

PATTERN-BASED SYSTEM DESIGN/OPTIMIZATION

Gönenç YÜCEL

g.yucel@tudelft.nl

Delft University of Technology (TU Delft)
2628 BX, Delft, The Netherlands

Yaman BARLAS

ybarlas@boun.edu.tr

Boğaziçi University (BU)
34342 Bebek, Istanbul, Turkey

ABSTRACT:

Despite its success and growing practitioner base, System Dynamics (SD) still lacks a strong and rich enough support toolbox, i.e. a set of formal mathematical tools that can support the modeler/practitioner in various stages including model identification, calibration, behavior analysis, policy design and sensitivity analysis. The study presented in this paper is an attempt towards developing such a support tool that can be used for pattern-based parameter search, which may be utilized in model identification, validation and policy analysis stages. The tool mainly incorporates a 2D pattern recognition algorithm and an optimization heuristic in order to search values for selected model parameters that yield a model behavior similar to the desired one in terms of pattern characteristics. The proposed tool is implemented, and a series of test experiments are conducted on three sample models in order to reveal the performance of it. Based on these experiments, the primary assessment about the proposed method is that its performance is quite satisfactory and it stands as a promising automated parameter search tool, which can be utilized even in the cases where data series representing the desired model behavior is missing.

INTRODUCTION:

Despite its success and growing practitioner base, System Dynamics (SD) still lacks a strong and rich enough support toolbox, i.e. a set of formal mathematical tools that can support the modeler/practitioner in various stages including model identification, calibration, behavior analysis, policy design and sensitivity analysis. In that respect, integration of new optimization tools to SD method and software is one of the topics listed by several authors as one of the key future challenges of the field (Richardson 1999; Coyle 2000).

The study presented in this paper is an attempt towards closing the mentioned gap, by developing a *pattern-based parameter search/optimization method* that can be utilized in model identification, validation and policy analysis stages. The method mainly incorporates a 2D pattern recognition algorithm and an optimization heuristic in order to search values for selected model parameters that yield a model behavior similar to the desired one in terms of pattern characteristics.

The overview of the paper is as follows; in the following section a brief discussion on the application of optimization in the context of SD will be provided. The section following that will discuss the pattern emphasis in SD and former research on pattern characterization. After these two sections, structure of the pattern-based parameter search method will be introduced. Before the conclusion section, results of the tests conducted with three different SD models are provided.

OPTIMIZATION IN SYSTEM DYNAMICS:

In several stages of a SD application, modeler faces the challenge of figuring out values of some model parameters in order to achieve a desired objective. Considering the number of parameters that can be manipulated in a modest SD model and non-linearity of the interactions, this may be a quite time consuming and inefficient search process. Researcher's knowledge about the system structure may provide valuable guidance for manual search and generally satisfactory results can be obtained. However, it is also the case that such intuitive and limited search strategies may miss parameter combinations that are desirable with respect to the concerns of the researcher. Utilizing optimization algorithms for these kinds of parameter search is a very promising application in terms of providing an automated and more efficient option to the researcher.

Due to the mathematical complexity of SD models, it is not possible to come up with a representation of these systems that allows the direct usage of non-linear optimization methods (e.g. representation as a constrained optimization problem to be used in constrained optimization approaches, or representation in terms of time functions to be used in unconstrained optimization heuristics). Hence a plausible way seems to be utilizing simulation-based optimization approaches, in which objective function being optimized is calculated via simulating the system.

Though being limited in number, there are some studies conducted on simulation-based optimization in the SD literature, which aim to combine optimization and SD (Keloharju and Wolstenholme 1988; Graham, Morecroft et al. 1992; Miller 1998; Dangerfield and Roberts 1999; Coyle 2000). Additionally, some SD simulation packages like Vensim[®], Powersim[®] and DYSMOD[®] incorporate a sort of search

heuristic in order to optimize/improve a given objective function based on a given SD model (Coyle 2000).

A major reason why optimization is rarely applied in SD studies is directly related to one of the main characteristics of SD approach; importance of the dynamics *pattern* observed, rather than a value that a system variable takes at a point in time during simulation. However, in the current state of the field established knowledge base that can be used to formulate objective functions capturing behavior patterns is very limited. The objectives of plausible optimization applications are generally end value of a parameter, deviance between two variables or peak value of a variable during the simulation. All of the literature mentioned above as examples from SD literature, utilize optimization algorithms on objectives in one of these forms. However, these types of objectives generally have limited relevance in the SD context. This relevance argument may be much clearer on the following example. Assume that desired dynamic pattern that is sought via a search heuristic is the one given in Figure 1.

This dynamic pattern (e.g. *Base Pattern*), known as S-shaped growth or logistics curve, can be characterized with an exponential growth phase followed by a goal-seeking phase leading to stabilization. Consider the two candidate patterns given in Figure 2 (e.g. *Candidate 1* and *Candidate 2*). Using a traditional evaluation criteria like ‘proximity at time 30’ or ‘minimum squared difference’, *Candidate 1* will perform much more better than *Candidate 2*. However, it is evident that in terms of *pattern* characteristics, *Candidate 1* has nothing to do with S-shaped growth type pattern, so in a pattern-based comparison *Candidate 2* is clearly superior to *Candidate 1*. Consider the case that researcher is seeking the policy that yields stabilizing growth, *Desired Pattern*. In that case, the optimization criteria should be able to detect the *pattern*-wise similarity between *Desired Pattern* and *Candidate 2*.

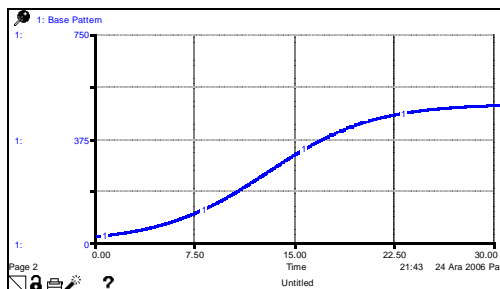


Figure 1: A desirable (objective) pattern that is S-shaped

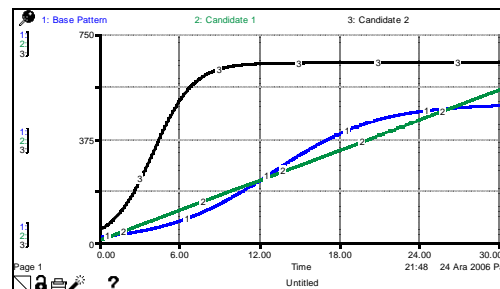


Figure 2: Two candidate patterns compared to the desirable one

PATTERN CHARACTERIZATION IN SD:

Despite the importance of characterizing the dynamic behavior generated by the model, research in this field is very limited. At the current state of the field, the only major attempt towards an automated and formal behavior characterization is the pattern classifying algorithm of Kanar and Barlas, which is a Hidden Markov Model-based pattern classifier (Barlas 1996; Kanar 1999; Kanar and Barlas *Under revision*). This pattern classifier is originally designed to be used for structure-oriented model validation purposes, in which model generated behavior pattern is evaluated against the modeler’s expectation under certain extreme conditions (Barlas 1996). Given an

expected dynamics pattern, algorithm is designed to return a likelihood value that represents the degree of model generated pattern's fit to that expectation. A set of basic dynamic pattern classes covering the majority of plausible behaviors is embedded in the algorithm, and algorithm is trained with time-series data that fit into one of those classes.

Despite the success of the algorithm and satisfactory coverage of the predefined pattern set, difficulty in integrating the algorithm with existing modeling platforms stands as an obstacle for widespread utilization. Recently, Boğ and Barlas developed a stand-alone software that can be integrated with Vensim modeling platform (Boğ and Barlas 2005), and Soylu (2006) made some improvements in the algorithm and its trained data base.

PATTERN-BASED PARAMETER SEARCH APPROACH:

Overview of the Approach:

As a first step towards an optimization-based support tool, a pattern-based parameter search approach is proposed and implemented. An overview of the implemented algorithm is provided in Figure 3. As seen in the figure, implementation stands on three pillars. The first one is the user, the modeler. Two basic inputs from the modeler are expected. The first input is the model structure, provided on a simulation platform. The second input is the parameter set for which values yielding desired model behavior will be sought, and the ranges for these parameters. Second pillar is the simulation platform. After each run, simulation platform will pass the simulation output in the form of a data series to the parameter search module for evaluation with respect to optimization objective, and then wait for another set of parameters in order to repeat the cycle. The third and final pillar is the pattern-based parameter search/optimization module. This module is mainly composed of two components. The first is the component that makes a pattern-based evaluation of the data series provided by the simulation platform after each simulation run. As a consequence of this evaluation, an output value is produced, which is utilized as the objective function value by the optimization component; the second part of this platform. After receiving this output value, optimization component generates another set of candidate parameter values by utilizing an optimization heuristic, and sends this new set to the simulation platform.

In short, the approach proceeds as follows; the optimization module generates candidate parameter values, and these values are evaluated based on the degree of fit between the desired pattern and the pattern obtained by using these parameter values. In order to do so, model is run with a candidate set of values and degree of fit of the resultant behavior is calculated by the pattern recognition module. Output from this evaluation is used by the optimization module to generate a new set of candidate parameter values. This cycle continues until a stopping condition is met (e.g. time constraint, degree of fit, lack of improvement in the last n steps, etc.).

