Identifying Dominant Behavior Patterns, Links and Loops Automated Eigenvalue Analysis of System Dynamics Models

Ahmed A. AbdelGawad
Information and Decision Support Center, the Cabinet
Cairo, Egypt
ahgawad@idsc.net.eg

Bahaa E. Aly Abdel Aleem
Information Technology Institute,
Giza, Egypt
baleem@iti.net.eg

Mohamed M. Saleh
Faculty of Computers and Information, Cairo University
Cairo, Egypt
m.saleh@fci-cu.edu.eg

Pål I. Davidsen
Department of Information Science, Bergen University
Bergen, Norway
davidsen@ifi.uib.no

Abstract

The method presented in this paper allows for an investigation of how model behavior is created from the underlying model structure and how this behavior feeds back to change the relative significance of the model behavior. The method also allows us to identify the dynamics of the relative significance of the various parameters that governs the gains of the links and loops of the model. The method has been implemented mathematically using Matlab based software developed for the purpose of facilitating an Eigenvalue analysis of models representing complex, dynamic systems. This work is based on control theory as well as the previous work on eigenvalue analysis in the system dynamics[*]*. It summarizes the work by Ahmed AbdelTawab AbdelGawad (2004) and Bahaa E. Aly Abdel Aleem (2004) on automating the process of analyzing the structural origin of various modes of behavior exhibited by a system dynamics model.*

The method outline and code developed in preparation for this paper is publicly available, allows us to analyze the relationship between model structure and behavior and may be implemented as par of any simulation package.

[*] Nathan B. Forrester (1983), Christian C. Kampmann (1996), Mohamed M. Saleh and Pål I. Davidsen (2000).

Eigenvalue Analysis in System Dynamics

System Dynamics Models

In this paper, we outline a method that, based upon the partitioning of model behavior over time, linearization and eigenvalue analysis, enable us to identify the dominant behavior modes and the relative contribution to these modes by the structural parameters, links and loops that govern the behavior of the model. To that end, we first review briefly how we typically may apply linear analysis to non-linear models.

Linear Models

A dynamic system is called linear when the principle of superposition holds. Also if cause and effect are proportional (Ogata, K., 1997). The model is thus said to be linear, if the following condition holds for all equations of its auxiliary variables as well as its net rates:

$$z_h = a_1 x_1 + \cdots + a_{N_x} x_{N_x} + b_1 z_1 + \cdots + b_{N_z} z_{N_z} + c_1 u_1 + \cdots + c_{N_u} u_{N_u} \quad (1)$$

Where: $x_i : i \in \mathbb{Z}^+ \le N_x$[†], $z_j : j \in \mathbb{Z}^+ \le N_z$ and $u_k : k \in \mathbb{Z}^+ \le N_u$ are the level, auxiliary and input[‡] variables respectively, $a_i : i \in \mathbb{Z}^+ \le N_x$, $b_j : j \in \mathbb{Z}^+ \le N_z$ and $c_k : k \in \mathbb{Z}^+ \le N_u$ are constants and N_x, N_z and N_u are the number of level, auxiliary and input variables respectively.

We may express every variable in the model by way of its deviation from a specific operating point, i.e. replace $x_1, \ldots, x_{N_x}$, $z_1, \ldots, z_{N_z}$, $u_1, \ldots$ and u_{N_u} by: $\tilde{x}_1 + \delta x_1, \ldots, \tilde{x}_{N_x} + \delta x_{N_x}$, $\tilde{z}_1 + \delta z_1, \ldots, \tilde{z}_{N_z} + \delta z_{N_z}$, $\tilde{u}_1 + \delta u_1, \ldots$ and $\tilde{u}_{N_u} + \delta u_{N_u}$ respectively. As the δ terms are very small in values, and the symbols with tilde represent the specific initial operating point values, then z_h may be expressed as $\tilde{z}_h + \delta z_h$:

$$\therefore \tilde{z}_h + \delta z_h = (a_1\tilde{x}_1 + \ldots + a_{N_x}\tilde{x}_{N_x} + b_1\tilde{z}_1 + \ldots + b_{N_z}\tilde{z}_{N_z} + c_1\tilde{u}_1 + \ldots + c_{N_u}\tilde{u}_{N_u}) + (a_1\delta x_1 + \ldots + a_{N_x}\delta x_{N_x} + b_1\delta z_1 + \ldots + b_{N_z}\delta z_{N_z} + c_1\delta u_1 + \ldots + c_{N_u}\delta u_{N_u})$$

Taking into consideration that the originally initial operating point should be selected from the behavior trajectory of the $\tilde{z}_h$, i.e. it satisfies the original equation of $\tilde{z}_h$. In mathematical terms that is:

$$\tilde{z}_h = a_1\tilde{x}_1 + \cdots + a_n\tilde{x}_n + b_1\tilde{z}_1 + \cdots + b_m\tilde{z}_m + c_1\tilde{u}_1 + \cdots + c_l\tilde{u}_l$$

$$\therefore \delta z_h = a_1\delta x_1 + \cdots + a_n\delta x_n + b_1\delta z_1 + \cdots + b_m\delta z_m + c_1\delta u_1 + \cdots + c_l\delta u_l \quad (2)$$

Nonlinear Models

In generally we should not expect systems to be linear and we should, consequently, be prepared to develop and analyze nonlinear models (systems representations). Thus we

---

[†] The Set of Positive Integers 1, 2, 3, ... , denoted $\mathbb{Z}^+$ (Weisstein, E. W. (1999) Concise Encyclopedia of Mathematics CD-ROM).

[‡] According to the concept of inputs in the control theory; inputs of the model are these influences (variables) that act on the model from outside and are not affected by what happens inside it (Kheir, Naim A., 1996), exactly this is the definition of the constants in a system dynamics model.

cannot expect equation (1) to hold for all model variables. Rather, we should consider the following, more general form of equations:

$$z_h = f\,(x_1,\ldots,x_{N_x},z_1,\ldots,z_{N_z},u_1,\ldots,u_{N_u}) \tag{3}$$

Where: $f\,(.)$ is a nonlinear function. For most practical purposes, for a model to be successfully analyzed, it must be linear. Thus there is a need for transforming a nonlinear model to a linear version, yet one that step by step, generates the same behavior as the original, non-linear model.

Model Linearization

This transformation (model linearization) processes may be accomplished using a ***Taylor Series*** expansion.

Taking as our point of departure the nonlinear equation (3), and by expressing all variables of the model as a deviation from a particular operating point, i.e. by replacing x_1, … , x_{N_x} , z_1, … , z_{N_z} , u_1, … and u_{N_u} by: $\tilde{x}_1+\delta x_1$, … , $\tilde{x}_{N_x}+\delta x_{N_x}$, $\tilde{z}_1+\delta z_1$, … , $\tilde{z}_{N_z}+\delta z_{N_z}$, $\tilde{u}_1+\delta u_1$, … and $\tilde{u}_{N_u}+\delta u_{N_u}$ respectively as in the case of the linear model, we obtain:

$$\begin{aligned}\tilde{z}_h &= f\,(\tilde{x}_1,\ldots,\tilde{x}_{N_x},\tilde{z}_1,\ldots,\tilde{z}_{N_z},\tilde{u}_1,\ldots,\tilde{u}_{N_u})\\ \therefore \tilde{z}_h+\delta z_h &= f\,(\tilde{x}_1+\delta x_1,\ldots,\tilde{x}_{N_x}+\delta x_{N_x},\\ &\qquad \tilde{z}_1+\delta z_1,\ldots,\tilde{z}_{N_z}+\delta z_{N_z},\\ &\qquad \tilde{u}_1+\delta u_1,\ldots,\tilde{u}_{N_u}+\delta u_{N_u})\end{aligned} \tag{4}$$

Equation (4) could be expanded using Taylor Series:

$$\begin{aligned}\tilde{z}_h+\delta z_h = f\,(&\tilde{x}_1,\ldots,\tilde{x}_{N_x},\tilde{z}_1,\ldots,\tilde{z}_{N_z},\tilde{u}_1,\ldots,\tilde{u}_{N_u})\\ &+\left.\frac{\partial f}{\partial x_1}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}\delta x_1+\ldots+\left.\frac{\partial f}{\partial x_{N_x}}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}\delta x_{N_x}\\ &+\left.\frac{\partial f}{\partial z_1}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}\delta z_1+\ldots+\left.\frac{\partial f}{\partial z_{N_z}}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}\delta z_{N_z}\\ &+\left.\frac{\partial f}{\partial u_1}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}\delta u_1+\ldots+\left.\frac{\partial f}{\partial u_{N_u}}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}\delta u_{N_u}\\ &+H.O.T.\end{aligned}$$

Where: $\tilde{\mathbf{x}}=\begin{bmatrix}\tilde{x}_1\\ \tilde{x}_2\\ \vdots\\ \tilde{x}_{N_x}\end{bmatrix}$, $\tilde{\mathbf{z}}=\begin{bmatrix}\tilde{z}_1\\ \tilde{z}_2\\ \vdots\\ \tilde{z}_{N_z}\end{bmatrix}$, $\tilde{\mathbf{u}}=\begin{bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \vdots\\ \tilde{u}_{N_u}\end{bmatrix}$ and $H.O.T.$ is the total amount of the higher order terms, taking into consideration that when the δ terms are infinitesimal, the higher order terms beyond the first order terms will have very small values, and may be ignored compared to those of the first order terms.

