11

System Dynamics

Preview

In this chapter, a general strategy for inductive modeling will be introduced, a strategy that will allow us to model systems with totally unknown meta–laws. Of course, so constructed models will not offer the same degree of validity as deductively constructed physical models, and it will be important to discuss how the validity of these models can be assessed. While the methodology can be used to construct models in a completely inductive manner, it will allow us to incorporate in our model any physical insight that we may possess about the functioning of the process under investigation. The methodology has been coined *System Dynamics.* This is unfortunately somewhat of a misnomer. Didn't we discuss the "dynamics of systems" in this book all along? Didn't I reference in Chapter 4 a book entitled "System Dynamics" that talks about simple electrical and mechanical systems? In this new context, "System Dynamics" denotes a specific semi–physical and semi–inductive modeling methodology. To minimize the confusion, from now on, I shall always capitalize the term "System Dynamics" when I refer to this particular modeling methodology, and I shall not capitalize the term "system dynamics" when I refer to the dynamics of systems in general.

11.1 Introduction

If you were asked at this point to model the entire world to be able to speculate about the destiny of this planet of ours, you would probably be at a loss. You know how to model electrical circuits,

but the world? When you will have reached the end of this chapter, you will know how to "model the world".

The first exercise in *every* modeling endeavor is to choose the facets of the investigated system that we wish to capture in our model. Obviously, it would be very unrealistic to assume that we can capture each and every facet of the system in a model unless we decide to duplicate the system itself. We make this selection by identifying a set of variables which will be the key players in our model.

In physical modeling, we usually did not pay much attention to this step of the modeling cycle, since the choice of variables came about very naturally. For example, in an electrical circuit, we started by deciding whether we were to capture the electrical phenomena only, or whether we were to include thermic phenomena as well. Thereafter, we applied the well–established meta–knowledge of electrical circuitry to generate the model. Only in the end, we chose our "key players", which at that time we called our *state variables*, simply as the outputs of every single integrator in the model. This means that we created our model first (on the basis of available meta–knowledge), and that we chose our state variables only later.

How are we going to choose our state variables when we "model the world"? State variables capture significant quantities of a system, quantities which have the property that they *can accumulate* over time (which is, of course, just another way of saying that they are outputs of integrators). Typical candidates for state variables might be populations, money, frustration, love, tumor cells, inventory on stock, and knowledge. In System Dynamics, state variables are called either *levels* [11.6] or *stocks* [11.15], depending on which reference we use.

A first type of equation used in all System Dynamics models captures the fact that the change of each level (or stock) over time can be expressed as the difference between inflows and outflows of this level variable, for instance:

$$\dot{P} = BR - DR \tag{11.1}$$

The derivative of a population P with respect to time in a closed system can be expressed as the difference between the births per unit time (the so–called birth rate, BR) and the deaths per unit time (the so–called death rate, DR). Table 11.1 lists inflows and outflows for the levels that were proposed above.

Table 11.1 Inflows and outflows of typical level variables

level	inflow	outflow
population	birth rate	death rate
money	income	expenses
frustration	stress	affection
love	affection	frustration
tumor cells	infection	treatment
inventory on stock	shipments	sales
knowledge	learning	forgetting

In System Dynamics, the inflows and outflows are called either *rates* [11.7] or *flows* [11.15], depending on which reference we use.

From Table 11.1, we can learn a number of things:

(1) The concepts of *levels* and *rates* are extremely general. Applications can be found everywhere.

(2) Rate variables could be levels at the same time (frustration). We shall see how we deal with this problem.

(3) System Dynamics modelers are notoriously "sloppy" with their nomenclature. Of course, *treatment* is not the outflow of *tumor cells*, but treatment leads to the death of tumor cells, and if we say that one unit of treatment can be equated with one unit of killed tumor cells at all times, we may not need to distinguish between these two variables in the model.

After we defined all the rates, we need to come up with equations that relate the rates back to the levels. In a more familiar terminology, we need to come up with a set of *state equations*. These equations can possibly be simplified by introducing auxiliary variables. In System Dynamics, these auxiliary variables are sometimes called *converters* [11.15].

So far, nothing extraordinary has been detected about the System Dynamics methodology. What is particular about System Dynamics is the way in which the state equations are formulated. This will be the next point on the agenda.

11.2 The Laundry List

In a first attempt to derive a set of state equations, we could try to enumerate all the factors that have an influence on the rate variables, for example:

$$\left.\begin{array}{c} \textit{population} \\ \textit{material standard of living} \\ \textit{food ratio} \\ \textit{crowding ratio} \\ \textit{pollution} \end{array}\right\} \rightarrow \textit{birth rate} \qquad (11.2)$$

In our world model, the birth rate is influenced by the population, the material standard of living, the food ratio, the crowding ratio, and the pollution. Such an enumeration is called a *laundry list* [11.15]. The influencing factors may be levels, rates, or converters.

Of course, we must be careful to avoid such dubious relations as:

$$\textit{death rate} \rightarrow \textit{birth rate} \qquad (11.3a)$$

$$\textit{birth rate} \rightarrow \textit{death rate} \qquad (11.3b)$$

which is just another way of saying that we should avoid the creation of *algebraic loops* among the rate variables.

Laundry lists are the first step on the way to deriving state equations.

11.3 The Influence Diagram

Once we have designed laundry lists for all rate variables, we can connect all these laundry lists in one flow chart. Fig.11.1 shows the flow chart of Gilpin's model [11.8].

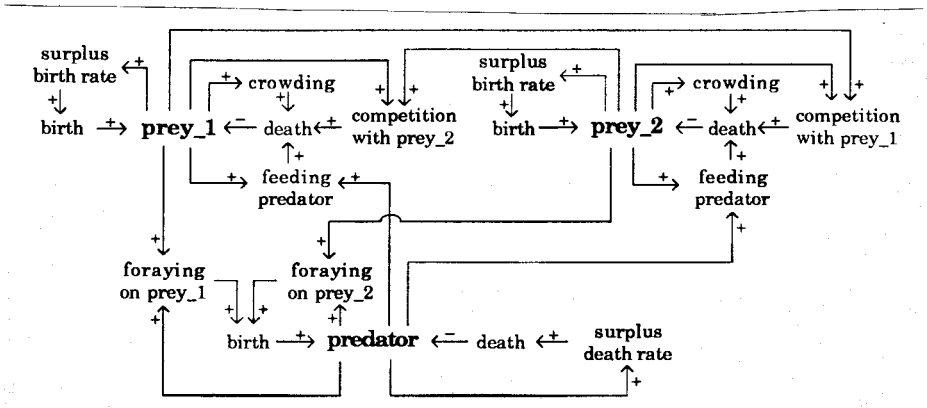

Figure 11.1. Influence diagram for Gilpin's model

This type of flow chart is referred to as either *influence diagram* [11.3] or *causal loop diagram* [11.15], depending on which reference we use.

Notice the sign which is attached to all paths in an influence diagram. These signs describe whether the influence is *positive* or *negative.* For example, the birth rate influences the population in a positive sense, whereas the death rate influences the population in a negative sense.

Influence diagrams are somewhat similar to *block diagrams* and to *signal flow graphs.* Yet, they are also quite different. We can draw an influence diagram at a much earlier instance of the modeling cycle than we can draw a block diagram or a signal flow graph. While the influence diagram tells us which variable depends on which other variables, it does not reveal the *nature* of this dependency. An influence diagram is therefore much less formal than either a block diagram or a signal flow graph. This is a major strength of influence diagrams, and at the same time, it is also a major weakness.

If we are told which state variables are to describe our "world model", we are ready to draw an influence diagram, but we would not know yet how to draw either a block diagram or a signal flow graph. This is clearly a strength. However, due to their informality, influence diagrams are somewhat unorganized. We tend to create too many dependencies — after all, ultimately, every event in this world is somehow related to every other event. The question is whether a proposed dependency contributes *significantly* to our modeling effort. A methodology which does not encourage us to distinguish between significant and insignificant relations, is certainly problematic. The lack of rigor is, therefore, a major weakness of influence diagrams, and it is, by the way, a major weakness of the System Dynamics methodology as a whole. Using System Dynamics, we are all too easily seduced into creating models that are much too complicated, far more complicated than the quantity and quality of the available data are able to support and to validate. We shall return to this point later in this text.

The signs that are attached to each influence path allow us to analyze the stability behavior of our model in qualitative terms. If we follow influence paths around a closed loop, we can count the negative signs that we meet along the way. If the total sum of negative signs is *even*, we have identified a *positive feedback loop.* Positive

feedback loops are always *unstable*, i.e., they are responsible for unbounded growth in a model. For example, if we extract the feedback loop:

$$Prey\_1 \rightarrow (+)surplus\ birth\ rate \rightarrow (+)birth \rightarrow (+)Prey\_1 \qquad (11.4)$$

from the rest of the model, we observe that the level $Prey\_1$ will grow exponentially beyond all bounds. If the total sum of negative signs is *odd*, we have identified a *negative feedback loop*. Negative feedback loops are more difficult to assess. If the number of level variables that we pass on our way around the loop is one or two, the negative feedback loop is certainly *stable*. For example, if we analyze the feedback loop:

$$Predator \rightarrow (+)surplus\ death\ rate \rightarrow (+)death \rightarrow (-)Predator \qquad (11.5)$$

we come across one negative sign, and therefore, this identifies a negative feedback loop. Along the way, we meet one level variable only (the predator), and thus the negative feedback loop is stable. If we extract this loop from the overall model, we observe that the predator population exhibits an exponential decay. Unfortunately, the influence diagram does not tell us which are level variables and which are not, which is another obvious shortcoming of influence diagrams.

If we meet more than two level variables along a negative feedback loop, we cannot tell whether the loop is stable. Stability will then depend on the total open–loop gain. This observation is related to the *Nyquist stability criterion* for feedback control systems. Most references on System Dynamics state incorrectly that "negative feedback loops are always stable". This is because the System Dynamics methodology is rarely applied to high order models with very complex feedback mechanisms, and if it is, the stability properties of these models will not be analyzed in qualitative terms. Moreover, most researchers of System Dynamics are not versed in control theory.

We notice further that Fig.11.1 is less concise and less easy to read than the explicit state–space model itself:

$$\dot{x}_1 = x_1 - 0.001\ x_1^2 - 0.001\ x_1 x_2 - 0.01\ x_1 x_3 \qquad (11.6a)$$

$$\dot{x}_2 = x_2 - 0.0015\ x_1 x_2 - 0.001\ x_2^2 - 0.001\ x_2 x_3 \qquad (11.6b)$$

$$\dot{x}_3 = -x_3 + 0.005\ x_1 x_3 + 0.0005\ x_2 x_3 \qquad (11.6c)$$

which contains even more information than the influence diagram since it explicitly states the nature of all dependencies. This is not a shortcoming of the influence diagram, but simply signifies that we have badly abused the System Dynamics methodology. System Dynamics is a modeling tool, not a model documentation tool. Once we have a working state–space model, it makes little sense to go back and construct a System Dynamics model after the fact. System Dynamics models are useful on the way to eventually producing state–space models for systems for which we lack applicable meta–knowledge, and not the other way around. We never create a System Dynamics model for an electrical circuit. This would be pure nonsense.

Furthermore, System Dynamics is a poor methodology to describe strongly intermeshed systems, i.e. systems in which every state variable is tightly coupled with every other state variable, as this is the case in Gilpin's model. The reason for this statement will become clear in due course.

11.4 The Structure Diagram

To avoid some of the problems associated with influence diagrams, Forrester [11.5] suggested another representation, the *structure diagram.* Structure diagrams are similar to influence diagrams, but they distinguish clearly between levels, rates, and converters. Fig.11.2 shows the structure diagram of Gilpin's model.

Level variables are represented by square boxes. Most levels are bracketed by two little clouds which represent the *sources* and *sinks* of the level. Sources provide an infinite supply of the material which is stocked up (accumulated) in the level variable. Sinks provide an unexhaustible dumping place for the same material. The double lines from the source cloud to the level and further to the sink cloud symbolize the *flow of material.* Single lines symbolize the *flow of information,* i.e., they denote control signals. This is similar to the power bonds *vs* the signal paths in the bond graph modeling approach. However, in bond graph modeling, we concentrated on the flow of power, i.e., the power bonds were the dominant elements in the model. Here, we concentrate more on the flow of information, and therefore, the single lines are dominant.