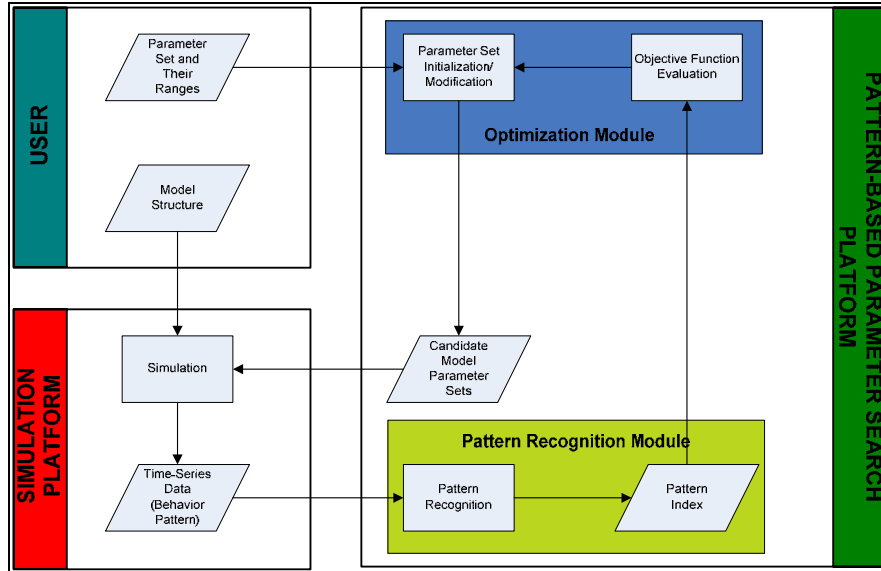


Figure 3: Overview of the proposed pattern-based parameter search/optimization approach

Specifications of the Implementation:

As seen in the overview, simulation platform should be capable of communicating with an external software or a computer code, or it should be capable of taking inputs from and passing outputs to a common platform with the optimization and pattern recognition modules. Most of the existing SD application software (e.g. Vensim[®], Stella[®], Powersim[®], etc.) at least has the second capability; hence technically it is possible to implement this approach using any of these software. However, for the sake of focusing purely on the performance of the approach and avoiding complications related to communicating different software platforms, we conducted simulations in MATLAB[®], which is also the platform utilized for optimization and pattern-recognition processes. Hence, all sample models to be discussed in the following section are constructed in MATLAB[®].

Basically, a pattern-recognition algorithm to characterize a 2D pattern (e.g. dynamic behavior of the model) is required for the implementation of the approach. In this experimental implementation, we decided to utilize the algorithm developed by Barlas and Kanar (Barlas 1996). One of the reasons of this choice was the fact that this algorithm was already been trained to recognize a set of patterns most relevant to SD models. Second reason was the availability of the code and documentation for the algorithm., We utilized the recently enhanced and improved version of this algorithm, discussed in (Soylu 2006).

Final part of the implementation is the optimization module. A genetic algorithm implementation is used for this module. Genetic algorithm is an optimization heuristic, developed by John Holland (Goldberg 1989; Coveney and Highfield 1995; Holland 1995), that works according to evolutionary principles. Algorithm starts with an initial population, in which each individual corresponds to a candidate solution. In each iteration, individuals with high fitness values (e.g. a function value attributed to each individual based on the given objective) are chosen as parents and new individuals to generate the following generation are produced with these parents. In order to do so,

two basic mechanisms are used. First is mutation; a new candidate is created changing some part of an existing candidate. The second one is crossover; a new candidate is created by combining some parts of two existing candidates. In this way the population of candidate solutions is expected to evolve into a better state with respect to the objective. The algorithm iterates in this way in order to find a solution for the given problem. Reader may refer to (Keane 1996), (Raynard-Smith, Osman et al. 1996) and (Goldberg 1989) for a more comprehensive description of genetic algorithms. In our experimental study we utilized the “*Genetic Algorithm and Direct Search Toolbox*” of MATLAB[®].

Specifications of the Genetic Algorithm (GA) Used:

As it is evident from the general description of the GA given above, there is a set of constructs that needs to be specified for an implementation. In this part, specifications of our GA implementation are provided.

Coding the candidate solutions, and solution populations:

Each individual in the population (e.g. each candidate solution) corresponds to a set of parameter values in our case. For example, in an optimization problem formulated in order to seek values for two model parameters, each individual in the population is a 2D vector, each dimension corresponding to one of the parameters. The size of the population is set to be 30. One important point about the initialization step is that genetic variety shall be maximal in the initial population. This is generally done via random initialization of the individuals. A similar approach is utilized in our case. Each dimension of the vector is initialized using a uniform distribution and feasible range given for the parameter represented by that dimension of the vector.

Fitness function:

Pattern recognition algorithm is utilized in order to calculate the fitness value of individuals in the population. For each parameter value vector in the population, model is run and the likelihood of the resultant pattern with respect to desired pattern is calculated. The likelihood value returned by the algorithm serves as the fitness value of the parameter value vector in the population.

Generating New Individuals:

In each iteration, 5 individuals (*elites*) from the existing population are selected in order to survive for the next generation. Then individuals are selected with respect to the fitness value in order to constitute the parents for the new 25 individuals to be created. The selection function used in this implementation is the ‘*stochastic uniform selection*’ option provided in the MATLAB. 20 of the new individuals are generated by the cross-over operation. During the cross-over operation, two parents are selected. Then each dimension of the new individual is determined using the same dimension of the parents. For example, in a search with 2 parameters the value of first parameter of the new individual is determined using the values for the first parameters of the parents. In our case value for the children is determined as a weighted average of the parents’. The weight used in this operation is randomly determined in each case and lies in the range of [0, 1]. 5 of the new individuals are generated via mutating existing individuals. In this operation, an individual in the existing population is chosen and a random number is added to each vector entry of that individual. This operation is used in order to increase the genetic variety in the population and prevent lock-in to local optima. The

random number to be added in each mutation comes from a Normal distribution with mean 0 and variance 2. The combination of 5 elite individual, 20 individuals generated by cross-over and 5 generated by mutation constitutes the new population for the genetic algorithm. These individuals are evaluated with respect to the fitness function and the whole cycle for generating new individuals is repeated.

Stopping Criteria:

Two stopping conditions are utilized in our GA implementation. The first one is the number of generations. The algorithm is set to stop after 100 generations. The second condition is the number of consecutive generations by which best solution found does not improve (e.g. stall generations). This is set as 30 generations in our case.

EXPERIMENTS WITH THE IMPLEMENTED ALGORITHM:

In order to test the performance of the developed algorithm under different conditions, a series of experiments are conducted. At this initial stage of research, initial settings of the genetic algorithm are preserved throughout experimentation stage (e.g. no modification like increasing the population size or changing stopping criteria is done in order to compensate the performance change). Mainly three aspects of the model/problem are altered during the experimentation stage;

1. Size of the model (e.g. second order linear, third order linear, third order non-linear)
2. Number of model parameters altered by optimization algorithm (e.g. 2, 3, 5 and 7)
3. Feasible ranges of parameters altered by optimization algorithm (e.g. narrow or wide)

A set of experiments are conducted with different combinations of these three aspects. In each experiment, the algorithm is used to find a vector of parameter values that yield a specific desired pattern. For each specific pattern, 30 trials are performed¹. Due to the stochastic initialization step of the optimization algorithm, it is possible to obtain different results in these 30 trials. Then, all 30 parameter value vectors are tested on the model for visual evaluation.

In the following sections, the models used for experimentation are introduced and the results are summarized in the corresponding figures. Although all 30 vectors from each trial are evaluated visually, a sample set of 4 points are introduced with their corresponding model behavior instead of giving all 30 of them. There may be two exceptions for this. The first one is the case where algorithm returns a parameter set which is *visually* judged as unsuccessful with respect to the desired pattern. These cases are discussed individually. The second exception is cases where the parameter values returned by the algorithm cluster around less than 4 distinct points. In those cases, naturally less than 4 points will be provided. Apart from the summarized results given in the figures, brief discussions about the exceptional and problematic cases are also provided where necessary.

¹ The number of replications/trials for each case, 30, is coincidentally equal to the population size of the genetic algorithm. They have no direct relation.

Finally, since pattern identification algorithm developed by Barlas and Kanar (Barlas 1996; Kanar 1999) is utilized in this research, a subset of the pattern set defined by Barlas and Kanar is used as the set of plausible patterns. Reader may refer to Appendix A for the set of patterns used in this research.

Thermostat Model:

The first model used for experimentation is a simple second order linear stock adjustment structure, the *thermostat model* (Figure 4). The differential equations corresponding to this simple system and analytical solution for the parameter values that yield oscillations are provided in Appendix B. In the instance of the model used for test purposes, *Perceived State* and *State* variables are both set to -10, and *Goal* is set to 0 at $t=0$. Time horizon is set to be 150 time units and variable of interest is chosen to be the *State* variable. Only 2-parameter optimization experiments are conducted with this model. The two parameters, for which the pattern-based optimization algorithm is set to seek values, are the *Perception Delay* and the *Adjustment Time*. The feasible range for both of these parameters is set to be $[0, 20]$.

The results of the tests conducted on this model are summarized in Figure 5. In the negative exponential growth case only two different points are identified; (4.3, 1.0) and (6.9, 1.0). Among them the second point was the dominant one (22 of 30 results). The algorithm returned no other point in these 30 runs.