Also as $\tilde{z}_h = f\,(\tilde{x}_1,\ldots,\tilde{x}_{N_x},\tilde{z}_1,\ldots,\tilde{z}_{N_z},\tilde{u}_1,\ldots,\tilde{u}_{N_u})$, the last equation may be reduced to:

$$\delta z_h = \sum_{i=1}^{N_x} \left.\frac{\partial f}{\partial x_i}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta x_i + \sum_{j=1}^{N_z} \left.\frac{\partial f}{\partial z_j}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta z_j + \sum_{k=1}^{N_u} \left.\frac{\partial f}{\partial u_k}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta u_k \qquad (5)$$

Where: $\left.\frac{\partial f}{\partial x_i}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} : i \in \mathbb{Z}^+ \le N_x$, $\left.\frac{\partial f}{\partial z_j}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} : j \in \mathbb{Z}^+ \le N_z$ and $\left.\frac{\partial f}{\partial u_k}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} : k \in \mathbb{Z}^+ \le N_u$ are all constants, and could be replaced by $a_i : i \in \mathbb{Z}^+ \le N_x$, $b_j : j \in \mathbb{Z}^+ \le N_z$ and $c_k : k \in \mathbb{Z}^+ \le N_u$ respectively, resulting in

$$\delta z_h = a_1 \delta x_1 + \cdots + a_{N_x} \delta x_{N_x} + b_1 \delta z_1 + \cdots + b_{N_z} \delta z_{N_z} + c_1 \delta u_1 + \cdots + c_{N_u} \delta u_{N_u} \qquad (6)$$

By comparing equations (3) and (6) – taking into consideration that the δ terms represents a small deviation (change) from the original term; i.e. still expressing the original term if the initial value of the original term is exactly known – then the nonlinear relation may be replaced by a linear one.

State Space Form

At this point, we want to transform the state equations of the linear (linearized) model into a general matrix (state space) form for the purpose of analysis. By applying equation (5) to the h[th] element in the net rate vector, the net rate may be expressed as a polynomial of the first degree of auxiliary variables:

$$\delta \dot{x}_h = \sum_{i=1}^{N_x} \left.\frac{\partial f}{\partial x_i}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta x_i + \sum_{j=1}^{N_z} \left.\frac{\partial f}{\partial z_j}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta z_j + \sum_{k=1}^{N_u} \left.\frac{\partial f}{\partial u_k}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta u_k \qquad (7)$$

Taking into consideration the fact that $h \in \mathbb{Z}^+ \le N_x$, the three summations constitute the results of matrix multiplications so that equation (7) can be rewritten to be:

$$\delta \dot{\mathbf{x}} = \mathbf{J}_{\dot{\mathbf{x}},\mathbf{x}}\big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta \mathbf{x} + \mathbf{J}_{\dot{\mathbf{x}},\mathbf{z}}\big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta \mathbf{z} + \mathbf{J}_{\dot{\mathbf{x}},\mathbf{u}}\big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta \mathbf{u} \qquad (8)$$

Where: $\delta \dot{\mathbf{x}} = \begin{bmatrix} \delta \dot{x}_1 \\ \delta \dot{x}_2 \\ \vdots \\ \delta \dot{x}_{N_x} \end{bmatrix}$, $\delta \mathbf{x} = \begin{bmatrix} \delta x_1 \\ \delta x_2 \\ \vdots \\ \delta x_{N_x} \end{bmatrix}$, $\delta \mathbf{z} = \begin{bmatrix} \delta z_1 \\ \delta z_2 \\ \vdots \\ \delta z_{N_z} \end{bmatrix}$ and $\delta \mathbf{u} = \begin{bmatrix} \delta u_1 \\ \delta u_2 \\ \vdots \\ \delta u_{N_u} \end{bmatrix}$ are the deviations in the net rates, level variables, auxiliary variables and the input variables vectors respectively.

Also, $\mathbf{J}_{\dot{\mathbf{x}},\mathbf{x}}\big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} = \left.\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{x}}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}$, $\mathbf{J}_{\dot{\mathbf{x}},\mathbf{z}}\big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} = \left.\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{z}}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}$ and $\mathbf{J}_{\dot{\mathbf{x}},\mathbf{u}}\big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} = \left.\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{u}}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}$, and those $\mathbf{J}$'s are called the ***Jacobian***[§].

By applying equation (5) to the g[th] element in the auxiliary variables vector:

[§] Named after the German mathematician Carl G. J. Jacobi (Kreyszig, E., 1993).

$$\mathbf{J} = \frac{\partial([x, y])}{\partial([u, v])} = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{bmatrix}$$

$$\delta z_g = \sum_{i=1}^{N_x} \left.\frac{\partial f}{\partial x_i}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta x_i + \sum_{j=1}^{N_z} \left.\frac{\partial f}{\partial z_j}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta z_j + \sum_{k=1}^{N_u} \left.\frac{\partial f}{\partial u_k}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta u_k \tag{9}$$

And again, $g \in \mathbb{Z}^+ \le N_x$ so that the three summations constitute the results of matrix multiplications again. As a result, equation (9) can be rewritten to be:

$$\delta\mathbf{z} = \mathbf{J}_{\mathbf{z},\mathbf{x}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta\mathbf{x} + \mathbf{J}_{\mathbf{z},\mathbf{z}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta\mathbf{z} + \mathbf{J}_{\mathbf{z},\mathbf{u}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta\mathbf{u} \tag{10}$$

Where: $\mathbf{J}_{\mathbf{z},\mathbf{x}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} = \left.\frac{\partial\mathbf{z}}{\partial\mathbf{x}}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}$, $\mathbf{J}_{\mathbf{z},\mathbf{z}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} = \left.\frac{\partial\mathbf{z}}{\partial\mathbf{z}}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}$ and $\mathbf{J}_{\mathbf{z},\mathbf{u}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} = \left.\frac{\partial\mathbf{z}}{\partial\mathbf{u}}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}$.

Equations (8) and (10) could be merged into the following form:

$$\begin{bmatrix}\delta\dot{\mathbf{x}}\\ \hline \delta\mathbf{z}\end{bmatrix} = \left.\begin{bmatrix}\mathbf{J}_{\dot{\mathbf{x}},\mathbf{x}} & \mathbf{J}_{\dot{\mathbf{x}},\mathbf{z}}\\ \hline \mathbf{J}_{\mathbf{z},\mathbf{x}} & \mathbf{J}_{\mathbf{z},\mathbf{z}}\end{bmatrix}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \begin{bmatrix}\delta\mathbf{x}\\ \hline \delta\mathbf{z}\end{bmatrix} + \left.\begin{bmatrix}\mathbf{J}_{\dot{\mathbf{x}},\mathbf{u}}\\ \hline \mathbf{J}_{\mathbf{z},\mathbf{u}}\end{bmatrix}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta\mathbf{u} \tag{11}$$

As the matrix $\left.\begin{bmatrix}\mathbf{J}_{\dot{\mathbf{x}},\mathbf{x}} & \mathbf{J}_{\dot{\mathbf{x}},\mathbf{z}}\\ \hline \mathbf{J}_{\mathbf{z},\mathbf{x}} & \mathbf{J}_{\mathbf{z},\mathbf{z}}\end{bmatrix}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}$ relates all the variables of the model to each other using their gain values, it is called the ***System Jacobian*** (Kampmann, C. E., 1996), or it could be called the ***Full Gain Matrix*** after the full version of the model – in contrast to the ***Compact Gain Matrix*** of the compact version of the model (Saleh, M.; Davidsen, P. I., 2000) – and because it contains the gains of all model links.

