Generalised Loop Deactivation Method

An Extension of Ford's Behavioural Approach to Loop Dominance Analysis and its Applications

H. Willem Geert Phaff
Delft University of Technology
Faculty of Technology, Policy and Management
Jaffalaan 5, 2628 BX, Delft
The Netherlands
Tel: +31-15-2783010
h.w.g.phaff@tudelft.nl

April 3, 2008

Abstract

This paper presents progress on the conceptualisation and implementation of an extended version of Ford's behavioural approach (Ford 1999) to feedback loop dominance analysis. The need for the extension of the original method is discussed, as are the methodological consequences of changing and implementing the method. The changed method presented here is refered to as the Generalised Loop Deactivation Method. The automated version of this method is tested on three models. The first of these is used to check the results of the method, the second to discuss several methodological changes and the third model is used to demonstrate how to detect superfluous structure in a larger model. Significant findings include recommendations on how to eliminate loops from a model, a fully automated version of the loop deactivation method and an extension of its use into model simplification, moving the method beyond loop dominance analysis.[1]

[1] My gratitude goes out to Els van Daalen, Jill Slinger and David Ford, who provided valuable feedback on my initial thoughts and writings.

1 Introduction

One of the prominent unanswered questions in System Dynamics is how structure drives behaviour. Several authors have identified the development of formal methods to find out how structure drives behavior as a major outstanding issue (Sterman 2000, Richardson 1996). Consequently, in recent years, a wide range of methods aimed at resolving this issue has been developed.

The most notable of these methods are eigen-structure based methods (Kampmann 1996a, Kampmann and Oliva 2006, Saleh et al. 2006), the pathway participation method (Mojtahedzadeh et al. 2004) and Ford's behavioural approach (Ford 1999). The first of these, the eigen-structure based methods, use the response of eigen-structure of linearised versions of the model to perturbation of elements to assess the influence of those elements on the model's behaviour. The eigen-structure based methods are considered a highly mathematical and formal approach to explaining model behaviour. The second, the pathway participation method, uses the contribution of causal structure to the behaviour of a selected variable of interest to trace dominant structure around the model. The pathway participation method is implemented in the DIGEST software package (Mojtahedzadeh et al. 2004). The third method, and focus of this paper, is Ford's behavioural approach. This method uses the deactivation of loops and the difference in output between the original model and the modified model to asses the role of deactivated loops.

The philosophy underlying Ford's behavioural method is much in line with the classic methods that rely on hypothesis testing (Richardson and Pugh 1981). In contrast to the classic hypothesis testing, however, Ford provides a more systematic approach to both what modifications will be made to the model and how the role of a loop is assessed based on the effect of that modification. Ford (1999) assesses the role of a loop by deactivating it and comparing the behaviour of the modified model with the original.

This paper presents progress on the formulation, automation and application of an extended version of Ford's behavioural approach. The modified method is referred to as the Generalized Loop Deactivation Method (GLDM); 'generalized' due to its applications beyond loop dominance analysis, 'loop deactivation' since this is the core mechanism of the method. Modifications to the original method are listed and their effects on the method's application are discussed using three brief analyses. The first of these analyses is a simple verification of the method by inspecting whether it provides similar results to the non-automated version of the method. The second addresses several methodological changes. The third analysis uses the automated method to search for possible ways to simplify the model under investigation.

The structure of this paper is as follows: we will first define why and how we extended Ford's behavioural approach in Section 2. After which we test the method and discuss several methodological issues. With the tested method, we then demonstrate a particular use case on the Market Growth model (Forrester 1968) and compare this to eigenvalue elasticity analysis. Finally we close with conclusions and recommendations for further research.

2 Modifications to the Method

In this section we will discuss the extensions made to the method and the background for making htem. Note that since viewing a model as a directed graph (Diestel 2005) is central to elements in the method, several terms will be used synonymously; an edge refers to the link between two variables, vertex is used as a synonym to variable.

Ford (1999) introduced a flexible, formal method for assessing the role of particular loops in a model. Based on deactivating a selected subset of loops, the method provides analysts with a structured approach to assess the role of the loop at during a particular interval of behaviour. The method is supported by a rigorous analysis procedure that is able to deal with multiple loops stronly influencing behaviour.

In a nutshell, the procedure consists of selecting a variable of interest, dividing its behaviour up into different intervals and eliminating a selected set of loops for each interval. Based on the difference between the behaviour of the unmodified model and the model with a loop eliminated, the role that that loop has in generating behaviour in that interval is assessed. By eliminating more than one loop at a time, the method is able to provide the analyst with conclusions even in situation where multiple loops share dominance. There are however, several remaining challenges concerning the method and how to execute it.

First, the method is labour intensive, which has consequences for how it can be used in analysis. A decent size model easily has tens of loops in it and isolating every single loop (even if that is possible) and manually deactivating them would take hours, if not days. Consequently, the analyst has to limit himself to assessing the role of only a selected subset of the loops in the model. An automated version of the method that rapidly scans the loops in the model would circumvent this disadvantage. Second, the method assumes the interesting loops in the model to be known; it is not formally supported by structural analysis. Third, there are several methods available to the analyst in eliminating a loop. A loop is eliminated by deactivating a link that is unique to that loop (not included in any other loop). the link, the loop is eliminated. The open question is to what value to fix the link. Fourth, the method restricts itself to eliminating loops only at inflection points or minima/maxima of the trajectory of the selected variable of interest. More detailed information on the role of a loop can be obtained by allowing more flexibility in this step of the method.

The issues identified in the above paragraph will be discussed in this section. We will first discuss what edges (links) to deactivate, how to deactivate them and when to deactivate them. After this, the main ideas behind the automation of the method are explained.

2.1 What edges in the model to deactivate

The published version of Ford's behavioural approach has no explicit method for identifying the edges to deactivate. It assumes sufficient existing knowledge of the feedback structure of the model to identify these by hand. Consequently, the method can be extended by using existing formal methods for loop detection.

Since it is necessary to identify the edges that make up the loops in a system, a representation of feedback structure is needed that provides the analyst and method with this information. The formal representation of a feedback loop that directly provides the required information is the Directed Cycle Matrix (DCM, an edge-cycle incidence matrix (Oliva 2004, Kampmann 1996b)). Any loop detection method that produces such a matrix can be used as input for the method. The modified method presented in this paper makes use of the Shortest Independent Loop Set (SILS) method as developed by Oliva (2004). The SILS method guarantees that any loop that has a unique edge associated with it will automatically be included in the loop set. Alternatives include an extensive

depth-first search, the algorithm as designed by Kampmann (1996b) or selecting loops by hand and building a DCM.

Since the directed cycle matrix identifies what edges make up what loops, it can assist in executing Ford's behavioural approach in two ways:

1. Finding those edges that uniquely identify a loop. Any edge for which the associated row sum is 1 occurs in only one loop. These edges identify the set of loops that can be eliminated independently; by deactivating these edges only one loop is eliminated. Any loop that does not have an edge uniquely associated with it, can not be analyzed independent from other loops.

2. Finding the loops eliminated by deactivating a particular edge. If the analyst starts out with a set of edges, the DCM allows for the identification of the loops associated with each edge.

For all intents and purposes, the DCM provided by the structural analysis tools is considered as representative for the loop set in the model. If the analyst wants to use a particular loop set, he need only replace the loop detection method.

2.2 How to Deactivate them

A loop is eliminated by deactivating an edge between to variables. To deactivate an edge the representation of the indepedent variable (the 'from' variable) is set to a fixed value *within the equation of the dependendent variable* (the 'to' variable). So, if $a = b \cdot c$ and we seek to deactivate the edge from b to a, we set the equation of a to $a = b_s \cdot c$, where b_s is some fixed, constant value. The equation of b itself is not changed. We identify three methods of deactivating links in the model, distinguished by what fixed value the representation of the independent variable is set to: fixing a specific relation to zero, to steady state gain, or the value it has at the moment of deactivation[1].

With regards to the first method, setting a link to zero often eliminates more than just one link. For example, if an auxilary a is determined according to $a = b \cdot c$, setting the $b \rightarrow a$ link to zero by changing the equation of a to $a = 0 \cdot c$ also deactivates the $c \rightarrow a$ link. Taking into account that a itself is set to zero as well, this change will probably be felt throughout the entire system; the shock will propagate through the system. The deactivation method does not fully isolate the feedback loop the analyst is investigating.

