To Main Proceedings Document

Superimposing Dynamic Behavior on Causal Loop Diagrams of System Dynamic Models

Magne Myrtveit

Founder and Senior Vice President
Powersim AS, Hellandsneset, N-5936 Manger, Norway
Telephone: +47 88 02 34 34 – Facsimile: +47 56 37 35 00
E-mail: magne.myrtveit@powersim.no
Web page: http://www.powersim.no

Mohamed Saleh

Powersim AS, Sandbrugaten 5-7,
Postboks 3961, Dreggen, N-5835 Bergen, Norway
Telephone: +47 55 60 65 12 – Facsimile: +47 55 60 65 01
E-mail: mohamed.saleh@powersim.no
Web page: http://www.powersim.no

1 Abstract

The importance of feedback analysis is stressed by SD the community, and the subject is taught at most SD classes. In the practice of business simulations, the identification and study of feedback loops is not so ubiquitous as it deserves. The weak technical support for dynamic analysis of model structures makes it difficult to apply feedback analysis on large models. In addition, the understanding of the dynamic nature of models seems to be quite rudimentary in many areas.

Powersim has since 1995 developed new mathematical and technological solutions for analyzing and visualizing dynamic aspects of SD models. This paper concentrates on some technological results of this research.

Our approach is based on modes of behavior, loops and links (influences) of a model, as well as extensions to the causal loop diagram (CLD) representation of models. The overall behavior mode of a model is determined by the sum of all the elementary modes at any time. Each mode describes growth, decay or linear development, and it can in addition have an element of oscillation. By sorting the elementary modes according to strength, the dominating behavior of the model is quickly identified. The paper also describes how loops of a system can be identified and sorted according to contribution to a given behavior mode. Diagrammatic display of contribution to behavior modes (and loops) is performed using line styles and colors of links in a CLD. This creates a visual link between the structures of a model and their relative contribution to selected loops as well as the overall behavior of the model. This special way of dynamic coloring creates a “heat map” that identifies the most active structural parts of a model at any time.

2 Introduction

Powersim has since 1995 dedicated resources to work related to improving the mathematical and the technological support for analyzing dynamic systems. Initially, the focus was on identifying needs and pointing out directions for research and development. Since the beginning of 1999 the work has been coordinated with the University of Bergen, through a PhD program. The effort around improving the mathematical and the technological support for dynamic analysis and visualization has been carried out in parallel. This paper concentrates on technological results, whereas another paper at this conference (Saleh and Davidsen 2000) covers the underlying mathematics in detail.

3 The Challenge

System Dynamics Models are used to capture and understand important aspect of real or imaginary systems. One important reason for using models is that a model can be kept simpler than the real system it portrays. However, once a model grows bigger than a few variables, it starts to possess the same characteristics that make the real world difficult to understand. One set of difficult questions relate to model behavior:

- Prior to running the simulation: How will this model behave?
- After running the simulation: Why did this particular behavior occur?
- How can we change the model to improve its behavior (measured by certain objectives)?

And as if these questions were not enough, there is also a range of difficulties concerning the structure of large models:

- What does this model contain? Where are the boundaries? What is included? What is left out?
- Which structures (among all the details) are the most important for the model behavior?

This paper concentrates on the dynamic aspects of models, while Myrtveit (2000) puts the focus on the structural aspects.

4 Visualizing Structure and Behavior Separately

System dynamisists have developed various ways of helping us understand the structure and behavior of a model. The simplest solutions deal with model structure and model behavior separately. Time tables and time graphs are commonly used to display how variable values develop over time. Symbols representing model variables can be linked together in various ways to form Causal Loop Diagrams (CLD) or Accumulator Flow Diagrams (AFD) visualizing the structure of a model.

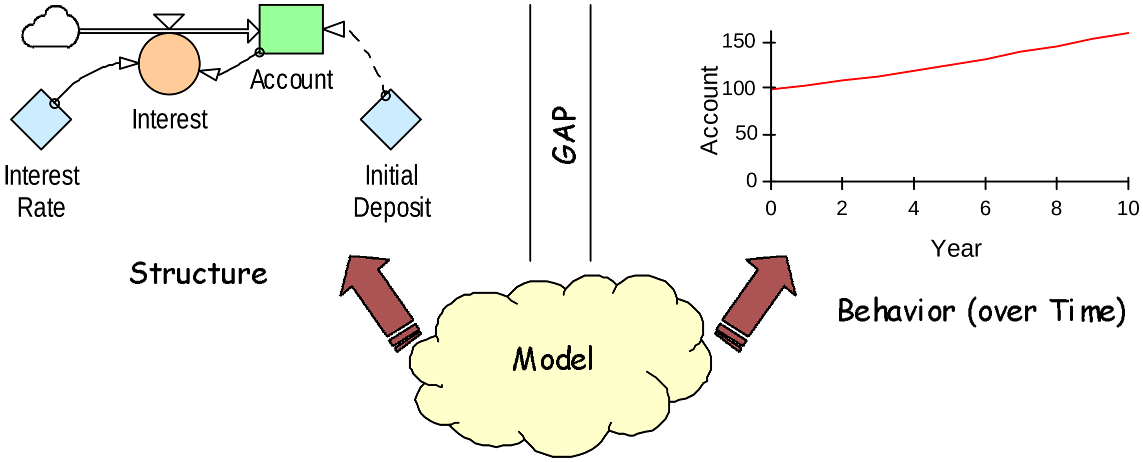


Figure 1: Structure and Behavior visualized separately

The links that are connecting up variable symbols of a Causal Loop Diagram can be traced to form closed paths, or feedback loops. Feedback loops can be categorized as balancing or reinforcing, based on the polarities of the links involved in each loop.

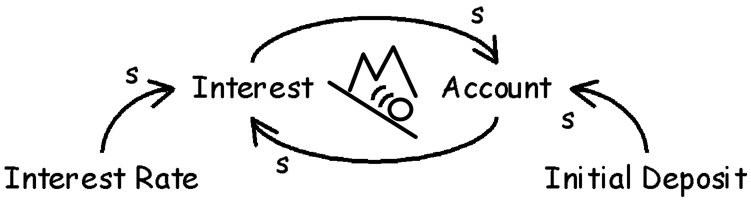


Figure 2: Causal Loop Diagrams show structure and behavioral implications together

In the above figure, the polarity of each link is visualized. The snowball symbol in the center indicates that the enclosing feedback loop is a reinforcing one.

The nature (balancing or reinforcing) of a loop will change if one of its links changes polarity (from negative to positive, or vise versa). But even when loop polarities stay the same throughout a simulation; their relative strengths normally vary over time in a non-linear model. Knowing the nature of loops is not enough to determine behavior. In Figure 2 there is only one loop, but all we can say, is that the behavior of this system will be diverging—either going up or going down. The signs of the *Interest Rate* and the *Initial Deposit*, will determine if the *Account* will grow upwards or downwards—two quite different scenarios for the owner of the account.

