roots, its lower and upper bounds are needed.

Because the functional matrix contains defined functional forms only, the equation-solving system knows the derivatives. Thus, the elements of the Jacobian matrix are evaluated analytically [13]. Therefore, the Newton-Raphson method can efficiently solve the nonlinear simultaneous equations.

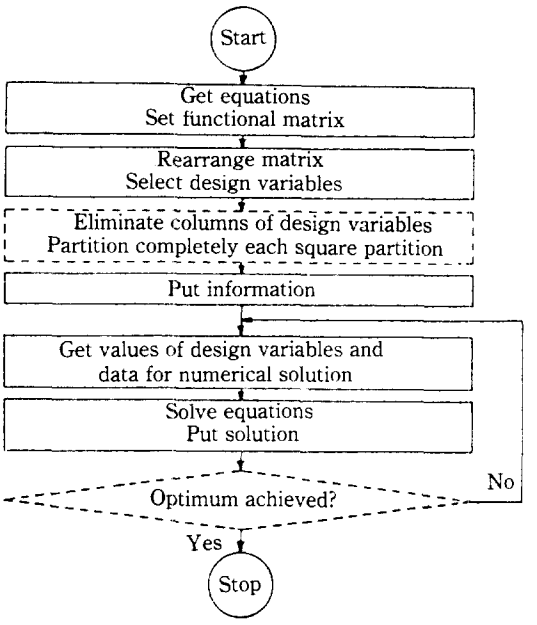

Fig. 3. Schematic diagram of equation-solving system.

Equation-Solving System

Fig. 3 represents the equation-solving system developed in this work. The steps enclosed with dashed lines in that figure are not currently contained in the system. The equation-solving step uses the Harwell subroutine libray to inverse the matrix.

A CASE STUDY

Modeling

A hypothetical process shown in Fig. 4, which was used by Evans [16] for an example of process analysis in the sequential modular approach, is to be treated as an example. The reactor is an isothermal CSTR, and the separator is an adiabatic flash drum. The liquid phase reactions in the CSTR are described on Table 2, and the physical properties of the components are listed on Table 3. Equations which simply model the CSTR-flash process neglecting the pump and the valve are listed on Table 4, and their variables on Table 5. The equation set has 8 degrees of freedom.

Design

A design problem of obtaining the product purity of 0.95, which gives 32 nonlinear simultaneous equations, was solved efficiently. If the sequential modular approach were used, this problem would require time-consuming iterative simulations. The result is on Table 6, where underlined numbers represent given values. The purge fraction of the splitter must be set to 0.204,

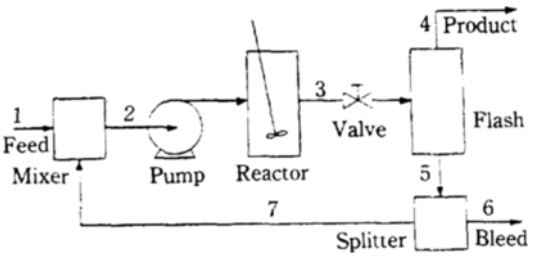

Fig. 4. CSTR-flash process [16].

Table 2. Reactions in CSTR-flash process [16].

Number	Reaction		Order
1	A (feed)	→ B (product)	First
2	B (product)	→ C (by-product)	Second

$a_1 = 0.1269 \times 10^{12}$ hr^{-1} $E_1/R = 9944$ °K

$a_2 = 0.1921 \times 10^{6}$ m^3/(kg-mole hr) $E_2/R = 6617$ °K

Table 3. Physical properties of components in CSTR-flash process [16].

Physical property	Component		
	A	B	C
ρ_i, kg-mole/m^3	10.69	10.69	10.69
λ_i, kJ/kg-mole	40890	37560	42010
C_i, kJ/(kg-mole °K)	165.8	169.5	191.6
A_i	13.15	12.19	13.77
B_i, °K	5634	4272	6023

if the other units are to be unchanged.

