

Loops that Matter

William Alexander Schoenberg

Thesis for the degree of Philosophiae Doctor (PhD)
University of Bergen, Norway
2020

UNIVERSITY OF BERGEN



Loops that Matter

William Alexander Schoenberg



Thesis for the degree of Philosophiae Doctor (PhD)
at the University of Bergen

Date of defense: 06.11.2020

© Copyright William Alexander Schoenberg

The material in this publication is covered by the provisions of the Copyright Act.

Year: 2020

Title: Loops that Matter

Name: William Alexander Schoenberg

Print: Skipnes Kommunikasjon / University of Bergen

Scientific environment



This doctoral work has been conducted under the supervision of:

- Professor: Dr. Pål Davidsen, University of Bergen, Department of Geography
- Co-President: Dr. Robert Eberlein. isee systems inc. Lebanon New Hampshire
- Professor: Dr. Birgit Kopainsky, University of Bergen, Department of Geography

Acknowledgements

There are many people to whom I owe a great debt of gratitude to. Without their help and support this work would never have existed. I thank Pål Davidsen, Bob Eberlein, and Birgit Kopainsky for always being there for me and continuously giving to me the benefits of their decades of experience, not only as system dynamicists but as caring and compassionate individuals. I also thank Bobby Powers. It was Bobby's immediate support and continual willingness to modify his personal work to suit my needs that this project ever moved farther than a daydream on long car ride home while stuck in traffic.

Abstract

Understanding the origins of behavior in time dependent, non-linear, dynamic mathematical systems with feedback is of critical importance for doing causally accurate science. The modern-day formal methods of loop dominance analysis used to understand the structural foundations of behavior within these systems struggles to find a foothold within the system dynamics community of practitioners. Within the machine learning field, discovering and measuring causality within feedback systems is a challenge which does not yet have a solution. The modern-day methods within the machine learning field struggle with the discovery of causal structure in directed acyclic graphs.

This PhD presents two linked methods for solving these challenges. The first ‘Loops that Matter’ is a formal method of loop dominance analysis which is capable of discovering the strength and polarity of each feedback loop in any time dependent mathematical system. The second, ‘Feedback System Neural Network’, is an ordinary differential equation based machine learning method capable of discovering causal relationships in non-linear dynamic feedback rich systems directly from data using key mathematical constructs developed as a part of Loops that Matter.

In addition, this PhD discovers, develops and matures the technology to machine generate high quality animated simplified causal loop diagrams of model structure based upon the Loops the Matter method. All technology has been matured to the point of inclusion in the commercial software application Stella v2.0 which enjoys widespread use throughout the scientific and educational communities related to system dynamics.

This PhD will walk the reader through the theories underlying the design of the methods, the implementation of all the aforementioned methods and the necessary challenges which were solved in order to make the methods widely applicable and usable by those within the scientific and educational communities.

Abbreviations

ANN – Artificial Neural Network

CLD – Causal Loop Diagram

DDWA – Dynamic Decomposition Weight Analysis

EEA – Eigenvalue Elasticity Analysis

FSNN – Feedback System Neural Network

ILS – Independent Loop Set

LEEA – Loop eigenvalue elasticity analysis

LTM – Loops that Matter

ML – Machine Learning

MLP – Multilayer Perceptron Network

PPM – Pathway Participation Metric

ODE – Ordinary Differential Equation

SD – System Dynamics

SDA – Structural Dominance Analysis

SFD – Stock Flow Diagram

SILS – Shortest Independent Loop Set

List of Publications

- Schoenberg, W., Davidsen, P., & Eberlein, R. (2020). Understanding model behavior using the loops that matter method. *System Dynamics Review* accepted for publication.¹
- Schoenberg, W. (2019). LoopX: Visualizing and understanding the origins of dynamic model behavior. *arXiv preprint arXiv:1909.01138.*²
- Schoenberg, W. (2019). Feedback System Neural Networks for Inferring Causality in Directed Cyclic Graphs. *arXiv preprint arXiv: 1908.10336.*²
- Eberlein, R. & Schoenberg, W. (2020). Finding the Loops that Matter. *arXiv preprint arXiv:2006.08425.*²
- Schoenberg, W., & Eberlein, R. (2020). Seamlessly Integrating Loops That Matter into Model Development and Analysis. *arXiv preprint arXiv:2005.14545.*²

¹ Reprints were made with permission from John Wiley and Sons.

² Reprints were made with the permission of the authors.

Contents

Scientific environment.....	1
Acknowledgements.....	2
Abstract.....	3
Abbreviations.....	4
List of Publications	5
Contents.....	6
1. Introduction	8
1.1 <i>Problem Definition</i>	8
1.1.1 Performing automated loop dominance analysis in System Dynamics models.....	8
1.1.2 Visualizing and understanding automated loop dominance analyses in System Dynamics	10
1.1.3 Causal understanding in Machine Learning	11
1.2 <i>Research Questions</i>	12
2. Review: Prior Art in Loop Dominance Analysis & Measuring Causality in System Dynamics	14
2.1 <i>The 'Art of Model Analysis'</i>	14
2.2 <i>Eigenvalue Elasticity Analysis</i>	15
2.2.1 The advantages of EEA in loop dominance analysis.....	18
2.2.2 The disadvantages of EEA in loop dominance analysis	18
2.3 <i>The Pathway Participation Metric Based Approaches</i>	21
2.3.1 The advantages and disadvantages of PPM based approaches & the Loop Impact method....	22
3. Summary of Articles: The Loops That Matter Method	24
3.1 <i>Discovery: The Underlying Mathematics</i>	24
3.1.1 LTM metrics.....	25
3.1.2 Conclusions on the mathematics	28
3.2 <i>Refinement: How to Choose Which Loops to Analyze</i>	29
3.3 <i>Prototype Implementation: Visualization & Understanding</i>	31
3.4 <i>Production Implementation: The Challenges of Building a Production Ready System</i>	35
3.4.1 A major methodological / implementation challenge.....	36
3.4.2 Improvements and advancements in the visualization technology.....	36

3.4.3	Conclusions on the production implementation.....	38
3.5	<i>A Cross-Disciplinary Application: Integrating the Link Score into New Machine Learning Method for Automated Causal Inference.....</i>	39
3.5.1	An introduction to neural networks.....	40
3.5.2	The new machine learning method for automated causal inference	42
4.	Conclusions.....	45
5.	References.....	47
6.	Article #1: Understanding model behavior using loops that matter.....	51
7.	Article #2: LoopX: Visualizing and understanding the origins of dynamic model behavior	98
8.	Article #3: Feedback System Neural Networks for Inferring Causality in Directed Cyclic Graphs	128
9.	Article #4: Finding the Loops that Matter	149
10.	Article #5: Seamlessly Integrating Loops That Matter into Model Development and Analysis	171

1. Introduction

This thesis presents a new technique, loops that matter (LTM), for understanding the origins of behavior in any time dependent mathematical model including highly non-linear, feedback rich, ordinary differential equations (ODEs).

The concept of a time dependent mathematical model referred to in this thesis is a system of equation(s) such that all equations provide the necessary information to calculate values for all variables enumerated over a specific and restricted singular (typically time) dimension. The system of equations may be differentiable (or not), may contain feedback (or not), and may be continuous, (or not). As long as the system provides a mathematically calculable causal structure to evaluate, the system meets the requirements for analysis as discussed in this thesis.

1.1 Problem Definition

At its highest level, this thesis is about discovering and quantifying the nature of true causal relationships in complex dynamic systems. This has arguably been the singular fundamental focus of scientific discovery since the ancient Greeks. This thesis develops and applies the LTM approach for measuring and quantifying causation within the System Dynamics (SD) and Machine Learning (ML) fields. Both fields struggle in their own way, each with their own unique set of diverse processes, to understand the singular set of causal drivers behind problems across a wide variety of subject area domains. At its core, this thesis presents the necessary mathematical and algorithmic techniques required to automate the process of bringing into the light the key causal mechanisms driving dynamic behavior across a wide spectrum of mathematically rigorous modeling techniques encompassing both fields and their constituent methodologies.

1.1.1 Performing automated loop dominance analysis in System Dynamics models

Within the field of System Dynamics, one of its holy grail pursuits is a software system capable of performing loop dominance analysis and presenting the results in

the form of animating Stock Flow Diagrams and/or Causal Loop Diagrams (SFD, CLD) (Sterman, 2000).

Starting from the beginning and from a high level perspective, the System Dynamics process has the practitioners build a structure which they feel adequately represents the problem at hand, after which they work to understand how that structure functions, what the key drivers of behavior are. Finally they deploy that understanding to develop policies to solve the problem, all in a highly iterative fashion (Forrester, 1994). Key to that process is model understanding, and without that step, effective policies cannot be discovered and employed (Sterman, 2000).

The current state of the art in the System Dynamics field relies on either practitioner intuition and experience (the art of modeling and model analysis) or complex algorithmic analysis. The former is taught as part of the methodology of model building, while the latter comes from 40 years of work on techniques to derive and explain model behavior based on the analysis of structure (see for example: Graham, 1977; Forrester 1982; Eberlein, 1984; Davidsen, 1991; Mojtabahedzadeh, 1996; Ford, 1999; Saleh, 2002; Mojtabahedzadeh et al., 2004; Goncalves, 2009; Saleh et al., 2010; Kampmann, 2012; Hayward and Boswell, 2014; Moxnes and Davidsen, 2016; Oliva, 2016; Sato, 2016; Hayward and Roach, 2017; Naumov and Oliva, 2018).

Ford (1999, p.4-5) most clearly states the needs of the System Dynamics field as it relates to loop dominance analysis:

“To rigorously analyze loop dominance in all but small and simple models and effectively apply analysis results, system dynamicists need at least two things: (1) automated analysis tools applicable to models with many loops and (2) a clear and unambiguous understanding of loop dominance and how it impacts system behavior.”

Loop dominance analysis sheds light on the origins of behavior in SD models by relating observed behavior back to the feedback process(es) that created it (Forrester 1961; Richardson 1991). Loop dominance analysis is concerned with the discovery of

the strength and polarity of the key feedback loops existent within SD models as time progresses within those models (Richardson, 1995). By understanding the relationships between the key feedback loops within model, practitioners can then more easily identify the key leverage points within the system which may be used as the targets for policy (Saleh et. al, 2010).

1.1.2 Visualizing and understanding automated loop dominance analyses in System Dynamics

Notwithstanding the problems with discovering a methodology to do loop dominance analysis, there are also a wide ranging set of problems related to the visualization and understanding of the results from loop dominance analyses. In the final chapter of “Business Dynamics” Sterman, (2000) identifies three critical attributes of software for the future of SD:

- Automated identification of dominant loops and feedback structure
- Visualization of model behavior
- Linking behavior to generative structure

Automated identification of dominant loops and feedback structure follows along with the methodological developments supporting loop dominance analysis, for this thesis, namely LTM. Visualization of model behavior is concerned with improvements to graphics and animations necessary to make clear the full dynamics of large, complex models with high dimensional and parametric spaces. Finally linking behavior to generative structure is concerned with automatically displaying the results of loop dominance analyses directly on the SFD or CLDs representing model structure to make clear the relationships between key structures and observed behaviors.

Without visualization and communication of the results of an analysis, there is not understanding, and with improvements in visualization comes strong benefits for understanding. Any tool purporting to unlock the knowledge held within the mathematical structure of models, must present some form of user interface for relaying that insight to practitioners. Without such a tool, the knowledge generated by the loop dominance analysis becomes itself locked away out of grasp of the

practitioner. And when graded against increased understanding the analysis becomes just as useful as the raw structure of the system of equations. The minimal complement of visualization techniques relies on standard graphs and table-based data visualizations of understandable and at some level intuitive metrics. The complexity of the underlying metrics, and the interpretability of the generated visualizations (explanations) is the driving force behind the general utility and relevance of the technique under evaluation. While simple graphs and tables do perform the necessary function of relaying the insight gleaned from the loop dominance analysis, they do not present the same intuitive user experience that is being called for by Sterman (2000).

1.1.3 Causal understanding in Machine Learning

Modern day machine learning methods including probabilistic modeling, kernel machines or deep learning, have arisen from a strong, almost single minded, focus on empiricism making use of the extraordinary amounts of observational data available from a plethora of sources (Ghahramani, 2015; Schölkopf & Smola, 2008; Goodfellow et. al., 2016). When viewed through the lens of accurate prediction power, machine learning has proven to be quite successful, but present-day machine learning techniques typically fail to reveal the fundamental causal mechanisms driving behavior. Although, it cannot be ignored that interpreting the structure behind black box, deep learning models is an active research area (Montavon et. al, 2018). To make full use of the technological advancements in machine learning significant emphasis must be placed on finding a valid and interpretable causal understanding of the underlying real-world system (Runge et. al, 2019).

The field of observational causal inference exists alongside the ML field and is focused on drawing conclusions about causal connections between variables by studying the response in an effect variable when a cause is being changed (Pearl, 2009). Largely based on statistics, this field has been growing and developing since the 1950s. Observational causal inference as a field started with the seminal works of Wiener and Granger (Wiener, 1956; Granger, 1969). The most well-known method for observational causal inference is the Granger causality which tests whether omitting the past of a time series X in a time series model including Y 's own and other

covariates' past increases the prediction error of the next time step of Y (Granger, 1969). The Granger causality is useful in discovering specific causal links in a system, but it fails to generalize to complex non-linear systems, typically failing to identify all of the links in the networks of feedback relationships which govern these systems from a wholistic perspective (Spirtes & Zhang, 2016).

Non-Granger methods in observational causal inference can be categorized into the following three broad categories as done in the literature by Runge et. al (2019): nonlinear state-space methods, causal network learning algorithms and structural causal model frameworks. All of these approaches are designed to discover causality in directed acyclic graphs and are incapable of operating in complex dynamic feedback rich environments which the System Dynamics literature argues is at the core of the most intractable and relevant problems (Sterman, 2000).

Taking the current state of the art in ML, and automated observational causal inference into account, it becomes clear to see that the ML field lacks a direct understanding of the relevance of the causal structures underlying their models to real world systems.

1.2 Research Questions

Taking the full weight of the enumerated problems above, the following two research questions are presented.

1. How can the origins of behavior be algorithmically discovered in any time dependent mathematical model, discrete or continuous, linear or non-linear, feedback rich or not?
2. How can the origins of behavior in any mathematical system be visualized, animated and simplified such that practitioners can most easily understand the relationships between the causal mathematical structure of models and observed behavior?

The purpose of these questions is to stimulate and guide the development of a methodology required to do loop dominance analysis in concert with machine learning based causal inference. Along with the development of the methodology, these questions call for the construction of a technology to both employ the methodology and to visualize the results in a production ready, practitioner usable manner capable of operating on a highly diverse set of input models.

2. Review: Prior Art in Loop Dominance Analysis & Measuring Causality in System Dynamics

Properly understanding the context which surrounded the creation of LTM requires a proper understanding of the over 40 years of research that has already happened within the SD field on loop dominance analysis, and particularly automated loop dominance analysis. This thesis uses the term loop dominance analysis as opposed to the presently more in-vogue term, structural dominance analysis to discuss prior art. This thesis classes loop dominance analysis as a strict subset of structural dominance analysis where the only difference is loop dominance analysis implies that the major element of the presented analysis is the feedback loop as opposed to any other divisible aspect of model structure such as a causal pathway (chain of variables) or an individual link.

This thesis divides prior art into three major categories, each presented in their own section below. The first, ‘Art of Model Analysis’ referrs to the non-automated approaches and methodologies developed to study patterns of loop dominance within SD models both before and after the advent of the automated processes. The second and third each refer to a unique, pre-established automated process for doing the same. The LTM approach forms a third, unique approach to automating loop dominance analysis and is covered in Chapter 3.

2.1 The ‘Art of Model Analysis’

Because loop dominance analysis is such an important part of the modeling process, the work of performing analyses to understand the origins of behavior in SD models has had to have been done, even if by hand. These by-hand processes, which this thesis terms the ‘Art of Model Analysis’ is what has been practiced by the large majority of practitioners since the dawn of the field. Even though practitioners have not had an arsenal of readymade automated tools at their disposal, they also haven’t had a complete lack of process to follow. The problem with these approaches is that they have varying levels of subjectivity and require years of experience and intuition

to develop the necessary skills to execute on them at a highly proficient level and can potentially yield erroneous conclusions if misapplied (Oliva, 2016).

Practitioners have for the most part used a formulaic, if not, rigorous exploration of their models via parameter tweaking in repeated simulations either amplifying or reducing feedback loop gains, loop cutting, and other methods of causal tracing to understand how their models function (Richardson 1986; Richardson 1991). A second set of less used approaches consist of model reduction or simplification practices which seek to retain specific behavior modes when removing discrete chunks of model structure as measured intuitively or statistically (Eberlein, 1984; Richardson, 1986). A third set of more rigorous numerical techniques is based around iterative sensitivity-like analysis which forms the ‘Reality Check’ feature of the Vensim software as discussed by Peterson and Eberlein (1994). The ‘Reality Check’ feature couples model inputs to expected behavior patterns, and the software then runs numerous parameterizations of the model checking for adherence to the behavioral expectations. The final major piece of research in this track of by-hand model analysis is Ford’s (1999) behavioral approach to feedback loop dominance which sets out a rigorous non-automated method for identifying dominant loops over the course of a single simulation. This is done by correlating behavior patterns to structural mechanisms that are known capable of producing the identified behavioral modes in a single time series.

2.2 Eigenvalue Elasticity Analysis

The first major effort towards automated loop dominance analysis techniques is today referred to in its general form as Eigenvalue Elasticity Analysis (EEA). N. Forrester in his PhD thesis of 1982 first documented that eigenvalue elasticities are usable to explain the relative gains of feedback loops in linear systems. Ever since, significant effort has been expended to further develop the technique. EEA is today focused on determining how model structure produces differing weighted combinations of behavior modes (Saleh, 2002), (Kampmann et al., 2006), (Saleh et al., 2010), (Oliva, 2016).

Oliva best summarizes and defines EEA in his 2016 paper which is quoted below:

Eigenvalue elasticity analysis (EEA) is a set of methods to assess the effect of structure on behavior in dynamic models (Kampmann, 2012; Kampmann and Oliva, 2006; Oliva, 2015). It works by considering model behavior as a combination of characteristic behavior modes and assessing the relative importance of particular elements of system structure in influencing these behavior modes....EEA uses linear systems theory to (i) decompose the observed behavior into its constituent behavior modes, such as oscillation, growth and exponential adjustment; and (ii) outline how a particular behavior mode and its appearance in a given system variable depend upon particular parameters and structural elements (links and loops) in the system.

The process for performing EEA based analyses in their various forms is represented in the following high-level schematic (Figure 1) reproduced from Naumov and Oliva (2018).

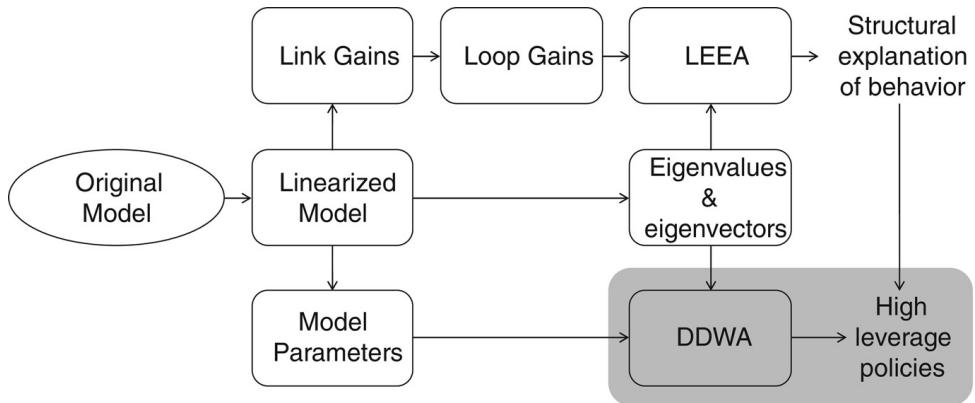


Figure 1: Process diagram demonstrating the various steps in performing an EEA analysis. Reproduced directly from Naumov and Oliva 2018.

Starting with an original model, the first step is linearization. Using the linearized model, the eigenvalues and eigenvectors of the system matrix are calculated. Loop eigenvalue elasticity analysis (LEEA), a specific method for determining loop

dominance, is performed using the eigenvalues and eigenvectors. LEEA relates model structure to behavior, produced by that structure, by measuring the influence of the model's feedback loops on the behavior modes of a selected set of stocks (Oliva, 2015). There is also dynamic decomposition weight analysis (DDWA) which is performed to identify leverage points within the system, by discovering how the model parameters affect the projections for the behavior modes in the stocks of interest (Saleh et. al, 2010).

To date the largest model analyzed using the EEA toolset, now called the structural dominance analysis (SDA) toolset (Naumov and Oliva, 2018) is Oliva's 2016 LEEA and DDWA of his own service quality model. In that work, Oliva expanded the scope of the methodology such that it became capable of working with larger models including stochastic variance. Naumov and Oliva also expanded the SDA toolset, so that is capable of performing DDWA at any time in the simulation, rather than just at the initial time (Naumov and Oliva, 2018).

To perform a LEEA (or any loop dominance analysis) the feedback complexity of models needs to be reduced such that the process of finding all the loops to analyze does not overwhelm the computational capability of the computer. The problem arises as the number of stocks in the model increases. Assuming a maximally connected model, where each stock directly affects each other in the model, then the feedback complexity of the model scales factorially. Consequently, a 10 stock, maximally connected model, with just a single causal pathway connecting each directed pair of stocks, would contain 3,628,800 feedback loops, while a 20 stock model with the same property contains $2.432902e+18$ feedback loops (Kampmann, 2012). To manage this problem Kampmann (2012) developed the concept of an independent loop set (ILS) and an associated algorithm to identify such a loop set. The ILS maps all of the loops in the model onto a singular set of selected, 'independent' loops which can be used to reproduce the full feedback structure of the model, while keeping the number of feedback loops to be enumerated and analyzed at a manageable level. Typically, the size of the ILS linearly scales with the number of variables. The ILS is determined before any simulation or measurement of link

strength or loop strength is completed. Oliva (2004), expanded upon the ILS, developing the concept of the shortest independent loop set (SILS) which makes the ILS deterministic by only including the geodetic, shortest, feedback loops necessary to rebuild the full feedback structure of the model. The SILS is the de-facto standard for determining which loops to analyze in a LEEA analysis and is at the basis of most EEA analyses.

2.2.1 The advantages of EEA in loop dominance analysis

EEA methods are the most encompassing ones. Their potential for enhancing our understanding of complex, dynamic systems surpasses other methods, - including LTM. EEA is a general analytical method which is capable of analyzing the behavioral state space of a model and speaks not only to the behavior modes observed in simulation results with a single set of initial conditions and parameters, but also to behavior modes that the model may produce under a different, but related set of initial conditions and parameters. Additionally, according to Duggan and Oliva (2013) EEA provides a specific, complete and correct accounting of the relationships between model structure and model behavior.

Oliva (2016) characterizes his implementation of the EEA analyses on his service quality model as efficacious, efficient and effective. Oliva (2016) states the methods are efficacious, meaning they work, producing results even in larger models (13 stocks) and models with stochastic variance. Oliva (2016) states that the method is efficient because the analysis of the service quality model took significantly less time to do using EEA than by hand via the ‘Art of Model Analysis’ techniques. Finally, Oliva (2016) states the method is effective because the results make sense, matching those resulting from the application of traditional model analysis techniques.

2.2.2 The disadvantages of EEA in loop dominance analysis

There are a variety of weaknesses in the EEA method which come part and parcel with the strengths. The EEA method is mathematically complex. Proficiency in the application of this method and the interpretation of its results requires a deep understanding of linear algebra and an appreciation of the mechanisms underlying the

analysis. Next, EEA requires the linearization of models during analysis, and the current SDA toolset (Naumov and Oliva, 2018) further requires that models be made continuous, and places restrictions on the usage of macros (both builtins and user generated) and does not operate on arrays. Finally, there are significant flaws with the design and construction of the SILS (and ILS), that are used to choose which loops to analyze using LEEA (Güneralp, 2006; Huang et. al, 2012).

The process by which models are made ready for analysis by the SDA toolset changes model results and impacts the large majority of models developed and used by SD practitioners. When Oliva analyzed his service quality model he had to remove an unchanging stock in order to produce a full rank system matrix, remove all smooth and delay function, remove all uses of MIN or MAX functions, remove all IF THEN ELSE statements, and remove all table functions. The MIN, MAX, IF THEN ELSE and table functions were replaced by continuous approximations of their original counterparts, and the smooth and delay functions expanded by hand, but regardless there were noticeable, and therefore significant changes to the behavior of the model analyzed as compared to the original model and this can be seen in Figure 2 which is a direct reproduction of Fig. 2 from Oliva 2016.

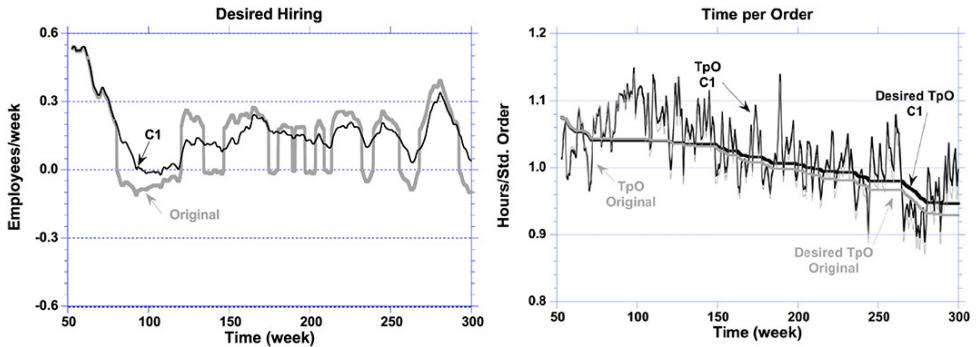


Figure 2: Behavior difference between original and continuously differentiable model as analyzed by Oliva (2016). This figure is a direct reproduction from Oliva 2016.

The problems with the ILS and SILS are more insidious. Any solution to the problem of factorial feedback complexity in dynamic systems definitionally cannot be based on a static analysis of the network structure of model equations, because the network

structure of model equations does not on its own encode enough information about the observed dynamics of the model to inform the end-user of the most important feedback relationships to generated behavior. Only with the inclusion of the parameter values, and by actually producing the simulation results can a model inform the end user of the nature of the relationships between the feedback loops and the observed behavior. The ILS and SILS are putting the cart before the horse, trying to simplify the feedback loops to analyze before even understanding which feedback loops have value towards the understanding of a specific behavioral outcome.

Huang et. al., (2012) criticize the SILS in particular for not identifying, and therefore LEEA for not analyzing the major, dominant reinforcing feedback loop in the example system picture below in Figure 3.

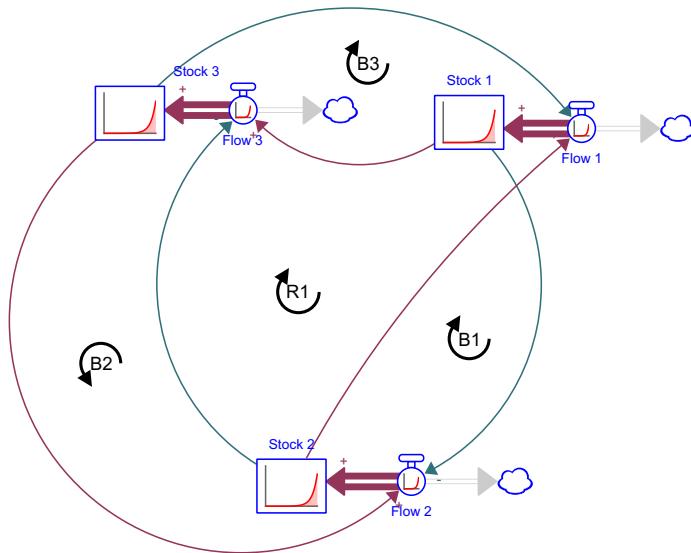


Figure 3: Feedback model which demonstrates the flaw with the SILS. The SILS identifies all balancing loops but fails to identify the reinforcing loop R1. This model produces exponential growth which is only capable of being produced by R1.

A repetition of the same criticism of the SILS approach comes from Güneralp (2006) who uses the SILS procedure in the predator-prey model showing how in a specific common construction of it, that the SILS fails to detect the second-order loop “that is critical in connecting the two populations”. Güneralp says that, “Obviously, the SILS

approach needs to be further scrutinized on low-order as well as high-order, dense, non-linear systems where model reduction tools such as this are most needed”.

2.3 The Pathway Participation Metric Based Approaches

The pathway participation metric (PPM) based approaches don’t use linear algebra, and linear systems analysis as their takeoff point. PPM as originally formulated by Mojtabahedzadeh (1996) focuses on the links between variables in a model and uses its understanding of those relationships to build up towards the impacts of causal pathways, or potentially feedback loops on stocks of interest (Mojtabahedzadeh et. al., 2004). The PPM approach went through a reinvention process with the work of Hayward and Boswell (2014) and the development of the ‘Loop Impact Method’.

In Mojtabahedzadeh’s (2004) implementation of PPM in the tool Digest, structural dominance analysis is performed for each phase of behavior for a selected stock. The tool, Digest, takes the time series data from a user specified variable of interest and slices it into n pieces where each piece demonstrates one of the key 7 patterns of behaviors by making partitions such that the slope and convexity (i.e. the first and second time derivatives do not change sign). For each slice, a single dominant pathway (causal chain, from an ancestor stock, or exogenous variable to the stock of interest) is chosen as being responsible. The PPM itself is presented in Equation 1 for the link from the variable x to the variable z .

Equation 1: The pathway participation metric for the link $x \rightarrow z$

$$PPM = \frac{\partial f}{\partial x} \cdot \frac{\dot{x}}{\dot{z}}$$

The total pathway participation metric for all pathways into the singular net flow of a stock is apportioned out to each individual pathway affecting that net flow, and the dominant pathway is the one with the highest magnitude which has the same sign as the net change of the variable under consideration. The contribution of each pathway is calculated by introducing small, minute changes to the ancestral stock values. The Digest implementation of the PPM approach may select a single exogenous causal

pathway as dominant, or potentially feedback loops if the analysis is performed for a series of stocks which are all interconnected. A dominant feedback loop is found by tying together all of the selected dominant pathways determined at each step in the calculation for each interconnected stock. The PPM is computed in time with the simulation model, is dependent upon the initialization of the stocks, and the chosen set of parameters. Unlike EEA based methods, PPM based methods only provide insight into the specific behavior modes observed over the course of a single simulation run.

2.3.1 The advantages and disadvantages of PPM based approaches & the Loop Impact method

The PPM approach brings with it many considerable advantages over the EEA based approaches. First, that it in theory requires no manipulation of model structure, nor is there a need to linearize or make continuous all equations in the model.

Mojtahedzadeh (2004) also claims that repeated applications of the PPM method as implemented in Digest will converge on a single unique piece of structure as the most influential for a specific time period for the stock under study. Kampmann and Oliva (2009) state that one of the key benefits to the PPM method is its direct connection between behavior and structure.

Kampmann and Oliva (2009) have criticized PPM for its inability to clearly explain oscillatory behavior and also because PPM may fail to identify structure when there are two pathways of similar importance (Kampmann and Oliva, 2009). Hayward and Boswell (2014) have responded to those criticisms by simplifying PPM into the Loop Impact Method. The Loop Impact Method can be implemented in a standard system dynamics model (and software) by adding equations to the model. No change in the underlying software is required. The key difference of the Loop Impact method as compared to PPM is that it does not look for dominant pathways (impacts from one stock to another), but instead focuses on the direct impacts that one stock has on another chaining together pathways following the structure of the model which measures loop impacts (a measure specifically created for this method which is related to the PPM) on a singular stock (Hayward and Boswell 2014). In addition, the Loop

Impact Method identifies instances where multiple loops are required to explain the behavior of a stock.

Expanding on the work done by Hayward and Boswell (2014), Sato (2016) has modified the Loop Impact Method specifically codifying the impacts of force in an engineering sense. Hayward and Roach (2017) have also developed a framework around the loop impact method couched in the mathematics of Newtonian physics, to explain the model as a series of interacting forces. The stated purpose of the underlying common research thread between these authors of the Loop Impact Method and its derivatives, is to provide a more intuitive and complete understanding of loop dominance in system dynamics models.

3. Summary of Articles: The Loops That Matter Method

This portion of the thesis will walk the reader through the five research articles which make up the research covered by this thesis. Each article was individually written for its own purposes, and each was placed in this order to guide the reader through the discovery, refinement, implementation and application of the LTM method and its visualization.

The LTM method, specifically the algorithm, and theories behind its development are the solution to Research Question #1. The method is capable of reporting on the origins of behavior in any time dependent mathematical model. Section 3.1 demonstrates the underlying mathematics of the LTM method, showing the reader how LTM is capable of determining the origins of behavior in time dependent mathematical models. Section 3.2 uses the link score metric introduced in section 3.1 to solve the problems with the ILS and SILS identified in section 2.2.2, which allows the LTM method to be broadly applied to large models with extreme levels of complexity, containing tens of millions or more feedback loops. Section 3.3 introduces the reader to the theories which underly the visualization and communication of the results of an LTM analysis directly answering Research Question #2. Next, Section 3.4, discusses the challenges which had to be overcome to generate a production ready implementation of the LTM algorithm and its associated visualization techniques in a commercial software product that is used widely within the field of SD. Finally, Section 3.5 that demonstrates the cross disciplinary application of the LTM methodology as a part of a broader machine learning technique to do automated causal inference in feedback rich non linear systems.

3.1 Discovery: The Underlying Mathematics

The first step in the process of developing the LTM method was to establish the fundamental mathematical basis which underlies its implementation as both an algorithm capable of analyzing models and a software tool capable of visualizing and

making clear the results of the analysis. This work is documented in Article #1 of this thesis which is in the appendix. The LTM method performs a formal assessment of dominant structure and behavior as categorized by Duggan and Oliva (2013). Like PPM based techniques, the LTM method is built around the observation of how modelers perform the art of model analysis to understand the origins of behavior. LTM interacts directly with the full network of model equations, walking the causal pathways between state variables (stocks), and calculating in time with the simulation, metrics that measure the contribution (ex: force, strength) and polarity of each link in the network of model equations. Unlike PPM and similar to LEEA, the LTM approach produces metrics which interpret the origins of behavior for the entire model¹ rather than just the behavior of a single state variable.

3.1.1 LTM metrics

LTM computes three new metrics which were first defined, presented, and defended in Article #1. The three metrics are the link score, the loop score and the relative loop score.

The link score

The first metric introduced was the link score. The link score is a measure of the contribution and polarity of any link in a model from an independent to dependent variable regardless of whether or not the link contains an integration process. The link score concept tracks the concept of the link gain, and when multiplied through pathways up until but not including the stock, is the same as Richardson's (1995) concept of the dominant polarity, and Kampmann's (2012) loop gain. What makes the link score unique from PPM based techniques is that it can also handle links which represent an integration process. The link score is computed once per each time interval in the model and is computed for each link in the model. There are two methods for calculating the link score depending upon whether the link contains an

¹ For cases where each stock in the model is able to either directly or indirectly impact each other stock in the model. For models where this assumption does not hold true LTM informs on the origins of behavior in each giant connected component of the model where this assumption holds true.

integration process or not. Article #1 demonstrates that the two methods produce exactly the same measure and therefore can both be referred to as the link score.

Below in Equation 2 is the definition of the link score from Article #1 of a link that does not contain integration from Article #1 assuming there are two inputs (x and y) to the dependent variable z characterized by the equation $z = f(x, y)$. The link score for the link $x \rightarrow z$ when written in a discontinuous form based upon the implementation of the calculation is (See Article #1 for continuous analytical form):

Equation 2: The discontinuous form for the link score equation which matches how the implementation of the calculation works moving in time with the dt of the model.

$$LS(x \rightarrow z) = \begin{cases} \left(\left| \frac{\Delta_x z}{\Delta z} \right| \cdot sign \left(\frac{\Delta_x z}{\Delta x} \right) \right), \\ 0, \quad \Delta z = 0 \text{ or } \Delta x = 0 \end{cases}$$

In Equation 2 Δz is the change in z from the previous time to the current time. Δx is the change in x over that same time step. $\Delta_x z$ is the change in z with respect to x . From a computational perspective $\Delta_x z$ which is called the partial change in z with respect to x , is the amount z would have changed, conditionally, if x had changed the amount it did, but y had not changed. The first major term in Equation 2 represents the magnitude of the link score, the second is the link score polarity.

For links which contain an integration process the equation for determining the contribution and polarity of the link from an inflow (i) or outflow (o) to a state variable (s) is reproduced from Article #1 as Equation 3.

Equation 3: Link score for all links from derivatives (flows) to state variables (stocks) (both inflows and outflows are covered). The simple one inflow and one outflow case is presented and is easily generalized.

$$\text{Inflow: } LS(i \rightarrow s) = \left(\left| \frac{i}{i - o} \right| * 1 \right) \quad \text{Outflow: } LS(o \rightarrow s) = \left(\left| \frac{o}{i - o} \right| * -1 \right)$$

The loop score and the relative loop score

The second key metric introduced in Article #1 is the loop score as shown in Equation 4. The loop score tracks the concept of the loop gain and is the result of the

multiplication of all link scores for all links in a loop. This is a demonstrably unique measure which bears some rough similarity to the Loop Impact metric of Hayward and Boswell (2014), but is unique because it is capable of including the links which contain integration processes allowing a single value to be assigned as the contribution of a loop.

