# BEHAVIOR ANALYSIS SOFTWARE FOR LARGE DYNAMO MODELS

Alan K. Graham
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, Massachusetts

Alexander L. Pugh III
Pugh-Roberts Associates
Cambridge, Massachusetts

## ABSTRACT

An experimental software package is being used as an extension to the DYNAMO IV compiler to linearize the model at any point during a simulation, compute the eigenvalues and eigenvectors of the linearized system, identify the levels important in producing each behavior mode, and compute the elasticity of a given eigenvalue (corresponding to elasticity of period and damping) with respect to all model parameters. The package is intended to help modelers understand the causes of behavior in very complex models, both for debugging implausible behavior, and for presenting the causes of plausible behavior more convincingly. The package is able to work for the System Dynamics National Model, a model of around 300 levels. Practical experience has uncovered some difficulties in making the analysis useful, but these are being surmounted. The experience suggests that mathematical methods should be used extensively "in the field" before being offered as candidates for expanding the paradigm of System Dynamics modeling.

## INTRODUCTION

Understanding why a model behaves as it does is an integral part of System Dynamics. If the behavior is unrealistic, the causes need to be isolated so the model structure can be revised. If the behavior is realistic and being explained to clients, or if alternative policies are being designed, the dominant mechanisms need to be isolated and understood. The customary method of



understanding model behavior has been to examine simulations, formulate hypotheses about what is causing the behavior, and then make further simulations to test and refine the hypothesis. While satisfyingly simple for analyzing behavior of small models, such hypothesis testing becomes very laborious for large models. For the System Dynamics National Model, one author (AKG) has performed over 3,000 simulations on various versions of the model, which is not unusual for such large models.

Given the contributions that mathematical analysis has made to virtually every branch of science over the centuries, it is not unreasonable to wonder why mathematical methods have not been a standard part of System Dynamics practice. If one takes a close look at the control theory literature, one discovers that the majority of the methods are aimed at problems others than analyzing the important causes of system behavior; topics include solving for system behavior, optimizing system behavior, and optimally estimating either parameter or level values. There are also a variety of sensitivity measures that are functions of time (change in behavior as a function of parameter change), or sensitivity of the cost function.

The result closest to being useful for understanding model behavior is the sensitivity of the eigenvalues (which characterize the possible modes of model behavior) or the eigenvectors (characterizing the movement of the levels in each behavior mode), to parameter variation. But as a means of practical model analysis, these do not provide very much useful information, because the quantities are all dimensioned differently. Looking at eigenvalue sensitivities to parameters to determine structural importance is like comparing apples per orange to grapes per banana.

Recently, two Ph.D. theses have layed the groundwork for more useful mathematical tools. The first, by Jose I. Perez, derives a formula for a nondimensional measure of the importance of a given

level variable in creating a given behavior mode (i.e. for a given eigenvalue), called participations [1]. He also created an algorithm for model simplication, that in effect changes the level variables not important to a given behavior mode into constants or auxilliaries. The algorithm preserves the causal structure of the model, and makes minimal changes to the parameter values to produce exactly the same eigenvalue in the simplified model as within the full system. However, as we rarely if ever focus on a few modes to the total exclusion of others, we have not attempted to implement this algorithm.

The second thesis, by Nathan B. Forrester, addressed the analysis of the causes of business cycle oscillations in a small model [3]. Forrester introduced the idea of eigenvalue elasticities with respect to parameters--nondimensional quantities that could be compared with one another to assess the importance of each parameter, and the causal link in the model that it characterized.

These two recent theses show that there are useful mathematical analyses of large systems to be done. The System Dynamics National Model Project is a situation where there is considerable application for a tool that could speed up behavior analysis [4]. The model is intended to show at least three basic behavior modes (50-year long wave, 3-7 year business cycle, and inflation from expansion of money supply) in a framework rich enough for a variety of policy evaluations. The current model has about 300 levels, and it is difficult and sometimes extremely time-consuming to understand why the model behaves as it does. Given the massive need for model analysis and the "pilot study" by Nathan Forrester, it seemed worth making the investment in software to produce a behavior analysis package coupled with the DYNAMO compiler.