The propagation effect mentioned above will also be present when using the second method of deactivation. Using this method, to eliminate the edge $a \rightarrow b$, the equation for a would be changed to $a = b_s \cdot c$, where b_s is the steady state value of b. Especially in higly connected, non-linear systems in a state that is relatively far from steady state, setting the value of a relationship to its value at steady state will cause a significant shock to the system. In the example, the difference between b_s and b may be large, so modifying the equation for a would immediately significantly change the value of a as well.

The third and only remaining method is fixing the value of a link/relationship to the value at the time of elimination. So, at the time of elimination the equation for a becomes $a = b_e \cdot c$, where b_e is the value of b at the time of elimination. This seems to work well in smaller models. In larger models, however, it is often not able to cause a dramatic switch in behaviour immediately, where the other two methods do. At the moment of deactivation, the dependent variable (a in the example) retains the same value.

In a sense, the solution to this dilemma is not offered in this paper. The method, however, does use the third method as the default choice. Why this is the case is explained and demonstrated in Section 3.2.

2.3 When to Deactivate

Over an interval displaying one type of behaviour (balancing/exponential growth, balancing/exponential decline) there may be several loops generating the behaviour, one after the other. Only inspecting the effects of elimination at the beginning of the interval will never surface the later loops as being responsible for driving behaviour. So, in order to find those changes in driving structure, it is necessary to deactivate edges at more points in time.

Deactivating edges at more times of the analysis also has the benefit of providing a richer picture of the role of the eliminated loop over time. Consequently, we are less rigid in determining at what times a loop can be deactivated. While this reduces the rigor of the method slighty, it opens up novel ways of analyzing the role of a loop. This feature is illustrated in Section 3.2.

2.4 Specification

Even with the support of structural analysis (i.e. loop detection methods), the analyst would still be left with significant manual tasks. This subsection addresses how these, often menial, tasks have been automated. The tasks are: formulating switch variables, taking out combinations of loops and performing the analysis for more than one variable. Finally, it must be mentioned that, given that the method can be used in several different ways, many of the choices of how exactly to execute it are left to the analyst; they are arguments in calling the function. Flexibility and power of analysis are design requirements.

2.4.1 Formulating switch variables

In the previous formulation of the method, much of the analyst's time was invested in specifying switch variables. Variables that, at a certain point in time, would deactivate a certain loop. These take significant effort to formulate and, in addition, clutter up the equations. The automation uses a java-based representation of System Dynamics models[2] that mitigates the need for formulating the control variables (Appendix A.1).

2.4.2 Combinations of Loops

One of the strengths of Ford's behavioural approach is its ability to deal with multiple loops sharing dominance. The preferred method of deactivating combinations of loops, however, is labour intensive. In simple four loop model for instance, to test for shared dominance or shadow loops, the modeller would have to deactive 16 different combinations of loops for every time the analysis is executed, just to test all pairwise combinations. While this task is still feasible (if tedious) for a small model, even for medium sized models the costs in terms of effort and time would be disproportionate.

The implemented solution is to define what combinations of edges to deactivate in advance and to leave the actual deactivation to the method itself. Which combinations of loops to deactivate in what order is defined in the so-called deactivation matrix. The deactivation matrix represents what combinations of edges will be eliminated for each iteration of the method. The matrix is binary matrix of size $n \times m$, where n is the number of edges. Appendix A.1.1 contains further details.

2.4.3 Variable of Interest

One minor change is that the current version of the method explicitly allows for the selecting of more than one variable of interest. This enables the analyst to quickly analyze the effect of taking out a loop on, for instance, all

states or a selected set of performance indicators.

2.4.4 Control for the Analyst

Given the design requirement of flexibility, the analysis method has several options that can be set by the analyst. The elements of the method that are still debatable, such as the method of deactivating edges, are left to the analyst. The options are:

- **Variables of Interest.** The variables the analyst is interested in. The role of a loop will be assessed based on the change in behaviour of the variables of interest.
- **Times of Analysis.** The analyst has complete control over when the loops will be deactivated.
- **Edges.** If the analyst specifies the edges to be deactivated, the method will focus on these without using structural analysis to determine the edges to deactivate. If it is not given, the function will derives the edges it will deactivate from the Directed Cycle Matrix of its loop set.
- **Directed Cycle Matrix.** The Directed Cycle Matrix (DCM) represents the loop set the function uses to analyze the model. It is used in two ways. First, if the edges to deactivate are not given by the analyst, it will derive the edges to deactivate from this matrix. Second, if edges are given, it will determine the loops associated with the edges; in which loops the edges are included. If no DCM is given, the method will use the given structural methods to determine a DCM.
- **Structural Methods.** The methods used to determine the DCM if none is given. By default, the function uses the SILS algorithms (Oliva 2004) to determine the DCM.
- **Deactivation method.** This option controls the method by which the edges are deactivated. This is a reference to a function as defined by the analyst and hence can take any form desired, from setting edges to zero to using lookups to set specific edges to specific values. The default method is to fix the edges to the values they have at the moment of deactivation.
- **Deactivation Matrix.** This option defines the deactivation matrix according to which to deactivate edges and combinations of edges. If the analyst does not specify this, the method uses a $n \times n$ identity matrix, where n is the number of nominated edges.
- **Report interval.** Once the edges are deactivated, the model must be run in its modified form. The values of the variable(s) of interest have to be tracked over the course of the simulation run. The report interval controls how often the value of the variable of interest is checked. There is no default for this option; it has to be set.

3 Application

This section will show two different uses of Ford's behavioural approach. Namely:

1. The classical use of Ford's behavioural approach that aims to find dominant loops.
2. The inspection of the changing role of a loop during a highly dynamic interval.

The matlab code used to generate these analyses is included in Appendix D.

3.1 Finding Dominant Loops

This analysis is performed on the Yeast model(Güneralp 2006, Phaff et al. 2006, Mojtahedzadeh 2007). The approach is the default execution of Ford's behavioural approach on a previously analysed model. This section serves to show how to execute and set up the automated analysis, the actual results have been discussed by other authors in other papers . The analysis is of a small model and it is identical to the classic Ford method, but fully automated. The secondary purpose is to compare the results of the automated version of the method with the results of previous analyses. The code for this analysis can be found in Appendix D.1, line numbers will be mentioned in the text to indicate what section of code represents what activity.

First, the initial setup of the analysis defines the model, the integration settings, what function to use to analyze the model and what the variables of interest are (line 9 to 22). Second, a reference run of the model is generated and the times at which the model swithes behaviour pattern are detected (line 24 to 41). Note that the analysis run is done at the inflection points and maxima of the variable of interest. The classical variant of the method divides the behaviour of the variable of interest up into intervals based on the so-called atomic behaviour pattern (ABP, Ford (1999)). Balancing behaviour results in an ABP less than zero, accelarating behaviour results in an ABP greater than zero. In contrast to Ford, we calculate the ABP by

$$ABP = sign\left(\frac{dx}{dt}\right) sign\left(\frac{d^2x}{dt^2}\right) \tag{1}$$