Optimization

In the above example, all the 8 design variables were selected by the designer. However, the known variables are just the two of $x_{A1}=1$ and $x_{B1}=0$. If the other 6 design variables are selected by the equation-solving system, the set of equations is partitioned more effectively than in the above example. The rearranged

Table 4. Model equations for CSTR-flash process [16].

Equation	Number	Equation	Number
$x_{A1}+x_{B1}+x_{C1}=1$	1	$x_{A5}+x_{B5}+x_{C5}=1$	25
$F_1+F_7-F_2=0$	2	$T_4-T_5=0$	26
$x_{A1}F_1+x_{A7}F_7-x_{A2}F_2=0$	3	$\ln P_A+B_A/T_4=A_A$	27
$x_{B1}F_1+x_{B7}F_7-x_{B2}F_2=0$	4	$y_{A4}P_F-x_{A5}P_A=0$	28
$x_{C1}F_1+x_{C7}F_7-x_{C2}F_2=0$	5	$\ln P_B+B_B/T_4=A_B$	29
$x_{A1}C_A+x_{B1}C_B+x_{C1}C_C-z_1=0$	6	$y_{B4}P_F-x_{B5}P_C=0$	30
$x_{A7}C_A+B_7C_B+x_{C7}C_C-z_2=0$	7	$\ln P_C+B_C/T_4=A_C$	31
$x_{A2}C_A+x_{B2}C_B+x_{C2}C_C-z_3=0$	8	$y_{C4}P_F-x_{C5}P_C=0$	32
$F_1z_1T_1+F_7z_2T_7-F_2z_3T_2=0$	9	$y_{A4}C_A+y_{B4}C_B+y_{C4}C_C-z_9=0$	33
$F_2-F_3=0$	10	$z_9T_4+y_{A4}\lambda_A+y_{B4}\lambda_B+y_{C4}\lambda_C-z_{10}=0$	34
$T_3z_4=1$	11	$x_{A5}C_A+x_{B5}C_B+x_{C5}C_C-z_{11}=0$	35
$\exp[-(E_1/R)z_4]-z_5=0$	12	$x_{A3}C_A+x_{B3}C_B+x_{C3}C_C-z_{12}=0$	36
$\theta_1F_3-a_1\rho_RV_Rz_5=0$	13	$F_4z_{10}+F_5z_{11}T_5-F_3z_{12}T_3=0$	37
$\exp[-(E_2/R)z_4]-z_6=0$	14	$f_SF_5-F_6=0$	38
$\theta_2F_3-a_2\rho_R^2V_Rz_6=0$	15	$z_{13}+f_S=1$	39
$z_7-\theta_1=1$	16	$z_{13}F_5-F_7=0$	40
$x_{A32}z_7-x_{A2}=0$	17	$x_{A5}-x_{A6}=0$	41
$x_{B3}-z_8=0$	18	$x_{A6}-x_{A7}=0$	42
$\theta_2z_8+x_{B3}-\theta_1x_{A3}-x_{B2}=0$	19	$x_{B5}-x_{B6}=0$	43
$x_{A3}+x_{B3}+x_{C3}=1$	20	$x_{B6}-x_{B7}=0$	44
$y_{A4}F_4+x_{A5}F_5-x_{A3}F_3=0$	21	$x_{C5}-x_{C6}=0$	45
$y_{B4}F_4+x_{B5}F_5-x_{B3}F_3=0$	22	$x_{C6}-x_{C7}=0$	46
$y_{C4}F_4+x_{C5}F_5-x_{C3}F_3=0$	23	$T_5-T_6=0$	47
$y_{A4}+y_{B4}+y_{C4}=1$	24	$T_6-T_7=0$	48

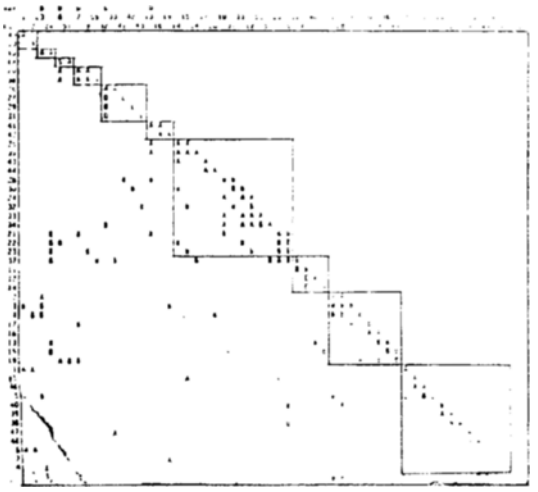

Fig. 5. Rearranged matrix of CSTR-flash process.

matrix is shown in Fig. 5, where letter 'D's on the top line represent the selected design variables.