The software now being developed performs its calculations in a particular order, because many computations use the results of



earlier computations. This paper will follow the same order of events, beginning with the automatic adaptation of a standard DYNAMO model to the needs of linear mathematical analysis.

## LINEARIZATION

The behavior analysis software is an enhancement to the DYNAMO IV compiler, both written by Alexander Pugh [5]. (At present, these are only available for IBM 370 computers.) As a result of integration with the compiler, the software deals with DYNAMO models as they are normally used--no special conversions are necessary. The first phase of the behavior analysis computations is linearization; subsequent phases use linear analysis techniques (some well-known, and some very new).

The user can run a simulation to any point in time, and linearize the model around the level values that exist at that time. More precisely, if the original nonlinear system is represented as

$$\frac{d\underline{x}}{dt} = \underline{f}(\underline{x}) \tag{1}$$

where

$\underline{x}$ - vector of level variables (n x 1)

$\underline{f}$ - rate of change function (n x 1)

then the linearized version is

(2) $$\frac{d\underline{x}}{dt} = \underline{A}\underline{x}$$

(3) $$a_{ij} = \frac{\partial f_i}{\partial x_j}$$

$\underline{x}$ - vector of level variables (n x 1)

$\underline{A}$ - linearized system matrix (n x n)

$a_{ij}$ - i,j$^{th}$ element of $\underline{A}$ (scalar)

The partial derivatives are computed by increasing each level in the system by 1 percent, and computing the fractional change in the rates of flow.

**EIGENVALUES**

Critical to all later developments is the computation of the eigenvalues, which are also known as roots of the characteristic equation, or just characteristic values. And the eigenvalues do indeed characterize--in a very concise manner--the behavior of the system: the behavior of the (linearized) system can be expressed as a weighted sum of eigenvectors (discussed below) multiplied by an exponential of each of the eigenvalues:

(4) $$\underline{x}(t) = \sum_m w_m \underline{r}_m \exp(s_m)$$

$\underline{x}$ - vector of level variables (n x 1)

$\underline{r}_m$ - right eigenvector for m$^{th}$ eigenvalue (n x 1)

$s_m$ - m$^{th}$ eigenvalue (complex scalar)

where each pair of eigenvectors and eigenvalues satisfies



(5) $$\underline{A}\underline{r}_m = \underline{r}_m s_m$$

$\underline{A}$ - linearized system matrix (n x n)

$\underline{r}_m$ - right eigenvector for m$^{th}$ eigenvalue (n x 1)

$s_m$ - m$^{th}$ eigenvalue (complex scalar)

(The term "right eigenvalue correctly implies that there is also a left eigenvalue, which will be discussed below.)

For clarity, the software does not print out the values of the eigenvalues directly, but rather the corresponding period and time constant for oscillatory modes, or the time constant for simple exponential modes.

The eigenvalue computation is performed with the QR algorithm, using double precision arithmetic [6]. Fortunately, the QR method is reliable and sufficiently fast to compute all the eigenvalues. The cost of computing the eigenvalues is the major computational expense. The cost goes up approximately as the cube of the number of levels; for the System Dynamics National Model, taking all 300 eigenvalues at evening rates costs about $40.

There is some difficulty at present with the core memory required for the eigenvalue computation. The memory holds three nxn double-precision matrices and one nxn single-precision matrix. For the System Dynamics National Model, the former each contain 300x300x4x2=720,000 bytes, which is a large enough requirement on the computer that the eigenvalue computation can normally be done only during the evening when the computer is lightly-utilized. However, the A matrix is mostly zeros (each level effects only some small number of other levels, nowhere close to the full 300 levels). This means that by using somewhat more complicated storage and computing algorithms (the so-called sparse matrix techniques), the memory requirements possibly could be cut

considerably, by storing and computing with just the nonzero elements.

By themselves the the eigenvalues are of relatively little value; they mostly confirm what is already known from simulating the model--the damping, period and time constants of the dominant behavior modes. More importantly, the eigenvalues are the foundation for succeeding computations. But one interesting finding did come from the eigenvalue computation itself.