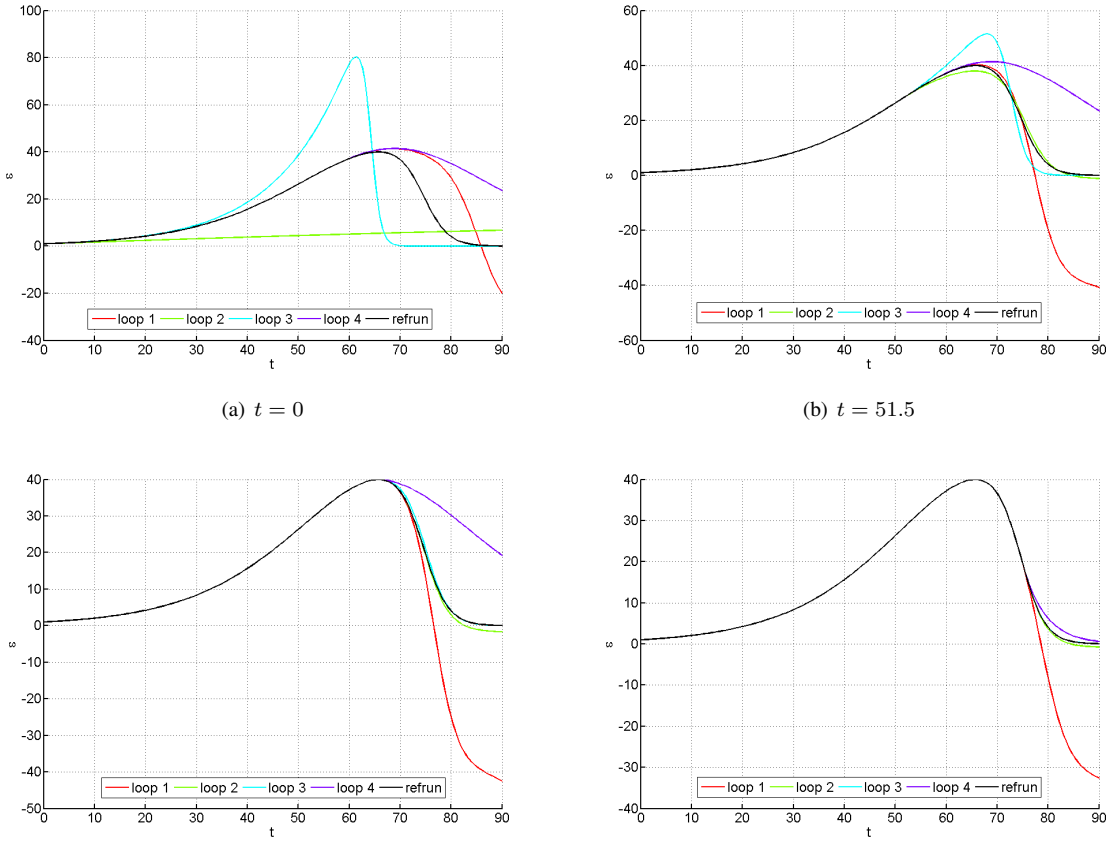
where x is the variable of interest. An interval is a period during which the ABP keeps the same sign. Given that the method is supposed to be run at any time the analyst desires and analysis times can be at any time during a model run, the ABP is determined outside the rest of the method. The times at which loops are deactivated are the times at which the ABP switches sign. These times are then passed to the method, no information on the ABP itself is passsed. Finding where the variable of interest switches behaviour pattern is a trivial exercise (line 37 to 41), assuming the behaviour of the variable is smooth.

After this, the analysis is executed and results are stored (line 43 to 54). The results of this can be found in Figure 1. To interpret the results of deactivating the loops, the method uses the ABP again. Only when the deactivation of a loop results in a radical change of behaviour is it considered dominant. A radical change of behaviour is defined as switching the ABP of the variable of interest immediately after the deactivation of the loop.

Consequently, using the switching of the ABP as a signal for dominance, we consider the following loops as chiefly responsible for behaviour:

- **Interval 1.** Loop 2, a positive birth loop is considered dominant.

- **Interval 2.** Loop 3, a loop constraining the variable of interest is considered dominant.
- **Interval 3.** No clear dominant loop. Further investigation leads towards a pair of loops dominating.
- **Interval 4.** Although not immediately apparent from the graph, eliminating loop 1, a balancing loop, results in a switch of behaviour pattern. Loop 1 is considered dominant in this interval.



(a) $t = 0$

(b) $t = 51.5$

(c) $t = 66$

(d) $t = 75$

Figure 1: These graphs show the effects of eliminating all the loops in the Yeast model at different moments in the model run. Each line is named after the eliminated loop. The more a line deviates from the original model (refrun), the more influential the eliminated loop was. According to Ford's original method, only if the elimination of a loop triggers a change in atomic behaviour pattern is it considered 'dominant'. Results confirm those found in other applications of the method (Mojtahedzadeh 2007, Phaff et al. 2006).

3.2 Analyzing the Effect of One Particular Loop

This demonstration focusses on the changing role of a single loop. In this case, the Self Ordering loop in the Long Wave model (Sterman 1985) is analyzed. This is a highly nonlinear economic model developed by Sterman (1985) to explain the long term economic cycles caused by capital self-ordering in the simplest possible terms. Two different methods of eliminating an edge are contrasted; setting a link to steady state gain and fixing the link to its value at the moment of deactiviation. The second feature shown in this script is the ability to focus on a particular interval within a run, at desired moments. While the model runs for around 150 time units, we are only interested in the behaviour in the last 20 time units. The analysis is performed at a relatively high frequency: every 1 time unit. The matlab code can be found in Appendix D.2; line numbers will be mentioned in the text to indicate what section of code represents what activity.

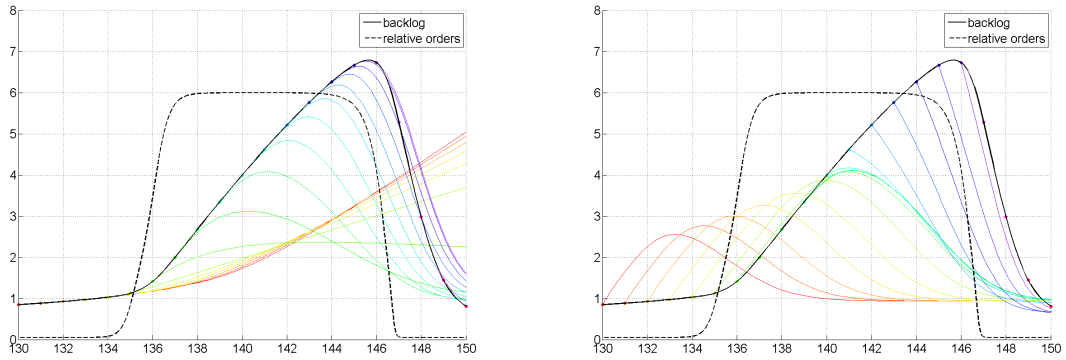
Again, the analysis starts with setting certain elements general to the all analyses and generating a reference run to compare the results of deactivating the loops to (line 8 to 27). For the reference run, we are interested in two variables; *backlog* and *relative orders*. *Relative orders* is tracked to indirectly determine the gain of the Self Ordering loop; if it is at it's minimum or maximum, the gain of the Self Ordering loop is 0. The tracking of the two selected variables starts and ends at the same time as the analysis.

With the reference run generated and the model reset to its initial state, the analysis is set up. First, the edge is selected that uniquely controls the activity of the Self Ordering loop (line 32 to 34); this is the edge leading from *desired production* to *desired cap(ital)* (Ford 1999, Kampmann 1996b). This is then set in the analysis data, as the only edge to eliminate (line 42). The analysisfunction is set to the GLDM, while the deactivation matrix is 1 and the variable of interest is set to *backlog* (line 43 to 46). The times at which the loop is deactivated are set to every time unit after $t = 130$, so the loop is deactivated at $t = 130, t = 131, t = 132, \ldots t = 150$ (line 47). The analysis is then executed (line 48 to 49).

To run the same analysis with a different method of deactivation, all the analyst has to do is add two lines to define the method of eliminating the edge. The lines define a function according to which the edge is taken out, this function is then set as a value in the appropriate field of the analysis data (line 39 to 40). In effect, the function just sets the edge to a predetermined value; when applied to multiple edges, the function could use a hash-table (with edges and values as key-value pairs) to look up the steady state value of each edge. The analysis method is set up as below.

The results of both analyses are plotted according to a color-scale (HSV, Foley and van Dam (1982)), based on which analysis is plotted; later analyses are higher in the color scale. Besides the results of the analysis, the original trajectory of the variable of interest is also plotted in the black, solid line, while *relative orders* is drawn next to it, in the dashed black line (Figure 2). The graphs clearly show that setting the link to steady state gain results in the trajectory of *backlog* immediately deviating significantly from the reference behaviour. Consequently, the method would consider the Self Ordering loop dominant *when its gain is zero*, since the behaviour pattern switches even when relative orders is at its maximum. The default method however shows that at the moments where *relative orders* is in flux (its time derivative is not zero), the behaviour of the modified model deviates faster from the reference run than when *relative orders* is not in flux. While the classical method would only consider the loop dominant at $t = 136$, it shows that the modified model deviates more quickly from the reference run when *relative orders* is in flux. So, dominance would be harder to allocate in the final case, but we can clearly see when the loop becomes more or less influential in the analyzed interval.

In conclusion, the method of setting a link to its steady state value is shown to result in the classic method



(a) Default Method

(b) Steady State Gain

Figure 2: These graphs show the effects of taking out the Self Ordering loop in the Long Wave model according to two different methods. The first method 'Default Method' sets the edge to the value it has at the moment of deactivation. The second method 'Steady State Gain' sets the link to the value that it would have if the model were in steady state. The solid black line is a time plot of the variable of interest, *backlog*. The dashed black line is a plot of *relative orders*, which shows where the gain of the Self Ordering loop is 0; the gain is zero where the time derivative of *relative orders* is zero.

attributing dominance to a loop when its gain is zero. This casts serious doubt on the reliability of this method of elimination. The default method of deactivating an edge is set to fixing it to its value at the moment of deactivation. The analyst can still change this to any method he deems more appropriate.