Also, from equation (10);

$$\therefore \delta\mathbf{z} = (\mathbf{I} - \mathbf{J}_{\mathbf{z},\mathbf{z}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}})^{-1} \mathbf{J}_{\mathbf{z},\mathbf{x}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta\mathbf{x} + (\mathbf{I} - \mathbf{J}_{\mathbf{z},\mathbf{z}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}})^{-1} \mathbf{J}_{\mathbf{z},\mathbf{u}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} \delta\mathbf{u}$$

By substituting into equation (8), we obtain;

$$\therefore \delta\dot{\mathbf{x}} = \left(\mathbf{J}_{\dot{\mathbf{x}},\mathbf{x}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} + \mathbf{J}_{\dot{\mathbf{x}},\mathbf{z}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} (\mathbf{I} - \mathbf{J}_{\mathbf{z},\mathbf{z}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}})^{-1} \mathbf{J}_{\mathbf{z},\mathbf{x}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}\right)\delta\mathbf{x}$$
$$+ \left(\mathbf{J}_{\dot{\mathbf{x}},\mathbf{z}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} (\mathbf{I} - \mathbf{J}_{\mathbf{z},\mathbf{z}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}})^{-1} \mathbf{J}_{\mathbf{z},\mathbf{u}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} + \mathbf{J}_{\dot{\mathbf{x}},\mathbf{u}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}\right)\delta\mathbf{u}$$

By defining $\mathbf{A} = \mathbf{J}_{\dot{\mathbf{x}},\mathbf{x}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} + \mathbf{J}_{\dot{\mathbf{x}},\mathbf{z}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} (\mathbf{I} - \mathbf{J}_{\mathbf{z},\mathbf{z}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}})^{-1} \mathbf{J}_{\mathbf{z},\mathbf{x}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}$

and $\mathbf{B} = \mathbf{J}_{\dot{\mathbf{x}},\mathbf{z}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} (\mathbf{I} - \mathbf{J}_{\mathbf{z},\mathbf{z}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}})^{-1} \mathbf{J}_{\mathbf{z},\mathbf{u}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}} + \mathbf{J}_{\dot{\mathbf{x}},\mathbf{u}}\Big|_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}$, then

$$\delta\dot{\mathbf{x}} = \mathbf{A}\delta\mathbf{x} + \mathbf{B}\delta\mathbf{u} \tag{12}$$

In control theory context, the matrix $\mathbf{A}$ is called the ***System Matrix***, while in a system dynamics context; it would be the ***Compact Gain Matrix*** as stated previously. The matrix $\mathbf{B}$ is called the ***Input Matrix*** or ***Control Matrix***.

We may use the same method as above to identify;

$$\delta\mathbf{y} = \mathbf{C}\delta\mathbf{x} + \mathbf{D}\delta\mathbf{u} \tag{13}$$

Where $\delta\mathbf{y} = \begin{bmatrix} \delta y_1 \\ \delta y_2 \\ \vdots \\ \delta y_{N_y} \end{bmatrix}$ is the output[**] vector. In a control theory context, $\mathbf{C}$ and $\mathbf{D}$ are the ***Output Matrix*** and ***Feedforward Matrix*** respectively.

Also, $\mathbf{C} = \mathbf{J}_{\mathbf{y},\mathbf{x}}\Big|_{\bar{\mathbf{x}},\bar{\mathbf{z}},\bar{\mathbf{u}}} = \dfrac{\partial \mathbf{y}}{\partial \mathbf{x}}\Big|_{\bar{\mathbf{x}},\bar{\mathbf{z}},\bar{\mathbf{u}}}$ and $\mathbf{D} = \mathbf{J}_{\mathbf{y},\mathbf{u}}\Big|_{\bar{\mathbf{x}},\bar{\mathbf{z}},\bar{\mathbf{u}}} = \dfrac{\partial \mathbf{y}}{\partial \mathbf{u}}\Big|_{\bar{\mathbf{x}},\bar{\mathbf{z}},\bar{\mathbf{u}}}$.

Going back to equation (12), and dividing both sides by δt that represents a very small time change, taking limits on both sides assuming δt approaches $\mathbf{0}$,

$$\lim_{\delta t \to 0} \frac{\delta \dot{\mathbf{x}}}{\delta t} = \mathbf{A} \lim_{\delta t \to 0} \frac{\delta \mathbf{x}}{\delta t} + \mathbf{B} \lim_{\delta t \to 0} \frac{\delta \mathbf{u}}{\delta t}$$

From the definition of differentiation:

$$\therefore \ddot{\mathbf{x}} = \mathbf{A}\dot{\mathbf{x}} + \mathbf{B}\dot{\mathbf{u}} \tag{14}$$

At this stage, we assume that $\mathbf{A}$ and $\mathbf{B}$ are constants and do not include any functions of time. Under normal model simulation conditions, the $\mathbf{u}$ vector is a constant vector implying that $\dot{\mathbf{u}}$ equals $\mathbf{0}$.

$$\therefore \ddot{\mathbf{x}} = \mathbf{A}\dot{\mathbf{x}} \tag{15}$$

Equation (15) represents a homogeneous system of linear simultaneous differential equations of the first order. This system may be solved to find some analytical expression for the $\dot{\mathbf{x}}$ vector. To solve such a system we need to find out the characteristic equation and the eigenvalues of the system.

Eigenvalues and the Characteristic Equation

Eigenvalues

Eigenvalues[††] are a special set of scalars (real or complex numbers) associated with a linear system of equations (the linear or linearized system of equations of the model in a matrix form – equation (15)). They are also known as the characteristic roots or proper values, or latent roots (Kreyszig, E., 1993) (Weisstein, E. W. (1999) (Concise Encyclopedia of Mathematics CD-ROM). The eigenvalues are computed as the roots of the characteristic equation.

The Characteristic Equation

The characteristic equation (polynomial) is the equation that is solved to find a matrix's eigenvalues. From equation (12), the matrix $\mathbf{A}$ is a matrix of a system of linear equations. If there is a vector $\mathbf{r} \neq \mathbf{0}$ such that;

$$\mathbf{A}\mathbf{r} = \lambda\mathbf{r} \tag{16}$$

[**] Output variables are observable quantities and are measurable (Kheir, Naim A., 1996). In system dynamics model this definition is still valid, and outputs are always the choice of the user or the modeler. They can be any collection of the variables of the model.

[††] In German it is called Eigenwert, "Eigen" is a German word that means Proper, while "wert" means Root (Kreyszig, E., 1997).

Where λ is a scalar value, then λ is one of the eigenvalues and $\mathbf{r}$ is its corresponding right eigenvector (called right because the vector is multiplied to the right of matrix $\mathbf{A}$, and it would have been the left eigenvector if the multiplication had been to the left of matrix $\mathbf{A}$). To compute the eigenvalues and their corresponding right eigenvectors, equation (16) could be reduced to:

$$\therefore (\mathbf{A} - \lambda\mathbf{I})\mathbf{r} = \mathbf{0}$$

Using Cramer's Rule, a system of linear equations has nontrivial solutions only if the determinant of the system vanishes, so we obtain the characteristic equation (Kreyszig, E., 1993):

$$|\mathbf{A} - \lambda\mathbf{I}| = \mathbf{0} \tag{17}$$

This equation has solutions that equal the number of rows or columns of the $\mathbf{A}$ matrix. The set of all solutions of equation (17) is the set of eigenvalues. By taking each eigenvalue and substituting in equation (16), we obtain its corresponding eigenvector.

The State Space Form Solution

The solution of equation (15) could be on the following form (Kreyszig, E., 1993):

$$\dot{\mathbf{x}} = c_1 e^{\lambda_1 (t - t_0)}\mathbf{r}_1 + c_2 e^{\lambda_2 (t - t_0)}\mathbf{r}_2 + \cdots + c_n e^{\lambda_n (t - t_0)}\mathbf{r}_{N_x} \tag{18}$$

Where: c_1, c_2, … and c_{N_x} are constants, $\mathbf{r}_1$, $\mathbf{r}_2$, … and $\mathbf{r}_{N_x}$ are the right eigenvectors[‡‡] of the system and λ_1, λ_2, … and λ_{N_x} are their corresponding eigenvalues. The constant term c_i may be computed using the initial conditions at $t = t_0$, $\dot{\mathbf{x}} = \tilde{\dot{\mathbf{x}}}$. Note that the $\tilde{\dot{\mathbf{x}}}$ vector is very well-known at every time step. Thereafter, at every new time step, the vector $\tilde{\dot{\mathbf{x}}}$ is known from the pervious step–, substituting in equation (18):

$$\therefore \tilde{\dot{\mathbf{x}}} = \mathbf{r}_1 c_1 + \mathbf{r}_2 c_2 + \cdots + \mathbf{r}_{N_x} c_{N_x}$$

The last formula expresses the matrix product of a vector containing all the c_i terms and the full right eigenvector $\mathbf{r} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \cdots & \mathbf{r}_{N_x} \end{bmatrix}$[§§].

$$\therefore \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{N_x} \end{bmatrix} = \mathbf{r}^{-1}\tilde{\dot{\mathbf{x}}} \tag{19}$$

By utilizing the phasor form of complex number (Edminister, Joseph A., 1972), and looking back at the analytical solution of the $\dot{\mathbf{x}}$ vector, equation (18), we may note that:

$$\because e^{\lambda_i (t - t_0)} = e^{(\sigma_i \pm j\omega_i)(t - t_0)}$$

$$\therefore e^{\lambda_i (t - t_0)} = e^{\sigma_i (t - t_0)} e^{\pm j\omega_i (t - t_0)}$$

Equation (18) could be rewritten as:

[‡‡] These set of eigenvectors are should be linearly independent or their corresponding eigenvalues are different.