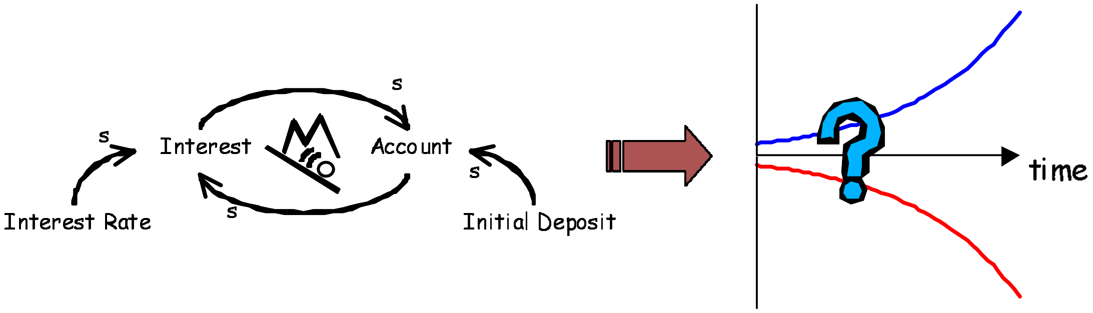


Figure 3: Loop polarities are not enough to predict behavior

From the above figure, it becomes clear that there is a distinction between behavior and behavior mode. In a single-loop system, it is trivial to determine the behavior mode, as either diverging or converging. But, even then, we need to know the initial system state and precise model equations in order to determine the actual behavior over time.

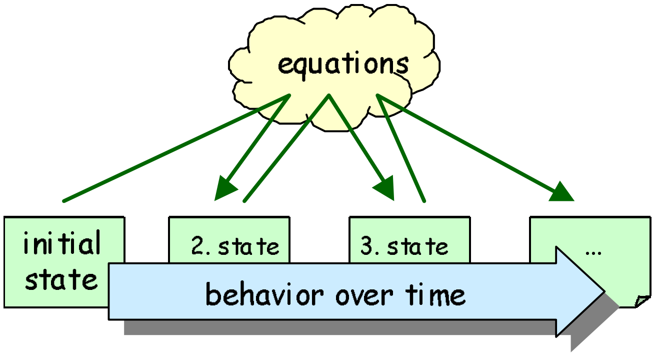


Figure 4: State + Equations → Behavior

The behavior mode of a simulation at any given time is determined by the strongest feedback loop(s). The overall pattern of behavior over time can be related to changing relative strengths of feedback loops. A system with one balancing and one reinforcing loop produces S-shaped development if the reinforcing loop dominates in the first phase, and the balancing loop dominates in the second phase. This is an example of an archetype. One way to define an archetype is to say that it is a given network of feedback loops, where the relative strengths of the loops change over time, causing a characteristic behavior. The structure, behavior and reasons for shifts in loop dominance are three important keys to learning from archetypes. The relationship between archetype and structure is not always easy to spot. The feedback loops that dominate during the various phases of a simulation can be part of a much larger network of links and loops, making the archetypal structure almost impossible to spot. It is also the case that the loops of an archetype can be modeled in many different ways.

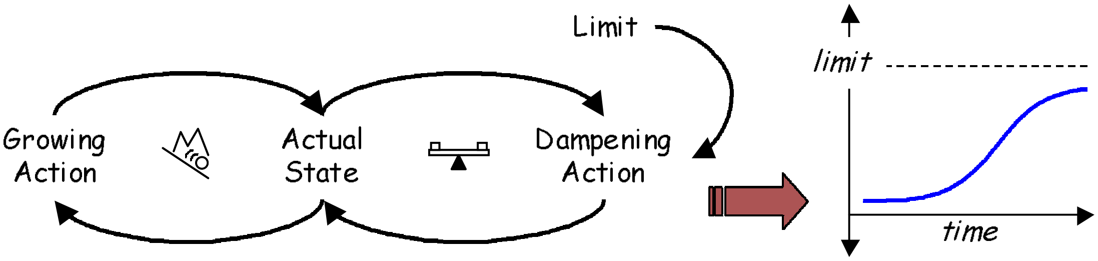


Figure 5: Archetypes relate behavior patterns to shifts in loop dominance

The above figure illustrates the “limits growth” archetype (Senge 1990). A model with this structure will among all its loops have two loops that will be dominant during different stages of the simulation. One loop is reinforcing, and it will be dominant in the first phase—the divergent phase. The other loop is balancing, and it will be dominant in the last phase—the converging phase.

When it comes to model structure, it is quite common to subdivide a model it into sectors or subsystems. This way the topology of the model can be made clearer. Each part can be studied and understood independently at a detailed level, while the overall relationships between the parts can be described at a higher level. This approach is related to object-oriented thinking, and provides a means of dealing with (structural) complexity through abstraction.

Feedback loops (and archetypes) do normally not stay within subsystem boundaries. On the contrary, feedback is to a large extent a phenomenon that takes place between subsystems. As a consequence, it is not feasible to construct subsystems around archetypes. It is also the case that, while subsystems highlight topological structure, they obscure feedback structure, and vice versa.

Within sectors or subsystems, models are typically described using accumulator flow diagrams. The advantage of this diagram type is that it has distinct symbols for representing the two fundamental building blocks of SD models, the levels (states) and the rates (flows). Accumulator flow diagrams are very well suited for visualizing (and building) the inner workings of a system or subsystem. The symbolic building blocks of variable symbols (levels and auxiliaries) and relationship symbols (links and flows) are too abstract, however, for capturing the conceptual structure of systems as composed from a hierarchy of subsystems (see Myrtveit 2000).

Accumulator flow diagrams do not lend themselves well to a feedback perspective. First of all, accumulator flow diagrams have several different kinds of symbols for variables and dependencies. In particular, flow symbols can confuse the direction of influence between a flow variable and a level. As an example, it is not particularly clear from the AFD view below that B influences A, and not vice versa.

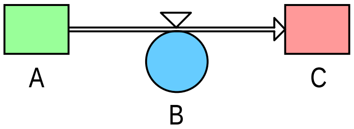


Figure 6: Flows do not represent direction of dependencies well

The use of snapshots (aliases, ghost variables) in AFD also contributes to hiding influences and feedback structures.

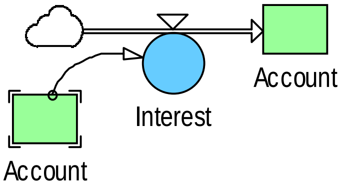


Figure 7: Snapshots obscure feedback view

The above discussion calls for multiple visualization of a model.

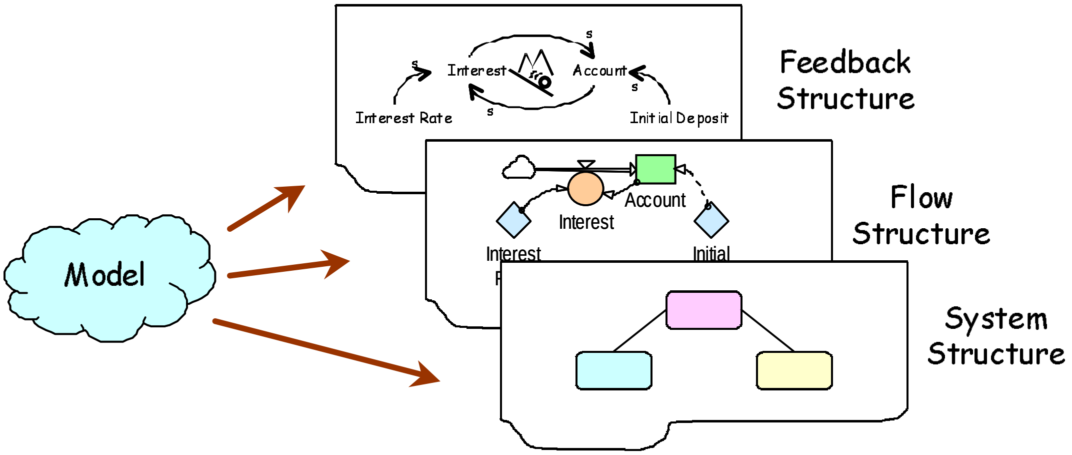


Figure 8: We need multiple views to capture different aspects of a model

Both subsystems and feedback loops are useful in coping with model complexity. Subsystems allow the modeler to work with larger building blocks, while the dominant feedback loops isolate the parts of the model that are responsible for its (modes of) behavior. Therefore, the SD technology should be developed in a direction where the modeler can maintain views of the hierarchical structure, the major feedback loops, as well as the underlying levels and flows. The remainder of this paper concentrates on the feedback view of a model, and how this view can be used to understand the relationship between model structure, state, and behavior.

