Behavior Analysis and Testing Software (BATS)

Can Sücüllü
Boğaziçi University, Industrial Engineering Department
34342 Bebek Istanbul Turkey
+90 212 359 73 43
can.sucullu@boun.edu.tr

Gönenç Yücel
Boğaziçi University, Industrial Engineering Department
34342 Bebek Istanbul Turkey
+90 212 359 45 75
gonenc.yucel@boun.edu.tr

Abstract

Analysis of model behavior is mainly conducted in a pattern-based manner in system dynamics (SD) methodology. In pattern-based evaluation of model outputs, similarity of the overall behavior pattern (e.g. S-shaped-growth, oscillations) and of specific pattern characteristics (e.g. inflection points, periods, amplitudes) are more important than point-by-point similarity measures such as sum-of-squared errors. Although some output analysis tools/software that address this special pattern focus are available, they lack usability and are fragmented. In this study, new standalone analysis software, namely Behavior Analysis and Testing Software (BATS), is developed. It integrates a pattern classification algorithm and a set of statistical methods for analysis of steady-state behaviors. Apart from enabling comparison of behaviors with these algorithms/methods, BATS includes structured processes that enable user to conduct automated hypothesis testing, behavior space exploration, and sensitivity analysis. In its current state, BATS can seamlessly communicate with SD modeling software (Vensim) and other common data sources. This study provides illustrative examples of how BATS can assist the modeler and/or analyst in various phases of modeling; indirect structure testing, output evaluation, sensitivity analysis, policy analysis. Considering its pattern-orientation, user-friendly interface, and communication with modeling software BATS can be an important contribution to the analysis toolset of SD methodology.

Keywords: system dynamics, model analysis, support tool, behavior classification, validation, sensitivity analysis, software implementation

Word count: 4980

Introduction

One of the key characteristics of the system dynamic approach that distinguishes it from other simulation-based approaches is the emphasis on dynamic model behavior, rather than events or system states at specific time points. Dynamic behavior of a model can be described as the output patterns that are generated by the operation of model variables over simulated time. Behavior analysis is encountered in various stages of a modeling study. In model construction, it is important for parameter estimation and calibration of the model according to the historical data or some desired behavior. Model validation can be discussed in two categories; structural validity and behavior pattern validity (Barlas, 1996). In structural validity there are certain tests that involve simulation (e.g., extreme condition tests), in which comparing the model performance with the hypothesized real behavior is essential. In behavior pattern comparison, the similarity of pattern related characteristics such as trends, means, periods, amplitudes between model output and real life data is important. In sensitivity analysis, there is a need for categorization of large numbers of model outputs with respect to certain rules and purposes (e.g., find the parameter range that eliminates oscillations). Finally, in policy analysis and design, the outputs are analyzed subject to certain policy objectives based on pattern related information. Table 1 summarizes these stages and typical behavior related questions that characterize each one.

Table 1. Stages of system dynamics methodology that involve output evaluation.

Task	Purpose	Example Question
Model Calibration & Parameterization	Fine tune the model equations/parameters to match/fit the historical time-series	What is the best value of my parameter x in order my model to best fit the historical behavior?
Model Validation and Testing	Gain confidence on model structure and its behavior mode by passing various validation tests	Does the model generate similar results with the expected behavior of the real system under the same extreme conditions?
Behavior Sensitivity Analysis	Identify different dynamic pattern modes and behavior sensitivity of the model	What kinds of behavior modes does the model generate within the feasible range of parameters?
Policy Analysis and Design	Design and test alternative policies under various scenarios	Which policy, represented by a set of parameters, yields the most robust behavior?

Owing to the long-term policy orientation of system dynamics modeling, the task of dynamic behavior analysis or model output evaluation necessitates special care. It should be done in a pattern-oriented manner, i.e. the primary concern is on pattern related measures. For example, whether the behavior is an s-shaped-growth pattern or not is more important than the exact value at a certain point in time. In the absence of formal methods for pattern evaluation, all the aforementioned stages of a modeling study that require output evaluation are prone to being subjective and qualitative. This necessitates development of formal output analysis methods in order to provide stronger basis for SD studies. This need has also been discussed by several authors (Richardson, 1996; Barlas, 1996, 2007; Sterman, 2000). Moreover, a set of attempts has been made towards developing such methods, tools and procedures (e.g. Forrester and Senge 1990; Barlas 1996; Ford and Flynn, 2005; Kampmann and Oliva, 2006; Phaff,

2006; Yücel and Barlas 2011) but a brief literature review reveals that they are utilized at a much less then desired or deserved level in practice.

The aforementioned underutilization can be attributed to several factors one of which is the difficulty of conducting the particular analysis method for SD practitioners. This study primarily aims to address this issue. Based on a review of SD literature, we compile a set of tools/methods that have been proposed for pattern-oriented output analysis. Based on this compilation, we aim to develop a user-friendly environment that will enable SD practitioners to easily apply these formal methods during different stages of a modeling study.

Background and Objectives

SD model output is a typical time-series, i.e. collection of data points over a time interval. The straightforward approach to measuring similarity or dissimilarity of two time-series is to compare their single statistics such as means, variances, and final or cumulative values. Another approach might be to consider every pair of data points and obtain a time-series of functions of pairwise differences, which is the basis for point-based metrics such as sum of squared errors (SSE), mean squared error (MSE), R^2, and Theil statistics (Theil, 1966; Sterman, 1984). However these approaches have a potential of yielding incorrect results. For example, let us consider MSE, a frequently used statistical metric calculated by the formula: $MSE = \frac{1}{n}\sum(X_i - Y_i)^2$. The inappropriateness of arbitrarily using MSE for comparison of behavior patterns is illustrated in Figure 1. In this plot, an arbitrary negative exponential curve outperforms the model output when their proximities to reference oscillatory behavior are measured by MSE. Here, small phase and bias shift results in a large penalty due to the point-based orientation of the technique. However, from a pattern-based perspective it is visible that this model behavior has similar pattern characteristics with the actual data; they belong to same behavior class (constant oscillation) and have similar periods and amplitudes.

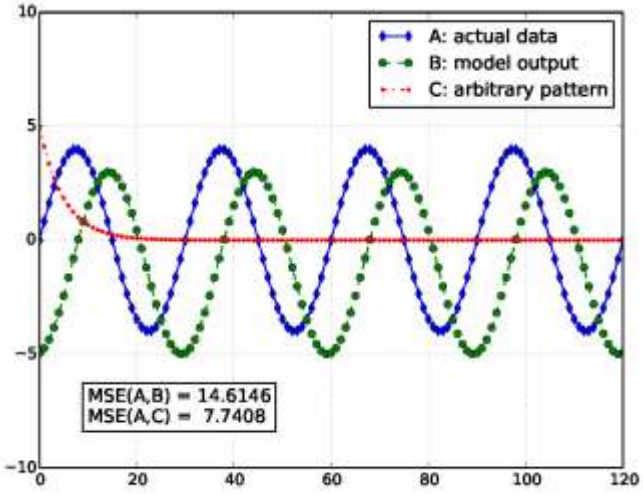


Figure 1. Illustrative example for showing the inappropriateness of MSE.

In order a formal approach to be relevant for the SD practice, it should have the capability to capture the pattern-wise similarity of model output to the actual output given in Figure 1. As discussed in the previous section, there are various attempts to this pattern-orientation issue in methodological SD model analysis literature. In this study, we focus on two of them; *multi-step procedure* for behavior comparison and *ISTS algorithm* for behavior classification. In the following sections we briefly describe these two previous studies, identify the problems encountered in model analysis practice, and propose motivations for this study.

Multi-Step Procedure for Behavior Comparison

In the previous section, it is illustrated that individual statistical methods are not suitable to be used directly for evaluation of dynamic behaviors. Nevertheless, with appropriate ordering and organization available statistical methods can be beneficial. An attempt to come up with a relevant set and logical ordering of methods has been made in a series of research, namely multi-step procedure (formerly 6-step) by Barlas (1996). This procedure contains the following functions and methods; trend regression, autocorrelation, spectral density, amplitude estimation by trigonometric function fitting and winters forecasting, calculation of mean and variance, cross correlation, and calculation of discrepancy coefficient. Interested reader may refer to the following studies that build on this procedure for further information; Barlas (1989, 1990, 1996); Barlas and Erdem (1994); Barlas et al. (1997); Bozyayla (2001). It is appropriate to note that the methods presented here are specifically relevant for comparison of steady-state periodic behaviors. They are originally developed for testing behavior validity of SD models, and for this specific purpose the logical ordering depicted in Table 2 can be followed. In addition to this usage principle, these methods can also be useful for other model analysis purposes such as model calibration, sensitivity analysis, and policy analysis since each individual step enables extracting information on pattern related components of a dynamic behavior.

Table 2. Multi-step procedure.

	MULTI-STEP PROCEDURE
1	Trend comparison and removal
2	Period comparison using autocorrelation and spectral density function
3	Autocorrelation test for lag comparison
4	Comparing the averages
5	Comparing the variations
6	Amplitude comparison
7	Testing phase lags using crosscorrelation function
8	Overall summary measures: coefficient of similarity

ISTS Algorithm for Behavior Classification

This algorithm was developed by (Kanar, 1999; Kanar and Barlas, 1999) originally for structure-oriented behavior testing (Barlas, 1996), especially for extreme-condition testing. It takes a time-series as input

and returns a table of likelihoods for similarity to generic behavior patterns that are frequently encountered in theory and practice. These patterns are from six behavior families; constant, growth, decline, growth-and-decline, decline-and-growth, and oscillatory. Including variations, the total number of behavior classes that can be recognized and classified by ISTS Algorithm is twenty-five. They are illustrated in Figure 2. Behavior classes are labeled, e.g. *plinr* represents positive linear growth (complete list of descriptions of these labels can be found in the appendix). The algorithm is a supervised one, namely it can recognize classes that it is previously trained to do so. In the training process, training set includes variants of the basic pattern class with noise and this set is used to estimate the transition probabilities that represent the class the best. The procedure is repeated for all basic behavior patterns to parameterize hidden Markov models.