Given the fact that most System Dynamics models are fairly smoothly nonlinear, our initial expectation was that the choice of what conditions at which to linearize the system would make little difference. However, System Dynamics models are also pervasively nonlinear, so many coefficients of the A matrix will change, depending on where the model is linearized. A distinctly different set of eigenvalues can result from changing the level values around which the model is linearized. (It may be that although eigenvalues differ, the important mechanisms and levels stay the same; we do not have enough experience with the package to know yet.)

Printing out 300 eigenvalues for a large system like the SDNM may seem like an excess of information, especially if one thinks about parameter elasticities for each. Our software prints all the eigenvalues in the order of the smallest to the largest real part. As this led to about 300 real and complex numbers, we will truncate this list in the future. (Controling the amount of output has been an issue in each step of our development. This is the only place where we did not limit output, but just sorted it.)

Fortunately, the vast majority of eigenvalues are simply exponential convergences back to steady-state, or very highly-damped oscillations. For the SDNM, there are usually no more than five interesting modes--expanding modes (both smooth and oscillatory), damped oscillations at two or three periods, and the



longest and shortest exponential approach just about fill the list. So it should be straightforward to pare down the list of eigenvalues of interest to the modeler.

## EIGENVECTORS

Like the eigenvalues, the usefulness of eigenvectors lies mostly in later computations. Equation (5) above gives the definition of the right eigenvlue. By symmetry, the left eigenvalue is the same equation, but with the eigenvector to the left (instead of right) of the system matrix A:

(6) $$\underline{l}_m \underline{A} = \underline{l}_m s_m$$

$\underline{l}_m$ - left eigenvector for $m^{th}$ eigenvalue (n x 1)

$\underline{A}$ - linearized system matrix (n x n)

$s_m$ - $m^{th}$ eigenvalue (complex scalar)

The computations described below require both the left and right eigenvectors. Most discussions of eigenvectors use equations that yield all n right eigenvectors, and then compute from those all n left eigenvectors. But for large models, the cost of computing all eigenvectors is about three times as large as computing the eigenvalues themselves, and the algorithms are very vulnerable to singularity problems. After some analysis and search, we arrived at an algorithm that computes just one eigenvector at a time, so that only two passes are needed to compute both the left and right eigenvectors [7]. Given that only a very small number of eigenvalue/eigenvector pairs represent interesting dynamics, this algorithm saves computation time, in addition to being robust.

Although the meaning of the left eigenvector is rather abstruse [8], the right eigenvector has a pleasingly simple interpretation. If the equilibrium values of the levels are defined as a vector of zeroes, then consider what happens if the levels are initialized in proportion to a particular right eigenvector:

(7) $\underline{x}(0) = k\underline{r}_m$

$\underline{x}$ - vector of level variables (n x 1)

$\underline{r}_m$ - right eigenvector for $m^{th}$ eigenvalue (n x 1)

Using the rate of change from equation (2) and the equivalence in equation (5),

(8) $\left.\frac{d\underline{x}}{dt}\right|_{t=0} = \underline{A}k\underline{r}_m = k\underline{r}_m s_m$

$\underline{x}$ - vector of level variables (n x 1)

$\underline{A}$ - linearized system matrix (n x n)

$\underline{r}_m$ - right eigenvector for $m^{th}$ eigenvalue (n x 1)

$s_m$ - $m^{th}$ eigenvalue (complex scalar)

So the rate of change vector is exactly parallel to the initial condition. In fact, this will always be true, and



(9) $\underline{x}(t) = k\underline{r}_m \exp(s_m t)$

$\underline{x}$ - vector of level variables (n x 1)

$\underline{r}_m$ - right eigenvector for $m^{th}$ eigenvalue (n x 1)

$s_m$ - $m^{th}$ eigenvalue (complex scalar)

In other words, the right eigenvector specifies a set of initial disturbances from equilibrium such that the levels continue to show behavior proportional only to that disturbance, and the variation over time is completely characterized by the eigenvalue. In short, the right eigenvector specifies how levels move within each particular behavior mode of the linearized system.