In addition, the automated version of the method makes it easy to eliminate links at many moments in time close together. The closer these moments are together, the more precisely the analyst can follow the changing influence of the loop.

4 Structured Approach to Model Simplification

Given the nature of the method presented here, it can be used to automatically scan structure, investigating the effects of eliminating a large amount of links in the system. Investigating dominance is not the only use of this procedure. One of the more promising uses lies in finding those loops that do not have a significant influence on behaviour; these loops might point to what structure in the model is superfluous.

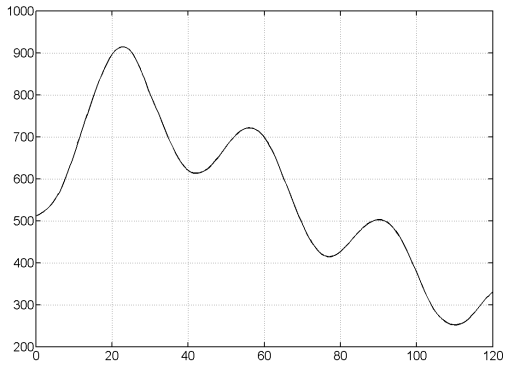
Consequently, another potential application of the modified version of Ford's behavioural approach is to systematically eliminate relations from the model to see how it could be simplified. In the analysis presented in this paper, we will eliminate edges according to the default method, but will do so only from the start of the model run. In effect, we simplify the model to have one less edge and does so for each edge in the model that uniquely identifies a loop. We will then, in contrast to Ford's behavioural approach, focus on the loops whose elimination made *no* significant difference in generating the reference behaviour of the model. The mechanics of the method are similar to Ford's, but the difference lies in what conclusions are drawn from the results.

The example model the techique will be applied on is Forrester's Market Growth Model. In addition, the implemented version allows for the analyst to track the behaviour of several variables of interest in one analysis. Consequently, the effect of eliminating loops is judged by the change in behaviour in two different variables.

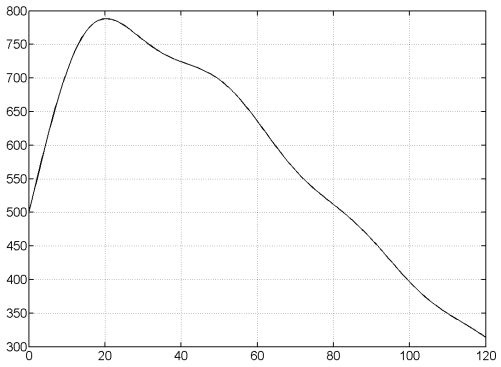
The matlab code can be found in Appendix D.3; line numbers are listed in the text.

4.1 Deactivating the Loops

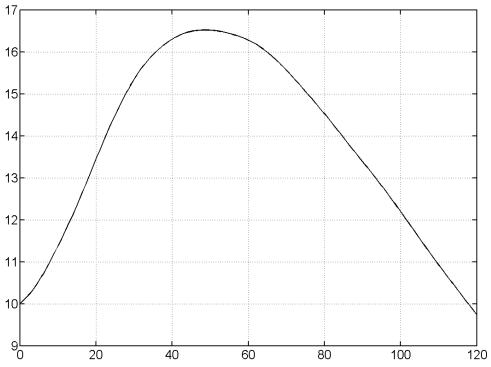
An analysis performed on the Market Growth Model (Forrester 1968, Morecroft 1983) will be used to demonstrate the approach. The Market Growth Model is a classic Forrester model, used by Morecroft (1983) to showcase the modelling of bounded rationality in System Dynamics Models. It is also an example of the 'growth and underinvestment' archetype (Senge 1990). The model represents a electronics company where an initial increase in salespeople leads to and increase in the budget for salespeople, which, in return results in more salespeople. These sales drive the selling of items. The problem is, however, that the production capacity of the company cannot keep up, which results in the speed at which orders are delivered rapidly declining. This leads to the eventual collapse in orders booked (Figure 3). The appendix contains both the code for an eigenvalue elasticity analysis and the GLDM-based analysis.



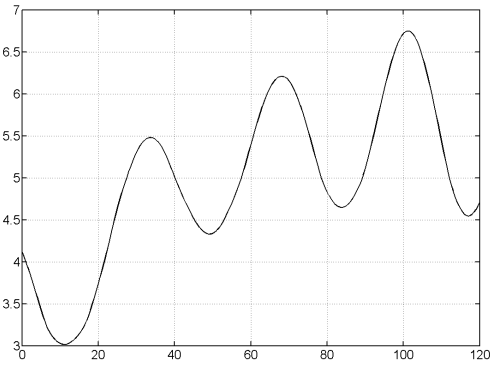
(a) Orders Booked



(b) Production Capacity



(c) Sales People

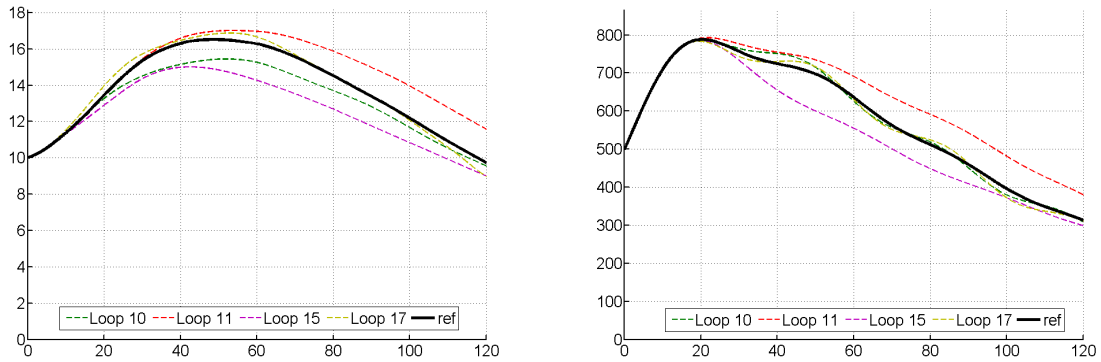


(d) Delivery Delay

Figure 3: Behaviour of several of the central variables in the Market Growth model.

Selecting *Production Capacity* and *Sales People* as variables of interest for the analysis, we run the analysis as described; for $t = 0$ only and with the default selection of loops and edges based on the SILS algorithm (line 47 to 63). Reference runs are generated as well. The analysis only has to be run once when analyzing more than one variable of interest.

The 14 different model runs - each of them associated with the elimination of one particular loop - are plotted for all variables of interest showing the effects of eliminating all loops that could be isolatedn (Appendix B). This, however, suffers from readability issues, so for further analysis we will use these graphs with most of the loops not shown, focussing on the loops whose elimination does not have much of an effect on the behaviour of the variables of interest, and, hence, are good canditates for being simplified away (Figure 4).



(a) Sales People

(b) Production Capacity

Figure 4: Behaviour of the variables of interest with only four eliminated loops.

The loops that could possibly contain superfluous structure are Loop10, Loop11, Loop15 and Loop17. Taking the integral over time of the absolute value of the difference, divided by time for each loop for each variable of interest gives us the result as in Table 1. Loop 10 and Loop 17 consistently perform better (lower difference) than Loop 11 and Loop 15. This measure of devation from the reference trajectory (Hearne 1987) would not be suited for all models[3], but in this case gives a reasonable indication of how much the behaviour of the variable of interest has changed. Given the results of eliminating the loops, we focus on Loop10 and Loop 17 in finding superfluous

Table 1: Integral of the absolute difference between the reference run and the modified models.

	Loop 10	Loop 11	Loop 15	Loop 17
Prod. Cap.	7.4764	47.8136	41.5809	7.6618
Salespeople	0.6434	0.8302	1.2360	0.2182

structure. Both loops include the utilization of capacity (Figure 5). Running the model and inspecting the value of this variable, we see that it hardly deviates from its initial value, leading to the possibility of excluding the variable and thereby simplifying the model. Elimination of the variable results in a run virtually identical to the reference run.