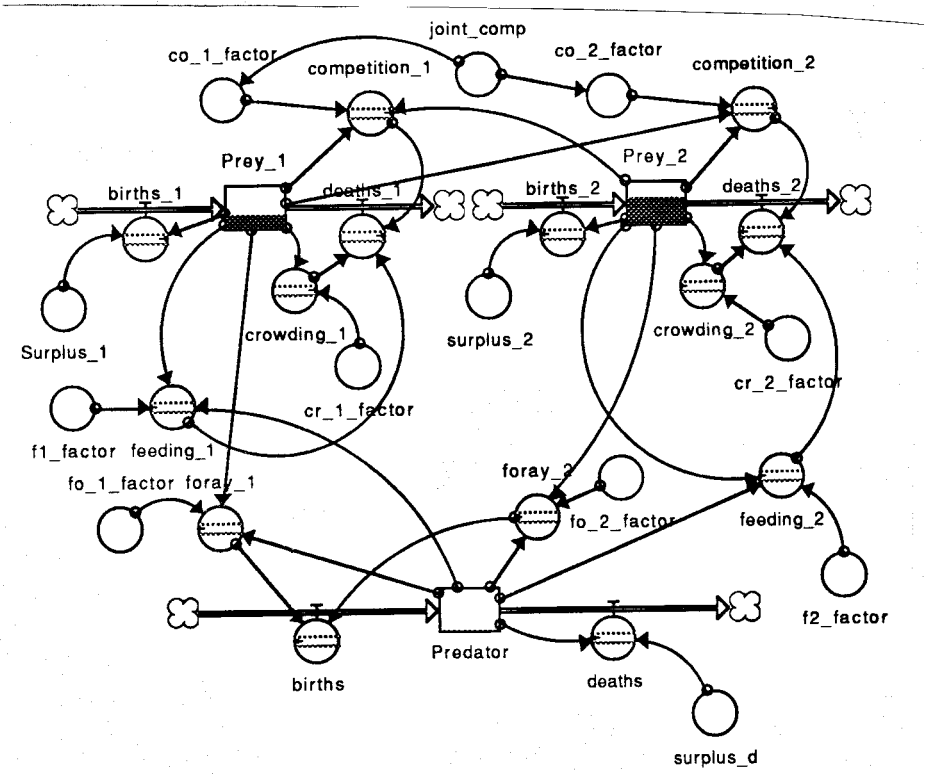

Figure 11.2. Structure diagram for Gilpin's model

Rate variables are denoted by circles with an attached valve. The rate variables control the flow into and out of the storages (levels, stocks) symbolically by opening/closing the valve that they are responsible for. However, the rate variables do not themselves decide upon the amount of opening/closing of the valves. They are only the guardians of the valves, i.e., they are the paid laborers who operate the valves in accordance with what they are told to do.

Each rate variable has one or several masters who tell it how far to open/close its valve. This is symbolized by one or more control signals (single lines) ending in the rate variable. They can emanate from anywhere in the model, even from another rate variable. However, if rate variables are themselves used as masters, we have to be extremely careful that we avoid creating algebraic loops.

Converter variables are denoted by circles without an attached valve. They are messengers. They collect command information from one or several sources, preprocess it, and deliver it to a rate variable or another messenger. They are introduced strictly for convenience.

Sometimes, it is useful to maintain several state variables in the model which are of the same type. For example, if we compartmentalize an ecosystem, populations of the same species appear in each of the compartment models. In this case, migration of an animal from one compartment to the next is symbolized by material flow that emanates from one level variable, and proceeds directly to the next level variable. A rate variable between the two levels controls the migration rate between the compartments.

If we were to express the populations of Gilpin's model in terms of their energy content, we could model the process of feeding as an energy (material) flow from the two prey levels to the predator level with a branch–off at the valves denoting the feeding efficiency factors. However, the System Dynamics methodology is poorly equipped to model energy flows in a decent manner. If that is what we have in mind, we are much better advised to construct a bond graph.

The structure diagram of Fig.11.2 was constructed using the modern and beautifully engineered System Dynamics modeling system, STELLA [11.15]. During the simulation, STELLA animates the graph by showing the amount of stock currently accumulated in each of the levels, and by depicting with "needle gauges" those rate and converter variables the values of which vary over time. Fig.11.2 was captured immediately after a simulation run was completed. It contains the animation information at final time.

STELLA has been nicely integrated into the MacIntosh workbench environment. It makes optimal use of the object orientation of the MacIntosh operating environment, and it makes creating System Dynamics models a joy. Contrary to the older days when the graphical representation tools of System Dynamics were auxiliary tools designed to help the System Dynamics modeler to develop his or her models using paper and pencil, and then left it up to the user to convert the resulting models into a state–space description, in STELLA, all these design tools have been fully integrated into the software. The user designs her or his System Dynamics model on the screen using an object oriented schematic capture program. S/he can design his or her model in a strictly top down manner, i.e., s/he can start with what s/he knows about her or his system. Unknowns

are denoted with a question mark. S/he can then stepwise refine his or her model until all question marks have disappeared. At such time, STELLA can convert the model to a state–space model, and can simulate the resulting state–space model using traditional simulation techniques.

STELLA demonstrates drastically the distinction between modeling and simulation. *Modeling* is the art of capturing physical, and other, phenomena in a mathematical language. The *modeling software* supports the act of modeling, i.e., it helps the user to formulate and formalize his or her conceptions about the functioning of the real system in terms of a mathematical language. *Simulation* is the art of applying mathematical descriptions of stimuli to the mathematical description of the system, and of performing manipulations on these mathematical descriptions in such a way that *model behavior* is being extracted which resembles the *system behavior* that we would experience if we were to apply the real stimuli to the real system. The *simulation software* supports the act of simulation, i.e., it helps us to transform the mathematical description of the model together with the mathematical description of the input stimuli into trajectory behavior.

STELLA provides us with an excellent modeling environment. STELLA provides us also with a (much less great) simulation environment. Why STELLA is less convincing in its simulation capabilities than in its modeling capabilities will be shown in due course. However, this problem can be fixed. The important issue is that we realize that the two methodologies: modeling and simulation, are *fundamentally different*, and that it is possible to code them in the software separately in two distinct modules. Most modeling and simulation tools mix the modeling aspects with the simulation aspects. STELLA (and DYMOLA) are commendable exceptions to this commonly found confusion. As a consequence, it will be possible to fix what is wrong with STELLA's simulation engine in a manner that is entirely transparent to the user. This can furthermore be achieved without modifying a single line of code in the modeling software.

One of the strongest arguments in favor of STELLA is its carefully written manual which I strongly recommend for further reading [11.15].

11.5 Structure Characterization

The next question is how do we go about determining the *structure* of our state equations, i.e., how do we determine the precise nature of the relations that influence the rate variables. This process is called *structure characterization.*

One of the real strengths of the System Dynamics methodology is that it allows us to blend deductive with inductive modeling techniques. Whenever we possess physical insight into how a rate variable is influenced by the system, we should use it by all means. This is the best thing that can happen to us. However, let us discuss the case where we lack such intuition. What shall we do about this problem?

Let us return to the world model. Our laundry list suggested that the birth rate BR depends on a number of factors, namely the population P, the material standard of living MSL, the food ratio FR, the crowding ratio CR, and the pollution POL:

$$BR = f(P, MSL, FR, CR, POL) \tag{11.7}$$

An important concept in System Dynamics is the exploitation of *small signal analysis.* If we don't know enough about a system as a whole, maybe we can at least capture the behavior of that system in the vicinity of its current system state. Accordingly, in each state equation, we capture the "normal" (i.e., known) behavior, and we model alterations from the normal state as small signals. Applied, to eq(11.7), we write:

$$BR = BRN \cdot f(P, MSL, FR, CR, POL) \tag{11.8}$$

where BRN denotes the *normal birth rate*, i.e., the birth rate observed at the time when the model was created, say 1970.

Physical intuition dictates that the birth rate must be proportional to the population, thus:

$$BR = BRN \cdot P \cdot f(MSL, FR, CR, POL) \tag{11.9}$$

Now, we are at the end of our physical insight. We want to assume that eq(11.9) expresses a *static relationship* among all variables. If such an assumption is unjustified, we have chosen our levels incorrectly.

The next principle observed in most System Dynamics models is that the modeling task can be simplified if we postulate that all

variables influence the state equation *independently* of each other, i.e.:

$$BR = BRN \cdot P \cdot f_1(MSL) \cdot f_2(FR) \cdot f_3(CR) \cdot f_4(POL) \qquad (11.10)$$

Of course, this is a gross simplification which is in no way supported by physical evidence. Yet, it makes our lives easier, and since we don't have anything better to go by, this may be preferable to having no model at all. At least, it reduces drastically the amount of measurement data that we shall need in order to come up with models for the functions f_i. If we need n measurement points to decently identify any one of the functions f_i, we need $4n$ measurement points to identify all four functions as compared to n^4 points to identify the combined multivariable function.

Due to the assumption of small signal analysis, each of these functions must pass through the point $< 1.0, 1.0 >$, i.e., we normalize each of the influencing variables such that, in the year 1970, it assumes a value of 1.0. For example, the global material standard of living in the year 1970 is 1.0. The effect of the material standard of living on the birth rate in the year 1970 must thus also be 1.0, since *BRN* denotes the "normal" birth rate, i.e., the global birth rate in the year 1970.

If we define the *absolute material standard of living* as the yearly income of an individual, *MSL(t)* can be computed as the absolute material standard of living of the average inhabitant of this planet at time t divided by the absolute material standard of living of the average inhabitant of this planet in the year 1970.

How do we determine $f_1(MSL)$? We compartmentalize our world in the year 1970. We find that the material standard of living in the year 1970 differs drastically from one country to another, and so does the birth rate. Thus, we correlate the birth rate in different countries with the observed material standard of living in those countries, and voilà, the desired function f_1 has been identified. It turns out that the less money people have to raise children, the more they seem to enjoy having them.

This example shows drastically the dangers of this type of modeling. The love story cliff is treacherously close. It is our responsibility to ensure a proper cause/effect relation between the variables that we correlate in this manner. Numerous reasons can be mentioned why people in the third world produce more children:

(1) They have, on the average, a poorer education and don't understand birth control so well.

(2) They are too poor to buy a proper retirement policy, and children are often the only way to ensure their material well–being when they are old.

(3) Children are often their only means of entertainment. In first world countries, children are frequently experienced as hampering a successful career. Thus, potential parents must be truly convinced that children is what they really want. Otherwise, they may be better of without them.

All these pieces of "physical" insight get diluted in the global generalization of a relation between the material standard of living and the birth rate.

Moreover, a correlation between variables does not prove the existence of a direct causal relation at all. Maybe, both variables are caused by yet another factor which influences both variables simultaneously. Just for fun, I once correlated the statistics of birth rates in Switzerland over the past 50 years with the statistics on stork populations — and indeed, a strong positive correlation could be observed among these two variables. I leave the conclusion to the reader. We shall talk more about *causality* in the next section of this chapter.

While we can leave static functional relationships among variables in a tabular form, and simply interpolate from th table during the simulation, a technique which was introduced in Chapter 5, it is sometimes useful to determine an explicit formula that captures the functional relationship. Fig.11.3 shows a set of "measured" data that relates a variable y to a variable x.

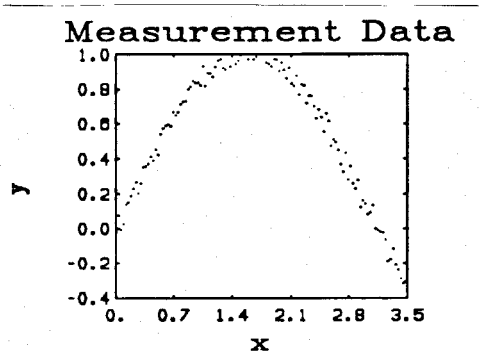

Figure 11.3. A set of "measured" data of y as a function of x

These "measured" data were produced in CTRL–C (MATLAB) by adding 5% noise to both the input and the output of a sine wave generator:

```
[> x = 0 : 0.03 : 3.5;
[> k1 = 0.1 * RAND(x) - 0.05 * ONES(x);
[> k2 = 0.1 * RAND(x) - 0.05 * ONES(x);
[> y = SIN(x + k1) + k2;
```

Let us immediately forget where these data came from. The question of interest is whether or not we can identify the structure of the source from the measured data. We shall test the following three hypotheses:

$$y_a = \sin(x) \tag{11.11a}$$
$$y_b = a + b \cdot x + c \cdot x^2 \tag{11.11b}$$
$$y_c = a + b \cdot x + c \cdot x^2 + d \cdot x^3 \tag{11.11c}$$

We can identify an optimal set of parameters using regression analysis. Let me illustrate the concept by means of hypothesis #3.