For shedding light on what causes a given mode of behavior, the right eigenvector is fairly opaque, for it tells about the movements of the levels, each measured in its own units of measure. Moreover, there is no distinction within the right eigenvector between levels that are essential to creating a behavior mode, and levels that are merely driven by the behavior and are not essential--the eigenvector indiscrimminately mixes the tails with the dogs that wag them. But it is possible to circumvent these difficulties by combining both the left and the right eigenvectors, to compute participations.

**PARTICIPATIONS**

The participation of the $l^{th}$ level in the $m^{th}$ mode is defined as

(10) $$p_{ml} = l_{ml} r_{ml}$$

$p_{ml}$ - participation of $l^{th}$ level in $m^{th}$ mode (complex scalar)
$l_{ml}$ - $l^{th}$ element of $\underline{l}_m$ (complex scalar)
$r_{ml}$ - $l^{th}$ element of $\underline{r}_m$ (complex scalar)

provided that $\underline{l}_m$ and $\underline{r}_m$ have been normalized so that

(11) $$\underline{l}_m' \underline{r}_m = 1$$

$\underline{l}_m$ - left eigenvector for $m^{th}$ eigenvalue (n x 1)
$\underline{r}_m$ - right eigenvector for $m^{th}$ eigenvalue (n x 1)

For an oscillatory behavior mode, then, the participation factor will be complex. It is provable that the sum of all the participations for a particular eigenvalue is 1+0j (j=sqrt(-1)) [9]. Moreover, the real part of the participations is nonnegative, so they are well-behaved as indices of importance; the participations of the key levels are typically greater than .1 in magnitude.

There are at least two interpretations of the participation factors. One interpretation revolves around the response of the system to an initial disturbance in one level, say $x_1(0)=1$ [10]. The response will be a weighted sum of exponentials representing each of the eigenvalues (and thus behavior modes) of the system.



Those weights are the participation factors, so that the participaiton factors for that level give the extent to which a disturbance in that level is able to excite the behavior mode. As argued heuristically by Graham [11], the cause of oscillatory behavior is the ability for a disturbance in one level to propagate itself around a loop. So the participation factors would seem to identify the levels that are most instrumental in causing a given behavior mode to persist.

The other interpretation of the participation factors revolves around commonality with the equations for sensitivity (see below); mathematically, the participation factor is equal to the sensitivity of the eigenvalue with respect to the gain from the given level to itself--the sensitivity of a minor loop around the level. So levels with high participation factors can also be thought of as the levels which, when they are constrained by increasing the strength of a minor negative loop around them, have the most effect on the behavior. Again, this interpretation relates back to the identification of levels whose freedom of motion is important to creating the behavior mode.

In earlier versions of the software, we have experimented with heuristics using the participation factors as the basis for selecting which data to compute and display. That isn't done now, so there are only two uses left for the participation factors. The first use, of course, is to refine one's ideas of what causes the behavior mode. Knowing the important levels is not the same as knowing the important connections between the levels, but the levels by themselves are sometimes useful, especially when a level is unexpectedly important, for that points up part of the model that is operating differently than expected.

The second use of the participation factors is as a (relatively) unique identifier for a behavior mode. The eigenvalue software hasn't enough information to print out a behavior mode under the label "business cycle." It comes out as

mode number 85, or such. Only the period and damping from the eigenvalue, and knowing the important levels from the participations lets one identify a given mode as "the business cycle." The damping and period of the eigenvalue plus the important participationg levels provide a unique idnetification for a given behavior mode.

Participations identify the key levels involved in a particular mode but do not explicitly identify loops. To obtain more explicit information about which loops are important to the model behavior, it is natural to turn to modifications of traditional linear analysis--elasticities.

## GAIN ELASTICITIES

Given that the endogenous dynamics of the linearized system are fully determined by the gain coefficients within the A matrix, it is both logical and mathematically simple to address the question of where the important gain coefficients (sometimes called coupling time constants) are, within the A matrix. One could look at the ability of changes in each of the gain elements of the A matrix to alter the eigenvalues; this should tell us where the important connections between levels lie.