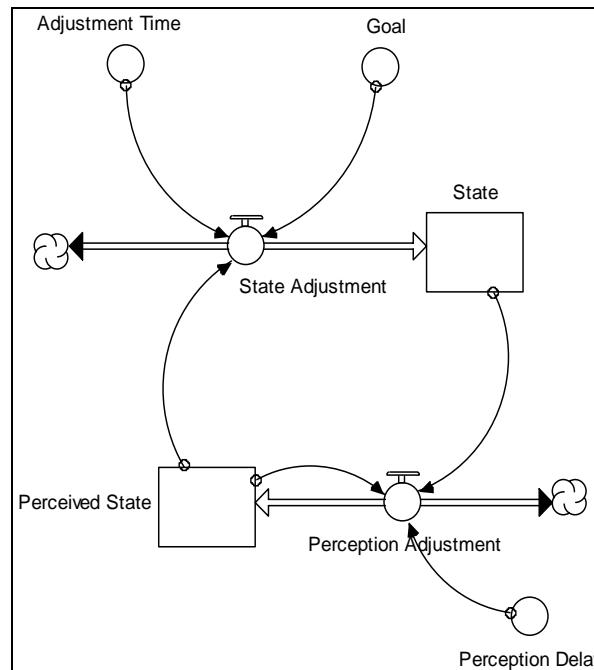


Figure 4: Stock-flow diagram for the thermostat model

As mathematically demonstrated in Appendix B, this model can demonstrate only stable oscillations and oscillating behavior is obtained when the *Adjustment Time* is less than four times the *Perception Delay*. All the results provided by the algorithm clearly satisfy this relationship. The points returned by the algorithm clearly clustered around the edges of the end points of the ranges given for the parameters. The value of the *Adjustment Time* was 1.0 for all 30 runs. On the other hand, the values returned for the

Perception Delay were between 19 and 20 in 26 cases among 30. The model outputs generated with these points were quite satisfactory in terms of obtaining the desired behavior.

In the growth-and-decline case, a similar clustering around the point (1.1 ; 1.2) is observed (23 of 30 results). Although corresponding model behavior demonstrates basic characteristics of the sought behavior (e.g. *growth with decreasing rate of change followed by a decline to equilibrium; decline level less than growth level*), visually they are not evaluated as pure growth and decline behaviors.

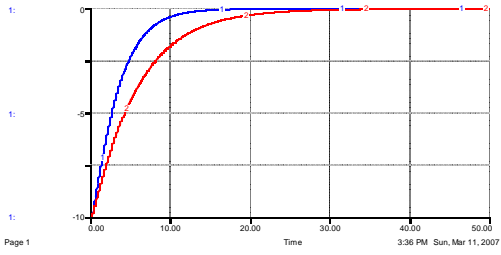
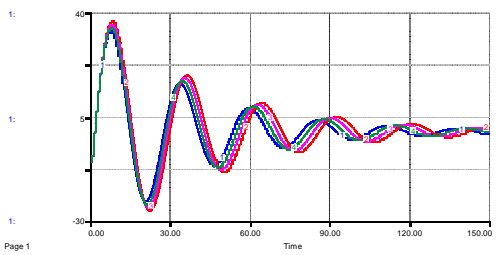
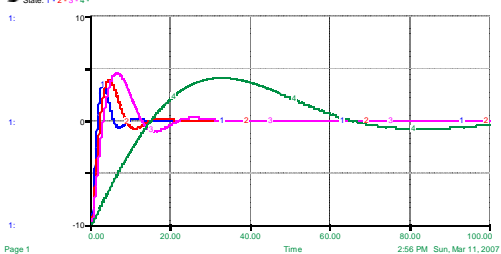
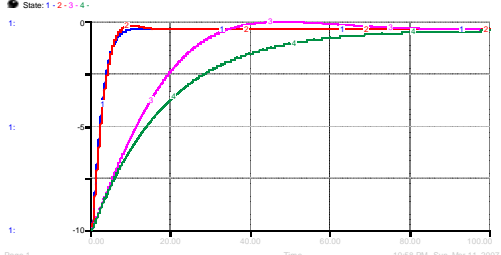
Desired Behavior Pattern	Parameter values returned by the algorithm (Adj. Time ; Perc. Del.)	Behavior patterns obtained by using returned parameter values
<p>Negative Exponential Growth</p>	<p>X1: (4.3 ; 1.0) (line 1) X2: (6.9 ; 1.0) (line 2)</p>	
<p>Oscillations</p>	<p>X1: (1.0 ; 17) (line 1) X2: (1.0 ; 18) (line 2) X3: (1.0 ; 19) (line 3) X4: (1.0 ; 20) (line 4)</p>	
<p>Growth and Decline</p>	<p>X1: (1.1 ; 1,2) (line 1) X2: (1.7 ; 2.0) (line 2) X3: (2.4 ; 3.3) (line 3) X4: (12.8 ; 16.0) (line 4)</p>	
<p>S-shaped Growth</p>	<p>X1: (3.3 ; 1) (line 1) X2: (3.7 ; 1.4) (line 2) X3: (19.5 ; 8.4) (line 3) X4: (19.9 ; 1.1) (line 4)</p>	

Figure 5: Results of the experiments with the thermostat model (*Search with 2 parameters*)

Finally we have conducted tests for two behavior patterns that are not possible to obtain just by modifying the *Adjustment Time* and the *Perception Delay*, given the particular initial conditions of the model. Since the *Perceived State* and the *State* variables are set to be equal to -10 and the *Goal* parameter is set to be 0, rate of change for the *State* will be decreasing during the initial phase of the model run, independent of the values of *Adjustment Time* and *Perception Delay*. Hence it is not possible to obtain pure S-shaped growth, for example. In order to test the behavior of the pattern recognition algorithm, S-shaped growth is also set as a goal and corresponding results can be seen Figure 5. As it can be seen, none of the patterns generated is an S-shaped growth pattern. The dominant result in this experiment (X3, found in 22 of 30 runs) yields a negative exponential growth pattern. It is seen that the pattern recognition algorithm evaluates negative exponential growth as the most proximate one to an S-shaped growth pattern, probably due to the fact that negative exponential growth is typically the second half of a pure S-shaped growth. To take the experiment further, a third parameter is allowed to the optimization algorithm, which makes it possible to produce an S-shaped growth pattern. In this additional experiment, a 3-parameter optimization is performed by adding the initial value of the *Perceived State* to the variables which can be altered by optimization algorithm. 10 runs are performed for this additional experiment, and a single point is obtained in these runs. The point and the resultant behavior pattern are given in Figure 6. As it can be seen, the pattern generated by the parameter values returned by the algorithm is an S-shaped growth pattern.

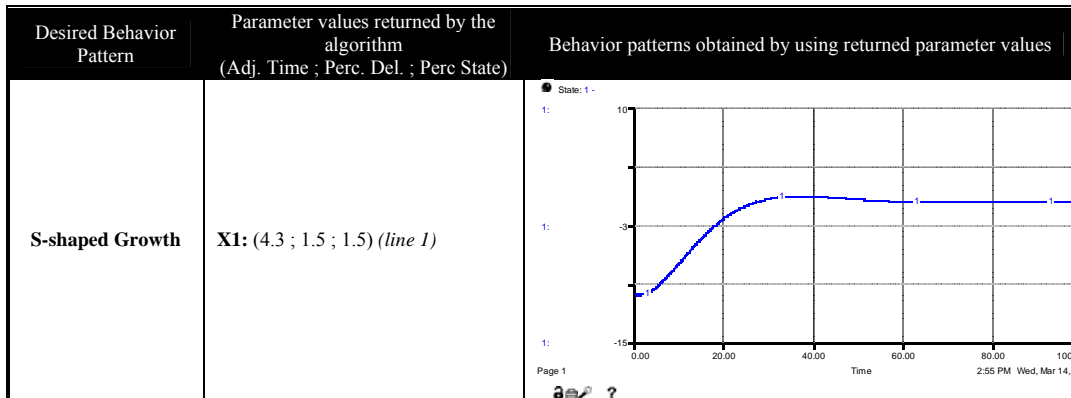


Figure 6: Search for S-shaped growth pattern with the thermostat model (Search with 3 parameters)

Stock Management Model with Two Stage Supply Line:

The second model used for experimentation is a third-order linear stock management model, given in Figure 7. Specifications of the model are provided in Appendix C. Basically, we have conducted experiments with different number of parameters to be altered by the optimization algorithm. 2-parameter, 3-parameter, 5-parameter and 7-parameter search experiments are conducted. For each of these cases, we have searched for three different desired patterns; oscillations, negative exponential growth and S-shaped growth. Our aim in this set of experiments was to observe the changes in the overall performance due to increased-decreased number of parameters.

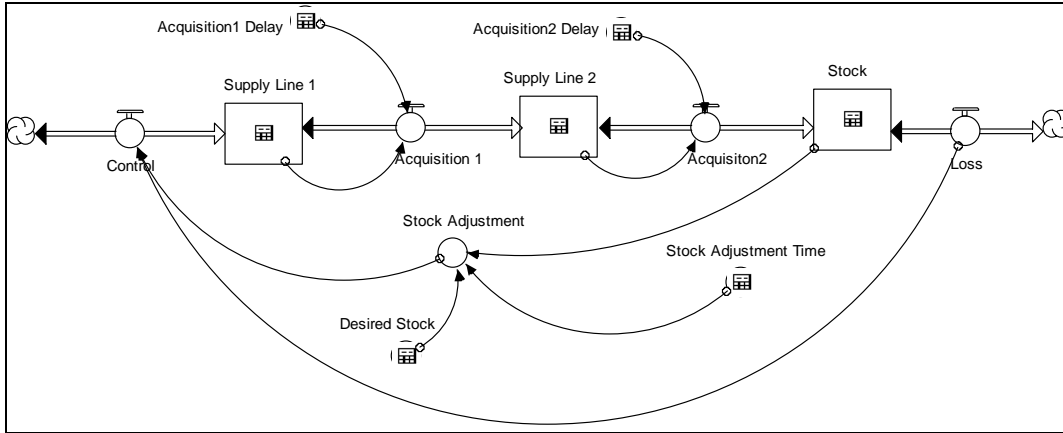


Figure 7: Stock-flow diagram for stock adjustment model with two stage supply line

The list of model parameters selected as the independent variables for the optimization algorithm in each of these experiment sets is provided in Figure 8.