Equation 4: Definition of loop score, for the loop x which contains n links for each source variable S to the target variables T .

$$\text{Loop Score}(L_x) = (LS(s_1 \rightarrow t_1) \cdot LS(s_2 \rightarrow t_2) \dots \cdot LS(s_n \rightarrow t_n))$$

The loop score is a dimensionless value which samples the effort a loop is expending to change the behavior of the stocks it connects at each calculation interval of the model. As the link score can be thought of as the force of an independent variable pushing on the result of a dependent variable, the loop score can be thought of as the force of one feedback loop pushing on the behavior of all the stocks (and therefore all variables) it connects.

The third and final key metric introduced in Article #1 is the relative loop score (Equation 5) which compares the contribution of feedback loops to determine which are dominant at any point in time. The relative loop score requires no independence across the loops it compares and ideally uses the exhaustive set of feedback loops as the basis for comparison if possible. Article #4 and section 3.2 cover the problem of feedback loop selection which was discussed in Section 2.2.2.

Equation 5: Definition of the relative loop score for the loop x normalized over all loops n analyzed in the chosen loop set.

$$\text{Relative Loop Score}_{L_x} = \left(\frac{\text{Loop Score}(L_x)}{\sum_{y=0}^n |\text{Loop Score}(L_y)|} \right)$$

The sign of a relative loop score represents the polarity of the feedback loop. The relative loop score is a normalized measure taking on a value between -1 and 1. It reports the polarity and instantaneous fractional contribution of a feedback loop to the change in value of all stocks in the feedback loop set it is a member of. By comparing loop scores, it can easily be determined which loops are dominant, i.e. contribute the

most (over 50%) to the behavior of all stocks in the feedback loop set under study. This normalization is critical to maintaining scores that are easy to work with.

3.1.2 Conclusions on the mathematics

Article #1 goes on to demonstrate the efficacy and effectiveness of the LTM method on three example models from the literature where it shows that LTM is capable of revealing the same depth of insight as any other formal method of loop dominance analysis including the PPM, Loop Impact and EEA methods without any of the complexity in mathematics or restrictions on either the scope of the analysis or the models analyzed that is imposed by these methods.

The LTM method firmly demonstrates a generally applicable solution to Research Question #1. The link score is capable of identifying the contribution and polarity of any independent variable to any dependent variable whose relationship is specified as an algebraic or differential expression. With contribution that is multiplicative across causal chains, it is possible to generate explanations for the origins of behavior in any mathematical system representable as a network of equations. It is the link score metric which allows LTM to analyze models containing no feedback which present their structure as directed acyclic graphs. The subtle methods of construction of the link score, namely allows for its calculation directly in time with the model being simulated and for its general applicability to discrete event simulation characterized by discontinuous calculation intervals. Finally, the flexible nature of this construct, namely that the only requirement for calculating a link score is the existence of a dependent and an independent variable linked by an algebraic or differential expression, allow for its application to agent based models and causally structured machine learning techniques such as artificial neural nets. The link score is the basis for the generality of the method.

For systems which include feedback, the loop score and relative loop score allow for a parsimonious representation of the origins of behavior as the trajectories of relative loop score for all loops of interest in the model. Because the dynamics of feedback systems can be described by the combination of effects generated by feedback loops

(Sterman, 2000), the patterns in the trajectories of the relative loop scores for all the feedback loops of interest makes clear the exact and objective origins for the behavior in these systems.

By being able to operate in any mathematical model with a time dimension LTM has demonstrated a complete and robust solution to Research Question #1.

3.2 Refinement: How to Choose Which Loops to Analyze

For LTM to be as broadly applicable as possible, and to make LTM more generally applicable as a solution to Research Question #1, it was critical to find a solution to the problem of reducing feedback loop complexity as demonstrated by Kampmann (2012). Without a solution to the problem of reducing the feedback complexity of a model, the LTM method would only be applicable to models where the full feedback complexity is easily enumerable. As described in Section 2.2.2 the already existing solutions typified by the ILS and SILS were not acceptable when judged against the criteria of being able to find an explanatory loopset for analysis across all models.

Article #4 presents the discovery of the strongest path algorithm which walks the network structure of model equations where the link score is used as the edge weight, identifying the most important feedback loops at each dt in the simulation time period and adding them to the feedback loopset for analysis. Article #4 demonstrates that applying the strongest path algorithm at each dt of the model has the following attributes:

1. It significantly reduces the feedback complexity of feedback dense models.
2. It identifies the feedback loopset for analysis quickly.
3. It includes the most explanatory feedback loops over the course of the simulation time period, and only ignores relatively unimportant feedback processes or feedback processes that it has already captured a significant portion of.

The strongest path algorithm has its roots in the shortest path algorithm of Dijkstra (1959). The steps for the algorithm appear below:

1. Compute the link scores for every connection in the model. Some of these may be 0.
2. For every variable in the model sort the outbound links (places where the variable is used) by the link scores so that the first link has the biggest (absolute value) score.
3. For every stock in the model (all loops involve a stock) start the search:
 - a. Go through each outbound link in order, multiply by the link score of that link then test the variable the link points to using the current path score
 - i. If the variable is the starting variable, record the loop and the associated loop score (which is the path score from above)
 1. Need to check loop for uniqueness. If we already have it, then ignore it.
 - ii. If the variable is being visited already (a loop not involving the starting point) just return as the loop will be found starting from another stock.
 - iii. If the variable has been visited and has a higher path score from that visit just return.
 - iv. If the variable has not been visited, or has a lower path score, record the new path score and execute step a. above.

The strongest path algorithm is not without its shortcomings. It does not guarantee the identification of the singular strongest feedback loop, although it ought to always find a loop very similar to the singular strongest loop. It can fail to identify the single strongest feedback loop in convoluted models where the identification of the strongest feedback loop requires searching causal pathways which are never locally optimal (at any point in time) regardless of the starting location for the search.

The strongest path algorithm for identifying a suitable loopset for analysis improves the LTM method when measured against the requirements of Research Question #1.

3.3 Prototype Implementation: Visualization & Understanding

The first step in the process towards a solution for Research Question #2 was to develop a prototype tool to incubate and exercise the variety of visualization techniques which could be used to make clear the results of an LTM analysis. Article #2 along with the development of the prototype tool LoopX provided this opportunity.

As discussed in Section 1.1.2, Sterman (2000) among others make strong calls for intuitive visualizations and animations of the nature of causality in SD models. Specifically, Sterman (2000) calls for animated stock and flow diagrams where the links and flows between variables in an SFD animate in time with the simulation of the model. LoopX was the first tool of its kind to implement this technology and push the boundaries by also machine generating high quality CLDs which can be algorithmically simplified and animated.

The basis for all animation in LoopX is the link score, specifically the link score relative across all inbound links to a dependent variable. The relative link score allows for a quick and intuitive interpretation of the relative determinants of behavior of any dependent variable by apportioning out the causes of change to each of its inputs. Strong loops then become visible as loops built up of relatively thick links, and weak loops are those which contain thin links. The tradeoffs in animation technology are discussed in depth in Article #2 laying out the full reasoning for choosing the relative link score. A quick summary of those arguments is that the relative loop score (or loop score) is not directly mappable to the network of links because any one link may be resident in a very large number of loops whose score may vary greatly.

As shown in Figure 4 included within all visualizations produced by LoopX is a table which labels each loop showing its instantaneous relative loop score, and its average magnitude of the relative loop score. This allows users to quickly identify which feedback loops are dominant at any particular instant, and which feedback loops are most explanatory over the entire time range of the simulation. All loop labels are

clickable which allows for the feedback loops to be highlighted directly within the structure of the model, allowing users to understand which model processes the metrics refer to.

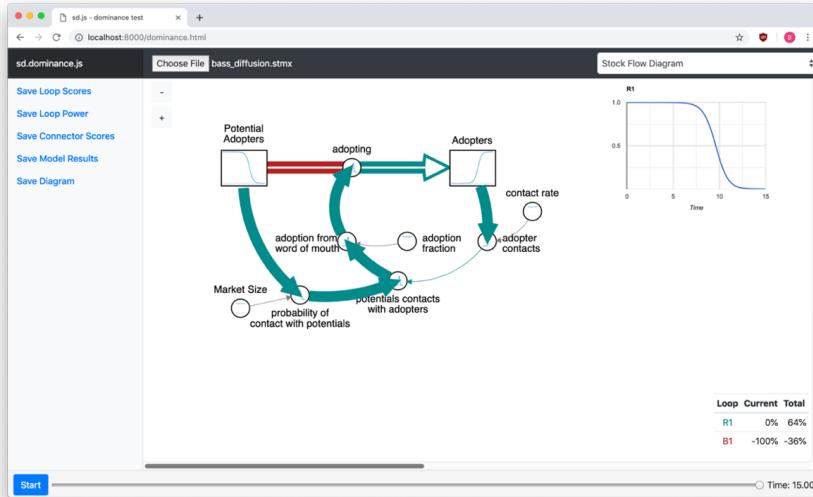


Figure 4: Demonstration of the LoopX tool showing an animated SFD for the Bass diffusion model. Figure reproduced directly from Article #2.

The next major challenge in developing LoopX was discovering how to machine generate high quality CLDs directly from the network of model equations. The field of automated graph drawing provided many useful insights on this task. The most important algorithm family to the automated generation of high quality CLDs is the force directed layout. A force directed layout algorithm solves the problem of the placement of nodes in 2D space, such that symmetry is generated in the resulting diagram, and edge lengths are approximately equal, by simulating a physical system of weights (nodes representing the variables) connected by springs (edges representing the links between the variables) and minimizing the total energy of the system.

The best force directed graph algorithm for autogenerating CLDs discovered was the Kamada Kawai approach (1989) which is based on the concept of an ideal distance between node pairs. The best pre-existing implementation of that algorithm found

was Graphviz's neato (Ellson et. al., 2004). Graphviz is an open source toolkit for solving graph generation problems which has been developed at Bell Labs and neato is their implementation of a Kamada Kawai force directed graph. The problem with force directed graphs as implemented is that, by default, straight edges, which do not emphasize a portrayal of the loops, were generated. Also, any edge curving, previously implemented, had no concept of feedback. Therefore, a key contribution of the research of this thesis is an algorithm to curve the edges (links) in a machine generated simplified CLD in a way that emphasizes the feedback. This algorithm is now the default for curved edges in neato.

The edge curving algorithm follows a simple heuristic derived from the observation of CLD diagram drawing by hand. The heuristic specifies that the center of the circle which forms the arc that the edge will follow, must be the average center of the variables which form the shortest feedback loop that the edge is a member of. This edge curving heuristic relies upon the attributes of force directed graphs which place nodes that are related closest together. This heuristic produces loops that look circular except for in degenerate cases where the force directed layout fails to produce good local clusters and the shortest feedback loops are relatively far flung in the 2D space. A good example of this approach can be seen below in Figure 5.



Figure 5: Autogenerated CLD of the full model structure of Forrester's 1968 Market Growth model showing the ability of the edge curving algorithm to generate CLDs which emphasize the loops. Reproduced directly from Article #2.

The second major visualization challenge solved in Article #2 was to develop an algorithm to automatically simplify the generated CLDs (explanations for the origins of behavior) based upon the user's desired cognitive complexity of the diagram. Two new parameters were introduced facilitate the automated simplification of CLDs. The first parameter introduced is the link inclusion threshold. This parameter is used to select only the dependent variables that have links incoming where the variance in the

magnitude of the relative link score of one of those incoming links is above the threshold. This parameter is used to filter out variables (typically auxiliaries) which aren't key to the explanation for the origins of behavior. The second parameter introduced is the loop inclusion threshold which is used to select the stock and flows of the most important feedback loops to keep in the simplified CLD. The loop inclusion threshold defines the cut-off for automatically keeping the stock and flows of any feedback loop. To be kept, the feedback loop's average magnitude of the relative loop score must be above the threshold. These two parameters produce a list of variables to keep, which are then re-connected by depth first traversal of the full network of model equations. The algorithm walks the network of model equations searching for any causal pathway which directly relates pairs of kept variables without passing through other kept variables.

Combining the model simplification parameters with the automated CLD drawing algorithms allows the end-user to instantaneously reshape the pictured network of causal loops in real time according to their desired level of detail. This functionality in combination with the animation technology discussed in Article #2 (which is directly applicable to both SFDs and simplified CLDs) provides a clear and direct solution to Research Question #2.

3.4 Production Implementation: The Challenges of Building a Production Ready System

The final challenge of bringing the LTM method to the general SD public was to produce a production ready implementation of the LTM method and the LoopX inspired visualizations in Stella Architect, a commercial software product widely used throughout the field of SD. The groundwork for this implementation was put in place first by the production of the LTM analysis program and the prototype visualization tool LoopX (Articles #1, #2 and #4). But there were still challenges to tackle before the implementation could be considered robust relative to the varied kinds of modeling constructs used by practitioners. Article #5 demonstrates the variety of technical challenges that were overcome along the way to making the LTM method

production ready. It also performs an analysis of Mass' (1975) Economic Cycles model and a second discrete model which demonstrates the broad nature of the approach and how it is directly applicable to models with discrete time elements.

3.4.1 A major methodological / implementation challenge

The major methodological/implementation challenge to overcome was how to handle macros such as DELAY or SMOOTH which incorporate a complex hidden internal structure between the source and target of what appears to the end user to be a single link. This problem is especially complex because there may be multiple causal pathways with differing strengths and potentially even polarities, in addition to the possibility of whole sets of feedback loops between the source of the link, and its destination from the perspective of the modeler.

The solution to this challenge was a simple heuristic applied at each calculation interval. The link score of a pathway which passes through a macro is the composite link score of the pathway with the strongest absolute magnitude of its pathway score across all pathways through the macro. The effect of this decision rule is that the specific structure represented by a link through a macro is not fixed throughout the course of the simulation run. The key reasoning behind that decision is that this heuristic most accurately represents the strongest component of the causal pathway which is typically what is desired when analyzing models, to find the strongest, or most explanatory pathway from source to target.

3.4.2 Improvements and advancements in the visualization technology

The first notable change to the model simplification process implemented in LoopX was to stop using a search over the entire network of model equations to reconnect the variables selected by the link and loop filtering processes. The problem with the search over the entire network of model equations was that it ignored why a certain variable was kept and brought forth links (and therefore loops) of demonstrable unimportance in highly simplified CLDs. The solution to the problem required that the link inclusion threshold no longer singularly define what variables to keep. The link inclusion threshold now does the following series of actions. First it marks a

variable as being kept like it always has. Then it also marks the link(s) which made that variable be kept, along with the strongest feedback loop that those links are a member of. Using that information, it becomes possible to tie each full feedback loop to the simplified feedback loop which represents it.

The second major advancement was the introduction of the composite relative loop score which measures the contribution and polarity of multiple loops represented by a single loop in a simplified CLD to model behavior. Because the algorithm knows which full feedback loops are represented by which simplified feedback loops, it becomes possible to calculate a composite relative loop score for each simplified loop. A composite relative loop score is the direct sum of the relative loop scores from all full loops which share their single simplified feedback loop. Since relative loop scores are normalized, and each full loop is only marked as being represented by a single simplified feedback loop, the sum of the composite relative loop scores for all loops in a simplified CLD represents the portion of the full behavior of the model which is explainable by the simplified CLD. The sum of all composite relative loop scores for all simplified loops in a CLD is an attribute which proxies the quality of the simplified CLD. Numbers closer to 1 mean that more of the model's behavior is being explained by that diagram, closer to 0 mean that less behavior is represented in the presented simplified CLD.

The third major improvement was a mathematical basis for representing the concept of the confidence in the polarity of a simplified link. A confidence value is generated using Equation 6, where r is the sum of the single strongest instantaneous reinforcing pathway scores across the entire simulation covered by a single simplified link and b is the sum of the single strongest instantaneous balancing pathway scores across the entire simulation represented by that same single simplified link.

Equation 6: Demonstration of how to calculate the confidence in the polarity of a simplified link

$$\text{confidence} = \frac{|(r - |b|)|}{(r + |b|)}$$

This confidence value makes it very clear when a simplified link is representing two pathways of different polarities. A confidence value of 0.99 (or lower) was chosen as the cutoff point which is used to change the color representing the polarity of the simplified link to gray (to represent mixed polarity) which makes it abundantly clear that the simplified CLD being observed is over-simplified for the model being examined.

The final improvement to the visualization process was targeted at models with higher levels of complexity, specifically models with more stocks. The original definition of the link inclusion and loop inclusion threshold state that anytime a stock is kept, so are its flows, regardless of the relative link scores or relative loop scores of those flows. In large models with large numbers of stocks, especially where the feedback loops tend to be long, this accounts for a large majority of the complexity retained, while not necessarily adding anything to understanding. Therefore, a third simplification parameter, a boolean, was introduced allowing the user to control if flows are automatically kept if a stock is kept.

3.4.3 Conclusions on the production implementation

Putting the full set of LTM technologies into production demonstrates significant results on the solutions to both research questions. The design of the composite relative loop score and the simplified link polarity confidence significantly improves the evaluation of the quality of the autogenerated simplified CLDs. The implementation of a solution for link which span across macros demonstrably improves the general applicability of the algorithm of models.

Figure 6 shows an example of a simplified CLD created using Stella Architect based on a version of Forrester's 1968 Market Growth model which contains macros. Figure 6 demonstrates the power of the model simplification approach which has stripped away the majority of the feedback complexity of the model as well as cut in half the number of variables while maintaining the ability to explain 96.3% of the model's behavior. The ability to perform analyses like this is proof of the answers to both research questions.

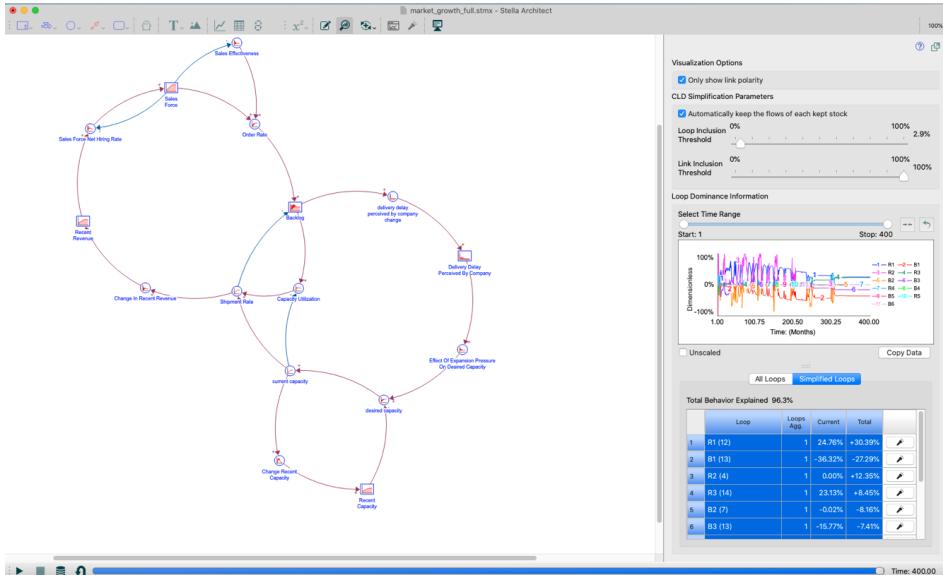


Figure 6: Screenshot of Stella Architect demonstrating the complete implementation of the LTM method and associated visualization technologies in Forrester's 1968 market growth model. Notice the indicator of simplified CLD quality represented by the total behavior represented.

3.5 A Cross-Disciplinary Application: Integrating the Link Score into New Machine Learning Method for Automated Causal Inference

The power of the LTM method for discovering the origins of behavior in any time dependent mathematical system has direct utility outside of the field of SD. Article #3 uses the link score as an integral component of a new method called Feedback System Neural Networks (FSNNs) for doing automated causal inference using a machine learning approach. The FSNN method developed and described in Article #3 exists at the intersection of the fields of SD and Machine Learning (ML). This work is a demonstration of the success of the LTM method on Research Question #1.

The FSNN method, which was derived and analyzed in Article #3 constructs an n-dimensional system of ordinary differential equations (ODEs) that is parameterized to fit empirical data and then analyzed using the link score to discover the origins of

behavior. The n-dimensional system of ordinary differential equations (ODEs) is the same thing as a system dynamics model.

As discussed in the problem definition in Section 1.1.3 current approaches for automated causal inference are not designed to work in feedback rich environments. Instead, the methods which are classified in the literature by Runge et. al, (2019), are only applicable to structures represented as directed acyclic graphs. The technique presented here represents a major advancement in the state-of-the-art.

3.5.1 An introduction to neural networks

To properly understand the implementation of the technique it is necessary to understand the context and the history behind the construction of artificial neural nets (ANNs). ANNs are a series of mathematical expressions, referred to as artificial neurons, where the organization of the artificial neurons is inspired by the structure of the brains of animals. ANNs ‘learn’, during a process that is called training, by being exposed to data, which is called training data. The training process parameterizes the directed acyclic graph of artificial neurons using a gradient descent optimization process. This is more commonly referred to as back-propagation. The parameters are chosen to maximize the identification of a given set of preordained proper outputs for the given set of inputs. The effectiveness of the ANN is measured on testing data (specifically data which was not trained on) to discover the rate of failure of the ANN. Training and testing data are organized as a series of known inputs, which each have a series of known outputs.

The standard artificial neuron is called a perceptron and was defined by Rosenblatt (1957). An example is shown in Figure 7. The mathematical equation of an artificial neuron is shown in Equation 7. An artificial neuron takes a series of inputs, weights them individually, sums the now weighted inputs, and adds a bias. It takes that value and passes it through a non-linear activation function, f , which is the output of the neuron. Activation functions can take a wide range of non-linear shapes, anything from STEP functions, via SIGN functions, to Sigmoid functions are usable as

activation function depending upon the application domain of the ANN. For all examples in this thesis the *tanh* function is used as the activation function.

Equation 7: Mathematical expression for an artificial neuron. Rosenblatt 1957.

$$Y = f \left(\sum (\text{input} * \text{weight}) + \text{bias} \right)$$

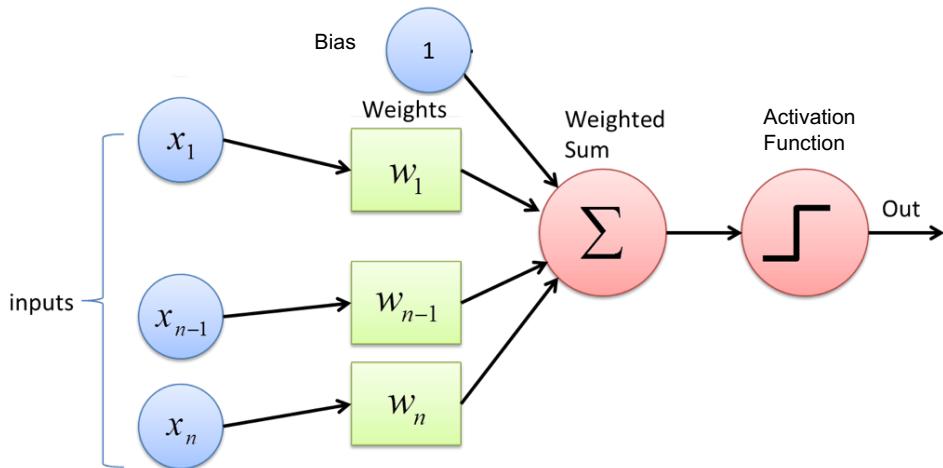


Figure 7: Structure of an artificial neuron as constructed by Rosenblatt (1957).

The perceptron is typically used as a node in a multilayer perceptron network (MLP) where artificial neurons are arranged as a series of layers between the raw input data and the generated output data as shown in Figure 8. These networks are sometimes referred to as feed-forward neural networks because data is passed along chains from the inputs to the outputs. MLPs can have any number of layers between the input and output data, and those layers are referred to as hidden layers. The standard usage of neural networks is in supervised learning, which refers to the concept that the output(s) are known for each set of input(s), and training uses back-propagation to modify the weights and bias of each artificial neuron of the MLP to minimize the prediction error in the training data.

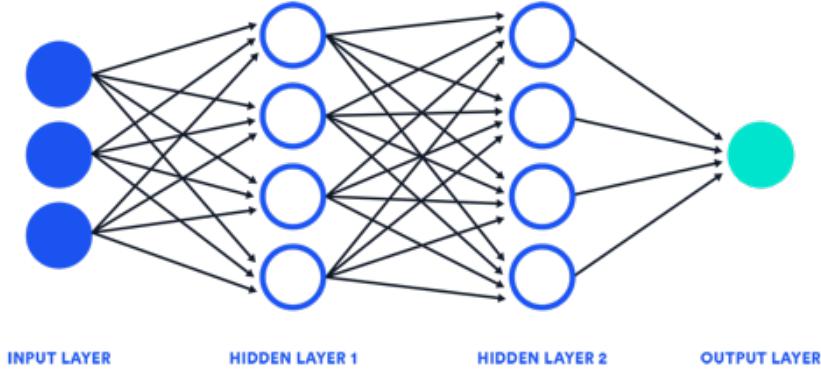


Figure 8: Structure of a simple MLP ANN without any recursion. There are 3 inputs and a single output. This 2 hidden layer MLP ANN learns the relationships between the 3 input series and the single output series.

MLP ANNs like the one in Figure 8 can be used as a stand-in for any continuous function which relates the inputs and the outputs. The universal approximation theorem (Cybenko, 1989) states that a standard feed-forward MLP ANN with a single hidden layer containing a finite number of artificial neurons can approximate continuous functions in Euclidian space under mild assumption on the activation function. The theorem states that simple neural networks may represent a wide variety of functions under specific parameterizations, but it doesn't say anything about the gradients which may or may not lead to those parameters. Because of the conclusions which can be drawn from the universal approximation theorem, neural networks become powerful tools for building mathematical systems which relate inputs to outputs.

3.5.2 The new machine learning method for automated causal inference

A high-level overview of the FSNN technique is presented below in Figure 9. The process starts with a n-dimensional set of time series data which is used to construct a model. Each dimension in the original dataset is modeled as a stock. The net flow for each stock is an MLP ANN which takes as its inputs the current value for each and every stock in the model and produces as an output the net change in the stock it's pointing to. The model is trained via an optimization process which calibrates the

model to reproduce the original dataset. The model, using the set of parameters which produces behavior in the stocks which matches most closely with the original dataset is analyzed using LTM, specifically the composite relative link score to discover the strength of the causal pathways to and from each directed pair of stocks in the system. The composite relative link scores for those pathways are then visualized so that results of the LTM analysis can be interpreted and easily understood showing the end user one possible causal structure which would explain the origins of the given data.

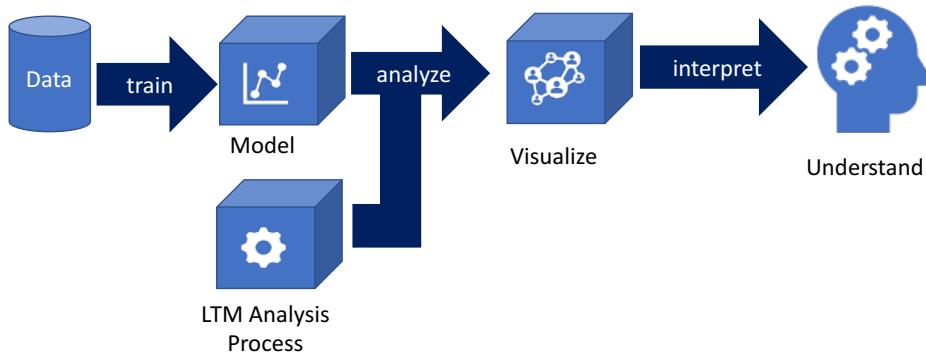


Figure 9: High level overview the machine learning approach to causal inference which includes LTM.

The FSNN method acts as a machine that produces behaviorally accurate, feedback rich structural hypotheses directly from data where the polarity and contribution of each link between all stocks in the system is known. The machine takes as its input time series data measured from the real world, and, with no additional input, produces a behaviorally relevant causal dynamic hypothesis. Such causal dynamic hypotheses are much more easily validated by practitioners of either SD or ML than standard neural nets and other ML technologies. Plus, the validation performed is of a much higher quality because it is a structural validation of the causal structure of the system rather than a behavioral validation of the outputs of the model.

Article #3 goes into depth demonstrating how behavioral validation of machine generated models can lead to completely erroneous conclusions about the causal structure of a system. The generated model produced in Article #3 was able to properly identify the causal structure of the ground truth system, correctly identifying

which states were directly related, and what the polarity of those relationships was. The generated model though was not the singular well defined model of the ground truth system because the nature the those relationships in the ground truth system and the generated model did produce exactly the same set of composite relative link scores. This lack of overlap of the composite relative link scores in the ground truth system and the generated model was exploited when running the generated model on data far outside of the range it was trained on, showing how generated model performs poorly in those cases. This clearly proved that the generated model was not properly parameterized and could not be considered directly equivalent to the ground truth system.

By combining SD and ML Article #3 demonstrates a powerful new machine learning technique for finding the causality in arbitrary data sets. By using MLP ANNs to represent the net flows (derivatives for the states) the universal approximation theorem (Cybenko, 1989) ensures that each net flow is capable of reproducing the real-world function which links the stocks together. By linking all stocks together in a directed cyclic graph this new machine learning method becomes much more likely to find accurate causal models for the data. This is because as J. Forrester might say: non-linear feedback systems are at the heart of complex dynamic real world problems and those systems are best represented in mathematics as systems of ODEs containing feedback and time delays which this new method does. Article #3 represents the potential power of LTM and its visualization, the solution to Research Questions #1 and, #2 to change the nature of how causally accurate science is performed moving into the future.

4. Conclusions

The LTM method and its associated visualization technologies, presented in this thesis, provide clear solutions to the problems of discovering and understanding the structural origins of behavior in time dependent mathematical systems. These problems are important because without modelers (both SD and ML) having a deep understanding of the causal structures which drive the behavior of their models, the risk exists that they may develop incorrect understanding leading to flawed policy recommendations and bad outcomes in the real world.

Article #1 covers in depth the necessary mathematical concepts which are at the heart of the LTM method. Article #1 demonstrates and explains how LTM functions and provides a clear and direct interpretation for the three key metrics (link score, loop score, relative loop score) it introduces. Article #1 provides a direct answer to Research Question #1 which is concerned with developing a general solution for identifying the origins of model behavior in any time dependent mathematical system.

Article #2 presents the required visualization, animation, and simplification technologies which allow for the broad uptake of the LTM method among modelers. Article #2 demonstrates how to machine-generate simplified CLDs directly from the network of model equations using the loop inclusion and link inclusion thresholds based on the LTM metrics presented in Article #1. Article #2 also achieves the long-standing goals as written by Sterman (2000) in the final chapter of Business Dynamics, for software as it relates to the visualization and communication of loop dominance information. Article #2 provides a directly applicable, highly intuitive and usable solution for visualizing the origins of model behavior which is asked for in Research Question #2.

Article #3 is a cross disciplinary approach to automating the process of causal discovery in highly non-linear feedback rich systems. It presents a new approach to automated causal inference which has attributes of both SD and ML embedded within. The method discussed in Article #3 is a very clear demonstration of the potential unlocked by the discovery of the LTM method supporting the broader usage

of the concepts of systems dynamics, and structural dominance analysis by a much larger slice of the scientific community which ought to lead to better policy outcomes. By combining ANNs, non-linear feedback systems, and structural dominance analysis Article #3 demonstrates that the LTM method as presented in Article #1 provides a solution to the problem of identifying the origins of model behavior in any time dependent mathematical system (Research Question #1).

Article #4 introduces the strongest path algorithm and presents a solution to the challenge raised by Kampmann (2012) to identify the more important set of feedback loops for analysis. This article allows LTM to be directly applicable to giant model which exhibit extreme levels of feedback complexity. The strongest path algorithm really highlights and underscores the utility of the link score measure while reinforcing the usefulness of the LTM method as whole. This refinement to the LTM method ensures that LTM provides a complete solution to Research Question #1.

Article #5 walks the reader through a series of challenges related to making the implementation of the LTM method and its visualization robust enough to be used in a production environment within a generally available commercial software package (Stella Architect). This article refines and develops the technologies presented in both Article #1 and Article #2, demonstrating their use within practitioner developed models containing macros and discrete time elements. Article #5, and specifically the production implementation of the LTM method and its associated visualization technologies in Stella Architect demonstrates how this thesis, and the research contained within it have successfully answered both research questions.

5. References

- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4), 303-314.
- Davidson P. 1991. The Structure-Behavior Graph. The System Dynamics Group, MIT, Cambridge, MA.
- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269–271.
- Duggan J. and Oliva R. 2013. Methods for identifying structural dominance. *System Dynamics Review* 29(Virtual Special Issue). Available at [http://onlinelibrary.wiley.com/journal/10.1002/\(ISSN\)1099-1727/homepage/VirtualIssuesPage.html](http://onlinelibrary.wiley.com/journal/10.1002/(ISSN)1099-1727/homepage/VirtualIssuesPage.html).
- Eades, P. 1984. A heuristic for graph drawing. *Congressus numerantium*, 42, 149-160.
- Eberlein, R. L. 1984. *Simplifying dynamic models by retaining selected behavior modes* (Doctoral dissertation, Massachusetts Institute of Technology).
- Ellson, J., Gansner, E. R., Koutsofios, E., North, SC, & Woodhull, G. (2004). Graphviz and dynagraph—static and dynamic graph drawing tools. In *Graph drawing software* (pp. 127-148). Springer, Berlin, Heidelberg.
- Ford D. 1999. A behavioral approach to feedback loop dominance analysis. *System Dynamics Review* 15(1): 3–36.
- Forrester, JW. (1961). Industrial dynamics. 1961. *Pegasus Communications*, Waltham, MA.
- Forrester, JW. 1968. *Market growth as influenced by capital investment*. Industrial Management Review.
- Forrester, JW. 1994. System dynamics, systems thinking, and soft OR. *System dynamics review*, 10(2-3), 245-256.
- Forrester N. 1982. A Dynamic Synthesis of Basic Macroeconomic Theory: Implications for Stabilization Policy Analysis. PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Gansner, ER. 2014. Using Graphviz as a Library (cgraph version). *published online August 21*.
- Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553), 452.

- Gonçalves P. 2009. Behavior modes, pathways and overall trajectories: eigenvector and eigenvalue analysis of dynamic systems. *System Dynamics Review* 25(1): 35–62.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
- Graham AK. 1977. Principles of the Relationship Between Structure and Behavior of Dynamic Systems. PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, 424-438.
- Güneralp, B. 2006. Towards coherent loop dominance analysis: progress in eigenvalue elasticity analysis. *System Dynamics Review*, 22(3), 263-289.
- Hayward J, and Boswell GP. 2014. Model behaviour and the concept of loop impact: a practical method. *System Dynamics Review* 30(1): 29–57.SEP
- Hayward, J, and Roach PA. 2017. Newton's laws as an interpretive framework in system dynamics. *System Dynamics Review* 33(3-4): 183-218.
- Huang, J., Howley, E., and Duggan, J. (2012). Observations on the shortest independent loop set algorithm. *System Dynamics Review*, 28(3), 276-280.
- Kamada, T., & Kawai, S. 1989. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1), 7-15.
- Kampmann CE. 2012. Feedback loop gains and system behaviour (1996). *System Dynamics Review* 28(4): 370–395.
- Kampmann CE, and Oliva R. 2006: Loop eigenvalue elasticity analysis: Three case studies. *System Dynamics Review*. 22(2): 141-162.
- Kampmann, CE., and Oliva, R. 2008. Structural dominance analysis and theory building in system dynamics. *Systems Research and Behavioral Science: The Official Journal of the International Federation for Systems Research*, 25(4), 505-519.
- Kampmann CE., and Oliva R. 2009. Analytical methods for structural dominance analysis in system dynamics. In Meyers RA (ed.) *Encyclopedia of Complexity and Systems Science* Springer, New York NY, 8948–8967.
- Mass, NJ. 1975. Economic Cycles: An Analysis of Underlying Causes, Wright. Cambridge Massachusetts.
- Mojtahedzadeh MT. 1996. Structural Analysis of the URBAN1 model. University at Albany, SUNY, NY 12222, USA.

- Mojtahedzadeh MT, Andersen D, and Richardson GP. 2004. Using Digest to implement the pathway participation method for detecting influential system structure. *System Dynamics Review* **20**(1): 1-20.
- Montavon, G., Samek, W., & Müller, K. R. 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, **73**, 1-15.
- Moxnes, E, and Davidsen PI. 2016. Intuitive understanding of steady-state and transient behaviors. *System Dynamics Review* **32**(2): 128-153.
- Pearl, J. 2009. Causal inference in statistics: An overview. *Statistics surveys*, **3**, 96-146.
- Naumov, S, & Oliva, R. 2018. Refinements on eigenvalue elasticity analysis: interpretation of parameter elasticities. *System Dynamics Review*, **34**(3), 426-437.
- Naumov S, Oliva R. 2018. Structural dominance analysis toolset. Available: <http://iops.tamu.edu/faculty/roliva/research/sd/sda/> [05 Feb 2020].
- Oliva, R. 2004. Model structure analysis through graph theory: partition heuristics and feedback structure decomposition. *System Dynamics Review*, **20**(4): 313-336.
- Oliva R. 2016. Structural dominance analysis of large and stochastic models. *System Dynamics Review* **32**(1): 26-51.
- Richardson, GP. 1986. Dominant structure. *System Dynamics Review*, **2**(1), 68-75.
- Richardson, GP. 1991. *Feedback thought in social science and systems theory*. University of Pennsylvania.
- Richardson GP. 1995. Loop polarity, loop dominance, and the concept of dominant polarity. *System Dynamics Review* **11**(1): 67–88.
- Rosenblatt, F. 1957. *The perceptron: A probabilistic model for Visual Perception*. In Proceedings of the 15th International Congress of Psychology, North Holland (pp. 290-297).
- Runge, J., Bathiany, S., Boltt, E., Camps-Valls, G., Coumou, D., Deyle, E., ... & van Nes, E. H. 2019. Inferring causation from time series in Earth system sciences. *Nature communications*, **10**(1), 2553.
- Saleh MM. 2002. The Characterization of Model Behavior and its Causal Foundation. PhD Thesis, University of Bergen, Bergen, Norway.
- Saleh MM, Oliva R., Kampmann CE, Davidsen PI. 2010. A comprehensive analytical approach for policy analysis of system dynamics models. *European Journal of Operational Research* **203**(3): 673-683.