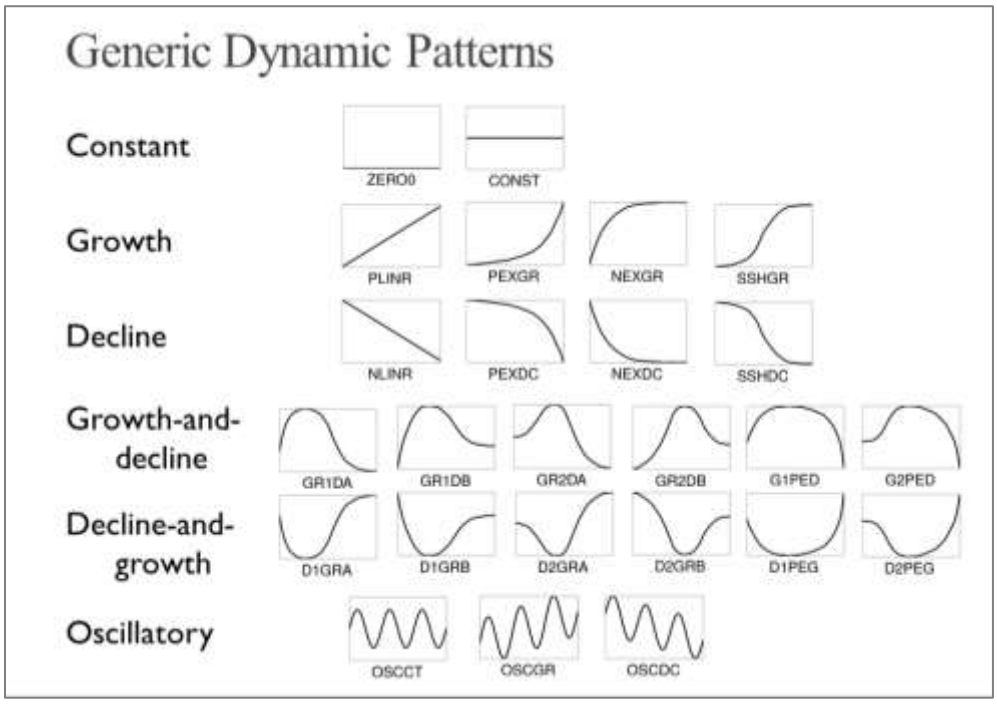


Figure 2. Generic dynamic patterns (Kanar, 1999).

In the classification process, the algorithm normalizes the time-series on y-axis, and splits it into six segments. Each segment is then characterized by a feature vector of mean, slope, and curvature. The fundamental principle is that the likelihood of a type of segment being followed by a particular segment type can be used to distinguish the generic behavior patterns. For example, in an exponential growth pattern, a segment with positive slope and positive curvature is most likely to be followed by another segment of the same character (positive slope and curvature). However, in an S-shaped growth pattern, the likelihood of the preceding section to have a positive slope but a negative curvature is significantly higher. Once the sequence of all segments is analyzed, state-optimized likelihoods for this particular sequence belongs to generic pattern classes are calculated and reported. This is done for all classes and twenty-five likelihoods are obtained. From those results, it is possible to extract meaningful information

on the most likely pattern class to which the input behavior belongs to. A pseudo-code of the algorithm is given in Figure 3. Interested reader may refer to Kanar and Barlas (1999), Kanar (1999), and Soylu (2006) for further information on both training and classification processes related to ISTS Algorithm.

```
Receive the time-series data, X, to be classified
Normalize X using min-max normalization procedure to obtain X_N
Split X_N into six segments of equal-length, and obtain S_1, S_2, ..., S_6
For each segment S_i {
  Calculate mean
  Calculate slope
  Calculate curvature
}
For all behavior classes B_j {
  Calculate the state-optimized likelihood of X_N belonging to the class B_j
}
Return the results for all behavior classes
```

Figure 3. ISTS Algoritm.

Problem Description

As discussed in the previous parts, a set of formal methods and procedures have been proposed to the field in order to satisfy the need for pattern-based output evaluation. Despite their availability, the utilization of these tools is very limited. In that respect, it is possible to discuss that there are some barriers between SD practitioners and these formal analysis methods and procedures.

Firstly, there is a usability issue related with these tools. They are fragmented, separated, and lack organizations that brings them together. Some of these methods require familiarity with advanced software or programming environment such as Matlab (Yücel and Barlas, 2011) and Matematica (Kapmann and Oliva, 2006), which may lie outside the expertise of certain SD practitioners. It is also observed that these applications require several intermediary preparation/transformation steps in order to actually run the tool and get the results. This lack of automation makes the analysis process time-consuming, especially when lots of experiments are involved. The time-consuming problem is discussed in detail by several authors (Hearne, 1985; Drechsler, 1998; Miller, 1998).

The second issue is related with connectivity and compatibility. Lack of communication of the developed tool with modeling software (e.g. Stella, Vensim) brings forth burden to overall analysis process. The significance of this integration is mentioned in the literature (Barlas, 1996; Richardson, 1999; Ekşin, 2009; Yücel and Barlas, 2011). Establishing this connection eliminates the requirement for adoption of a new analysis language and possible transfer errors. It also enables to preserve SD specific modeling techniques, e.g. graphical/table functions and delay formulations, and maintains numerical consistency.

The third issue is the lack of user-friendliness, which is encountered in the available software development attempts. There are two software applications from the literature that we decide to mention in this context. The first one is Behavior Testing Software - *BTS* (Barlas et al., 1997) based on

multi-step procedure described above, and the second one is *SiS* (Boğ and Barlas, 2006) built upon ISTS Algorithm. As the modern software standards and design principles constantly develop and there is a necessity for the analysis software to follow them. The available projects in the literature fail to satisfy certain requirements such as efficient graphical user interface, multi-tasking, elaborate design, customization, cross-platform compatibility (i.e. works on Windows and Mac), being able to export quality graphs, and so on.

The problems mentioned above reveals that without achieving certain objectives it is highly unlikely for any support tool to reach a wider user base. In order to overcome these problems, we propose to design new analysis software that has:

- Structured and automated analysis processes
- Communication with SD modeling software and compatibility with standard data types
- User-friendly design for efficient human software interaction

In other words, it is aimed to design a usable software application for the modeler/analyst to perform pattern-oriented behavior analysis at various steps of a modeling study/project.

Behavior Analysis and Testing Software (BATS)

The software developed during this study, i.e. Behavior Analysis and Testing Software (BATS), is aimed to support evaluation, validation, sensitivity analysis and policy analysis processes in a SD study. It has pattern classification capabilities and consists of various statistical analysis tools for comparison of time-series. BATS is written on pure Python[1], which is an object-oriented, dynamically typed, high-level programming/scripting language. In its current form, BATS is standalone software that is delivered in single executable file. It naturally involves intense user interaction and this need is addressed by designing and building an effective user-friendly Matplotlib embedded wxPython based graphical user interface (GUI). BATS' GUI implements an IDE-style tabbed document interface framework, which enables a tabbed working environment that is widely used in new generation of software (e.g. Chrome, Firefox, Matlab, etc.). In addition, Matplotlib offers high-quality graphical output capabilities that allow print-quality exporting of results. Using its tabbed interface it is possible to work with multiple operations and customize screen views. BATS makes use of Numpy for algebraic and matrix computations. Numpy is the fundamental scientific computing package of Python which allows fast, convenient and efficient computation. BATS is extensible and designed for becoming a platform for future extensions. In Figure 3, several screen shots from BATS are illustrated in order to provide an idea about its interface to the reader.

As mentioned earlier, ability to communicate with potential data sources is an important requirement for software like BATS to be useful. In that respect, BATS is able to import external data from spreadsheets and text files, and moreover communicate directly with Vensim via Vensim DLL. The important contribution is that models developed on Vensim can be directly controlled with BATS by sending commands (e.g. set parameter value, run) and importing outputs. These outputs can be used for

[1] Available at http://www.python.org

all analysis features included in BATS. Lastly, it is possible to manually input time-series into BATS using its drawing pad feature.