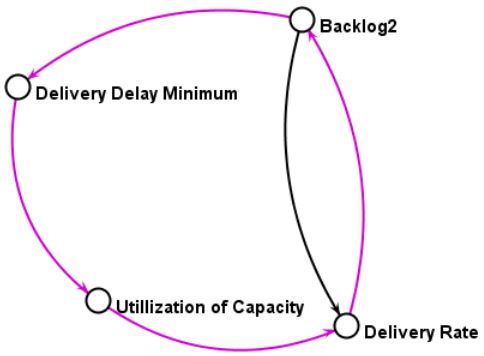


Figure 5: Loop 10 in the Market Growth model, as outlined in the magenta lines. The black line indicates that a smaller loop is included in the larger loop.

5 Conclusions

5.1 Methodological properties

In this article we set out to present progress on the formulation, automation and application of an extended version of Ford's behavioural approach. We have shown the following:

- Succesful automation of the method. By combing Ford's standard existing method, with an existing Java library for System Dynamics models, we were able to fully automate the procedure listed by Ford (1999). Every step, from the identification of feedback structure to deactivating loops is automated. Results of previous analyses have been reproduced.
- Method of Deactivation. Using the Long Wave model, we compared two methods of deactivating edges. The method of setting a link to its steady state value is shown to result in the classic method attributing dominance to a loop when its gain is zero. The default method of deactivating an edge is set to fixing it to its value at the moment of deactivation. The analyst can still change this to any method he deems more appropriate.
- Times of Deactivation. We demonstrated how deactivating a loop at times close together provides more detailed insight into how the influence of a loop changes over time.
- Extension of the method by using it for model simplification. The automation of the method allows for a much more rapid scan of the effect of eliminating feedback loops. The analyst can use this to find those loops whose elimination does not effect the reference mode of behaviour. This can be used as a way to identify superfluous structure in the model.

In short, the labor intensive nature of Ford's behavioural approach affects its methodological properties. Given the limited time an analyst has at his hands, an unautomated version of the method is not well suited to be used to answer the question what structure is driving behaviour. It is good at assessing the role of a one or several selected loops, but the effort required to eliminate all loops even in ILSs is large to the point of it being unlikely to be done. Consequently, exploring the model and finding influential loops without having a preconception of what loops are influential is difficult.

If the method is automated however, this changes. Loops and their related edges can be identified quickly and the modifications to the model done automatically, changing the time required from hours, or even days, to minutes. The effect is that the questions that can be answered with the method, change. It becomes more suited for initial exploration and can more quickly investigate the changing role of a selected loop over time.

In addition, the ability to quickly eliminate certain loops, makes it possible to use the method to scan for superfluous structure. The GLDM methods moves beyond dominance analysis. Instead of looking for loops whose elimination has an immediate and strong effect on the variable of interest's behaviour, we focus on the loops whose elimination virtually does not change the behaviour. The variables included in these loops could be candidates for removal from the model, thus simplifying the model.

5.2 Remaining Challenges

We identify several points that remain for further research. The first of these comes from the background of the method as a tool for finding 'dominant' structure. The original method has as a condition for attributing dominance

to a loop that the variable of interest should switch the sign of its ABP after the loop is eliminated. This presents an unambiguous conclusion regarding which structure is considered to be dominant at a certain point in time. Strictly keeping to only this method of assessing the role of a loop, however, seems to limit the conclusions that can be drawn by using the behavioural approach. Why not extend to additional conclusions? For instance, if the variable of interest (VOI) shows exponential growth, and the deactivation of a particular loop results in even faster growth, a constraining role can be attributed to that loop. The latter could have policy relevance if the problem owner is interested in accelerating growth, but would never have been found if the method would have only focused on finding dominant structure. Similar conclusions can be drawn with regards to loops that have (in oscillatory systems) an influence on frequency or dampening.

In summary, the ABP-condition seems to be based on dominance thinking and the perceived need of finding dominant structure. If more than just the dominance thinking is allowed, the analyst is allowed to use a more qualitative assessment of the role of a loop and he might end up with a richer story of how structure drives model behaviour.

Second, related to the first, is that one possible addition to the automated version could be a quantitative measure of influence for a loops. For instance, a possible measure would integrating the difference between the original model run and the modified model over a fixed, short time interval. The higher the difference, the more influential the loop. Other sources for measures include pattern matching (Yucel and Barlas 2007) and metrics generally used for validation (Barlas 1989). A measure for indicating the influence of a loop would also eliminate the need for finding pairs of shadow loops; the two loops would just be considered highly influential.

Third, as it stands now, the method is only suited for incremental simplification. It is good at finding minor, superfluous sections of structure; eliminating a few variables at most. In contrast, methods such as the one presented by Eberlein (1989) or methods such as dimensional analysis can be used to reduce the model a much smaller version; e.g. from a 10^{th} order model to a 3^{rd} order model. The current method is only suited eliminate superfluous structure. Also, note that the current simplification technique is based on a model reference run. The fact that the eliminated structure did not play a large role in generating the reference run, does not mean that a) there is no region in state space where it does not play a role and b) that it does not have an influence on the performance of policy options that modify structure.

Additional testing can be done by using the method as support for analysing the behaviour of separated submodels or other sections of the model. Since the structure to be eliminated can be set by the analyst, one could think of eliminating the connections between submodels, or all connections between non-parameter elements one by one. Again, combinations can be deactivated as well, even though the number of possible combinations grows quickly with the number of disconnected edges.

Another way of further testing the modified method is comparing it to different methods of formal model analysis. In his original paper Ford (1999), compared the method to eigenvalue elasticity analysis. A preliminary comparison of the GLDM to eigenvalue elasticity analysis indicate that results agree with eachother, but only to a certain degree Appendix C.

And, finally, there is no better assessment of a method's actual power than applying it in a project or study. The current paper only demonstrates potential uses on well-known, small models, but the method's true test would be its application in a full-fledged modelling study.

Notes

[1]This issue is also partially addressed by Phaff et al. (2006). We offer new perspectives on the issue however, so it is discussed here as well.

[2]The java-based representation is part of a larger effort to make models more independent from how they are simulated and in what context they are used.

[3]In an oscillatory model, if the elimination of a loop results in a small phase shift, the metric indicates a large difference between the original and the modified model

References

Barlas, Y.: 1989, Multiple tests for validation of system dynamics type of simulation models, *European Journal of Operational Research* **42**, 59–87. 17

Diestel, R.: 2005, *Graph Theory*, Springer–Verlag Heidelberg, New York. 3

Eberlein, R. L.: 1989, Simplification and understanding of models, *System Dynamics Review* **5**(1), 51–68. 17

Foley, J. D. and van Dam, A.: 1982, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley. 10

Ford, D.: 1999, A behavioural approach to feedback loop dominance analysis, *System Dynamics Review* **15**(1), 3–36. 1, 2, 3, 7, 10, 16, 17

Forrester, J. W.: 1968, Market growth as influenced by capital investment, *Industrial Management Review (MIT)* **9**(2). 2, 12

Güneralp, B.: 2006, Towards coherent loop dominance analysis: progress in eigenvalue elasticity analysis, *System Dynamics Review* **22**, 263–289. 7, 23

Hearne, J. W.: 1987, An approach to resolving the parameter sensitivity problem in system dynamics methodology, *Applied Mathematical Modelling* **11**, 315–318. 15

Kampmann, C. E.: 1996a, Feedback loop gains and system behaviour, *Proceedings of the 1996 International System Dynamics Conference*, System Dynamics Society, Albany, NY, Boston. 2

Kampmann, C. E.: 1996b, Feedback loop gains and system behaviour. Unpublished. 3, 4, 10

Kampmann, C. E. and Oliva, R.: 2006, Loop eigenvalue elasticity analysis: three case studies, *System Dynamics Review* **22**, 141–162. 2

Mojtahedzadeh, M. T.: 2007, Do the parallel lines meet? a comparison between pathway participation metrics and eigenvalue analysis, *Proceedings of the 2007 International System Dynamics Conference*, System Dynamics Society, Boston, MA. 7, 9

Mojtahedzadeh, M. T., Andersen, D. and Richardson, G. P.: 2004, Using digest to implement the pathway participation method for detecting influential system structure, *System Dynamics Review* **20**(1), 1–20. 2

Morecroft, J. D. W.: 1983, System dynamics: Portraying bounded rationality, *Omega* **11**, 131–142. 12

Oliva, R.: 2004, Model structure analysis through graph theory: partition heuristics and feedback structure decomposition, *System Dynamics Review* **20**(4), 313–336. 3, 6

Phaff, H. W. G., Slinger, J. H., van Daalen, C. and Güneralp, B.: 2006, Investigating model behavioural analysis: A critical examination of two methods, *Proceedings of the 2006 International System Dynamics Conference*, System Dynamics Society, Nijmegen. 7, 9, 18