[§§] Although $\mathbf{r}$ is a lower case letter, it is used to express the right eigenvectors matrix; because this matrix is the arrangement of right eigenvectors beside each other in columns. Also $\mathbf{l}$ would be used ro express the left eigenvalues matrix.

$$\therefore \dot{\mathbf{x}} = \sum_{i=1}^{N_x} c_i e^{\sigma_i (t-t_0)} e^{\pm j\omega_i (t-t_0)} \mathbf{r}_i$$

Note that $\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_{N_x} \end{bmatrix}$, also $\mathbf{r}_i = \begin{bmatrix} r_{i1} \\ \vdots \\ r_{iN_x} \end{bmatrix}$, so that we may write only the k[th] net rate:

$$\therefore \dot{x}_k = \sum_{i=1}^{N_x} c_i e^{\sigma_i (t-t_0)} e^{\pm j\omega_i (t-t_0)} r_{ki} \tag{20}$$

Where: c_i is a constant and we may multiply it to $\mathbf{r}_i$ (that contains constant terms) that results in another vector of constants $c_i \mathbf{r}_i = \begin{bmatrix} c_i r_{i1} \\ \vdots \\ c_i r_{iN_x} \end{bmatrix} = \begin{bmatrix} c_{i1} \\ \vdots \\ c_{iN_x} \end{bmatrix}$:

$$\therefore \dot{x}_k = \sum_{i=1}^{N_x} c_{ik} e^{\sigma_i (t-t_0)} e^{\pm j\omega_i (t-t_0)} \tag{21}$$

The behavior of one net rate is a linear combination of terms, where each term is associated with one of the eigenvalues. This means that all eigenvalues of the system have a specific effect on every net rate, characterized by an amplification value. Note that the exponential term is the sources of the behavior (dynamics) of the model.

Identifying the Dominant Eigenvalue

The entire behavior of a level variable is the sum of the modes of behavior corresponding to the eigenvalues of the model, each of which contributes to a fraction of that behavior. So that it is possible to specify one eigenvalue (or a conjugate pair of such values if the eigenvalue has imaginary part) or more that most predominantly affect the behavior of that level variable.

The identification process of the dominant eigenvalue depends mainly on an experimental method suggested by Saleh, M. and Davidsen, P. (2000, 2001). This method relies on empirical experiments in which only one eigenvalue is allowed to affect the behavior while blocking all the other eigenvalues. Thus one is able to identify the dominant eigenvalue.

This method might be improved by computing the percentage of contribution of each eigenvalue to the level under study, a computation that would allow arranging them according to their dominance over the level behavior. Thus it would be possible to operate with more than only one dominant eigenvalue. Back to equation (21), that may be rewritten to say:

$$\dot{x}_k = c_{1k} e^{\lambda_1 (t-t_0)} + c_{2k} e^{\lambda_2 (t-t_0)} + \cdots + c_{nk} e^{\lambda_n (t-t_0)}$$

Or,

$$\dot{x}_k = \dot{x}_{k1} + \dot{x}_{k2} + \cdots + \dot{x}_{kn}$$

Where $\dot{x}_{ki} = c_{ik} e^{\lambda_i (t-t_0)}$, by applying superposition. Note that each $\dot{x}_{ki}$ presents the effect of only the i[th] eigenvalue λ_i on the total $\dot{x}_k$:

By integrating both sides of the last equation, there are two cases:

If $\lambda_i \neq 0$:

$$\delta x_{ki} = \frac{c_{ik}}{\lambda_i}\left(e^{\lambda_i \delta t} - 1\right) \tag{22}$$

If $\lambda_i = 0$:

$$\delta x_{ki} = c_{ik}\,\delta t \tag{23}$$

Where $\delta x_{ki} = x_{ki} - \tilde{x}_{ki}$, by calculating the term δx_{ki} for each eigenvalue, it would be possible to distinguish the contribution of each eigenvalue in the behavior of x_k , where:

$$\therefore \text{contribution}_{ki} = \frac{\delta x_{ki}}{\delta x_k} \tag{24}$$

Where: contribution_{ki} is the contribution in the behavior of the k[th] state due to λ_i only, also we should note that the term $\delta x_k = \sum_{i=1}^{n} \delta x_{ki}$ expresses the total contribution of all eigenvalues in the value of the k[th] state.
By arranging the values of contribution of each eigenvalue in a descending order, it is possible to identify the dominance order of those eigenvalues. In this context primarily the most dominant eigenvalue would be considered, - i.e. the eigenvalue with the most significant contribution, yet one may test the effect of the second and/or the third and/or any higher order dominant eigenvalue on the behavior of a state.

The Dominant Eigenvalue Elasticity

The aim of the analysis process is to identify the leverage points in the model structure. Therefore, the aim of this section is to relate the dominant eigenvalue to the links (i.e. relationships) of the model using the concept of eigenvalue elasticities as suggested by Forrester, N. (1982).

The Eigenvalue Sensitivity

For a link that starts from a level variable x_j and ends at the net rate of another level variable x_i , the k[th] eigenvalue sensitivity to the gain of that link s_{kij} is defined as the change in the k[th] eigenvalue due to the change in the gain of that link:

$$s_{kij} = \frac{\partial \lambda_k}{\partial a_{ij}} \tag{25}$$

Or, in matrix form,

$$\mathbf{S}_k = \frac{\partial \lambda_k}{\partial \mathbf{A}} \tag{26}$$

The matrix $\mathbf{S}_k$, can be directly computed (Saleh, M., 2003);

$$\mathbf{S}_k = \mathbf{l}_k \mathbf{r}_k^{\,T} \tag{27}$$

Where $\mathbf{l}_k$ and $\mathbf{r}_k$ are the left and right eigenvectors of the k[th] eigenvalue respectively.

The Eigenvalue Elasticity

For a link that starts from a level variable x_j and ends at the net rate of another level variable x_i, the k[th] eigenvalue elasticity for the gain of that link E_{kij} [***] is defined as the relative change in the k[th] eigenvalue to the relative change in the gain of that link (Saleh, M., 2003):

$$E_{kij} = \frac{\delta\lambda_k / \lambda_k}{\delta a_{ij} / a_{ij}} \tag{28}$$

$$\therefore E_{kij} = \frac{1}{\lambda_k} \frac{\delta\lambda_k}{\delta a_{ij}} a_{ij}$$

Using the definition from equation (25);

$$\therefore E_{kij} = \frac{1}{\lambda_k} s_{kij} a_{ij} \tag{29}$$

Or, in matrix form;

$$\therefore \mathbf{E}_k = \frac{1}{\lambda_k} \mathbf{S}_k \,.*\mathbf{A} \text{ [†††]} \tag{30}$$

Where $\mathbf{E}_k$ is the k[th] eigenvalue elasticity matrix for the compact version of the model.

The Dominant Eigenvalue Elasticity Values of the Links of the Compact Model

By applying equation (30) to compute the dominant eigenvalue elasticity values of the links of the compact model we may relate the system behavior to the links of the compact version of the system dynamics model, i.e. the links between state variables and net rate variables.

At this point an interesting property of the eigenvalue elasticity measure – noted by Forrester, N. - should be restated: The eigenvalue elasticity values is like electric current, that is, all eigenvalue elasticity values entering a variable in the model should leave that variable as well (Forrester, N. B., 1982). This property was proven to exist by Saleh, M. (Saleh, M., 2003) –like: Kirchoff Current Law, in Electric Circuits (Edminister, Joseph A., 1972). This interesting property greatly helps distributing the dominant eigenvalue elasticity value of the link between two variables in the compact version of the model, among the links between the same two variables, in the full version of the model.

[***] The paper's normal mathematical symbolic notation would be contradicted for the elasticity matrix elements; upper case letters would be used instead of lower case letters, in order not to confuse the reader with the exponent function.

[†††] The symbol $.*$ is the Mathworks' Matlab notation for array multiplication operator (each element from matrix to the left of the operator is multiplied by the corresponding element from the matrix to the right of the operator).

The Dominant Eigenvalue Elasticity Values of the Links of the Full Model

Returning to equation (11), the full gain matrix is $\left[\begin{array}{c|c} \mathbf{J}_{\dot{x},x} & \mathbf{J}_{\dot{x},z} \\ \hline \mathbf{J}_{z,x} & \mathbf{J}_{z,z} \end{array}\right]_{\tilde{\mathbf{x}},\tilde{\mathbf{z}},\tilde{\mathbf{u}}}$, for any two variables linked in the model, the full gain matrix has a corresponding element that has a value that equals the gain between those two variables, taking into consideration that this element column number is the number of the variable that the link starts from, and that the element row number is the number of the variable that the link ends up at, provided that the variables of the model are numbered. The other elements of the full gain matrix that do not correspond to links in the model take values equal to zero.

In Graph Theory; such a matrix is called a digraph (directed graph) or an adjacency matrix. Various well-know exhaustive search algorithms may be applied on that digraph to find paths between two variables or to find loops, - the details of which are out of this scope of this paper[‡‡‡].