Assuming that we have n measurements, we can plug all these n measurement data into the hypothetical equation:

$$y_{c_1} = a + b \cdot x_1 + c \cdot x_1^2 + d \cdot x_1^3 \tag{11.12a}$$
$$y_{c_2} = a + b \cdot x_2 + c \cdot x_2^2 + d \cdot x_2^3 \tag{11.12b}$$
$$\vdots$$
$$y_{c_n} = a + b \cdot x_n + c \cdot x_n^2 + d \cdot x_n^3 \tag{11.12n}$$

which can be rewritten using a matrix notation as follows:

$$\begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} y_{c_1} \\ y_{c_2} \\ \vdots \\ y_{c_n} \end{pmatrix} \tag{11.13}$$

or:

$$\mathbf{V(x)} \cdot \mathbf{coef} = \mathbf{y} \tag{11.14}$$

where $\mathbf{V(x)}$ is a so–called Vandermonde matrix spanned over the vector $\mathbf{x}$. Assuming that $n > 4$, we are confronted with an overdetermined linear regression problem that we can solve in a least square's sense:

$$\mathbf{V(x)} \cdot \mathbf{coef} - \mathbf{y} = \mathbf{res} \tag{11.15}$$

where **res** is the vector of residua. We try to determine the coefficient vector **coef** such that the L_2–norm of res:

$$||\mathbf{res}||_2 = \sqrt{\sum_{\forall i} res_i^2} \tag{11.16}$$

is minimized. This problem can be solved by computing the *pseudoinverse* of the rectangular matrix $\mathbf{V(x)}$:

$$\left(\mathbf{V(x)}' \cdot \mathbf{V(x)}\right) \cdot \mathbf{coef} = \mathbf{V(x)}' \cdot \mathbf{y} \tag{11.17}$$

where $\mathbf{V(x)}' \cdot \mathbf{V(x)}$ is a square Hermitian matrix of size 4×4 which can be inverted:

$$\mathbf{coef} = \left(\mathbf{V(x)}' \cdot \mathbf{V(x)}\right)^{-1} \cdot \mathbf{V(x)}' \cdot \mathbf{y} \tag{11.18}$$

where $\left(\mathbf{V(x)}' \cdot \mathbf{V(x)}\right)^{-1} \cdot \mathbf{V(x)}'$ is called the *pseudoinverse* of $\mathbf{V(x)}$. In MATLAB or CTRL–C, this problem can be conveniently coded using the "\" operator:

$$[> \quad coef = V \backslash y \tag{11.19}$$

For the given data set, the following optimal functions were found:

$$y_a = \sin(x) \tag{11.20a}$$
$$y_b = -0.0288 + 1.2511 \cdot x - 0.3938 \cdot x^2 \tag{11.20b}$$
$$y_c = -0.0833 + 1.4434 \cdot x - 0.5325 \cdot x^2 + 0.0266 \cdot x^3 \tag{11.20c}$$

Fig.11.4 shows the three fitted curves plotted as solid lines over the measured data set. Obviously, all three hypotheses can be easily defended on the basis of Fig.11.4. I also computed the L_2–norms of the three residua vectors and found:

$$||\mathbf{res}_1||_2 = 0.3967 \tag{11.21a}$$
$$||\mathbf{res}_2||_2 = 0.5395 \tag{11.21b}$$
$$||\mathbf{res}_3||_2 = 0.4857 \tag{11.21c}$$

The residuum of the sine wave fit is slightly better, but not sufficiently so to guarantee that this is the correct hypothesis. I also

computed the cross–correlations between x and y for the three hypotheses, and found that the largest cross–correlations are as follows:

$$\|\mathbf{corr}_1\|_\infty = 52.7961 \tag{11.21a}$$
$$\|\mathbf{corr}_2\|_\infty = 52.6210 \tag{11.21b}$$
$$\|\mathbf{corr}_3\|_\infty = 52.6761 \tag{11.21c}$$

Again, the sine wave fit turned out to be slightly better, but this test shows even less significant differences than the last one.

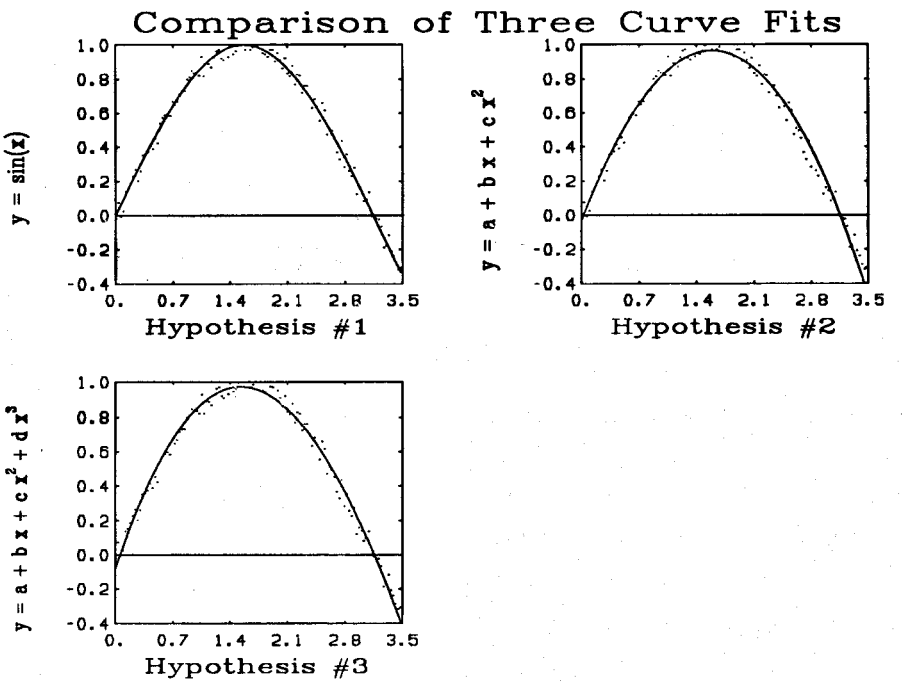

Figure 11.4. Curve fitting of "measured" data set

Ivakhnenko and Lapa devised a technique which often allows us to distinguish between different hypotheses regarding functional relationships [11.11]. They suggested that we apply a transformation to both the data and the hypothesized curves such that, after the transformation, the hypothesized curves look sufficiently different, so that the measured data can be easily associated with one, but not with the others of the proposed hypotheses.

A good transformation for our case might be the following: We remember that $\sin^2(x) + \cos^2(x) = 1.0$. We therefore apply the following transformation to the data:

$$\mathbf{y_{new}} = \mathbf{y}^2 + \cos^2(\mathbf{x}) \tag{11.22}$$

The three hypotheses turn into:

$$y_{new_a} = 1.0 \tag{11.23a}$$
$$y_{new_b} = (a + b \cdot x + c \cdot x^2)^2 + \cos^2(x) \tag{11.23b}$$
$$y_{new_c} = (a + b \cdot x + c \cdot x^2 + d \cdot x^3)^2 + \cos^2(x) \tag{11.23c}$$

Fig.11.5 shows the transformed data.

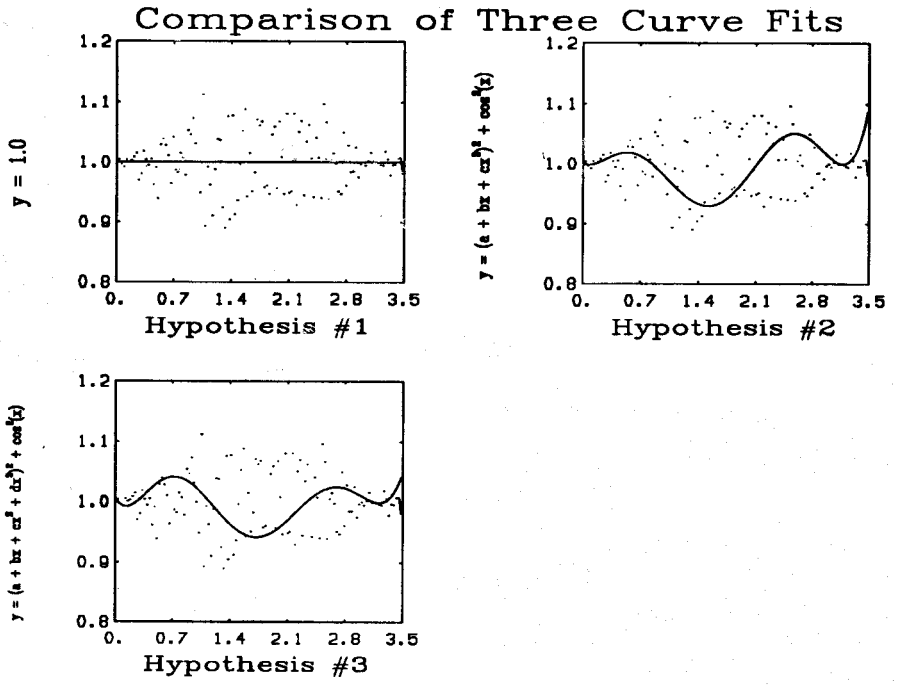

Figure 11.5. Curve fitting of transformed data set

Clearly, the three curves can be easily distinguished from each other. Unfortunately, the "measured" data are spread so widely now that we still cannot associate them with certainty with any one of the three curves. I recomputed the residua and correlations, and found:

$$\|\mathbf{res}_1\|_2 = 0.4922 \quad , \quad \|\mathbf{corr}_1\|_\infty = 117.0747 \qquad (11.24a)$$
$$\|\mathbf{res}_2\|_2 = 0.6199 \quad , \quad \|\mathbf{corr}_2\|_\infty = 116.8677 \qquad (11.24b)$$
$$\|\mathbf{res}_3\|_2 = 0.5829 \quad , \quad \|\mathbf{corr}_3\|_\infty = 116.9146 \qquad (11.24c)$$

The situation is still unchanged. We have indications that the sine wave approximation is slightly better, but we cannot be sure that this hypothesis is physically correct. Had we added only 1% noise to our data instead of 5% noise, the applied transformation would indeed have disqualified the second and the third hypothesis.

What this example teaches us is how little it takes before structural information gets lost in measurement data. This makes the inductive structure characterization a tough and often unsolvable problem.

11.6 Causality

Causality is a very natural notion in our everyday lives. An action causes a reaction. However, in the casual context, some time usually elapses between the action and the reaction, i.e., the concept of causality is more commonly applied to discrete–event systems than to continuous–time systems.

In engineering, we usually call a response of a system "causal" if it occurs simultaneously with the input stimulus, or if it lags behind the input stimulus. We call it "non–causal" if it occurs prior to the input stimulus. Of course, non–causal responses cannot be generated by physical systems.

It is now interesting to discuss whether, given two different continuous signals, it can be decided that one of the two signals has been caused by the other. This is a favorite topic in the artificial intelligence literature which is full of sometimes rather obscure notions about causality [11.12].

Let us analyze an electrical resistor which is characterized by its voltage and its current. Does it make sense to say that the voltage across the resistor causes a current to flow, or does it make more sense to say that the current through the resistor causes a drop of potential? In the active form, *both* statements are correct. Yet, given a measured voltage across and a measured current through a resistor,

we cannot conclude after the fact which caused what. This is clearly a chicken–and–egg problem.

Let us apply the following experiment. Fig.11.6 shows three different circuit elements, a resistor, a capacitor, and an inductor connected to a noise voltage source.

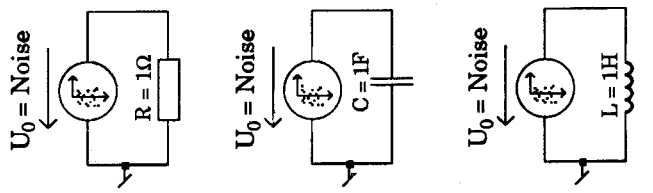

Figure 11.6. Three different circuit configurations

In all three cases, we shall measure the current that flows through the element. We shall then compute the cross–correlation between the input signal u and the output signal i in all three cases. In MATLAB or CTRL–C, this is easiest accomplished with several fast Fourier transforms. The following CTRL–C (or MATLAB) code produces the desired cross–correlation function between the input signal u and the output signal i.

```
[>  u = [u, ZROW(u)];
[>  i = [i, ZROW(i)];
[>  sui = FFT(u). * CONJ(FFT(i));
[>  rui = IFFT(sui);
[>  corr = [rui(129 : 256), rui(1 : 128)];
```

We assume that both vectors u and i contain 128 measured data points. Zero–padding is applied to provide the required memory cells for the fast Fourier transform. The vector *rui* contains the correlation function, but not in the right sequence. The first half of the vector contains increasing positive values of delay between the input and the output, and the second half of the vector contains decreasing negative values of delay between the input and the output. The last line of code puts the correlation function into the correct sequence.

Fig.11.7 shows the results obtained for the three circuit configurations.

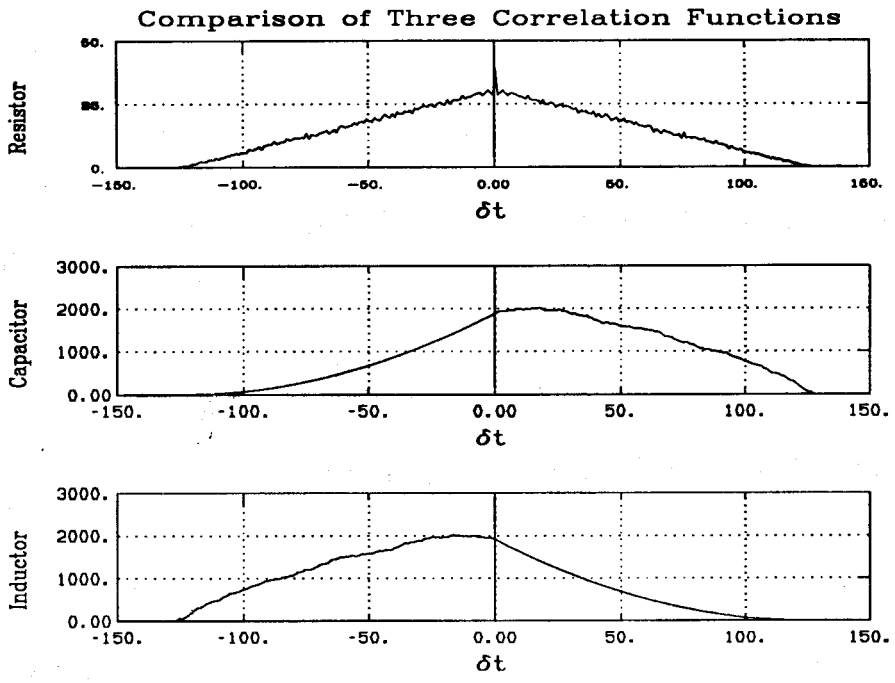

Figure 11.7. Cross–correlations for the three circuit configurations

As expected, the cross–correlation assumes its peak value at $\delta t = 0.0$ for the resistor. It assumes its peak value for $\delta t > 0.0$ for the capacitor, and it assumes its peak value for $\delta t < 0.0$ for the inductor. Consequently, we can say that the current "lags behind" the voltage in the inductor, and it "leads" the voltage in the capacitor.