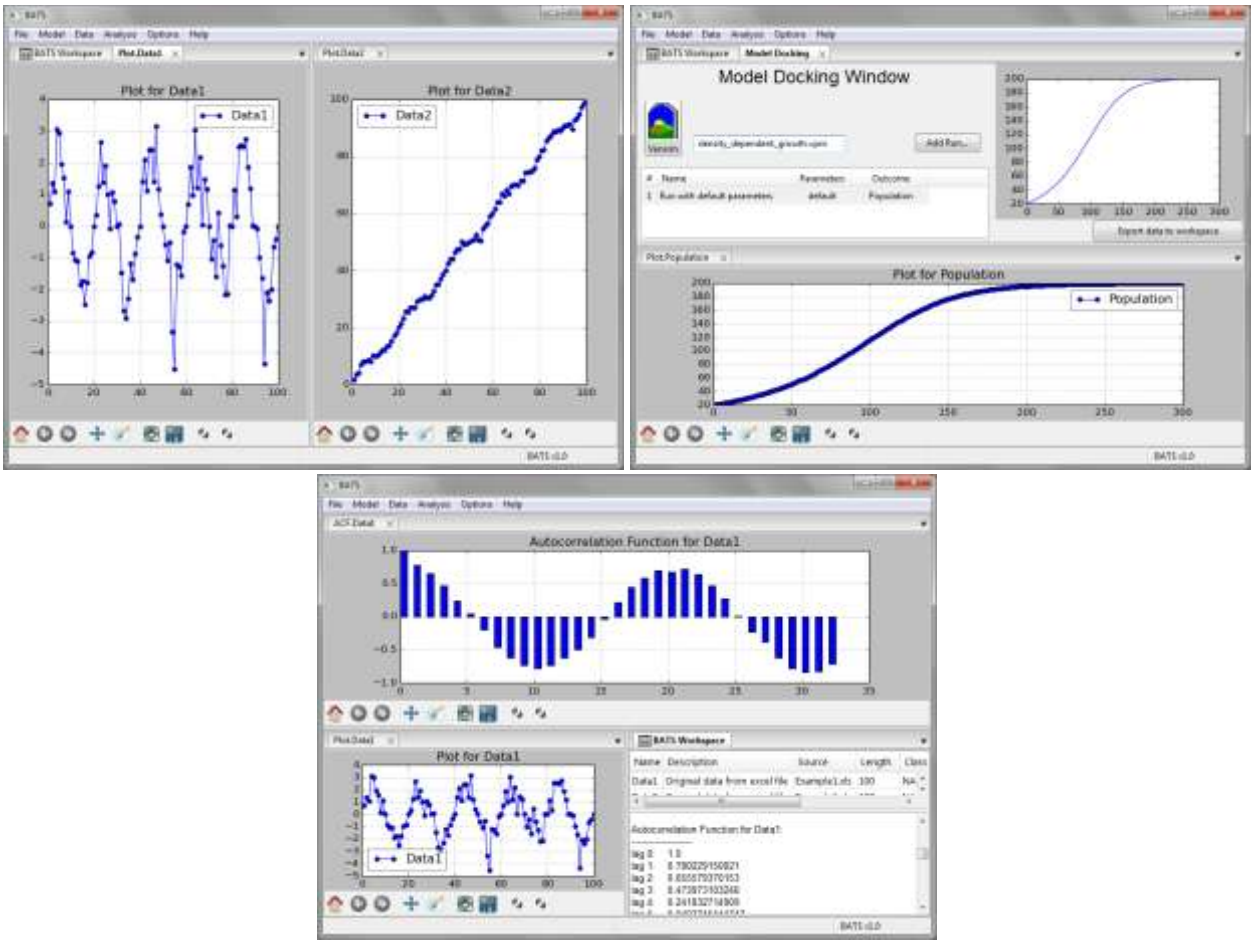


Figure 4. Custom screen views of BATS.

BATS covers functionalities of two aforementioned model analysis software applications (BTS and SiS) and also introduces new procedures and features. In summary, operations of BATS can be grouped under three categories; behavior input and preprocessing, behavior analysis, and model analysis. Behavior inputting features include reading from external files, drawing a new data, splitting/cropping existing data, extracting selections of data, applying exponential smoothing and moving average filtering techniques. Behavior analysis features include behavior classification, trend regression, autocorrelation, spectral densities, amplitude estimation, cross-correlation, summary statistics, and graphical comparison. Model analysis features include a general purpose model docking window, hypothesis tester, behavior space classifier, and behavior class mapper. In Figure 5, current features and functionalities of BATS are illustrated.

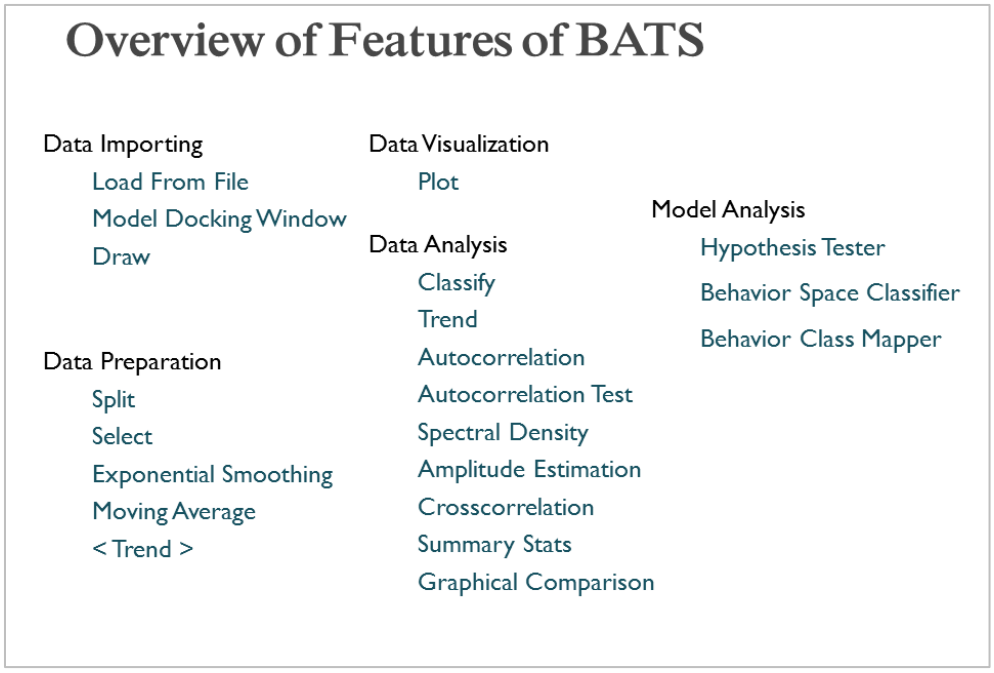


Figure 5. Overview of Features of BATS.

In the Figure 6, brief information on software architecture is provided. Different building blocks, their interaction and the flow of information between them, as well as communication with Vensim, Excel and *user* are compactly illustrated in this figure.

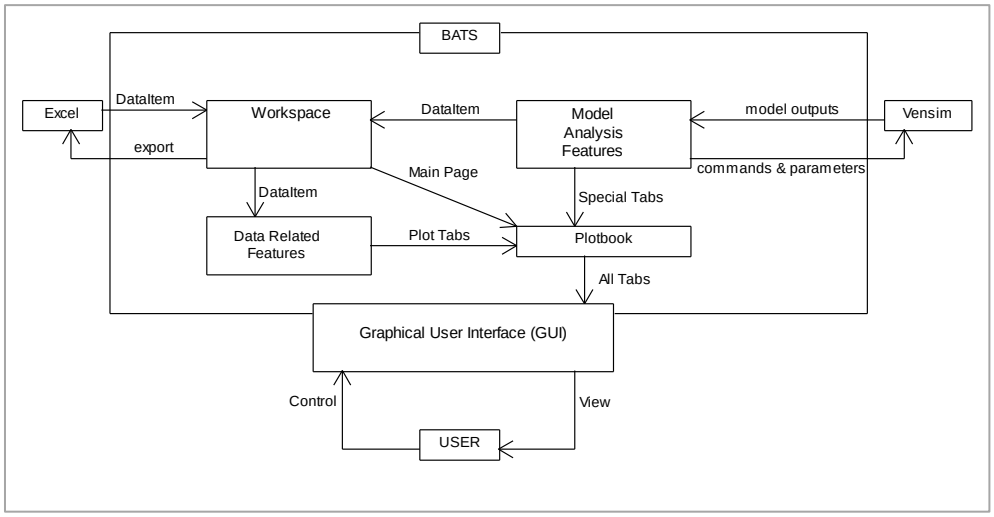


Figure 6. Communication capabilities with external software and internal operations of BATS.

In the following sections of this paper, alternative usage modes of BATS are demonstrated.

BATS for Structure-oriented Behavior Testing

Structure-oriented behavior tests are strong behavior tests that can provide information on potential structural flaws (Barlas, 1996). Extreme condition testing is one of the important and widely used techniques in this group of model evaluation methods. In order to demonstrate BATS in this usage mode, we use *density-dependent growth* model (Barlas, 2002, p. 22 and Sterman, 2000,p. 118). The stock-flow diagram of the model can be seen in Figure 7 and model equations can be seen in the appendix.

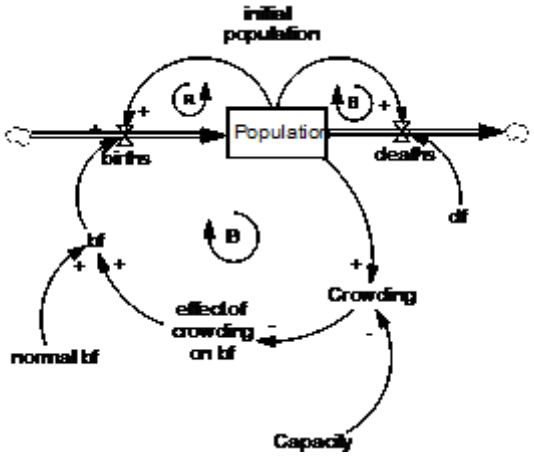


Figure 7. Stock-flow diagram of density-dependent growth model.

In order BATS' model analysis features to be utilized, as a preparatory step, the model developed on Vensim should be published as (.vpm) file. In this demonstration for structure-oriented behavior testing with BATS we use *Hypothesis Tester* feature (see Figure 8 for screenshot of *Model* menu group).

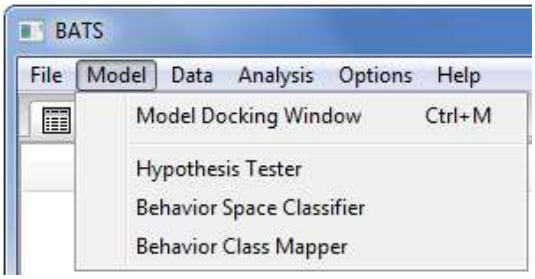


Figure 8. Screenshot of *Model* menu group.

After the feature is selected and the Vensim model is connected to BATS, the settings dialog box for the method appears (see Figure 9), in which test name, parameter name, parameter value, outcome variable of interest, and hypothesized behavior for the outcome variable are specified by the user.

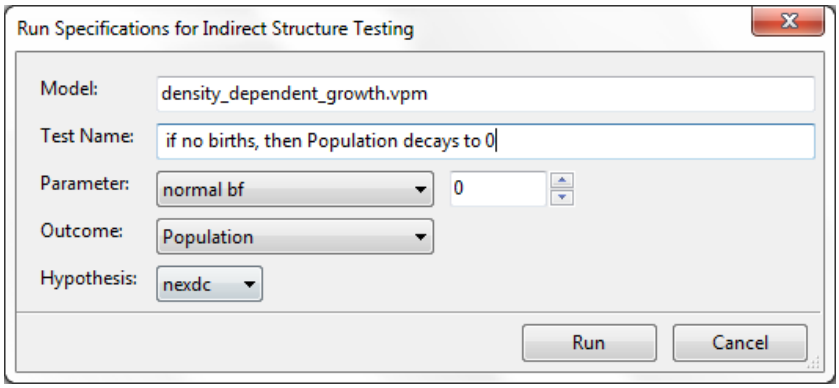


Figure 9. Settings dialog box for Hypothesis Tester feature.