Considering the compact gain matrix and its corresponding computed dominant eigenvalue elasticity values matrix, in the compact version of the model, only variables that have a nonzero element in the compact gain matrix, have a corresponding nonzero element in the compact dominant eigenvalue elasticity values matrix and vice versa; i.e. only variables that has a gain characterizing the relationship between them, have a dominant eigenvalue elasticity value. This is valid because if there is a gain between two variables in compact model, this means that there is a direct or indirect link or links between those two variables in the full version of the model. A zero gain, on the other hand, means that there is no link between the two variables, and thus there would be no dominant eigenvalue elasticity value between them (equation (29)).

Suppose that the kth eigenvalue is the dominant eigenvalue. Using equation (29) we may compute the dominant eigenvalue elasticity value of the link between the level variable x_j and the net rate of the level variable x_i, i.e. E_{kij}, to the level variable x_i directly rather than to the net rate of the level variable x_i, and it would be more simple to drop the level variable statement and directly say x_j and x_i.

Using one of the path identification algorithms to extract paths from the full model, it would be possible to identify the direct and indirect paths that starts from x_j and ends at x_i, excluding paths that pass through other states because the gains and the dominant eigenvalue elasticity values of those paths that pass through other states are included within their original paths. Taking them into consideration within this computation leads to erroneous redundancy. Let those identified paths be P_{ji_1}, P_{ji_2}, … and P_{ji_N} where N is the total number of paths that starts from x_j and ends at x_i, excluding paths that pass through other states:

$$P_{ji} = \left\{ P_{ji_1}, P_{ji_2}, \ldots, P_{ji_N} \right\}$$

[‡‡‡] The interested reader might refer to "Graph Theory" (Diestel, R., 2000), "Advanced Engineering Mathematics" (Kreyszig, E., 1993) and the "Digraph toolbox" (Bahar, M.; Jantzen, J., 1995)

Let the gains of those paths be $g_{P_{ji_1}}$, $g_{P_{ji_2}}$, … and $g_{P_{ji_N}}$ respectively. And let their dominant eigenvalue elasticity values be $E_{kP_{ji_1}}$, $E_{kP_{ji_2}}$, … and $E_{kP_{ji_N}}$ respectively. Note that the sum of the gains and the sum of the dominant eigenvalue elasticity values of those paths together equals a_{ij} and E_{kij} respectively (Saleh, M.; Davidsen, P. I., 2000).

$$a_{ij} = \sum_{P_s \in P_{ji}} g_{P_s}$$

$$E_{kij} = \sum_{P_s \in P_{ji}} E_{kP_s}$$

The gain of each individual path i.e. the values of $g_{P_{ji_1}}$, $g_{P_{ji_2}}$, … and $g_{P_{ji_N}}$ could easily be computed by multiplying the gains of the elements g_{ℓ_r} (links) of each path individually from the full gain matrix (Kuo, B. C., 1995) and (Ogata, K., 1997).

$$g_{P_{ji_s}} = \prod_{\ell_r \in P_{ji_s}} g_{\ell_r}$$

To compute the dominant eigenvalue elasticity value of each path individually (Saleh, M.; Davidsen, P. I., 2000):

$$E_{kP_{ji_s}} = g_{P_{ji_s}} \frac{E_{kij}}{a_{ij}} \tag{31}$$

Or, by utilizing equation (29):

$$E_{kP_{ji_s}} = g_{P_{ji_s}} \frac{s_{kij}}{\lambda_k} \tag{32}$$

Where $E_{kP_{ji_s}}$ is the dominant eigenvalue elasticity value for the path P_{ji_s}. Note that, $E_{kP_{ji_s}}$ is also the dominant eigenvalue elasticity value for every element in the path P_{ji_s}. Also, it is important to note that one link ℓ_r could be a member of more than one path in the full version of the model and each of those paths has its distinct dominant eigenvalue elasticity value. In this case, its dominant eigenvalue elasticity value of that link is the sum of all dominant eigenvalue elasticity values of all paths that pass through this link (Forrester, N. B., 1982) and (Saleh, M., 2003).

$$E_{k\ell_r} = \sum_{P_{ji_s} \supset \{\ell_r\}} E_{kP_{ji_s}} \tag{33}$$

The dominant eigenvalue elasticity values for all links of the full version of the model could be computed using equation (33), i.e. the ***Full Dominant Eigenvalue Elasticity Value Matrix***.

Also by utilizing the eigenvalue to gain sensitivity definition, we can find the dominant eigenvalue sensitivity for all links in the full the model:

$$E_{k\ell_r} = s_{k\ell_r} \frac{g_{\ell_r}}{\lambda_k}$$

From equation (32), (33) and as previously indicated; $g_{P_{ji_s}} = \prod_{\ell_v \in P_{ji_s}} g_{\ell_v}$:

$$\therefore s_{k\ell_r} = \lambda_k \sum_{P_{ji_s} \supset \{\ell_r\}} \left(\left(\prod_{\ell_v \in P_{ji_s} - \{\ell_r\}} g_{\ell_v} \right) \frac{E_{kij}}{a_{ij}} \right) \tag{34}$$

The Dominant Eigenvalue Elasticity for the Inputs

To fully benefit from the knowledge of the dominant eigenvalue elasticity values of the system dynamics model links, it should be possible to change the gains of those links so as to change the dominant eigenvalue, i.e. changing the gain of a link so that the eigenvalue and thus the system behavior would change in a desirable way.
As already stated, the model inputs or constants or parameters are the controllable part of the model. Thus it should be possible to compute the dominant eigenvalue elasticity values for the model inputs. Utilizing the following relationship (Saleh, M., 2003);

$$E_{ku_q} = \frac{\delta\lambda_k / \lambda_k}{\delta u_i / u_q} \tag{35}$$

Where E_{ku_q} is the dominant eigenvalue elasticity value for the input u_q.
By rearranging the terms of equation (35);

$$\therefore E_{ku_q} = \frac{1}{\lambda_k}\frac{\partial\lambda_k}{\partial u_q}u_q$$

Applying the Chain Rule (Kreyszig, E., 1993), yields;

$$E_{ku_q} = \frac{1}{\lambda_k}\left(\sum_{r=1}^{N_\ell}\frac{\partial\lambda_k}{\partial g_{\ell_r}}\frac{\partial g_{\ell_r}}{\partial u_q}\right)u_q$$

Where N_ℓ is the number of all links in the full model.

From the definition of the eigenvalue to gain sensitivity $s_{k\ell_r} = \frac{\partial\lambda_k}{\partial g_{\ell_r}}$:

$$\therefore E_{ku_q} = \frac{1}{\lambda_k}\left(\sum_{r=1}^{N_\ell} s_{k\ell_r}\frac{\partial g_{\ell_r}}{\partial u_q}\right)u_q \tag{36}$$

Where $s_{k\ell_r}$ can be computed directly using equation (34).

The Dominant Eigenvalue Elasticity for the Loops

The loops of a model are among the most meaningful building blocks in system dynamics. As already stated, the full gain matrix is a digraph, and could be searched to find paths between two variables, and loops by identifying paths that starts from a variable and ends at the same variable. By identifying loops, their gains and dominant eigenvalue elasticities may be identified.

A.1.1.1 Identifying Loops in the Model

Kampmann, C. E. (1996) suggested a binary matrix that relates the links with the loops:

$$\begin{bmatrix}\ell_1\\ \ell_2\\ \vdots\\ \ell_{N_\ell}\end{bmatrix} = \mathbf{C}\begin{bmatrix}\kappa_1\\ \kappa_2\\ \vdots\\ \kappa_{N_\kappa}\end{bmatrix} \tag{37}$$

Where κ_i expresses the i[th] loop, ℓ_j expresses the j[th] link. N_κ and N_ℓ are the number of all loops and all links in the model respectively. The matrix $\mathbf{C}$ would be a non-square binary matrix:

$$\mathbf{C} = \left[c_{ij} \right]$$

Where $c_{ij} = 1$ if the link ℓ_j is a component in loop κ_i, **0** otherwise.

A.1.1.2 Linearly Independent Loops and their Dominant Eigenvalue Elasticity Values

As stated above, Forrester, N. B. (1982) discovered the similarity between eigenvalue elasticity and an electric current. Also Kampmann, C. E. (1996) suggested equation (37), but he stated that solving this equation in its form would not be possible. Moreover, he added (from the graph theory) that, for this equation to be solvable, the set of all loops should be replaced by a smaller set of (*total number of links – total number of variables* + 1) loops. Note that this is exactly the number of any selected linearly independent loop set.
So that in equation (37), by replacing the $\mathbf{C}$ with a smaller matrix $\mathbf{C}_r$, to relate the eigenvalue elasticity values of links with that of a linearly independent loop set of loops.

$$\begin{bmatrix} E_{k\ell_1} \\ E_{k\ell_2} \\ \vdots \\ E_{k\ell_{N_\ell}} \end{bmatrix} = \mathbf{C}_r \begin{bmatrix} E_{k\kappa_1} \\ E_{k\kappa_2} \\ \vdots \\ E_{k\kappa_{N_\kappa}} \end{bmatrix} \tag{38}$$

Where $E_{k\kappa_i}$ and $E_{k\ell_j}$ express the dominant eigenvalue elasticity values of the i[th] loop and the j[th] link respectively.
Or, in matrix form:

$$\mathbf{E}_{k\ell} = \mathbf{C}_r \mathbf{E}_{k\kappa} \tag{39}$$

Equation (39) could easily be solved for $\mathbf{E}_{k\kappa}$ using least squares solution. But the real problem is how to select the matrix $\mathbf{C}_r$ from the rows of the matrix $\mathbf{C}$, in other words; how to select the linearly independent loops set.