Richardson, G. P.: 1996, Problems for the future of system dynamics, *System Dynamics Review* **12**(2), 141–157. 2

Richardson, G. P. and Pugh, A. L.: 1981, *Introduction to System Dynamics Modelling with DYNAMO*, Productivity, Cambridge, MA. 2

Saleh, M., Oliva, R., Kampmann, C. E. and Davidsen, P.: 2006, Eigenvalue analysis of system dynamics models: Another perspective, *Proceedings of the 2006 International System Dynamics Conference*, System Dynamics Society, Nijmegen. 2

Senge, P. M.: 1990, *The fifth discipline; the art and practice of the learning organization*, Doubleday, New York. 12

Sterman, J.: 1985, A behavioral model of the economic long wave, *Journal of economic behavior & organization* **6**(1), 17–53. 10

Sterman, J. D.: 2000, *Business Dynamics. Systems Thinking and Modeling for a Complex World*, Irwin McGraw-Hill, Boston, MA. 2

UML: 2008. Retrieved 11 Februari, 2008.
URL: *http://www.uml.org* 21

Yucel, G. and Barlas, Y.: 2007, Pattern-based system design/optimization, *Proceedings of the 2007 International System Dynamics Conference*, System Dynamics Society, Boston, MA. 17

A Specification

This appendix focusses on some of the details involved in specifying the Generalized Loop Deaactivation Method.

A.1 Formulating switch variables

In the previous formulation of the method, much of the analyst's time was invested in specifying control variables. Variables that, at a certain point in time, would deactivate a certain loop. These cost significant effort to formulate and, in addition, clutter up the equations. The automation uses a java-based representation of System Dynamics models[4] that mitigates the need for formulating the control variables.

In short, the full automation of Ford's behavioural apporach is made possible due to how variables obtain the values of other variables. At any moment in time, the only variables that have a value in a SD model are states and parameters. If any other variables wishes to determine its value, it needs to know the values of the variables it is dependent on, its predecessors, and use those to calculate its value. Consequently, to calculate the value, it needs to know the value of its predecessors, who face the same problem in their turn. In the implementation this results in a recursive method call over the graph of variables. The recursion stops as soon as it arrives at a variable with a known value; a state or a parameter.

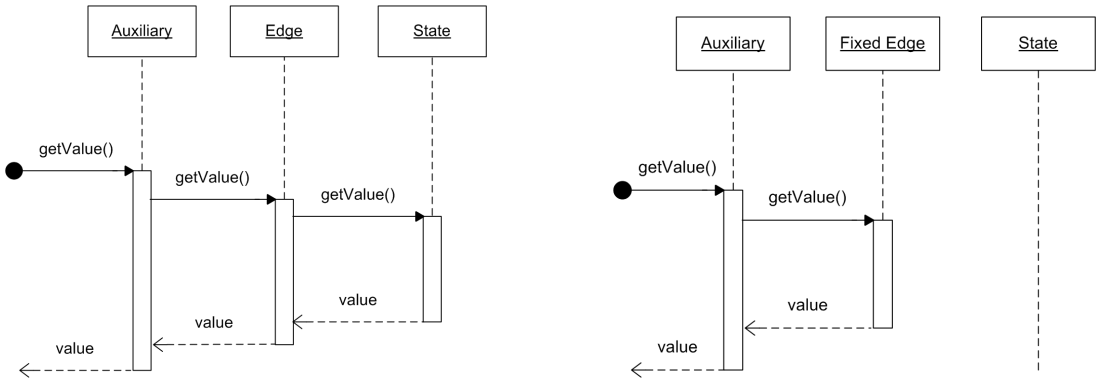
In our java-based representation, however, a variable never obtains the value of a predecessor directly, it asks the edge between the two vertices the value of the predecessor. For the analyst this opens up the possibility to manipulate the edge and consequently the value of the predecessor as it is used in the successor, without changing the predecessor itself. This is without introducing any control variable and it is a generic construction used for all connections between all variables. To complete the setup, every edge can be fixed to any given value; it returns the fixed value without calling the predecessor, effectively cutting the link between two variables. Figure 6 provides an overview of an auxiliary obtain the value of a state, both with and without the link from the state to the auxiliary fixed to a certain value.

A.1.1 Combinations of Loops

One of the strengths of Ford's behavioural approach is its ability to deal with multiple loops sharing dominance. The preferred method of deactivating combinations of loops, however, is labour intensive. In simple four loop model for instance, to test for shared dominance or shadow loops, the modeller would have to deactive 16 different combinations of loops for every time the analysis is executed, just to test all pairwise combinations. While this task is still feasible (if tedious) for a small model, even for medium sized models the costs in terms of effort and time would be disproportionate.

The implemented is solution to define what combinations of edges to deactivate in advance and to leave the actual deactivation to the method itself. Which combinations of loops to deactivate in what order is defined in the so-called deactivation matrix. The deactivation matrix represents what combinations of edges will be eliminated for each iteration of the method. The matrix is binary matrix of size $n \times m$, where n is the number of edges. One colum represents what edges of the selected edges are to be turned off for this iteration of the method.

For example, in an analysis using three edges, a 3×3 identity matrix signifies that, for every point of analysis in time, the analysis will only deactivate the individual edges in sequence, no combinations. In contrast, the



(a) Normal operation of the edge

(b) Operation if the edge is fixed

Figure 6: UML sequence diagrams (*UML* 2008) demonstrating the recursive method call used to obtain values. The boxes at the top are objects, the time axis is down. If a dotted line becomes a box, the object is active. Arrows indicate exchanged messages between the objects. The left diagram shows the sequence used in normal operation, with an active link. The auxiliary obtains the value of the state indirectly by calling a method of the edge. The right diagram shows the procedure for the same structure, but with a fixed edge. The auxiliary calls the method on the edge, but the edge never contacts the state and just returns the fixed values.

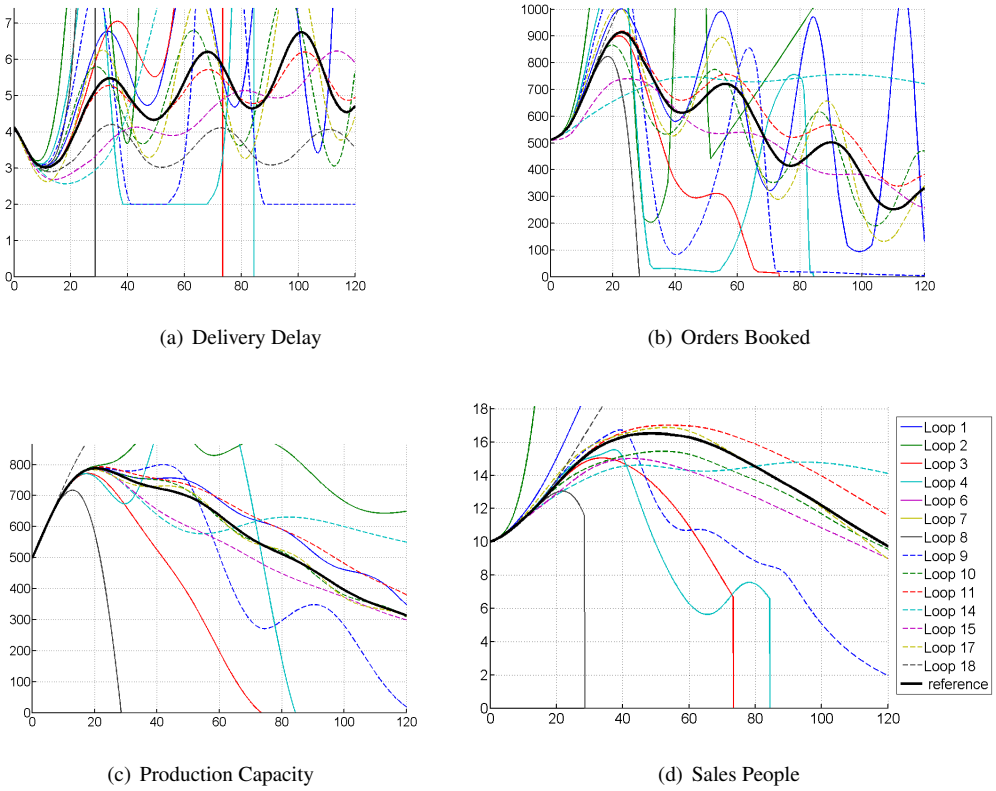
deactivation matrix D, where

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{2}$$

will first test each of the three edges individually, then all combinations of two edges and finally all edges together. The deactivation matrix can be used to specify searches for shadow loops as desired by the analyst. While formulating the matrix does not require much analyst-time, running the model, investigating a multitude of combinations may still take significant computer-time.