We might therefore be inclined to say that, in an inductor, the voltage causes current to flow, whereas in a capacitor, the current causes a drop of potential. The resistor can assume either of the two causalities. This definition is consistent with our earlier use of the term "causality" as we came across it in the context of *bond graph modeling*, and, from a computational point of view, this terminology makes a lot of sense.

Yet notice that, in the above experiment, the voltage truly "caused" the current to flow in all three situations. In every single case, the voltage source was the physical origin of the observed phenomenon. Does this mean that the capacitor configuration is a "non–physical system"? Of course not! Obviously, we can connect a

voltage source to a capacitor. However, the system is indeed *structurally singular*. We are made believe that the capacitor can store energy, but, in the given configuration, it cannot. It simply attempts to differentiate the input.

Did we just discover that a physical system can indeed anticipate the input, i.e., that physical systems can have precognitions or at least premonitions of coming events? The answer is: in some way, yes. If we assume that before time $t = 0.0$, every single signal in the system was zero (a common notion in many areas of engineering design, such as in control theory), our system has no way of knowing what is to come. If we try to apply a step voltage input to our capacitor, we have difficulties to do so since this would imply an infinitely large current at time zero which would require an infinite amount of power to be put in the system. For sure, the system will not "accommodate" our desire by starting the current a little earlier. In this sense, the configuration is indeed non–physical. Yet, in a statistical sense, i.e. under *statistical steady–state conditions*, we can measure a cross–correlation function in which the current leads the voltage. In this context, the system can indeed anticipate future inputs. In other words, knowledge of the past behavior of a system allows us to anticipate future behavior, and in a statistical sense, this even includes future inputs. To obtain the capacitor curve of Fig.11.7, I cheated a little bit. I applied the same experiment as for the inductor, and then simply swapped the input and the output. However, I also tried the experiment with numerical differentiation. Due to the (even theoretical) impossibility of computing the derivative of a noise signal, the correlation is much smaller, but the current still leads.

We just learned that the discrimination of true causality from measurements alone is a hopeless undertaking. However, for practical purposes, we shall *define* the terms *causal* and *causality* as follows:

(1) Two signals are coupled through a *causal relation* if the (either positive or negative) cross–correlation between them is strong.

(2) In addition, two signals are coupled through a *causality relation* if one signal lags behind the other.

(3) If two signals show little or no cross–correlation, we call them causally unrelated.

(4) If two signals show a strong cross–correlation, but neither of the two signals lags behind the other, the signals are causally related, but their causality cannot be decided.

(5) If two signals show a significant amount of cross–correlation, and if signal #2 lags behind signal #1, we claim that signal #1 has caused signal #2. The stronger the cross–correlation of the two signals, the more pronounced is the causal relation among the two signals. The larger the lag time, the stronger is their causality relation.

11.7 Differential vs. Difference Equations

Traditionally, System Dynamics researchers have always formulated their state–space models as a set of *difference equations.* This has two reasons:

(1) Originally, System Dynamics had been closely linked with the DYNAMO simulation language [11.14] which solves only difference equations and no differential equations.

(2) It has been often claimed that the major application areas for System Dynamics are found in management and in soft sciences. The quality of the data available in these fields does not justify the accuracy that numerical integration provides. Difference equations are cheaper to solve, and they will serve the same purpose.

In DYNAMO, a difference equation for the change in population would be formulated in the following way:

$$P.K = P.J + (DT)(BR.JK - DR.JK) \tag{11.25}$$

The index K denotes the next time step, whereas the index J indicates the current time step. JK denotes the interval between the current and the next time step. Obviously, we can reinterpret eq(11.25) as follows:

$$P_{k+1} = P_k + \Delta t \cdot (BR - DR) \tag{11.26}$$

which is equivalent to our differential equation:

$$\dot{P} = BR - DR \tag{11.27}$$

if the fixed step forward Euler algorithm:

$$P_{k+1} = P_k + \Delta t \cdot \dot{P}_k \tag{11.28}$$

is used for the numerical integration. DYNAMO operates on state equations, but forces the user to explicitly formulate her or his integration algorithm. Usually, this will be a fixed–step forward Euler algorithm. We can trick DYNAMO into the use of higher order integration algorithms if we code them explicitly as difference equations. This is most unfortunate. Why should we have to code the integration algorithm manually if good simulation languages, such as ACSL, exist that will take care of the numerical integration for us? DYNAMO is simply a very old–fashioned language, and there is no excuse for using it any longer (as a matter of fact, there has not been an excuse for using it for quite some time already).

What about the second argument? It is indeed correct that most System Dynamics applications do not justify a sophisticated integration algorithm to be used from the point of view of a consistent data representation. However, we shall see in the second volume of this textbook that the numerical accuracy of an integration scheme is dictated by three separate components: consistency, convergence, and numerical stability. A numerical scheme is called *consistent* if the analytical (i.e., infinitely accurate) solution of that scheme decently agrees with the analytical solution of the original problem. A scheme is called *convergent* if the local error of the scheme goes to zero if the discretization interval Δt approaches zero. The scheme is called *numerically stable* if the local errors do not accumulate over many steps, i.e., if the local error is a good indicator of the global error as well. From a point of view of consistency, it is perfectly legitimate to represent typical System Dynamics problems through a set of coarsely discretized difference equations. However, the requirement of numerical stability may still suggest the use of a sophisticated higher order numerical integration scheme, even though the accuracy requirements of our model are low.

As I explained earlier, System Dynamics is a *modeling* methodology, which is totally unrelated to the underlying *simulation* methodology. STELLA does not use DYNAMO as its simulation engine. The developers of STELLA recognized rightly that System Dynamics models can (and should) be mapped into state–space models. STELLA currently offers three different numerical integration rules. The default method is forward Euler (like in DYNAMO), but in addition, STELLA offers two fixed step Runge–Kutta algorithms, one of second order, and one of fourth order. However, this is insufficient. Variable step algorithms are mandatory for many non–linear problems in order to obtain accurate simulation results at a decent execution speed. Also, the manner in which the integration algorithms are implemented can have drastic effects on their execution

speed. In fact, little reason exists why a developer of a modeling tool should have to reinvent the wheel and create his or her own simulation engine as well. It would have been equally easy to map the state–space model that is being created by STELLA's excellent modeling engine into a decent simulation language such as ACSL. ACSL has been developed over many years, and while ACSL still has its problems, its simulation engine is far better than what the average software designer could possible come up with on her or his own.

To prove my point, I ran the Gilpin model of Fig.11.2 in STELLA. Fig.11.8 shows the results for a joint competition factor of $k = 1.0$.

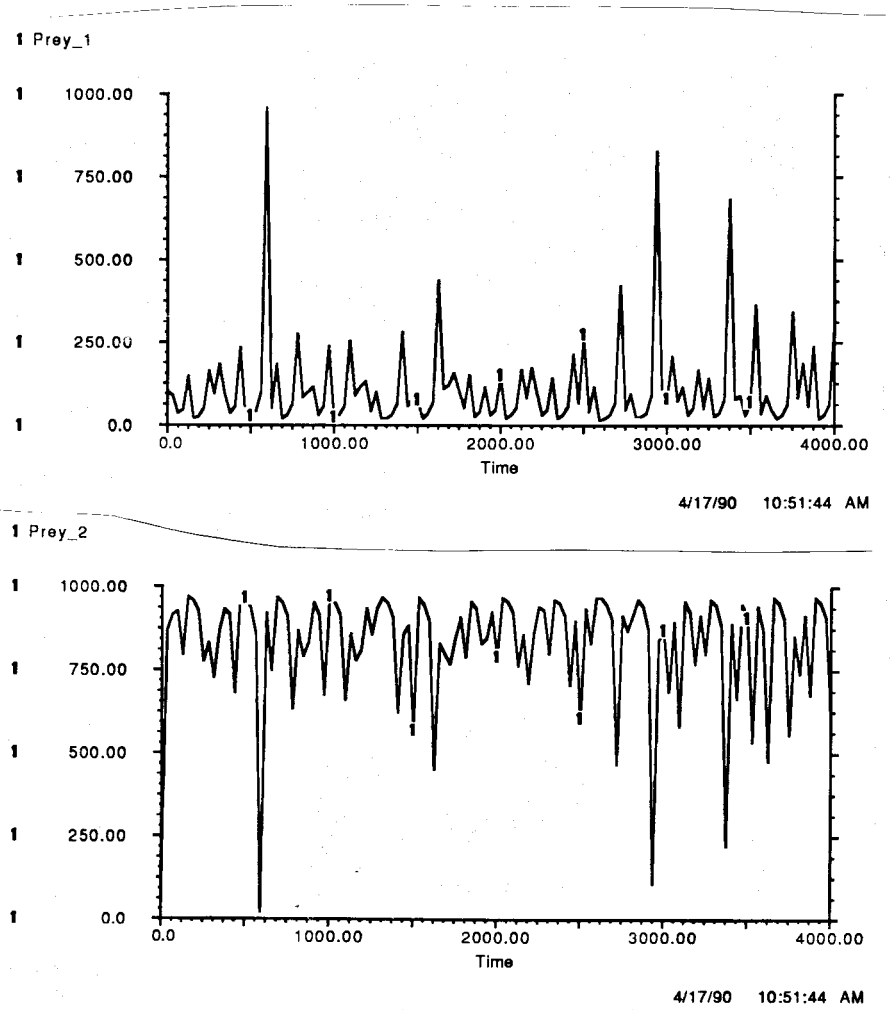

Figure 11.8a. Gilpin's model simulated in STELLA

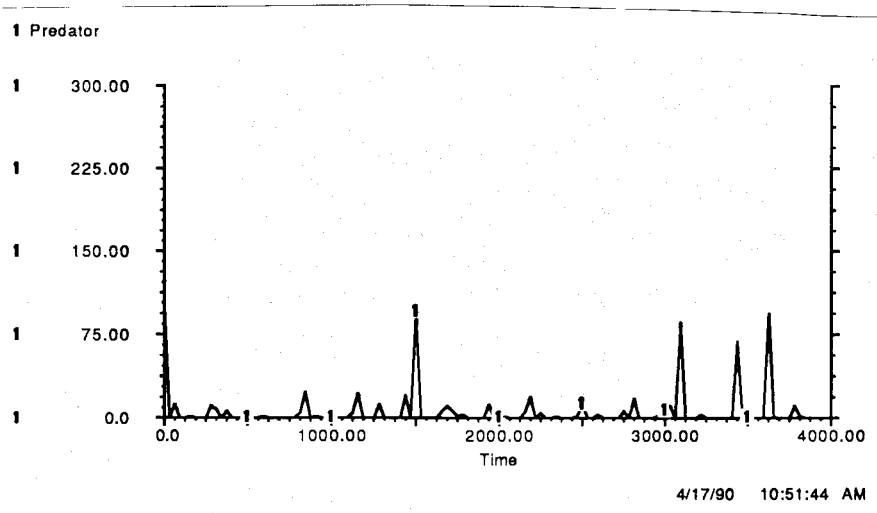

Figure 11.8b. Gilpin's model simulated in STELLA (continued)

In order to obtain any meaningful results at all, I had to use the fourth order Runge–Kutta algorithm with a step size of $\Delta t = 1.0$. The results shown in Fig.11.8 are still not very accurate. We would have to use a considerably smaller step size to e.g. accurately determine the competition factor $k = 1.089$ at which the first prey x_1 dies out. Yet, I could not reduce the step size any further since one simulation run required already an execution time of 30 *min* on my MacPlus. The ACSL program employed in Chapter 10 required an execution time of 8 *sec* for a fourth order Runge–Kutta algorithm with an *average* step size of $\Delta t_{avg} = 0.125$ on a VAX–8700. However, during each of the peaks, ACSL reduced the step size to a value of roughly $\Delta t_{min} = 10^{-6}$. Of course, the VAX–8700 is a substantially faster (and more expensive) machine than my MacIntosh. However, the observed speed difference is primarily caused by the fact that STELLA *interprets* the state–space model at run time rather than *compiles* it beforehand at least into threaded code (as DESIRE does). Consequently, STELLA can be used for fairly simple models only.

However, it would be an easy task to cure this deficiency. All that needs to be done is to turn STELLA into a program generator (similar to DYMOLA) which automatically generates an ACSL program or a DESIRE program from the state–space model, and uses that for simulation. This would speed up the execution time of STELLA simulation runs by approximately a factor of 100.