- Sato, JB. 2016. State space analysis of dominant structures in dynamic social systems. PhD Thesis, University of Washington, St. Louis, MO.
- Schölkopf, B., & Smola, AJ. 2008. Learning with kernels. MIT Press, Cambridge, MA. *Learning with kernels. MIT Press, Cambridge, MA.*
- Spirites, P., & Zhang, K. 2016. Causal discovery and inference: concepts and recent methodological advances. *In Applied informatics* (Vol. 3, No. 1, p. 3).
- Sterman JD. 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Irwin/McGraw-Hill, Boston.
- Verma, T, & Pearl, J. 1990. Causal networks: Semantics and expressiveness. In *Machine intelligence and pattern recognition* (Vol. 9, pp. 69-76). North-Holland.
- Wiener, N. 1956. The theory of prediction. Modern mathematics for engineers. *New York*, 165-190.

6. Article #1: Understanding model behavior using loops that matter

By: William Schoenberg, Pål Davidsen, and Robert Eberlein.

Abstract

The relationship between structure and behavior is central to System Dynamics, but effective tools required to understand that relationship still elude us. The current state of the art in the field of loop dominance analysis relies on either practitioner intuition and experience or complex algorithmic manipulation in the form of eigenvalue analysis or pathway participation metrics. This paper presents a new and distinct numeric method based on a different measure, the loop score, to determine the contribution of a loop to a model's behavior at each instant in time. This allows us to discover the origin of model behavior. The method was inspired by observations of the patterns in the changes of the values of variables during simulations and has been tested and refined using empirical evaluation on a variety of models. The method also offers a promising approach to the visualization and aggregation of simulation results.

The problem

The strong relationship between structure and behavior is fundamental to system dynamics (Sterman, 2000). A model uses parametric and structural assumptions to produce behavior. A system dynamics practitioner must generally engage in the following process: create a structure underlying the problem under consideration, understand how that structure works to produce that problem, and figure out how to improve that structure so as to address the problem. The second step in this process is the focus of this paper and is the key to successfully performing the third step. By referring back to the model structure, the practitioner can explain the reasons why the observed behavior has been generated (Richardson, 1996). Based on that understanding, the practitioner may propose changes in input values or model structure (system configuration) that will cause more favorable behavior to be produced.

The current state of the art in the field relies on either practitioner intuition and experience (the craft of modeling and model analysis) or complex algorithmic analysis. The former is taught as part of the methodology of model building, while the latter comes from 40 years of work on techniques to derive and explain model behavior based on the analysis of structure (see for example: Graham, 1977; Forrester 1982; Eberlein, 1984; Davidsen, 1991; Mojtahedzadeh, 1996; Ford, 1999; Saleh, 2002; Mojtahedzadeh et al., 2004; Güneralp, 2006; Gonçalves, 2009; Saleh et al., 2010, Kampmann, 2012; Hayward and Boswell, 2014; Moxnes and Davidsen, 2016; Oliva, 2016; Sato, 2016; Hayward and Roach, 2017; Naumov and Oliva, 2018; and Oliva, 2020).

Ford (1999, p.4-5) clearly stated the needs of the system dynamics field as they apply to loop dominance analysis:

“To rigorously analyze loop dominance in all but small and simple models and effectively apply analysis results, system dynamicists need at least two things: (1) automated analysis tools applicable to models with many loops and (2) a clear and unambiguous understanding of loop dominance and how it impacts system behavior.”

In this paper, we discuss a new loop dominance analysis method and demonstrate its performance on the two tests set out by Ford.

We define loop dominance as a concept which relates to the entirety of a model, as opposed to loop dominance being something that affects a single stock. For loop dominance to apply to the entire model, we require that all stocks are connected to each other by the network of feedback loops in the model. For models where there are stocks that do not share feedback loops, we consider each subcomponent of inter-related feedback loops individually, and we refer to each model substructure as having a separate loop dominance profile. Our measurement of loop dominance is specific to the particular time period selected for analysis. We say that a loop (or set of loops) is

dominant if the loop(s) describe at least 50% of the observed change in behavior across all stocks in the model over the selected time period.

The purpose of this research is to create a loop dominance analysis method that is accurate and practical. While this new method may not generate new insights relative to existing analysis methods, it has two characteristics that make it more practical. First, it is relatively easy to understand for people who have built and analyzed system dynamics models, even those with a limited mathematical background. Second, it lends itself completely to presenting behavior over time graphs that mix loop dominance measures and model behavior, making it easy to see the evolution of loop dominance. These characteristics, combined with the very straightforward computational techniques we employ, mean that such an analysis can easily be built into existing system dynamics modeling environments. This will make finding the loops that matter over time as easy as it is to watch a graph of behavior unfold over time. Such ease of use means the methods will routinely remove one more obstacle in our ability as a field to do good work.

First, we discuss the existing approaches used to automate the process of discovering loop dominance. Thereafter, we introduce our new method, starting with links and building up to loops. Then we demonstrate the new method applied to three models that have already been analyzed using existing automated loop dominance analysis techniques, so as to demonstrate how well the new approach works. Finally, we conclude with a discussion of the benefits and weaknesses of the new method.

Literature review

The current state of the art in the use of mathematical methods for determining loop dominance revolves around two methods. The first one is based on eigenvalue elasticity analysis, the second one, uses the pathway participation metric and causal pathways.

Eigenvalues and eigenvectors, specifically eigenvalue elasticity analysis (EEA)

Forrester (1982) was the first to document that eigenvalue elasticities could be used to explain the relative contributions of different loops in models of linear systems. Since then, the formal method of eigenvalue elasticity analysis (EEA) has been further developed and is now used to determine how model structure produces the dynamic modes of behavior in a model, specifically those modes of behavior characterizing the state variables or stocks, (Saleh, 2002; Kampmann et al., 2006; Saleh et al., 2010; Oliva, 2016). Using EEA, the combination of behavior modes expressed by a model is characterized by the eigenvalues and eigenvectors of that model. This analysis is built on the observation that the behavior of a linear system can be expressed as a weighted combination of behavior modes, each characterized by a decoupled (or pairwise coupled) set of eigenvalues (Saleh et. al., 2010). EEA is applied to examine both link and loop significance with regard to the dynamic behavior of the model. It does so by identifying the relationship, expressed in the form of the elasticity, between the parameters that make up the gains of individual feedback loops in the model and the eigenvalues (and sometimes eigenvectors) that characterizes the dynamic behavior of the model. The significance of a loop to generating behavior is expressed by the eigenvalue elasticity of its gain (how strongly a change in the loop gain impacts the eigenvalue associated with the behavior of interest). Note that this may not only be used to identify the root cause of a model's behavior, but also the leverage points for controlling a system (policy entry points) provided the model is an accurate representation of that system.

Kampmann (2012) developed the concept of the independent loop set (ILS) which maps all of the loops in the model onto a singular set of independent loops that produces the full behavior of the model so that the analysis can be effectively completed and interpreted. Oliva (2004) extended Kampmann's work on the ILS by developing an ILS composed only of geodetic loops which he termed the shortest independent loop set (SILS). That is the de-facto standard for determining which loops to analyze in EEA.

Pathway participation metric (PPM) and other causal pathway techniques

The pathway participation metric (PPM) approach does not use eigenvalues to describe model structure. Rather, it focuses on the links between variables (Mojtahedzadeh et al, 2004). The starting point of the PPM approach is the behavior of a single variable, typically a stock. The behavior of that single variable is partitioned in time, based on periods in which the variable maintains its slope and convexity (i.e. the first and second time derivatives do not change sign) (Mojtahedzadeh et al, 2004). The behavior of the variable in each of these phases is characterized by 7 patterns enumerated by Mojtahedzadeh et al, (2004). The PPM approach then determines dominance by tracing along the causal pathways between the stock under study and its ancestor stocks so as to determine which structure is the most influential one in terms of explaining the pattern of behavior exhibited by that stock during the selected phase. Mojtahedzadeh et al., (2004) explains that it does so by determining the magnitude of the change in the net flow of the stock under study by making minute changes to that stock. The method then compares these changes in the net flow in order to determine the change of the largest magnitude in the same direction as the stock under study, - thereby identifying the most important (dominant) pathway governing the behavior of that stock during that phase.

Relative to the very general EEA approach that yields results that are interpretable across the entire model, PPM is considerably more specific, only explaining the impact of feedback loops on specific stocks of interest. In addition, PPM does not identify the dominance of behavior modes and, consequently, only provides information about behavior that can be readily observed in the trajectory of stock behavior.

The advantage of the PPM approach is that it does not require any manipulation of the model structure and that it's implementation does not require model structure to be continuous. Moreover, according to the research by Mojtahedzadeh (1996), the application of the PPM method will cause a convergence on a unique piece of structure as the one most influential with regard to the phase of its behavior under

study. Kampmann and Oliva (2009) state that one of the key benefits to the PPM method is its ability to associate directly model behavior to structure.

Kampmann and Oliva (2009) have criticized PPM for its inability to clearly explain oscillatory behavior and also because PPM may fail to identify structure when there are two pathways of similar importance (Kampmann and Oliva, 2009). Hayward and Boswell (2014) have responded to those criticisms by simplifying PPM into the Loop Impact method. The Loop Impact method can be implemented in a standard system dynamics models (and software) by adding equations to the model. No change in the underlying software is required. The key difference of the Loop Impact method as compared to PPM is that it does not identify dominant pathways (impacts from one stock to another), but instead focuses on the direct impact that one stock has on another to identify loops which dominate the behavior of the selected stock. Pathways are chained together according to the structure of the model and these chains of pathways are used to measure the Loop Impact metric which yields insight into which loop dominate the behavior of the selected stock (Hayward and Boswell, 2014). In addition, the Loop Impact method identifies instances where multiple loops are required to explain the behavior of a stock.

Expanding on the work done by Hayward and Boswell (2014) has modified the Loop Impact method specifically codifying the impacts of force in an engineering sense. Hayward and Roach (2017) have also developed a framework around the Loop Impact method couched in the mathematics of Newtonian physics, to explain the model as a series of interacting forces. The stated purpose of the underlying common research thread between these authors of the Loop Impact method and its derivatives, is to provide a more intuitive and complete understanding of loop dominance in system dynamics models.

The loops that matter method

Introduction to the loops that matter method

In this paper, we present the LTM (Loops That Matter) method which, under the categorization scheme of Duggan and Oliva (2013), algorithmically performs a “formal assessment of dominant structure and behavior” for models of any size, complexity, or dimensionality. Like PPM, this method is derived, in part, from observations of the way that experienced modelers perform analysis to determine the sources of observed behavior. Unlike EEA, and similar to PPM, this method does all of its calculations directly on the original model equations, walking the causal pathways between stocks through all intermediate variables making it easier to understand the measurements of loop contribution to model behavior and link contribution to feedback loop dominance. The method uses only values computed during a regular simulation of any model, - including models with discrete characteristics. Because there is no model transformation taking place, there is also no canonical form that definitively and uniquely defines the characteristics of model behavior the way that the eigenvalues and eigenvectors do in the case of linear models. Instead, we have developed a metric that tracks the concept of link gain such that the chain rule of differentiation can be used in defining a second metric we have developed to track the concept of loop gain. This guarantees that the description of dynamic behavior will be the same, independent of the exact form of structurally equivalent models (many variables with simple equations or few variable with complex equations). This property of the LTM method is proven in Appendix II. While our definition of loop dominance is not demonstrably unique among structurally equivalent models, it seems unlikely that alternative sets of equations would change the ranking of loop scores so as to give substantively different results.

LTM is applicable to more than regular continuous SD models. LTM is applicable to agent based as well as discrete time models as long as the structure of these model is represented as a network of equations to be evaluated at known time points. The reason that LTM is so broadly applicable is because LTM is only computed at each

time point that the model is calculated, using only the existing causal relationships and variable value computations as inputs. LTM ensures that it does not affect the validity of the analyzed model by not manipulating equations or using variable-values not computed as part of the regular simulation. An example of a manipulation, required by a method, that affects the validity of a model under analysis is a change in a discrete, integer only variable, say something like product color from 2 to 2.1. This change would result in a logical error in the model (even though the model may be calculated without error) because the modeler did not anticipate a non-integer value for that variable and, therefore, the model responds in an unanticipated way.

LTM introduces two metrics. The first one, the link score, is used to measure the contribution and polarity of a link between an independent variable and a dependent variable. The second one is the loop score that measures the contribution of a feedback loop to the behavior of the model and is indicative of the feedback polarity. Loop scores and link scores are calculated at each time interval during the simulation run. The analysis of the relative loop scores at a particular point in time identifies the loops that dominate that model behavior. The display of the relative loop scores over time contributes to our understanding of why the model behaves the way it does, i.e. points to the feedback loops that govern current behavior. By definition, loop scores are completely insensitive to the number of variables and links in a loop.

We use the standard definition of a loop: It is a set of interconnections between variables in a model that form a closed path from a variable back to itself. By definition any such loop must include at least one state variable (so as to avoid a simultaneity). We refer to the interconnections as links. Loop scores are computed as products of link scores, and the definition of a link score is tailored to this specific use. The link score computation has been defined for the sole purpose of determining loop dominance, specifically to be used in the loop score calculations.

Introduction to the link score measure

The link score measures the contribution of a value change in an independent variable to a value change in a dependent variable and also the associated polarity. A link score

is calculated for each link in the network of model equations, - including the links from flows to stocks. Because links from flows to stocks constitute an integration process, whereas links into other variables do not, we have devised two methods for calculating the link score, where both have the same conceptual interpretation. The first method we discuss below is for links which do not constitute an integration process, for instance a link from one auxiliary to another. The second method we discuss handles the links from flows to stocks that are integrations. Both methods produce a link score, and these link scores are directly comparable and are multiplied together to produce loop scores.

The link score is not a general metric that can be used to describe the contribution or importance of any specific link in isolation. It must be considered in the context of the loops it is contained in. The most obvious manifestation of the link score's lack of generality is that a link originating from a variable that is not changing its value (including a parameter), have a score that, by definition, is 0. This is so, because when a variable in a loop does not change value, the loop is inactive and, therefore, currently of no consequence. Even though the link score for all links originating from variables whose value do not change, are 0, such variables are still significant to the loop dominance analysis because they may, by serving as parameters in other equations, influence the way that other variables change. For example, a time constant might condition some other link score, which consequently changes the score of a loop containing that link.

Defining link scores for links without integration

To simplify the presentation, we will define the link score assuming there are two inputs (x and y) to the dependent variable z characterized by the equation $z = f(x, y)$. This easily generalizes to the case where there are more (or fewer) inputs into the equation defining z .

The link score for the link $x \rightarrow z$ is computed using the formula in Equation 1. Equation 1 shows the discrete form for the link score equation which matches the implementation of the calculation and is computed each dt .

$$LS(x \rightarrow z) = \begin{cases} \left(\left| \frac{\Delta_x z}{\Delta z} \right| \cdot sign \left(\frac{\Delta_x z}{\Delta z} \right) \right), \\ 0, \quad \Delta z = 0 \text{ or } \Delta x = 0 \end{cases} \quad (1)$$

In Equation 1, Δz is the change in (the value of) z from the previous time to the current time. Δx is the change in x over that interval. $\Delta_x z$ is the change in z with respect to x over that interval. From a computational perspective $\Delta_x z$ which we call the partial change in z with respect to x , is the amount z would have changed, conditionally, if x had changed the amount it did, but y had not changed (i.e. *ceterus paribus*). The first major term in Equation 2 represents the magnitude of the link score, the second one is the link score polarity.

The exceptions for no change in x or z are included for completeness. If x does not change then the links from x , and any loops involving x , will have a score of 0. (Similarly, if z does not change then any links into or out from z will have a score of 0 and any loops involving those links are inactive). In such cases, we simply define the value of the link score from x to z to be 0.

The first major term in Equation 1 is the link score magnitude $\left| \frac{\Delta_x z}{\Delta z} \right|$ which describes the effect (force is a good analogy) that an input (i.e. independent variable) x has on an output (i.e. dependent variable) z , relative to the total effect exerted on z . Unlike a partial derivative, which describes how sensitive z is to changes in x , this magnitude describes how much the change in x has contributed to the total change in z .

The first term is dimensionless and represents the contribution of x to the change in z . If all of the link scores have the same sign, it expresses the fraction of the change in z that results from the change in x . If the equation for z is linear (uses only addition and subtraction), then the values will always be in the range between 0 and 1. When there are both positive and negative link polarities in a nonlinear equation, the link score magnitude for such a nonlinear equation may take on a very large value. But this does not jeopardize the overall analysis of loop dominance because it is the relative

values of loop contribution that are analyzed at each point in time and large magnitudes counteract each other in such a comparison.

The second major term in Equation 1 is the polarity of the link $\text{sign}\left(\frac{\Delta_x z}{\Delta x}\right)$ which is defined as the sign of the partial difference at time t . This formulation is functionally the same as the one used in Richardson 1995. We use the partial difference notation in order to maintain consistency with the link score magnitude whereas Richardson uses the partial derivative notation. Our reformulation of Richardson's polarity makes it easier to calculate the link score because the $\Delta_x z$ value can be used in both the magnitude and polarity computation.

In Appendix I we show how Equation 1 can be recast using partial differences, and, in the limit, partial derivatives. This is helpful when comparing our approach with other metrics such as PPM (Mojtahedzadeh, 1996) and Loop Impact (Hayward and Boswell, 2014).

Defining link scores for links which contain integration processes

The link from flows to stocks represent an integration process, and stocks can only change over time as a result of the specific values of the associated flows. So time must pass in order for a flow rate value to materialize in a change in associated the stock value. This is very different from above where a dependent auxiliary changes *immediately* in response to a *change* in one of its independent variables. As an example, if a stock only has a single flow, and that flow value is a constant, that stock will change over the course of the simulation. Consequently, the link between the flow and the stock is active and the link score is non-zero. Since links from flows to stocks are the only links with this unique property, the computation is different in form, but constitutes the same concept as the one used in Equation 1.

Assume the stock equation $s = \int(i - o)$ where s is the stock, i is the inflow, and o is the outflow. So we assume a single inflow and outflow for simplicity of presentation, - the generalization to multiple inflows and outflows is straightforward. Equation 2 demonstrates the link score for all links from flows to stocks.

$$\text{Inflow: } LS(i \rightarrow s) = \left(\left| \frac{i}{i - o} \right| * 1 \right) \quad \text{Outflow: } LS(o \rightarrow s) = \left(\left| \frac{o}{i - o} \right| * -1 \right)$$

(2)

If you compare Equation 2 to Equation 1 you will notice that they represent the same concept. The value of the flow (i or o) is exactly the same concept as the partial change in z with respect to x ($\Delta_x z$) from Equation 1, where the flow is x and the stock is z . From the perspective of the stock, - the value of the flow (when multiplied by dt) is the amount the stock will change if no other flows were active. The denominator ($i - o$) is the same concept as Δz : It is the change in the stock (z) from the previous time to the current time (again when multiplied by dt). Finally $sign\left(\frac{\Delta_x z}{\Delta x}\right)$ is replaced with simply +1 or -1 based on whether the flow is an inflow or an outflow, since the polarity of flows are fixed. These are the design elements that unite the two methods for calculating the link score, demonstrating that both measure the same concept.

As we do in Equation 1, we assume that the link score is 0 for all links from a flow to a stock if the net flow of the stock ($i - o$) is 0. As discussed above, the assumption of a 0 link score in this case does not change any loop scores since any link originating from a stock whose value does not change, will have a score of 0. We can safely set to 0 all link scores from a flow to a stock whose value does not change because link scores are multiplied into loop scores and any loop which passes through such a stock is inactive (and will have score 0).

In this formulation, a value of 0 for an inflow or outflow will result in a 0 link score. If the inflow and outflow are nearly balanced, so that there is only a small change in the stock, the scores for the links from the flows to the stock will be large, yet close in value. This happens because the denominator of Equation 2 approaches 0 faster than the numerator in such cases.

This formulation for capturing the effect of a flow on a stock is different from both the PPM and the Loop Impact method and is the characteristic of the LTM method that allows it to define a single number representing the importance of a whole loop. Both

the PPM and the Loop Impact method consider links involving integration in the same manner as they consider algebraic links. Specifically, they measure the change in a stock value relative to the change in value of the associated flow (the second derivative). That limits the scope of those methods so as only to address the effects of a loop on a single stock. Equation 2 directly uses the flow value relative to the change in value of the associated stock (the first derivative).

Link score computation examples

Table 1 demonstrates the process for calculating the link score magnitude for an auxiliary (non-stock) variable. It uses the equation $z = 2x + y$ to demonstrate how to calculate a link score magnitude. In this specific case, there are two link score magnitudes that must be calculated, one for the link $x \rightarrow z$ and one for the link $y \rightarrow z$. To calculate the link score magnitude for the link $x \rightarrow z$, first determine Δz which is the actual change in z (5). Next determine the partial change in z with respect to x , represented with the symbol $\Delta_x z$, by substituting into the equation for z the previous value of y (4) and the current value of x (7). Then take the computed value of z using those values (18) and subtract from it the previous value of z (14) to yield $\Delta_x z$ (4). To complete the calculation of the link score magnitude, divide $\Delta_x z$ (4) by Δz (5) to get the result that 4/5ths of the change in z is caused by the change in x .

Table 1: Components necessary to calculate the link score magnitude for the links $x \rightarrow z$ and $y \rightarrow z$ based on the equation $z = 2x+y$.

Variable	Time 1	Time 2	Variable Change	Partial Change in z	Link Score Magnitude
x	5	7	$\Delta x = 2$	$\Delta_x z = 4$	$\left \frac{\Delta_x z}{\Delta z} \right = \frac{4}{5}$
y	4	5	$\Delta y = 1$	$\Delta_y z = 1$	$\left \frac{\Delta_y z}{\Delta z} \right = \frac{1}{5}$
$z = 2x + y$	14	19	$\Delta z = 5$	-	-

The absolute value and sign are used in the link score definitions because the change in the dependent variable may be positive or negative for reasons unrelated to the change in the independent variable. For example, in Table 2 Δz is negative while the partial change in z ($\Delta_x z$) is positive, therefore the link score magnitude before taking the absolute value ($\frac{\Delta_x z}{\Delta z}$) is negative even though x has a positive influence on z . In Table 2, incorrect polarities would result if we did not use the absolute value in computing the link score magnitude.

To measure polarity and complete the link score calculation, we must multiply the link score magnitude by the polarity ($\text{sign} \left(\frac{\Delta_x z}{\Delta z} \right)$) as defined by Richardson (1995). In Table 2, we apply this to obtain the correct polarities. To calculate the polarity of the link $x \rightarrow z$, start by following the same procedure as in Table 1 to calculate the partial change in z ($\Delta_x z$). That procedure yields a $\Delta_x z$ of $2/3$ (0.67). Next determine the change in x which, in this case, is 2 . Finally take the sign of $\Delta_x z$ (0.67), divided by Δx (2), which is $+1$. This is the polarity for the link $x \rightarrow z$.

Table 2: Demonstration of correct polarity when calculating the link score magnitude for the links $w \rightarrow z$, $x \rightarrow z$, and $y \rightarrow z$ based on the equation $z = (w + x)/y$.

Variable	Time 1	Time 2	Variable Change	Partial Change in z	Link Score Magnitude	Link Polarity	Link Score
w	7	10	$\Delta w = 3$	$\Delta_w z = 1$	$\left \frac{\Delta_w z}{\Delta z} \right = 5$	$sign\left(\frac{\Delta_w z}{\Delta w}\right) = +1$	$\left \frac{\Delta_w z}{\Delta z} \right \cdot sign\left(\frac{\Delta_w z}{\Delta w}\right) = 5$
x	2	4	$\Delta x = 2$	$\Delta_x z = 0.67$	$\left \frac{\Delta_x z}{\Delta z} \right = 3.33$	$sign\left(\frac{\Delta_x z}{\Delta x}\right) = +1$	$\left \frac{\Delta_x z}{\Delta z} \right \cdot sign\left(\frac{\Delta_x z}{\Delta x}\right) = 3.33$
y	3	5	$\Delta y = 2$	$\Delta_y z = -1.2$	$\left \frac{\Delta_y z}{\Delta z} \right = 6$	$sign\left(\frac{\Delta_y z}{\Delta y}\right) = -1$	$\left \frac{\Delta_y z}{\Delta z} \right \cdot sign\left(\frac{\Delta_y z}{\Delta y}\right) = -6$
$z = \frac{(w+x)}{y}$	3	2.8	$\Delta z = -0.2$	-	-	-	-

Defining loop scores

The loop score is defined below in Equation 3 and where L_x refers to the loop x being studied. $LS(s_1 \rightarrow t_1)$ refers to the link score for the first link in the loop from the independent source variable s_1 to dependent target variable t_1 . Loop scores are calculated by multiplying all link scores in the loop from that first one, $s_1 \rightarrow t_1$, to the last, $s_n \rightarrow t_n$, where n is the number of links in L_x (so that t_n is the same as s_1). Note that this expression multiplies both the magnitude and the sign of the different link scores, - with an odd number of negative links yielding a negative loop and an even number yielding a positive loop. The multiplication of link scores is consistent with the chain rule of differentiation as demonstrated in Appendix II and accurately represents the nullifying effect of an inactive link in an otherwise active loop. This means that any loop containing an inactive link is assigned the loop score 0.

$$Loop\ Score(L_x) = (LS(s_1 \rightarrow t_1) \cdot LS(s_2 \rightarrow t_2) \dots \cdot LS(s_n \rightarrow t_n)) \quad (3)$$

The loop score, like the link score, is a dimensionless quantity. Just as the link score can be thought of as the force that one variable applies to another, the loop score can be thought of as the force one feedback loop applies to the behavior of all the stocks (and hence all the variables) it connects.

The definition of loop score is distinct from the Loop Impact of Hayward & Boswell (2014) because in the Loop Impact method the products of impacts equals the loop gain, whereas the loop score will always compute to 1 in an isolated loop as discussed in Appendix II. When comparing all loops, the loop score measures the relative importance of each feedback loop to the behavior of a model, rather than to the behavior of a single stock as is done in PPM and the Loop Impact method.

In order to establish a common baseline for comparing the contribution of feedback loops, we need to identify which feedback loops to include in the comparison. For every example presented in this paper, we analyze all feedback loops organized by cycle partition (Oliva, 2004). For models with a single cycle partition (where every

stock in the model has a path to and from every other stock in the model), we compare the loop score across all loops in the model. For models where this is not true, and there are many cycle partitions we only compare the loop scores across all loops which effect the same sub-set of all the stocks in the model. The LTM method does not require that loops be independent for the analysis to be valid and therefore does not restrict analysis to the minimal independent loop set identified by Oliva (2004), or any other identified set of loops. We leave for future work to determine how to best reduce the feedback loop complexity of larger models as discussed by Kampmann (2012) and note that the work of Güneralp (2006) and Huang et. al, (2012) show that restricting loops under consideration can filter out important loops. In what follows, we consider all identified connected loops, independent or not in the topological sense.

Once the set of feedback loops being analyzed has been determined, we define the relative loop score of each loop by dividing the loop score by the sum of the loop scores of all loops in the cycle partition. Equation 4 shows this computation for the loop X normalized over all loops n analyzed in the chosen cycle partition. The sign of a relative loop score still represents the polarity of the feedback loop in question. The relative loop score is a normalized measure taking on a value between -1 and 1. It reports the polarity and fractional contribution of a feedback loop to the change in value of all stocks at a point in time. By comparing relative loop scores, we determine which loops contribute the most to the behavior of all stocks in the cycle partition under study.

$$\text{Loop Score}_{L_X} = \left(\frac{\text{Loop Score}(L_X)}{\sum_{Y=0}^n |\text{Loop Score}(L_Y)|} \right) \quad (4)$$

Loop scores, like link scores, can become very large and difficult to interpret without this normalization. This is especially true as an equilibrium is approached because the denominator of Equation 2 (the net flow of a stock) approaches zero faster than any individual flow. The same is true of Equation 1 where the change in the target

variable, z , may approach zero regardless of the large partial change in the target with respect to the source. We demonstrate the asymptotic behavior of the loop score in the analysis of the Bass diffusion model. In that case, even though the score of the loops effectively approaches infinity, the transition from positive to negative loop dominance is smooth and clearly visible when using relative loop scores because they are normalized. An example of loop score values that can't be meaningfully compared is shown below in the case of the inventory workforce model, where a feedback loop for smoothing demand is not comparable with the others since it is not coupled with inventory-workforce adjustment process.

Computational considerations

Using the LTM method, we make our computations as time progresses in the model. The first computation can be made only after the model has been initialized and moved forward in time. In the results we present, we use the model's dt or time step to determine how often to compute link and loop scores. This is most straightforward using the Euler integration method. In principle the computation could proceed also at a longer or shorter sampling interval, allowing it to work with other integration methods such as Runge-Kutta.

The computational efficiency of this method has not been examined in depth and, in particular, it has not been analyzed in the case of large models with hundreds of stocks and millions of feedback loops. For models of smaller size and complexity (between 2 and 20 stocks, and less than 50 feedback loops), we have found through experience that the largest computational burden is not caused by what is required to calculate the link and loop scores, but rather by the calculations required to identify the full set of feedback loops.

An implication of the calculation method we present, is that the equations in the model will be computed not just once as it is typically done to simulate a model, but repeated once for each independent variable in the equation. This can multiply the number of computations by 2 or more (depending on equation complexity), which is similar to the computational requirements for linearization. In short, however, the

computation times are modest and quite similar to simply simulating the model. For reference, the analysis of all of the models in the paper, including Forrester's 10 stock market growth model takes less than 1 second, including the time to parse the XMILE representation of the model, find all the loops, partition the cycles, and calculate all loop dominance metrics presented.

The computation of the link and loop score metrics, nonetheless, does require that equations be computed multiple times per dt and this cannot be done in standard software. For this paper, we have modified an open source and publicly available simulation engine sd.js (Powers, 2019) to simulate the model and perform the link and loop score calculations. The pseudo code for this computation is shown below in Figure 1.

```

for (let target in model.variables) {
    let value = target.currentValue;
    let previousValue = target.previousValue;

    if (target.isStock) {
        let sumOfFlows = 0;
        for (let source in target.sources) {
            if (target.isInflow(source))
                sumOfFlows += source.previousValue;
            else
                sumOfFlows -= source.previousValue;
        }

        for (let source in target.sources) {
            if (sumOfFlows == 0) {
                LINKSCORE[source,target] = 0;
            } else if (target.isInflow(source)) {
                LINKSCORE[source,target] = ABS(source.previousValue / sumOfFlows);
            } else {
                LINKSCORE[source,target] = -ABS(source.previousValue / sumOfFlows);
            }
        }
    } else if (value == previousValue) {
        for (let source in target.sources) {
            LINKSCORE[source,target] = 0;
        }
    } else {
        for (let source in target.sources) {
            let tRespectSource = <calc. target, use current source, prev. of rest>;
            let deltaTRespects = tRespectSource - previousValue;
            let deltaSource = source.currentValue - source.previousValue;
            let deltaTarget = value - previousValue;
            let sign = 1;

            if (deltaSource != 0 && deltaTRespects != 0) {
                sign = SIGN(deltaTRespects / deltaSource);
            }

            LINKSCORE[source,target] = ABS(deltaTRespects / deltaTarget) * sign;
        }
    }
}
}

```

Figure 1: Pseudo code for calculating all link scores in a model after calculating a dt of the model

Figure 1 shows how we walk through all the variables in the model calculating the link scores for all links coming in to each variable. First we check, for each variable (*target*), whether it is a stock or not. If the *target* is a stock then we directly apply Equation 2 to determine the link score from each flow (*source*) into the target. If *target* is not a stock, we apply Equation 1. When applying Equation 1 for each *source / target* combination, the first thing we do is determine what the value for *target* would have been if only *source* changed (*tRespectSource*). To calculate

tRespectSource, we call another subroutine which is capable of recalculating *target* using the current value of *source*, and the previous value of all other variables. Using *tRespectSource*, we determine the partial change in *target* with respect to *source* (*deltaTRespectS*). The change in *source* and *target* are straight forward calculations using the current and previous values of *target* and *source*. Finally, we calculate the sign, making sure we do not produce a divide by zero, and then the link score, using the already calculated values.

Application of the LTM method to the Bass diffusion model

We have used a variant of the Bass diffusion model (Bass, 1969), pictured in Figure 2 below, to demonstrate the ability of the LTM method to reproduce the standard explanation for the behavior of that model. Richardson (1995), using his dominant polarity method, says the following about how a logistic model (e.g. the Bass diffusion model) works:

In the logistic equation, a shift in loop dominance occurs when the level reaches half its maximum value, the point of inflection in the logistic curve.

This understanding has been confirmed by Kampmann and Oliva (2017) who apply traditional, manual, loop dominance analysis to reach this same conclusion.

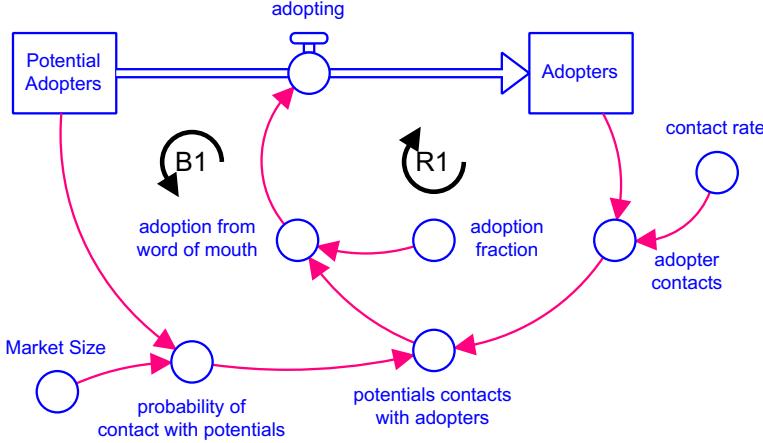


Figure 2: The stock and flow structure of the Bass diffusion model analyzed

This version of the Bass diffusion model runs from Time 0 to Time 15 with the inflection point reached between time 9.5625 and 9.625. The value of Market Size is 1,000,000 people with one initial adopter. The value of the contact rate is 100 people per person per year, and the value of the adoption fraction is 0.015 (dimensionless). The equations for this model follow the standard formulation. This model contains two loops, one balancing and one reinforcing.

- Balancing (B1)
 - probability of contact with potentials
 - potentials contacts with adopters
 - adoption from word of mouth
 - adopting
 - potential adopters

- Reinforcing (R1)
 - adopter contacts
 - potentials contacts with adopters
 - adoptions from word of mouth
 - adopting
 - adopters

The results of the LTM analysis which appear in Table 3 and Figure 3 demonstrate that LTM reproduces the same standard explanation for behavior as Richardson (1995), Kampmann and Oliva (2017) report. In Table 3 the calculation of the loop

score and relative loop score of B1 and R1 at specific points in time is demonstrated. Table 3 shows that the two loops shift in dominance during the time between time 9.5625 and 9.625 which is when the inflection occurs. In addition, Table 3 confirms that the proper polarity is assigned to each loop and link. Figure 3 supports the standard explanation of the model's behavior by showing that the relative loop score magnitude for both loops passes through 0.5, the threshold for dominance, at the inflection point.

Table 3: Loop scores and relative loop scores in the Bass diffusion model calculated to 4 significant digits

Link	T ₁	T _{9.5}	T _{9.5625}	T _{9.625}	T ₁₅
probability of contact with potentials → potentials contacts with adopters	0.000	9.958	9358	10.91	1.000
potentials contacts with adopters → adoption from word of mouth	1.000	1.000	1.000	1.000	1.000
adoption from word of mouth → adopting	1.000	1.000	1.000	1.000	1.000
adopting → potential adopters	-1.000	-1.000	-1.000	-1.000	-1.000
Potential adopters → probability of contact with potentials	1.000	1.000	1.000	1.000	1.000
B1 Loop Score	0.000	-9.958	-9358	-10.91	-1.000
B1 Relative Loop Score	0.000	-0.465	-0.488	-0.512	-1.000
adopter contacts → potentials contacts with adopters	1.000	11.46	9806	10.41	0.000
Adopters → adopter contacts	1.000	1.000	1.000	1.000	1.000
adopting → Adopters	1.000	1.000	1.000	1.000	1.000
potentials contacts with adopters → adoption from word of mouth	1.000	1.000	1.000	1.000	1.000
adoption from word of mouth → adopting	1.000	1.000	1.000	1.000	1.000
R1 Loop Score	1.000	11.46	9806	10.41	0.000
R1 Relative Loop Score	1.000	0.535	0.512	0.488	0.000

Table 3 identifies which links contribute most critically to the feedback loop score changes. The majority of links have a link score of 1.000 because the target variables for those links only have a single variable which points to them. This is true of both the auxiliaries and the stocks. The only link score which exhibits a change in contribution over time in the loop B1 is the link ‘Probability of contact with potentials → potentials contacts with adopters’. This is because that link is the only link which has a changing score. It is the key link in the loop B1 and is responsible for the changes in B1’s loop score and is therefore critical to the overall shift in loop dominance. The link (‘adopter contacts→ potentials contacts with adopters’) is its counterpart in the loop R1 and it is just as critical to the overall shift in loop dominance from R1 to B1. Figure 2 shows that these two links are located at the junction between the reinforcing and balancing feedback loops. Independently, by just examining the structure, we can confirm that these links are most important with regard to the model behavior because these links points to the variable where the feedback loops interact with each other after tracing through the loops starting with the stocks.