We can define a nondimensional quantity, the elasticity of the $m^{th}$ eigenvalue with respect to some gain parameter $a_{ij}$, in the standard form for elasticities used in economics. For brevity, these will be referred to as gain elasticities.



$$(12) \qquad e_{mij} = \frac{\partial s_m}{\partial a_{ij}} \times \frac{a_{ij}}{s_m}$$

$e_{mij}$ - elasticity of $m^{th}$ eigenvalue with respect to $a_{ij}$ (complex scalar)
$s_m$ - $m^{th}$ eigenvalue (complex scalar)
$a_{ij}$ - i,$j^{th}$ element of $\underline{A}$ (scalar)

This gain elasticity can be computed using a well-known formula for response to any system parameter

$$(13) \qquad \frac{s_m}{p_g} = \underline{l}_m \cdot \frac{\partial \underline{A}}{\partial p_g} \underline{r}_m$$

$s_m$ - $m^{th}$ eigenvalue (complex scalar)
$p_g$ - $g^{th}$ parameter (scalar)
$\underline{l}_m$ - left eigenvector for $m^{th}$ eigenvalue (n x 1)
$\underline{A}$ - linearized system matrix (n x n)
$\underline{r}_m$ - right eigenvector for $m^{th}$ eigenvalue (n x 1)

So

$$(14) \qquad e_{mij} = \underline{l}_m \cdot \frac{\partial \underline{A}}{\partial a_{ij}} \underline{r}_m = l_{mi} r_{mj} a_{ij} / s_m$$

$e_{mij}$ - elasticity of $m^{th}$ eigenvalue with respect to $a_{ij}$ (complex scalar)
$\underline{l}_m$ - left eigenvector for $m^{th}$ eigenvalue (n x 1)
$\underline{A}$ - linearized system matrix (n x n)
$\underline{r}_m$ - right eigenvector for $m^{th}$ eigenvalue (n x 1)
$a_{ij}$ - i,$j^{th}$ element of $\underline{A}$ (scalar)

Once the eigenvectors are known, then, the computation of the gain elasticities is very simple.

So the elasticity of an eigenvalue with respect to a parameter is defined as the relative change in the eigenvalue devided by the relative change in the parameter. Thus an elasticity of .5 implies that a 1% increase in the parameter will result in a .5% increase in the eigenvalue. If the eigenvalue is complex, both the real and imaginary parts will increase .5%. By considering the elasticities of the eigenvalue with respect to the links between the levels (the A matrix) one should be able to identify the key feedback loops that determine a mode.

The elasticities as defined above have several interesting properties. They are nondimensional, and can thus be compared with one another, even for parts of the system with vastly differing units of measure. Moreover, they are complex numbers, and the real and imaginary parts contain information about the effect of the parameter on period and damping. If the real part is positive, an increase in the parameter will decrease the undamped natural period $P_u$, which is defined as

(15) $$P_u = 2pi/|s_m|$$

$P_u$ - undamped natural period (time units)

pi - 3.141592...

$s_m$ - $m^{th}$ eigenvalue (complex scalar)

$P_u$ will usually be shorter than the actual period; the name for this quantity comes from the formula for the behavior of a second-order negative feedback loop with minor loops (either positive or negative)--$P_u$ is what the period would be if the minor



loops are removed. Generally, one can expect the direction of change of $P_u$ to match that of the actual period, except in cases where there are large changes in the damping.

The imaginary part of the gain elasticity gives the effect on the damping ratio, which is defined as

(16) $$d = -\cos(\angle s_m) = -\mathrm{Re}(s)/|s_m|$$

d - damping ratio (dimensionless)

cos - cosine function

$\angle$ - function taking angle of a complex number

$s_m$ - $m^{th}$ eigenvalue (complex scalar)

Re - function taking real part of a complex number

|| - function taking magnitude of a complex number

A damping ratio of 1.0 denotes a simple exponential convergence, and -1.0 a simple exponential growth. A damping index between 1.0 and 0 denotes convergent oscillation, 0 sustained oscillations, and between 0 and -1.0 divergent oscillations. Now if the complex part of the elasticity is positive, an increase in the parameter will decrease the damping ratio, i.e. damping will decrease [13]. The preceding rules are not easy to remember; we anticipate that in the future, the software will calculate elasticity of period and damping directly.