Also, *discrete interventions* are common elements of System Dynamics models. We shall meet one such discrete intervention in a later section of this chapter in which we shall model the spread of an influenza epidemic. In STELLA, the user is forced to employ rather clumsy ways to implement such interventions. This is due to the fact that STELLA does not know the concept of *discrete events*. A "discrete intervention" is simply a *time–event* in terms of our previously used nomenclature. Since ACSL provides for explicit mechanisms to describe time–events, the resulting models could be made simpler, more elegant, and more robust if ACSL were used as the target language. The interpretive way in which STELLA's simulation engine currently operates is fine for model debugging purposes and for simple models, i.e., it would make sense to retain this capability as an alternative, and add the program generation capability to the code as a new feature.

11.8 The Larch Bud Moth Model

In Chapter 10, we discussed a population dynamics model of the larch bud moth, *Zeiraphera diniana* (Guenée) as observed in the upper Engadine valley of Switzerland. At that time, we used several simple two (three) species predator–prey models to describe the limit cycle behavior of the insect population.

Let us repeat this analysis now using the System Dynamics modeling methodology which enables us to incorporate into the model as much physical knowledge about the dynamics of the system as we are able to gather.

It is known that (in the upper Engadine valley) the female insects deposit their eggs in the larch trees during the month of August every year, which then stay in an extended *embryonic diapause* that lasts until the spring of the following year. During the fall, the eggs are preyed upon by several species of *Acarina* and *Dermaptera*. During the winter, the eggs are naturally parasitized by a species of *Trichogramma*. The surviving eggs are ready for hatching in June. Extensive studies have shown the winter mortality to be a constant fraction of the deposited eggs:

$$Small\_Larvae = (1.0 - winter\_mortality) \cdot Eggs \tag{11.29}$$

The average winter mortality is 57.28%. Many of the small larvae die from *incoincidence*, i.e., from a bad location, for example, if the

branch where the egg was deposited dies during the winter. Measurements have shown that the incoincidence factor depends linearly on the raw fiber content of the needles of the larch tree:

$$incoincidence = 0.05112 \cdot Rawfiber - 0.17932 \tag{11.30}$$

where the coefficients were found by linear regression. The raw fiber content is expressed in %. It varies between $Rawfiber_{min} = 12\%$ for healthy trees, and $Rawfiber_{max} = 18\%$ for heavily damaged trees. From this, we can compute the number of larvae that survive to become large larvae:

$$Large\_Larvae = (1.0 - incoincidence) \cdot Small\_Larvae \tag{11.31}$$

The large larvae move around and cause damage to the larch forest by eating the needles. Some of the large larvae die from *starvation*, i.e., if their food demand is not met. Others die due to *physiological weakening*. If the raw fiber content is too high, they may still survive the larvae state and cause more damage by eating, but they may die during their chrysalis state. We can thus compute the number of adult insects as follows:

$$Insects = (1.0 - weakening) \cdot (1.0 - starvation) \cdot Large\_Larvae \tag{11.32}$$

It was observed that the logarithm of the starvation factor depends linearly on the quotient of *foliage* and *food demand*:

$$starvation = \exp(-\frac{foliage}{food\_demand}) \tag{11.33}$$

and that the weakening factor is again linearly dependent on the raw fiber content of the needles:

$$weakening = 0.124017 \cdot Rawfiber - 1.435284 \tag{11.34}$$

The foliage is linear in the raw fiber content and in the number of trees:

$$foliage = (-2.25933 \cdot Rawfiber + 67.38939) \cdot nbr\_trees \tag{11.35}$$

where the number of trees was counted to be $nbr\_trees = 511147$. The food demand is linear in the number of large larvae:

$$food\_demand = 0.005472 \cdot Large\_Larvae \tag{11.35}$$

Since only female insects contribute to the next generation, we compute:

$$Females = sex\_ratio \cdot Insects \tag{11.36}$$

where the average percentage of female insects was found to be 44%, thus $sex\_ratio = 0.44$. The number of eggs deposited for the next generation of insects can be computed as:

$$New\_Eggs = fecundity \cdot Females \tag{11.37}$$

where the *fecundity* depends again on the raw fiber content of the needles. If the raw fiber content was high, the weight of the chrysalis will be low. Such insects may still survive, but they exhibit a lower fecundity than insects that were nourished well during their larvae state:

$$fecundity = -18.475457 \cdot Rawfiber + 356.72636 \tag{11.38}$$

This concludes the life cycle of the larch bud moth.

Let us now analyze the life cycle of the trees. The damage caused by the insects can be expressed as follows:

$$defoliation = (1.0 - starvation) \cdot (\frac{food\_demand}{foliage}) \tag{11.39}$$

The raw fiber *recruitment*, i.e., the change in needle quality is a complex function of the current needle quality and the current defoliation:

$$New\_Rawfiber = grecr(defoliation, Rawfiber) \cdot Rawfiber \tag{11.40}$$

The function *grecr* is an experimental function which models the change in raw fiber content as a function of defoliation. If little defoliation takes place, the trees will slowly recover to their optimal raw fiber content of 12%. If heavy defoliation takes place, the raw fiber content slowly increases until it reaches its maximum value of 18%. Fischlin and Baltensweiler modeled this phenomenon in the following manner [11.4]:

IF $defoliation < 0.4$
  THEN IF $Rawfiber < 11.99$
    THEN $grecr = 1.0$
    ELSE $zRaw = 0.425 + abs(\frac{18.0 - Rawfiber}{Rawfiber - 11.99})$
      IF $zRaw > (Rawfiber - 11.99)$
        THEN $grecr = \frac{11.99}{Rawfiber}$
        ELSE $grecr = 1.0 - \frac{zRaw}{Rawfiber}$
        END IF
    END IF
  ELSE IF $defoliation < 0.8$
    THEN $grecr = 1.0 + \frac{(defoliation - 0.4) \cdot (18.0 - Rawfiber)}{0.4 \cdot Rawfiber}$
    ELSE $grecr = \frac{18.0}{Rawfiber}$
    END IF
  END IF

This completes the description of the model. While regression analysis was used on several occasions to determine optimal parameter values, this is basically a *physical model* of the population dynamics system. Parameter fitting was applied in a strictly local manner to fit no more than two parameters at a time to a set of input/output measurements. I am therefore much more inclined to believe that I understand now what is going on in this system than I was after identifying the (much simpler) Lotka–Volterra model of Chapter 10.

Let us go ahead and try to capture this model in STELLA. Fig.11.9 shows the structure diagram of this model:

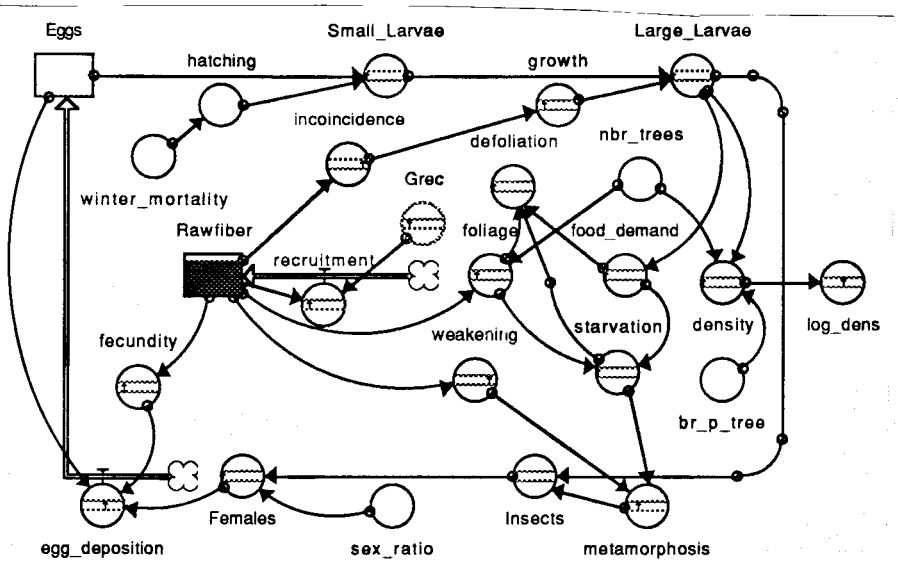

Figure 11.9. Structure diagram of the Larch Bud Moth model

With the state variables, I had to cheat a little bit. STELLA is designed to solve differential equations rather than difference equations, thus, by modeling *Eggs* as a level variable and *egg_deposition* as a rate variable, STELLA forces us to accept the equation:

$$\frac{d}{dt}Eggs = egg\_deposition \tag{11.41}$$

or, by using the (standard) fixed step Euler algorithm:

$$New\_Eggs = Eggs + \Delta t \cdot egg\_deposition \tag{11.42}$$

Comparing eq(11.42) with eq(11.37), we find that:

$$egg\_deposition = fecundity \cdot Females - \frac{Eggs}{\Delta t} \tag{11.43}$$

In the same manner, we find that:

$$recruitment = Grecr \cdot Rawfiber - \frac{Rawfiber}{\Delta t} \tag{11.44}$$

In our case, Δt was set to 1.0.

I had more problems with the function *Grecr*. STELLA does not provide us with procedural sections in the way ACSL does. This is a common disease of most graphically oriented modeling languages. I had to code the function in the following way:

$Grec$ = **IF** $(defoliation < 0.4)$ **THEN** $Grec1$ **ELSE** $Grec2$
$Grec1$ = **IF** $(Rawfiber < 11.99)$ **THEN** $Grec3$ **ELSE** 1.0
$Grec2$ = **IF** $(defoliation < 0.8)$ **THEN** $Grec4$ **ELSE** $Grec5$
$Grec3$ = **IF** $(zRaw < (Rawfiber - 11.99))$ **THEN** $Grec6$ **ELSE** $Grec7$
$Grec4 = 1.0 + (defoliation - 0.4) * (18.0 - Rawfiber)/(0.4 * Rawfiber)$
$Grec5 = 18.0/Rawfiber$
$Grec6 = 11.99/Rawfiber$
$Grec7 = 1.0 - zRaw/Rawfiber$
$zRaw = 0.425 +$ **ABS**$((18.0 - Rawfiber)/(Rawfiber - 11.99))$

Since I didn't want to clutter my nice structure diagram with all this detail, I used STELLA's *ghost* feature. I created ghosts of *Rawfiber* and *defoliation*, and computed *Grecr* in a separate structure diagram (within the same model) as shown in Fig.11.10.

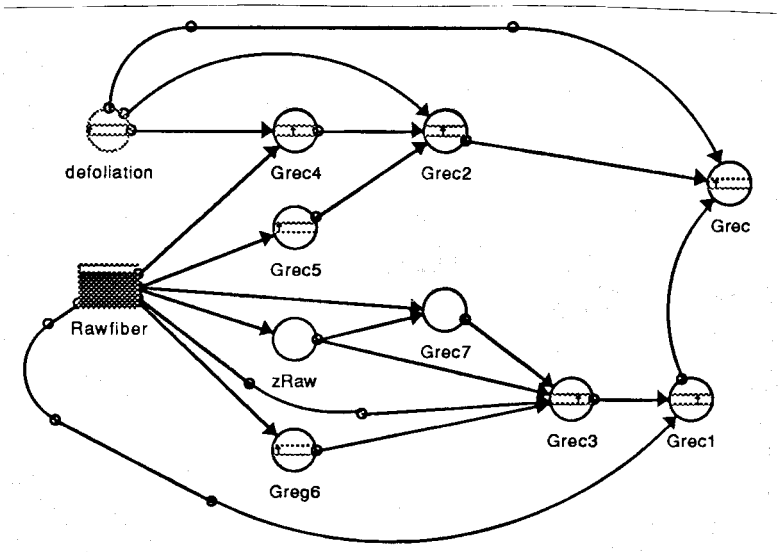

Figure 11.10. Structure diagram of the Grecr function

Finally, I made a ghost of *Grecr*, and copied it back into my original structure diagram. This is a way to create STELLA submodels. Of course, if so much detail is provided to and desired in a model such as expressed in the *Grecr* function, System Dynamics may not be the most appropriate modeling methodology any longer. It would have been equally easy to take the equations as given, and code an ACSL program directly.

Finally, I added yet another separate structure diagram to capture the measurement data such that they can be plotted together with the simulated data on one graph:

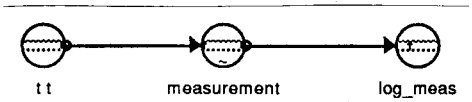

Figure 11.11. Structure diagram of the measurement data

The *tt* block contains simply the simulation clock:

$$tt = TIME \tag{11.45}$$

and *measurement* is a tabular function that contains the measurement data. The measurement data is given in Table 11.2.

Table 11.2 Measurement data for the Larch Bud Moth model

time [*year*]	*larvae density* [#/*kg*]
1949	0.018
1950	0.082
1951	0.444
1952	4.174
1953	68.797
1954	331.760
1955	126.541
1956	21.280
1957	2.246
1958	0.085
1959	0.080
1960	0.371
1961	1.638
1962	22.878
1963	248.817
1964	184.272
1965	3.116
1966	0.019
1967	0.002
1968	0.059
1969	0.197
1970	1.068
1971	10.569
1972	173.932
1973	249.612
1974	176.023
1975	4.749
1976	0.014
1977	0.008
1978	0.056

Notice that STELLA allows us to maintain several unconnected structure diagrams in the same model.