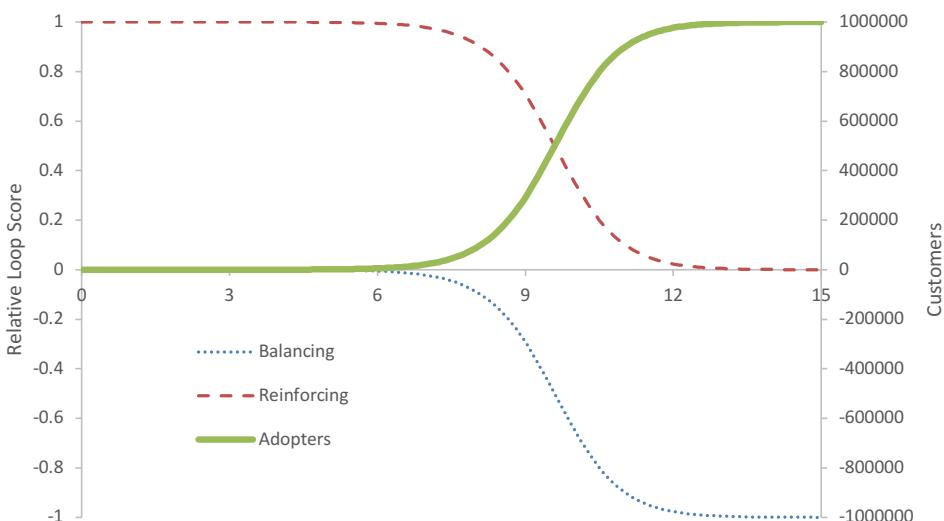


Figure 3: Bass diffusion relative loop scores plotted over time against Adopters.

If we were to simulate this model with a dt approaching 0, then the loop score magnitude for both loops, R1 and B1, would approach infinity at the inflection point. This happens because the Δz values from the link scores (Equation 1) for the two links into ‘potentials contacts with adopters’ approach 0, so that both loops have a score approaching infinity. In single stock systems, instants in time when loop scores approach infinity, represent shifts in the feedback loop dominance of the model, which in the case of the bass diffusion model is when both loops are pushing very hard to change behavior, and are, therefore, canceling each other out. This is markedly different from what we find using PPM or the Loop Impact method where the infinities occur at maximum and minimum stock values (when curvature is no longer changing). In the PPM-based approaches the zeroes represent the inflection points.

The loop score values in Figure 4 for the Bass diffusion model demonstrates why loop scores are compared in a relative fashion. When ‘potentials contacts with adopters’ passes through its maximum, Figure 4, which is on a log scale, shows that the absolute value of both loop scores approaches infinity. The drastic change in scale makes Figure 4 ineffective with regard to quickly and accurately determining the dominant loops in the system, but it does demonstrate the magnitude of the effort the loops are expending to change the stocks at each point in time. When the loop scores for a positive and negative loop in a model are both high, both loops are strongly contributing to behavior, working in opposite directions and cancel each other out resulting in a small change in the stocks. The non-normalized loop score metric, because of the dramatic magnitude changes, does not add much to our intuitive understanding of which loops are dominant. After being compared to the other loops, a simple and straightforward analysis can be conducted of the loop dominance profile of a model.

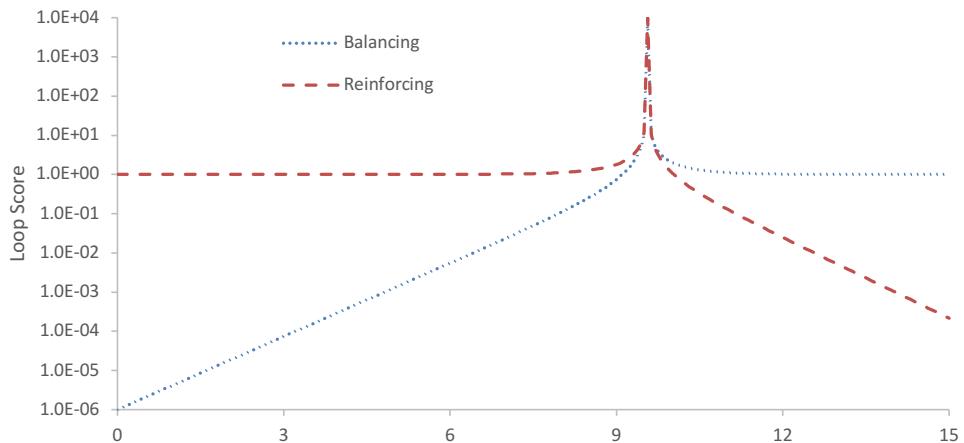


Figure 4: The base 10 logarithm of the absolute value of Bass diffusion loop score values

The LTM analysis of the Bass diffusion model has replicated the standard explanation for behavior in the model as performed by Richardson (1995) and Kampmann and Oliva (2017). In addition, the LTM analysis has identified the specific links in the loops that first and foremost explain the shifts in feedback loop dominance and has identified the overall effort the loops in the model are expending to change behavior.

Application of the LTM method to the yeast alcohol model

The yeast alcohol model is analyzed to demonstrate the efficacy of the LTM method. This analysis reinforces the notion that the LTM method is able to yield the same insights into behavior as previous analyses of this model using Ford's behavioral approach, PPM, Loop Impact and EEA (Saleh, 2002; Güneralp, 2005; Phaff et al., 2006; Mojtabahedzadeh, 2008; Hayward and Boswell, 2014).

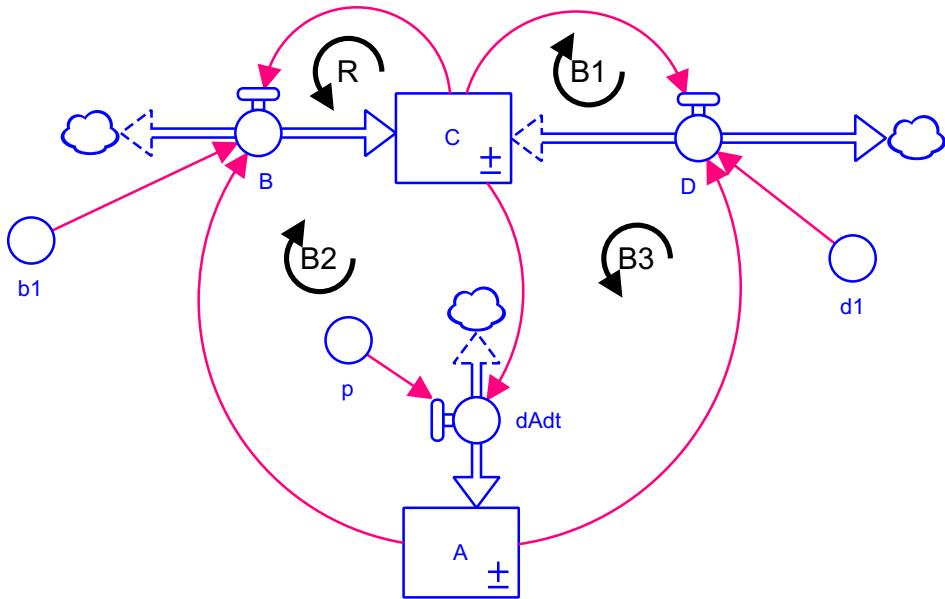


Figure 5: The stock and flow structure of the yeast alcohol model analyzed

Figure 5 shows the structure of the model as analyzed using a dt of .5. The model structure; $B = C*(1.1-0.1*A)/b1$, $D=C*\text{EXP}(A-11)/d1$, $dAdt = p * C$, is initialized with; $A=0$, $B=1$, $b1=16$, $d1=30$, and $p=0.01$. The model contains 4 loops, all in a single cycle partition. Loop R, represents the birth² (and, late in the simulation, the deaths) of the cells C, characterized by b1 and Alcohol (A). Loop B1 represents the natural death of the cells. The main link in Loop B2 represents the slowing of the birth of cells due to the presence of alcohol. The main link in Loop B3 represents the increasing death of cells due to the presence of alcohol. This model produces the overshoot and collapse behavior seen in Figure 5 which matches exactly the behavior generated by Phaff et al. (2006) and Mojtahezdzadeh (2008), and is very similar to the

² Notice that there is a flaw in the standard formulation of B in this model causing B to take negative values and the polarity of R to change so that it acts as an additional “deaths loop” under conditions of high levels of alcohol A. This flaw has not been corrected in order to maintain the consistency of the model across analyses in the literature.

results that Hayward and Boswell (2014) generated using their slightly altered structure of the model³.

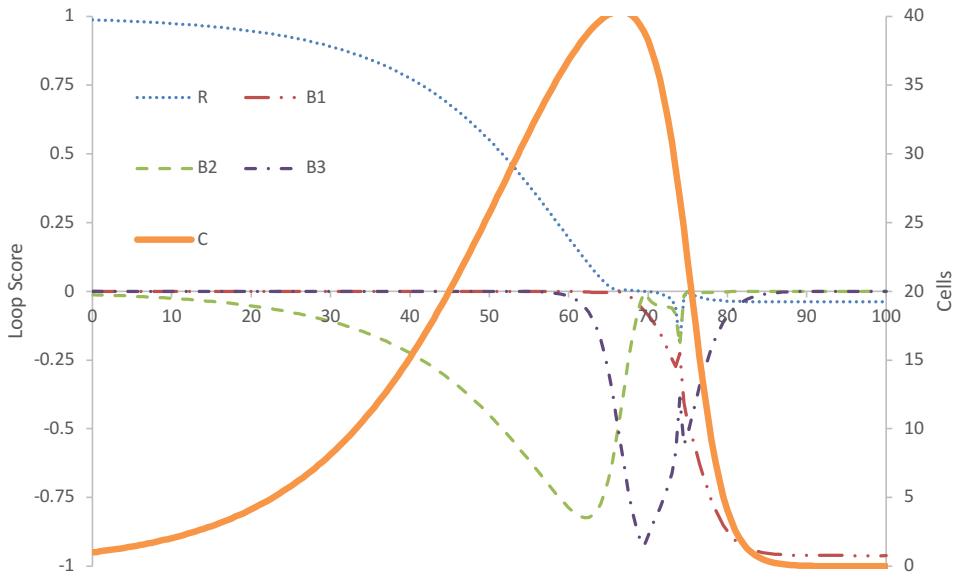


Figure 6: Yeast Alcohol loop scores plotted against the variable C (i.e yeast cells)⁴

³ Hayward and Boswell (2014) use the same parameterization of the yeast alcohol model as us and the others, but appear to have used a Stella version of this model where uniflows were used for B and D. This subtle change to structure corrects the formulation flaw in the births loop (referenced in footnote 1) and causes their model results and loop dominance analysis to differ slightly from the other analyses and our own.

⁴ At time 74 no single feedback loop is dominant because this is the point where R is at its strongest as a balancing feedback loop (because of the model formulation error in footnote 1). After time 70 when the birth rate is negative R is acting in a similar fashion as B1. At Time 74 summing the contribution of R & B1 yields a relative loop score which is stronger than B3, but still not over 50%, B3 is the single strongest feedback loop at that exact moment and we therefore consider it alone to be dominant across phase 3.

Table 4: Plot showing the relative loop scores against the variable C (yeast cells) which reveals the dominant loops in yeast alcohol model.

Time range	Phase 1: 0-51.5	Phase 2: 52-66	Phase 3: 66.5-75	Phase 4: 75.5-100
Dominant loop	R	B2	B3 ³	B1

Table 4 identifies the dominant loops for each phase of the model's behavior. Comparing these results with Ford's (1999) behavioral approach as applied by Phaff et al. (2006), the LTM method identifies exactly the same 4 phases and agrees with the behavioral analysis in principal. The LTM method's only disagreement is that phase 3 is dominated by B3 alone, not B2 and B3. This same difference is raised by the PPM approach of Mojtahedzadeh (2008) and the Loop Impact approach of Hayward and Boswell (2014) where PPM and the Loop Impact method identify that Phase 3 is dominated by B3 rather than B2 and B3 together as Phaff et al.'s implementation of Ford's behavioral approach analysis would suggest. This shows that the LTM analysis of this model matches Ford's behavioral approach with the noted discrepancy and matches exactly the PPM analysis done by Mojtahedzadeh. When we compare Table 4 to Hayward and Boswell's (2014) PPM based Loop Impact method, we agree in principal with their results, but the slight change in the structure of their model² prevents full agreement.

Our results in Figure 6 and Table 4 match the EEA analysis of this model performed by Phaff et al. (2006). That analysis shows that the behavior of Phase 1 is dominated by R with B2 restraining (increasingly) the growth of C. In phase 2 EEA shows that B2 is now dominant, but R is still a significant factor in explaining the growth in C which matches what LTM concludes. This can be seen in Figure 6 because B2 has a relative loop score less than -0.5 and R is the only other active loop until time ~60 where B3 is activated in preparation for phase 3. In phase 3, EEA points to B1 and B3 together as responsible for the behavior of C. The LTM analysis matches the EEA analysis, with the caveat that the LTM analysis finds that B3 is solely dominant throughout that time period. EEA then reports that during phase 4, B1 is dominant over B3 which again matches the LTM analysis. That can be seen in Figure

6 as the loop score of B1 starts growing quickly at the end of phase 3, reaching nearly -1 shortly after the start of phase 4.

Looking only at the progression of relative loop scores for each of the loops as shown in Figure 6, the LTM analysis reveals some additional interesting insights. The first of these is evident from Time 74 to 76 when R becomes a significant (but not dominant) balancing feedback loop (due to the model flaw). This is due to excess levels of Alcohol (A), causing the births process to run backwards, making R into an additional loops that drains C like B1 does. At this point the relative loop scores of B2 and B3 have maximums in magnitude. This happens because those loops trade their dominance off over the course of the simulation, one rising while the other is falling. R actually has a similar local maximum in magnitude at time 74 when it reaches its peak in contribution as a balancing feedback loop. The number of stocks in a feedback loop has no direct relationship to the number, or presence, of maximums in magnitude observed in relative loop scores.

The LTM analysis of the yeast alcohol model demonstrates a shared understanding of the model with both EEA, and PPM. Moreover, the understanding coming from Ford's behavior analysis technique is also shared, demonstrating that the LTM method yields the same level of insight into this model as these other techniques.

Application of the LTM method to the Inventory Workforce model, - understanding oscillations

To demonstrate that the LTM method performs appropriately on a full range of SD models we have analyzed a two state oscillatory model, - a version of the inventory workforce model originally proposed by Mass and Senge's (1975). Analyzing an oscillatory system differentiates the PPM based approaches from the LTM method and EEA based approaches. The version of this model analyzed is the one prepared by Gonçalves (2009) for his EEA model analysis. The model is shown in Figure 7. We use the analysis of this model also to demonstrate the impact of parameters on loop dominance patterns.

We compare and contrast the results of the LTM analysis with the ones produced by the EEA analysis Gonçalves (2009), and by two PPM-based analyses of the original Mass and Senge model by Mojtabahedzadeh (2008) and Hayward and Roach (2017). The only difference between Gonçalves' version of the model (the one we analyzed) and Mass and Senge's (1975) model is the addition of the feedback loop B3 in Figure 7 which does not significantly affect the cause of the oscillation, merely the exact shape of the oscillatory mode of behavior.

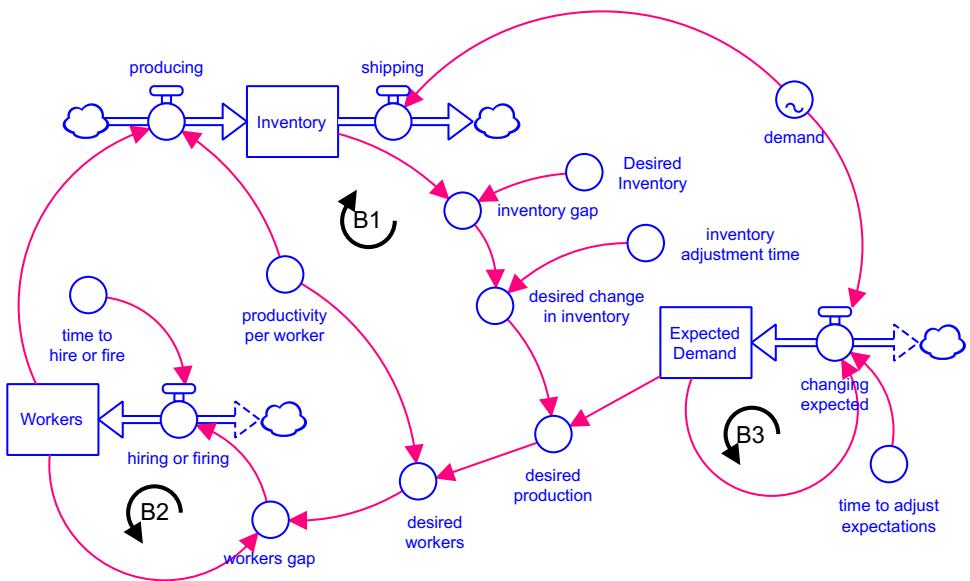


Figure 7: The stock and flow structure of the inventory workforce model analyzed

Our implementation of the Gonçalves (2009) model runs from Time 0 to Time 60. The model has only three balancing feedback loops that appear in two different cycle partitions. This split happens because inventory and workforce interact, but expected demand is driven only by demand itself. The graphical function inside of the 'demand' variable acts like a step function, triggering a single increase in demand between times 1 and 2 which sets off a damped oscillation in both workers and inventory. The loops of this model are shown in the list below, organized by cycle partition, containing loops that have loop scores comparable to each other, as exhibited in Figure 8.

- Cycle Partition 1
 - Major Balancing (B1)
 - Inventory
 - inventory gap
 - desired change in inventory
 - desired production
 - desired workers
 - workers gap
 - hiring or firing
 - Workers
 - producing
 - Minor Balancing (B2)
 - Workers
 - workers gap
 - hiring or firing
- Cycle Partition 2
 - Expected Demand Loop (B3)
 - Expected Demand
 - changing expected

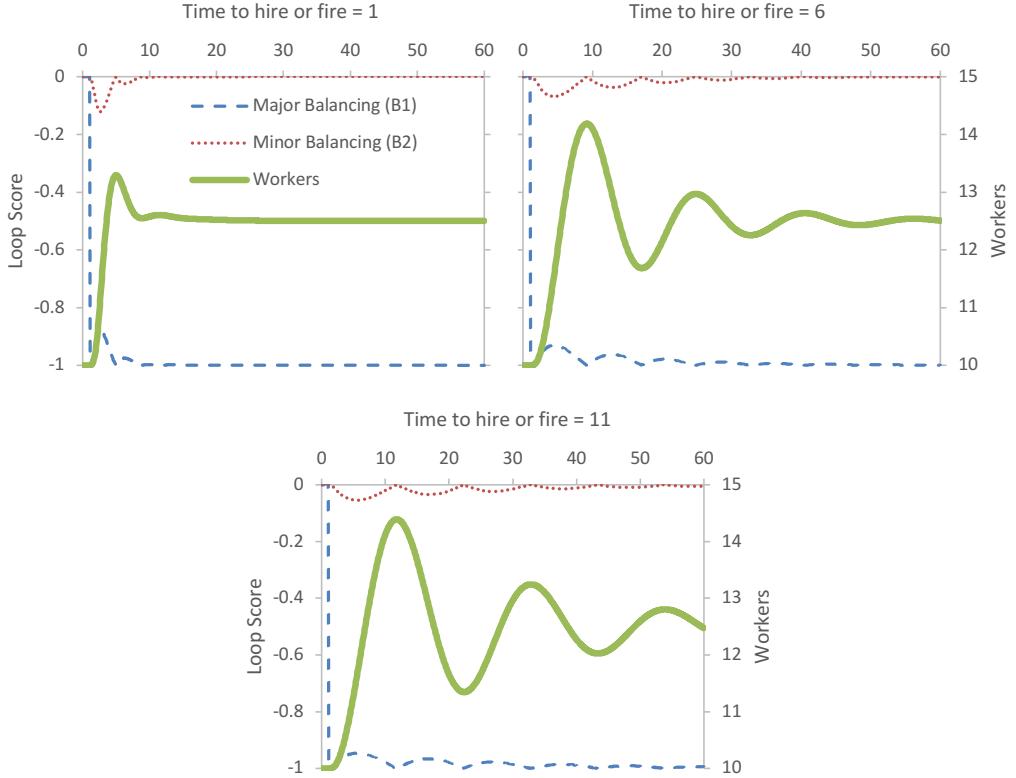


Figure 8: Results of LTM analysis of the Inventory Workforce model showing the effect of time to hire or fire on loop dominance and Workers.

Before the explanation of the results, note in, Figure 8, the time period before the shock in demand. During that period the model is in equilibrium, unchanging.

Therefore the LTM method cannot inform the analysis of the model because all link scores are 0.

The two loops in this model that contain the stocks with the oscillatory behavior are B1 and B2 of Set 1. As shown in Figure 8, in all three parameterizations of the model the dominant loop describing the large majority of the change in the behavior of the worker and inventory stocks, and, therefore, the oscillations in the model is dominated by the major balancing loop B1. Figure 8 shows that there is a contribution by B2, which is dependent on the value of the parameter time to hire or fire (affecting the shape of the damped oscillation), even though B1 is responsible

for the oscillation. The longer B2 is active, the more pronounced the oscillations are. This tells us that by increasing the time to hire or fire, we increase the contribution of B2 (relative to B1). While it is true that time to hire or fire is strongly tied to the contribution of B2, it also directly impacts the contribution of B1 (i.e. independent of its impact on B2). Figure 8 shows the total relative effects of the change in this parameter on the relative contributions of B1 and B2. The net total effect of time to hire or fire on B1 and B2 causes the oscillations to become more pronounced (less damped) and to last longer.

The LTM method's conclusions about the impacts of time to hire or fire on the oscillation matches both the conclusions derived by EEA and PPM, - but differs from the PPM-based analyses conducted by both Mojtahedzadeh (2008) and Hayward and Roach (2017) in its explanation of dominant loops. Gonçalves (2009) EEA analysis of this model shows that the oscillatory mode of behavior arises primarily from the loop gains associated with the impacts from the major loop B1 and the damping effect is a function of the minor loop B2. The PPM-based methods show that the behavior of the model is dominated by both B1 and B2 in a cyclical process. Mojtahedzadeh (2008) states that the loop dominance pattern reported by the PPM method is not suitable for analyzing the causes of oscillation in this model, and that, instead, the explanation of the oscillation must be based on pathway frequency and stability factors which are PPM values taken at different points in a single cycle of the oscillation. Using the values of the PPM at those specific timepoints, he concludes that the source of the oscillation is B1, and that B2 is responsible for the dampening effect. This conclusion matches the results from both the EEA and the LTM analyses. The problem with PPM's shifting loop dominance pattern in oscillatory models is explained by Kampmann and Oliva (2008). They point out that methods based solely on the PPM for determining loop dominance are problematic in oscillatory systems like the inventory workforce model (sinusoidal oscillators) because the sign of the PPM changes even though the relative contribution of the loops in the model remains constant.

The LTM analysis of the inventory workforce model has demonstrated the efficacy of the method in the analysis of oscillatory systems. The analysis of the inventory workforce model shows that the LTM method produces loop dominance patterns explaining oscillations which are the same as the EEA analyses. PPM-based analyses require more complex metrics to identify a single feedback loop as being responsible for the oscillatory mode of behavior. The LTM analysis produces conclusions about the causes of oscillation and the impacts of parameters on those oscillation that match the conclusions from both EEA- and PPM-based analyses.

Discussion and conclusions

As demonstrated, the LTM method provides an easy computational way to identify and understand, which feedback loops in a model dominate the model behavior or, in other words, contribute the most to the behavior of all the stocks in the model at any specific point in time. Dominance is typically model wide (with the noted exception of feedback loops that are not coupled), based on the effect on all variables, and is typically driven by the stocks that are changing most quickly in proportion to each other. As we have seen throughout the examples presented, the relative loop score measure of dominance coincides well with our structure-based understanding of relatively simple systems. The LTM method offer several important benefits.

The most important benefit of the LTM method is that it is generally applicable to all models without modification. Although, both EEA and PPM, may, in principle, be applied to all models, the LTM method has the advantage of being directly applicable to discrete and discontinuous models. This is because LTM analyses are conducted over time in lock step with the behavior produced by the model, do not depend on model modifications and use only values resulting from computations that are conducted as part of the normal simulation process.

The second considerable benefit of the LTM method is that the format of the results of the analysis are simple, easily interpretable graphs of behavior over time. The LTM method makes use of the existing skill-sets of all modelers and most model consumers and is thus easily accessible to all. The LTM method's metrics are

calculated and reported over time. Therefore it takes no additional skill or training to read and interpret the graphs of the relative loop and link scores. Considerable work has been done on both PPM and EEA to define relatively easy to understand and interpret metrics. But we believe that the simplicity and accessibility of the link and loop scores provide an easier path to this type of analysis for most people.

The third noteworthy benefit to the LTM method is its relative computational and conceptual simplicity which is a major advantage over both EEA and PPM. As currently developed, the method does not use complex mathematical constructs that are not already in use by the majority of practitioners. From a mathematical perspective, the concept of the partial change in $z(\Delta_x z)$ is the most challenging part of the method because of its unfamiliar terminology, and not necessarily because of any inherent complexity in the idea itself. The advantage of such a simple method is that it can be understood by all practitioners so that when it comes time to apply the method, practitioners will know ‘what it is saying’ (i.e. how to interpret the results) due to the transparency of the method.

The fourth and final benefit of the LTM method is that it is relatively easily implemented in existing simulation engines without requiring modifications to existing structures within those engines. This represents a considerable advantage over EEA and PPM, and is on par with Loop Impact method. We base this conclusion upon our own experiences implementing simulation engines in the past, - including the engines behind Stella and Vensim, and the level of effort it has taken to modify Powers’ sd.js engine. This means its uptake should be relatively painless by software vendors in the field.

A weakness in the LTM method is that it is may not be used to determine loop dominance without a change in the model state. As a model approaches equilibrium, we can see the loop scores balance one another even as they become unbounded, but when a model is in equilibrium, all loop scores are defined to be 0. Therefore, models in equilibrium cannot be analyzed using the LTM method. An example of this is a simple “bathtub” population model where the birth fraction equals the death fraction.

The limitations of the link score causes the loop score for both loops to be 0 because there is no change across time. This is a disadvantage relative to EEA which, if the model is close to linear, will provide accurate information under equilibrium conditions. An unsatisfactory solution to this problem from a purely methodological perspective is to start introducing minute changes in these situations in order to measure those changes' effects on loop dominance. But doing so would have major ramifications on the utility of the method for discrete and discontinuous models where information is likely encoded as specific, logically meaningful integer values. An alternative approach would be for the model author to offset the model state from its equilibrium using a STEP function or some other modeling construct so as to expose dynamic behavior.

An additional weakness of the LTM method, which some may consider to be a strength, is that it focuses exclusively on endogenously generated behavior. Such a focus is a hallmark of System Dynamics, but is problematic for models where behavior is driven through external forcing functions that dominate the effects of feedback in the model. The inventory workforce model contains some elements of this because of the external demand signal which necessitates the separation of the feedback loops into two sets. But in the inventory workforce case, the external signal was only required to start the oscillation. Loop dominance, in highly forced models (to a much greater extent than in the case of the inventory workforce model) may have little to do with behavior endogenously generated by the feedback loops. Models of this sort are currently much better analyzed using the Loop Impact method of Hayward and Boswell (2014). Although, the link score metric could be used to measure these exogenous contributions, and with future work, the LTM method could properly analyze models of this kind.

Future work

There are a variety of interesting extensions to the LTM method that combine it with other analysis techniques. The most obvious one is to combine it with Monte Carlo analysis so that the realized behavior sets encompass more of the potential behavior

sets. Another is during extreme condition testing one could use the LTM method to show that the model is producing the right results for the right reasons. The LTM method could also be combined with optimization, for example using optimizers to maximize or minimize loop scores. This would allow practitioners to maximize the activity of favorable loops while minimizing the activity of unfavorable loops in order to automatically generate better, more robust policy recommendations. Another area of study would include loop scores in the outputs of Monte-Carlo sensitivity analyses which would allow us to measure the robustness of loop dominance to policy or parameter changes. Monte Carlo analysis could also be used to measure the sensitivity of loop score to changes in parameter values.

A second critical area of future work is to use the LTM method to development new and exciting visualization tools, including animated stock-and-flow diagrams, where the links and flows change color and size due to changes in polarity or link score, in response to the call for such graphics by Sterman in *Business Dynamics* (2000). Going even further, the LTM method allows for the possibility of automated CLD generation and animation. Because the LTM method is able to tell, on a link-by-link basis, which are the key (dynamic) links in the model, it is possible, using the method, to automatically generate a CLD collapsing all of the ‘unimportant’ static links with scores of 0, +1.0, or -1.0 into links that are conveying a change. This will allow for an automated generation of structurally correct, minimal CLDs that accurately portray the structural components that predominantly produce the dynamics of the model, and are laid out by the computer according to best practices.

Finally, it is necessary to test and analyze larger and more varied models if we are to increase our confidence in the general utility of the LTM method. We are hopeful that the techniques laid out in this paper, will offer a significant utility and enhance the analysis and understanding of a wide set of SD models by SD users.

Appendix I: Link Scores and Partial Derivatives

We have presented the computational equations for loop scores and link scores using separate magnitude and sign components. While this is the most intuitive

presentation, it is also possible to recast the equations using partial differences which allows easier comparison with other model analysis methods.

It is straightforward to manipulate Equation 1 to have the form shown in Equation 5. Equation 5 demonstrates the link score computed using a partial difference and magnitude adjustment.

$$LS(x \rightarrow z) = \begin{cases} \left(\frac{\Delta_x z}{\Delta x} \cdot \left| \frac{\Delta x}{\Delta z} \right| \right), \\ 0, \quad \Delta z = 0 \text{ or } \Delta x = 0 \end{cases} \quad (5)$$

Here the first term is just the partial difference, which has the direction of change and the sensitivity of z to x . The second term adjusts this by the realized changes so that the link score reflects the amount of contribution as opposed to the sensitivity.

Dividing both the top and bottom of the second term by Δt gives Equation 6.

$$LS(x \rightarrow z) = \begin{cases} \left(\frac{\Delta_x z}{\Delta x} \cdot \left| \frac{\Delta x / \Delta t}{\Delta z / \Delta t} \right| \right), \\ 0, \quad \Delta z = 0 \text{ or } \Delta x = 0 \end{cases} \quad (6)$$

If the model being analyzed (including $f(x, y)$) is continuously differentiable, then as Δt approaches 0, so too will Δx and we can write Equation 6 as Equation 7 which is a simplified continuous representation of the link score metric:

$$LS(x \rightarrow z) = \begin{cases} \left(\frac{\partial z}{\partial x} \cdot \left| \frac{\dot{x}}{\dot{z}} \right| \right), \\ 0, \quad \dot{z} = 0 \text{ or } \dot{x} = 0 \end{cases} \quad (7)$$

Equation 7 *demonstrates* the relationship of the link score to the partial derivative which are key to Pathway Participation Metrics and the Loop Impact, which also form the basis for the linearized representation of model equations used to compute

eigenvalues. The addition of the second term converts the potential contribution to the realized contribution. This is why the loops that matter method gives results that are largely in line with other approaches when normalized, but at the same time are not identical in absolute value

Appendix II: Important Analytic Characteristics of Link Scores

There are two notable characteristics of link scores that it is useful to elucidate. The first is invariance under formulation, something that is relied upon when computing loop scores. The second is the observation that isolated loops will always have a loop score of 1 or -1.

Consider invariance under formulation changes. Put simply, it should not matter whether we connect two variables with one complicated equation, or three variables with two simpler equations. To demonstrate this, without loss of generality, suppose that there are two formulations for the variable z . The first is directly as $f(w, x, y)$, the second indirectly as $g(u, y)$ where $u = h(w, x)$. In this case we first compute the link score from x to u as shown in Equation 8, then the link score from u to z as shown in Equation 9.

$$LS(x \rightarrow u) = \begin{cases} \left(\left| \frac{\Delta_x u}{\Delta u} \right| \cdot sign \left(\frac{\Delta_x u}{\Delta u} \right) \right), \\ 0, \quad \Delta u = 0 \text{ or } \Delta x = 0 \end{cases}$$
(8)

$$LS(u \rightarrow z) = \begin{cases} \left(\left| \frac{\Delta_u z}{\Delta z} \right| \cdot sign \left(\frac{\Delta_u z}{\Delta z} \right) \right), \\ 0, \quad \Delta z = 0 \text{ or } \Delta u = 0 \end{cases}$$
(9)

The composite link score (a link score multiplied along a causal pathway) is computed as the product of the two link scores as shown in Equation 8 and Equation 9, which can be rewritten as Equation 10 by multiplying by $\left| \frac{\Delta x}{\Delta x} \right|$ which is just 1.

$$LS(x \rightarrow z) = \begin{cases} \left(\left| \frac{\Delta_x u}{\Delta u} \right| \cdot \left| \frac{\Delta_u z}{\Delta z} \right| \cdot sign\left(\frac{\Delta_x u}{\Delta x}\right) \cdot sign\left(\frac{\Delta_u z}{\Delta u}\right) \right), \\ 0, \quad \Delta z = 0 \text{ or } \Delta u = 0 \text{ or } \Delta x = 0 \end{cases} \quad (10)$$

$$LS(x \rightarrow z) = \begin{cases} \left(\left| \frac{\Delta_x u}{\Delta x} \right| \cdot \left| \frac{\Delta_u z}{\Delta u} \right| \cdot \left| \frac{\Delta x}{\Delta z} \right| \cdot sign\left(\frac{\Delta_x u}{\Delta x}\right) \cdot sign\left(\frac{\Delta_u z}{\Delta u}\right) \right), \\ 0, \quad \Delta z = 0 \text{ or } \Delta u = 0 \text{ or } \Delta x = 0 \end{cases} \quad (11)$$

Applying the chain rule for partial differences this is the same Equation 11.

Cancelling the Δx terms we get the expression in Equation 12.

$$LS(x \rightarrow z) = \begin{cases} \left(\left| \frac{\Delta_x z}{\Delta x} \right| \cdot \left| \frac{\Delta x}{\Delta z} \right| \cdot sign\left(\frac{\Delta_x u}{\Delta x}\right) \cdot sign\left(\frac{\Delta_u z}{\Delta u}\right) \right), \\ 0, \quad \Delta z = 0 \text{ or } \Delta u = 0 \text{ or } \Delta x = 0 \end{cases} \quad (12)$$

After reduction, the composite link score is shown as Equation 13.

$$LS(x \rightarrow z) = \begin{cases} \left(\left| \frac{\Delta_x z}{\Delta z} \right| \cdot sign\left(\frac{\Delta_x u}{\Delta x}\right) \cdot sign\left(\frac{\Delta_u z}{\Delta u}\right) \right), \\ 0, \quad \Delta z = 0 \text{ or } \Delta u = 0 \text{ or } \Delta x = 0 \end{cases} \quad (13)$$

The *sign* term in Equation 12 and Equation 13 should be clear, since it just tracks the directional change, and thus we have the same result we would get with all the computation in a single equation.

This chaining is the same as that observed in Richardson (1995) and Kampmann (2012) and allows us to look at models as they are constructed, and not in any condensed form.

It is worth noting that this equivalence fails if $\Delta u = 0$, even when both Δx and Δz are nonzero. That is, if the intermediate variable is not changing, the link score becomes 0 even when the input and ultimate output are changing. It is easy to construct models that have this characteristic (in fact the Bass Diffusion model can be written with total population computed by adding the two stocks), and the 0 value is helpful at showing that the potential feedback is not real.

The second observation on loop scores is that for a single positive or negative loop the score will be +/-1, though the gain around the loop could be significantly different. This is easy to see in a model of (net) population growth, since the link score from the stock to the flow will be 1 as only the stock is changing the flow, and with only a single net flow the link score from the flow to the stock will also be one. Similar logic applies to an exponential drain, though in this case the link score from the flow to the stock is -1 so the loop score becomes -1.

This value of 1 is true regardless of the population growth rate, or residence time, in the above example. This is an important distinction between the loop score and gain around a loop. It is also interesting to think about what happens to the single loop score as additional loops are added. For example, adding deaths to a population model with only births would give link scores from the flows into the stock based on their relative value. In this case both of the loop scores will have magnitude greater than 1, and the closer the flows are the bigger the scores. This is, as discussed, why the relative loop scores are reported as the basis for analysis and emphasizes how distinct the loop score is from a representation of gain. Put another way, loop scores do not predict the speed of change, but only show which part of structure is dominant at any point in time.