A number of properties of elasticities with respect to elements of the A matrix have yet to be fully exploited. It is possible to show, using implicit differentiation and invariance with respect to choice of time units and units of measurement of the levels, that the elasticities for all elements sum to 1, and the sum of elasticities from all levels to a given level equals

the sum of all elasticities from that level--a kind of Kirchoff's law for all linear systems. Nathan Forrester has extended this particular property of gain elasticities to be able to place a value on the importance of every loop of the model, for a given mode of behavior [14]. This is an exciting result, but it appears that it will require considerable computational effort--apparently, models with more that 25 or 50 levels will require unresonable computer resources.

When the software was first being implemented, it was fully expected that the elasticities with respect to elements of the A matrix would give good indication of the important structural connections for a given behavior mode. But such was not the case. The failure arose because System Dynamics models frequently represent conserved flows--of orders, of goods, of people, and of money. When such relationships are expressed in matrix form, the model parameters controlling the conserved flow in effect influence the matrix twice--once for the outflow from one level, and once for the inflow to the other level. But the formula for elasticity with respect to a single element of the A matrix represents the results of changing only one of the pairs of such parameters at a time. In effect, elasticity with respect to such an element of the A matrix represents the sensitivity of the behavior to allowing more to flow into one level than comes out the other in a situation where the modeler intended conserved flows. Such elasticities might be called elasticities of violating conservation laws.

The System Dynamics National Model has numerous conserved flows of money. And, as one might imagine, several of the most interesting behavior modes are sensitive to money creation. So the list of the most important elasticities was filled with, for example, the elasticity of the parameters governing accounts payable. There were so many of these "elasticities of violation



of conservation" that the causal structure determining the behavior was not discernable from the elasticities with respect to the A matrix.

What seems necessary to determine the significant structural connections, then, is the assessment of those connections as they appear in the DYNAMO model, in which parameters are expressed in physically meaningful terms--in terms of conserved flows and model variables.

## PARAMETER ELASTICITIES

Elasticities of the eigenvalues with respect to the parameters of the model can explicitly identify the "handles" of the model; if we wish to change a mode we must change one (or more) of the parameters that affects it.

An elasticity of eigenvalues with respect to model parameters can be defined analogously to gain elasticities:

$$(17) \qquad E_{mg} = \frac{\partial s_m}{\partial p_g} \times \frac{p_g}{s_m} = \underline{l}_m' \frac{\partial \underline{A}}{\partial p_g} \underline{r}_m \times \frac{p_g}{s_m}$$

$E_{mg}$ - elasticity of $m^{th}$ eigenvalue with respect to $g^{th}$ parameter (complex scalar)

$s_m$ - $m^{th}$ eigenvalue (complex scalar)

$p_g$ - $g^{th}$ parameter (scalar)

$\underline{l}_m$ - left eigenvector for $m^{th}$ eigenvalue (n x 1)

$\underline{A}$ - linearized system matrix (n x n)

$\underline{r}_m$ - right eigenvector for $m^{th}$ eigenvalue (n x 1)

The partial derivative of the A matrix with respect to the parameter poses a problem. To compute this in brute-force

fashion, one varies the parameter, recomputes the A matrix, and compares it with the original A matrix to determine the matrix of partial derivatives. Computing the A matrix requires running the model one time interval n times, one for each level. To perform an exhaustive parameter analysis on the System Dynamics National Model with perhaps 300 parameters and 300 levels calls for 90,000 model evaluations--too many to be practical.

As an interim measure, the parameter elasticity software asks for a list of parameters for which elasticities are desired. While this is very useful, one will never discover an unexpectedly sensitive parameter, if every parameter elasticity that is to be computed must be asked for.