> *"The rank of a matrix A is the maximum number of linearly independent columns of A; or it is the order of the largest nonsingular matrix contained in A."*
> *–Kuo, B. C. (1995).*

This makes it easy to find identify $\mathbf{C}_r$, knowing that it is not unique for the model, i.e. there could be more than one linearly independent loops set (Kampmann, C. E., 1996). Al Also Kampmann suggested that the user should select the most significant set for his model from the user's point of view.
Referring to a simple yeast cells model, - by computing the rank of the matrix $\mathbf{C}$; it equals 4. This is while the total number of links equals 10 and the total number of variables equals 7, so that the number of linearly independent loops set, should be 10 – 7 + 1 = 4, which is the same result of the rank. This also means that all loops in the model are linearly independent, i.e. $\mathbf{C}_r = \mathbf{C}$.

Thus by substituting into equation (38) and solving it for the eigenvalue elasticity values of the loops, this system of equations constitutes an over determined system; i.e. the number of equations is greater than the number of unknowns. But it is still a consistent system that could be solved and give exact values to the unknowns.

Appendix:
Eigenvalue Analysis: Computer Implementation

Introduction

This chapter focuses on the implementation of the functions of the Simulation package[§§§] and Analysis package[****] using the programming language of Mathworks Matlab mathematical package.
The Simulation package functions deal directly with the model file –in Powersim constructor text file format; parse the model, arrange its equations and simulate, then give their outputs to the Analysis package.
The Analysis functions aim at applying the eigenvalue analysis steps on system dynamics model parsed and simulated using the Simulation package, and print out its outputs into a text file.
Shown in figure 1; the context level diagram of both packages together, which declares the relation among the main entities and both packages as a single process. While in figure 2; the data flow diagram (DFD) level zero of them both too; which declares the relation and data flow between them in more details.

The Main Function

The starting point of the script is the "main" function. This function is called with the following command:

```
main( inFileName , outFileName , initialTime , finalTime , timeStepLength )
```

The "main" function has five input arguments:
The "***inFileName***" which specifies the model's equations file name, a Powersim model saved as equations, which is normally a text file.
The "***outFileName***" which specifies the file containing the results of applying the script to a certain model. The output file is created by the script and it is normally a text file too.
The next three arguments specify the simulation start time, the simulation end time and the simulation time step respectively; these three parameters are used by the simulation module.

[§§§] The Simulation package was implemented by Bahaa El-Din Ali Abdel-Aleem as a technical part of his master thesis – Bergen University (2004).
[****] The Analysis package was implemented by Ahmed AbdelTawab AbdelGawad as a technical part of his master thesis – Bergen University (2004).

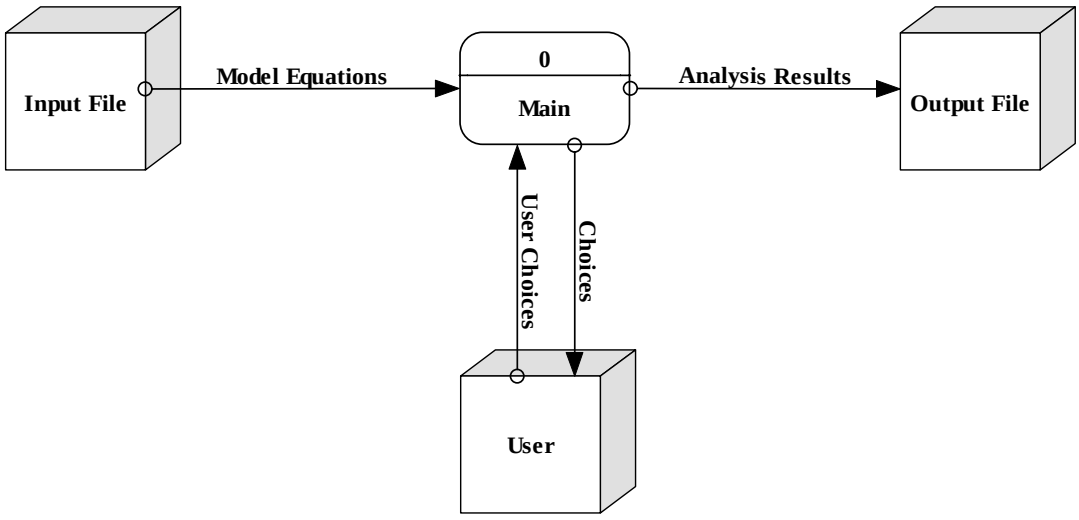

Figure 1: The Context Level Diagram of Both Simulation and Analysis Packages Together

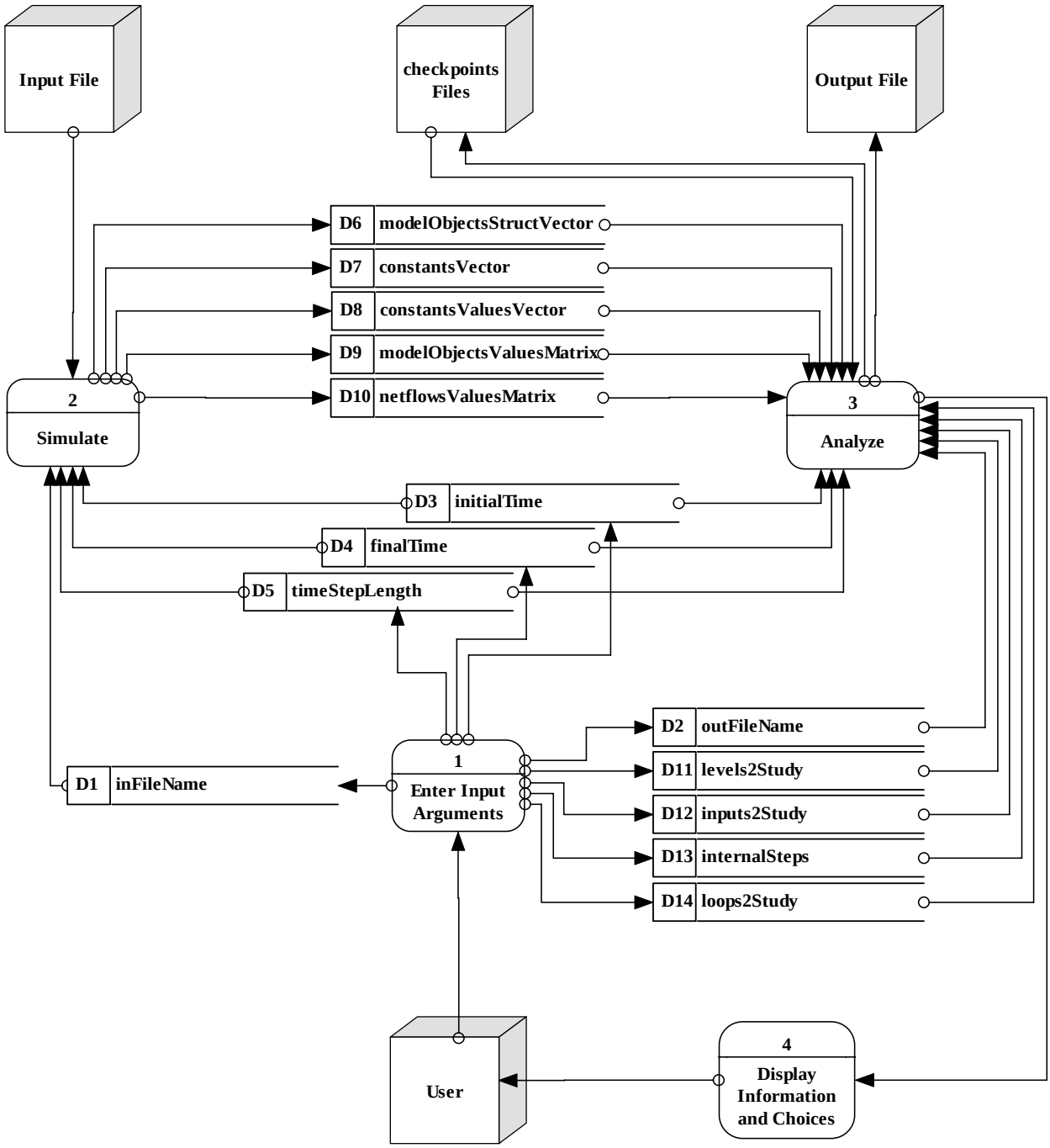

Figure 2: The Data Flow Diagram (DFD) Level Zero: Simulation and Analysis Packages