5 Mathematical Foundation

During simulation, the behavior of a model can be compared to waves on the ocean. The wave pattern on the water surface is composed from many waves. Large waves can be caused by a storm in the ocean, other waves are reflections from the shore or caused by a ship passing by, and there can be ripples caused by a breeze sweeping over the surface. Each feedback loop of a model contributes to the behavior pattern of the model. Some loops contribute much, others less.

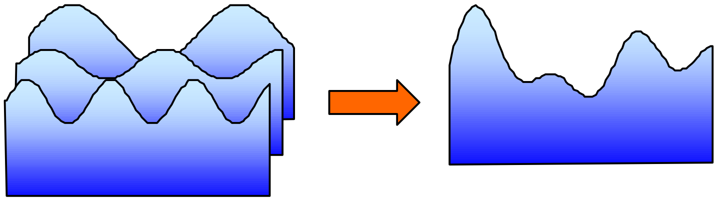


Figure 9: Elementary waves combined into one composite wave

The elementary modes of model behavior are not only the oscillating ones, like the ocean waves. In models we typically have amplification and dampening, growth and decay. This can cause even more interesting—and more complex—behavior patterns than those you can observe on water.

5.1 Elementary Modes of Behavior

For linear models eigenvalues are commonly used to describe the behavior modes of systems. In general, an eigenvalue is a complex number, where the real part is linked to growth and the imaginary part is linked to oscillation. A large real part represents rapid growth.

An eigenvalue represents an elementary mode of behavior of a model, much like an elementary wave on the ocean, as illustrated in Figure 9. A given eigenvalue λ is related to (an element of) the model behavior like this:

$elementarymode(\lambda) = Ce^{\lambda t}$; where C is a constant and t is time.

The unit of λ is 1/s (when time is measured in seconds). The doubling time of the elementary mode is defined like this:

$doublingtime(\lambda) = \ln 2 / \lambda.re$; where λ.re is the real part of λ

A negative doubling time means that the model is converging, i.e., that we have to go back in time to find the doubling point. The negative value of the doubling time defines the half-life time.

$halflifetime(\lambda) = -doublingtime(\lambda) = -\ln 2 / \lambda.re$

An elementary mode oscillates with the following frequency:

$frequency(\lambda) = \lambda.im / 2\pi$; where λ.im is the imaginary part of λ

5.2 Non-Linear versus Linear Models

In a linear model, the eigenvalues stay constant during the course of the simulation.

The mode of behavior for a state variable is defined as

$$\sum_{i=0}^{N-1} C_i e^{\lambda_i t}$$

Variables outside of loops only influence the constants C_i.

In the long run, the strongest eigenvalue will always become dominant in such models.

SD models are typically non-linear models. The loop gains of such models depend on the actual system states at any given time. In turns, this leads to changing eigenvalues over time. Thus, in a dynamic non-linear model, we may never reach the mode indicated by the eigenvalues at any given time.

For dynamic non-linear models, it is better to take and instantaneous view rather than a long-term view, and identify the behavior mode that contributes the most to the current development of the system. This can be done by investigating the relative effect of each eigenvalue on the current rate of change in model state. It turns out that it is not necessarily the largest eigenvalue that dominates the behavior in the short-term. (If the largest eigenvalue were given the chance to work over time without loosing its dominant position, that eigenvalue would eventually take over the dominance over the behavior.) Saleh and Davidsen (2000) explain how eigenvalues can be sorted by their effect on the current behavior.

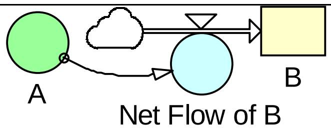
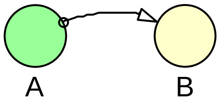
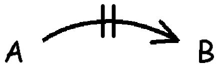
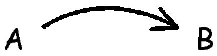
For non-linear system dynamics models, the imaginary part of an eigenvalue cannot be used as a reliable measure for oscillation frequency. The reason for this is that eigenvalues change over time due to the feedback structures of the model, causing oscillation frequency, growth and decline to change from time step to time step.

Since eigenvalues change over time for non-linear, dynamic models, the eigenvalues must be computed periodically, or in response to a request from the user.

5.3 Gain

At the lowest level, model relationships are defined as links between variables. The gain of a link from A to B is defined as:

$$gain(A,B) = \begin{cases} \partial B/\partial A & \text{; when B is not a level} \\ \partial(\partial B/\partial t)/\partial A & \text{; when B is a level} \end{cases}$$

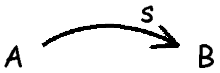
	When B is a level, the gain of a link from A to B is defined as the gain of the link from A to the net flow of B.	*When B is not a level, the gain of a link from A to B is defined as the partial derivative ∂B/∂A.*
AFD		
CLD		

Gains are not easily used in comparing strengths of different links, however, since the measurement unit of a gain depends on the measurement units of the variables at both ends of the link. Comparing the numerical values of two gains is like comparing applies to bananas. Gains are, however, important in computing some other measurements that can be used to compare and visualize the relative strengths of links, and their importance relative to a given mode of behavior.

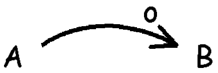
5.4 Polarity

The sign of the gain defines the polarity of a link, and is used to decorate the link symbols of a CLD with +/- (or s/o) indicating direction of influence.

$$polarity(A,B) = sign(gain(A,B))$$



Positive polarity



Negative polarity

5.5 Contribution

The sensitivity of a given elementary mode, or eigenvalue (λ) to a change in a given gain (g) is defined as:

$$sensitivity(\lambda, g) = \frac{\partial \lambda}{\partial g}$$

The elasticity of a given elementary mode (λ) to a given gain (g) is defined as the percentage change in mode due to a percentage change in gain, like this:

$$elasticity(\lambda, g) = \frac{\partial \lambda / \lambda}{\partial g / g} = \frac{\partial \lambda}{\partial g}\frac{g}{\lambda}$$

We define the contribution as the elasticity times lambda, like this:

$$contribution(\lambda, g) = g \frac{\partial \lambda}{\partial g}$$

The contribution of a link is a measure of how much the link contributes to a given elementary mode. The sum of all link elasticities of a model is one, and the sum of all link contributions is lambda.