Here in this example, the *nexdc* label represents a hypothesized behavior pattern of negative exponential decline (or goal-seeking decay) if the birth fraction is set to zero. Hypothesis Tester uses ISTS Algorithm and calculates likelihoods for all twenty-five behavior classes. If the likelihood value for the hypothesized behavior pattern is higher than the confidence level (currently set as -3), conclusion *PASSED* is filled in the corresponding test row. It is also possible to observe all the likelihood values by looking at their labels and exporting that particular simulation run to internal data keeping system for further analysis (see Figure 10).

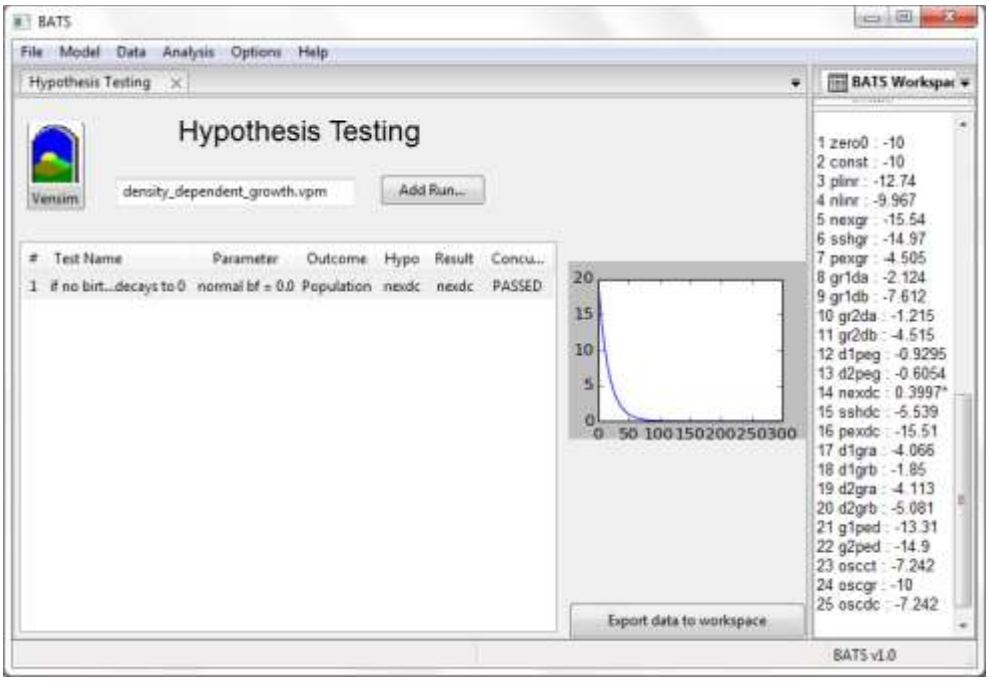


Figure 10. Results of the first hypothesis in Hypothesis Tester.

Hypothesis Tester stores the results of the previous tests, so that overall assessment of the structural validity of the model can be made in a user-friendly manner. A representative analysis session that involves six experiments is illustrated in Figure 11.

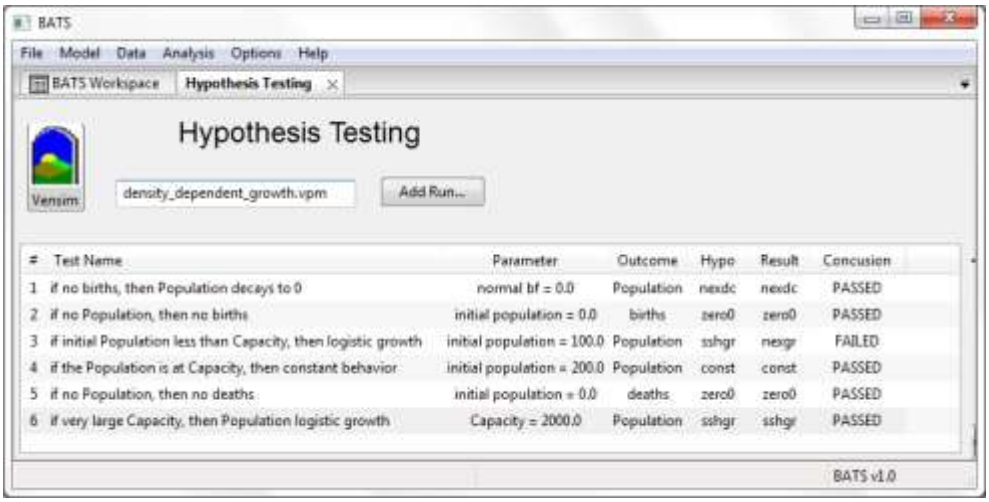


Figure 11. Screenshot for Hypothesis Tester after a representative extreme condition testing process.

BATS for Behavior Pattern Comparison

Behavior pattern comparison is conducted differently with for transient and steady state modes. For steady-state behaviors or parts of behaviors, the multi-step procedure can be utilized, whereas for transient behaviors graphical measures can be applied (Barlas, 1996). In this demonstration two time-series are imported from an Excel and a .txt file using Load from File option (see Figure 12 for screenshot of Data menu group).

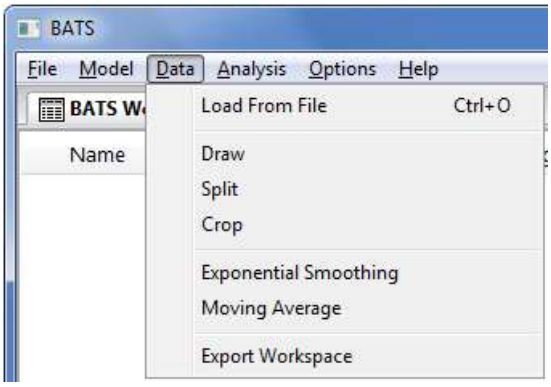


Figure 12. Screenshot for *Data* menu group.

The screenshot in Figure 13 displays the all behavior analysis features currently available in BATS. These menu items correspond to individual methods in aforementioned multi-step procedure and ISTS algorithm. In addition to them a new feature called graphical comparison is included, which enables manual comparison of behavior patterns.

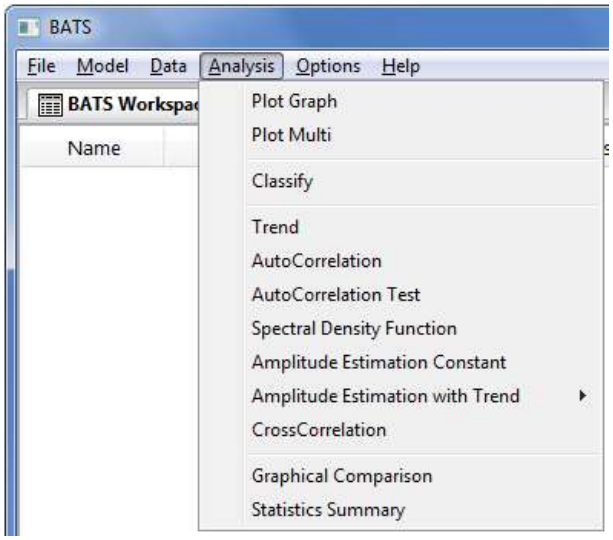
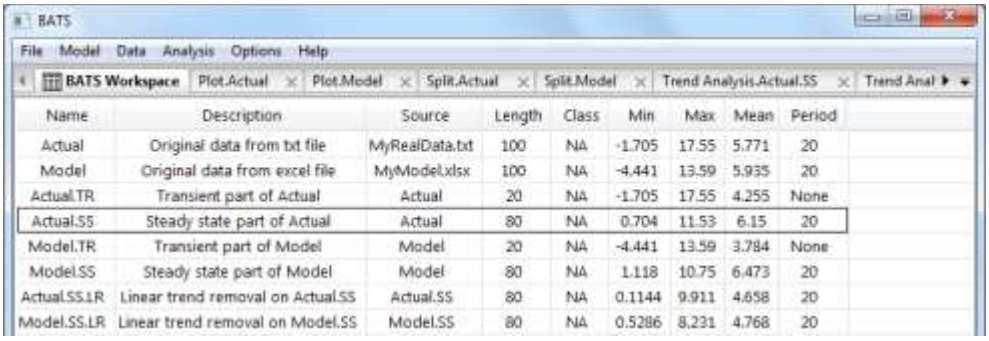


Figure 13. Screenshot for *Analysis* menu group.

Throughout this demonstration, the two time-series are referred as *actual* and *model*. In Figure 14, a screenshot of the workspace of BATS during this demonstration is pictured. The naming convention and outputs of operations can also be visualized in this figure. As the operations are held, new data series are created and stored, and for every item in the workspace statistical summary information is displayed in order to provide quick feedback to user.



Name	Description	Source	Length	Class	Min	Max	Mean	Period
Actual	Original data from txt file	MyRealData.txt	100	NA	-1.705	17.55	5.771	20
Model	Original data from excel file	MyModel.xlsx	100	NA	-4.441	13.59	5.935	20
Actual.TR	Transient part of Actual	Actual	20	NA	-1.705	17.55	4.255	None
Actual.SS	Steady state part of Actual	Actual	80	NA	0.704	11.53	6.15	20
Model.TR	Transient part of Model	Model	20	NA	-4.441	13.59	3.784	None
Model.SS	Steady state part of Model	Model	80	NA	1.118	10.75	6.473	20
Actual.SS.LR	Linear trend removal on Actual.SS	Actual.SS	80	NA	0.1144	9.911	4.658	20
Model.SS.LR	Linear trend removal on Model.SS	Model.SS	80	NA	0.5286	8.231	4.768	20

Figure 14. Workspace of BATS during behavior pattern comparison demonstration.

After successful import, the initial step is to visualize the data using the *Plot* feature (see Figure 15).