Experiment Set	Independent Variables
2-parameter case	Acquisition Delay ² , Stock Adjustment Time
3-parameter case	Acquisition Delay 1, Acquisition Delay 2, Stock Adjustment Time
5-parameter case	Acquisition Delay 1, Acquisition Delay 2, Stock Adjustment Time, Desired Stock, Stock (initial value)
7-parameter case	Acquisition Delay 1, Acquisition Delay 2, Stock Adjustment Time, Desired Stock, Stock (initial value), Supply Line 1 (initial value), Supply Line 2 (initial value),

Figure 8: Set of independent parameters used in different experiments

The results obtained in the 2-parameter optimization case are summarized in Figure 9. In the oscillation case, the results are clustered around 3 points. It is possible to verify both by visual inspection and mathematical analysis (*see Appendix C*) that the patterns obtained by these points are oscillations. In the negative exponential case, very proximate parameter value vectors are found. Two of these vectors are also given in Figure 9.

In the optimization runs for S-shaped growth pattern, obtained results seem to generate two different patterns. One of them is the pattern generated by using X1 or X4, which is evaluated to be a pure S-shaped growth pattern. The second pattern is the one generated by X2 or X3. This pattern seems very similar to an S-shaped growth pattern in the time horizon of the experiment (100 time units). However, it is evident that the *Stock* variable does not converge to an equilibrium level with a decreasing rate of change by the end of growth phase as it should be in a pure S-shaped growth pattern; a decline behavior can be detected close to the end of simulation time horizon. So, with a longer time horizon it can be seen that actual behavior of the *Stock* is either growth-and-decline

² For the 2-parameter case, a single acquisition delay (Acquisition Delay) is used for the output flow of both supply line stocks (e.g. Acquisition Delay 1= Acquisition Delay 2= Acquisition Delay)

or oscillation. This observation points the sensitivity of the used algorithm to the time horizon chosen.

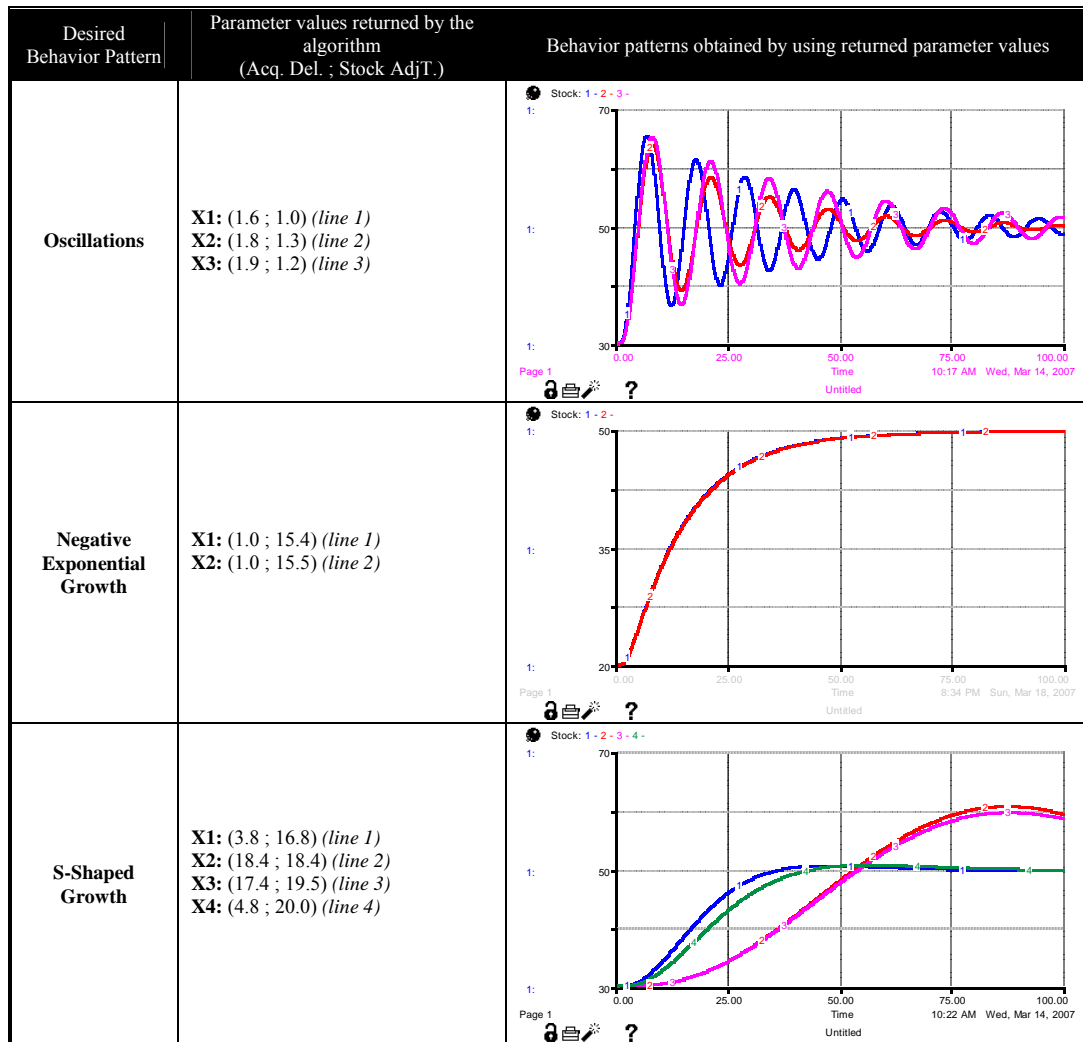


Figure 9: Results of the experiments with the stock management model (Search with 2 parameters)

The results obtained in 3-parameter, 5-parameter and 7-parameter search experiments are given in Figure 10, Figure 11 and Figure 12, respectively. Apart from a set of exceptions, the results obtained during these experiments were very promising.

Desired Behavior Pattern	Parameter values returned by the algorithm (Acq. Del. 1 ; Acq. Del. 2 ; Stock AdjT.)	Behavior patterns obtained by using returned parameter values
Oscillations	<p>X1: (1.5 ; 1.0 ; 1.1) (line 1)</p> <p>X2: (4.9 ; 1.0 ; 1.2) (line 2)</p> <p>X3: (8.8 ; 1.0 ; 1.0) (line 3)</p> <p>X3: (2.8 ; 3.4 ; 2.0) (line 4)</p>	
Negative Exponential Growth	<p>X1: (1.0 ; 1.0 ; 15.5) (line 1)</p> <p>X2: (1.2 ; 1.1 ; 15.7) (line 2)</p>	
S-Shaped Growth	<p>X1: (3.9 ; 3.8 ; 16.7) (line 1)</p> <p>X2: (18.3 ; 17.0 ; 19.1) (line 2)</p> <p>X3: (4.3 ; 3.4 ; 16.7) (line 3)</p>	

Figure 10: Results of the experiments with the stock management model (Search with 3 parameters)

Two of the problematic cases are observed in S-shaped growth experiment in 5-parameter search³; X3 and X4. X3 in that case yielded a decline-and-growth type of behavior, instead of the desired S-shaped growth behavior. In this case, it is seen that according to the likelihood value returned by the pattern recognition algorithm, observed decline-and-growth pattern is evaluated to be somehow similar to S-shaped growth. In fact, ignoring the initial 10 time periods, behavior is a perfect S-shaped growth. A similar shortcoming may be accepted as the underlying reason for the pattern caused by X4. First half of the pattern is an S-shaped growth pattern. Probably due to this part, pattern recognition algorithm attributes this pattern a high likelihood value. These two cases point out the necessity for improvement in the pattern recognition algorithm in order to increase its discriminative power.

³ Although it is not evident due to scaling problems, pattern generated by using X2 in this case is evaluated to be pure S-shaped growth.

Desired Behavior Pattern	Parameter values returned by the algorithm (Acq. Del. 1 ; Acq. Del. 2 ; Stock AdjT. ; Des Stock ; Stock Ini)	Behavior patterns obtained by using returned parameter values
Oscillations	<p>X1: (1.1 ; 1.6 ; 1.0 ; 30.2 ; 1.5) (line 1)</p> <p>X2: (1.7 ; 1.0 ; 1.0 ; 45.0 ; 12.0) (line 2)</p> <p>X3: (3.4 ; 1.2 ; 1.3 ; 32.0 ; 20.6) (line 3)</p> <p>X3: (1.0 ; 4.3 ; 1.0 ; 41.3 ; 11.7) (line 4)</p>	<p>Stock: 1 - 2 - 3 - 4 -</p> <p>Page 1 Time: 11:18 AM Wed, Mar 14, 2007 Untitled</p>
Negative Exponential Growth	<p>X1: (1.0 ; 1.0 ; 1.0 ; 15.3 ; 30.6 ; 13.4) (line 1)</p> <p>X2: (1.0 ; 1.0 ; 15.6 ; 42.5 ; 10.0) (line 2)</p> <p>X3: (1.1 ; 1.0 ; 1.0 ; 15.5 ; 46.0 ; 24.0) (line 3)</p> <p>X4: (1.0 ; 1.0 ; 15.4 ; 34.6 ; 28.0) (line 4)</p>	<p>Stock: 1 - 2 - 3 - 4 -</p> <p>Page 1 Time: 8:47 PM Sun, Mar 18, 2007 Untitled</p>
S-Shaped Growth	<p>X1: (3.9 ; 4.0 ; 17.3 ; 30.1 ; 27.7) (line 1)</p> <p>X2: (3.9 ; 3.9 ; 16.7 ; 45.7 ; 18.3) (line 2)</p> <p>X3: (5.6 ; 2.2 ; 16.6 ; 45.5 ; 28.7) (line 3)</p> <p>X4: (15.2 ; 19.5 ; 19.7 ; 35.6 ; 29.5) (line 4)</p>	<p>Stock: 1 - 2 - 3 - 4 -</p> <p>Page 1 Time: 11:35 AM Wed, Mar 14, 2007 Untitled</p>