References

- Bass, FM. (1969). A new product growth for model consumer durables. *Management science* **15**(5): 215-227.
- Davidsen P. 1991. The Structure-Behavior Graph. The System Dynamics Group, MIT, Cambridge, MA.
- Duggan J. and Oliva R. 2013. Methods for identifying structural dominance. *System Dynamics Review* **29**(Virtual Special Issue). Available at [http://onlinelibrary.wiley.com/journal/10.1002/\(ISSN\)1099-1727/homepage/VirtualIssuesPage.html](http://onlinelibrary.wiley.com/journal/10.1002/(ISSN)1099-1727/homepage/VirtualIssuesPage.html).
- Eberlein RL. 1984. Simplifying dynamic models by retaining selected behavior modes. PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Ford D. 1999. A behavioral approach to feedback loop dominance analysis. *System Dynamics Review* **15**(1): 3–36.
- Forrester N. 1982. A Dynamic Synthesis of Basic Macroeconomic Theory: Implications for Stabilization Policy Analysis. PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Graham AK. 1977. Principles of the Relationship Between Structure and Behavior of Dynamic Systems. PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Gonçalves P. 2009. Behavior modes, pathways and overall trajectories: eigenvector and eigenvalue analysis of dynamic systems. *System Dynamics Review* **25**(1): 35–62.
- Güneralp B. 2005. Progress in eigenvalue elasticity analysis as a coherent loop dominance analysis tool. In Proceedings of the 2005 International System Dynamics Conference, Boston, MA. System Dynamics Society: Albany, NY.
- Güneralp, B. 2006. Towards coherent loop dominance analysis: progress in eigenvalue elasticity analysis. *System Dynamics Review*, **22**(3), 263-289.

- Hayward J, and Boswell GP. 2014. Model behaviour and the concept of loop impact: a practical method. *System Dynamics Review* **30**(1): 29–57.
- Hayward, J, and Roach PA. 2017. Newton's laws as an interpretive framework in system dynamics. *System Dynamics Review* **33**(3-4): 183-218.
- Huang, J., Howley, E., and Duggan, J. (2012). Observations on the shortest independent loop set algorithm. *System Dynamics Review*, **28**(3), 276-280.
- Kampmann CE. 2012. Feedback loop gains and system behaviour (1996). *System Dynamics Review* **28**(4): 370–395.
- Kampmann CE, and Oliva R. 2006: Loop eigenvalue elasticity analysis: Three case studies. *System Dynamics Review*. **22**(2): 141-162.
- Kampmann, CE., and Oliva, R. (2008). Structural dominance analysis and theory building in system dynamics. *Systems Research and Behavioral Science: The Official Journal of the International Federation for Systems Research*, **25**(4), 505-519.
- Kampmann CE., and Oliva R. 2009. Analytical methods for structural dominance analysis
in system dynamics. In Meyers RA (ed.) *Encyclopedia of Complexity and Systems Science* Springer, New York NY, 8948–8967.
- Kampmann, C., and Oliva, R. (2017). Analytical methods for structural dominance analysis in system dynamics. *Encyclopedia of Complexity and Systems Science* (2017).
- Mass, NJ, and Senge, PM. (1975). Understanding oscillations in simple systems. *System Dynamics Group Working Paper D-2045-2, MIT*.
- Mojtahedzadeh MT. 1996. Structural Analysis of the URBAN1 model. University at Albany, SUNY, NY 12222, USA.

- Mojtahedzadeh M, Andersen D, and Richardson GP. 2004. Using Digest to implement the pathway participation method for detecting influential system structure. *System Dynamics Review* **20**(1): 1-20.
- Mojtahedzadeh M. 2008. Do parallel lines meet? How can pathway participation metrics and eigenvalue analysis produce similar results?. *System Dynamics Review*. **24**(4): 451-478.
- Moxnes, E, and Davidsen PI. 2016. Intuitive understanding of steady-state and transient behaviors. *System Dynamics Review* **32**(2): 128-153.
- Naumov S, and Oliva R. 2018. Structural dominance analysis toolset. Available: <http://iops.tamu.edu/faculty/roliva/research/sd/sda/> [05 Feb 2020].
- Oliva, R. (2004). Model structure analysis through graph theory: partition heuristics and feedback structure decomposition. *System Dynamics Review*, **20**(4): 313-336.
- Oliva R. 2016. Structural dominance analysis of large and stochastic models. *System Dynamics Review* **32**(1): 26-51.
- Oliva, R. (2020). On structural dominance analysis. *System Dynamics Review* **36**(1): 1-21.
- Phaff HWG., Slinger JH, Güneralp B., and van Daalen CE., 2006. Investigating model behavioural analysis: A critical examination of two methods. In *Proceedings of the 24th International Conference of the System Dynamics Society*. Nijmegen, The Netherlands, System Dynamics Society.
- Powers, R. (2019). SD.js in-browser system dynamics model simulation and display. GitHub repository: <https://github.com/bpowers/sd.js>
- Richardson GP. 1995. Loop polarity, loop dominance, and the concept of dominant polarity. *System Dynamics Review* **11**(1): 67–88.
- Richardson GP. 1996. Problems for the future of system dynamics. *System Dynamics Review* **12**(2): 141-157.

Saleh MM. 2002. The Characterization of Model Behavior and its Causal Foundation. PhD Thesis, University of Bergen, Bergen, Norway.

Saleh M., Oliva R., Kampmann CE, Davidsen PI. 2010. A comprehensive analytical approach for policy analysis of system dynamics models. *European Journal of Operational Research* **203**(3): 673-683.

Sato, JB. (2016). State space analysis of dominant structures in dynamic social systems. PhD Thesis, University of Washington, St. Louis, MO.

Sterman JD. 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Irwin/McGraw-Hill, Boston.

7. Article #2: LoopX: Visualizing and understanding the origins of dynamic model behavior

By: William Schoenberg

Abstract:

It is a fundamental precept of System Dynamics that structure leads to behavior. Relating the two is a roadblock for using feedback models because substantial experimentation or the application of specialized analytic techniques that are difficult to employ is required. LoopX builds understanding of structure as it determines behavior by rendering and highlighting structure responsible for behavior as it unfolds. LoopX builds on the Loops that Matter (Schoenberg et. al, 2019) approach to analyzing loop dominance by presenting its results in an easy to use, interactive, software tool. This is a significant step forward in the challenges of automatically visualizing model behavior and linking it to generative structures identified in Sterman (2000). LoopX machine generates high quality CLDs from model equations at different levels of detail based on the dynamic importance of links and variables, in addition to animating them based on their importance to the origins of model behavior.

Introduction

This paper presents a new and highly usable solution to three important challenges identified in the final chapter of *Business Dynamics* (Sterman 2000): It addresses “Automated identification of dominant loops and feedback structure”, calculating and displaying the evolution of loop dominance as a model simulates; It improves on “Visualization of model behavior”, using animation to coordinate the display of structural dominance evolution with the behavior over time of model variables; It addresses “Linking behavior to generative structure”, using animation of automatically aggregated diagrams that connect the loop dominance analysis with the

model structure via connectors and flows that change size and colors over the course of a model simulation.

Simple systems are usually easy to analyze with intuition and trial-and-error, but with larger systems that are characterized by high feedback loop complexity, the risk of incorrect explanation rises (Oliva, 2016). It is this threat of failure which makes these three challenges posed by Sterman (2000) so relevant. Currently the domain of objective feedback loop dominance analysis is limited to a relatively select few practitioners with a high degree of expertise and training. The lack of tools for parsing and developing insight in large causal models often acts as the limit on the utility of large models to general audiences (Schoenenberger et. al, 2017). The incidence of these problems with presenting models with the intent to develop understanding is not a new occurrence, a cursory literature returns a 1976 paper (republished in 1986) which refers to problems in methods for simplifying the presentation of model structure via causal loop diagrams developed even earlier than that (Richardson, 1986). Clearly, any solution to Sterman's three challenges must help to reduce the barriers to entry for model understanding and analysis, expanding our depth of understanding of the models which are at the heart of our field via improved communication of complexity and its origins.

The foundation of this work is the Loops that Matter (LTM) technique for determining loop dominance (Schoenberg et. al, 2020). Building on the LTM method, the solution to these three challenges employs the use of Causal Loop Diagrams (CLDs) as well as Stock and Flow Diagrams (SFDs) as a vehicle for representing system structure to model consumers.

This paper presents LoopX which is a tool that is capable of reading in and analyzing an XMILE model. The tool allows the model to be simulated, and also analyzed by LTM generating a full complement of link and loop scores describing the origins of model behavior from a loop dominance perspective. LoopX is capable of rendering the model as a stock and flow diagram based on the layout decisions made by the model author. LoopX also machine generates high quality CLDs from the network of

model equations at user specified levels of complexity presenting a minimum number of variables and links which are deemed necessary (by the LTM analysis) to understand the dynamics of the shifting loop dominance at the requested cognitive complexity level. All diagrams, machine or human generated, are animated portraying dominance information via flows and connectors which change colors and size in real time as the model simulates. All loops are identifiable directly within the context of all of the aforementioned diagrams.

Problem Statements

LoopX required the development of solutions to the following three main problems:

1. How can high quality CLDs be machine generated from the network of model interconnections?
2. How can models be aggregated and simplified without losing information important to model understanding while retaining relative simplicity?
3. How can the results of an LTM loop dominance analysis be easily visualized and communicated?

Literature Review

This review combines literature from the graph theory and system dynamics fields to provide the reader with the requisite knowledge for understanding the current state of the art as it applies to each of the three problem statements. This helps to place the development of LoopX into context among the existing technologies.

Techniques for machine generation of network graphs, the basis on which CLDs are formed

Most important to the automated generation of high quality CLDs is the force directed layout algorithm. A force directed layout algorithm solves the problem of the placement of nodes in 2D space, such that symmetry is generated, and edge length is approximately equal, by running a physics simulation of weights connected

by springs and minimizing the total energy of the system. The first force directed layout technique used steel rings to represent each node and then connected those rings using logarithmic springs (Eades, 1984). In this version of the algorithm, attractive forces were only calculated between neighbors, and repulsive forces were calculated between all node pairs (Eades, 1984). This process ensured that neighbors were always close by but limited the scope of the N-squared problem.

The next evolution in the force directed algorithm was to introduce the concept of an ideal distance between every node pair based on the shortest path between each node pair, and to use Hooke's Law, meaning real world realistic linear springs (Kamada and Kawai, 1989). The Kamada Kawaii approach solved partial differential equations based on Hooke's Law to optimize layout applying all forces between all node pairs in an iterative fashion (Kamada and Kawai, 1989). A gradient descent optimization process used to terminate the simulation when a global minimum in the energy state of all the springs was found (Kamada and Kawai, 1989).

Development of Graphviz, an open source toolkit for solving these graph generation problems took place in parallel to these developments at Bell Labs. Graphviz contains many different automated layout mechanisms, but the mechanism most relevant to CLD generation is called neato, which is based on the work of Eades, Kamada and Kawai among others. The layout algorithm used in neato that we are interested in, is derived from the Kamada Kawai algorithm. It assumes there is a linear spring between every pair of nodes, each with an ideal length (Gansner, 2014). The ideal distance between each node pair is the result of a function computed for each pair; the ideal length function we are interested in uses the shortest path between the two nodes to determine the ideal distance between these nodes, although many other choices are offered. Neato is able to turn a static text file with a description of the graph as lists of nodes and edges into a 2D diagram quickly (North, 2004).

Neato performs the following series of high-level steps in its operation. First neato parses its input file which specifies the list of nodes and all edges which connect them, as well as any arguments which are used to control the layout process. Second

it constructs and simulates the physics simulation to lay out the nodes in 2D space. Neato allows the user to specify the starting location of each node, and since the layout process relies on a gradient descent optimization this allows the user to potentially identify starting locations for the nodes which produce higher quality diagrams than others. At this point neato executes any node overlap removal operations. Finally, once all nodes are laid out, neato draws the edges connecting the nodes according the options specified in the input file.

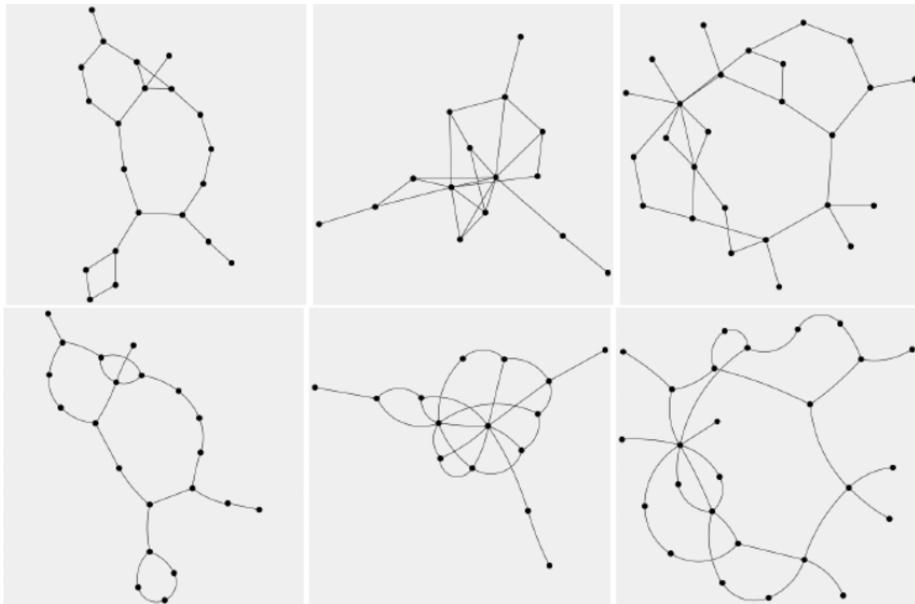


Figure 1: Example of standard straight-line and Lombardi-style force directed graphs copied from Chernobelskiy et al 2011.

Neato, like all force directed graph algorithms, produces, by default, edges that are straight. There are disadvantages though for using straight edges especially as it relates to user understanding of the generated graph diagrams (Xu et al. 2012). Graphs of the type produced by force directed algorithms with curved edges are generally called near-Lombardi or Lombardi-style diagrams, and CLDs are a form of Lombardi-style diagram. Figure 1, presents examples of straight edge force directed graphs and their Lombardi style complements. When using a curved edge, Lombardi-style diagram, there are significant user performance improvements on graph related tasks such as determining the shortest path, node degree calculations and common

neighbor determinations (Xu et. al., 2012). The problem with algorithmic Lombardi-style diagram generation and force directed graphs in general, is the lack of the concept of directed edges and especially directed cycles (links and loops as SD conceives of them). Lombardi-style diagrams tend to produce circular shapes that are not loops. To meet the needs of the SD community, and to be able to apply force directed graph algorithms like neato to generating CLDs, an algorithm for determining how to curve each edge in a way which emphasizes the loops needed to be developed.

Techniques for model aggregation and simplification

As covered in the introduction, the problems of simplifying dynamic complexity for wider consumption has been studied since the formative years of the field. Early research discusses how CLDs alone do not give an accurate enough picture of model structure so that behavior modes can be predicted and understood (Richardson, 1986). Richardson (1986) argues for caution when using CLDs to aggregate and simplify model diagrams, and that information is often lost in that process.

The most famous examples of feedback simplification techniques are the independent loop set and its refinement, the shortest independent loop set (ILS and SILS) respectively (Kampman, 2012) and (Oliva, 2004). These graph theory techniques for the partitioning of the cycles (feedback loops), implicit in the network of model structure, arose due to the complexity faced when performing and analyzing the results of the eigenvalue elasticity method of loop dominance analysis (EEA) or when trying to find high leverage points for policy intervention. In anything but small models both authors were faced with a relatively large list (compared to the number of variables in the model) of feedback loops which were all tightly interrelated. The SILS concept pairs down the number of feedback loops to the set of geodetic (shortest) loops which are necessary to fully describe the feedback loop complexity of the model. This reduces the number of loops present in a fully accurate CLD of the entire network of model structure, focusing user attention on the loops which are most easily influenced by policy.

Built on the ILS and SILS, (Schoenenberger et. al, 2015 and 2017) present the use of variety filters derived from interpretative model partitioning, structural model partitioning and the ADAS method (algorithmic detection of archetypal structures), to communicate intuition from large models. Their audience are those who would normally be overwhelmed by the size and complexity of the models being studied. This work also builds upon earlier studies of model simplification done by Eberlein (1989), which uses linearization, and on Sysel and Barlas' (2006) aggregation method. The variety filters technique presents the user with structural clusters of model variables based on state of the art statistical and graph theory techniques as a way of visualizing and understanding nearness and hierarchy. With interpretative clustering, model complexity is filtered via studies of the relationships between pairs of model sectors. Using ADAS which is applied to the above generated clusters, users select a stock of interest as well as an archetypal structure to find, and the algorithm returns the feedback loops which contain the variable in the system archetype specified. This significantly reduces the number of feedback loops to be studied by the end user pairing down the complexity of the model.

The Forio Model Explorer feature of Forio Simulate is an example of a simplistic Kamada Kawai style force directed rendering of model structure which was later evolved into supporting a secondary hierarchical layout engine with rudimentary aggregation steps taken to either only show two degrees of distance from a variable of interest or all of the links between two variables of interest with a filter based on path length. The Forio Model Explorer was studied and was compared to traditional hand drawn CLDs in an attempt to measure the effectiveness of the automated diagramming and aggregation techniques (Schoenberg, 2009). All tests were inconclusive, showing no reported differences in learning outcomes, but diagrams generated were of significantly less quality, lacked any of the positive attributes of Lombardi-style diagrams and were not focused on feedback loop behavior. This work appears to be the most recent previous attempt at using aggregation and force directed graphs to solve the challenges laid out by Sterman (2000).

The Loops That Matter Method

The LTM method (Schoenberg, et. al, 2020) performs a formal assessment of dominant structure and behavior as categorized by Duggan and Oliva (2013). The LTM method is built around the observation of how modelers perform the art of model analysis to understand the origins of behavior. LTM interacts directly with the full network of model equations, walking the causal pathways between all variables in the model, calculating in time with the simulation, metrics that measure the contribution (ex: force, strength) and polarity of each link in the network of model equations. The LTM approach produces metrics which interpret the origins of behavior for the entire model⁵ rather than just the behavior of a single state variable.

The first metric introduced by the LTM method is the link score. The link score is a measure of the contribution and polarity of any link in a model from an independent to dependent variable regardless of whether or not the link contains an integration process. The link score concept tracks the concept of the link gain, and when multiplied through pathways up until but not including the stock, is the same as Richardson's (1995) concept of the dominant polarity. The link score is capable of being calculated for every link in the mode, including those which contain an integration process. The link score is computed once per each time interval in the model and is computed for each link in the model. There are two methods for calculating the link score depending upon if the link contains an integration process or not. Schoenberg et. al. (2020) demonstrates that the two methods produce exactly the same measure and therefore can both be referred to as the link score.

Equation 1 is the definition of the link score of a link that does not contain integration assuming there are two inputs (x and y) to the dependent variable z characterized by the equation $z = f(x, y)$. The link score for the link $x \rightarrow z$ when written in a

⁵ For cases where each stock in the model is able to either directly or indirectly impact each other stock in the model. For models where this assumption does not hold true LTM informs on the origins of behavior in each giant connected component of the model where this assumption holds true.

discontinuous form based upon the implementation of the calculation is (See Schoenberg et. al, 2020 for continuous analytical form):

Equation 1: The discontinuous form for the link score equation which matches how the implementation of the calculation works moving in time with the dt of the model.

$$LS(x \rightarrow z) = \begin{cases} \left(\left| \frac{\Delta_x z}{\Delta z} \right| \cdot sign\left(\frac{\Delta_x z}{\Delta x}\right) \right), \\ 0, \quad \Delta z = 0 \text{ or } \Delta x = 0 \end{cases}$$

In Equation 1 Δz is the change in z from the previous time to the current time. Δx is the change in x over that same time step. $\Delta_x z$ is the change in z with respect to x . From a computational perspective $\Delta_x z$ which is called the partial change in z with respect to x , is the amount z would have changed, conditionally, if x had changed the amount it did, but y had not changed. The first major term in Equation 1 represents the magnitude of the link score, the second is the link score polarity.

Equation 2: Link score for all links from derivatives (flows) to state variables (stocks) (both inflows and outflows are covered). The simple one inflow and one outflow case is presented and is easily generalized.

$$\text{Inflow: } LS(i \rightarrow s) = \left(\left| \frac{i}{i - o} \right| * 1 \right) \quad \text{Outflow: } LS(o \rightarrow s) = \left(\left| \frac{o}{i - o} \right| * -1 \right)$$

For links which contain an integration process the equation for determining the contribution and polarity of the link from an inflow (i) or outflow (o) to a state variable (s) is shown in Equation 2. This allows the LTM method to measure the link score which is the contribution and polarity of each individual flow to the value of the stock.

The second key metric produced by the LTM method is the loop score as shown in Equation 3. The loop score tracks the concept of the loop gain and is the result of the multiplication of all link scores for all links in a loop. This is a demonstrably unique measure which bears some rough similarity to the Loop Impact metric of Hayward and Boswell (2014) but is unique because it is capable of including the links which contain integration processes allowing a single value to be assigned as the contribution of a loop.

Equation 3: Definition of loop score, for the loop x which contains n links for each source variable S to the target variables T .

$$\text{Loop Score}(L_x) = (LS(s_1 \rightarrow t_1) \cdot LS(s_2 \rightarrow t_2) \dots \cdot LS(s_n \rightarrow t_n))$$

The loop score is a dimensionless value which samples the effort a loop is expending to change the behavior of the stocks it connects at each calculation interval of the model. As the link score can be thought of as the force of an independent variable pushing on the result of a dependent variable, the loop score can be thought of as the force of one feedback loop pushing on the behavior of all the stocks (and therefore all variables) it connects.

The third and final key metric produced by the LTM method is the relative loop score (Equation 4) which compares the contribution of feedback loops to determine which are dominant at any point in time. The relative loop score requires no independence across the loops it compares and ideally uses the exhaustive set of feedback loops as the basis for comparison.

Equation 4: Definition of the relative loop score for the loop x normalized over all loops n analyzed in the chosen loop set.

$$\text{Loop Score}_{L_x} = \left(\frac{\text{Loop Score}(L_x)}{\sum_{y=0}^n |\text{Loop Score}(L_y)|} \right)$$

The sign of a relative loop score represents the polarity of the feedback loop. The relative loop score is a normalized measure taking on a value between -1 and 1. It reports the polarity and instantaneous fractional contribution of a feedback loop to the change in value of all stocks in the feedback loop set it is a member of. By comparing loop scores, it can easily be determined which loops are dominant, i.e. contribute the most (over 50%) to the behavior of all stocks in the feedback loop set under study. This normalization is critical to maintaining scores that are easy to work with.

Overview of the LoopX software

Figure 1 demonstrates a high-level organization of the subprocesses inside of the LoopX software. The LoopX is a web-based software tool capable of reading in an

XMILE model file which must contain a stock and flow diagram. The LoopX tool is capable of simulating a limited set of XMILE models and performing a loop dominance analysis on those models using LTM. The LoopX tool is also capable of rendering models as either animated SFDs or CLDs based upon the loop dominance analysis.

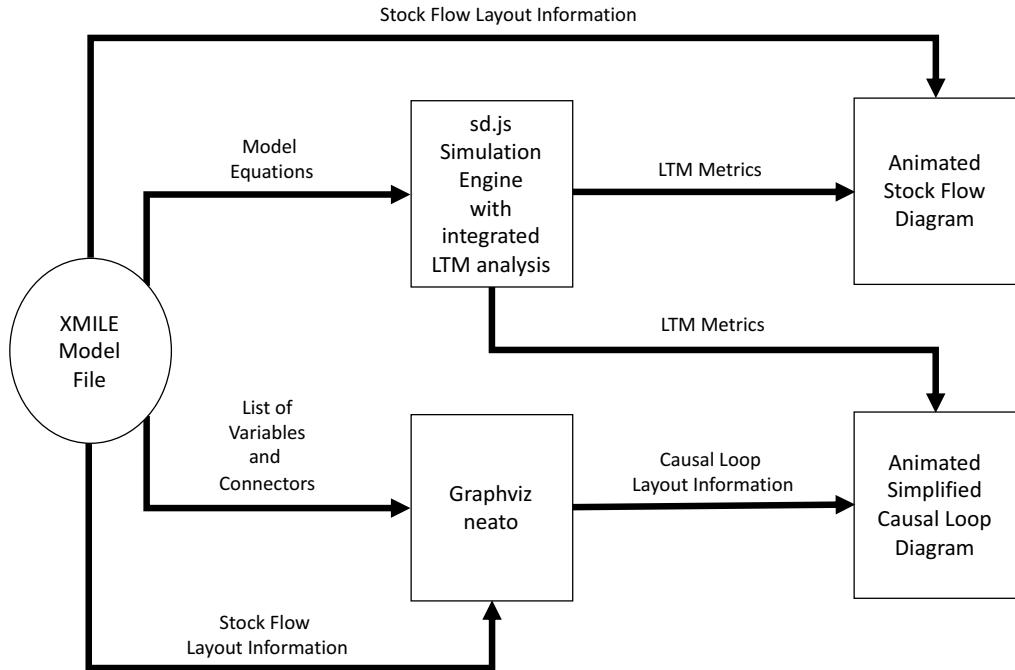


Figure 1: Schematic showing the relationship of the various processes in the LoopX software. Each arrow is labeled with the information that passed between processes (the squares).

LoopX starts with a user provided XMILE model file. Once the file is parsed the model equations are used by the sd.js simulation engine constructed by Powers (2019) which has been modified by Schoenberg et. al., (2020) to simulate the model while performing the LTM analysis. Each link in the model has its link score calculated for each dt. Each loop in the model has its loop score and relative loop score calculated for each dt in the model. These are the LTM metrics referred to in Figure 1. The model file is also used to generate a list of variables (nodes) and connectors (edges) and that information along with the initial position of the variables

in the stock flow diagram is passed to neato which uses them to perform the automated CLD generation. Neato runs a Kamada Kawai force directed graph algorithm to produce the static CLD that is then animated by LoopX using the LTM metrics. The animated stock and flow diagram combines the static stock and flow diagram information from the XMILE file and the LTM metrics from the simulation engine.

Solutions to the three major challenges

1. *How can high quality CLDs be machine generated from the network of model interconnections?*

Neato is used to generate the static causal loop diagrams which are then later animated by LoopX to reveal the origins of model behavior based upon an LTM analysis. As identified in the literature review, the problem with using neato and force directed graph algorithms in general, is that they produce diagrams with all straight edges, or they produce diagrams with curved edges that do not emphasize the feedback loops. To develop machine generated CLDs which emphasize the loops, the development of LoopX required a solution to this problem. Figure 2 (as well as all of the other CLDs shown in this paper) demonstrate the solution to that problem.

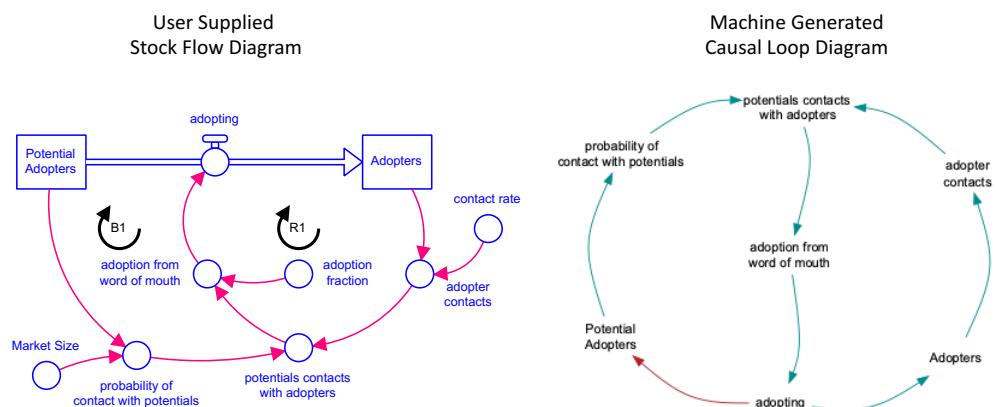


Figure 2: Example machine generated CLD from user supplied SFD of the Bass diffusion model (1969).

To make neato capable of drawing edges in a way that emphasizes the feedback loops, the edge drawing step of neato had to be modified. The solution to the edge curving problem is a simple algorithm whose implementation has been accepted into the publicly available version of neato and is invoked whenever the user specifies that they want their edges to be curved. The edge curving algorithm follows a simple heuristic derived from the observation of CLD diagram drawing by hand. The heuristic specifies that on an edge by edge basis, the center of the circle which forms the arc that the edge will follow, must be the average center of the nodes which form the shortest feedback loop with length greater than 2 that the edge is a member of. The two-node exception is handled separately within the neato codebase, and produces paired directed edges that do not over-emphasize the cycle, - producing elongated ellipse structures that cover an area relative to the number of nodes. This edge curving heuristic relies upon the attributes of force directed graphs which place nodes that are related closest together. This heuristic produces loops that look circular except for in degenerate cases where the force directed layout fails to produce good local clusters and the shortest feedback loops are relatively far flung in the 2D space.

To minimize the incidence of degenerate diagrams due to path dependency issues in neato use as the initial position of each node, the position of the variable it represents in the stock and flow diagram. This is a generally useful way to make sure that local clusters are being preserved which is important for keeping short loops near each other in the machine generated diagram. This is based on the assumption of a good quality stock and flow diagram which has also kept the variables most closely related to each other near in physical space.

Finally, to raise the probability of generating high quality CLDs, - LoopX uses the following neato specific settings. First, LoopX instructs neato to use the Prism algorithm (Gansner & Hu, 2010) (a proximity graph-based algorithm) to prevent the overlap of variable names in the CLD by setting the ‘overlap’ attribute of the graph to ‘prism’. This instructs neato to remove overlapping variable names in a way which minimizes the disturbance to the layout created by the physics simulation. Second,

LoopX instructs neato to use the KK mode during the diagram layout step so that neato uses a variant of the gradient descent process originally proposed by Kamada and Kawai (1989), for solving the optimization problem during the node placement step. This is done by setting the graph attribute ‘mode’ to ‘KK’. Third, LoopX instructs neato to set the ideal edge length based on the ‘shortpath’ model (default option) for computing its distance matrix, i.e. using the shortest path between two nodes as the ideal length of the spring between each node pair. This is done by setting the graph attribute ‘model’ to ‘shortpath’. Finally, LoopX invokes the edge curving algorithm for emphasizing feedback loops by setting the graph option ‘splines’ to ‘curved’, - otherwise a straight-edged diagram would be produced.

2. How can models be aggregated and simplified without losing information important to model understanding while retaining relative simplicity?

Based on the LTM loop dominance analysis, LoopX introduces two new parameters which are used to specify how to simplify causal loop diagrams in a way which maximizes the explanatory nature of the diagrams while removing variables and feedback complexity. The parameters are used to filter the full feedback complexity of the model, reducing the number of variables and therefore the links between variables in a CLD in such a way as to minimize the loss of descriptive power as measured by the relative loop score.

The first parameter is the ‘link inclusion threshold’ which filters variables from the simplified CLD by measuring the maximal variance in the contribution of links. To do that, a method must be derived to measure the variance in link contribution. The ‘relative link variance’ calculation is demonstrated in Equation 5 and measures the variance in the contribution of a link across the entire time period of the simulation. The relative link variance measures the change in the percentage contribution of an independent variable x , to a dependent variable y , across the entirety of a simulation run. The basis for the calculation is the relative link score which takes the link score metric from the LTM analysis and calculates a normalized value across all independent variables for a dependent variable. The relative link score describes the

contribution and polarity of a link as a percentage of the total contribution across all incoming links of a dependent variable.

Equation 5: Relative Link Variance, calculated for the link $x \rightarrow y$ which measures the variance in the magnitude of the relative link score over the full simulation time. The relative link score is the link score normalized across all determinants of a target variable. In this case the relative link score is normalized over all determinants of y .

$$\begin{aligned} & \text{Relative Link Variance}(x \rightarrow y) \\ &= \max(\text{abs}(\text{Relative Link Score}(x \rightarrow y))) \\ &\quad - \min(\text{abs}(\text{Relative Link Score}(x \rightarrow y))) \end{aligned}$$

The link inclusion threshold is used mainly to filter the number of auxiliary variables that appear in the simplified CLD rendered by LoopX. The link inclusion threshold has a range of [0-1]. Only variables which are pointed to by at least one link with a relative link variance greater than or equal to this parameter, are included in the simplified CLD. If a stock is included, LoopX automatically includes the flows to make clear to the end user that the stocks require flows to change. The link inclusion threshold allows the user to specify from a loop dominance perspective which (mainly auxiliary) variables to remove from the simplified CLD. The reason that typically only auxiliary variables are filtered by the link inclusion threshold is because flow-to-stock links typically have high relative link variance because there are large changes in link score as the model approaches and leaves equilibrium states. Links with a high relative link variance are those which change their contribution the most over the course of the simulation run. Those links, therefore, tend to point to the sources of non-linearity in models and are, typically, variables that are important for understanding model behavior. Links which do not change at all over the course of the simulation run, regardless of the specific level of contribution to their dependent variable, have a relative link variance of 0, and point to variables that are likely unimportant and are good candidates for elimination. These variables tend to exist for the modeler to simplify equations.

The second parameter used to simplify CLDs is the ‘loop inclusion threshold’ which is mainly used to filter the number of stocks and associated flow variables that appear

in the simplified CLD. The loop inclusion threshold has a range of [0-1], representing the average magnitude of the relative loop score from LTM. Only loops that have an average magnitude of the relative loop score greater than or equal to this parameter have their stocks and flows automatically included in the simplified CLD regardless of the link inclusion threshold filtering. This allows the user to specify from a loop dominance perspective which stock and flows (as well as their associated direct feedback loops) do not need to be in the rendered graph, - facilitating the creation of a smaller simplified CLD. The average magnitude of the relative loop score is measured over the entire time period of the simulation run, starting in the first instant where the loop becomes active. This delayed averaging avoids penalizing loops during the initialization phase of model behavior. The average magnitude of the relative loop score describes the percentage of the change in behavior of the stocks in the model which the loop is responsible for. Loops with a low average magnitude of the relative loop score, for instance, 0.01 would only describe 1% of the total behavior of the model over the simulated time horizon and probably do not need to be included in a simplified diagram in order to understand the origins of the generated model behavior from a structural perspective.

The link and loop inclusion thresholds are applied to all links and loops in the model and produce a filtered list of variables to keep directly from the network of model equations. From that, a corresponding list of links needs to be generated, - one that matches the reality of the true feedback loop connections in the fully disaggregated model but does not show all of the individual steps along the way. In other words, the filtering process solves the aggregation problem from a variable perspective and now the links need to be generated such that they make sense at both this new level of aggregation and yet still represent the connections of a totally disaggregated model.

The process used to generate the links performs a depth first search for each possible link in the new limited set of variables generated by the filtering of the link and loop inclusion thresholds. For each candidate simplified link the search traverses the full model, testing if it can walk the network of full model equations from the source of the candidate link to the target without passing through a variable already in the

limited set specified by the parameter filters (or a variable already visited in the search). If it can find a pathway in the full disaggregated model which properly represents the candidate link, the candidate link is kept. All kept simplified links are both valid in the fully disaggregated model, because the search was able to find a causal chain which that link represents, and each simplified link only represents a single step in the aggregated model.

Next the application of this technique is studied in the context of Forrester's 1968 Market Growth model picture in Figure 3. The Market Growth model with all macros expanded contains 23 feedback loops, 48 variables with 10 stocks. The full CLD containing the full feedback complexity and network of equations involved in feedback loops is presented in Figure 4 and the first chosen level of simplification in Figure 5.