Finally, we need to compute the output function of our model. The measurement data were expressed in number of larvae per kilogram dry needle biomass, i.e., we must compute the larvae density:

$$density = \frac{Large\_Larvae}{br\_p\_tree \cdot nbr\_trees} \tag{11.46}$$

where $br\_p\_tree = 91.3$ denotes the amount of dry needle biomass per tree expressed in kilograms.

Since STELLA does not provide for a logarithmic format of its graphs, we need to compute, in the model, the logarithms of the larvae density and of our measurement data. This concludes the overall model description. Fig.11.12 shows the resulting graph which looks similar to the one given in the paper by Fischlin and Baltensweiler [11.4].

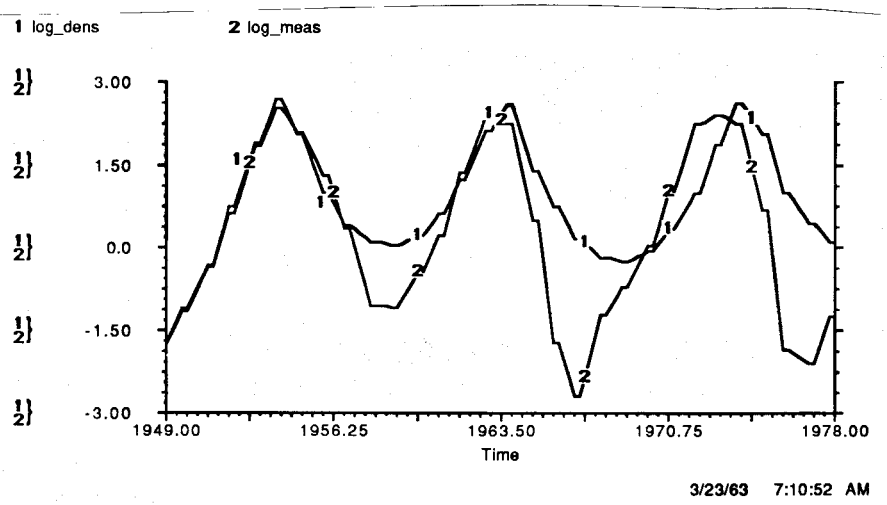

Figure 11.12. Simulation results of the Larch Bud Moth model

The graph looks a little crooked. STELLA assumes all variables to be continuous over time. Since our state variables stay constant for a period of one year (our "step size"), the graph shows a staircase function.

11.9 The Influenza Model

Let us look at another system now. We wish to study the spreading of an influenza epidemic. This model has four state variables. We start out with a population of 10000 *non-infected* individuals. At a given time, a stem of influenza bacteria is introduced into the system. Infection occurs spontaneously, and some of the previously

non–infected individuals become *infected.* The bacteria spread in the infected individuals, and after some time, they fall *sick.* The total number of contagious people (both those who are already sick, and those who haven't broken down yet) spread the disease further and recruit new infected people among the non–infected population. The sick people eventually get cured due to the natural defenses of their immune systems. This creates a population of *immune* people. However, the body "forgets" the previous exposure after some time, and also, the bacteria have a tendency to mutate. Thus, the immune people turn into non–infected people again who are susceptible to reinfection. The purpose of this example is to show how easy it is to create intuitive models using the System Dynamics methodology.

The structure diagram of this system is shown in Fig.11.13.

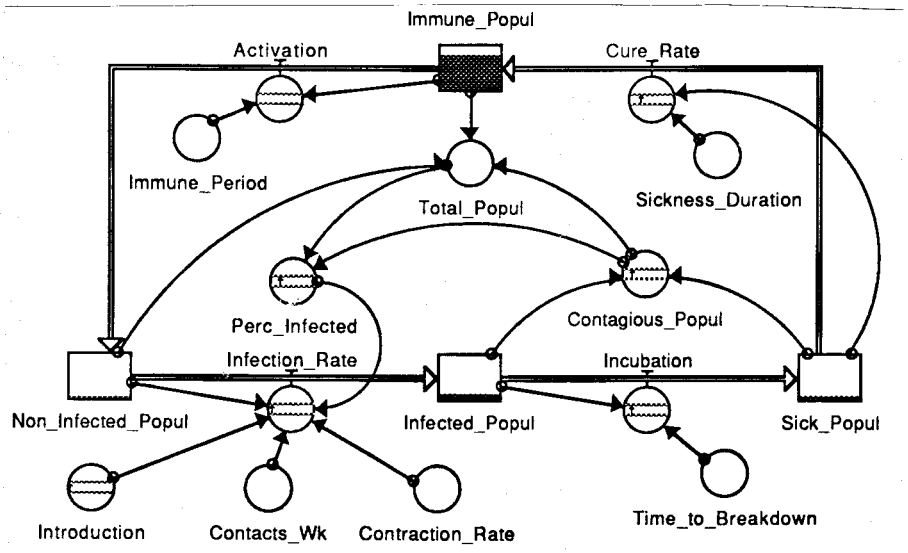

Figure 11.13. Structure diagram of the influenza model

This diagram shows a new element. Outflows of one level can be used as inflows to another. The rate between two levels controls the flow from one level to the other, as a faucet controls the flow of water from the storage tank to the sink.

The time to breakdown (incubation period) is 4 *weeks*, the actual sickness lasts for 2 *weeks*, and the immune period lasts for 26 *weeks*. This allows us to write down a number of equations already:

$$Incubation = INT(\frac{Infected\_Popul}{Time\_to\_Breakdown}) \tag{11.47a}$$

$$Cure\_Rate = INT(\frac{Sick\_Popul}{Sickness\_Duration}) \tag{11.47b}$$

$$Activation = INT(\frac{Immune\_Popul}{Immune\_Period}) \tag{11.47c}$$

A population of 10000 is not sufficient to ignore the "quantization error". The INT function ensures that every one of the four populations will always contain an *integer* number of people.

The contagious population is the sum of the infected and the sick:

$$Contagious\_Popul = Infected\_Population + Sick\_Population \tag{11.48}$$

The total population is the sum of all four populations:

$$Total\_Popul = Non\_Infected\_Popul + Contagious\_Popul + Immune\_Popul \tag{11.49}$$

which, of course, is constant (10000) in our model. The percentage of currently infected people is:

$$Perc\_Infected = \frac{Contagious\_Popul}{Total\_Popul} \tag{11.50}$$

The infection rate can be computed as the product of the non–infected population P_{NI}, the number of contacts that a non–infected person has per week C_{Wk}, the percentage of contagious people $Perc_C$, and the chance of contracting the disease in such a contact $Rate_C$:

$$Infection\_Rate = INT(P_{NI} \cdot C_{Wk} \cdot Perc_C \cdot Rate_C) \tag{11.51}$$

where the average number of weekly contacts of one person with another is constant: $C_{Wk} = 15$, and the contraction rate per contact is also assumed to be constant: $Rate_C = 0.25$.

The introduction of the disease occurs as a *discrete event* during the eight's simulated week. STELLA does not provide for a mechanism to describe discrete events. Thus, we must model the introduction with a PULSE function:

$$Introduction = PULSE(1.0, 8.0, 1E3) \tag{11.52}$$

introduces a pulse of height 1.0 at time 8.0 with a repetition frequency of 1000.0. This is simply added to the infection rate:

$$Infection\_Rate = INT(P_{NI} \cdot C_{Wk} \cdot Perc_C \cdot Rate_C + Introduction) \tag{11.53}$$

Finally, we must ensure that not more people are ever being infected than the entire pool of non–infected persons, thus:

$$Infection\_Rate = MIN(INT(P_{NI} \cdot C_{Wk} \cdot Perc_C \cdot Rate_C + Introduction), P_{NI}) \tag{11.54}$$

This concludes the description of the influenza model. Fig.11.14 shows the results of this model being simulated over a period of one year.

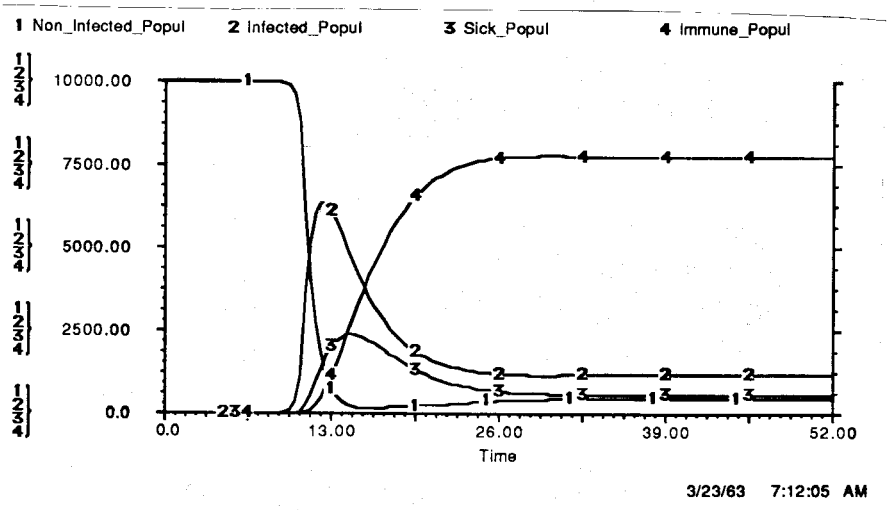

Figure 11.14. Simulation results for the influenza model

After the introduction, the influenza epidemic spreads rapidly. Within six weeks, the percentage of sick people reaches its maximum of roughly 25%. A steady–state is reached about 20 *weeks* after the introduction. Obviously, the disease does not die out naturally. A

certain percentage of the population loses immunity sufficiently fast to get reinfected before the bacteria stem has disappeared.

It might have been more realistic to model the four variables: *Contacts_Wk*, *Time_to_Breakdown*, *Sickness_Duration*, and *Immune_Period* through random functions rather than through constants, but let us save this modification for a homework problem.

11.10 Forrester's World Model

Let us now try to apply our methodology to a more ambitious problem. One of the most famous System Dynamics models ever developed and published was Forrester's world model [11.7]. How did this come about? In the late sixties, a number of concerned scientists tried to decide whether there was a way to determine the destiny of the human race. They called themselves the "Club of Rome", and they started to investigate means to determine our future on this planet. In the sequence, Jay Forrester applied his System Dynamics methodology to this problem, and published 1971 his most famous book *World Dynamics.*

Forrester decided that the "world" can be captured by five state variables (levels): *population*, *pollution*, *non-recoverable natural resources*, *capital investment*, and *percentage of the capital invested in the agricultural sector.* His simulation starts in the year 1900, and the initial conditions for the five levels are:

$$Population = 1.65 \cdot 10^9 \tag{11.55a}$$

$$Pollution = 2.0 \cdot 10^8 \tag{11.55b}$$

$$Natural\_Resources = 9.0 \cdot 10^{11} \tag{11.55c}$$

$$Capital\_Investment = 4.0 \cdot 10^8 \tag{11.55d}$$

$$CIAF = 0.2 \tag{11.55e}$$

He came up with the following structure diagram which I redrew in STELLA.

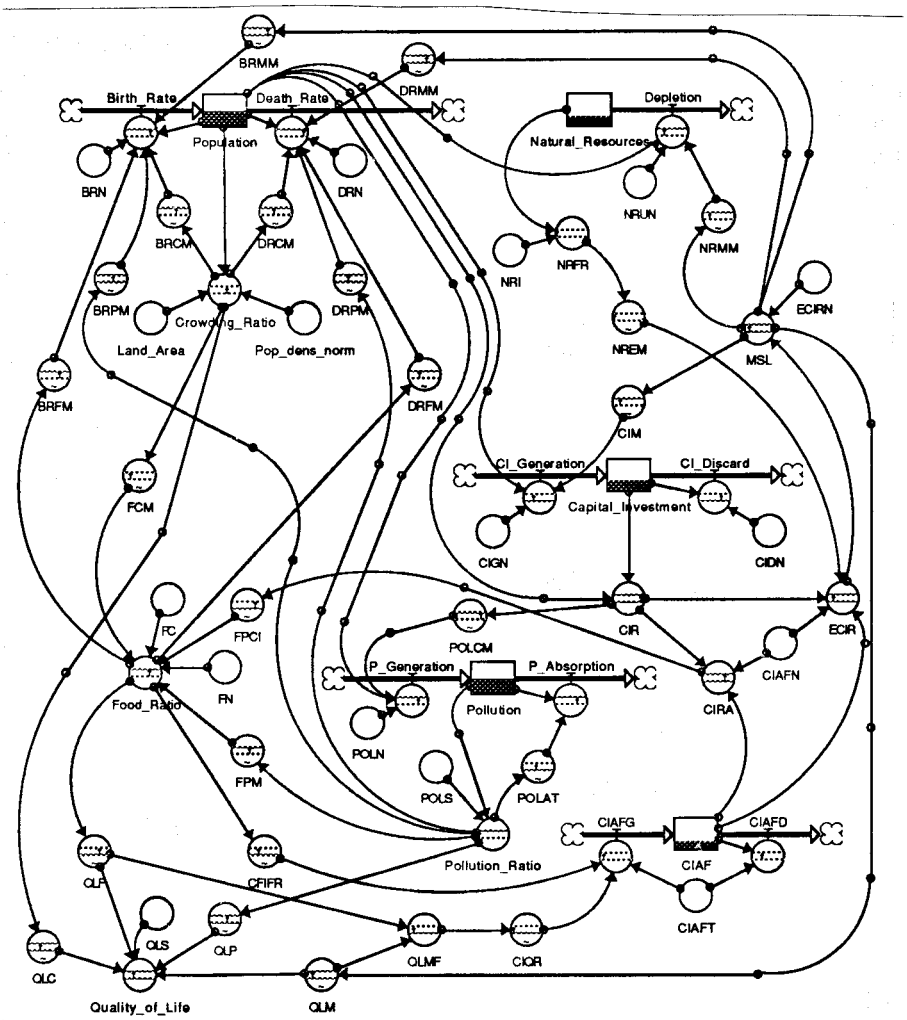

Figure 11.15. Structure diagram of Forrester's world model

As usual, I captured the model at the end of the simulation. Consequently, those converters that do not contain a needle gauge are constants. Forrester used the following values for his constants:

$$BRN = 0.04 \quad (\textit{normal birth rate}) \tag{11.56a}$$

$$CIAFN = 0.3 \quad (\textit{CIAF normalization}) \tag{11.56b}$$

$$CIAFT = 15.0 \quad (\textit{CIAF time constant}) \tag{11.56c}$$

$$CIDN = 0.025 \quad (\textit{normal capital discard}) \tag{11.56d}$$

$$CIGN = 0.05 \ \ (\textit{normal capital generation}) \tag{11.56e}$$
$$DRN = 0.028 \ \ (\textit{normal death rate}) \tag{11.56f}$$
$$ECIRN = 1.0 \ \ (\textit{capital normalization}) \tag{11.56g}$$
$$FC = 1.0 \ \ (\textit{food coefficient}) \tag{11.56h}$$
$$FN = 1.0 \ \ (\textit{food normalization}) \tag{11.56i}$$
$$Land\_Area = 1.35 \cdot 10^8 \ \ (\textit{area of arable land}) \tag{11.56j}$$
$$NRI = 9.0 \cdot 10^{11} \ \ (\textit{initial natural resources}) \tag{11.56k}$$
$$NRUN = 1.0 \ \ (\textit{normal resource utilization}) \tag{11.56l}$$
$$POLN = 1.0 \ \ (\textit{normal pollution}) \tag{11.56m}$$
$$POLS = 3.5999 \cdot 10^9 \ \ (\textit{standard pollution}) \tag{11.56n}$$
$$Pop\_dens\_norm = 26.5 \ \ (\textit{normal population density}) \tag{11.56o}$$
$$QLS = 1.0 \ \ (\textit{standard quality of life}) \tag{11.56p}$$

Converters with a needle gauge but without a tilda denote algebraic relations. These were given as follows:

$$CIR = \frac{Capital\_Investment}{Population} \tag{11.57a}$$
$$CIRA = CIR \cdot \frac{CIAF}{CIAFN} \tag{11.57b}$$
$$Crowding\_Ratio = \frac{Population}{Land\_Area \cdot Pop\_dens\_norm} \tag{11.57c}$$
$$ECIR = NREM \cdot CIR \cdot \frac{1.0 - CIAF}{1.0 - CIAFN} \tag{11.57d}$$
$$Food\_Ratio = FPCI \cdot FCM \cdot FPM \cdot \frac{FC}{FN} \tag{11.57e}$$
$$MSL = \frac{ECIR}{ECIRN} \tag{11.57f}$$
$$NRFR = \frac{Natural\_Resources}{NRI} \tag{11.57g}$$
$$Pollution\_Ratio = \frac{Pollution}{POLS} \tag{11.57h}$$
$$QLMF = \frac{QLM}{QLF} \tag{11.57i}$$
$$Quality\_of\_Life = QLS \cdot QLC \cdot QLF \cdot QLM \cdot QLP \tag{11.57j}$$

where *CIR* denotes the capital investment ratio, *ECIR* stands for the effective capital investment ratio, *MSL* is the material standard of living, and *NRFR* is the fraction of the non–recoverable natural resources remaining. Most of the equations are self–explanatory. A

detailed rationale for these equations can be found in Forrester's book.

The converters with a needle gauge and a tilda are tabular functions of their respective inputs. The following tables list the tabular functions:

Table 11.3a Tabular function that depends on *CIR*

	POLCM
CIR	*P_Generation*
0.0	0.05
1.0	1.00
2.0	3.00
3.0	5.40
4.0	7.40
5.0	8.00

Table 11.3b Tabular function that depends on *CIRA*

	FPCI
CIRA	*Food_Ratio*
0.0	0.50
1.0	1.00
2.0	1.40
3.0	1.70
4.0	1.90
5.0	2.05
6.0	2.20

Table 11.3c Tabular functions that depend on *Crowding_Ratio*

	BRCM	**DRCM**	**FCM**	**QLC**
Crowd_Rat	*Birth_Rate*	*Death_Rate*	*Food_Ratio*	*Qual_Life*
0.0	1.05	0.9	2.4	2.00
0.5				1.30
1.0	1.00	1.0	1.0	1.00
1.5				0.75
2.0	0.90	1.2	0.6	0.55
2.5				0.45
3.0	0.70	1.5	0.4	0.38
3.5				0.30
4.0	0.60	1.9	0.3	0.25
4.5				0.22
5.0	0.55	3.0	0.2	0.20

Table 11.3d Tabular functions that depend on *Food_Ratio*

	BRFM	**DRFM**	**CFIFR**	**QLF**
Food_Ratio	*Birth_Rate*	*Death_Rate*	*CIAFG*	*Qual_Life*
0.00	0.0	30.0	1.00	0.0
0.25		3.0		
0.50		2.0	0.60	
0.75		1.4		
1.00	1.0	1.0	0.30	1.0
1.25		0.7		
1.50		0.6	0.15	
1.75		0.5		
2.00	1.6	0.5	0.10	1.8
3.00	1.9			2.4
4.00	2.0			2.7

Table 11.3e Tabular functions that depend on *MSL*

	BRMM	**CIM**	**DRMM**	**NRMM**	**QLM**
MSL	*Birth_Rat*	*CI_Gen*	*Death_Rat*	*Depletion*	*Qual_Lif*
0.0	1.20	0.1	3.00	0.00	0.2
0.5			1.80		
1.0	1.00	1.0	1.00	1.00	1.0
1.5			0.80		
2.0	0.85	1.8	0.70	1.80	1.7
2.5			0.60		
3.0	0.75	2.4	0.53	2.40	2.3
3.5			0.50		
4.0	0.70	2.8	0.50	2.90	2.7
4.5			0.50		
5.0	0.70	3.0	0.50	3.30	2.9
6.0				3.60	
7.0				3.80	
8.0				3.90	
9.0				3.95	
10.0				4.00	

Table 11.3f Tabular function that depends on *NRFR*

	NREM
NRFR	*ECIR*
0.00	0.00
0.25	0.15
0.50	0.50
0.75	0.85
1.00	1.00

Table 11.3g Tabular functions that depend on *Pollution_Ratio*

	BRPM	**DRPM**	**FPM**	**Polat**	**QLP**
Poll_Rat	*Birth_Rat*	*Death_Rat*	*Food_Rat*	*P_Absorp*	*Qual_Lif*
0.0	1.02	0.92	1.02	0.6	1.04
10.0	0.90	1.30	0.90	2.5	0.85
20.0	0.70	2.00	0.65	5.0	0.60
30.0	0.40	3.20	0.35	8.0	0.30
40.0	0.25	4.80	0.20	11.5	0.15
50.0	0.15	6.80	0.10	15.5	0.05
60.0	0.10	9.20	0.05	20.0	0.02

Table 11.3h Tabular function that depends on *QLMF*

	CIQR
QLMF	*CIAFG*
0.0	0.7
0.5	0.8
1.0	1.0
1.5	1.5
2.0	2.0

Some of the functions that depend on the same input variable have values specified at different sampling points, and some (more scary!) have values specified over inconsistent domains. However, the program runs without error, and thus, let us not be discouraged by such minor details.

Finally, I need to write down the equations for the rates:

$$Birth\_Rate = Population \cdot BRN \cdot BRCM \cdot BRFM \cdot BRMM \cdot BRPM \tag{11.58a}$$

$$CIAFD = \frac{CIAF}{CIAFT} \tag{11.58b}$$

$$CIAFG = \frac{CFIFR \cdot CIQR}{CIAFT} \tag{11.58c}$$

$$CI\_Discard = CIDN \cdot Capital\_Investment \tag{11.58d}$$

$$CI\_Generation = CIGN \cdot CIM \cdot Population \tag{11.58e}$$

$$Death\_Rate = Population \cdot DRN \cdot DRCM \cdot DRFM \cdot DRMM \cdot DRPM \tag{11.58f}$$

$$Depletion = Population \cdot NRUN \cdot NRMM \tag{11.58g}$$

$$P\_Absorption = \frac{Pollution}{POLAT} \tag{11.58h}$$

$$P\_Generation = Population \cdot POLN \cdot POLCM \tag{11.58i}$$

This completes the description of the model. Simulation results are shown in Fig.11.16:

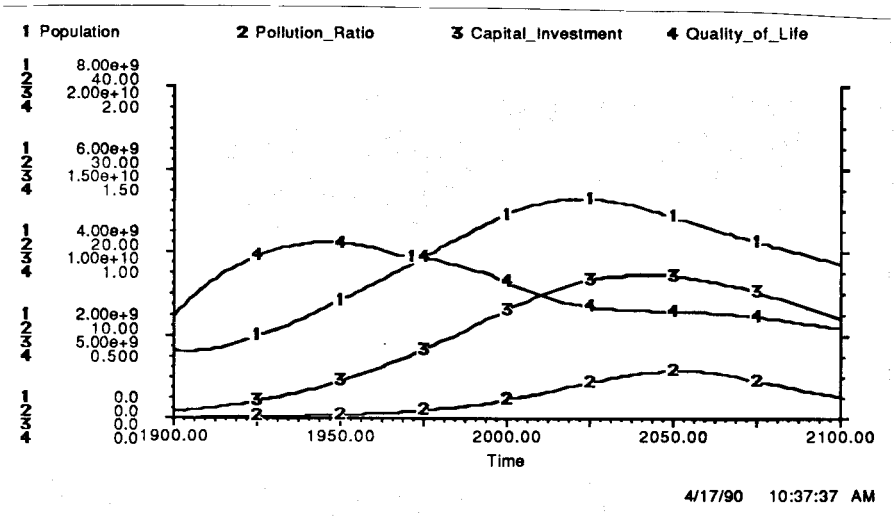

Figure 11.16. Simulation results for Forrester's world model

These results agree with those that were presented in Forrester's book. Notice that I silently converted the former difference equations of Forrester's original text into a set of differential equations. In my simulation, I used a step size of $\Delta t = 0.125$ instead of $\Delta t = 1.0$ as suggested by Forrester in order to make the graphs look smoother.

Forrester drew quite a bit of heat for his publication, from politicians since they didn't *want* to hear what he had to say, and from his colleagues since they considered his methods not sufficiently "solid" in scientific terms (and maybe since they were a little jealous of his unquestionable success). World modeling remained a fashionable topic for a number of years. Several other authors published their newest "findings" which I read with unbroken fascination and growing frustration. In order to avoid the criticisms and mockeries of their colleagues, all the later authors kept the details of their models for themselves, and published only their "results". Forrester was maybe a little naïve, but he was at least honest. He played with open cards, and for this, he has my full respect.

What were these dramatic revelations that Forrester disclosed in his book? He discovered that a physical system with finite energy cannot exhibit a behavior in which any variable grows to infinity.

The population will have to stagnate, and so will the GNP. As a control engineer, I don't need a simulation model to determine that this is in fact a truism. There cannot be a question about the fact that we shall *not* see an annual increase of the GNP of 3% or whatever for an infinite period of time. However, what was really new (and frightening) about Forrester's model was the fact that the stagnation is not just an abstract concept, something that might happen to our descendants in the year 5000 A.D., but that it will happen real soon, probably within the next 50 to 200 years. Our children may still see the day when recycling will no longer be an alibi exercise, but will become a bitter necessity, and when fines for causing environmental damage will no longer merely serve the purpose of ensuring the re–election of some politicians, but where the fines will be so stiff that it will be more economical for potential offenders to avoid causing the damage in the first place. I am not talking here about accidents as they can and will always happen, but about the reckless and purposeful contamination of our scarce and irreplaceable living space.

11.11 Model Validation

Since we have moved away quite a bit from the rigid meta–laws of the hard physical sciences that we started out with, it is now time to talk a little more about the process of *model validation.* How do we ensure that our models reflect reality? How do we verify, even if a given model reflects well the already observed behavior of the real system, that model predictions of future behavior do reflect the true future of the real system under investigation?

It is still too early to address this problem to its full extent. I shall do so in the second volume of this textbook when I also discuss the verification of simulation results. But at least, I wish to point out now some of the pitfalls of System Dynamics models in particular.

One of the real problems with System Dynamics models is the fact that the step of model validation is *separated* from the process of trajectory behavior generation. It is all too easy to forget the implicit (and explicit) assumptions behind a model once it has been properly "debugged" (i.e., it no longer produces any error messages!), and believe in simulation results just because they look elegant and maybe plausible. Certainly, Forrester succumbed to this temptation

in the conclusions of his text on *World Dynamics*, and we all do at times.