Figure 11: Results of the experiments with the stock management model (Search with 5 parameters)

Desired Behavior Pattern	Parameter values returned by the algorithm (Acq. Del. 1 ; Acq. Del. 2 ; Stock AdjT. ; Des Stock ; Stock Ini ; SupLine1 Ini ; SupLine2 Ini)	Behavior patterns obtained by using returned parameter values
Oscillations	<p>X1: (1.0 ; 11.1 ; 1.0 ; 38.9 ; 26.0 ; 35.3 ; 52.8) (line 1)</p> <p>X2: (2.7 ; 2.6 ; 1.5 ; 35.2 ; 29.0 ; 37.9 ; 43.6) (line 2)</p> <p>X3: (1.0 ; 5.0 ; 1.3 ; 30.0 ; 26.1 ; 34.9 ; 30.0) (line 3)</p> <p>X4: (1.0 ; 5.0 ; 1.3 ; 30.0 ; 26.1 ; 34.9 ; 30.0) (line 4)</p>	
Negative Exponential Growth	<p>X1: (2.0 ; 1.2 ; 9.5 ; 39.5 ; 10.0 ; 24.0 ; 20.1) (line 1)</p> <p>X2: (1.5 ; 1.5 ; 11.2 ; 47.5 ; 11.0 ; 25.0 ; 22.5) (line 2)</p> <p>X3: (2.0 ; 1.5 ; 17.0 ; 30.3 ; 10.0 ; 28.1 ; 20.0) (line 3)</p> <p>X4: (1.0 ; 1.4 ; 10.0 ; 48.5 ; 10.5 ; 20.0 ; 20.3) (line 4)</p>	
S-Shaped Growth	<p>X1: (2.7 ; 4.9 ; 16.9 ; 48.0 ; 10.1 ; 36.7 ; 38.0) (line 1)</p> <p>X2: (6.1 ; 2.6 ; 19.1 ; 47.0 ; 11.0 ; 51.7 ; 30.0) (line 2)</p> <p>X3: (4.7 ; 2.7 ; 15.7 ; 41.2 ; 10.5 ; 39.9 ; 32.8) (line 3)</p> <p>X4: (1.1 ; 8.3 ; 19.9 ; 46.6 ; 13.7 ; 32.6 ; 58.9) (line 4)</p>	

Figure 12: Results of the experiments with the stock management model (Search with 7 parameters)

Market Growth Model:

The third model used for experimentation purposes is a third-order non-linear market growth model, whose stock-flow diagram is given in Figure 13. As in the former case, this model is also capable of demonstrating various behavior patterns dependent on the initial conditions and parameter values. Differing from the former model, this model incorporates two non-linear effect functions (e.g. *Deleffect* and *Backleffect*).

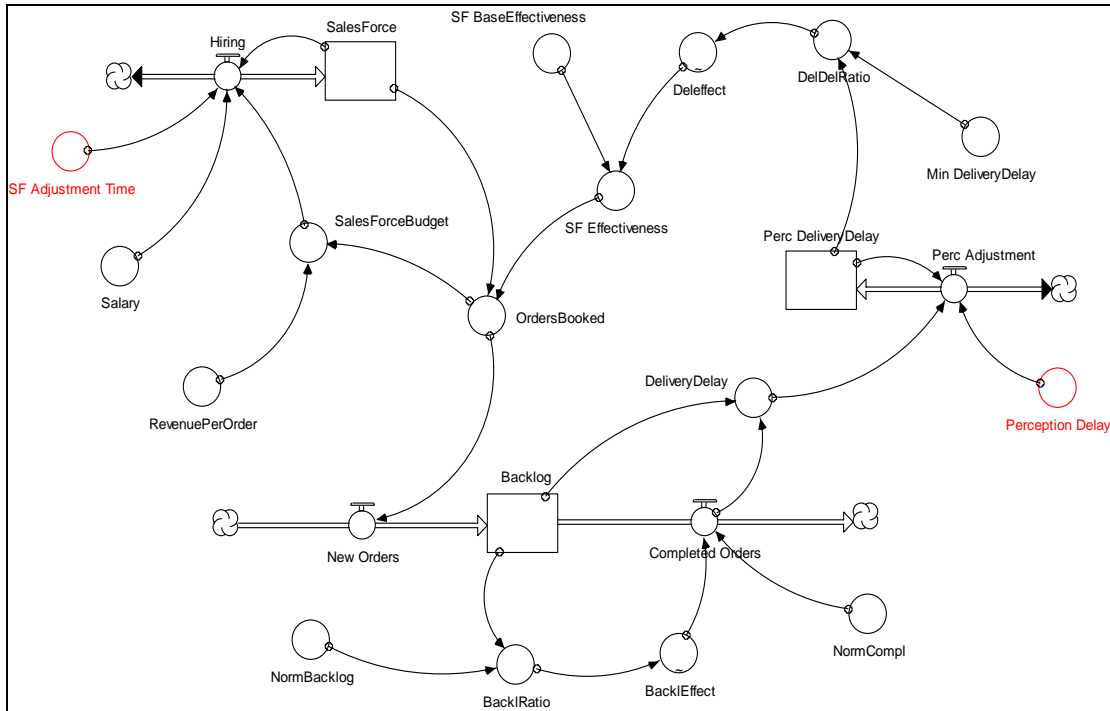


Figure 13: Stock-flow diagram for market growth model

We basically conducted two sets of tests using this model. In the first set, we tried 2-parameter search. Then, in the second set 3-parameter search with extended ranges is performed. In order to see the impact of the ranges on the algorithm's performance, feasible ranges for the parameters are extended significantly. The parameters used as variables for optimization and their corresponding feasible ranges can be seen in Figure 14.

Experiment Set	Parameters Used	Feasible Ranges
1	Perception Delay SF Adjustment Delay	[0, 20] [0, 20]
2	Perception Delay SF Adjustment Delay Min DeliveryDelay	[0, 80] [0, 80] [0, 40]

Figure 14: Parameters used and their ranges in experiments with market growth model

The results of the first set of experiments, where search is performed over two parameters, are summarized in Figure 15 and Figure 16. Tests are conducted for five different patterns (negative exponential growth, oscillation, growth-and-decline, S-shaped growth, and positive exponential growth).

The results obtained for the first four patterns are presented in Figure 15. Among those four experiments, growth-and-decline case turned out to be the most unsuccessful performance of the algorithm in this experimental study. Among 30 trials, algorithm returned 10 parameter value vectors very close to X2, though behavior pattern obtained by X2 is clearly not a growth-and-decline one. It is concluded that further experimentation is required for deeper investigation for this type of lock-in of the optimization algorithm.

Desired Behavior Pattern	Parameter values returned by the algorithm (PercDel. ; SFAdjT.)	Behavior patterns obtained by using returned parameter values
Negative Exponential Growth	X1: (1.0 ; 7.5) (line 1) X2: (1.0 ; 6.5) (line 2) X3: (1.0 ; 6.0) (line 3)	
Oscillations	X1: (1.3 ; 1.0) (line 1) X2: (2.0 ; 1.5) (line 2) X3: (2.9 ; 2.0) (line 3)	
Growth and Decline (gr2db)	X1: (40.0 ; 18.9) (line 1) X2: (35.0 ; 35.6) (line 2) X3: (1.0 ; 2.5) (line 3)	
S-Shaped Growth	X1: (1.9 ; 6.9) (line 1) X2: (1.9 ; 14.4) (line 2) X3: (2.9 ; 15.8) (line 3) X4: (3.5 ; 10.0) (line 4) X5: (5.8 ; 18.0) (line 5)	

Figure 15: Results of the experiments with the market growth model (Search with 2 parameters)

The test conducted in order to obtain positive exponential growth also deserves some extra discussion. The algorithm returned a single value vector in all 30 trials; (20 ; 39). In a time horizon of 100 time periods, which is the horizon used in the experiments, the result seems to be quite satisfactory (see Figure 16). However, when the time horizon is increased to 150 time periods, it can be seen that model is demonstrating an S-shaped growth type of behavior and will be stabilizing in the steady state. This case stands as another instance of sensitivity of the results to the time horizon chosen, which is also discussed in some of the previous cases.

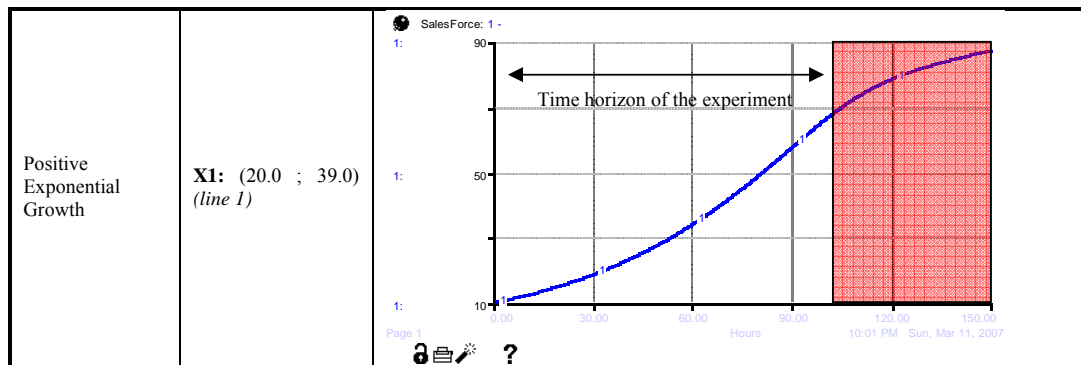


Figure 16: A Result of experiments with the market growth model (Search with 2 parameters)

A more extended experimentation is conducted in the 3-parameter search setting with the same model. In this case optimization is performed over *Perception Delay*, *SF Adjustment Time*, and *Min Delay* parameters. The results of these experiments are summarized in Figure 17, Figure 18 and Figure 19.