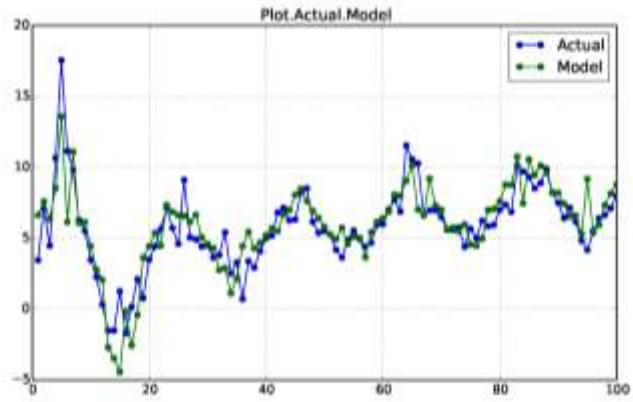


Figure 15. Two patterns that are used for behavior pattern testing.

It is observed that both of the two data series involve a transient phase. In order to analyze behavior patterns transient and steady-state parts should be analyzed separately (Barlas, 1996, p. 195). These two parts are separated using the *Split* feature and steady-state parts of the behavior patterns are obtained (the reader may refer to Figure 14 for updates in the workspace).

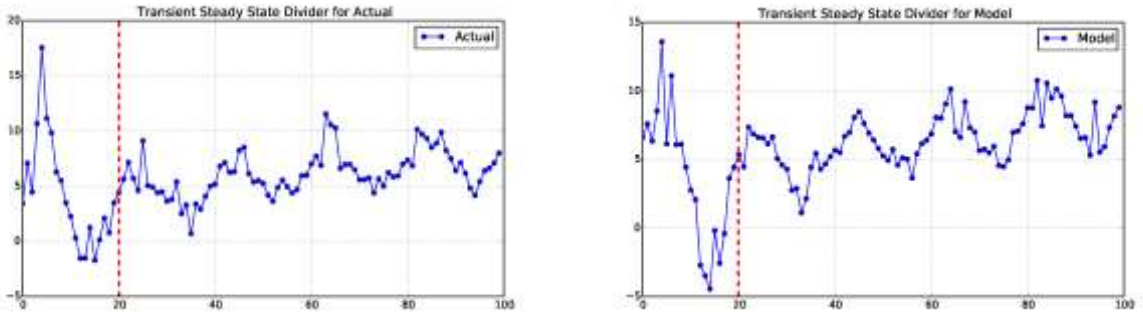


Figure 16. Transient steady-state behavior splitter tool.

After the separation of transient phases from the behavior patterns, it is now possible to conduct multi-step procedure in Table 2 (also see Barlas, 1996, p. 195). First step of the procedure is to estimate and compare trends (see Figure 17). The linear trend is estimated and removed by using the *Trend* feature, and trend related information (slope and intercept) is stored for comparison of patterns.

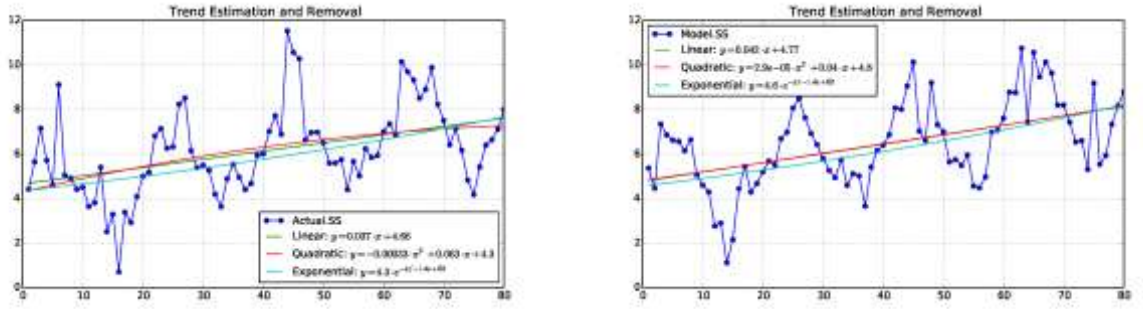


Figure 17. Trend estimation.

The final transformed states of the behavior patterns, after transient phase removal and de-trending, can be seen in Figure 18.

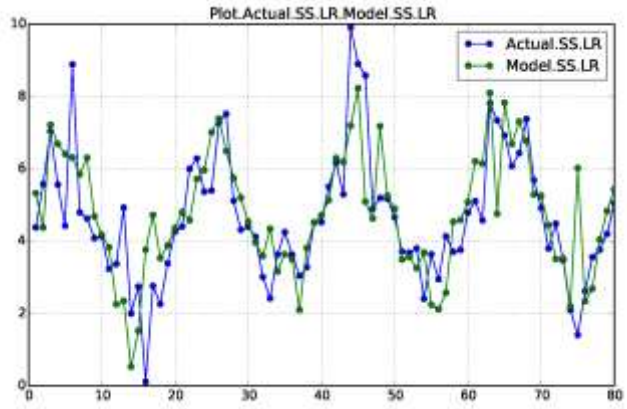


Figure 18. Plot of de-trended steady-state parts of behavior patterns.

Autocorrelation results show that both signals have a period of 20 (see Figure 19).

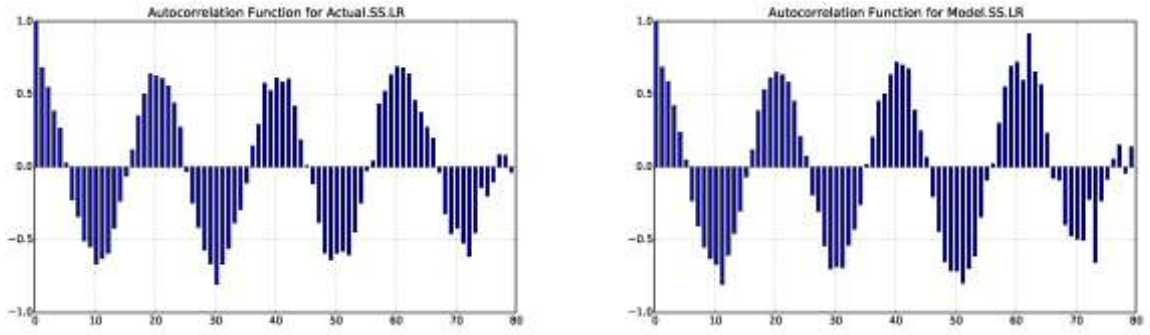


Figure 19. Autocorrelation function results.

Periods can also be estimated using spectral density function as in Figure 20. The results are in coherence with the results of autocorrelation method. The highest energy content is observed at lag 20, which indicates the primary period. In this demonstration the behavior patterns do not involve multiple periodicities so either autocorrelation or spectral density function can be utilized for period estimation. For estimation of periods from behavior that involve multiple periodicities spectral density function offers more extensive results (Bozyayla, 2001).

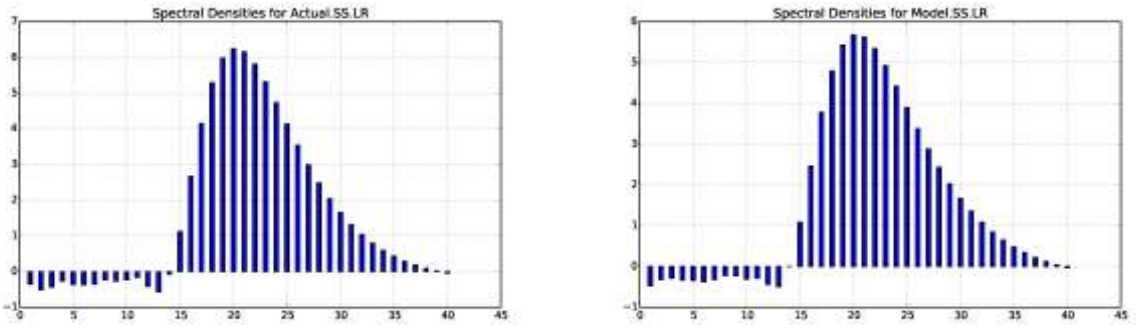


Figure 20. Spectral density function results.

Autocorrelation test constructs a confidence interval around differences of autocorrelation lags. In this example the differences lie inside the confidence region (see Figure 21). This indicates that autocorrelation patterns of these two time-series are quite similar, and there is nothing that suggests that they are different (for further information on this test the reader may refer to Barlas, 1990).

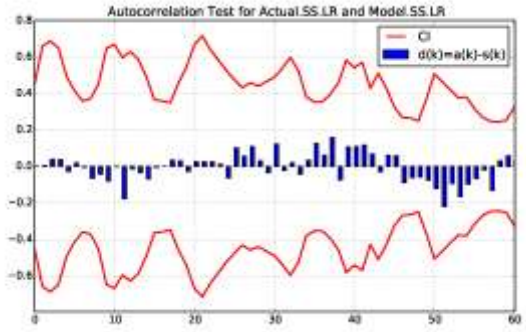


Figure 21. Autocorrelation test results.

The amplitudes are estimated using trigonometric function fitting. The amplitudes are found as 2.04 and 1.95 respectively. The values can be read from the plot and also from the log keeping system in BATS (see Figure 22).

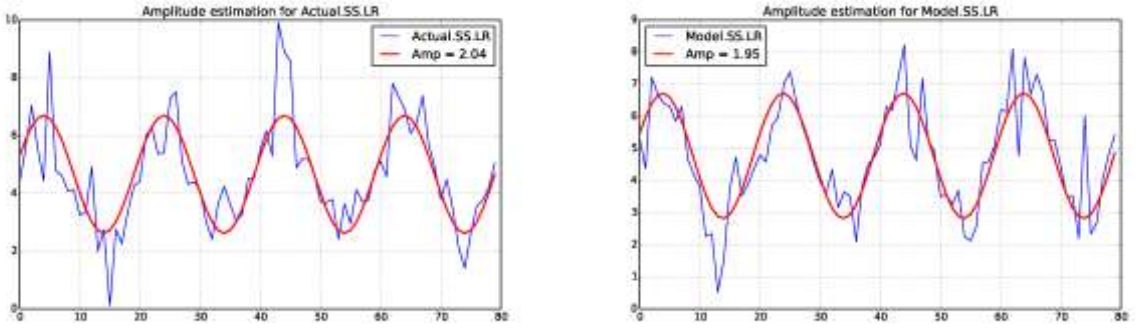


Figure 22. Amplitude estimation using trigonometric function fitting.