There is a way around this "curse of dimensionality." Equation (15) can be expressed as

$$E_{mg} = \sum_{ij} \frac{\partial a_{ij}}{\partial p_g} e_{mij} \qquad (18)$$

$E_{mg}$ - elasticity of $m^{th}$ eigenvalue with respect to $g^{th}$ parameter (complex scalar)

$a_{ij}$ - $i,j^{th}$ element of $\underline{A}$ (scalar)

$p_g$ - $g^{th}$ parameter (scalar)

$e_{mij}$ - elasticity of $m^{th}$ eigenvalue with respect to $a_{ij}$ (complex scalar)

The vast majority of the partial derivatives will be zero; only those links from level to level that the given parameter effects will have nonzero partial derivatives. So for the SDNM, of the 300x300=90,000 partial derivatives, perhaps an average of between 2 and 20 are nonzero. So if it is known which level-to-level links a parameter influences, then only 2 to 20 model evaluations would yield the parameter elasticity, instead of the 300. With



such a reduction in computation, exhaustive parameter search becomes feasible even for large models.

Parameter elasticities are another answer to the question "what's causing this behavior?" Parameters with large elasticities should define the important channels of influence in the model. Moreover, the parameter elasticities have the same interpretations as the gain elasticities discussed above for the effect of the given parameter change on both period and damping. (Technically, the analysis is valid only for small parameter changes, but usually within the range of plausible values, both large and small parameter changes have the same qualitative effect). So in addition to identifying important structure, the parameter elasticities should also be very useful in adjusting the behavior to real data.

## CONCLUDING REMARKS

There are a number of avenues for extending the work now in progress, ranging from the mechanical to the conceptual. Mechanically, the software will become steadily more user-friendly and efficient of computer space and time. As mentioned above, the eigenvalue algorithms compute with an a matrix of the full nxn size, which for the System Dynamics National Model presses against the limits of MIT's IBM 370. It will be desirable at some point to adapt the algorithms to work with information expressed in a more compact format, so that they can be run taking less space (of course, smaller models don't take as much space).

At the conceptual end of the spectrum lie two ideas mentioned previously: Forrester has outlined an algorithm to identify dominant loops, and Perez has specified in detail an algorithm for producing a simple model with the same principle behavior mode and the same causal structure as a more complex model. These would both be interesting to look into.

Finally, we need to continue to gather operating experience in the uses and pitfalls of linearized analysis. Are there more phenomena like "elasticity of violating conservation" that undermine the usefulness of the analysis? Can linearization away from equilibrium or even at extreme conditions tell us anything about the dominant structure? Are there systems where the eigenvalue analysis is misleading for the full nonlinear system? We need more experience with actual application to be able to say when, how, and by whom this software package ought to be used.

Some observations on the System Dynamics paradigm are in order. "Classical" System Dynamics is a self-contained discipline that can be taught to a wide variety of people. It is closely oriented toward solving the real problems at hand. While mathematical tools increase the modeler's power, they also complicate the process of modeling. For which modelers and tasks is the complication worthwhile? This question vis a' vis eigenvalue analysis is similar to the larger question of how much control theory students should learn [15]. The consensus of the field to date seems to have been that very little formal mathematics should be taught to people who are only taking two or three courses in System Dynamics. Indeed, at MIT, it is only Ph.D. candidates who are required to take seperate control theory courses. Now that control theory concepts are becoming more embedded in the easy-to-use software described here, it may be that some mathematical content is justified in the third course or so in SD.

To the extent that a mathematical tool is not well-understood in practice, it should remain in the domain of the professional SD researcher, not students. One can imagine scenarios where students, encountering phenomena like "elasticities of violating conservation" are frustrated and disappointed not only with the tool, but by extension with the whole field of System Dynamics.



And even if a mathematical tool is taught to students, a number of undesirable tendencies may emerge unless care is taken. There is a tendency for the tools to select the problems tackled and the formulations used. For example, the field of operations research was originally research into operations, with results applied immediately in the front lines of World War II [16]. But over the years, the field has become academized, with elegant theorems about optimal queueing policies that may never be applied, and certainly don't address the today's major problems.