Both elasticity and contribution can be explained using the electric current metaphor. In a closed circuit, the electric current *I* will be the same in all sections of the circuit. As an example, the currents of the circuit below are such that *I3 = I1+I2*.

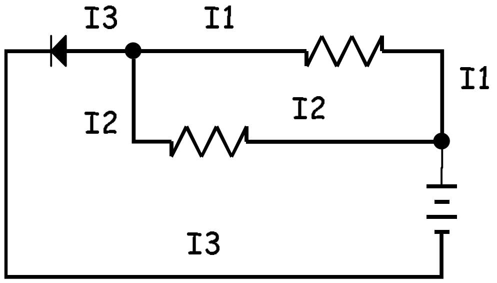


Figure 10: Elasticity and contribution can be explained using the electric current metaphor

The above diagram has two loops. The bottom, inner loop has a current equal to *I2*, while the outer loop has a current equal to *I1*. The sum of the loops carries the total current, *I3*.

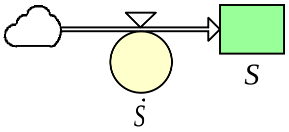
The contribution attribute of a link serves as a good indication of how important that link is in creating a given elementary mode (as defined by the associated eigenvalue).

Only links that are part of loops have contribution to the behavior modes. This corresponds to electric current, which exists only in closed circuits (loops).

5.6 Simplified Solutions for States and Flows

Any model can be simplified to consist of level variables (states) and their net flows. We use the following notation for net flows:

$$netflow(S) = \dot{S}$$
$$\dot{S} = \partial S / \partial t$$



By investigating the relationships between the states of a model, it is possible to identify all loops. (All feedback loops have at least one level. Otherwise we would get circular definitions, which cannot be resolved mathematically.) Each loop will contribute a given amount to every eigenvalue of the model. This means that we, for a given eigenvalue can sort the loops of the model according to contribution to the behavior mode related to that eigenvalue.

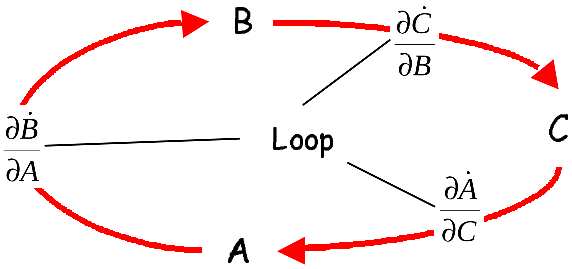
5.6.1 Links between states

If a level A influences another level B, we can imagine that there is a link from A to B. The gain of this link is defined as the change in B's flow due to a change in A. This can be expressed mathematically as $\partial \dot{B} / \partial A$, where $\dot{B}$ as a shorthand for B's net flow, $\partial B / \partial t$.

5.6.2 Loops through states

The gain of a loop is defined as the product of the gains of the individual links. As an example, if the levels A, B and C form a loop, the gain of that loop is defined as:

$$loop(A,B,C) = \frac{\partial \dot{B}}{\partial A}\frac{\partial \dot{C}}{\partial B}\frac{\partial \dot{A}}{\partial C}$$

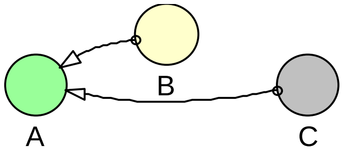

B
$\frac{\partial \dot{C}}{\partial B}$
$\frac{\partial \dot{B}}{\partial A}$
Loop
C
$\frac{\partial \dot{A}}{\partial C}$
A

For a system with N levels, there is a potential link between each level and every other level (including itself). This means that there is a maximum of N^2 links in a system (with only levels). Many of these links will have a gain of zero, and can be disregarded. The remaining links can be used to form all possible closed paths through the system. This will form the set of feedback loops. The strongest loop at any point in time, determines the mode of behavior for the system at that time.

The set of feedback loops can become very large, but many of the loops will normally have a relatively small gain. Kampmann (1996) describes a method for focusing on the independent set of loops, which is typically much less than the total number of loops in a model. This means that we can identify the set of dominant loops even for large models.

5.7 Computing gains of links and loops

Based on the above definitions, gains can be computed either through equations or by means of numerical methods. Let us look at an example where A is defined as 2B*C^2. The gain of the link from B to A is defined as $\partial\left(2BC^2\right)/\partial B$, which is the same as 2C^2. Correspondingly, the gain of the link from C to A becomes 4BC.



def A = 2B*C^2

gain(B,A) = 2C^2

gain(C,A) = 4BC

Figure 11: Example computation of link gains

The approach using equations (generated by symbolic derivation) to define gains is not always possible, or at least, very difficult sometimes. The reason for this is that it is not always easy to find a good representation of $\partial A/\partial B$. In such situations, we can use the fact that

$$\frac{\partial A}{\partial B} = \lim_{\Delta \to 0} \frac{A(B+\Delta) - A(B)}{\Delta}$$

This means that we can compute two values of A, one where B is equal to its current value, and another where B is increased a little. An even better approximation is to compute the gain as

$$\frac{A(B+\Delta)-A(B-\Delta)}{2\Delta}$$

The above approximation to differentiation is called finite difference approximation. The technique has two sources of error; the approximation error and the numerical round-off error. In order to reduce the approximation error, we can reduce the delta. This, however, will increase the round-off error, so there is a limit to how small delta can be. An implementation should consider using factorial base format for numbers. This format allows for operations on and representation of rational numbers without round-off error. (See Wayner 1988).

Note also that the finite difference approximation is easiest to perform on a *mini-model representation* of a system (See Scott 1999). Here we have only levels and rates. In the accumulator flow view, the mini-model representation gives a clear picture of states (levels) and changes (flows) of a model. In the causal loop view of a mini-model, we avoid problems computing gains of links in situations where there are multiple paths (through non-levels) from one variable to another. Below is an example:

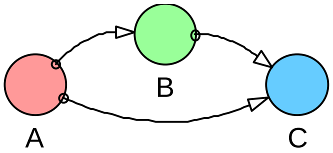


Figure 12: Example of model where some link gains cannot be computed directly through finite difference approximations

For the above model, we cannot compute the gain of the link from A to C using the formula directly:

$$\frac{C(A+\Delta)-C(A-\Delta)}{2\Delta}$$

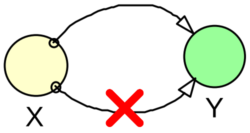
The reason is that there is also an influence from A to C that goes via B. An initial implementation should avoid the above problem by removing B from the causal loop view.

Let us define a new function, LINK(A,B), where A and B are expressions. The implementation of LINK can use numerical derivation always, or it can choose to evaluate an expression whenever symbolic derivation is easy to perform.

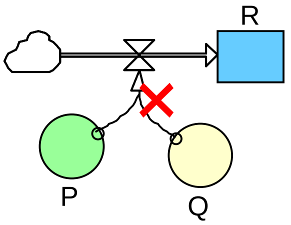
The new function can be used to define a variable representing the gain of each link. The gain variable for the link from A to B can be called A'B, and it will be defined as LINK(A,B). Note that if B is a level, the gain must be computed based on $\partial\dot{B}/\partial A$ instead of $\partial B/\partial A$.