Cross-correlation function measures the phase shift between two signals. In this example, maximum value is observed at lag 0, which indicates that the signals are in phase (see Figure 23).

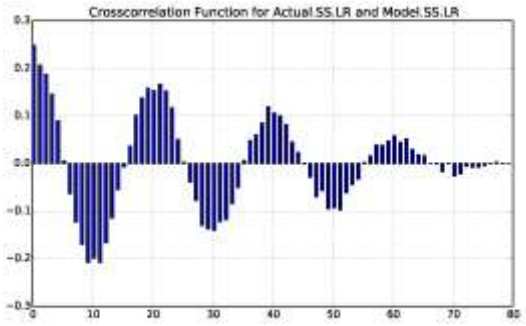


Figure 23. Cross correlation function results.

The means and variances are calculated using *Summary Statistics* feature, and reported to the user in the logging system (see Figure 24). This feature also calculates the coefficients of similarity, namely discrepancy coefficient (U) by Barlas (1989) and Theil statistics by Sterman (1984). U value takes values between 0 and 1, and rather large values are accepted if the other steps of the procedure designate that the behaviors are pattern-wise similar (0.3790 in this example).

Comparative Stats

	Actual.SS.LR	Model.SS.LR	Difference	Error:
Mean	4.657902	4.768117	-0.110216	-0.023662
Var	3.111435	2.670452	0.440983	0.141730

Single Stats

U	0.379063
MSE	1.671315
TheilM	0.007268
TheilS	0.010077
TheilC	0.982655

Figure 24. Summary statistics results.

In addition to application of multi-step procedure on the steady-state parts of the signals, the *Graphical Comparison* feature can be applied to transient parts. In this feature the user can manually indicate relevant custom key characteristics. These key characteristics can either be points or intervals, and either be on the x axis or y axis. In Figure 25, four different measures, namely *minimum level*, *maximum level*, *time between min and max*, and *overall change in level*, are calculated and compared. This new feature is implemented in BATS in order to enable performing flexible and manual comparison of pattern components such as maxima, minima, inflection points, equilibrium levels, durations, distances, and so on.

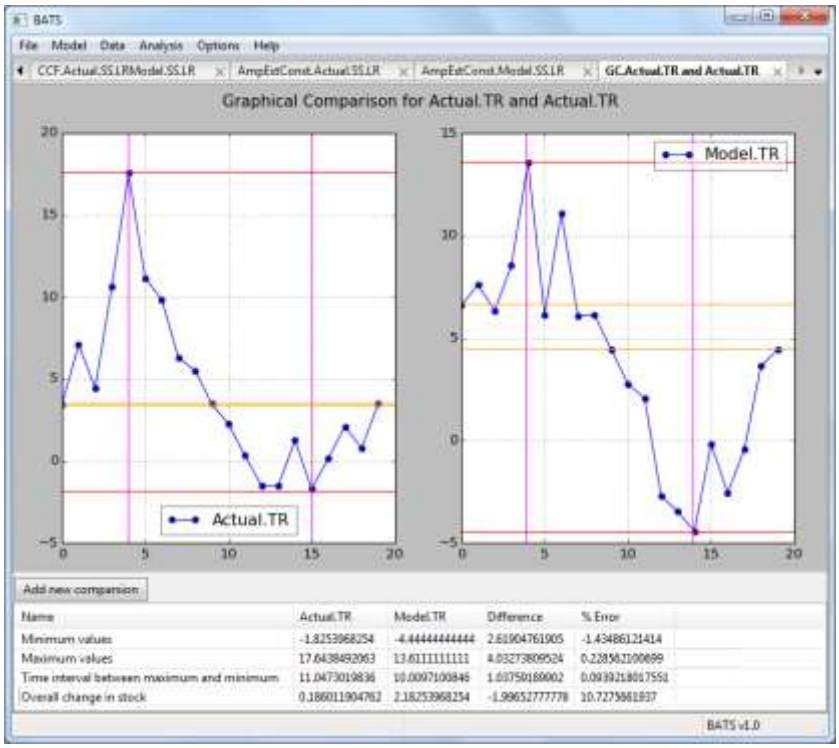


Figure 25. Comparison of transient parts of behaviors and illustration of Graphical Comparison feature.

The results of all operations used in this demostration are summarized in Table 3.

Table 3. Summary table for behavior validity pattern tests.

Metric	Actual	Model	Diff.	% Err.
Steady-state Part				
Linear trend slope	0.0370	0.0420	0.0050	0.1351
Primary period	20	20	0	0.0000
Autocorrelation test	Passed		-	
Mean	4.6579	4.7681	0.1102	0.0237
Variance	3.1114	2.6705	-0.4409	-0.1417
Amplitude	2.0400	1.9500	-0.0900	-0.0441
Phase Lag	0		-	
U	0.3791		-	
MSE	1.6713		-	
UM	0.0073		-	
US	0.0101		-	
UC	0.9826		-	
Transient Part				
Maximum	17.6438	13.6111	-4.0327	-0.2286
Minimum	-1.8254	-4.4444	-2.6190	1.4348
Time between max and min	11.0473	10.0097	-1.0376	-0.0939
Overall change	0.1860	2.1825	1.9965	10.7339

BATS for Sensitivity Analysis

In this demonstration, behavior sensitivity analysis of an SD model is analyzed using *Behavior Space Classifier* feature of BATS (see Figure 8 for the corresponding menu item). This feature finds behavior pattern sensitivity of the model with respect to changes in parameters. For this demonstration we use

the temperature adjustment model. This is a model for the stock control problem under the presence of two types of delays. The first one is the material delay and it represents the delay occurs due to the flow of hot/cold air in the air-conditioner system. The second one is the information delay and it represents the perception delay of the human being while feeling and measuring the actual temperature. The stock-flow diagram of the model can be seen in Figure 26.

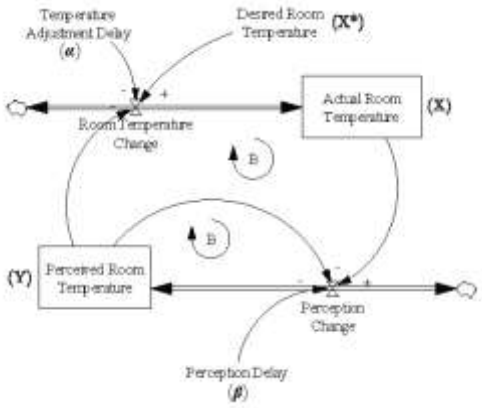


Figure 26. Stock-flow diagram of temperature adjustment model.

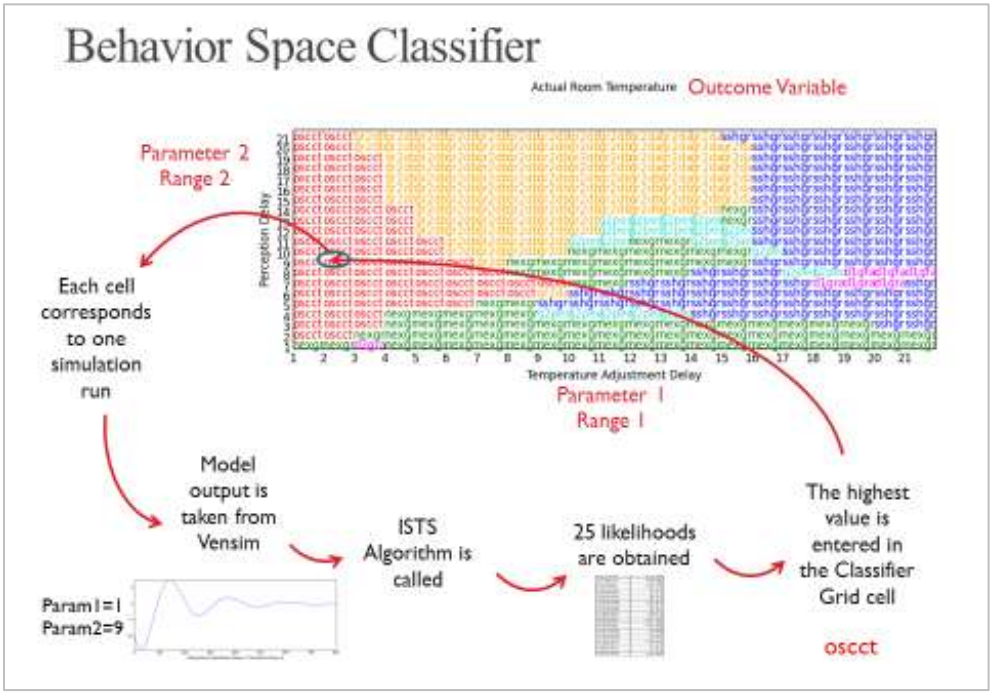


Figure 27. Workflow of Behavior Space Classifier.

Model file, output variable, sensitivity parameters and their ranges are collected as inputs from the user. Behavior Space Classifier produces a classifier grid drawn respect to sensitivity values of parameter 1 and parameter 2. In Figure 27 the workflow of this feature is illustrated. For each cell on the grid Vensim model is run with corresponding parameter values set and then the output is automatically obtained. For the model output ISTS Algorithm is called and all twenty-five likelihood values are calculated. The cell on

the grid is filled with the label of the behavior pattern class with the highest likelihood for that particular run. This process is automated, that is, the computations continue until all the feasible parameter range is searched and classified. The screenshot from the software can be observed in Figure 28. In this final plot, a coloring system is embedded to improve visual classification. In addition, the plot at the bottom is responsive, that is, as the user clicks a cell on the classifier grid, the plot updates to show the output of the corresponding run. In this figure, behavior when temperature adjustment delay is equal to 20 and perception delay is equal to 12 is illustrated (the run is also indicated by the red ellipse on the classifier grid). In this parameter combination, the highest likelihood value is from S-shaped growth pattern.