Mathematical tools such as eigenvalue analysis and parameter estimation pose a similar threat to System Dynamics:

> There is a temptation to exclude even realistic and necessary nonlinearities, or the use of important but nonmeasurable variables (goals, traditions, reputation, etc.), if they hinder the use of mathematical tools. (Find one microeconomics textbook that talks about goals, traditions or reputation!)
>
> There is a temptation to substitute computation for thinking; the phenomenon of doing many simulations instead of studying one insightfully is well enough known that it's come to be called "terminal hypnosis."
>
> Substituting computation for thinking also threatens to dramatically decrease the modeller's ability to speak insightfully about the problem with clients.
>
> The realistically robust models that typify many system Dynamics applications result from serious attention to the underlying structure of systems, rather than only its parameters. Even without mathematical methods that elevate the role of parameter values, there is a great temptation to manipulate model behavior with parameter changes, rather than seriously considering whether or not the structure is adequate.

In short, unless precautions are taken, mathematical tools are capable of undermining many of the unique advantages that System Dynamics now offers.

The cautions above notwithstanding, we believe that eigenvalue and elasticity analysis hold great promise for the field. We will continue research into its practical use.

NOTES

[1] Perez, Jose Ignacio Arriaga, Selective Modal Analysis with Applications to Electric Power Systems, unpublished Ph.D. dissertation, MIT School of Engineering, Cambridge, Massachusetts, June, 1981.

[2] Perez (note [1]), Chapters 4-5.

[3] Forrester, Nathan B., A Dynamic Synthesis of Basic Macroeconomic Theory: Implications for Stabilization Policy Analysis, unpublished Ph.D. dissertation, MIT Sloan School of Management, Cambridge, Massachusetts, August, 1982. Also available as System Dynamics Group Working Paper D-3384.

[4] Forrester, Jay W., Nathaniel J. Mass, and Charles K. Ryan, "The System Dynamics National Model: Understanding Socio-Economic Behavior and Policy Alternatives," Technological Forecasting and Social Change, vol. 9, pp. 51-68, 1976.

[5] DYNAMO IV is an extension of DYNAMO III embodying a more sophisticated integration algorithm. DYNAMO III is an extension to the "classic" DYNAMO, DYNAMO II, embodying matrices and vectors of variables, and the ability to preserve and resume at a simulated state of the system. For a description of all versions of DYNAMO, see Pugh, Alexander L. III, DYNAMO User's Manual, Sixth Edition, MIT Press, Cambridge, Massachusetts, 1983. For mathematical details of the DYNAMO IV integration algorithm, see Pugh, Alexander L. III, "Integration Method: Euler or Other for System Dynamics," TIMS Studies in the Management Sciences: System Dynamics, vol. 14, pp. 179-188, 1980.

[6] The algorithms for both the QR computation of eigenvalues and the single-mode computation of eigenvectors are described in Wilkinson, J. H. and C. Reinsch, Handbook for Automatic Computation, Volume II, Linear Algebra, Part 2, Springer Verlag, New York, 1971.

[7] See note [6]. The left eigenvector is computed as the right eigenvector of the A matrix transposed. Thus, the same algorithm computes both the left and right eigenvectors in seperate passes.

[8] Perez (note [1]), pp. 425-427.

[9] Perez (note [1]), pg. 91.

[10] Perez (note [1]), pg. 91.



[11] Graham, Alan K., Principles on the Relationship between Structure and Behavior of Dynamic Systems, unpublished Ph.D. dissertation, MIT School of Engineering, Cambridge, Massachusetts, August, 1977, Chapter 2.

[12] Porter, B. and R. Crossley, Modal Control Theory and Applications, Taylor and Francis, 1972.

[13] For an heuristic development of these rules, see Forrester (note [3]), pp. 58-59.

[14] Forrester (note [3]), pg. 225.

[15] Issues surrounding the use of mathematical analysis in System Dynamics are discussed in Graham (note [11]), pp. 309-313.

[16] For a poignant description of the early days of operations research, see Dyson, Freeman, Disturbing the Universe, Harper & Row, New York, 1979.