B Support for the Simplification of Market Growth

B.1 Results of Eliminating All Loops



(a) Delivery Delay

(b) Orders Booked

(c) Production Capacity

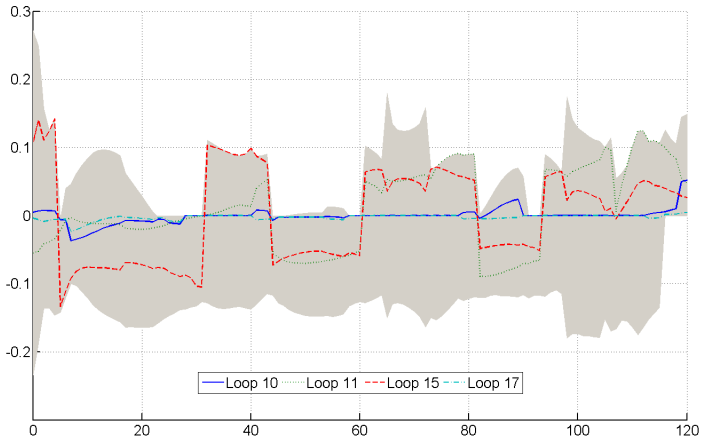
(d) Sales People

Figure 7: Graphs of the effects of eliminating all loops in Market Growth. The graph for Sales People shows the legend for which line represents the system with which loop eliminated.

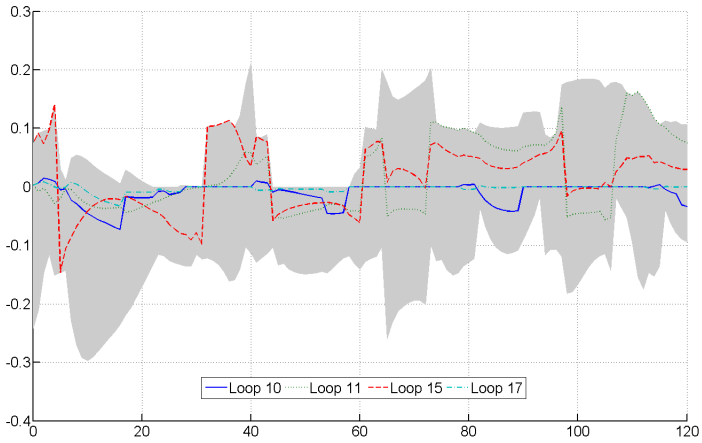
C Comparison with Eigenvalue Elasticity Analysis

To compare our results to a second method for formal model analysis, we analyze the Market Growth Model using eigenvalue elasticity analysis (EEA). The variant of EEA used is the one developed and published by Güneralp (2006), but fully automated and generalized. In setting up the method, we focus on the same two variables of interest as used in the previous analysis. The results can be found in Figure 8, the matlab code defining the analysis is Appendix D.3(line 14 to 26).

The two methods agree to some extent, but not in everything. While Loop 10 and Loop 17 stay fairly close to zero during the entire model run, we see that Loop 15 and Loop 11 are strongly influential over certain intervals. The influence of Loop 15 and 11, however, appears to alternate. It could be that, even though the loops are strongly influential at certain points in time, that their *net influence over time* is still low since the influence in a certain interval cancels out the effect built up in the previous interval. This raises questions on how to interpret the outcomes of the eigenvalue elasticity analysis and how they relate to the GLDM.



(a) Salespeople



(b) Production Capacity

Figure 8: Results EEA. The visible lines in the plot are the elasticities of the loops that are least influential according to their deactivation. The grey, shaded area shows the maximum elasticities at the point in time of all other loops in the Shortest Independent Loop Set (SILS).

D Matlab Code

This appendix contains the code for the analyses presented in this paper.

D.1 Dominance Analysis

```
function gldmYeastDemo()
%GLDMYEASTDEMO Short demo of the automated, generalised version of Ford

    import nl.tudelft.tbm.pa.sd.yeast.builder.*
    import nl.tudelft.tbm.pa.sd.support.*
    import java.util.Arrays

    % build and initialize the model
    yb = YeastBuilder();
    model = SDModelBuilder.buildModel(yb);
    model.setTime(0);
    model.initializeModel();

    % Define timespan and moments of analysis
    timespan = 0:0.1:90;

    % reportinterval to .1, normal ford with default selection
    analysisData.analysisFunction = @analyzeFord;
    analysisData.reportInterval = .5;

    % variables of interest
    analysisData.vois = selectVertex(model,'cells');

    % Store the initial values of the model
    inits = model.getStateValues();

    % Generate a reference run
    refSnap = 0:.5:90;
    [cellValues, analysisData] =...
    variableTracker(model, timespan, 0:.5:90, @ode1, analysisData);

    % Detect switches of atomic behaviour pattern.
    % Timestep is constant, so calculate gradient, gradient of the gradient
    % then multiply the signs of those to get the sign of the abp. Shift
    % the abp timeseries one place to find where it switched sign. Uses
    % these times as points of analysis.
    dx          = gradient(cellValues);
```

```
38      ddx          = gradient(dx);
39      abp          = sign(dx).*sign(ddx);
40      switches     = abp(2:numel(abp)).*abp(1:numel(abp)-1);
41      tSwitches    = refSnap(logical([0 switches<0]));
42
43      % Set the model to its original state again
44      model.setStateValues(inits);
45
46      % Set times at which to analyze, first one is of course t=0, the rest
47      % is from the pattern switches.
48      snapshots = [0 tSwitches];
49
50      % Run the analysis
51      % Results contains results, ad contains processed analysisdata (e.g.
52      % what loops were found, what was the deactivation matrix, etc.)
53      [results, ad] = formalAnalysis(model, timespan, snapshots,...
54          @ode1,analysisData);
55
56      % Plot the effects of eliminating all selected loops at the points
57      % of interest
58      % Make the graph pretty, colors, legends and a pink ribbon
59      % number of loops derived from deactivationMatrix
60      numLoops = size(ad.deactivationMatrix,2); % actually num deacts
61      cm = colormap(hsv(size(ad.deactivationMatrix,2)));
62      % Generate a legend for the graph
63      baseName = 'loop ';
64      baseName = repmat(baseName, [numLoops, 1]);
65      specLegend = [baseName num2str((1:numLoops)')];
66      specLegend = [specLegend; 'refrun'];
67
68      % Plot separate figures for different moments of analysis
69      for voiIdx = 1:numel(ad.vois)
70          for snapIdx = 1:numel(snapshots)
71              figure
72              hold on
73              for modIdx = 1 : size(ad.deactivationMatrix,2)
74                  allResults = results(snapIdx).modifiedRuns{modIdx};
75                  plot(results(snapIdx).reportVector, allResults(voiIdx,:)...
76                      , 'color', cm(modIdx,:),'linewidth', 2);
77                          grid
78                   % make the plot pretty
79                  set(gca, 'fontsize', 18);
```

```
                xlabel('t'); ylabel('\epsilon');
                set(gca, 'fontsize', 18);
                set(gca, 'Layer','top');
                set(gca, 'LineWidth', 2);
            end
            plot(0:.5:90,  cellValues, 'k', 'linewidth', 2);
            grid
            legend(specLegend);
        end
    end
end
```