The Simulation Package

The Parsing Process

In this part we will describe the model parsing module. The input to the parsing module is a Powersim model equations file, which is a Powersim model file saved in text format. Powersim model files when saved in text format have one general structure. If we take a closer look into that structure we will find that the file consists of four main sections. These sections are:
Ranges.
Independent Variables.
Dependent Variables.
Flows.
Next, we will show how the parsing module handles these files and what the form of its output is.
The whole parser module is written on the form of a single function. This function takes the model's text file name as its input argument and it returns an array of three vectors as its output.

```
[ modelObjectsStructVector , constantsVector , constantsValuesVector ] =
parser( inFileName )
```

These three vectors, modelObjectsStructVector, constantsVector and constantsValuesVector, are used in the simulation module as well as the analysis module.
To understand how the parser module generates its output from the model's text file. We have to study the three variables returned by the parser function first. The first of the variables returned by the parser function is the modelObjectsStructVector. The modelObjectsStructVector is an array of vectors which holds the state and auxiliary variables of the model, each in a separate vector:

```
modelObjectsStructVector = [levelsStructVector, auxiliariesStructVector]
```

Each of the levelsStructVector and the auxiliariesStructVector is an array of structures with the following member variables:
name: an array which holds variables' names.
equation: an array which holds variables' equations.
value: an array which holds variables' initial values.
state: a flag that takes the value 0 or 1. It distinguishes the state from non state variables. If the variable contained in the structure is a state variable, it takes the value 1; otherwise it takes the value 0.
Thus, the levelsStructVector, holds the names, equations and values of all stock variables of the model, each in a separate array. The names of the stock variables are stored in the "levelsVector" array, the equations of state variables are stored in the "flowsVector" array and finally the values of the state variables are stored in the "levelsInitVector" array, which holds only the numeric stock initial values. If a stock is initialized with a relation not a numeric value, this value is stored in the value member variable of the auxiliaries StructVector explained below.

Also, the names, equations and values of all the auxiliary variables of the model are each stored in a separate array inside the "auxiliariesStructVector". Names of the auxiliaries are stored in the "auxiliariesVector" array; equations of the auxiliaries are stored in the "auxiliariesEquationsVector" array, and finally values of the auxiliaries are stored in the "auxiliariesEquationsVector" array. Note that values and the equations of the auxiliaries are the same, since the values aren't evaluated until the simulation is performed. It should be noted here that auxiliaries mean all variables of the model, other than levels and constants
The next two variables returned by the parser function, namely: constantsVector, constantsValuesVector are two vectors holding the names of the model's constants and the values of the model's constants respectively.
Thus, the parser function reads the model's file and outputs all the variables in a way convenient for other parts of the script to use.

The Equations Arrangement Process

The arrange module does two main functions: It first arranges the variables of the model to put them into the right sequence for simulation; so that there are no unknowns in any certain step of simulation. Second, it calculates the values of all variables at the initial time of simulation so that the simulation can begin correctly.

```
modelObjectsStructVector = arrange( modelObjectsStructVector , constVector , constValVector , initialTime )
```

The above line of code shows the header of the arrange function. The function takes four input arguments; the three vectors returned by the parsing module normally they contain all the model variables and the fourth argument represents the initial time of the simulation. That is because the arrange function will need to calculate the whole model at the initial time of simulation. It returns the modelObjectsStructVector – described earlier - containing the model's variables arranged.

The Simulation Process

As in the previous parts of the script, and generally the formal method for coding, the simulation module is written as a function.

```
[ modelObjectsValuesVector , netflowsValuesMatrix ] = simulate( modelObjectsStructVector , constVector , constValVector , initialTime , finalTime , timeStep )
```

The purpose of the simulation module is to:

- Perform the simulation correctly.
- Generate, at all time steps, the numerical values of all variables of thy system.
- Calculate the slope vector of the system at all time steps.

The above outputs of the simulation module are the main inputs to the analysis module.
As stated before, the modelObjectsStructVector is returned by the arrange module with the variables arranged in the right sequence for simulation and with all values of the variables known at the initial time.
Thus, at the initial step of simulation, the values (from the value field) of the modelObjectsStructVector are substituted into the modelObjectsValuesVector.

```
modelObjectsValuesVector( 1 , : ) = double( [ modelObjectsStructVector.value ] );
```

Also, the netflowsValuesMatrix is calculated by substituting into it from the flowsVector.

```
netflowsValuesMatrix( 1 , : ) =  double( subs( flowsVector , {
modelObjectsStructVector.name } , { modelObjectsStructVector.value } ) );
```

Now, before the simulation loop, in which the values of all variables are calculated through the whole time span of simulation step by step, the values of all constants (since they don't change throughout the simulation) are stored into memory to be available through the simulation time.

```
for I = 1 : length( constVector ),
eval( [ char( constVector( I ) ) '=' num2str( constValVector( I ) ) ';' ] );
end
```

Since the following rule holds true:
Any step variable t+1 = state variable t +net flow * Δt (1)
Thus, in order to calculate the values of the state variables:
The netflowsValuesMatrix is first evaluated:

```
netflowsValuesMatrix( TIME + 1 , : ) = double( subs( flowsVector , {
modelObjectsStructVector.name } , { modelObjectsStructVector.value } ) );
```

Then, all the states of the system are calculated according to equation (1)

```
temp = num2cell( modelObjectsValuesVector( TIME , 1 : numStates ) + timeStep
* netflowsValuesMatrix( TIME + 1 , : ) );
[ modelObjectsStructVector( 1 : numStates ).value ] =  deal( temp{ : } );
modelObjectsValuesVector( TIME + 1 , 1 : numStates ) = [
modelObjectsStructVector( 1 : numStates ).value ];
```

Accordingly, the rest of the model is calculated for the next time step. The simulation loop continues until the specified time span ends. By the end of the simulation module we have its two output vectors modelObjectsValuesVector and netflowsValuesMatrix ready to be used in the analysis module.

The analysis Package

The Analysis function is the backbone function of the Analysis package. It calls all other functions of the package to complete the eigenvalue analysis of system dynamics model.

```
Analysis( modelObjectsStructVector , constantsVector ,  constantsValuesVector ,
modelObjectsValuesMatrix , netflowsValuesMatrix , initialTime , finalTime ,
timeStepLength , outFileName );
```

Follows another question to the user to decide the time steps to do the analysis process at, which is stored as a vector called internalStep.

User-Interactions Processes

The function needs user-interaction to make decisions. The function needs the user to decide the level variable he/she wants to study its behavior. So it prints a list of all level variables to the user, and waits for a choice, this choice is the number of the level to study form the list and it is stored in level2study.
Also, the function needs the user to decide his/her set of inputs out of the set of all constants in the model. So it prints a list of all constants to the user, and waits for a choice, this choice is a vector containg the numbers of the constants to study form the list and it is stored in inputs2study.
Another question has to be given to the user to decide the time steps to do the analysis process at, which is stored as a vector called internalStep.

Building the Full Gain Matrix

The Full Gain Matrix of the model is built in a symbolic form matrix symbolicFullGainMatrix, in the beginning. After that, through the analysis process a substitution is done and the Full Gain Matrix is in a numeric form matrix numericFullGainMatrix.

Computing the Numeric Full Gain Matrix

The numericFullGainMatrix is not computed till we reach what we call the TIME loop, this loop is the one that loops to make the analysis each element of the internalStep.

```
tempSymbolicFullGainMatrix = subs( symbolicFullGainMatrix , sym('TIME') , ( TIME * timeStepLength ) + initialTime );
numericFullGainMatrix = double( subs( tempSymbolicFullGainMatrix , modelObjectsNamesVector , modelObjectsValuesMatrix( TIME , : ) ) );
```

The numericFullGainMatrix is computed by substituting all variables' names with its corresponding simulation values at a specific time step, in the symbolic matrix symbolicFullGainMatrix.