In order for the LINK function to work, we need to ensure that the expression LINK(A,B) identifies one and only one link in the model. This puts some minor constraints on the way models are formulated:

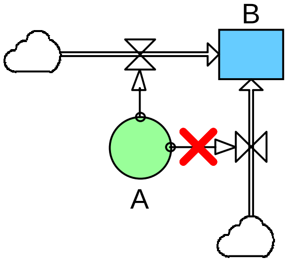
1. We cannot allow more then one link from a variable A to another variable B.



2. A flow (of a level) must be identified by one and only one variable. This means that flows cannot be composed from expressions involving more than one variable.



3. Because of the previous requirement, one variable cannot control two flows on the same level.



Note: The above rules assume scalar variables. For arrays, the requirements concern individual array elements.

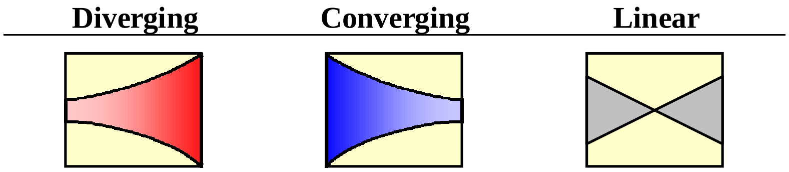
As with links, we can define loops as special variables in the model. A loop trough the variables A, B and C, can be defined using the link variables A'B * B'C * C'A, or by means of the LINK function, like this: LINK(A,B)*LINK(B,C)*LINK(C,A).

6 Visualization and User Interface

6.1 Elementary modes

The eigenvalues of the model are computed, sorted, and made available for the user to select from. The sorting of eigenvalues is not based on their numerical values, but on their relative contribution to the current development of the model. This can be done by stopping the effects of one eigenvalue at the time, and see how the development of the model is effected. The eigenvalue that has the greatest effect is defined as the dominant eigenvalue.

For a given eigenvalue, the elementary mode is described as either diverging, converging or linear. The following icons can be used as a direct representation of the current dominant mode:



The linear mode normally appears at the transition between diverging and converging mode, or vice versa.

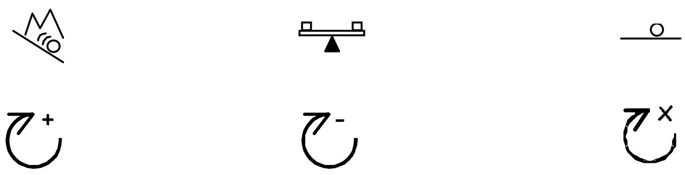
In a converging mode, the rate of change in system state will become smaller and smaller. This means that a convergent mode will be approaching equilibrium, i.e., a phase where the levels of the model do not change.

6.2 Loops

The CLD diagram can contain a vast number of loops, and there is no direct representation of a loop, other than the sequence of links that connect the variables of the loop. In general, it has no meaning to represent all the loops of a model. But the main loops can be important to identify.

Traditionally, modelers place loop symbols inside a CLD as placeholders for the most important loops. The same icons that apply for model behavior can be used for loops. Other commonly used symbols are shown below:

Diverging	Converging	Linear
R	B	L



It is not common to draw links from the link “variables” to the loop “variables”, as this can obscure the diagram.

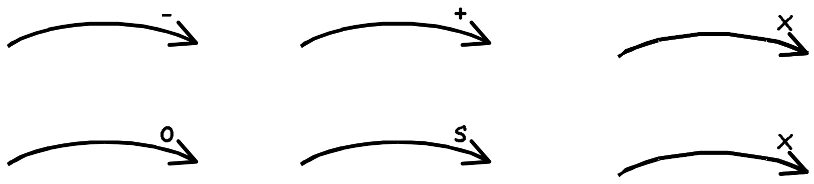
In a computerized diagram, the changes in polarity can be displayed dynamically during simulation. A software with possibilities for global analysis of the loops of a model during a simulation should allow the user to identify the dominating loops, and highlight the corresponding sections of the model. It should also be possible to use this information to create loop “variables” in the model.

6.3 Links

Link gains (and consequently also link polarities) are dynamic attributes of a model. When people put plus-es and minus-es in their diagrams that is because polarity tends not to shift during a simulation, but this is not always the case. Similarly, the polarity of a loop *can* change during a simulation.

The dynamically calculated *polarity* of a link can be used to place a + or - (s or o) symbol near the arrow head of the associated link symbol:

Negative Gain	Positive Gain	Neutral Gain



In addition, color and line width can be used to display *contribution* to the selected behavior. The user can make a global selection of elementary mode, and in addition a selection of zero or more loops. If no loops are selected, the contribution to the behavior of the entire model is visualized by the links. Otherwise, the contribution to the selected loops will be shown.

A color of red is used for reinforcing (diverging) behavior, and blue is used for balancing (converging) behavior. Grey is used for neutral contribution to the selected mode.

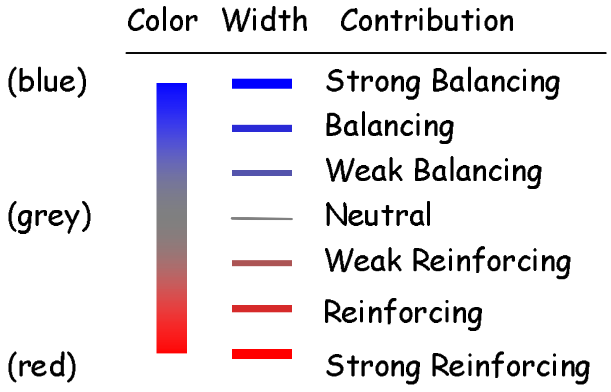


Figure 13: Color-coding for link contribution

6.4 Heat-map and other visualizations

The coloring and line width of links in a CLD diagram allows for rapid and direct identification of the active parts of a model, i.e., the parts that contribute the most to a given behavior mode. The essential structures of a model can be isolated like this:

1. Run the model and determine which loops are dominant at different times. (Ideally, the loop identification takes place every time step, but a larger analysis time step can be picked by the user).
2. Display all variables and links that are part of a loop that has been dominant in one of the phases identified above.

By simulating the model again, and animating in the diagram how each link contributes to the dominant loop, we can get an interesting picture of the changing dynamic forces of the model.

The identification of the structures contributing to a given behavior is important in understanding a model, and in knowing where to make changes in order to alter model behavior.

It is possible to divide a simulation run into phases, where each phase is dominated by a single loop. This set captures the most active structure of the model.

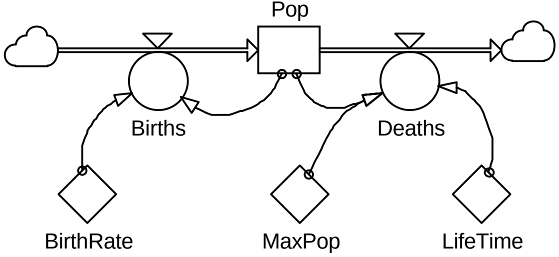
As an idea for future research, it would be interesting to see to what extent it is possible to identify archetypes from the transitions between dominant loops during a simulation.

An implementation of the techniques described here, should include a set of indicators for behavior, including eigenvalues and qualitative indicators such diverging or converging mode.

7 An Example

Let us take a look at a population model with one level and two flows, representing births and deaths. The growth in the population is controlled by a birth rate, an average lifetime, and a capacity (maximum population) of the environment where the population lives. The model looks like this, in AFD view.