D.2 The Role of One Loop

```
function gldmOneLoop()
%GLDMONELOOP Investigates the role of one loop in a model using the
% gldm
    import nl.tudelft.tbm.pa.sd.longwave.kampmann.*
    import nl.tudelft.tbm.pa.sd.support.*
    import java.util.Arrays

    lwk = LongWaveKampmann();
    model = SDModelBuilder.buildModel(lwk);
    model.setTime(0);
    model.initializeModel();

    % preliminary settings
    % The analysis only takes place in a short interval
    startAnalysis = 130;
    endAnalysis = 150;
    inits = model.getStateValues();
    timespan = 0:0.01:endAnalysis;

    % Generating a reference run
    % two variables of interest
    refad.vois = [  selectVertex(model, 'backlog'),...
                    selectVertex(model, 'relative orders')];
    % same interval as rest of analysis, different granularity
    refsnapshots = startAnalysis:.1:endAnalysis;
    % run with euler
    refBack = variableTracker(model, timespan,...
        refsnapshots, @ode1, refad);
    % reset model
```

```
30      model.setStateValues(inits);
31
32      % SelectEdges for the Self Ordering Loop, same as in Ford, 1999
33      L14edge         = selectEdge(...
34          model.getEdges.toArray, 'model.desired production',...
35                                  'model.desired cap');
36
37      % define way of eliminating edge, only applicable to this edge
38      % uncomment the lines below to make the analysis set the value
39      % of the link to steady state.
40  %    fixL14edge = @(edge)edge.freeze(1.8);
41  %    ad.eliminateEdgeFunction = fixL14edge;
42
43      ad.nominatedEdges = L14edge;
44      ad.analysisFunction = @analyzeFord;
45      ad.reportInterval = .1;
46      ad.deactivationMatrix = 1;
47      ad.vois = selectVertex(model, 'backlog');
48      snapFix = startAnalysis:endAnalysis;
49      [resultsFix, lalala] = formalAnalysis(model, timespan, snapFix,...
50          @ode1,ad);
51      % Make a pretty plot. Define colormap so that later eliminations get
52      % a color higher in the hsv scale.
53      figure
54      hold on
55      cm = hsv(numel(snapFix));
56      for snapIndex = 1 : size(snapFix,2)
57          theRun = resultsFix(snapIndex).modifiedRuns{1};
58          plot(resultsFix(snapIndex).reportVector, theRun,...
59              'color', cm(snapIndex,:),...
60              'linewidth',1);
61          scatter(snapFix(snapIndex),theRun(1),30,cm(snapIndex,:),'filled');
62      end
63      % plot voi and relative orders in same figure
64      set(gca, 'LineStyleOrder', '-|--');
65      h = plot(refsnapshots, refBack, 'k', 'linewidth',2);
66
67      legend(h, 'backlog','relative orders')
68      axis([130 150 0 8]);
69      grid
70  end
```

D.3 Simplification

This code sets up both the eigenvalue elasticity analysis and the GLDM for the simplification

```
function gldmCompareEEAMG()
    import nl.tudelft.tbm.pa.sd.marketgrowth.subs.*;
    import nl.tudelft.tbm.pa.sd.support.*
    import java.util.Arrays

    % build the model
    subs = MGSubBuilder();
    model = SDModelBuilder.buildModel(subs);
    model.setTime(0);
    model.initializeModel();

    %store initial values
    inits = model.getStateValues();
    timespan = 0:0.1:120;
    snapshots = [0:1:120];

    % First, do the guneralp-variant of eigenvalue elasticity analysis
    % setup the analysis
    ad.analysisFunction = @analyzeGuneralp;
    % variables of interest
    ad.vois =...
        [selectVertex(model, 'model.sam.Salespeople'),...
         selectVertex(model, 'model.prod.Production Capacity'),...
         ];
    % execute
    [results, resad] = formalAnalysis(model, timespan, snapshots, @ode4, ad);

    % Process the results to plottable matrices
    salesPeopleIdx =...
         find(cellfun(@(state)state==ad.vois(1),...
         cell(model.getStates().toArray)));

    prodCapIdx =...
         find(cellfun(@(state)state==ad.vois(2),...
         cell(model.getStates().toArray)));

    resMatrix = zeros(numel(snapshots),10,18);
    for idx = 1:numel(snapshots)
        resMatrix(idx,:,:) = results(idx).overRes;
```

```
40          absMaxPos(idx,:) = max(squeeze(resMatrix(idx,:,:)),[],2)';
41          absMaxNeg(idx,:) = min(squeeze(resMatrix(idx,:,:)),[],2)';
42      end
43
44      % reset the model
45      model.setStateValues(inits);
46      % select only t=0 as a moment of analysis
47      fsnapshots = [0];
48      % reportinterval to .1, normal ford with default selection
49      fad.reportInterval = .1;
50      fad.analysisFunction = @analyzeFord;
51      % variables of interest
52      fad.vois =...
53          [selectVertex(model, 'model.sam.Salespeople'), ...
54           selectVertex(model, 'model.prod.Production Capacity')];
55
56      % Generate a reference run, tracking variables of interest
57      frefAd.vois = fad.vois;
58      [frefrun] = variableTracker(model, timespan, timespan, @ode4, frefAd);
59      model.setStateValues(inits);
60
61      % Run the formal analysis
62      [fresults, fresAd] = formalAnalysis(model, timespan, fsnapshots,...
63          @ode4,fad);
64
65      % Here be dragons. Uhm, plotting instructions...
66      % This constitutes more than half the code, but the generates the
67      % exact plots in the paper.
68      % Plot the effects of eliminating all selected loops
69
70      for associatLoop = 1:numel(fresAd.associatedLoops)
71          loopnames{associatLoop} = ['Loop ',...
72               num2str(fresAd.associatedLoops{associatLoop})];
73      end
74
75      for voiIdx = 1:numel(fad.vois)
76          for snapIdx = 1:numel(fsnapshots)
77              figure('name',...
78                  char(fad.vois(voiIdx).getLocalName().toCharArray()'));
79              set(gca, 'NextPlot', 'replacechildren');
80              set(gca,'LineStyleOrder',{'-','--','-.'});
81              hold on
```

```
82              allResults = zeros(numel(fresults(snapIdx).reportVector),...
83                  size(fresAd.deactivationMatrix,2));
84              for modIdx = 1 : size(fresAd.deactivationMatrix,2)
85                  allVois = fresults(snapIdx).modifiedRuns{modIdx};
86                  allResults(:,modIdx) = allVois(voiIdx,:);
87              end
88              plot(fresults(snapIdx).reportVector,...
89                      allResults, 'LineWidth',2);
90              legend(loopnames,'Location','EastOutSide');
91              plot(timespan, frefrun(voiIdx,:), 'k', 'linewidth', 2);
92              set(gca, 'XLim', [0 120]);
93              set(gca, 'YLim', [0 max(frefrun(voiIdx,:))*1.1]);
94              grid
95              set(gca,'LineWidth',2);
96              set(gca,'FontSize',20);
97          end
98      end
99
100
101     % make two plots of the EEA for each voi. First the raw data, then the
102     % non-interesting loops shaded
103
104     for loopIdx = 1:size(resad.dcm, 2)
105         loopnames{loopIdx} = ['Loop ',...
106              num2str(loopIdx)];
107     end
108
109      figure
110      set(gca, 'NextPlot', 'replacechildren');
111      set(gca,'LineStyleOrder',{'-','--','-.'});
112      ph = plot(snapshots,...
113          squeeze(resMatrix(:,salesPeopleIdx,:)), 'LineWidth',2);
114      legend(ph, loopnames, 'Location','EastOutside');
115      set(gca, 'linewidth',2);
116      set(gca, 'FontSize',20);
117      grid
118
119      figure
120      hold on
121      set(gca, 'linewidth',2);
122      set(gca, 'FontSize',20);
123      grid
```

```
124       arh = area(snapshots, absMaxPos(:,salesPeopleIdx));
125       arh2 = area(snapshots, absMaxNeg(:,salesPeopleIdx));
126       set(arh, 'FaceColor', [.7 .7 .7]);
127       set(arh2, 'FaceColor', [.7 .7 .7]);
128       h= plot(snapshots,...
129           squeeze(resMatrix(:,salesPeopleIdx,[10,11,15,17])));
130       legend(h, ['Loop 10';'Loop 11';'Loop 15';'Loop 17']);
131
132       figure
133       set(gca, 'NextPlot', 'replacechildren');
134       set(gca,'LineStyleOrder',{'-','--','-.'});
135       ph2 = plot(snapshots,...
136           squeeze(resMatrix(:,prodCapIdx,:)), 'LineWidth',2);
137       legend(ph2, loopnames, 'Location', 'EastOutSide');
138       set(gca, 'linewidth',2);
139       set(gca, 'FontSize',20);
140       grid
141
142       figure
143       hold on
144       set(gca, 'linewidth',2);
145       set(gca, 'FontSize',20);
146       grid
147       arh = area(snapshots, absMaxPos(:,prodCapIdx));
148       arh2 = area(snapshots, absMaxNeg(:,prodCapIdx));
149       set(arh, 'FaceColor', [.7 .7 .7]);
150       set(arh2, 'FaceColor', [.7 .7 .7]);
151       h= plot(snapshots, squeeze(resMatrix(:,prodCapIdx,[10,11,15,17])));
152       legend(h, ['Loop 10';'Loop 11';'Loop 15';'Loop 17']);
153   end
```