Computing the Compact Gain Matrix

The numericGainMatrix is computed directly after the numericFullGainMatrix is computed.

```
A12 = numericFullGainMatrix( 1 : numStates , numStates + 1 : end );
A21 = numericFullGainMatrix( numStates + 1 : end , 1 : numStates );
A22 = numericFullGainMatrix( numStates + 1 : end , numStates + 1 : end );
numericGainMatrix = A12 * inv( eye( size( A22 ) ) - A22 ) * A21;
```

Computing the Eigenvalues

After the numericGainMatrix is computed. Its Eigenvalues are computed using the Matlab built-in eig function.

```
[ rightEigenVector , diagonalEigenMatrix ] = eig( numericGainMatrix );
```

As appears form the code lines, not only the Eigenvalues are computed and place on the diagonal of diagonalEigenMatrix, but also their corresponding right Eigenvector rightEigenVector.
Directly in the next step, we compute the Eigenvalues corresponding left Eigenvector:

```
leftEigenVector = inv( rightEigenVector ).';
```

Identifying the Dominant Eigenvalue

The dominant Eigenvalue identification is done in another function dominant, the same way stated in the Mathematical Background:

```
[ dominantEigenVector( TIME , : ) , dominantEigenPositionVector( TIME , : ) ,
dominancePercentageVector( TIME , : ) ] = dominant( rightEigenVector ,
leftEigenVector , diagonalEigenMatrix , netflowsValuesMatrix( TIME , : ).' ,
netflowsValuesMatrix( TIME + 1 , : ).' , timeStepLength , level2study , TIME );
```

Computing the Dominant Eigenvalue Compact Elasticity Values Matrix

The cmpElastMat performs the task of computing the dominant Eigenvalue compact elasticity values matrix.

Computing the Dominant Eigenvalue Full Elasticity Values Matrix

The cmpElastMat –the same function of the last section– does the task of computing the dominant Eigenvalue full elasticity values matrix.

Computing the Dominant Eigenvalue Inputs Elasticity Values Vector

Computing the dominant Eigenvalue parameters elasticity values is done using a function called cmpParamElastVec.

Identifying all Loops

This section and all the following sections come after the end of TIME loop.
The loops in the model are identified using a function called indCycles.

```
[ allCyclesVerticesMatrix , independentCyclesVerticesMatrix ,
independentCyclesEdgesMatrix ] = indCycles( modelAdjacencyMatrix ,
modelAdjacencyMatrix2Edges , modelObjectsNamesVector );
```

The indCycles function uses an exhaustive search algorithm to find the paths that starts at some vertex in a digraph and ends at the same vertex; this algorithm is utilized by using a function called allcycsn (please, refer to the Mathematical Background Chapter and Appendices).

Selecting a Set of Independent Loops

The independent loop set is selected through the running of indCycles function.
The first task done in this context is to find the Binary expression of the loops matrix.

The independent loop set is not unique (refer to the Mathematical Background), but the function tries to select the most suitable set, by asking the user to select the most important loops –from his point of view–, out of the all loops set.

Computing the Dominant Eigenvalue Independent Loops' Elasticity Values

The dominant Eigenvalue independent loops' elasticity values are computed by calling cmpIndCycElast function.

```
independentCyclesElasticityMatrix = cmpIndCycElast(
independentCyclesEdgesMatrix , numericLinkElasticityMatrix );
```

The cmpIndCycElast function, finds a least squares solution for the equation relates the independent loop set dominant Eigenvalue elasticity values vector and the links' dominant Eigenvalue elasticity values (please, refer to the Mathematical Background Chapter).

```
Cr = independentCyclesEdgesMatrix.';
independentCyclesElasticityMatrix = Cr \ numericLinkElasticityMatrix;
```

Printing the Results

The last step in the implementation of the Eigenvalue analysis function is to print the results to a file; to give the user the chance to make further analysis on those results. The printing task is done using a function called printAll.

```
printAll( level2study , modelAdjacencyMatrix , internalStep ,  timeSteps ,
dominantEigenVector , dominancePercentageVector , numericLinkGainMatrix ,
numericLinkElasticityMatrix , numericParameterElasticityMatrix ,
independentCyclesElasticityMatrix , allCyclesVerticesMatrix ,
independentCyclesVerticesMatrix , modelObjectsNamesVector , constantsVector
, outFileName );
```

This function nearly takes all the needed outputs to print out, and it returns nothing, just it gives out a file that has the name stored in outFileName.
It prints the following:
All Eigenvalues and Their Dominance Percentage
Links' Gains
Links' Dominant Eigenvalue Elasticity Values
Links' Dominant Eigenvalue Elasticity Values (Sorted)
Inputs' Dominant Eigenvalue Elasticity Values
Inputs' Dominant Eigenvalue Elasticity Values (Sorted)
All Loops
User-selected Linearly Independent Loops
User-Selected Linearly Independent Loops' Dominant Eigenvalue Elasticity Values
User-Selected Linearly Independent Loops' Dominant Eigenvalue Elasticity Values (Sorted)

References

Abdel Aleem, Bahaa E. Aly. 2004. An Automated System to Analyze System Dynamics Models - Case Study: Commodity Production Cycles, Bergen: University of Bergen.
AbdelGawad, Ahmed A. 2004. An Automated System to Analyze System Dynamics *Models*, Bergen: University of Bergen.
Bahar, M. and J. Jantzen 1995). *Digraph toolbox*.
Belikov, B. S. 1986. General methods for solving physics problems. Moscow, Mir publishers.
Beltrami, E. 1998. *Mathematics for Dynamic Modeling*, Academic Press.
Bowman, C. F. 1994. Algorithms and Data Structures an Approach in C. New York, Oxford University Press Inc.
Deif, A. S. 1982. Advanced Matrix Theory for Scientists and Engineers, Abacus Press.
Diestel, R. 2000. *Graph Theory*. New York, Springer-Verlag.
Dorf, R. C. and R. H. Bishop 2000. *Modern Control Systems*, Prentice Hall.
Edminister, J. 1972. Schaum's Outline of Electric Circuits, McGraw-Hill.
Forrester, J. W. 1968. Market Growth as Influenced by Capital Investment. *Industrial* Management Rev. MIT. **9:** 83-105.
Forrester, J. W. 1975. Market Growth as Influenced by Capital Investment. *Collected Papers of Jay W. Forrester.* Cambridge MA, Productivity Press.
Forrester, J. W. 1978. Market Growth as Influenced by Capital Investment. *Managerial Applications of System Dynamics.* E. B. Roberts. Cambridge MA, Productivity Press.
Forrester, N. B. 1982. A Dynamic Synthesis of Basic Macroeconomic Theory: Implications for Stabilization Policy Analysis, M. I. T.
Forrester, N. B. 1983. Eigenvalue Analysis of Dominant Feedback Loops. Intl. System Dynamics Conf., Chestnut Hill, MA.
Friedland, B. 1987. *Control System Design*, McGraw-Hill Book Company.
Gopal, M. 1993. *Modern Control Theory*, Wiley Eastern Limited.
Gordon, G. 1989. *System Simulation*. New Delhi, Prentice-Hall of India.
Grantham, W. J. and T. L. Vincent 1993. Modern Control Systems Analysis and Design, John Wiley & Sons, Inc.
Hanselman, D. C. and B. R. Littlefield 1997. *Mastering MATLAB 5: A Comprehensive* Tutorial and Reference, Prentice Hall.
Kampmann, C. E. 1996. *Feedback Loop Gains and System Behavior*. 1996 International System Dynamics Conference, Cambridge, Massachusetts, System Dynamics Society.
Kendall, K. E. and Kendall, J. E. 2002. *Systems Analysis and Design*. Upper Saddle River, NJ, Prentice-Hall Inc.
Kheir, N. A. 1996. Systems Modeling and Computer Simulation. New York, Marcel Dekker Inc.
Kreyszig, E. 1993. *Advanced Engineering Mathematics*. New York, John Wiley & Sons, Inc.
Kuo, B. C. 1995. *Automatic Control Systems*. Englewood Cliffs, NJ, Prentice-Hall, Inc.
Nise, N. S. 1994. *Control Systems Engineering*, John Wiley & Sons, Inc.
Ogata, K. 1997. *Modern Control Engineering*. Upper Saddle River, NJ, Prentice-Hall Inc.
Palm, W. J. 2000. Introduction to Matlab 6 for Engineers, McGraw-Hill Science/Engineering/Math.
Richardson, G. P. and A. L. Pugh, III 1981. *Introduction to System Dynamics Modeling with DYNAMO*. Cambridge MA, Productivity Press.

Saleh, M. 1998. An Object Oriented Approach to Automate System Dynamics Model *Optimization*, Department of Information Science, University of Bergen, Norway.
Saleh, M. 2002. The Characterization of Model Behavior and its Casual Foundation, Department of Information Science, University of Bergen, Norway.
Saleh, M. and P. I. Davidsen 2000. An Eigenvalue Approach To Feedback Loop *Dominance Analysis In Non-Linear Dynamic Models*. 18th International Conference of the System Dynamics Society, Bergen, Norway, System Dynamics Society.
Saleh, M. and P. I. Davidsen 2001. *The Origins of Business Cycles*. The 19th International Conference of the System Dynamics Society, Atlanta, Georgia, System Dynamics Society.
Sterman, J. D. 2000. Business Dynamics : Systems Thinking and Modeling for a Complex
World. Boston, Irwin/McGraw-Hill.
The Mathworks 2002. *Using Matlab Version 6*, The Mathworks, Inc.
The Mathworks 2002. *Using Simulink Version 5*, The Mathworks, Inc.
Weisstein, E. W. 1999. Concise Encyclopedia of Mathematics CD-ROM.
White, J. 2004. *System Dynamics* (22.554 and 24.509).