The process optimization is performed by determining the solution which minimizes or maximizes an appropriate objective function. As the equation-solving is repeated, the selected 6 design variables are adjusted by some of control strategies such as the steepest descent or ascent method, the pattern search, etc.

DISCUSSION

The functional matrix was found to be a versatile tool for treating a large number of equations. However, the constraint on the functional form causes the increase in the number of equations. Shacham [17] states that the computational difficulty can be reduced by changing nonlinear equations into linear ones and additional nonlinear ones and applying appropriate tearing. However, the convergence rate is not so important. The increase in the size of equation set is a burden to equation-solving.

The structure of the functional matrix can be improved so that any type of equation can be stored. In that case, however, analytical evaluation of Jacobian matrix will become much more difficult. The algorithm given by Book and Ramirez [12] checks the multiplicity of the root of the variable which appears in only one equation, while ordering equations, and if that output variable involves multiple roots, searches for a set of linear simultaneous equations which can replace that output variable. This strategy involves a combinatorial problem.

If the above-mentioned search is done, the equation which has the output variable involving multiple roots must enter into the set of linear simultaneous equations, making that variable become a design variable. Otherwise, that variable persists as an output variable. Therefore, it makes the same effect avoiding the combinatorial problem to order equations ignoring the functional form first and then select design variables properly.

Some optimal design variable selection may assemble several contiguous partitions into one large linear

Table 5. Variables for CSTR-flash process.

Variable	Number	Variable	Number	Variable	Number
x_{A1}	1	x_{C7}	21	P_A	41
x_{B1}	2	F_1	22	P_B	42
x_{C1}	3	F_2	23	P_C	43
x_{A2}	4	F_3	24	z_1	44
x_{B2}	5	F_4	25	z_2	45
x_{C2}	6	F_5	26	z_3	46
x_{A3}	7	F_6	27	z_4	47
x_{B3}	8	F_7	28	z_5	48
x_{C3}	9	T_1	29	z_6	49
y_{A4}	10	T_2	30	z_7	50
y_{B4}	11	T_3	31	z_8	51
y_{C4}	12	T_4	32	z_9	52
x_{A5}	13	T_5	33	z_{10}	53
x_{B5}	14	T_6	34	z_{11}	54
x_{C5}	15	T_7	35	z_{12}	55
x_{A6}	16	V_R	36	z_{13}	56
x_{B6}	17	P_F	37		
x_{C6}	18	f_S	38		
x_{A7}	19	$_1$	39		
x_{B7}	20	$_2$	40		

Table 6. Design of CSTR-flash process.

Unit	Specification				
Reactor	Volume = **0.283** m^3			Temperature = **478** °K	
Separator	Pressure = **1** atm				
Splitter	Purge fraction = 0.204				
Stream	Flow rate kg-mole/hr	Temperature °K	Composition A	B	C
Feed	**45.4**	**298**	**1**	**0**	0
Reactor output	76.1	478	0.1217	0.7215	0.1568
Product	37.5	370	0.0267	**0.95**	0.0233
Recycle	30.7	370	0.2140	0.4994	0.2865

partition. However, the detection of such a situation involves another combinatorial problem. The current equation-solving system can not overcome the boundaries of partitions, but gives the best results in most cases.

In the case study, each initial guess value for solution of nonlinear simultaneous equations had to be at least of the same order of magnitude as that of the solution. Unless a numerical method which is not sensitive to the initial guess or a system which gives the initial guess is developed, the equation-based analysis of large chemical processes will be very difficult. When the iterative solution converged, the CPU time required to read, rearrange and solve the equations listed on Table 4 was less than 5 seconds on VAX-11/750.