Results of the first three experiments are given in Figure 17. In the first two patterns tested (oscillations and growth-and-decline (*gr1da*)), all 30 solutions returned by the algorithm yield quite satisfactory results. It is worth mentioning that in the growth-and-decline case, despite the characteristic similarities among 4 different growth-and-decline patterns, all solutions yield patterns that satisfy the exact specifications of the desired pattern (see Appendix A for pattern codes and their specifications).

The third pattern tested was another growth-and-decline pattern (*gr1db*). One of the points returned by the algorithm in this experiment was (1.1 ; 11.0 ; 1.0), which yields a sort of negative exponential growth type of behavior (see the bottom graph in the *gr1db* case in Figure 17). Among 30 trials, the algorithm returned this specific point twice. Analyzing the objective function values, it is concluded that pattern identification algorithm attributes a low objective value for this point, which indicates that the pattern is not similar to the desired one. Hence, it is concluded that failure is mainly due to the optimization heuristic utilized, not due to the pattern recognition algorithm.

The experiments with two growth-and-decline patterns (*gr2da* and *gr2db*) and negative exponential pattern are summarized in Figure 18. In both of the growth-and-decline cases, all results returned by the algorithm are evaluated to be successful, except 1 point in growth-and-decline (*gr2db*) case returned by the algorithm once in 30 trials (see *X4* in *gr2db* case in Figure 18). Although that particular point yields a growth-and-decline behavior in general, the observed pattern does not perfectly match the *gr2db* type of growth-and-decline pattern.

Desired Behavior Pattern	Parameter values returned by the algorithm (PercDel. ; SFAdjT ; MinDelay)	Sample Output with Found Parameter Values
Oscillations	<p>X1: (2.7 ; 2.5 ; 1.3) (line 1) X2: (3.8 ; 3.5 ; 1.3) (line 2) X3: (7.2 ; 9.6 ; 1.8) (line 3) X4: (9.4 ; 13.8 ; 1.9) (line 4)</p>	
Growth and Decline (gr1da)	<p>X1: (5.7 ; 5.2 ; 10.0) (line 1) X2: (14.0 ; 5.0 ; 6.5) (line 2) X3: (39.8 ; 5.1 ; 2.9) (line 3) X4: (53.0 ; 5.0 ; 1.0) (line 4)</p>	
Growth and Decline (gr1db)	<p>Top Graph: X1: (2.2 ; 17.4 ; 6.6) (line 1) X2: (55.5 ; 24.6 ; 1.5) (line 2) X3: (64.5 ; 51.9 ; 1.4) (line 3) X4: (74.8 ; 23.2 ; 1.0) (line 4)</p> <p>Bottom Graph: X5: (1.1 ; 11.0 ; 1.0) (line 1)</p>	

Figure 17: Results of the experiments with the market growth model (Search with 3 parameters)

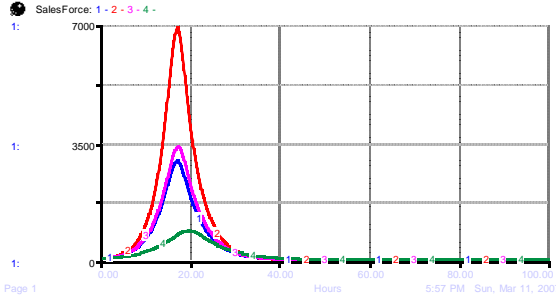
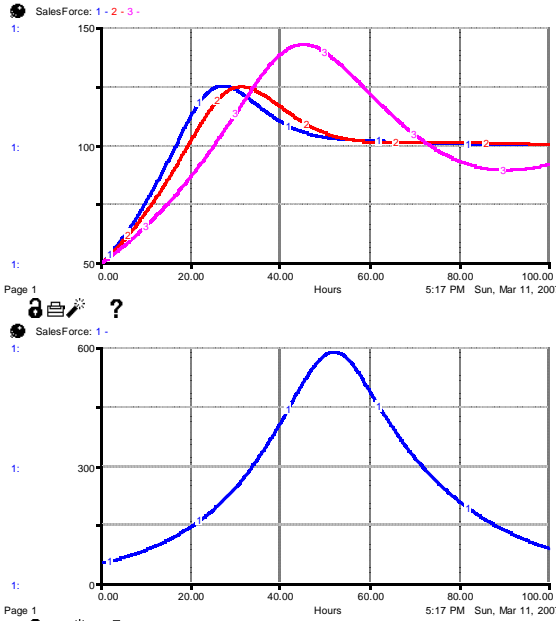
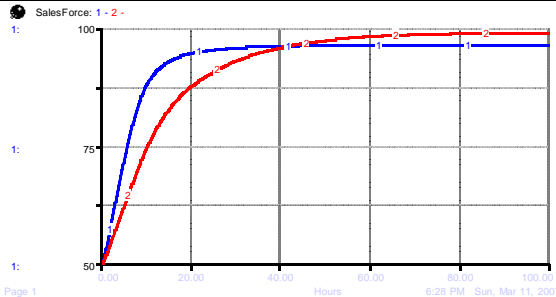
Desired Behavior Pattern	Parameter values returned by the algorithm (PercDel. ; SFAadjT ; MinDelay)	Sample Output with Found Parameter Values
Growth and Decline (gr2da)	X1: (27.5 ; 3.7 ; 8.0) (line 1) X2: (52.9 ; 3.1 ; 8.1) (line 2) X3: (62.6 ; 3.6 ; 4.5) (line 3) X4: (78.3 ; 5.7 ; 2.1) (line 4)	
Growth and Decline (gr2db)	Top Graph: X1: (3.9 ; 22.9 ; 3.7) (line 1) X2: (5.1 ; 26.9 ; 4.1) (line 2) X3: (12.3 ; 35.7 ; 5.9) (line 3) Bottom Graph: X4: (71.3 ; 18.7 ; 8.8) (line 1)	
Negative Exponential Growth	X1: (1.0 ; 9.0 ; 1.0) (line 1) X2: (1.4 ; 17.8 ; 1.2) (line 2)	

Figure 18: Results of the experiments with the market growth model (Search with 3 parameters)

Final set of behavior patterns tested are given in Figure 19. First of the tested patterns is S-shaped growth, in which a couple of unsatisfactory points are returned by the algorithm. One of those point is (1.0 ; 18.0 ; 1.0) (see X1 in sshgr case in Figure 19). This point yields a pure negative exponential growth pattern. On the other hand, second unsatisfactory result was (3.3 ; 1.0 ; 2.8), which yielded two consecutive growth-and-decline patterns. In the latter case, pattern identification algorithm captured the

mismatch between the desired and produced pattern, but optimization algorithm locks-in to the point X5 in 1 of 30 experiments. This indicates the need for extended experimentation with differing settings of the optimization algorithm. On the other hand, algorithm returned X1 in 3 of 30 runs. This may also be partially attributed to optimization algorithm, but it is also seen that discriminative power of the pattern identification algorithm is low between S-shaped growth and negative exponential growth patterns.

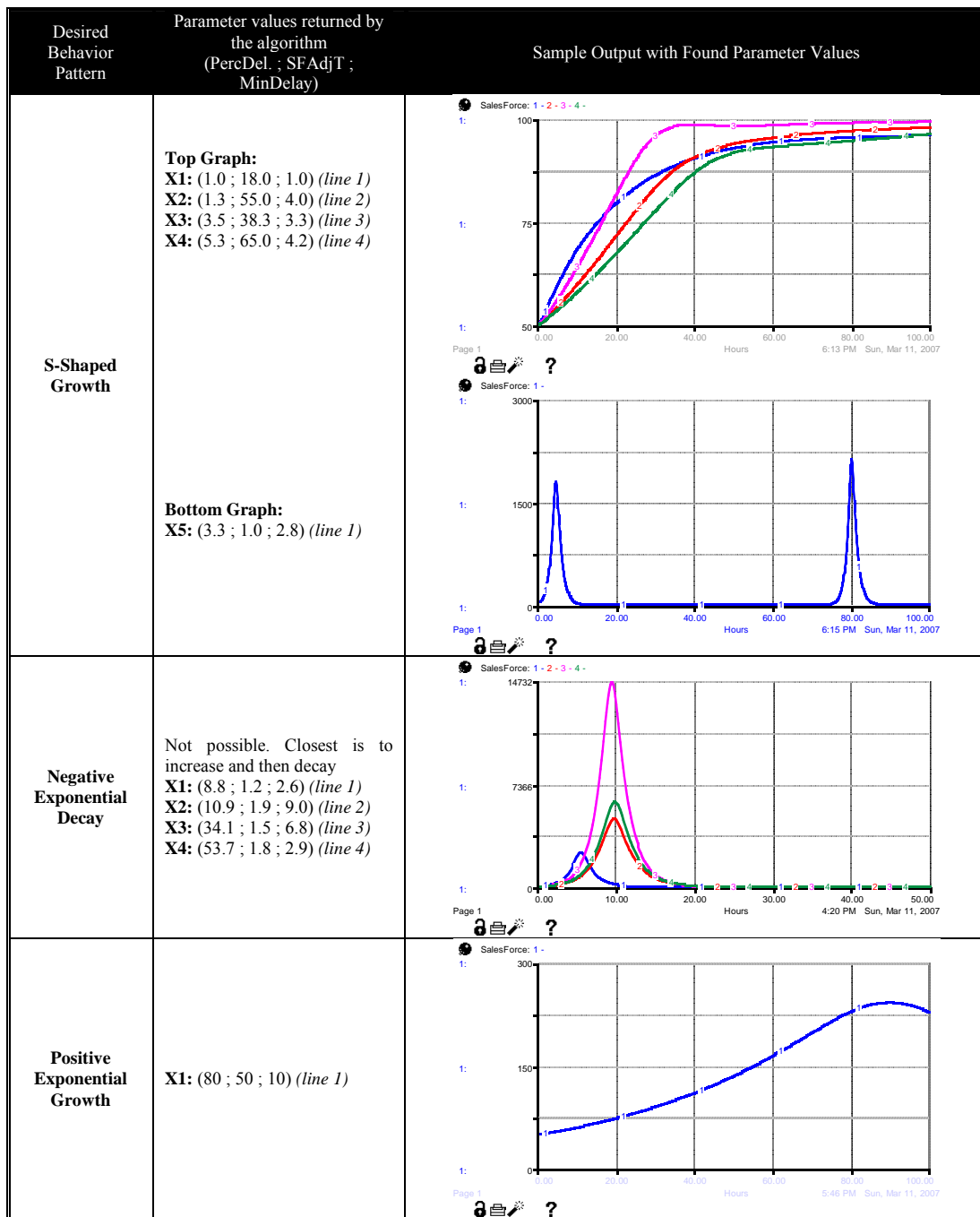


Figure 19: Results of the experiments with the market growth model (Search with 3 parameters)