Diagram **Equations (Powersim Constructor 2.51)**



init	Pop = 2 {Individuals}
flow	Pop = -dt*Deaths +dt*Births
aux	Births = Pop*BirthRate
aux	Deaths = Pop/(LifeTime*(MaxPop-Pop)/MaxPop)
const	BirthRate = 0.2 {1/Year}
const	LifeTime = 20 {Years}
const	MaxPop = 100 {Individuals}

Figure 14: Accumulator flow diagram and equations

The model has two simple feedback loops, as displayed in the CLD below.

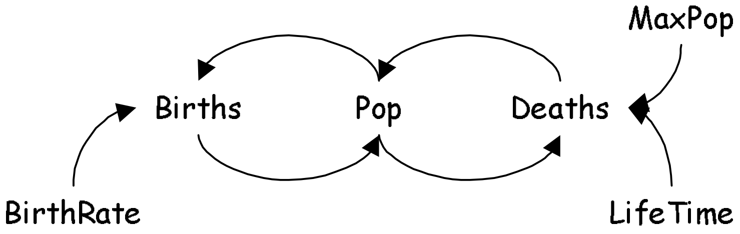


Figure 15:Causal loop diagram without link and loop polarities

The gains of the links can be evaluated like this:

Link	Definition of link gains	Initial Gain	Polarity
Births'Pop	BirthRate	0.2 year^{-1}	S
Pop'Births	+1	1.0	S
Deaths'Pop	1/LifeTime*MaxPop^2/(MaxPop-Pop)^2	0.0521 year^{-1}	S
Pop'Deaths	-1	-1.0	O

The gains of the two loops can be computed by multiplying together the gains of their links, like this:

Loop	Definition of loop gains	Initial Gain	Polarity
Births'Pop'Births	Births'Pop*Pop'Births	0.2 year^{-1}	R
Deaths'Pop'Deaths	Deaths'Pop*Pop'Deaths	-0.0521 year^{-1}	B

Note the unit of measure for the gains. In a simple loop, the gain can be expressed as:

$$loopgain = Xt^{-N}$$

where N is the order of the loop (the number of levels) and *t* is the time unit of the simulation. For composite loops (see 8.4) there are several sub loops involved, so we get a summation as the way to express the gain of a composite loop:

$$loopgain = X_1 t^{-1} + X_2 t^{-2} + \ldots + X_N t^{-N} = \sum_{i=0}^{N} X_i t^{-i}$$

The (initial) polarities can be used to put the (initial) polarity indicators on the link and the loop symbols of the diagram, like this.

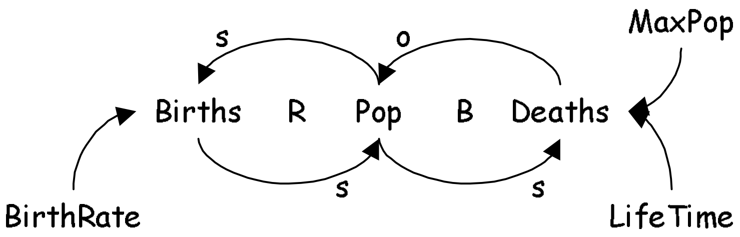


Figure 16: Causal loop diagram with dynamically calculated link and loop polarities

The polarities of the links that are outside of any loop can be computed the same way. This is left as an exercise for the interested reader.

As described earlier, the links and loops in the above diagram can be treated as variables. This means that their values can be displayed over time the same way as for regular variables. The time graph below shows the development of the link gains of the above model.

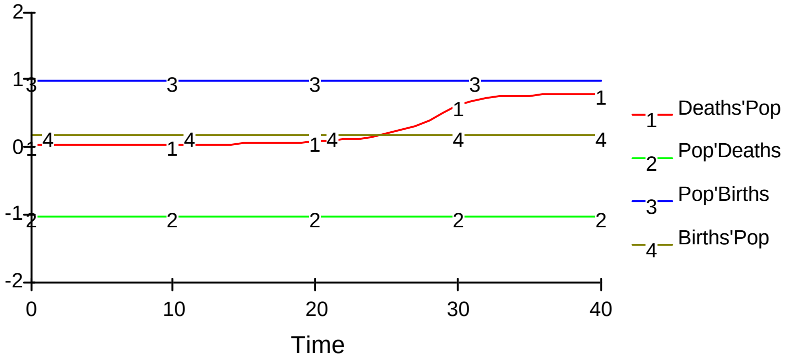


Figure 17: Link Gains

Note how the link from Population to Deaths (labeled 1) is strengthened as the population grows towards its capacity limit. Below, we have put the gains of the loops into a similar graph. In the graph to the left, the loop gains are shown unaltered, while their absolute values are displayed to the right. Note how the reinforcing loop (2) dominates until around time 25, where the balancing loop takes over.

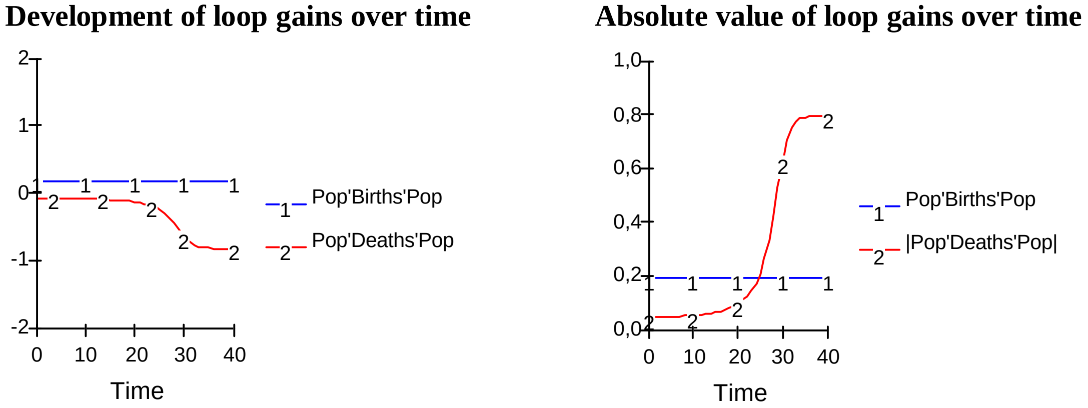


Figure 18: Loop Gains

The expression DERIVN(Pop, 2) can be used to compute the second time derivative Pop’’(t) of the variable Pop. We see that the value of Pop’’(t) is positive until time 25, when it goes negative. The positive section overlaps with the period where the reinforcing loop dominates, while the negative section corresponds to the part of the simulation where the balancing loop is strongest. The figure below shows the first and second time derivative of Pop.

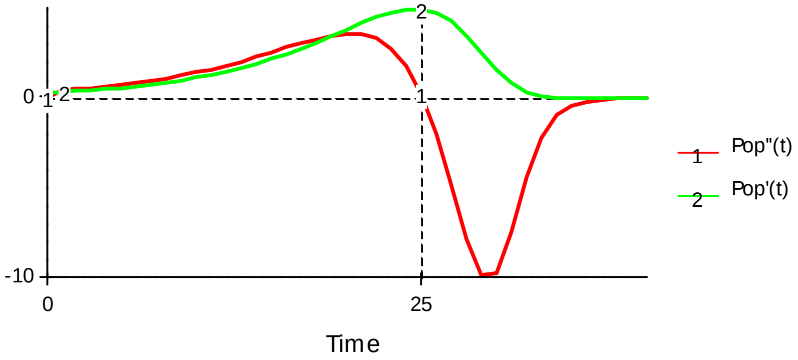


Figure 19: First and second order rate of change to the population state

Around time 25 Pop''(t) is zero. At this point the population is still growing, as can be seen from the value of Pop'(t), which denotes the net flow of the population. In fact, the growth rate is at its highest at the inflection point of Pop''(t).