Overall examination of the colored classifier grid supports the modeler/analyst while assessing sensitivity of behavior modes as a result of numeric parameter value changes in the model. It can be seen from the results that there fundamentally exists four different behavior modes. When the material delay is low (i.e. fast temperature change), but the perception delay is high (i.e. slow perception change) the actual temperature exhibits oscillatory behavior (indicated by red *oscct* label on the grid). Under the opposite conditions when the material delay is high and perception delay is low the temperature smoothly reaches the desired equilibrium level (green *nexgr* label). Following the oscillatory case, as the material delay increases the oscillations get weaker and the behavior becomes more like growth then decline behavior (yellow *gr2db* label). Finally, when both material and information delays are high the behavior become more s-shaped growth alike (blue *sshgr* label). The results comply with the analytical solution of the model and with a previous study in the literature (Yücel and Barlas, 2011).

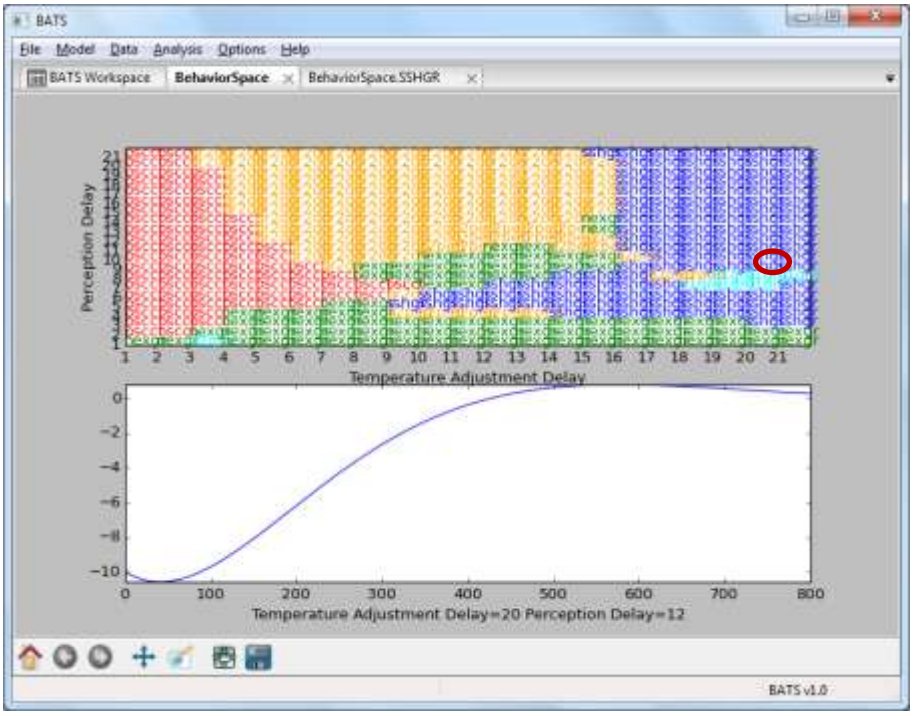


Figure 28. Screenshot for the classifier grid and model behavior plot of Behavior Space Classifier.

BATS for Policy Analysis

In this demonstration we suppose that the desired behavior for the *Actual Room Temperature* variable in the temperature adjustment model is an *S-shaped growth* pattern. As in the previous example, there are two decision parameters of sensitivity, namely *Temperature Adjustment Delay* and *Perception Delay*. The feasible ranges for these parameter are determined as [1,21]. The modeler is interested in changes in "s-shaped growthness" of model behavior with respect to parameter changes. In order to perform this task, *Behavior Class Mapper* feature of BATS is used (see Figure 8 for the corresponding menu item). This feature provides a method for assessing changes in the behavior mode of the model as a result of changes in its parameters. In the preparation step the user chooses a specific behavior pattern such as sshgr (s-shaped growth), as well as parameters of uncertainty and their ranges and the output variable of interest (actual room temperature in this case).

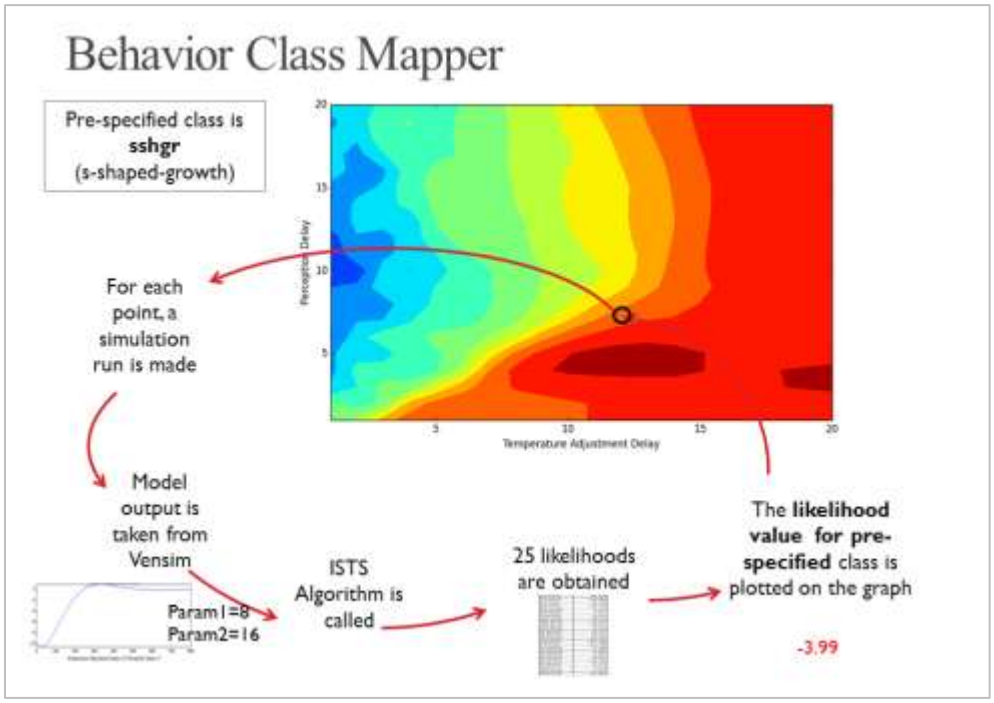


Figure 29. Workflow of Behavior Class Mapper.

In Figure 29, each point on the plot corresponds to one simulation run and for each run model output is obtained from Vensim. For each model output ISTS Algorithm is called and likelihood value for the pre-specified behavior class is plotted on the graph. This process is repeated for all parameter combinations. The intermediary parameter values are interpolated in order to obtain a contour plot with respect to parameter 1 and parameter 2. In this contour plot the warmer colors indicate higher likelihoods and vice versa. Screenshot from BATS using this feature is illustrated in Figure 30.

By looking at this contourplot it can be concluded that in order to stasify policy objective (i.e. find the parameter combination that yields S-shaped growth), the area indicated with the dark red can be chosen for values of parameters of uncertainity. It should be noted that this feature can be used for model

parameter calibration. Suppose that the modeler is searching for the parameter combination that yields S-shaped growth dynamics, then similar conclusions can be made using *Behavior Class Mapper* feature.

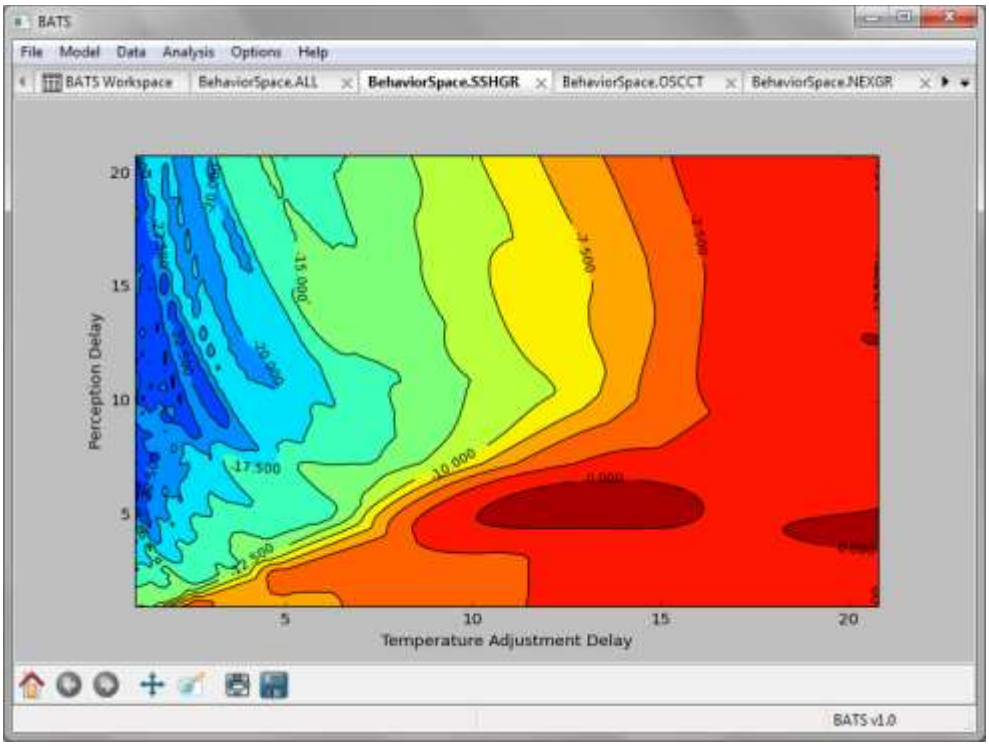


Figure 30. Screenshot for the resulting contour plot of Behavior Class Mapper.

Conclusion

System dynamics methodology necessitates formal quantitative output analysis methods according to the literature survey on both methodological studies and real life modeling projects. This need applies to all steps in the methodology, including model calibration, model testing, sensitivity analysis, policy analysis and design. Moreover, due to the policy-orientation of the approach, similarity of two dynamic behaviors should be assessed by their pattern related features such as trends, periodicities, fluctuations, amplitudes, etc. This renders traditional statistical curve fitting and forecasting techniques not applicable to behavior analysis. In the model analysis literature, there exists pattern oriented formal quantitative analysis techniques. However, these tools have not achieved large usage bases among the modelers/analysts in the field. Reasons behind this situation are identified as lack of automation of the analysis methods, lack of familiarity with the specialized programming environments, lack of communication with existing model building software and lack of user-friendliness. There are example efforts on various levels of automation from proof of concept to packaged software; however some of the aforementioned problems are valid also for them.