CONCLUSION

A new type of equation-solving system using the functional matrix was developed and tested in this work. The methodology given here was found to be an appropriate basis for the equation-based analysis of large chemical processes.

Any of currently useful equation-solving methods such as Newton-Raphson method, Broyden-Schubert method, etc. can not easily give the solution of large systems of highly nonlinear equations. The development of methods for efficient equation-solving has almost always depended on the improvement in numerical methods. The real bottleneck in the equation-solving procedure is the step of initial guess. If the human work in trial-and-error fashion can be performed by a system of artificial intelligence, the equation-solving system connected to the AI production system will be much more powerful than any one that has ever existed.

It is possible to develop a system which generates model equations for a given flowsheet. The equation-generating system which uses simple models and the equation-solving system which efficiently treats large systems of algebraic equations will make the equation-based analysis of large chemical processes feasible. The detailed optimization can be based on the sequential modular approach using the results obtained by the equation-based approach.

NOMENCLATURE

A_i : coefficient in vapor pressure correlation for component i
a_n : frequency factor of reaction n, hr^{-1} or $m^3/(kg\text{-}mole\ hr)$
B_i : coefficient in vapor pressure correlation for component i, °K
C_i : heat capacity of component i in liquid phase, kJ/(kg-mole °K)
E_n : activation energy of reaction n, kJ/kg-mole
F_j : flow rate of stream j, kg-mole/hr
f_k : purge fraction of unit k
P_i : vapor pressure of component i, atm
P_k : pressure in unit k, atm
R : gas constant, kJ/(kg-mole °K)
T_j : temperature of stream j, °K
V_k : volume of unit k, m^3
x_{ij} : mole fraction of component i in liquid phase in stream j
y_{ij} : mole fraction of component i in vapor phase in stream j
z_n : substitution variable

Greek Letters

θ_n : substitution variable
λ_i : heat of vaporization of component i, kJ/kg-mole
ρ_i : density of component i in liquid phase, kg-mole/m^3
ρ_k : density of fluid in unit k, kg-mole/m^3

REFERENCES

1. Rosen, E.M.: Foundations of Computer-Aided Chemical Process Design, Vol. 1, pp. 529-533, Engineering Foundation, New York (1981).
2. Evans, L.B.: ibid, pp. 425-469.
3. Steward, D.V.: SIAM Rev., **4,** 321 (1962).
4. Steward, D.V.: *SIAM J. Numer. Anal.*, Ser. B, **2,** 345 (1965).
5. Sargent, R.W.H. and Westerberg, A.W.: *Trans. of the Inst. of Chem. Eng.*, **42,** T 190 (1964).
6. Himmelblau, D.M.: *Chem. Eng. Sci.*, **22,** 883 (1967).
7. Tarjan, R.: *SIAM J. Comput.*, **1,** 146 (1972).
8. Lee, W., Christensen, J.H. and Rudd, D.F.: *AIChE J.* **12,** 1105 (1966).
9. Edie, F.C. and Westerberg, A.W.: *Chem. Eng. J.*, **2,** 114 (1971).
10. Ramirez, W.F. and Vestal, C.R.: *Chem. Eng. Sci.*, **27,** 2243 (1972).
11. Book, N.L. and Ramirez, W.F.: *AIChE J.*, **22,** 55 (1976).
12. Book, N.L. and Ramirez, W.F.: *AIChE J.*, **30,** 609 (1984).
13. Mattione, M.J.K., Meier W.J. and Book, N.L.: AIChE Symposium Series, No. 214, Vol. 78, 29 (1982).
14. Christensen, J.H.: *AIChE J.*, **16,** 177 (1970).
15. Westerberg, A.W., Hutchison, H.P. and Mortard, R.L. and Winter, P.: "Process Flowsheeting", pp.

77-101, Cambridge University Press, London (1979).
16. Evans, L.B.: "New Developments in Modeling, Simulation and Optimization of Chemical Processes" (lecture notes for special summer program of M.I.T.), Section 2 (1980).
17. Shacham, M.: *AIChE J.*, **30,** 92 (1984).