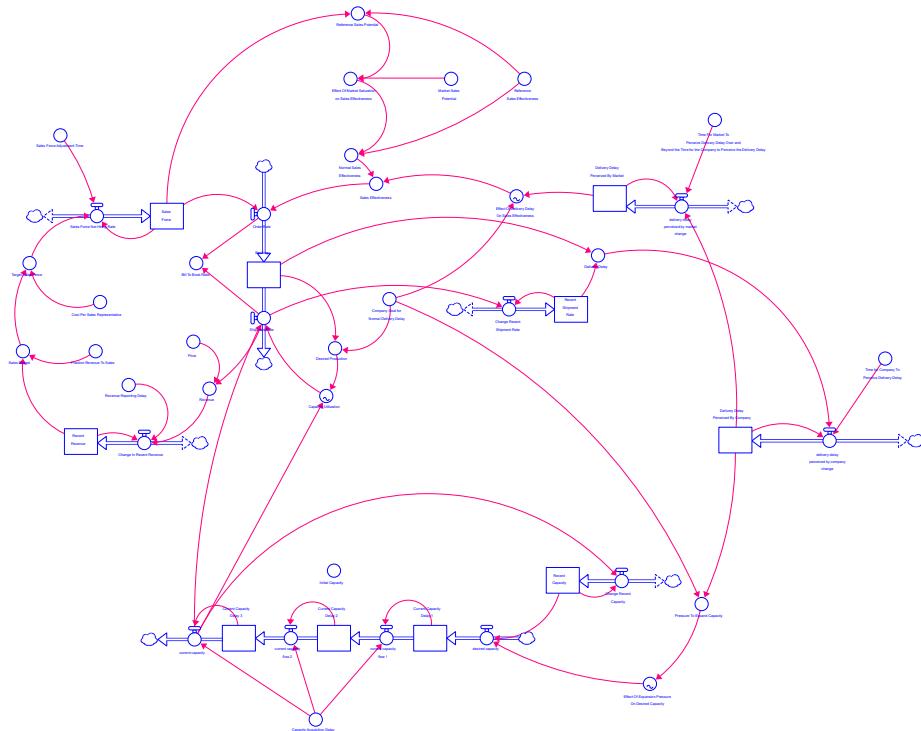


Figure 3: Stock and flow diagram of Forrester's 1968 market growth model. Full model is included in the online supplemental materials

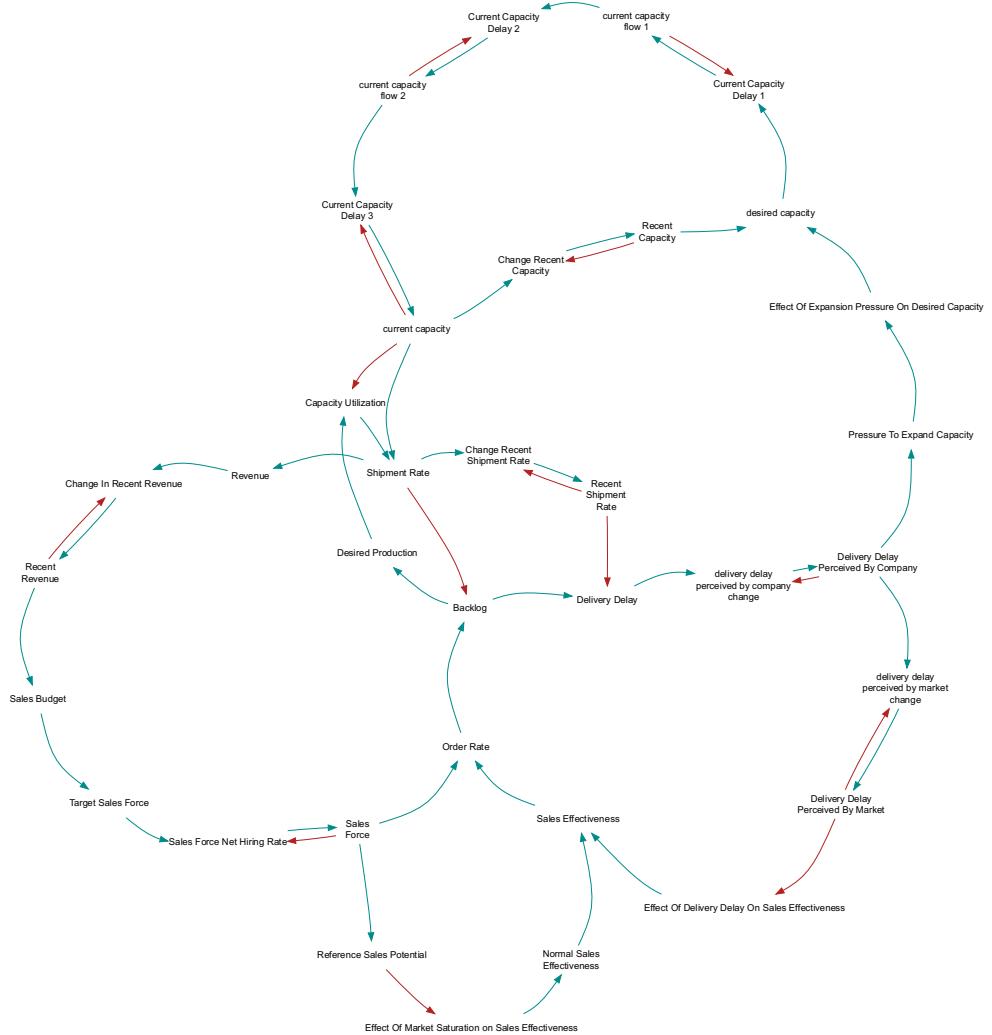


Figure 4: Autogenerated full CLD of Forrester's 1968 market growth model (link inclusion threshold 0%, loop inclusion threshold 0%). Red links are negative, green links are positive.

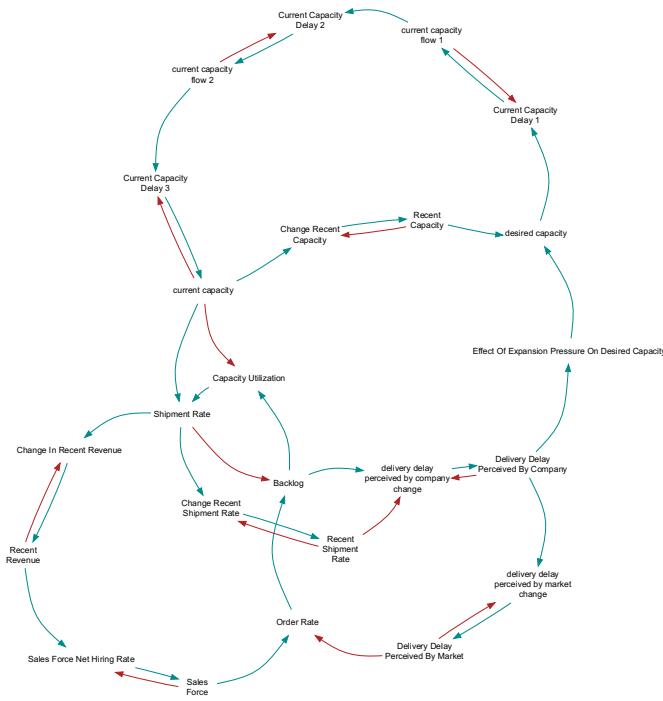


Figure 5: Autogenerated Simplified CLD of Forrester's 1968 market growth model (link inclusion threshold 100%, loop inclusion threshold 0%). Red links are negative, green links are positive.

The full CLD in Figure 4 is large and difficult to understand for those who are not familiar with Forrester's model. Figure 5 is much simpler and in all ways is superior to the full CLD. It contains less than half of the variables of Figure 4, but still portrays the full feedback complexity of the model. Figure 5 was generated using a link inclusion threshold of 100%. All stocks and flows in the model were kept in Figure 6 because the loop inclusion threshold was set to 0%. This diagram is still complex because it represents all 23 feedback loops and contains 10 stocks, 12 flows and 2 auxiliaries. To further simplify the diagram, stocks and flows need to be removed, which will reduce the feedback complexity of the simplified CLD.

Setting the loop inclusion threshold to 20% while keeping the link inclusion threshold at 100% generates the diagram seen in Figure 6 which contains 7 stocks, 10 flows, and 2 auxiliaries. The maximally simplified CLD is pictured in Figure 7 where the both the link and loop threshold are set to 100%. In its most simplified form, the

simplified CLD contains 4 variables, 2 flows 2 auxiliaries. The major tradeoff across Figure 4 and Figure 7 is the loss of descriptive power vs. the ease of cognition. This decision is best made on an individual by individual basis based on specific goals.

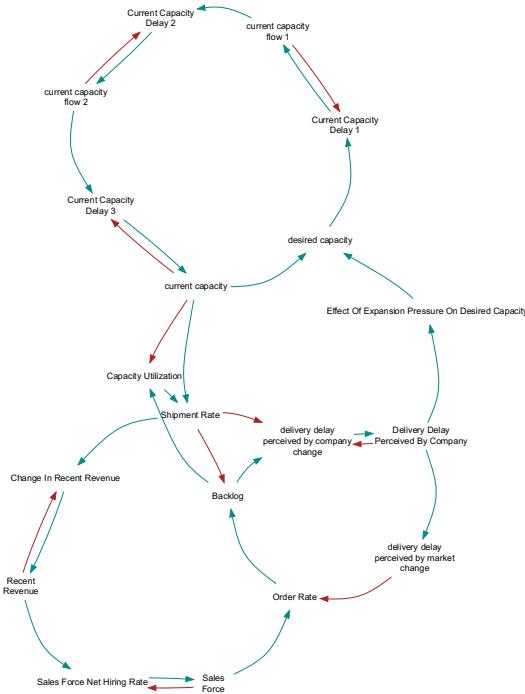


Figure 6: Autogenerated Simplified CLD of Forrester's 1968 market growth model (link inclusion threshold 100%, loop inclusion threshold 20%). Red links are negative, green links are positive.

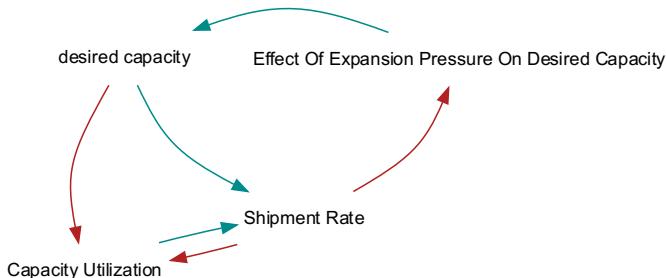


Figure 7: Autogenerated Simplified CLD of Forrester's 1968 market growth model (link inclusion threshold 100%, loop inclusion threshold 100%). Red links are negative, green links are positive.

As just demonstrated, when the simplification parameters are increased, fewer variables and therefore fewer feedback loops are presented to the end user. The link inclusion threshold typically removes complexity in the form of excess auxiliaries, and the loop inclusion threshold removes stocks, flows and feedback loops. These parameters provide the end user with tools to reshape the feedback loop complexity of the model on demand. Coupled with the machine generation of CLDs end users can now find CLDs which match their cognitive abilities and can be sure that the presented CLDs capture the most relevant portions of the feedback complexity of the model for that desired level of complexity.

3. How can the results of an LTM analysis be easily visualized and communicated?

By incorporating the results on an LTM analysis with an SFD or simplified CLD users can get information about the importance of feedback loops and links directly in the context of structure in an easy to interpret and understand way. By making SFDs and simplified CLDs animate, it becomes possible to visualize the feedback loop dominance profile of a model as the behavior unfolds in a way which is more directly related to the structure of the model as opposed to a comparative line graph.

In the animated SFDs or CLDs produced by LoopX, regardless of the type of diagram, color is used to represent the polarity of any link and thickness is used to represent the magnitude of the relative link score. In simplified CLDs where the links are built from causal pathways the relative link scores are multiplied to generate a relative link score for the simplified link. Users can change the diagrammatic representation of the model at any point in time via a simple dropdown box to allow for the most intuitive diagram to be presented. Users are able to obtain plots of behavior overtime for any variable in any diagram, as well as relative link scores and relative loop scores for any element by just clicking on the variable or connector or loop identifier in the diagram. All generated data is also offered for download in CSV form for external analysis and plotting.

On all diagrams, a table of relative loop scores is plotted showing the instantaneous contribution of each loop as well as by the relative average magnitude of the contribution over the entire model run of each loop to the behavior of all stocks in the model. This table is sorted by the relative average magnitude of the contribution to make the most important loops rise to the top. The loop identifier for any loop may be pressed, and, while held, highlights all variables and links in any of the diagrams that the loop represents, - removing all dominance information while doing so. This allows users to quickly identify what the meaning of all of the identified loops are and track them through any rendering of model structure as their placement does change since each diagram generation is a totally independent process as of the current writing of this paper.

An animation timeline is provided so that users may scrub through the visualization of loop dominance and pin it at any point in time to examine the state of the model (structurally or behaviorally) at that specific dt. Users are also free to adjust the link and loop inclusion thresholds at any point during the simulation as results are animating, or while they're scrubbing through results to explore the various levels of complexity in the explanations of model behavior. Figure 8 and Figure 9 depict the animation in both stock and flow and CLD diagrams of the bass diffusion model.

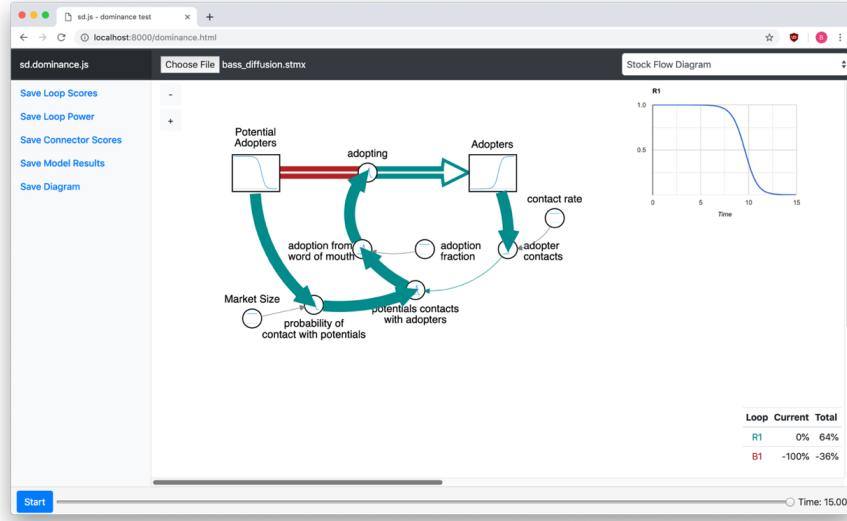


Figure 8: Screenshot of LoopX showing a stock and flow diagram of the Bass Diffusion model at the final time period. Notice the coloring of the split flow 'adopting'. Red links are negative, green links are positive.

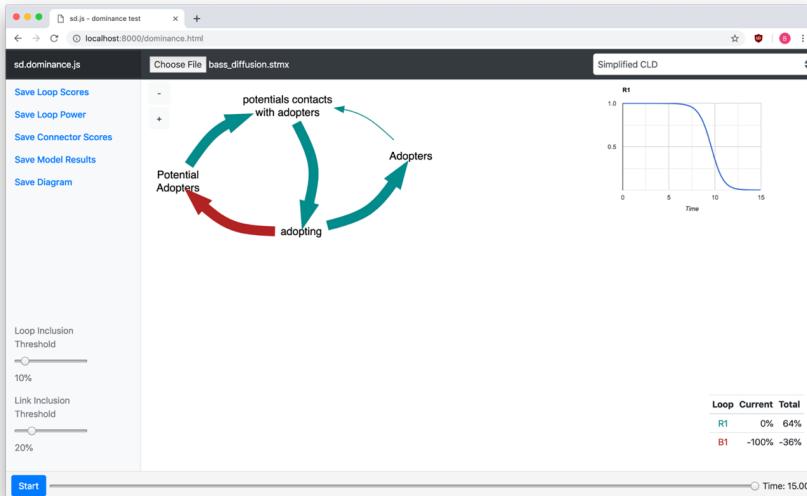


Figure 9: Screenshot of LoopX showing a simplified CLD of the Bass Diffusion model at the final time period. Notice how Potential Adopters is driving potentials contacts with adopters signifying the importance of B1. Red links are negative, green links are positive.

An interesting challenge in visualizing link importance and polarity in SFDs, is the case of flows. Flows are often times connected to two stocks and therefore have an

in-polarity and contribution and an out-polarity and contribution. In these cases (bass diffusion model), it would be non-sensical to render the whole flow with a single polarity and contribution because it is nearly always guaranteed that the flow has opposite polarities and often times it has different contributions. To solve this problem, it makes the most amount of sense to split the flow in half, rendering the pipe sections before and after the flow valve with the polarity and contribution associated with the connecting stock. This produces flows which make it very clear to users of the SFD of the hidden information links they contain. This is especially true re. the outflow from stocks where there is no arrowhead from the flow to the stock. This can be observed in Figure 8.

Discussion

CLD generation with neato and the improved edge curving algorithm is successful because the generated CLDs appear to be naturally drawn, yet they tend to follow best practices as laid out by Richardson (1986). Also, the generated CLDs tend to minimize the instances of non feedback looking like feedback while keeping short loops close in 2D space. This enables users of these diagrams to enjoy not only all of the benefits of curved edged diagrams as measured by Xu et. al, 2012, but in addition, feedback becomes much easier to identify at a glance. Finally, the techniques used to produce machine generated CLDs described in this paper can be applied independently from LTM and any other techniques discussed in this paper to generate high quality CLDs from network data.

The process for generating simplified CLDs works well because it maintains consistency of information regardless of aggregation level. Because each aggregated link is composed of a specific and known list of disaggregated links, animation and visualization of the relative link score is not affected, because the link score is designed to be multiplied. Because the aggregated diagrams are fully accurate representations of the relationships between the variables selected, there is confidence that information loss is minimal. Regardless all information loss is controlled directly by the user via the choice of value for the loop inclusion threshold parameter.

The loop inclusion threshold will allow for information loss when it is set such that the stocks of specified unimportant feedback loops are removed from the diagram. This means that the representation of those feedback loops in the simplified CLD can be lost if those stocks are not also resident in feedback loops of more importance.

The link inclusion threshold rarely leads to the loss of feedback loops in the aggregated diagrams because of the tendency of systems to generate large fluctuations in link score when approaching and leaving equilibrium states. Therefore, the link inclusion threshold exceedingly rarely tends to be the source of removal for stocks and their associated feedback loops from the aggregated diagram. The utility of the link inclusion threshold is to very quickly identify any relationships in the model that serve to expand the number of variables for reasons of equation simplification, - as opposed to for reasons of dynamic complexity. It acts as a surgical scalpel for cutting away all of the variables in the model that do not serve as the interface between feedback loops, - allowing users to be presented with diagrams that contain a minimum number of variables representing a maximal amount of dynamic complexity.

Simplified CLDs are potentially useful in a wide variety of contexts. The first is education where having access to accurate, but simpler depictions of structure could enhance learning opportunities. The second is during model construction, where simplified CLDs could be used by model authors to verify their understanding of the most important dynamics driving behavior in their models. Other use cases include presenting overviews of key model structures to policy makers. Generally, simplified CLDs are useful in any situation where someone may want to explain a model, but not have to step through the full stock and flow diagram.

The simplified CLD process may also be useful for model simplification before running the ADAS algorithm (Schoenenberger et. al., 2017) to reduce model structure complexity before attempting to pattern match system archetypes. The ADAS algorithm ought to do a better job of finding system archetypes in their reduced forms. This should be possible because the ADAS algorithm will not need to

consider so many different possible mutations of structure. Also, if the network searched, was limited by the loop and link inclusion thresholds then the results would also be limited to the structures which are provably most relevant to the model. This would ensure that the algorithm presents the most comprehensive list of matches that are the most relevant to system behavior.

Simplified CLDs though are not without their problems. Currently there is no indication of the quality of the generated diagram, and there is the potential for oversimplified explanations of behavior to be produced. Currently the only way to combat this is by manually checking to see how many of the important feedback loops as determined by their average magnitude of the relative loop score, are actually resident in the generated CLD and evaluating the utility of the CLD based on that. A second important problem is that the simplified links presented, while known to exist in the full model structure, may not be the most important causal pathway between the two variables. This is because a simplified link may represent many different causal pathways, each with its own different importance and the simplification method only chooses the first one and presents that as if it were the only pathway. This is a more significant problem to work around as it requires examining larger diagrams to specifically track the simplification as it happens.

The choice of the relative link magnitude normalized at each time step for each set of independent-to-dependent variables requires further explanation. A normalized value is required for the animation of connectors and flows because a maximum thickness needs to be set. Otherwise thickness would have no bar to measure it against. Without normalization, using the link score would create links whose thickness explodes towards infinity just as the link score does when models pass through or reach equilibrium states.

Another potential other choice for the source of data for the rendered link thickness could be an approach which would apply loop scores, sizing all of the links in a loop equally to better emphasize loop dominance. The problem with animating the loop score arises often in models of any significance where important loops are derivatives

of each other, sharing many links in common. The trouble in these cases boils down to how to represent and display the information that a link is resident in multiple loops, - each with their own level of significance. Theorized techniques include drawing multiple links, one for each loop, or flashing through representations over time (per each time step) in proportion to loop contribution. Realistically though, any techniques chosen to represent loop score over links resident in multiple loops, will not scale with model complexity. This problem is, moreover, compounded by the aggregation techniques presented. Because aggregate links, by their very nature, tend to be resident simultaneously in even more loops.

Other possible options for animating link thickness includes normalization of the link score to all link score values in the entire model at that specific time step, or normalization of the link score to all link score values in the entire model across all time steps. The potential benefit of normalizing link score across all links at all times, is to offer an impression of the activity level of the model as a whole overtime. This would make very clear when the model is reaching equilibrium, - as links would then tend to get thicker during these periods. The problem, though, is that link scores even when plotted on a logarithmic scale appear to be exponential in shape during equilibrium events. This means that dynamics would be all but impossible to observe at any time except for during equilibrium events because connectors and flows would constantly remain tiny. Normalization across all links at a specific time period suffers from the same general issue, except its applied not to the diagram over time, but to parts of the diagram that reach equilibrium marginally slower or faster than other parts. That would give a very misleading perception of the model, exhibiting drastic shifts in dominance from one time step to the next, which is not borne out by the data.

The flaws in using relative link score magnitude as the source of the animation of connector thickness is that loops are not made any more easily identifiable and that the loop power is completely unobservable because values are normalized at each time step. Ultimately, though, these downsides are mitigated via the loop legend that allows for quick and easy access to loop scores and, in the future, to loop power

information over time, as well as for a quick and easy way to identify any loop of interest.

Conclusions

In the final chapter of Business Dynamics (2000), Sterman issued many challenges for the future of system dynamics, three of which have been answered by the creation of LoopX. The first challenge; “Automated identification of dominant loops and feedback structure”, has been answered previously by other techniques including EEA, PPM. But, for this first time, one of these automated loop dominance analysis techniques has been automated and packaged in such a way that the outcomes are easily accessible to a wide swath of practitioners in the field. The second and third challenges; “Visualization of model behavior” and “Linking behavior to generative structure”, also have a long past set of accomplishments. LoopX represents a major success because it integrates loop dominance analysis techniques with model aggregation and visualization. LoopX produces high quality, easy-to-decipher, animated SFDs and high-quality machine generated animated CLDs of the origins of model behavior via the integration of the results of an automated loop dominance analysis done by LTM.

At the current date, LoopX represents only a start to what ultimately may be possible. Efforts must be undertaken to measure the effectiveness of these techniques for teaching purposes, practitioner purposes and, potentially, after future revisions for use by the general public, before any definitive statements can be made about achieving Sterman’s goals. Problems still need to be addressed include the scalability across giant models of a size such as T-21 or its brethren, which must include a significant re-engineering effort focused on deriving efficient solutions to the process of finding the simplified links. The ultimate viability of these techniques will be proven via their adoption in mainstream tooling.

References:

- Chernobelskiy, R., Cunningham, K. I., Goodrich, M. T., Kobourov, S. G., & Trott, L. (2011, September). Force-directed Lombardi-style graph drawing. In *International Symposium on Graph Drawing* (pp. 320-331). Springer, Berlin, Heidelberg.
- Eades, P. 1984. A heuristic for graph drawing. *Congressus numerantium*, 42, 149-160.
- Eberlein, RL. 1989. Simplification and understanding of models. *System Dynamics Review*, 5(1), 51-68.
- Forrester, JW. 1968. *Market growth as influenced by capital investment*. Industrial Management Review.
- Gansner, ER. 2014. Using Graphviz as a Library (cgraph version). *published online August 21*.
- Gansner, ER, & Hu, Y. 2010. Efficient, proximity-preserving node overlap removal. In *Journal of Graph Algorithms and Applications* (pp. 53-74).
- Kamada, T., & Kawai, S. 1989. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1), 7-15.
- Kampmann CE. 2012. Feedback loop gains and system behaviour (1996). *System Dynamics Review* 28(4): 370–395.<sup>[L]
[SEP]</sup>
- North, S. C. 2004. NEATO user's guide. *Murray Hill, NJ: AT&T Bell Laboratories*.
- Oliva, R. 2004. Model structure analysis through graph theory: partition heuristics and feedback structure decomposition. *System Dynamics Review*, 20(4): 313-336.
- Oliva R. 2016. Structural dominance analysis of large and stochastic models. *System Dynamics Review* 32(1): 26-51
- Powers, R. (2019). SD.js in-browser system dynamics model simulation and display. GitHub repository: <https://github.com/bpowers/sd.js>

- Richardson, G. P. (1986). Problems with causal-loop diagrams. *System dynamics review*, 2(2), 158-170.
- Saysel, AK, & Barlas, Y. (2006). Model simplification and validation with indirect structure validity tests. *System Dynamics Review*, 22(3), 241-262.
- Schoenberg, W. (2009). The Effectiveness of Force Directed Graphs vs. Causal Loop Diagrams: An experimental study. In *The 27th International Conference of the System Dynamics Society*.
- Schoenberg, W, Davidsen, P, Eberlein, R, 2020. Understanding model behavior using loops that matter. *Under Review at the SDR*
- Schoenenberger, Lukas, Alexander Schmid, and Markus Schwaninger. "Towards the algorithmic detection of archetypal structures in system dynamics." *System Dynamics Review*31.1-2 (2015): 66-85.
- Schoenenberger, L., Schmid, A., Ansah, J., & Schwaninger, M. 2017. The challenge of model complexity: improving the interpretation of large causal models through variety filters. *System Dynamics Review*, 33(2), 112-137.
- Sterman JD. 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Irwin/McGraw-Hill, Boston.
- Xu, K., Rooney, C., Passmore, P., Ham, DH., & Nguyen, PH. 2012. A user study on curved edges in graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12), 2449-2456.

8. Article #3: Feedback System Neural Networks for Inferring Causality in Directed Cyclic Graphs

By: William Schoenberg

Abstract

This paper presents a new causal network learning algorithm (FSNN, Feedback System Neural Network) based on the construction and analysis of a non-linear system of Ordinary Differential Equations (ODEs). The constructed system provides insight into the mechanisms responsible for generating the past and potential future behavior of dynamic systems. It is also interpretable in terms of real system variables, providing a wholistic, causally accurate, and systemic understanding of the real-life interactions governing observed phenomena. This paper demonstrates the generation of an n-dimensional ordinary differential equation model that can be parameterized to fit measured data using standard numerical optimization techniques. The model makes use of feed forward artificial neural nets to capture nonlinearity, but is a parsimonious and interpretable representation of the network of causal relationships in complex systems. The generated model can easily and rapidly be experimented with and analyzed to determine the origins of behavior using the loops that matter method (Schoenberg et. al 2019). A demonstration of the utility and applicability of the method is given, showing that it produces an accurate, and causally correct model for a three state, non-linear, complex dynamic system of known origin.

Generalization to other dynamic systems with other data sources is then discussed.

Introduction:

Successfully identifying the true causal relationships in complex systems that produce a certain dynamic behavior has arguably been a main focus of science throughout time. As science evolves and technology improves, the methods and tools we use to discover causality ought to change and adapt to keep pace. Present-day machine learning methods, including probabilistic modeling, kernel machines or deep learning, have arisen from a strong, almost single minded, focus on data-oriented

empiricism making use of the extraordinary amounts of observational data available from a plethora of sources (Ghahramani, 2015; Schölkopf & Smola, 2008; Goodfellow et. al., 2016). When viewed through the lens of accurate prediction power, machine learning has proven to be quite successful but, present-day machine learning techniques typically fail to reveal the fundamental causal mechanisms driving behavior. Although, it must be said that interpreting the structure behind black box, deep learning models is an active research area (Montavon et. al, 2018). To make full use of the technological advancements in machine learning, significant emphasis must be placed on finding a valid and interpretable causal understanding of the underlying real-world system (Runge et. al, 2019).

Observational causal inference as a field started with applied statistics and has grown out of the seminal works of Wiener and Granger (Wiener, 1956; Granger, 1969). The most well-known method for observational causal inference is Granger causality which tests whether omitting the past of a time series X in a time series model including Y 's own and other covariates' past increases the prediction error of the next time step of Y (Granger, 1969). Granger causality is useful in discovering specific causal links in a system, but it fails to generalize to complex non-linear systems, typically failing to identify all of the links in the networks of feedback relationships which govern these systems from a wholistic perspective (Spirtes & Zhang, 2016).

Non Granger methods in observational causal inference can be categorized into the following three broad categories as done in the literature by Runge et. al (2019): nonlinear state-space methods, causal network learning algorithms and structural causal model frameworks. Of particular interest are the techniques which underlie causal network learning algorithms, especially the PC algorithm and its brethren. These methods start with either an empty network or a fully connected network and iteratively add or remove edges testing against an invariant payoff function to evaluate the likelihood of the validity of the causality of the generated network structures (Verma & Pearl, 1990; Spirtes & Glymour, 1991; Zhang, 2008; Runge et. al, 2018; Runge, 2018). These methods are of particular interest because they start by generating a structure which then generates a behavior. Methods of this class have

the potential for the broadest of applications and are particularly relevant to systems which are representable as ordinary differential equations (ODEs), because networks of causal relationships are simply, and as Forrester argues, ideally represented (and therefore analyzed) in the form ODEs (1994).

The major problem with the state of the art in the field of observational causal inference is that it does not work in feedback rich systems which by definition cannot be represented as directed acyclic graphs. This method demonstrates a new approach to causal network learning algorithms which generates then analyzes an n-dimensional, non-linear, complex system of ODEs to do automated causal inference with a high level of accuracy from a holistic systems perspective incorporating feedback and time delays. This approach provides insight and understanding into the mechanisms responsible for generating past and future behavior of dynamic systems, giving users a holistic, causally accurate, and systemic understanding of the real-life networks governing observed phenomena.

The method

At the center of a FSNN (Feedback System Neural Network) is an attempt to apply the core principles of empirically driven machine learning to observational causal inference. Key to the function of this method is the construction of a model which is a highly non-linear complex system of ODEs evaluated over a time dimension. The system of ODEs, referred to as a generated model, is analyzed to determine the origins of behavior. In the generated model, the state variables constitute the memory of that model and their relationships (through perceptron firing) are the source from which behavior originates. To build the generated model, first the method constructs the system of ODEs which links the state variables together via their derivatives. Second via a process of optimization the generated model is parameterized to reproduce the observed state trajectories the system exhibited. Third, the now parameterized generated model is studied and the mathematical relationships between the state variables are objectively measured which clarifies the origins of behavior within the empirically trained, generated model. Finally, the causal explanation is

then extracted and can be validated by subject matter experts, turning the fruits of the method from a black box data-driven machine learning model into a transparent and analyzable structure-driven model.

The first step in creating the generated model is to generate a system of ODEs to be parameterized. To do that the state variables in the ground truth system are enumerated and initialized from observed data. Next, in the model being generated, the derivative of each state variable is set to be a function of every other state variable as well as itself. It is these relationships in the derivative functions that represent the opportunity for the learning that will take place during the model training/parameterization process. The process of constructing the derivatives for each state variable offers the end-user the ability to add a-priori knowledge to the model by specifying which causal relationships are known not to exist, and therefore can be excluded from the learning process. This is done by eliminating the relationship between any directed pair of source and target state variables.

The relationships between each directed pair of state variables can take any form that the user desires. There are generally two intervention areas available. The first is the form of the equation of the term(s) that will represent each state variable in the derivative of each other state variable including itself. The second, is the form of the equation to combine the terms representing each state variable in the derivative function of all state variables. The factors to consider in making the decision of what formulations to use are the general applicability of the range of producible behavior modes to the observational data being matched and the computational complexity in terms of the numbers of parameters necessary to train the generated model while being mindful of overspecification and degenerate payoff surfaces.

The optimal structure discovered to parameterize and train the model is one which connects all the state variables via a neural network as is done in the hidden state of neural ODEs (Chen et. al, 2018). For the purposes of this method, the states are not a part of the hidden network. Specifically, for each state variable in the model, there is a single multilayer perceptron neural net constructed such that there is one output,

which is the derivative for the state variable, and the input layer is constructed using the current value of all the state variables at each calculation interval. The hidden state of each of these neural networks is individually constructed according to best practice and the specific dataset being studied. This structure is ideal because neural nets are universal approximators, theoretically allowing them to learn any non linear function which dictates the relationships between state variables (Cybenko, 1989). The universal approximation theorem allows this meta-model to be applicable to any ground truth system. To specify that a certain relationship between state variables does not exist, simply remove the input node representing the source from the derivative function of the target's neural net's input layer.

The second step is model training/parameterization and is where an optimization process is run defining as a payoff function the minimization of the squared error between the training data and the calculated values for each state variable. Any ODE solver can be used to simulate the model, and in the example a Runge-Kutta 4 (RK4) solver was used. The optimization algorithm used in the example case was Powell's BOBYQA (bound optimization by quadratic approximation) as implemented in the public open source project DLib version 19.7.

The third step, once the trained model has been generated, is to analyze that generated model using the link score metric from the loops that matter method to measure and extract the meaningful causal relationships (Schoenberg et. al, 2019). The link score metric reports the contribution at a point in time that an independent variable (x) has on a dependent variable (z) arranged in the function $z = f(x, y)$. For this approach the link score is used to measure the impact that one state variable has on another so that the causal relationship between state variables can be quantified and understood. Equation (1) is reproduced from Schoenberg et. al. below. Link scores are computed using the calculation interval employed by the ODE solver.

The link score for the link $x \rightarrow z$ is:

$$LS(x \rightarrow z) = \begin{cases} \left(\left| \frac{\Delta_x z}{\Delta z} \right| \cdot sign \left(\frac{\Delta_x z}{\Delta x} \right) \right), \\ 0, \quad \Delta z = 0 \text{ or } \Delta x = 0 \end{cases}$$
(1)

Where Δz is the change in z from the previous time to the current time. Δx is the change in x . $\Delta_x z$ is the change in z with respect to x . From a computational perspective $\Delta_x z$ is the amount z would have changed, conditionally, if x had changed the amount it did, but y had not changed. The first major term in this equation represents the magnitude of the link score, the second is the link score polarity.

The magnitude of the effect (force is a good analogy) that x has on z is relative to all of the effects on z . This is a dimensionless quantity, and if all of the effects are in the same direction, it is the fraction of the change in z that originates in a change in x . If the formulation of z is linear, then the values are restricted to the range [0,1]. When there are negative and positive effects, these numbers may be very large in magnitude, but this does not harm the overall analysis of the generated model (Schoenberg, et. al, 2019). The absolute value is used because the change in z could be in either direction due to the effects from other variables, regardless of the magnitude of the effect that x has, implying that the polarity can and would be wrong.

The polarity of a link is defined as the sign of the partial difference at time t . This formulation is the same as the one used in Richardson (1995), though the formulation there was as a partial derivative, not difference. The polarity numerator is the same as it is for the magnitude, but the denominator is the change in x . When x does not change, the score is by definition 0, though the magnitude would be 0 in any case since no change in z would be attributable to x .

Link scores can be multiplied together following the chain rule of partial differentiation allowing the calculation of the system of equations to proceed at any

desired aggregation level (that is with more or fewer intermediate algebraic computations). Typically for models generated using this technique link scores are calculated along the pathway of the relationship from one state variable directly to the derivative of another state variable. It is possible to calculate link scores which pass through other state variables back to the original state variable forming complete feedback loops. This is valuable for models with known or physically bound structure, but of less value for all but the simplest of generated models. This is because this method of generating differential equations produces (assuming no a-priori information), a number of feedback loops which is the factorial of the number of state variables, numbers which quickly overwhelm computational and analytic evaluations.

When following the standard practice of calculating link scores which measure the impact of one state variable (source) on the derivative of another (target), the generated link scores are then normalized across all of the relationships which have the same target, at each point in time. This produces a signed percentage score which describes specifically and objectively what percentage of the behavior of the target state variable is contributed by the source state variable, including what the polarity of their relationship is at that specific point in time.

Application to a non-linear dynamic system

To properly demonstrate the application of a FSNN, a model of a three-state system was used as the ground truth. The ground truth system is an ODE which consists of three state variables ($State_1$, $State_2$, $State_3$) where each state variable is directly connected to itself, and its predecessor in a large feedback loop and each state is connected to itself with a short feedback loop. This system contains 4 feedback loops in total (all with a balancing polarity), and the equations are shown in Table 1. The training datasets were generated by simulating the ground truth system using an RK4 solver with a dt of 1/4 for 100 time units producing three full cycles of a damped oscillation shown in Figure 1. The training data was sampled at the end of each time

unit, giving 100 time points per state variable across two distinct initializations of the states. Initialization 1 (29, 96, 4), Initialization 2 (22, 11, 78).

Table 1: Equations for ground truth system with units. f is a monotonically increasing sigmoid function over the x range [0, 100] and y range [0, 100] with inflection point at (50, 50).

$\text{State}_1 = \text{flow}_2 - \text{flow}_1$	Units
$\text{State}_2 = \text{flow}_3 - \text{flow}_2$	Units
$\text{State}_3 = \text{flow}_4 - \text{flow}_3$	Units
$\text{flow}_1 = (g - f(\text{State}_3))/t$	Units/Time
$\text{flow}_2 = \text{State}_1/t$	Units/Time
$\text{flow}_3 = \text{State}_2/t$	Units/Time
$\text{flow}_4 = \text{State}_3/t$	Units/Time
$t = 5$	Time
$g = 75$	Units
f	Non-linear sigmoid function

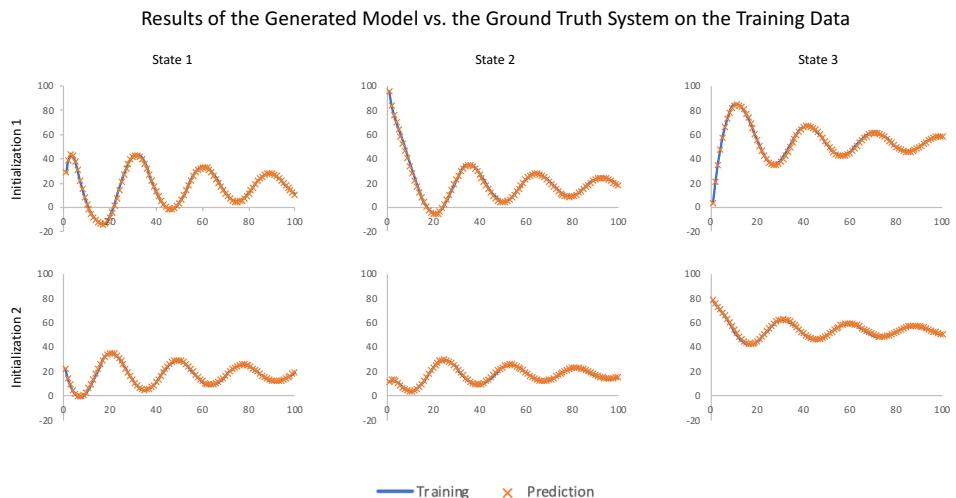


Figure 1: Performance of the generated model on the two training initializations. Initialization #1 (29, 96, 4), Initialization #2 (22, 11, 78).