CONCLUSIONS AND FURTHER RESEARCH:

A pattern-based parameter search/optimization method and the results obtained from various test experiments are presented in this paper. The conducted experimental study provides an idea about usability of the approach, and insights about future work that should follow. Our primary assessment about the proposed method is that it stands as a promising modeling support tool. In the majority of the experiments conducted, obtained results were satisfactory. Average run-time of the algorithm, which was around 4 minutes and the successful results support the method as an automated parameter search tool. Contrary to our expectations, time performance of the method was not influenced significantly by the number of model parameters used in the search process; there was only a slight increase in the run time of the algorithm. This observation indicates the viability of the proposed method as a support tool even for larger models.

Another advantage of the proposed method is that no reference time series is needed in order run our parameter search algorithm. The existing optimization applications aiming to optimize the model output, require data series representing the desired pattern. However, such a reference data series is may not be available and is not necessary in our case. The user has the opportunity to choose the desired pattern characteristics from a defined pattern template that covers most of the basic patterns that can be observed or targeted in SD modeling studies.

The proposed method has some limitations that should be improved. One of them is the lack of numeric features of the desired pattern in the objective function of the search algorithm. While the method searches for parameter values that yield desired pattern characteristics (e.g. goal-seeking growth), it does not pay attention to numeric aspects (e.g. growth stabilizing around level 300). This is one of the future research directions we intend to focus on.

Two basic future directions that will immediately follow this study are related to the optimization and pattern recognition algorithms. As discussed before GA has several design parameters (like population size, stopping criterion) that can influence the overall performance. Next phase of research focusing on these parameters is currently being designed. In this process, it is hoped that further improvements in the performance of the algorithm will be obtained.

Finally, another future work is to convert this method to a standalone support application that can work with a set of existing SD simulation software. Wide spread utilization of such a promising tool may be possible only in such a form.

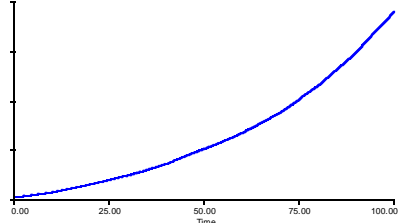
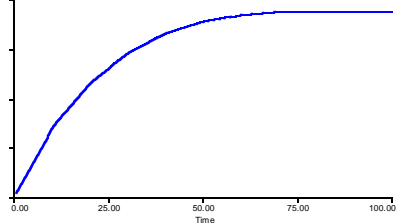
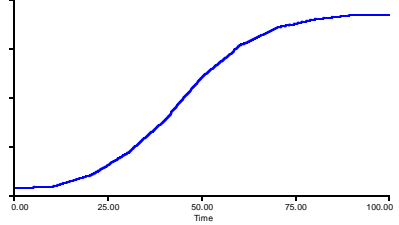
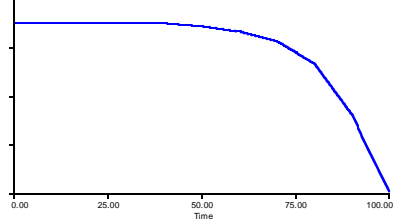
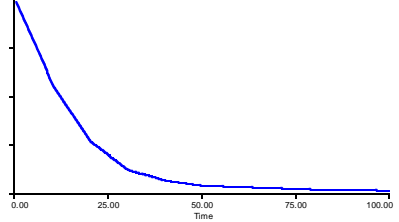
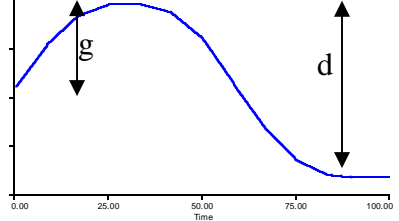
REFERENCES:

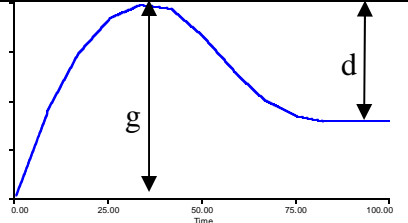
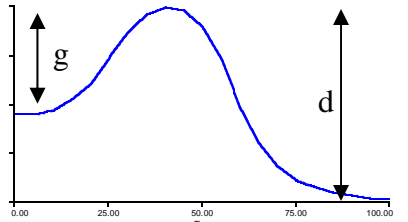
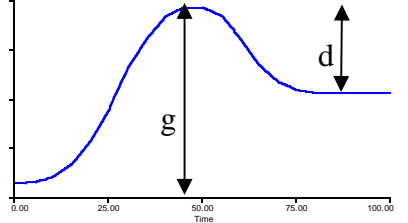
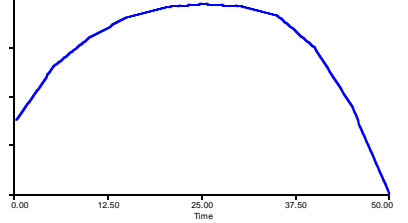
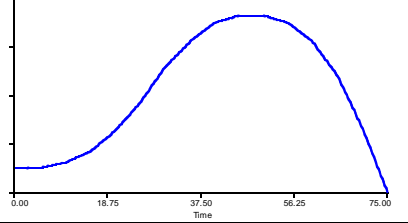
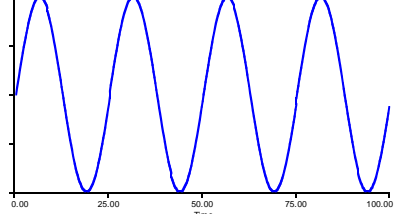
- Barlas, Y. (1996). "Formal aspects of model validity and validation in system dynamics." *System Dynamics Review* **12**(3): 183-210.
- Boğ, S. and Y. Barlas (2005). Automated dynamics pattern testing, parameter calibration and policy improvement. International System Dynamics Conference, Boston.
- Coveney, P. and R. Highfield (1995). Frontiers of complexity: the search for order in a chaotic world. New York, Ballantine Books.

- Coyle, G. (2000). "Qualitative and quantitative modelling in system dynamics: some research questions." System Dynamics Review **16**(3): 225-244.
- Dangerfield, B. and C. Roberts (1999). "Optimization as a statistical estimation tool: An example in estimating the AIDS treatment-free incubation period distribution." System Dynamics Review **15**(3): 273-291.
- Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, Addison-Wesley.
- Graham, A. K., J. D. W. Morecroft, et al. (1992). "Model-supported case studies for management education." European Journal of Operational Research **59**: 151-166.
- Holland, J. H. (1995). Hidden order : how adaptation builds complexity. Reading, Mass., Addison-Wesley.
- Kanar, K. (1999). Structure-oriented Behavior Tests in Model Validation. Department of Industrial Engineering. Istanbul, Boğaziçi University. **MSc**.
- Kanar, K. and Y. Barlas (*Under revision*). "Automated dynamic pattern recognition and testing." System Dynamics Review.
- Keane, A. J. (1996). A brief comparison of some evolutionary optimization methods. Modern Heuristic Search Methods. V. J. Raynard-Smith, I. H. Osman, C. R. Reeves and G. D. Smith. Chichester, John Wiley & Sons Ltd: 255-272.
- Keloharju, R. and E. F. Wolstenholme (1988). "The basic concepts of system dynamics optimization." Systems Practice **1**(1): 65-86.
- Miller, J. H. (1998). "Active Nonlinear Tests (ANT) of Complex Simulation Models." Management Science **44**(6): 820-830.
- Raynard-Smith, V. J., I. H. Osman, et al. (1996). Modern Heuristic Search Methods. Chichester, John Wiley & Sons Ltd.
- Richardson, G. P. (1999). "Reflections for the future of system dynamics." Journal of the Operational Research Society **50**(4): 440-449.
- Soylu, S. (2006). Generic Dynamic Patterns: Testing by Empirical Evidence. Industrial Engineering Dept. Istanbul, Bogazici University. **MSc**.