In this thesis two pattern-oriented quantitative behavior analysis methods are reviewed and integrated, and as a result a new standalone software package, namely BATS (Behavior Analysis and Testing

Software), is developed for pattern-oriented model output analysis. In its current state, considerable level of usability is achieved.

The new software is demonstrated in action for extreme condition testing of the density-dependent population growth model, behavior pattern comparison of two oscillatory time-series, and behavior sensitivity analysis of the temperature adjustment model. These demonstrations provide a guideline for available usage modes of BATS.

In its current version BATS is distributed as freeware software to the field[2]. Furthermore, BATS is designed and developed as an extensible platform. Future opportunities include integration of existing model/behavior analysis tools /methods in the literature such as statistical screening method by Ford and Flynn (2005), parameter specification by genetic algorithms by Yücel and Barlas (2011), and behavior clustering by Yücel (2012) to BATS. Future directions also includes design related improvements, additional connectivity with modeling software (e.g. Stella), and identification of new procedures and functionalities.

Appendix

Equations for Density-Dependent Growth Model

bf = normal bf*effect of crowding on bf
births = Population*bf
Capacity = 200
Crowding = Population/Capacity
deaths = Population*df
df = 0.06
effect of crowding on bf= WITH LOOKUP (Crowding, ([(0,0)-(2,1.33)], (0,1.33333), (0.166667,1.31667), (0.333333,1.3), (0.5,1.25), (0.666667,1.18333), (0.833333,1.1), (1,1), (1.16667,0.883333), (1.33333, 0.745), (1.5,0.583333), (1.66667,0.4), (1.83333, 0.211667), (2,0)))
FINAL TIME = 300
initial population = 20
INITIAL TIME = 1
normal bf = 0.06
Population= INTEG (births-deaths, initial population)
SAVEPER = 1
TIME STEP = 0.125

Equations for Temperature Adjustment Model

Actual Room Temperature = INTEG (Room Temperature Change, -10)
Desired Room Temperature = 0
FINAL TIME = 100
INITIAL TIME = 0
Perceived Room Temperature= INTEG (Perception Change, 5)

[2] As of March 2014, the project webpage is under construction. Please refer to main webpage of the research group http://www.ie.boun.edu.tr/labs/sesdyn

Perception Change = (Actual Room Temperature - Perceived Room Temperature) /Perception Delay
Perception Delay = 10
Room Temperature Change= (Desired Room Temperature - Perceived Room Temperature) /Temperature Adjustment Delay
SAVEPER = 1
Temperature Adjustment Delay = 10
TIME STEP = 0.125

Labels of generic dynamic patterns

The descriptions of labels for generic dynamic patterns in Figure 2 at page 5 are depicted in Table 4.

Table 4. Description of labels of generic dynamic patterns.

Class id	Description
ZERO	Zero
CONST	Constant
PLINR	Linear with positive slope
PEXGR	Positive exponential growth
NEXGR	Negative exponential growth
SSHGR	S-shaped growth
NLINR	Linear with negative slope
NEXDC	Negative exponential decline
SSHDC	S-shaped decline
PEXDC	Positive exponential decline
GR1DA	Growth with decreasing rate followed by decline to equilibrium (growth level is less than decline level)
GR1DB	Growth with decreasing rate followed by decline to equilibrium (growth level is greater than decline level)
GR2DA	S-shaped growth and decline to equilibrium (growth level is less than decline level)
GR2DB	S-shaped growth and decline to equilibrium (growth level is greater than decline level)
D1GRA	Decline with increasing rate followed by growth to equilibrium (decline level is less than growth level)
D1GRB	Decline with increasing rate followed by growth to equilibrium (decline level is greater than growth level)
D2GRA	S-shaped decline and growth to equilibrium (decline level is less than growth level)
D2GRB	S-shaped decline and growth to equilibrium (decline level is greater than growth level)
G1PED	Growth with decreasing rate followed by positive exponential decline
G2PED	S-shaped growth followed by positive exponential decline
D1PEG	Decline with increasing rate followed by positive exponential growth
D2PEG	S-shaped decline followed by positive exponential growth
OSCCT	Oscillation around constant mean
OSCGR	Oscillation around linearly growing trend
OSCDC	Oscillation around linearly declining trend

References

Anderson, O. D., 1982, Sample Serial Correlations From ARIMA Processes," in D. Anderson, O. and M. R.Perryman. (eds.), In Applied Time Series Analysis," pp. 1131- 1175, Netherlands, North-Holland.

Barlas, Y., 1989, “Multiple Tests for Validation of System Dynamics Type of Simulation Models.", European Journal of Operational Research, Vol. 42, No. 1, pp. 59-87.

Barlas, Y., 1990, “An Autocorrelation Function Test For Output Validation.", Simulation, Vol. 55, No. 1, pp. 7-16.

Barlas, Y., 1996, “Formal Aspects of Model Validity and Validation in System Dynamics.", System Dynamics Review, Vol. 12, No. 3, pp. 183-210.

Barlas, Y., 2002, “System Dynamics: Systemic Feedback Modeling for Policy Analysis," in Barlas, Y. (ed.), Knowledge for Sustainable Development - An Insight into the Encyclopedia of Life Support Systems," pp. 1131 -1175, UNESCO-EOLSS Publishers, Paris, France; Oxford, UK.

Barlas, Y., 2007, “Leverage Points to March Upward from the Aimless Plateau", "System Dynamics Review, Vol. 23, No. 4, pp. 469{473.

Barlas, Y. and A. Erdem, 1994, “Output Behavior Validation In System Dynamics Simulation," Proceedings Of The European Simulation Symposium, İstanbul, Turkey, pp. 81{84.

Barlas, Y. and K. Kanar, 1999, “A Dynamic Pattern-oriented Tests for Model Validation.", Proceedings of 4th Systems Science European Congress, İstanbul, Turkey, The System Dynamics Society.

Barlas, Y., H. Topaloğlu and S. Yılankaya, 1997, “A Behavior Validity Testing Sofware (BTS).", Proceedings of 15th International Conference of the System Dynamics Society, İstanbul, Turkey, The System Dynamics Society.

Bloomeld, P., 1976, Fourier Analysis of Time Series an Introduction, Wiley, NewYork.

Bog, S. and Y. Barlas, 2005, “Automated Dynamic Pattern Testing, Parameter Calibration and Policy Improvement," Proceedings of the 23rd International Conference of the System Dynamics Society, boston, MA, USA., The System Dynamics Society.

Box, G. and G. Jenkins, 1970, Time Series Analysis, Forecasting and Control, Holden Day Inc., San Francisco, CA.

Bozyayla, E., 2001, Evaluation of Different Model Behavior Validation Tests, M.S. Thesis, Boğaziçi University.

Drechsler, M., 1998, Sensitivity Analysis of Complex Models," Biological Conservation, Vol. 86, No. 3, pp. 401 - 412.

Ekşin, C., 2009, Application of Genetic Algorithms to Analysis and Policy Design in System Dynamics, M.S. Thesis, Boğaziçi University.

Ford, A. and H. Flynn, 2005, “Statistical Screening of System Dynamics Models,” System Dynamics Review, Vol. 21, No. 4, pp. 273-303.

Forrester, J. and P. Senge, 1980, “Test for Building Conference in System Dynamics Models," TIMS Studies in the Management Sciences, Vol. 14, pp. 209-228.

Güneralp, B., 2006, “Towards Coherent Loop Dominance Analysis: Progress in Eigenvalue Elasticity Analysis." System Dynamics Review, Vol. 22, No. 3, p. 263-289.

Hearne, J., 1985, “Sensitivity Analysis of Parameter Combinations," Applied Mathematical Modelling, Vol. 9, No. 2, pp. 106 - 108.

Kampmann, C. E. and R. Oliva, 2006, “Loop Eigenvalue Elasticity Analysis: Three Case Studies," System Dynamics Review, Vol. 22, No. 2, pp. 141-162.

Kanar, K., 1999, Structure Oriented Behaviour Tests in Model Validation, M.S. Thesis, Boğazici University.

Miller, J. H., 1998, “Active Nonlinear Tests (ANTs) of Complex Simulation Models." Management Science, Vol. 44, No. 6, pp. 820 -830.

Mitsa, T., 2010, Temporal Data Mining, Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, Taylor & Francis, Boca Raton, FL.

Peterson, D. W. and R. L. Eberlein, 1994, Reality Check: A Bridge Between Systems Thinking and System Dynamics." System Dynamics Review, Vol. 10, No. 2-3, pp. 159-174.

Phaff, W. G., 2006, Investigating Model Analysis: A Critical Examination of Current Methods for Model Behavioural Analysis in System Dynamics, M.S. Thesis, Delft University of Technology.

Richardson, G. P., 1996, “Problems for the Future of System Dynamics," System Dynamics Review, Vol. 12, No. 2, pp. 141{157.

Richardson, G. P., 1999, “Reflections for the Future of System Dynamics," The Journal of the Operational Research Society, Vol. 50, No. 4, pp. 440-449.

Soylu, S., 2006, Generic Dynamic Patterns: Testing by Empirical Evidence, M.S. Thesis, Boğaziçi University.

Sterman, J., 1984, Appropriate Summary Statistics for Evaluating the Historic Fit of System Dynamics Models," Dynamica, Vol. 10, pp. 51-66.

Sterman, J. D., 2000, Business Dynamics, McGraw-Hill, New York.

Theil, H., 1966, Applied Economic Forecasting, Studies in Mathematical and Managerial Economics, North-Holland Pub. Co., Amsterdam.

Yücel, G. and Y. Barlas, 2011, Automated Parameter Specification in Dynamic Feedback Models Based on Behavior Pattern Features." System Dynamics Review, Vol. 27, No. 2, pp. 195-215.

Yücel G., 2012, A Novel Way to Measure (Dis)similarity Between Model Behaviors Based on Dynamic Pattern Features, Proceedings of the 30th International Conference of the System Dynamics Society, St. Gallen, Switzerland, The System Dynamics Society.