The training progressed in a single phase where all network weights and bias parameters were set to 0 representing a starting point with no causal structure. Each neural net was constructed with three hidden layers, of 8 then 6 then 4 nodes, and each neural network took input from all three states and produced a single output. The activation function used in all nodes was a *tanh* function. All state variables used as inputs were linearly re-scaled to the range [-1, 1] by dividing the value of each state variable by 100 (an externally determined maximum potential magnitude for any of the states) and multiplying the value of the output node by 100 so that the state variable values would remain accurate to the ground truth system. Figure 1 demonstrates that the model performed well on the two training datasets, where the blue lines are the data generated by the ground truth system and the orange points are the predictions by the generated model.

To validate that the generated model is a causally accurate recreation of the ground truth system, the generated model must reproduce the proper causal connections as measured in the source model, which is demonstrated in Figure 2. A pathway based link score analysis of the first training set initialization of the ground truth system was performed, measuring the polarity, and nature of the relationship between each state variable and each other state variable including itself. The results of that analysis are the blue lines in Figure 2. Negative numbers mean that the relationship has a negative polarity, and are therefore balancing, positive numbers mean a positive polarity, and are therefore reinforcing. The magnitude of the numbers represents the strength of the connection at each point in time. The same analysis was then performed on the generated model using the same first training set initialization.

An analysis of Figure 2 yields that conclusion that the generated model has identified the proper causal links in the ground truth system, and that the generated model has properly identified the polarity for each of those links. Figure 2 also indicates that the generated model is not the singular well defined model of the ground truth system because the exact nature of the relationships doesn't follow the exact same pattern.

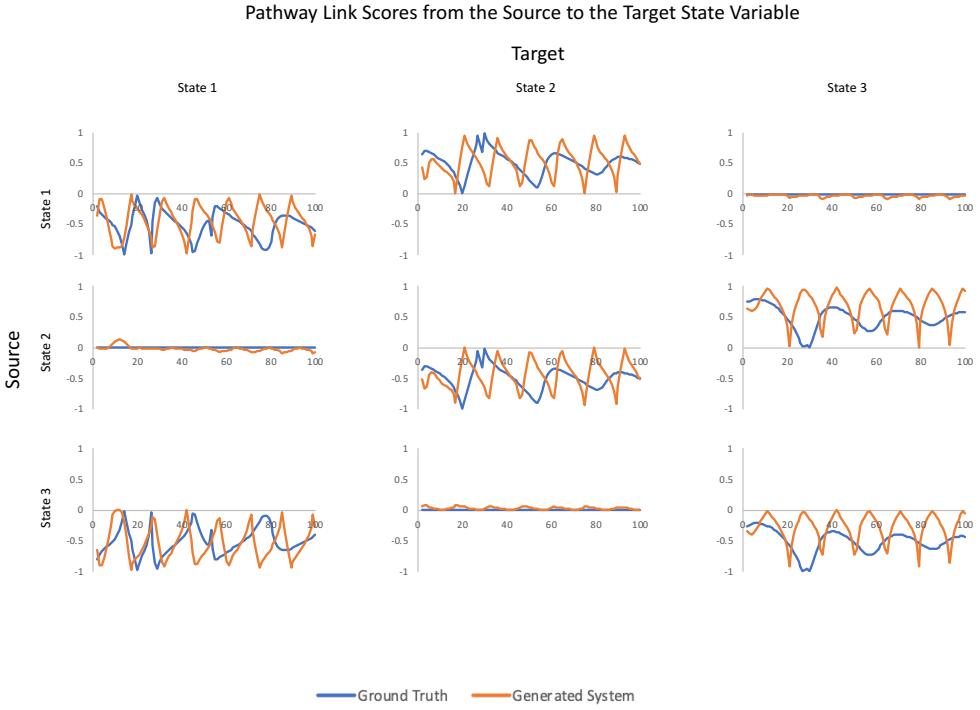


Figure 2: Demonstration of identification of proper causal links and their associated polarities. Graphs showing 0 magnitudes means there is no connection between the source and target. Negative values mean a balancing link, positive values a reinforcing link. The magnitude represents the strength of the connection at that instant in time.

Next, the generated model's performance on untrained, testing data was measured. Figure 3 demonstrates the results of the error distribution across 100 unique Monte Carlo generated, random initializations using Sobol sequences (Sobol et. al, 1999). Sobol sequences were used to best explore the selected input state space (Burhenne et. al, 2011). The selected input state space was limited to the condition that the sum of the initial value of the states must be within the range [30 – 150]. The distribution of that total initial value to each of the three states was unrestricted. This condition is based upon the sum of the initial states in the two initializations used to train the model, this important decision is explained in depth below in the next section.

Figure 3 demonstrates remarkable performance in reproducing the behavior of the ground truth system in the large majority of the test cases. The 95% confidence bounds of the error for all three states are close to 0 which demonstrates that the

generated model is an accurate recreation of the ground truth system in the behavioral space under the given range of initial conditions.

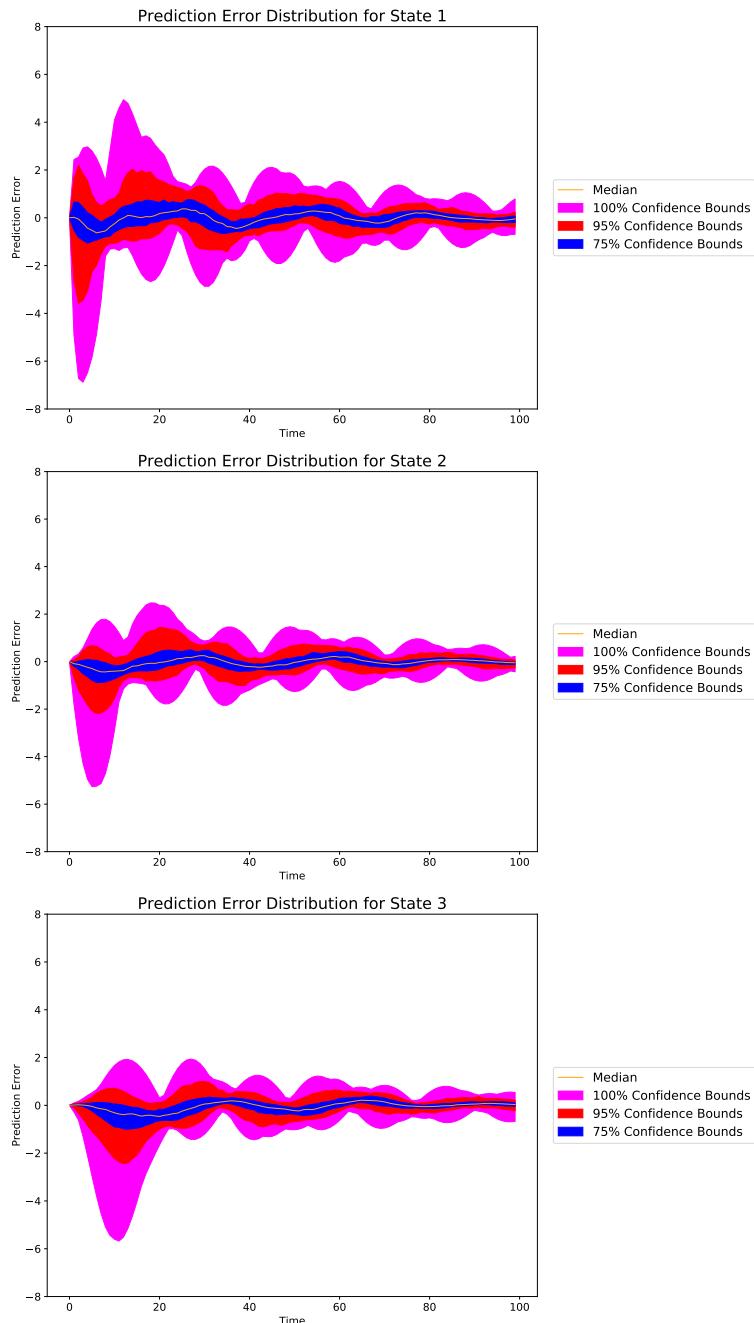


Figure 3: Prediction error for each of the three states from 100 Monte Carlo generated initializations of the generated model where the sum of the initial state values was between 30 and 150.

Challenges and considerations for dealing with non linear feedback systems

The above example demonstrates the problems with finding the singular well-defined model which perfectly explains the ground truth system in the structural space, and therefore as a result in the behavioral space as well. The generated model above was not the singular well-defined model for the ground truth system, because the link score profiles seen in Figure 2 did not match perfectly in all time points for all relationships. Instead, the generated model was one of a set of models which matches the behavior over a range of initializations well, because in the structural space it shares many similarities with the ground truth system. Because the true structure of the ground truth system was known, it was therefore determinable that the recreation of that ground truth system did not match in every way, but it was possible to confirm that the recreation matched the key, high level attributes of the causal structure including: identification of all correct causal links and their polarities, plus the general forms for the relationships governing the ground truth system's behavior. The consequence is, not surprisingly, that the model successfully passes the behavior validation tests to which it was exposed.

Even after infinite training on infinite data, the meta-model at the heart of this method may not be well defined for a specific ground truth system, and the chances of finding that singular well defined model falls quickly as the amount of training, or data is cut to finite levels. Because a neural network is a universal approximator (Cybenko, 1989), the meta-model is capable of producing an infinitely wide range of behavior patterns, encompassing an infinitely large set of possible models. After training, the generated model is just one possible explanation, a simple hypothesis for the causal relationships governing the origins of the given data. That explanation may just as easily be proven wrong by the introduction the next piece of data which encoded within it are more hints about the subtle facets of the structure of the ground truth system which created it. Finding the singular well-defined model of the ground truth system in a synthetic data experiment like the one presented here is a fool's errand, which, even if possible, potentially requires infinite training on infinite data because of the unbounded range of the ground truth system. For many systems, the

distinction between the generated behaviors of the singularly well-defined model of the system, and a (potentially quite large) set of highly accurate generated models of that same system, is minuscule to insignificant in the behavioral space, but vastly different in the structural (i.e. causal) space. Specifically, the causal structures behind those two classes of models can potentially be totally unrelated, sharing very little, if anything at all in common. At the end of the day, the only way to establish with any certainty real ground truth in unknown systems is via validation of and empirical experimentation on, the hypothesized causal structures, far beyond what any behavior-based validation regime can do.

The example analyzed in this paper is a demonstration of the method under a favorable set of conditions. In this experiment, the generated model could be trained across two distinct unrelated initializations, with full, perfectly accurate (no measurement error) and regular data available for all system states with no extraneous data. In addition, the magnitude for the system states was well-defined, and the training data sampled well the potential behavioral space of the testing data, while producing a well sloped payoff surface. The results of this experiment would have been different if any one part of this perfect storm of conditions was not present.

The best way to understand the construction and training of the generated model is to imagine the ground truth system, and therefore the generated model as an n -sized set of n -dimensional functions, where n is the number of state variables of the ground truth system being studied. Each one of those n -dimensional functions describes, irrespective of time, the functional relationship between all the states and the change in each specific state. Each one of those n -dimensional functions produces a manifold in state space, whose surface is the unique fingerprint that yields the information necessary to identify each of the n -dimensional functions which are the causal structure of the system. During training the goal is to examine as much of each manifold as possible so that the neural nets whose job it is to learn each of those n -dimensional functions can reproduce as much as is possible of the ground truth manifolds in the generated model. Each additional point in time that is used in

training brings along with it a new opportunity to examine another portion of each manifold.

Training across multiple distinct initializations helps to combat overfit, expands the range of the generated model, and reduces problems with confounders by exposing the generated model to more facets of the manifolds of the ground-truth system.

Multiple initializations better exercise the ground truth system, revealing additional information which would not be accessible from just a single simulation. Initializing the system in multiple distinct ways tends to break any false learned causations in the generated model, more commonly referred to as confounders. This is because, while in one initialization two state variables may exhibit similar state trajectories because of the impact by a third one, in a second initialization, the first and second states may exhibit opposite state trajectories hinting at a proper identification of state three as a driver of one and two. Even with just two distinct initializations, performance significantly improves regarding the identification of the proper causal structure because the incidence of confounders drops precipitously. This is because the chances of repeated correlations across any two specific timesteps in the first derivatives of the state variables drops.

Increasing the number of initializations trained on, increases the scope of the generated model because, during the training phase, the generated model explores more of the manifolds of the ground truth system. When the number of initializations used in training is low, only a few of the potential combinations of the values taken by the n state variables are actually encountered. Consequently, only a very small fraction of the total surface area of the n manifolds is actually explored and exploited in the identification of the underlying causal structure. Therefore, each neural network derived is generally only applicable to a small fraction of the full (potential) input space because the function which describes the relationships between the states at those unexplored values is still completely unknown.

Each unique initialization constitutes a combination of values from the n state variables and may produce a new n dimensional state trajectory. Each single

initialization tends to settle into a specific area of the manifolds generated by the ground truth system (with the exception of chaotic systems), therefore adding more samples to any one initialization tends to be less important. To demonstrate, all forms of non-linear dynamic systems at the limit of time can be categorized by their behavior. There are three such categories of systems. First there are chaotic systems which never settle into a regular behavior pattern. Second there are oscillatory systems, which are systems that at their limit produce a standing oscillation through the same sets of values. Finally, there are systems in equilibrium, which over time tend towards producing state variables which do not change. In all but the chaotic systems, eventually with enough time spent simulating, for each initialization a point will be reached where no additional portions of the manifolds are being explored because all combinations of state variables which are producible by that initialization have already been encountered. Therefore, in all but chaotic systems each new initialization brings with it more opportunity to find new never before seen combinations of the state variables, therefore providing insight on more of the manifolds of the ground truth system, and therefore improving the performance of the generated model. This is why each additional initialization of the ground truth system is so important for training purposes versus each additional sample of an already explored initialization.

For the case of the specific example above, the surface of the ground truth system's manifolds was not explored where multiple states took on values of 70 or greater. This is because the ground truth system always produces a dampened oscillation, and training took place with an initial sum of the states of 129, and 111. Therefore, as is demonstrated in Figure 4, when the initial sum of the states is greater than 150 the prediction begins to rise exponentially. It does this because at initialization values of 150 or greater, the probability of encountering multiple states which simultaneously have values over 70 rises. The prediction error gets exponentially worse as the sum of the initial states rises because the probability of encountering further unexplored areas of the manifolds significantly and quickly increases.

Maximum error distributed over the sum of the initial values of the state variables

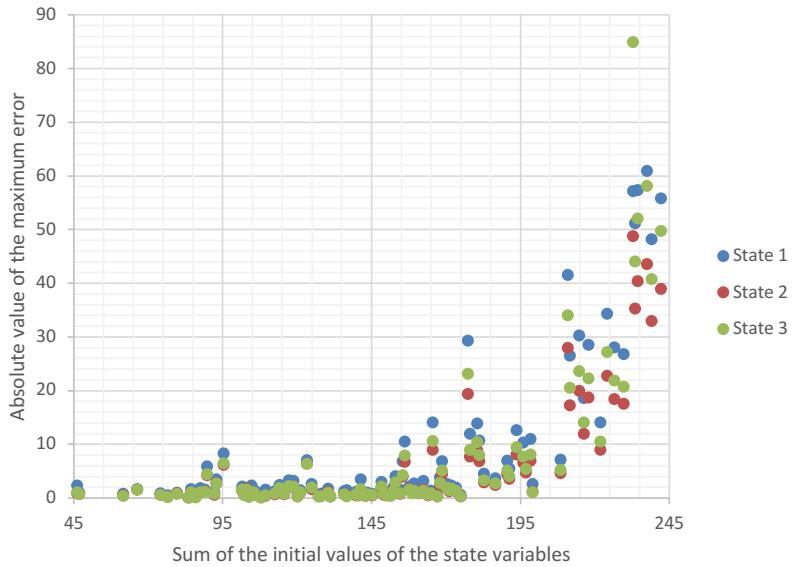


Figure 4: The absolute value of the maximum observed error distribution from 100 Monte Carlo generated, random initializations, using Sobol sequences (Sobol et. al, 1999). Sobol sequences were used to best explore the selected input state space (Burhenne et. al, 2011). This figure shows how leaving the training area adversely affects the generated model's utility, because the training has not been sufficient to discover the non linear functions governing the inter-relationships between the states in areas where there are two or more states which have relatively large values (+70) simultaneously. The generated model is effective over the range of 0 – 150 in the sum of the initial values of the 3 state variables.

When training across multiple initializations, it is important to verify that the initializations are indeed distinct, and that each set of initial states does not show up as a set of calculated values in any of the other training initializations, otherwise the initializations are linked (one initialization is the direct outcome of another), and there is no potential to see additional facets of the manifolds of the ground truth system structure because new state trajectories are not being produced. In real-world systems with an unknown ground truth, this process can be recreated by studying multiple unrelated instances of the system where each instance must share the same exact causal structure.

The utility of the method in situations with anything less than perfect data is still unexplored. Real world systems are messy, and it yet needs to be studied how the method handles measurement error, irregularity in collected data, missing data, and the incorporation of extraneous data. There are many significant challenges in this space still. Because the method always fits to, and therefore attempts to explain all given time-series data endogenously, any error in specification of either extraneous time series data, or measurement error in the data, will cause problems with learning the true causal relationships. The design of the method forces an endogenous explanation for all time-series data it trained on, including measurement error, and unrelated inputs. Both forms of error will be endogenized into the system, and therefore be made part of the feedback structure of the system, leading to improper identification of causal structure. Specifically, extraneous, causally unexplainable time-series inputs will cause the shape of the payoff surface to warp towards degeneracy, because the extraneous data will not be explainable by any of the given state variables and yet the method by its very design and construction must search for an endogenous explanation for its origins.

The final piece of key information known in the example case was the magnitude of the state variables. Because of the use of the *tanh* activation function, a linear rescaling of all state variables was done. This specific combination of a linear rescale, and *tanh* function causes problems with signal attenuation, although, these problems will be present in any combination of a normalization process with the application of a non-linear activation function. In this specific case, as the absolute value of any state variable grows above its associated magnitude, the behavior is pushed into the far reaches (domain) of the *tanh* function, where it planes out, reducing the marginal impact significantly and quickly. For example, functionally, the system responds in nearly the same way to a state variable which has a value two times the associated magnitude vs the same state variable having a value of that magnitude. The ramifications of this problem are especially important for time series forecasting, especially in systems with exponential data distributions because of the right skew in observable values which is fundamentally incompatible with a linear

scaling. This problem is even more insidious because setting larger and larger magnitudes then reduces the marginal impact of changes typically observed on the left side of the data distribution. Therefore, it appears that a logarithmic scaling may be called for in systems with exponentially distributed time series data. Regardless, further study is warranted to understand the relationship between activation function, the scaling function, and the time-series data distribution.

Conclusions

Even in the face of what may seem like insurmountable problems, this line of work, and within that scope, the FSNN in particular are of great value. This specific approach represents a significant improvement in our ability to undertake causally accurate science, using less human input, and has the potential to underlie a revolution in the way that causal structures are identified. For a moment, imagine that significant progress has been made addressing all the challenges mentioned above, and this method is functionally able to identify very nearly the singularly well-defined model with correct structural underpinnings among the large majority of ground truth systems under relatively achievable data requirements. In that case, this method becomes the basis for an artificial intelligence, surpassing that of human beings, which predicts and explains with a high level of accuracy never observed phenomena which occur deep into the future. As often said, "...with great challenges come great rewards..." .

References

- Burhenne, S., Jacob, D., & Henze, G. P. (2011, November). Sampling based on Sobol's sequences for Monte Carlo techniques applied to building simulations. In *Building Simulation 2011: 12th Conference of International Building Performance Simulation Association, Sydney, Australia, Nov* (pp. 14-16).
- Chen, T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in neural information processing systems* (pp. 6571-6583).
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4), 303-314.
- DLIB [Computer software]. (2017). Retrieved from <https://github.com/davisking/dlib>
- Forrester, J. W. (1994). System dynamics, systems thinking, and soft OR. *System dynamics review*, 10(2-3), 245-256.
- Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553), 452.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
- Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, 424-438.
- Montavon, G., Samek, W., & Müller, K. R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73, 1-15.
- Richardson GP. 1995. Loop polarity, loop dominance, and the concept of dominant polarity. *System Dynamics Review* 11(1): 67–88.
- Runge, J., Bathiany, S., Boltt, E., Camps-Valls, G., Coumou, D., Deyle, E., ... & van Nes, E. H. (2019). Inferring causation from time series in Earth system sciences. *Nature communications*, 10(1), 2553.

- Runge, J. (2018). Causal network reconstruction from time series: From theoretical assumptions to practical estimation. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(7), 075310.
- Runge, J., Nowack, P., Kretschmer, M., Flaxman, S., & Sejdinovic, D. (2017). Detecting causal associations in large nonlinear time series datasets. *arXiv preprint arXiv:1702.07007*.
- Spirites, P., & Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social science computer review*, 9(1), 62-72.
- Spirites, P., & Zhang, K. (2016, December). Causal discovery and inference: concepts and recent methodological advances. In *Applied informatics* (Vol. 3, No. 1, p. 3). SpringerOpen.
- Schoenberg, William, Davidsen, Pål, Eberlein, Robert, (2019). Understanding model behavior using loops that matter. *arXiv preprint arXiv:1908.11434*.
- Schölkopf, B., & Smola, A. J. (2008). Learning with kernels. MIT Press, Cambridge, MA. *Learning with kernels. MIT Press, Cambridge, MA*.
- Sobol, I. M., & Levitan, Y. L. (1999). A pseudo-random number generator for personal computers. *Computers & Mathematics with Applications*, 37(4-5), 33-40.
- Wiener, N. (1956). The theory of prediction. Modern mathematics for engineers. *New York*, 165-190.
- Verma, T., & Pearl, J. (1990). Causal networks: Semantics and expressiveness. In *Machine intelligence and pattern recognition* (Vol. 9, pp. 69-76). North-Holland.
- Zhang, J. (2008). On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16-17), 1873-1896.

9. Article #4: Finding the Loops that Matter

By: Robert Eberlein and William Schoenberg

Abstract

The Loops that Matter method (Schoenberg et. al, 2019) for understanding model behavior provides metrics showing the contribution of the feedback loops in a model to behavior at each point in time. To provide these metrics, it is necessary find the set of loops on which to compute them. We show in this paper the necessity of including loops that are important at different points in the simulation. These important loops may not be independent of one another and cannot be determined from static analysis of the model structure. We then describe an algorithm that can be used to discover the most important loops in models that are too feedback rich for exhaustive loop discovery. We demonstrate the use of this algorithm in terms of its ability to find the most explanatory loops, and its computational performance for large models. By using this approach, the Loops that Matter method can be applied to models of any size or complexity.

Overview

The Loops that Matter approach to understanding the structural sources of model behavior described in Schoenberg et. al., (2019) relies on the comparison of the identified feedback loops (using a loop score metric) in a model. In order to make that comparison, the set of feedback loops to use for comparison needs to be known. For small models, all the feedback loops can be found, scored, and compared. For large and feedback rich models, this is not practical just because of the overwhelming number of potential feedback loops as discussed in the background section.

This paper focuses on finding the set of feedback loops to compare when doing loop dominance analysis using the Loops that Matter method. The loopset selection problem that has been solved before as described in Kampmann (2012), and Oliva (2004) but those solutions do not turn out to be appropriate for the purpose of finding the most explanatory loops. The second section in this paper, on Independent and

Important loops demonstrates why this is the case. This demonstration is done using a simple arms race model, which is also helpful in understanding the approach we ultimately adopt in solving the search problem.

In the section on composite feedback structure, we will use a simple model for which the metrics of the Loops that Matter method (Schoenberg, et. al, 2020) can be computed by inspection. This simple model makes clear the dilemma of analyzing a complex model for which the importance of different links and loops is changing during the course of a simulation. As such, even though it is a trivial model, it provides a foundation for developing algorithms to discover important loops in complex models.

Once we have outlined where we want to go, we will discuss the path to get there. The development of the approach described in this paper has been a very iterative process, which many attempts showing promise abandoned in that development. In order to help future researchers who want to improve on our solution, we include discussion of several of those abandoned paths.

Finally, we present the approach that we settled on for production use. The algorithm chosen is, heuristic. It does not guarantee the discovery of the truly strongest loops, but we have reason to believe that it is likely to discover strong loops, and observations on the nature of strong loops in large and complex models give us confidence they will be similar to, if not, the strongest loops.

Ironically, almost all of the discussion in this paper will be based around very simple models for which complete enumeration of feedback loops (a problem long ago solved by Tarjan (1973) and others) is trivial. While we will discuss the application of the techniques to larger models such as the Urban Dynamics model (which has 43,722,744 feedback loops) the main concepts are most intuitively understood with small examples.

Background

Understanding the connection between structure and behavior requires both the recognition of system physics as shown in stock and flow diagrams, and the identification of feedback loops that are responsible for generating the behavior of interest. For small models identifying feedback loops is straightforward as it can typically be done visually and certainly computationally with little difficulty. For larger models, however, the number of feedback loops grows very quickly (potentially proportional to the factorial of the number of stocks) and can't be enumerated with any practicality (Kampmann, 2012). Since the purpose of finding the loops is to identify those that are important in generating behavior, we want to find an approach that identifies the loops most important to the observed behavior, and skips over those of less interest.

This difficulty is well recognized. Kampman (2012) makes it clear and suggests a method for solving it using what is termed an independent loop set that contains a number of loops which is typically proportional to the number of stocks. Oliva (2004) refines this by defining a shortest independent loop set that is both unique and easily discoverable. These solutions are based upon a static analysis of the model equations, and do not take into consideration the behavior which is produced by the model. While the independent loop set approach solves the identification problem in large models, it will not necessarily find the loops most important to understanding behavior as pointed out in Güneralp (2006) and Huang et al (2012). This is discussed further in the next section.

An alternative approach to discovering loops is to use a method that finds the more important loops first, and stops when the set of important loops have been found. This is what we describe in this paper; the method is a heuristic that gives good, but demonstrably incomplete, enumeration of important loops.

We base our discovery approach on the Loops that Matter technique for quantifying the contribution of loops outlined in Schoenberg et. al. (2019). This method is based on a link score that determined the contribution of a changing input to a changing

output in an equation. The score for a loop, is then the product of all the link scores. This means that as we attempt to discover loops, we can know the score of each loop we find as soon as we detect it. By prioritizing search based on link scores, our discovery approach completes in reasonable time even for large and highly interconnected models.

Independent and Important Loops

We use the simple, three-party arms race model shown in Figure 1 to demonstrate what an independent loop set (Kampman, 2012) is. This model also highlights why using the independent loop set will not always find the important feedback loops following the example of Huang et al (2012). Finally, we use this simple model to show how the loop discovery process can be sped up using its local characteristics.

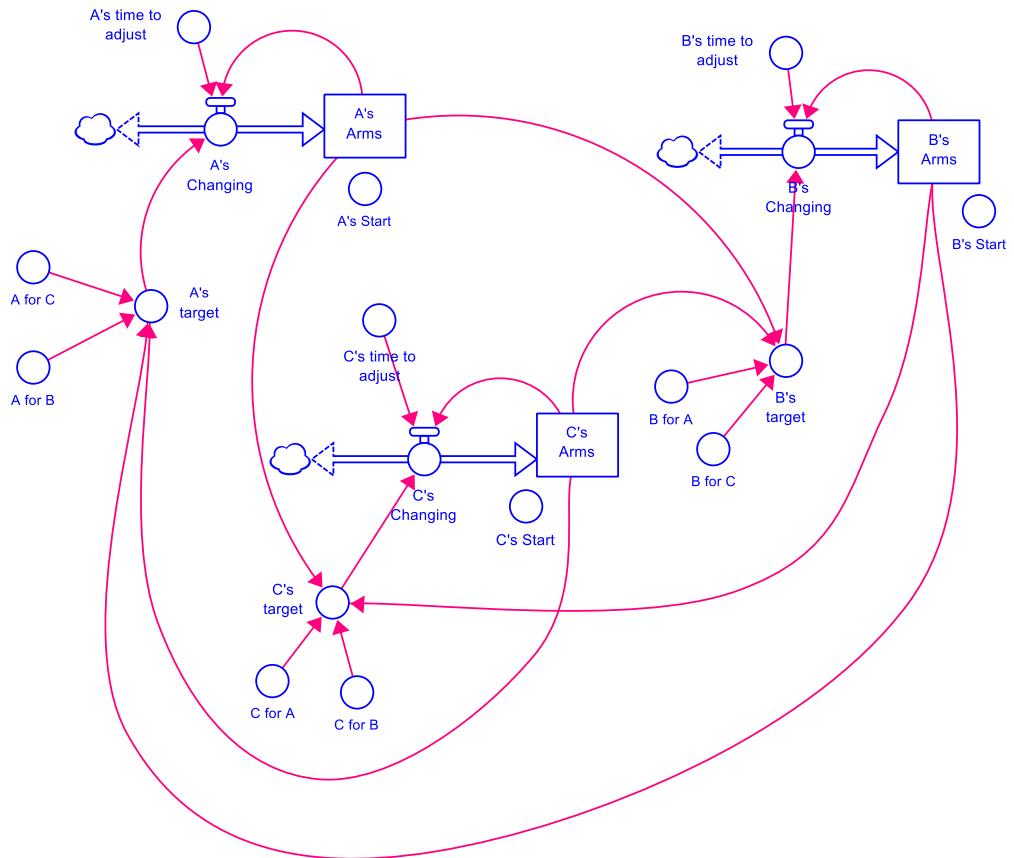


Figure 1: A simple three-party arms race model.

The model is set up so that A wants only parity with B and 90% of C, B wants parity with A and 110% of C, and C wants 110% of A and 90% of B. A starts at 50, B at 100 and C at 150. There are 3 balancing stock adjustment loops (the standard balancing loop in the arms race archetype), three pairwise reinforcing loops A, to B's target, to B, to A's target and so on (the standard reinforcing loops in the archetype) and then two reinforcing loops involving all three players (A to B's target to B to C's target to C to A's target and A to C to B (with intermediate)).

Following the terminology of Oliva (2004) and as identified by Huang et. al. (2012) in a similar model, the shortest independent loops would consist of the 3 stock adjustment loops, and the 3 pairwise reinforcing loops. The completeness of this set

of loops is demonstrated by the fact that all of the connections involved in all of the feedback loops are used in this set of loops. The longer loops connect A to B (already used in the AB pair) and then B to C (already used in the BC pair) and then C to A (already used in the AC pair).

However, simulating this model, or even just thinking through the relative gains from the description above, it should be clear that the pairwise reinforcing loops all have gain equal to or smaller than one. Thus, focusing only on those loops, the behavior will necessarily be adjustment toward balance or toward zero. Because the shortest independent loops include all of the connectors, breaking any loop is guaranteed to break one of the shortest independent loops. This completeness, while guaranteeing we will see an effect from any change in the connection strength from one link to another, does not mean we will see the most important effect. The A to B to C loop is the one that determines the long term behavior of this model as shown in Figure 2, though it does not during the short term adjustments at the beginning.

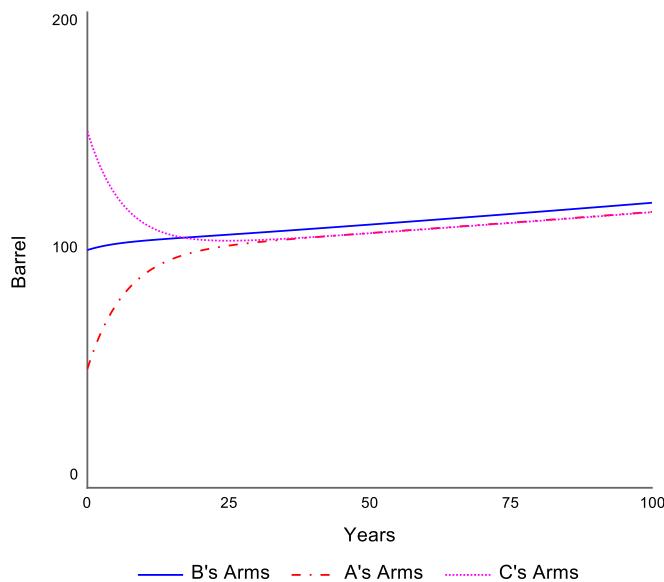


Figure 2: Behavior of the 3 states in the arms race model.

Which loop is important when can be seen quite easily using the loop score metric from the Loops that Matter method. Figure 3 shows the percent of behavior each loop is responsible for:

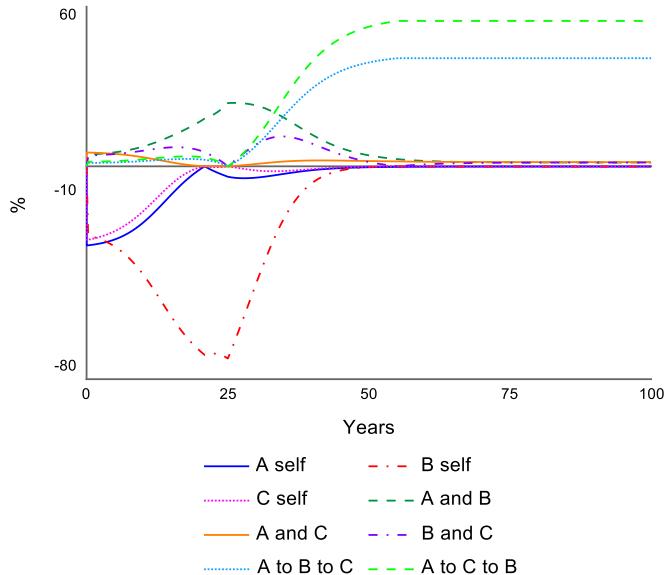


Figure 3: Relative loop scores for all loops in the arms race model

The paired interactions and self-corrections are important at the beginning. This is not surprising as we started the model quite far from a balanced trajectory. By about time 50 however, the model behavior is dominated by the two long loops.

For this model it is clear that we need to identify all of the feedback loops for analysis, as all of them (with the possible exception of the A and C interaction) are of consequence at some point in the simulation.

Once we have found the loops to analyze, the loop scores can be used to determine which loops are most explanatory. Until we find the loops, however, we can't apply the technique. As is clear from this example, we need to find things that will have different importance at different points in time, but we need to find everything that is ever important precisely to see which loops are driving behavior when.

The Composite Feedback Structure

When we look at a model and try to find feedback loops we do so by following links and flows from one variable to another. Assuming the model diagram is accurate, and we are patient and assiduous, this will define the universe of potential loops. This approach, however, includes loops that may never be active during the course of a simulation. For example, consider a very simple, if contrived, two stock model shown below in Figure 4:

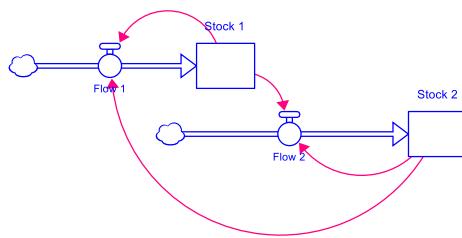


Figure 4: Simple model showing decoupled feedback loops.

Where

$$\text{Flow_1} = \text{IF } \text{Stock_2} > 50 \text{ THEN } \text{Stock_2}/\text{DT} \text{ ELSE } \text{Stock_1}/\text{DT}$$

And

$$\begin{aligned} \text{Flow_2} = & \text{ IF } \text{Stock_1} > 10 \text{ AND } \text{Stock_1} < 20 \text{ THEN } \text{Stock_1}/\text{DT} \\ & \text{ELSE } \text{Stock_2}/\text{DT} \end{aligned}$$

With both stocks starting at 1, until Stock_1 gets to 10 we just have the two minor loops running side by side. When Stock_1 is between 10 and 20 it drives Flow_2, but Flow_1 is determined only by Stock_1. Later, when Stock_2 finally reaches 50, it drives both flows. So what we have are two decoupled loops that alternatively drive the other flow. When that happens, there is only one feedback loop active as can be seen from in the Figure 5

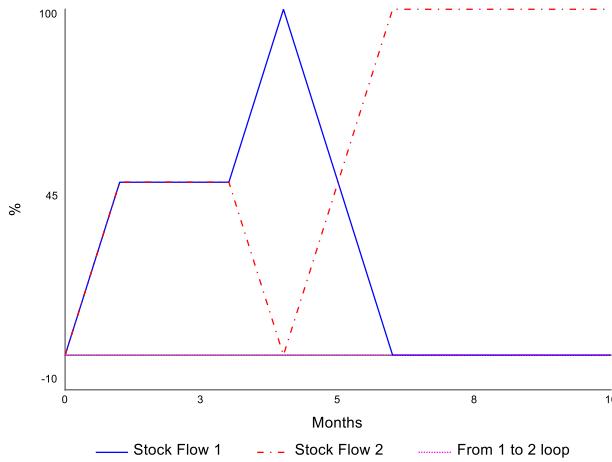


Figure 5: Behavior of the simple model shown in Figure 4

If, however, we were to measure the link scores for the components of the long loop they would look like this (Figure 6):

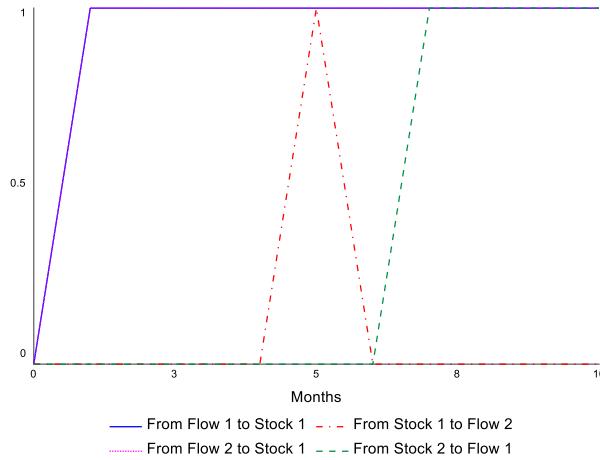


Figure 6: Link scores of all links from the model shown in Figure 4

The flow to stock connections are always 1, since there is only a single flow for each stock. All the link scores start at 0 because nothing has changed at the beginning of the simulation – a convention of the Loops that Matter scoring technique. The interesting things are the link from Stock_1 to Flow_2 is 1 at time 5, but otherwise 0 whereas the link from Stock_2 to Flow_1 is 0 until time 7 then 1 afterward (the dt in this model is 1).