For this simple model, with only one level, we have a single eigenvalue, λ, with the following definition:

Lambda = Births'Pop'Births + Deaths'Pop'Deaths

The figure below shows the development of Lambda over time.

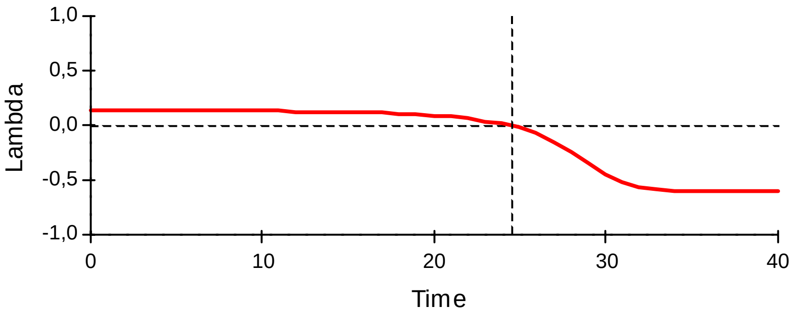


Figure 20: Development of eigenvalue

Based on the sign of the eigenvalue, we can see that the model is diverging (positive lambda) until around time 25, and diverging (negative lambda) thereafter.

7.1 Contribution of Links

As described in chapter 5.5, we can determine how much each link contributes to a given eigenvalue (and thereby a given elementary mode of behavior). Contribution is related to Lambda, Gains, Sensitivity and Elasticity. The table below gives the definitions of sensitivity, elasticity and contribution for the links of our example model.

Link	Sensitivity	Elasticity	Contribution
Births→Pop	Pop’Births	Births’Pop*Pop’Births/Lambda	Births’Pop*Pop’Births
Pop→Births	Births’Pop	- “ -	- “ -
Deaths→Pop	Pop’Deaths	Deaths’Pop*Pop’Deaths/Lambda	Deaths’Pop*Pop’Deaths
Pop→Deaths	Deaths’Pop	- “ -	- “ -

Figure 21: Link sensitivity, elasticity and contribution

The graphs below show the development in link sensitivity, elasticity and contribution over time. Only the contribution will actually be used in the visualization of the links, as shown in the diagrams in chapter 7.2.

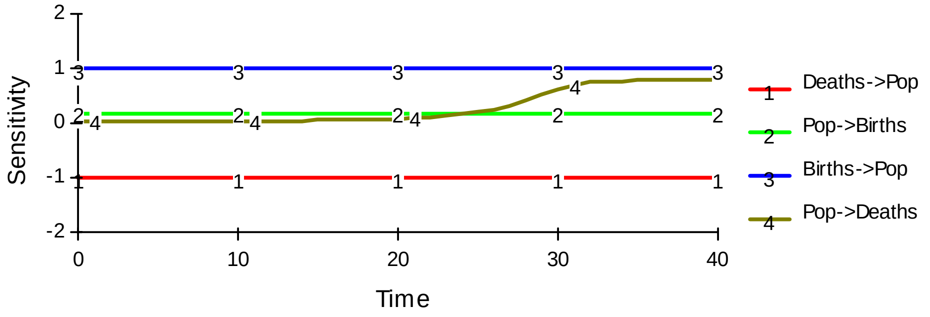


Figure 22: Link Sensitivities

The *elasticity* is the *sensitivity* times *gain* divided by *lambda*:

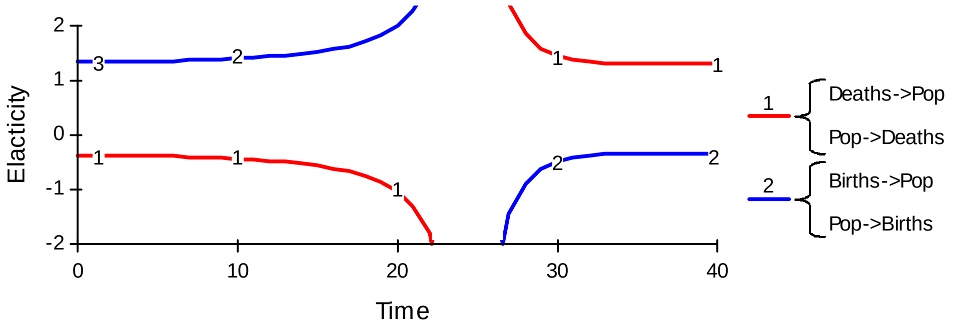


Figure 23: Link Elasticities

The *contribution* is the *elasticity* times *lambda*, which is the same as the *gain* times the *sensitivity*:

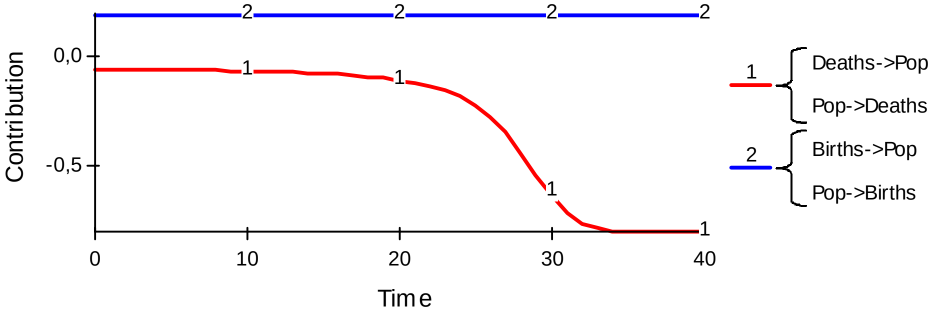


Figure 24: Link Contributions

Remember that the sum of the link contributions at any point in time is equal to lambda at that point. Compare the contributions of Figure 24 to the value of lambda in Figure 20.

7.2 Putting it all into play

In the diagrams below, we have used the *contribution* of the links to determine color and line with for the link symbols. The dominant *mode* is displayed in the lower, right hand corner of the diagram.

In the initial phase (time=10) the reinforcing loop is dominant, causing a diverging mode. The loop involving *Births* and *Pop* is drawn using red, thick lines. The balancing loop is actually counteracting, but it is much weaker than the first loop at this point in time.