APPENDIX A:

Behavior patterns used in experiments (adopted from Bog and Barlas, 2005)

Pattern Code	Description	Sample Pattern
pexgr	Positive exponential growth	
nexgr	Negative exponential growth	
sshgr	S-shaped growth	
pexdc	Positive exponential decline	
nexdc	Negative exponential decline	
grlda	Growth with decreasing rate followed by decline to equilibrium (growth level g is less than decline level d)	

gr1db	Growth with decreasing rate followed by decline to equilibrium (growth level g is greater than decline level d)	
gr2da	S-shaped growth with decreasing rate followed by decline to equilibrium (growth level g is less than decline level d)	
gr2db	S-shaped growth with decreasing rate followed by decline to equilibrium (growth level g is greater than decline level d)	
g1ped	Growth with decreasing rate followed by positive exponential decline	
g2ped	S-shaped growth followed by positive exponential decline	
oscct	Oscillations around constant mean	

APPENDIX B:
Supplementary Material for the Thermostat Model

1. Differential Equation Set:

$$\frac{dS}{dt} = \frac{1}{T_{Adj}} \times (Goal - PS)$$

$$\frac{dPS}{dt} = \frac{1}{T_{Perc}} \times (S - PS)$$

where;

S: State

PS: Perceived State

T_{Adj} : Adjustment Time

T_{Perc} : Perception Delay

2. Analytical Solution for Characteristics Equation and Eigenvalues

Since the model is simple, it is possible to check the analytical solution in order to have an idea about the type of behavior to expect under certain parameter conditions. When goal in the model is set to zero without losing generality, we get the following matrix representation for the system of differential equations;

$$\begin{bmatrix} \dot{S} \\ \dot{PS} \end{bmatrix} = \begin{bmatrix} 0 & \frac{-1}{T_{Adj}} \\ \frac{1}{T_{Perc}} & \frac{-1}{T_{Perc}} \end{bmatrix} \cdot \begin{bmatrix} S \\ PS \end{bmatrix}$$

Based on this matrix, roots of the characteristics equation are calculated as follows;

$$A - \lambda I = \begin{bmatrix} -\lambda & \frac{-1}{T_{Adj}} \\ \frac{1}{T_{Perc}} & \frac{-1}{T_{Perc}} - \lambda \end{bmatrix} \longrightarrow |A - \lambda I| = \lambda^2 + \frac{1}{T_{Perc}} \cdot \lambda + \frac{1}{T_{Adj} \cdot T_{Perc}}$$

$$r_{1,2} = \frac{-b \mp \sqrt{b^2 - 4ac}}{2a} \longrightarrow \lambda_{1,2} = \frac{\frac{-1}{T_{Adj}} \mp \sqrt{\frac{1}{T_{Perc}^2} - 4 \cdot \frac{1}{T_{Adj} \cdot T_{Perc}}}}{2}$$

Since T_{Adj} is positive by definition, this model cannot generate any unstable behavior patterns. Apart from that it can also be verified from the equation above that model will demonstrate stable oscillations when $T_{Adj} < 4.T_{Perc}$.

3. Model Equations from STELLA®

Perceived_State(t) = Perceived_State(t - dt) + (Perception_Adjustment) * dt

INIT Perceived_State = -10

Perception_Adjustment = (-Perceived_State+State)/Perception_Delay

State(t) = State(t - dt) + (State_Adjustment) * dt

INIT State = -10

State_Adjustment = (Goal-Perceived_State)/Adjustment_Time

Adjustment_Time = 2

Goal = 0

Perception_Delay = 1

APPENDIX C:
Supplementary Material for the Stock Adjustment Model

1. Differential Equation Set:

$$\frac{dSL_1}{dt} = \frac{1}{T_{Adj}} \cdot (DS - S) - \frac{1}{T_{Acq}} \cdot SL_1 + L$$

$$\frac{dSL_2}{dt} = \frac{1}{T_{Acq}} \cdot SL_1 - \frac{1}{T_{Acq}} \cdot SL_2$$

$$\frac{dS}{dt} = \frac{1}{T_{Acq}} \cdot SL_2 - L$$

where;

$SL_{1,2}$: Supply Line 1 and 2

S : Stock

DS : Desired Stock

T_{Acq} : Acquisition Delay

T_{Adj} : Adjustment Time

L : Loss

2. Analytical Solution for Characteristics Equation and Eigenvalues

When *Loss* and *Desired Stock* parameters in the model is set to zero without losing generality, we get the following matrix representation for the system of differential equations;

$$\begin{bmatrix} \dot{SL}_1 \\ \dot{SL}_2 \\ \dot{S} \end{bmatrix} = \begin{bmatrix} -\frac{1}{T_{Acq}} & 0 & \frac{-1}{T_{Adj}} \\ \frac{1}{T_{Acq}} & \frac{-1}{T_{Acq}} & 0 \\ 0 & \frac{1}{T_{Acq}} & 0 \end{bmatrix} \cdot \begin{bmatrix} SL_1 \\ SL_2 \\ S \end{bmatrix}$$

Corresponding characteristics equation is as follows;

$$\lambda^3 + \frac{2}{T_{Acq}} \lambda^2 + \frac{1}{T_{Acq}^2} \lambda + \frac{1}{T_{Adj} \cdot T_{Acq}^2} = 0$$

Without solving for exact roots of this equation, we can identify the relation between parameters that yield oscillatory behavior by using the Δ given below;

$$\Delta = 4b^3d - b^2c^2 + 4ac^3 - 18abcd + 27a^2d^2$$

where a is the coefficient of the highest degree term, and d is the coefficient of lowest degree term in characteristics equation. For $\Delta > 0$, the characteristics equation will have one real and two complex conjugate roots. Then;

$$\frac{T_{Acq}}{T_{Adj}} > \frac{4}{27} \Rightarrow \text{One real and a pair of complex conjugate roots}$$

3. Model Equations from STELLA®

$$\text{Stock}(t) = \text{Stock}(t - dt) + (\text{Acquisition2} - \text{Loss}) * dt$$

$$\text{INIT Stock} = 20$$

$$\text{Acquisition2} = \text{Supply_Line_2} / \text{Acquisition2_Delay}$$

$$\text{Loss} = 10$$

$$\text{Supply_Line_1}(t) = \text{Supply_Line_1}(t - dt) + (\text{Control} - \text{Acquisition_1}) * dt$$

$$\text{INIT Supply_Line_1} = \text{Acquisition2_Delay} * \text{Loss}$$

$$\text{Control} = \text{Loss} + \text{Stock_Adjustment}$$

$$\text{Acquisition_1} = \text{Supply_Line_1} / \text{Acquisition1_Delay}$$

$$\text{Supply_Line_2}(t) = \text{Supply_Line_2}(t - dt) + (\text{Acquisition_1} - \text{Acquisition2}) * dt$$

$$\text{INIT Supply_Line_2} = \text{Acquisition2_Delay} * \text{Loss}$$

$$\text{Acquisition_1} = \text{Supply_Line_1} / \text{Acquisition1_Delay}$$

$$\text{Acquisition2} = \text{Supply_Line_2} / \text{Acquisition2_Delay}$$

$$\text{Acquisition1_Delay} = 5$$

$$\text{Acquisition2_Delay} = 5$$

$$\text{Desired_Stock} = 50$$

$$\text{Stock_Adjustment} = (\text{Desired_Stock} - \text{Stock}) / \text{Stock_Adjustment_Time}$$

$$\text{Stock_Adjustment_Time} =$$

APPENDIX D:
Supplementary Material for the Market Growth Model

1. Model Equations from STELLA®

$$\text{Backlog}(t) = \text{Backlog}(t - dt) + (\text{New_Orders} - \text{Completed_Orders}) * dt$$

$$\text{INIT Backlog} = 8000$$

$$\text{New_Orders} = \text{OrdersBooked}$$

$$\text{Completed_Orders} = \text{BacklEffect} * \text{NormCompl}$$

$$\text{Perc_DeliveryDelay}(t) = \text{Perc_DeliveryDelay}(t - dt) + (\text{Perc_Adjustment}) * dt$$

$$\text{INIT Perc_DeliveryDelay} = \text{DeliveryDelay}$$

$$\text{Perc_Adjustment} = (\text{DeliveryDelay} - \text{Perc_DeliveryDelay}) / \text{Perception_Delay}$$

$$\text{SalesForce}(t) = \text{SalesForce}(t - dt) + (\text{Hiring}) * dt$$

$$\text{INIT SalesForce} = 10$$

$$\text{Hiring} = (\text{SalesForceBudget} / \text{Salary} - \text{SalesForce}) / \text{SF_Adjustment_Time}$$

$$\text{BacklRatio} = \text{Backlog} / \text{NormBacklog}$$

$$\text{DelDelRatio} = \text{Perc_DeliveryDelay} / \text{Min_DeliveryDelay}$$

$$\text{DeliveryDelay} = \text{Backlog} / \text{Completed_Orders}$$

$$\text{Min_DeliveryDelay} = 1$$

$$\text{NormBacklog} = 25000$$

$$\text{NormCompl} = 10000$$

$$\text{OrdersBooked} = \text{SalesForce} * \text{SF_Effectiveness}$$

$$\text{Perception_Delay} = 8$$

$$\text{RevenuePerOrder} = 10$$

$$\text{Salary} = 2000$$

$$\text{SalesForceBudget} = \text{OrdersBooked} * \text{RevenuePerOrder}$$

$$\text{SF_Adjustment_Time} = 20$$

$$\text{SF_BaseEffectiveness} = 100$$

$$\text{SF_Effectiveness} = \text{Deleffect} * \text{SF_BaseEffectiveness}$$

$$\text{BacklEffect} = \text{GRAPH}(\text{BacklRatio})$$

$$(0.00, 0.02), (0.333, 0.4), (0.667, 0.73), (1.00, 1.00), (1.33, 1.24), (1.67, 1.47), (2.00, 1.64), (2.33, 1.75), (2.67, 1.85), (3.00, 1.92), (3.33, 1.96), (3.67, 1.98), (4.00, 2.00), (4.33, 2.00), (4.67, 2.00), (5.00, 2.00)$$

$$\text{Deleffect} = \text{GRAPH}(\text{DelDelRatio})$$

$$(0.00, 4.00), (0.5, 4.00), (1.00, 3.98), (1.50, 3.88), (2.00, 3.70), (2.50, 3.44), (3.00, 3.02), (3.50, 2.52), (4.00, 2.00), (4.50, 1.46), (5.00, 1.02), (5.50, 0.7), (6.00, 0.54), (6.50, 0.44), (7.00, 0.38)$$