One remedy that comes immediately to mind is the following: never extrapolate any variables of a semi–physical model (such as a System Dynamics model) far beyond the range of observed values. It is absurd to believe that we can predict the reaction of our planet to a pollution value which is 100 times higher than any value that we have ever physically observed.

In practice, I suggest that the STELLA designers add a feature to their software which allows the users to attach "demons" to their models which can watch over the integrity of inherent model assumptions during the execution of simulation runs. The simplest form of such a demon is a threshold indicator which can be attached to any variable in the model. For example, when drawing a level variable denoting a population, we should be able to attach a demon to this variable which watches out that the population never decreases to a negative value. If this should ever happen, the simulation should immediately stop with an error message.

As can be noticed, a "demon" is nothing but an implementation of our well known termination conditions (called *termt* in ACSL). I suggest to implement demons as a new icon of the structure diagram. Demons have one or several inputs, but no outputs. Connectors can terminate in demons, but they can never emanate from demons. When we double–click on a demon, a window pops up which enables us to formulate a termination condition in which all the variables which are connected to the demon must be utilized. If we don't wish to clutter up our structure diagram with the demons, we can simply create ghosts of your "demonized" variables, and connect the demons to the ghosts in separate structure diagrams.

The second remedy that comes to mind is *sensitivity analysis*. The high "precision" (in terms of the number of displayed digits) that simulation results usually provide may be deceiving. It is quite obvious that parameter values are naturally associated with tolerance bands about a nominal value. Even in electrical circuitry, we know the value of a resistor only with an accuracy of either 2%, 5%, or 10%, depending on the price that we are willing to pay for the component. Consequently, it makes a lot of sense to investigate the sensitivity of the simulation results to parameter variations. If the sensitivity is large, we know that we must be extremely cautious in drawing conclusions about the system behavior. On the other hand, if the sensitivity is small, we can trust that our model exhibits

a behavior that resembles the true behavior of the real system. If you are interested in learning more about how this can be accomplished, solve homework problems hw(H11.3) and hw(H11.4). I shall return to this technique in more detail in the second volume of this textbook.

11.12 Summary

In this chapter, we have introduced a new modeling methodology, called System Dynamics, which enabled us to create rather quickly semi–formal models of "soft" systems, i.e., systems as they are found in the biological and social sciences, as well as in economy and business administration. A new tool, STELLA, was also introduced, a tool that has been specifically designed to facilitate the creation of System Dynamics models.

References

[11.1] François E. Cellier (1986), "Enhanced Run–Time Experiments for Continuous System Simulation Languages", *Proceedings SCS MultiConference on Languages for Continuous System Simulation*, (F.E. Cellier, ed.), San Diego, CA, pp. 78–83.

[11.2] François E. Cellier, and C. Magnus Rimvall (1989), "Matrix Environments for Continuous System Modeling and Simulation", *Simulation*, **52**(4), pp. 141–149.

[11.3] R. G. Coyle (1977), *Management System Dynamics*, John Wiley & Sons, London, U.K.

[11.4] Andreas Fischlin, and Werner Baltensweiler (1979), "Systems Analysis of the Larch Bud Moth System. Part 1: The Larch — Larch Bud Moth Relationship", Mitteilungen der Schweizerischen Entomologischen Gesellschaft, **52**, pp. 273–289.

[11.5] Jay W. Forrester (1964), *Industrial Dynamics*, MIT Press, Cambridge, MA.

[11.6] Jay W. Forrester (1968), *Principles of Systems*, Wright–Allen Press, Cambridge, MA.

[11.7] Jay W. Forrester (1971), *World Dynamics*, Wright–Allen Press, Cambridge, MA.

[11.8] Michael E. Gilpin (1979), "Spiral Chaos in a Predator–Prey Model", *The American Naturalist*, **113**, pp. 306–308.

[11.9] William L. Heyward, and James W. Curran (1988), "The Epidemiology of AIDS in the U.S.", *Scientific American*, October issue, pp. 72–81.

[11.10] James M. Hyman (1990), "Modeling The AIDS Epidemiology in the U.S.", private communication, Los Alamos National Laboratory, Los Alamos, NM.

[11.11] A. G. Ivakhnenko, and Valentin G. Lapa (1967), *Cybernetics and Forecasting Techniques*, American Elsevier Publishing, Series in *Modern Analytic and Computational Methods in Science and Mathematics*, **8**.

[11.12] Yumi Iwasaki, and Herbert A. Simon (1986), "Causality in Device Behavior", *Artificial Intelligence*, **29**, pp. 3–32.

[11.13] Jonathan M. Mann, James Chin, Peter Piot, and Thomas Quinn (1988), "The International Epidemiology of AIDS", *Scientific American*, October issue, pp. 82–89.

[11.14] George P. Richardson, and Alexander L. Pugh (1981), *Introduction to System Dynamics Modeling with DYNAMO*, MIT Press, Wright–Allen Series in System Dynamics, Cambridge, MA.

[11.15] Barry Richmond, Steve Peterson, and Peter Vescuso (1987), *An Academic User's Guide to STELLA*, High Performance Systems, Inc., 13 Dartmouth College Highway, Lyme, N.H. 03768.

Bibliography

[B11.1] Jean D. Lebel (1982), "System Dynamics", in: *Progress in Modelling and Simulation*, (F.E. Cellier, ed.), Academic Press, London, U.K., pp. 119–158.

Homework Problems

[H11.1] Influenza Model

Modify the influenza model given in this chapter in the following way:

$$Contacts\_Wk = U(5.0, 25.0) \qquad (H11.1a)$$

$$Time\_to\_Breakdown = N(4.0, 1.0) \qquad (H11.1b)$$

$$Sickness\_Duration = N(2.0, 1.0) \qquad (H11.1c)$$

$$Immune\_Period = N(26.0, 8.0) \qquad (H11.1d)$$

where $U(min, max)$ denotes a uniform distribution with a minimum value of min and a maximum value of max, and $N(\mu, \sigma)$ denotes a normal distribution with a mean value of μ and a standard deviation of σ. Repeat the simulation, and compare the results.

In a second simulation experiment, enhance the mean value of the immune period. Find the smallest mean value of the immune period necessary to ensure the natural extinction of the bacteria stem.

[H11.2] World Model

Jay Forrester determined that the stagnation in his world model was caused by the exhaustion of the non–recoverable natural resources. As a remedy, he recommended to reduce the normal utilization of natural resources (NRUN) to 25% of its former value in the year 1970.

Implement this change in STELLA using the STEP function. Simulate the modified system and check the results. Forrester discovered that the removal of this energy constraint just unchained another one. Now, the pollution explodes, and the population suffers a major breakdown due to an environmental disaster. He suggested to reduce the normal pollution also to 25% of its former value in the year 1970.

Implement also this change in STELLA using the STEP function. Forrester liked the resulting curve much better. Extend the simulation period from the year 2100 to the year 2500, and compare the new curve with the original curve. Interpret the results.

I discovered that the energy constraint on the natural resources is very beneficial to a smooth transition from the growth phase to the stagnation phase. As a control engineer, I know that, if I wish to reduce oscillations in a feedback control system, I should *reduce* the open–loop gain of the system, and not *enhance* it. Thus, since Forrester's model suggests that we have already exceeded the steady–state value of the system, the conservation of non–recoverable natural resources may be more detrimental than beneficial, since ultimately, we shall have to learn to live without them anyway. I once "optimized" the behavior of the world model in terms of a smooth transition to the stagnation phase. The optimization suggested that I should pump several thousand barrels of oil *into* the ground every day, i.e., that I should not *save* the non–recoverable natural resources, but *spend* them as quickly as I possibly can. An early shortage of these resources will ensure that our system exhibits as little overshoot behavior as possible.

[H11.3] Sensitivity in the Large

Reimplement Forrester's world model in ACSL. Execute the simulation under control of either MATLAB or CTRL–C. Assume that the four parameters *CIAFT*, *FC*, *NRUN*, and *POLN* are inaccurately known. Assume

that the given values have a tolerance band of ±25% associated with them. Rerun the simulation 16 times for all worst case combinations of these parameters. Import into CTRL–C or MATLAB the population variable from the ACSL model, and store all resulting populations together in a matrix. Use MATLAB's (CTRL–C's) MAX and MIN functions to compute upper and lower bounds, and plot the upper and lower envelopes of the population together on one graph as functions of time. What do you conclude?

[H11.4] Replication and Batch

This time, assume that the four parameters *CIAFT*, *FC*, *NRUN*, and *POLN* are uniformly distributed stochastic variables in the range *nominal value* ±25%. Perform 100 simulation runs with different seed values for the random number generators. Import from ACSL the population trajectories, and compute again upper and lower envelopes. What do you conclude?

[H11.5] Balancing Your Checkbook

A traveling sales person makes an average net income of $2000 a month, more precisely, his or her income is $N(2000, 200)$, i.e., the income is normally distributed with a mean value of $2000, and a standard deviation of $200. S/he likes to spend 90% of her or his average income. S/he computes his or her average income as a moving average of the real income over the past six months.

Since STELLA does not provide us with a moving average function, we need to construct this function the hard way. Make a ghost of the sales person's income, and delay it six times by one month in a separate structure diagram. Then add the six incomes up, and divide by six. This is the desired moving average. Make a ghost of the moving average, and copy it back into the main diagram.

Of course, a cash constraint exists. The sales person will not spend the desired amount of money unless s/he has sufficient cash in the bank. S/he considers $4000 in the bank sufficiently safe to spend what s/he likes to spend. His or her real consumption is the product of the desired consumption and the cash constraint. The cash constraint is 1.0 if s/he has $4000 or more in the bank, otherwise, it is linear in her or his current savings.

At the beginning of the simulation, the sales person has exactly $4000 in the bank. Four months into the first year, a competitor comes out with a new product that reduces the mean value of the poor sales person's income by $500. The standard deviation remains the same. Simulate the system over 3 *years*. Plot on one graph the monthly net income, the cash balance, and the monthly consumption of our Schlehmihl.

Research

[R11.1] The Epidemiology of AIDS

Study the October 1988 issue of *Scientific American*, a special issue devoted to the HIV infection, and in particular, the two articles on the epidemiology of AIDS by Heyward and Curran [11.9] describing the epidemiology of the disease in the U.S., and by Mann *et al.* [11.13] describing the international epidemiology of the disease. Scan the literature for further information to obtain as solid statistical material as possible to base your modeling efforts upon. A good source of data are the Proceedings of the Annual International Conference on AIDS. Develop a System Dynamics model (a modification of our simple influenza model) that describes the spreading of the disease, and which reflects the observed data well. As usual with these types of studies, the crux is with the data. The available data are often speculative, and are largely inconsistent. It would be fairly easy to determine an inductive model describing the spreading of the disease fairly accurately, but this doesn't help us much since we haven't observed a saturation period yet, and this is exactly what our model is supposed to predict. Thus, we need a semi–physical model.

One of the most interesting facts about the epidemiology of AIDS is that the reported AIDS population did not grow *exponentially* during the early stages of the disease as would be expected, but instead, it grew *polynomially* with the third power of the time t. A very good inductive model of the reported AIDS cases in the U.S. between 1980 and 1988 can be described through the formula [11.10]:

$$A(t) = 175 \cdot (t - 1981.2)^3 + 340 \qquad (R11.1a)$$

If we try to retrofit a physical model to the observed behavior, we need a differential equation for $A(t)$. From eq(R11.1a), we can find immediately by differentiation:

$$\dot{A}(t) = 525 \cdot (t - 1981.2)^2 \qquad (R11.1b)$$

which can be rewritten as:

$$\dot{A}(t) = \frac{3 \cdot (A(t) - 340)}{(t - 1981.2)} \qquad (R11.1c)$$

Consequently, in order to obtain a polynomial growth, our model must contain some sort of $\frac{1}{t}$ factor. Such a $\frac{1}{t}$ factor makes sense for a disease which loses its virulence through natural mutation or diffusion, or which leads to a growing body of immunized and therefore non–susceptible hosts. Unfortunately, neither of these two assumptions is plausible in the case of the AIDS disease. However, any decent semi–physical model of the epidemiology of AIDS will have to come up with a plausible explanation of

where this $\frac{1}{t}$ factor comes from, a hypothesis which should preferably be verifiable.

Come up with a consistent System Dynamics model describing the epidemiology of AIDS in terms of the susceptible population $S(t)$, the infected population $I(t)$, and the sick population $A(t)$, possibly compartmentalized into various risk groups. It may also be necessary to divide the infected population into three separate subgroups denoting the group of freshly infected individuals (who are highly contagious, since no antibody has been developed yet), the group of symptom–free infected individuals (not very contagious, since the antibody constantly destroys the free retrovirus in the blood stream), and the group with early symptoms (highly contagious since the antibody is about to lose the battle).

Associate tolerance bands with the parameters of your model, and perform a "sensitivity analysis in the large" (as described by the author in two previous articles [11.1,11.2]). From the sensitivity analysis, determine which will be the maximum and minimum values of reported AIDS infections in the years 1992, 1995, and 2000. When will the epidemic exhibit its peak value? What are the best and worst percentages of the infected population at that time? Will humanity survive the viral assault? How good is the confidence of your predictions?