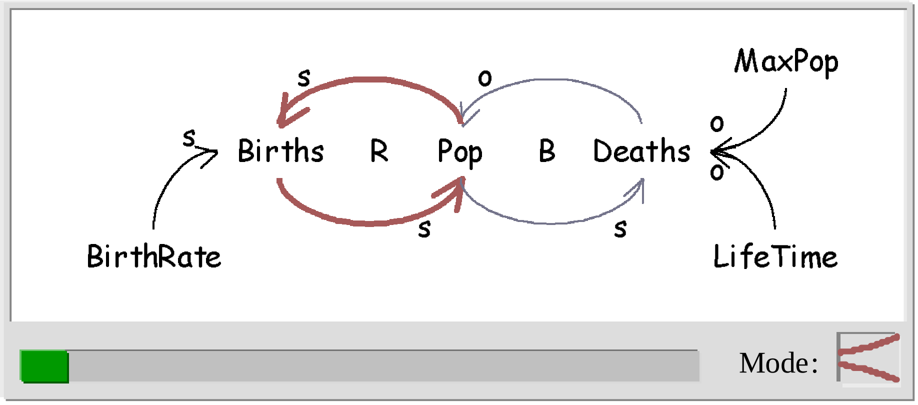


Figure 25: Initial Phase

Around time 25 the picture looks like this:

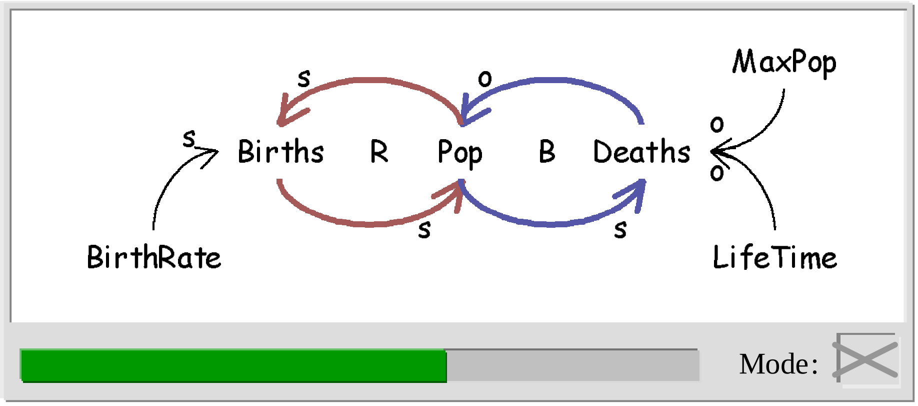


Figure 26: Transition Phase

The mode is now linear, and the two loops are equally strong, one pulling in each direction.

Towards the end of the simulation, the balancing loop takes over the dominance, and we get a picture like this one:

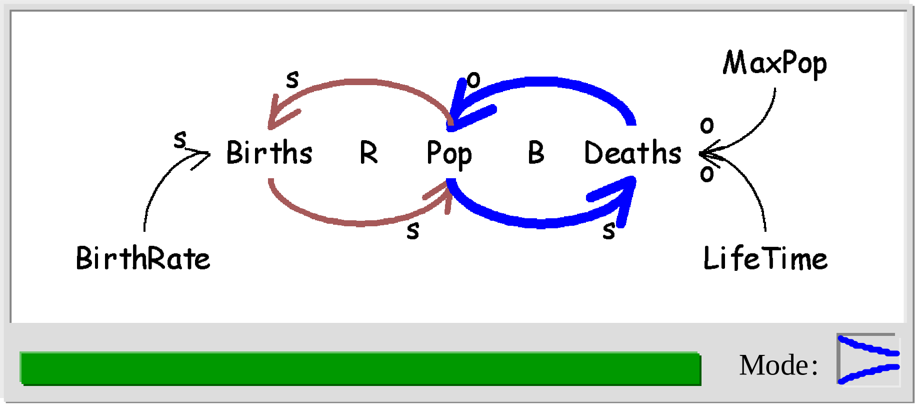


Figure 27: Final Phase

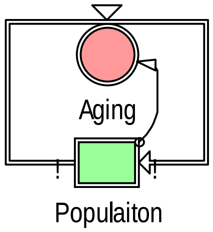
Here the balancing loop between Pop and Deaths has taken over the behavior, and put the simulation into a converging mode, as indicated in the lower, right-hand corner of the above diagram.

8 Modeling Constructs that Need Special Handling

Some modeling and diagramming constructs put special challenges on the analysis and visualization described above.

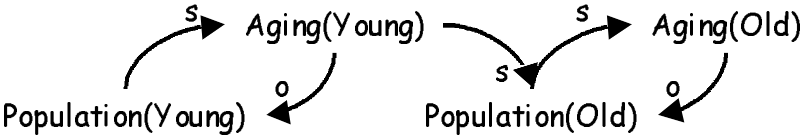
8.1 Dealing with arrays

In analyzing a model for eigenvalues, gains, etc. each array element must be treated as an independent scalar variable. For arrays, the following AFD diagram is possible, for example:



Here a population is an array, dividing the population into age groups. People flow out of one group, and into the next. (Births and deaths are omitted.)

A causal loop variant of this diagram can look like this:

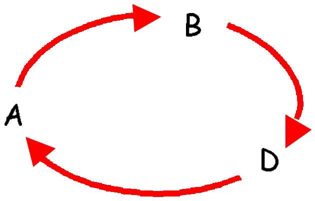


8.2 Partial Diagrams

In a partial causal loop diagram we must ensure that missing variables do not break the feedback paths through the model. This leads to the following rule:

If there is a dependency (in the equations of the model) between two variables A and B that are displayed in a causal loop diagram, there must be a path of links from A to B in the diagram.

As an example, let us assume that the underlying model has a loop that involves the variables A, B, C, D and back to A. The following diagram is valid (since C is not displayed, the diagram shows a link directly from B to D).



8.3 Internal Levels

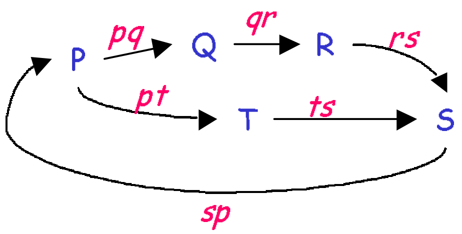
Internal levels of functions, such as DELAY, must be taken into account when computing eigenvalues, etc. of a model.

8.4 Composite loops

If we include auxiliary variables other than rates in our causal loop diagrams, we can get paths that split into sub-paths on the way around a feedback loop. In order to compute the gains of each of the links, we can make use of the following rules:

1. The gain of two parallel paths (starting and ending in the same variable) is equal to the sum of the gains of each path.
2. The gain of two sequential paths (one starts where the other ends) is equal to the product of the gains of each path.

The gain of the loop below is ((pq*qr*rs)+(pt*ts))*sp. (Letter-pairs represent link gains)



9 References

Richardson G. P. (1995). *Confusion in the polarity of major loops. Proceedings of the International Systems Dynamics Society*, Tolyo: System Dynamics Society

Kampmann, C. (1996): Feedback Loop Gains and System Behavior, *Proceedings of the International Systems Dynamics Society*, Boston: System Dynamics Society

Myrtveit, M. (2000). Object-oriented Extensions to System Dynamics, *Proceedings of the International Systems Dynamics Society*, Bergen: System Dynamics Society

Saleh M., Davidsen P.I. (2000). An eigenvalue approach to feedback loop dominance analysis in non-linear, dynamic models, *Proceedings of the International Systems Dynamics Society*, Bergen: System Dynamics Society

Senge, P. (1994). *The fifth discipline: the art and practice of the learning organization*. New York: Doubleday.

Scott, G. (1999) Mini-Model Presentations: A Tool for teaching dynamic systems thinking, *Proceedings of the International Systems Dynamics Society*, Wellington: System Dynamics Society

Wayner, P. (1988): Error-Free Fractions. *Byte Magazine*. June 1988.