In order to discover all of the feedback loops in a model, we need to include all links that are active at some point during the simulation. In this model, every link is active at some point in time, but not active at others. It is very common for links to become more and less active, though not necessarily completely inactive. As a consequence of this some loops are stronger at some times than others. To capture all the potential loops in this case, or the loops that could be strong at some point in time in the more general case, we need to pay attention to links that are strong at any time during the simulation. We do this by creating a composite link score that represents the contribution of a link over the course of the simulation. As long as the link is non-zero we will find all of the loops, and if we can choose the composite score well, it seems reasonable we will be able to identify the important loops.

One approach to creating the composite link scores to support loop discovery is to use the largest magnitude of the link score over all times. In this case, that would give us 1 for every link. Starting from either stock the connection from the stock's own flow, or the other stock's flow would be equally strong. Assuming we found all three feedback loops, they would all have a loop score of 1. Using this approach, the composite loop scores will always be as big, or bigger, than any actual loop score. This can be a problem for a big model, where the loop scores can become extremely large, and when using the largest link score magnitudes, the longer the loop the bigger the score. This approach is biased to finding longer loops, and also suffers from numeric problems because the size of the composite scores can easily exceed 1.0E300.

An alternative to approach to using the largest link score magnitude, is to take an average of the link score magnitude instead. In this case the flows would average to 1, while the link from Stock_1 to Flow_1 would average 0.5, from Stock_2 to Flow_2 0.9, from Stock_1 to Flow_2 0.1, and from Stock_2 to Flow_1 0.5. The strongest links out of either stock would be to its own flow. The minor loop for Stock_1 would get score 0.5, the minor loop for Stock_2 0.9 and the major loop 0.05. At first glance, this seems like a reasonably approximation since the minor loops are always 1 or 0 and the major loop is always 0. The problem is that with a small change

to the equations, the major loop could also have the same 1 or 0 behavior. Thus, with averages, it is generally true that the longer the loop, the lower the score. This approach is biased toward finding shorter loops, but does have nice numeric properties as the scores computed do not tend to grow abnormally large.

We spent a great deal of time trying to make both of these approaches to building a composite network converge, but ultimately the detected loops that ranked as most important, simply were not the most important loops when looking at the scores over the full simulation. Ultimately, we adopted the obvious solution, which is to do loop detection at every (or almost every) point in time. This solution does require many more discovery passes, but the strongest path algorithm used for each pass converges much more quickly than the searches that were required on the composite network making each pass much faster.

The composite feedback structure is used to perform an initial identification attempt that will find loops exhaustively if there are not too many (less than 1000). For this the actual link scores selected do not matter, but we use the maximum of all link scores in the computation. Many models have less than 1,000 loops and for these we don't need to go through the identification process described in the next sections.

The Strongest Path Technique

We set out to find an algorithm that was similar in spirit to the shortest path algorithm of Dijkstra (1959), derivatives of which form the bases for route selection in modern GPS systems. At its heart Dijkstra 's algorithm is simple. If the intent is to go from a to b, and while investigating possible routes we pass through c after 10 kilometers, then any other route that passes through c in more than 10 kilometers is not worth pursuing. This way the entire set of possible paths from a to b do not need to be explored.

The most direct analog to the shortest path search in the context of loop discovery would be to start from a and try to find paths to b, and instead of tracking the distance, which is additive, use the link score magnitude which is multiplicative.

Then, when we get to c we check if we have been to c before and if we have, whether the score we got on that path was bigger. If it was, we don't explore further.

Unfortunately, since we are trying to maximize instead of minimize, this approach does not lead to an exact solution. Specifically, if the path we took to c won't actually get us back to a, but instead will form a loop not involving a, then the score may be bigger than what we would get on the strongest path from a to a. This is easily seen in Figure 7:

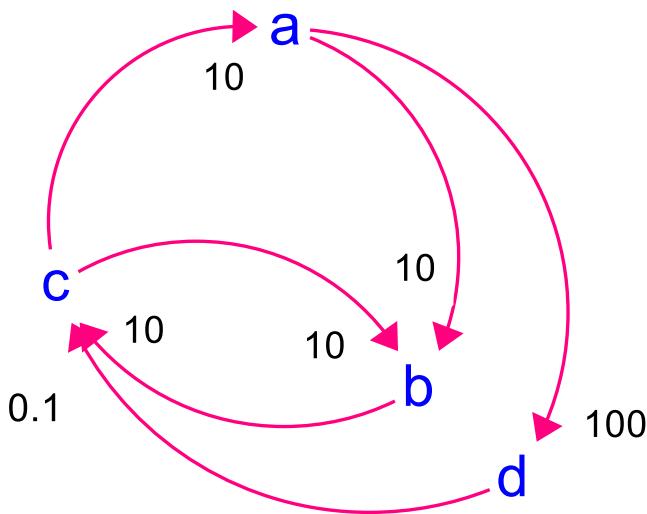


Figure 7: Demonstration of a failure case in directly applying Dijkstra's algorithm to loop finding

The numbers are the link scores. Starting from a and going to d gives us a score of 100, then to c gives us 10, then to b 100. Starting from a and going directly to b would give us a score of 10, which is less than 100 so we would not pursue that path. Thus, we would not find the loop a->b->c->a which has a score of 1000. Instead we would find a->d->c->a which has a score of 100.

In the above diagram, starting at b could give us the a->b->c->a loop, and starting at c might. It is however, possible (though messy) to create diagrams where starting anywhere would fail to find the strongest loop.

With those caveats in place, we can describe our loop discovery algorithm, this is repeated for each computational interval or a subset of those based on performance tuning.

1. Compute the link scores for every connection in the model. Some of these may be 0.
2. For every variable in the model sort the outbound links (places where the variable is used) by the link scores so that the first link has the biggest (absolute value) score.
4. For every stock in the model (all loops involve a stock) start the search:
 - b. Go through each outbound link in order, multiply by the link score of that link then test the variable the link points to using the current path score
 - v. If the variable is the starting variable, record the loop and the associated loop score (which is the path score from above)
 2. Need to check loop for uniqueness. If we already have it, then ignore it.
 - vi. If the variable is being visited already (a loop not involving the starting point) just return as the loop will be found starting from another stock.
 - vii. If the variable has been visited and has a higher path score from that visit just return.
 - viii. If the variable has not been visited, or has a lower path score, record the new path score and execute step a. above.

The algorithm has a similar computational burden to the shortest path algorithm (roughly proportional to the square of the number of variables). Sorting the edges by score helps to increase efficiency as it makes it more likely that the first visit to a

variable will be the one with the highest score. Most of the computational burden ends up being in the determination of uniqueness and the construction of the loop information for later processing.

Pseudo-code for the algorithm that is slightly more formal than the above outline is included in Appendix 1.

Completeness

It is possible to compare the loops found using the strongest path algorithm with an exhaustive search of loops for models with a relatively small (less than 100,000) set of unique loops. We can then compare the loops found using our approach to the full set of loops, in both cases sorting them by the average contribution of the loop to behavior over the course of the simulation.

For the Market Growth Model from Forrester (1968, as replicated in Morecroft (1983)) there are a total of 19 loops. When the loop discovery algorithm is run on this model it discovers all 19. A good result, so there is not much more to show.

For the service quality model of Oliva and Sterman (2001) there are a total of 104 loops in the main set of loops, though only 38 have a contribution of more than 0.01% to behavior based on the Loops that Matter contribution metrics. There are also 4 additional sets of loops which are smoothed quality measures that do not get used elsewhere in the model (a smooth is a single negative feedback loop). When loops are discovered using the strongest path algorithm a total of 76 loops are discovered with 28 having a 0.01% or greater contribution to behavior. Of the first 15 loops in the full set, the eighth is not present in the set generated by the strongest path algorithm. Table 1 compares the 8th loop in the full set to the 4th in both sets:

Table 1: Comparison of the 4th and 8th loops from the strongest path algorithm found in the Service Quality model

4 th Loop	8 th Loop
experience rate	experience rate
Experienced Personnel	Experienced Personnel
total labor	total labor
on office service capacity	on office service capacity
service capacity	service capacity
work pressure	work pressure
work intensity	work intensity
potential order fulfillment	potential order fulfillment
order fulfillment	order fulfillment
Service Backlog	Service Backlog
desired service capacity	desired service capacity
Change Desired Labor	Change Desired Labor
Desired Labor	Desired Labor
labor correction	labor correction
desired hiring	desired hiring
desired vacancies	
vacancies correction	
indicated labor order rate	indicated labor order rate
labor order rate	labor order rate
Vacancies	Vacancies
hiring rate	hiring rate
Rookies	Rookies

The two loops are identical through desired hiring, then the same again starting from indicated labor order rate. Indeed, this type of miss is quite typical. Large models

have many loops that are very similar, but not identical. The strongest path algorithm will often miss one of the shorter or longer loops with a similar set of elements.

Another interesting model is the economic cycles model from Mass (1975). This model has a total of 494 feedback loops (again there are some other set of loops in addition to the main set). When run through the strongest path algorithm there are 261 loops discovered. Out of the first 40 loops, only the 22nd and 40th are missing. Again the missing loops have much in common with the loops around them, though they are not quite as simply related as the service quality example.

Performance

Much of the work in the development of these algorithms used the Urban Dynamics model from Forrester (1969). When fully enumerated, this model has 43,722,744 loops. This is too big a number to compare the algorithm with the fully enumerated set. The determination of the loops is fairly fast, 10 to 20 seconds on an 8th generation intel core I7 processor. The strongest path algorithm discovers 20,172, though this is truncated to less than 200 for display, by only choosing those loops which describe at least 0.1% of the total behavior of the model when compared against all loops found at the termination of the search.

Even though we can't make definitive comparison against the full set of loops in the Urban Dynamics model, we can compare the results of different loop discovery approaches. We have experimented on this model with a number of different algorithms for loop discovery and saw the same pattern we highlighted for the service quality example. When an algorithm was tuned to run faster (and find fewer loops), the missing loops were quite similar, but somewhat longer or shorter, to those common to both tunings of the algorithm. Of course, with such a large set of loops, we can never know if we found the loop with the biggest score, but we have some confidence that we will find a loop that looks quite similar.

Mostly out of curiosity we also did a quick analysis of the World 3-03 model by Meadows et al (2004). This model has 330,574 loops when fully enumerated. Not

nearly as many as the Urban Dynamics model, but still too many to look at. The algorithm finds a total of 2,709 loops and this is truncated to 112 loops using the 0.1% contribution cutoff, with a number of both reinforcing and balancing loops being identified as important at different phases in the simulation. This computation takes about 4 seconds.

Paths not Taken

Jay Forrester often said that if we really want people to learn from us we should write about our failures. The development of the strongest path technique was the result of many attempts using different approaches that ultimately proved unsatisfactory.

The one thing that we got right from the beginning was the ordering of search nodes by their strongest connection. This seemed such an intuitive thing, that we never tested it, except by accident, and when we did the results were as expected. Ordering in this manner often speeds up search times by a factor of 3.

The initial attempts at getting algorithms to converge in reasonable time were focused on the remaining potential given the path that had been traversed, and were computed on the composite network. The first attempt at this was to trace the strongest links out of a node until we hit a terminal node or detected a loop and then use the biggest score found along that path as the potential of a node (variable) to add to a loop. Then, when the current score potential for the loop being discovered fell below a threshold (set based on loops already discovered) the search would be abandoned. Ultimately, the problem with this approach was that picking only the strongest outbound link seems at best modestly correlated with the real potential. The other problem is that the forward path might already include variables that have been visited.

To guard against counting the value of links already visited, we developed the concept of the total potential score remaining. This was computed as the product of the score of the strongest link out of every variable. As the variables are traversed looking for loops, the potential is decreased by the node being investigated, while the

current score is increased by the actual path out of the variable. The product of the current score and potential is thus monotonically decreasing. Picking an appropriate threshold at which to terminate the search (again based on loops already found) then provides a way to stop pursuing paths unlikely to uncover high scoring loops. This approach actually worked very well for composite scores that were based on the average of the scores over the simulation. The trouble with this approach was that the most powerful loops identified this way did not correlate at all with the most powerful loops as determined by looking at the total contribution of the loops to the dynamics. When we used the biggest values for the composite scores, the numbers were simply too large to provide reasonable cutoff thresholds.

A completely different approach we tried was to trim the feedback structure by getting rid of links. The expectation was that this would allow us to exhaustively search loops on a less connected model. The first attempt to do this removed links after they had been included in a requisite number of loops, based on the assumption that the strongest loops were found first. The second attempt to do this removed all the weak links. The difficulty with this approach was that the links being pulled out might be necessary to complete a loop with a high score, even though the link itself was not scored highly. This is not necessarily a fatal flaw in the approach, as we have seen that loops typically have siblings with a few more or fewer links. We were not, however, able to convince ourselves that this was all that was missing, nor were we sure of performance in models of different size.

We also explored the compaction of the model using only stock to stock connections. This has some theoretical appeal. Since all loops involve stocks, it seems like this would give us a smaller network to search and therefore improve performance. We also felt that the stock to stock connections could be selected for maximum strength and this would give more accurate results. As it turns out, computational speed improvements did not really occur. Ultimately, it is really the number of paths that exist and we don't change that much by eliminating non stocks, but instead just make more connections between the variables that do remain. Worse, when we do eliminate parallel paths, we have dropped potential feedback loops in the model.

These might be more easily interpretable than the ones selected, and including both of them might be informative.

Conclusions

We have demonstrated the importance of drawing from all loops when determining the loops that matter, and presented a technique for finding the most important set of loops using the strongest path algorithm that is practical and gives good results in large models. Though it is a heuristic, we feel that it gives a good balance of outcome and computational burden and has shown its utility for the models we have experimented with.

Ultimately, the value of the technique depends on its utility for providing understanding to model builders and consumers. As it is being embedded in a comprehensive framework for model analysis it is likely that users will find new needs relative to the discovery and display of loops.

Appendix I Algorithm Pseudo-Code

```

STACK - a vector of variables
TARGET - the stock currently under investigation
Function Check_outbound_uses(variable,score)
    If variable.visiting is true
        If variable = TARGET
            Add_loop_if_unique(STACK, variable)
        End if
        Return
    End if
    If score is less than variable.best_score
        Return
    End if
    Set variable.best_score = score
    Set variable.visiting = true
    Add variable to STACK
    For each link from variable
        Call Check_outbound_uses(link.variable, score*link.score)
    End for each
    Set variable.visiting = false
    Remove variable from STACK
End function
For each time in the run
    For each variable in the model
        For each link from variable
Set link.score from available data on link score (outside scope)
    End For each
    Set variable.best_score = 0
End For each
    For each stock in the model
        Set TARGET = stock
        Check_outbound_uses(stock, 1.0)
    End for
End for

```

Bibliography

- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269–271.
- Forrester, J. W. (1968). Market Growth as Influenced by Capital Investment. *Industrial Management Rev. (MIT)*, 9(2), 83-105.
- Forrester, J. W. (1969). Urban dynamics. Cambridge, Mass: M.I.T. Press.
- Güneralp, B. (2006). Towards coherent loop dominance analysis: progress in eigenvalue elasticity analysis. *System Dynamics Review*, 22(3), 263-289.
- Huang, J., Howley, E., and Duggan, J. (2012). Observations on the shortest independent loop set algorithm. *System Dynamics Review*, 28(3), 276-280.
- Kampmann CE. 2012. Feedback loop gains and system behaviour (1996). *System Dynamics Review* 28(4): 370–395.
- Mass, NJ. (1975) Economic Cycles: An Analysis of Underlying Causes, Cambridge, Massachusetts
- Meadows, DH., Randers, J., & Meadows, DL. (2004). The limits to growth: The 30-year update.
- Morecroft, JDW. (1983). System Dynamics: Portraying Bounded Rationality. *Omega*, 11(2), 131-142.
- Oliva, R. (2004). Model structure analysis through graph theory: partition heuristics and feedback structure decomposition. *System Dynamics Review*, 20(4): 313-336.
- Oliva R, Sterman JD. 2001. Cutting corners and working overtime: quality erosion in the service industry. *Management Science* 47(7): 894–914.
- Schoenberg, W., Davidsen, P., & Eberlein, R. (2019). Understanding model behavior using loops that matter. *arXiv preprint arXiv:1908.11434*.

Tarjan, R. (1973) Enumeration of the Elementary Circuits of a Directed Graph. *SIAM J. Comput.*, 2(3), 211–216.

10. Article #5: Seamlessly Integrating Loops That Matter into Model Development and Analysis

By: William Schoenberg and Robert Eberlein

Abstract

Understanding why models behave the way they do is critical to learning from them, and to conveying the insights they offer to a broad audience. The Loops that Matter methodology automatically shows which loops are dominating behavior at each point in time and generates simplified causal loop diagrams from a user adjustable set of important loops. This paper describes the challenges of implementing these tools into a fully functioning model development environment along with the solutions developed. The promise of the tools has, if anything, been amplified by the results of this implementation, and we give several examples of using the tools. For pedagogical models Loops that Matter can ease communication while speeding and deepening learning. For complex models the tools allow the extraction of realistic explanations of behavior in the form of animated simplified causal loop diagrams. For models with discrete and discontinuous elements, the bigger feedback picture is still easily discoverable. While there will doubtless be refinements and enhancement to the delivered tools, they represent a large step forward in our ability to understand models from conceptualization through delivery.

Introduction

Automatically discovering the origins of behavior within real-world models is a ‘holy grail’ pursuits for the field of system dynamics. Over the past 40 years, dozens of researchers have dedicated many hours and publications to this task (see for example: Graham, 1977; Forrester 1982; Eberlein, 1984; Davidsen, 1991; Mojtahedzadeh, 1996; Ford, 1999; Saleh, 2002; Mojtahedzadeh et al., 2004; Güneralp, 2006; Gonçalves, 2009; Saleh et al., 2010, Kampmann, 2012; Hayward and Boswell, 2014; Moxnes and Davidsen, 2016; Oliva, 2016; Sato, 2016; and Hayward and Roach, 2017). All of these approaches offer a wealth of benefits, some

unique to an approach, but many shared across approaches. To date, none of these approaches are in common use by a significant number of modelers, or students of modeling. Some of this is due to deficiencies, as each approach does have systematic limitations and blind spots (Kamppmann & Oliva, 2009), but more importantly all of them require the practitioner to do significant work. People with years of training, education, and experience, tend to rely on the understanding they develop as they build and work with models. Those with less experience are overwhelmed by the challenges of building good models and don't have the capacity to learn a totally new toolset at the same time.

All of this is prelude to saying that if tools for discovering the origins of behavior are simply part of the model development experience, they will, in turn, be part of how people understand the models they build and work with. For experienced practitioners they will both reinforce and challenge beliefs about what is truly driving behavior. For those with less experience they will offer pathways to understanding that are both discoverable and easy to communicate. That implementation, and the way it can be used for discovery and communication, is what we are presenting in this paper.

Starting with Version 2.0 of the Stella software products (including Professional, Architect, and eventually the online tools), the Loops that Matter method (Schoenberg et. al, 2019) are simply part of the modeling experience. It is turned on by a checkbox, and automatically reports information about loops and links that matter and allows visual exploration of the model structure crucial to driving behavior.

This paper deals with the implementation challenges and outcomes of the Loops that Matter method, not the method itself. Nonetheless, it is useful to provide some background on the different approaches there are to understanding behavior analytically, and where the method fits in. It shares many of the characteristics, and benefits of existing work, but has an advantage in computational simplicity. For completeness, we will then quickly define the link and loop scores that form the backbone of the method. This is followed by discussion of a number of

implementation challenges and opportunities, and finally a number of example applications to well known and pedagogical models.

Existing Approaches to Loop Dominance Analysis

Analytic methods for determining loop dominance have historically revolved around two approaches. The first is based on eigenvalue elasticity analysis (EEA), the second uses the pathway participation metric (PPM) and causal pathways.

EEA is used to determine what combination of behavior modes a given model structure produces (Saleh, 2002; Kampmann et al., 2006; Saleh et al., 2010; Oliva, 2016). EEA uses a linearization of the model and its associated eigenvalues and eigenvectors as the unit of analysis. It is the most encompassing method for structural analysis of models, but is limited in the set of the models it can analyze without modification due to its toolset (Oliva, 2016). Much work has been done to make tools supporting EEA easier to use, and usable on models with a wider variety of formulations, but it is fundamentally designed to work on continuously differentiable systems. Current software tools for performing EEA change model equations to meet that requirement which has measurable, impacts on simulation results (Oliva, 2016).

Current applications of EEA are based on an independent loopset as described by Kampmann (2012), or the unique shortest independent loopset as described by Oliva (2004). The independent loopset (and its variants) provides a reasonable number of loops to analyze and also maintains loop independence which many tools using the EEA approach rely upon. However, the use of an independent loopset (and its variants) can also limit the ability to understand classes of models where behavior is fundamentally driven by loops outside of these sets as described by Eberlein and Schoenberg (2020).

PPM, unlike EEA, does not use eigenvalues to describe model structure, instead it focuses on the links between variables, specifically tracing the causal pathways between stocks, and identifying the causal pathway most responsible for moving the stock in the direction of its net change (Mojtahedzadeh et al, 2004). PPM based

methods (including Hayward and Boswell's Loop Impact Method (2014)) study observed behavior modes relative to individual stocks in the model, rather than all stocks together. The PPM toolset is directly applicable to more models because it does not require continuously differentiable systems, but it has been criticized for its failure to clearly explain oscillatory behavior (Kampmann & Oliva, 2009)

The Loops that Matter Method

The Loops that Matter method uses computations based on actual variable values during simulation. As a consequence, it can be applied to models of any size or complexity without regard to continuity. It produces analyses which match those done by PPM and EEA when applied to the same models (Schoenberg et. al, 2019). The method, like PPM, builds upon observations of the way that experienced modelers perform analysis to determine the sources of observed behavior. The method does all calculations directly on the model equations, making it easier to understand the measurements of loop and link contributions to behavior. The method uses only values computed as part of the normal simulation of the model, making it applicable to models with discrete characteristics.

The Loops that Matter method produces three key metrics, the 'link score' for any link in the model, and 'loop score' as well as 'relative loop score' for any loop in the model. Together these metrics provide the necessary information to compute, visualize and animate the origins of behavior in system dynamics models. A full discussion of the Loops that Matter method is contained in Schoenberg et. al, 2019, but a brief summary of the method is provided below for convenience.

Summary of the Link Score

The link score (Equation 1) is best thought of as a contribution at one instant in time from an independent variable x to a dependent variable z or as a measure of the contribution and the polarity of the link from x to z . The link score for the link $x \rightarrow z$ is:

Equation 1: The discrete form for the link score equation which matches the implementation of the calculation and is computed each dt.

$$LS(x \rightarrow z) = \begin{cases} \left(\left| \frac{\Delta_x z}{\Delta z} \right| \cdot sign \left(\frac{\Delta_x z}{\Delta x} \right) \right), \\ 0, \quad \Delta z = 0 \text{ or } \Delta x = 0 \end{cases}$$

In Equation 1 Δz is the change in z from the previous time to the current time. Δx is the change in x over that interval. $\Delta_x z$ is the change in z with respect to x over that interval. From a computational perspective $\Delta_x z$ which is called the partial change in z with respect to x , is the amount z would have changed, conditionally, if x had changed the amount it did, but y had not changed. The first major term in Equation 1 represents the magnitude of the link score, the second is the link score polarity as defined by Richardson (1995).

Because stocks represent an integration process, they change over time as a result of the values of flows, not because of changes in those flow values. For this reason, the computation of link scores for links representing the effects of flows on stocks is different. A deeper discussion of the construction of this equation can be found in Schoenberg et., al. 2019. Assume the stock equation $s = \int(i - o)$ where s is the stock, i is the inflow, and o is the outflow. The link score in these cases is computed as seen in 14.

Equation 2: Link score for all links from flows to stocks (both inflows and outflows are covered)

$$\text{Inflow: } LS(i \rightarrow s) = \left(\left| \frac{i}{i - o} \right| * 1 \right) \quad \text{Outflow: } LS(o \rightarrow s) = \left(\left| \frac{o}{i - o} \right| * -1 \right)$$

One of the important attributes of link scores is that they can be multiplied together to give the effect along a path between an input and an output. This product is called the path score, and used to deal with hidden paths, such as those that arise from macros as discussed below.

Summary of the Loop Score and the Relative Loop Score

The loop score is computed by multiplying all link scores together for each link in a loop at a point in time. This is the same as the path score from a variable back to itself, because a loop is simply a closed path. The magnitude of the loop score represents the contribution of a loop at a time to changes in all model variables⁶. Loop scores can be very large in magnitude, so we typically use the relative loop score, a normalized value, computed so that the absolute value of all relative loop scores add to 100%. The relative loop score measures the percentage contribution a loop to the changes of all variables in the model¹ at each point in time.

Challenges to making Loops that Matter production ready

As should be clear from the preface, our goal in making Loops that Matter a part of Stella was to have something seamlessly integrated and, from the user's perspective, essentially free to use. It is a truism of software development that the easier you make a feature for users, the more the effort required to develop the feature. This is an expected part of all commercial software development. A great deal of effort went into making both the computational engine, and the user interface as efficient and effective as possible. There were, in addition, a number of obstacles to having the software simply work that are more conceptual. Some require extensions to, or at least new interpretation of, the Loops that Matter method. Some are decisions on how to best present information. All effect the results, as seen by the user, and so are worthy of discussion.

Macros

The first challenge to overcome was how to handle macros such as DELAY or SMOOTH which incorporate a complex hidden internal structure. Because of the hidden structure, links that appear to the practitioner as directly connected on the

⁶ Assuming all variables share the same feedback loops. For models with feedback loops that exist in isolation (either completely disconnected set of variables, or as inputs to, or outputs from, other parts of the model) the variables are broken into sets that share feedback loops, and the scores computed on each set.

diagram, are often times quite indirect which is demonstrated in the example below. This problem is especially complex because there may be multiple causal pathways with differing strengths and potentially even polarities between the input and output of what may appear to the practitioner to be a simple link. In addition, there may even be feedback loops within the macro equations themselves.

Take for example the DELAY3 macro pictured below in Figure 1. From the perspective of the practitioner there is a direct link between ‘input’ and ‘output using macro’, (with a second connection from ‘delay time’). When the full set of relationships underlying the macro is shown, however, we see a much less direct path from ‘input’ to ‘output using full structure’ even though they are identical computationally. Once the full structure of the macro is exposed it’s obvious that there are multiple distinct pathways from ‘delay time’ to the output, in fact there are six distinct causal pathways if we include the influences to the flows both directly, and through the upstream stocks. For the ‘input’, there is only a single pathway, but it goes through every stock in the chain. Therefore, what appears to be two simple direct links on the diagram is actually seven distinct causal pathways all but one (the connection of ‘delay time’ to ‘Flow 4’) involving intervening stocks. In addition, the structure itself contains three feedback loops; fortunately for this, and most the built in macros, the internal feedback loops do not cause any behavior by themselves.

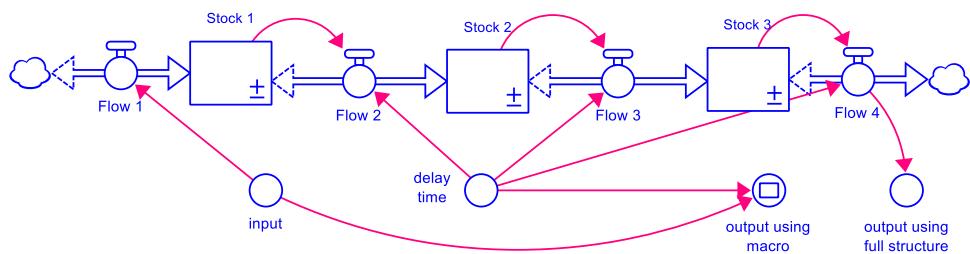


Figure 1: The structure of the DELAY3 macro, demonstrating the complex set of pathways between the arguments to the macro (input, and delay time) to the output.

The solution devised to this challenge is a simple heuristic applied at each calculation interval. The link score of a pathway which passes through a macro is the path score of the expanded pathway. If there are multiple pathways, we choose the path score

with the largest magnitude (positive or negative). The reasoning behind this definition is that it will maintain the integrity of the loop scores computed through the macros. If there is a single path through the macro the loop score will be exactly what it would have been if the macro had been expanded. If there are multiple paths (so multiple loops) the loop score will be the biggest of all the loops involving the macro. It is also possible, using this definition, to compute the link score as the simulation occurs with no post processing, supporting the definition of a PATHSCORE builtin function that can be used in the model. The link score for anything going into a macro is, thus, a composite. As a consequence, the structure represented by a link through a macro is not necessarily fixed throughout the course of the simulation run.

We considered, but ultimately rejected, an alternative way of dealing with macros that involved the determination of the dominant pathways through a macro by post-processing all of the path scores and picking the best one. This has the advantage that the structure within the macro is invariant, but would change loop scores relative to the case where the macro were expanded. A similar approach in which macros were given predefined pathways for link score computation was also rejected for similar reasons. Another approach considered was the expansion of all macros with the internal information directly exposed to the end user. This solution was rejected because it would be confusing to the practitioner and the macro variables would have to be given names based on usage which would be hard to read and be detrimental to the quality of simplified CLDs, which will be discussed later. In addition, expanding the macros would ignore one of the main reasons for using them, which is to prevent the clutter of unnecessary detail in visual representations.

The link score we use for macros (called the composite link score) does preserve overall loop scoring, but macros will still look different from their expanded versions. For example, if the DELAY3 macro is given a step input (and nothing else), the reported link score will always be 0. This is because the input is changing only at a single point in time, and the output is not changing at that time. Once the output starts to change, the input is no longer changing. Since the reported link score is the product of the internal link scores, one of which is always 0, the composite score is 0. This

makes macros somewhat inscrutable when they are not actually part of any feedback structure.

One other point worth noting on macros, is that the loops involving internal macro variables need to be trimmed of those internal variables before being reported. Similarly, loops internal to the macro (such as the first order drains in the DELAY3 example) are dropped altogether and not reported.

Discrete Variables and Stateful Functions

Stella includes a number of discrete elements (Conveyors, Queues and Ovens) as well as builtin functions like PREVIOUS that retain state values. We won't go into details here, as many of these have been dealt with on a case by case basis, but will just point out the principals by which we have dealt with them. Unlike macros, for which there is a rigorous solution to following paths, there are internal structures in the elements and functions that can't practically be exposed to complete link score computation. A conveyor, for example, can have thousands of individual elements waiting to be used at a later time. To compensate for this, we have approximated an instantaneous response to changing inputs based on the perfect mixing analog in traditional System Dynamics models.

Consider, for example, a conveyor. If the input changes nothing happens till much later in the simulation. This is distinct from a normal stock with a proportional outflow where a change in the stock value causes the outflow immediately changes. For the conveyor the change is not immediate, but eventual, although it still has the same basic character. So we treat conveyors as if the instantaneous response is the eventual response. This may cause some distortion in the time profile of the loop dominance, but it gets polarity and magnitude correct. It also, in practice, seems to work quite well as can be seen from one of the examples we discuss.

Too Many Loops

Feedback is pervasive, and some models have more feedback loops than can be dealt with in reasonable time by the software, let alone a practitioner. The solution to this

problem is to focus only on the loops that matter based on their relative loop scores. That solves the problem for the practitioner, but not for the computer in the cases where models have more than a few thousand loops. Since the number of potential loops is typically more than linearly proportional to the number of stocks in a model, it can get very large very fast (Kamppmann, 2012). How big it actually gets depends on the model, but a number of published models have too many loops to practically enumerate. As stated above, there are static analysis techniques for dealing with this problem, namely the independent loopset and the shortest independent loopset as used by EEA. Static analysis, unfortunately, does not allow the loops under consideration to change over the course of a simulation, which is exactly what is required as dominance shifts, and can therefore miss important feedback as demonstrated by Güneralp (2006) and Huang et. al, (2012). The strongest path algorithm described in Eberlein and Schoenberg (2020) solves this problem with a heuristic that finds the loops that matter at each point in time, with the resulting set of loops used to do further analysis.

The strongest path algorithm can change the loops that are identified based upon the parameterization of a simulation run. Just as the most important loops change with different parameters, so will the loops actually identified for large models. This is problematic if the intent is to study the impact of a loop across a wide set of input parameters, since the loop under study may not always even be identified. To address this problem, a LOOPSCORE builtin was devised which allows the practitioner to specify any arbitrary feedback loop, and compute its score over time. Inclusion of this guarantees that the loop will be reported on regardless of its importance based on the selected input space of the model. This also allows comparisons across runs to see how active a loop is under different scenarios. Loop scores are reported using the LOOOPSCORE builtin function but (because they reported as relative values) they are only available at the end of the simulation. It is, however, possible to compute the raw loop score using the PATHSCORE builtin previously described.

Visualization of the Loops that Matter

The final chapter of *Business Dynamics* (Sterman, 2000) makes clear its call for software that can perform “Automated identification of dominant loops and feedback structure”, while “Linking behavior to generative structure” and performing “Visualization of model behavior”. The LoopX tool developed by Schoenberg (2019) was the first of its kind to make major headway on these lofty aspirations. Using the link score, LoopX was able to create animated stock and flow diagrams where the connectors and flows change size and color relative to the magnitude and polarity of the link scores. Further still, LoopX demonstrated how to use the loop score in combination with the link score to quickly and dynamically simplify model presentation by automatically generating high quality animated causal loop diagrams, selecting only those variables and links which were absolutely necessary to explain the feedback behavior of the model at the requested level of complexity. That technology has been refined in the current implementation, and its scope expanded to encompass practitioner developed models with high levels of complexity.

The LoopX tool introduces two important terms, the link inclusion threshold and the loop inclusion threshold, both as a part of the CLD simplification process. The simplification process suggested by Schoenberg (2019) filters the list of variables in the simplified visualization by specifying which variables should be kept based upon the variation in the magnitude of the relative link score across the entire simulation period. Only variables with an inbound link (causal connection) with a relative link score that varies by at least the link inclusion threshold will be kept. The loop inclusion threshold keeps the stocks (and optionally flows) in every loop that explains, on average, at least the specified percent of model behavior. Since all loop scores are presented as relative values with magnitude adding to 100, this is straightforward to determine.

From an implementation perspective, the main problem with the methods laid out in the LoopX tool was that the variables to include would first be selected, then all pathways connecting those variables determined, and then the loops of the simplified

diagram detected. This approach has some computational shortcomings, as loop detection is not always easy to do, and some conceptual shortcomings as it is difficult to relate the simplified loops to the original loops and it is quite likely that unimportant loops will be included in the reported set of loops.

The implementation we developed uses the same metrics for the selection of variables to include, but then uses the original loops to create the simplified diagram. This way there is a two way mapping from the loops in the full diagram to the loops in the simplified diagram (though more than one loop in the original model may result in the same loop in the simplified diagram). This makes it possible to compute metrics such as the fraction of total model behavior explained by the simplified CLD as well as to attach scores to the simplified CLD relative to the original model. This also means that the connections to be included in the simplified diagram are the important ones. We do, however, perform a small amount of loop closure of the simplified diagram to improve the layout, and this can add some unimportant (though aesthetically pleasing) loops to the simplified CLD.

Because the source of all the feedback loops in the simplified CLD is known, each can be given a composite relative loop score. This is the sum of the relative loop scores from all full loops which reduce to that simplified loop. Since relative loop scores are normalized, and each full loop is only marked as being represented by a single simplified feedback loop, the sum of the composite relative loop scores for all loops in a simplified CLD represents the portion of the full behavior of the model explainable by the simplified CLD. This measure proxies the quality of the simplified CLD. Numbers closer to 100 mean that more of the model's behavior is being explained by that simplified CLD, closer to 0 mean that less behavior is represented in the presented simplified CLD.

A surprising outcome of deriving simplified loops from the full loops is that it is possible to produce simplified CLDs where the loops are disconnected. While at first glance this outcome may seem problematic, upon reflection it is representative of the feedback structure of the underlying model. Take for example Figure 2, where a

model is shown which has two strong minor feedback loops that are tied together by a weak major feedback loop. For this sample model, it makes sense that when the loop or link inclusion threshold are set high to achieve a simple CLD, the weak major loop (and its links) disappear, leaving two unconnected minor feedback loops as the primary drivers of model behavior.

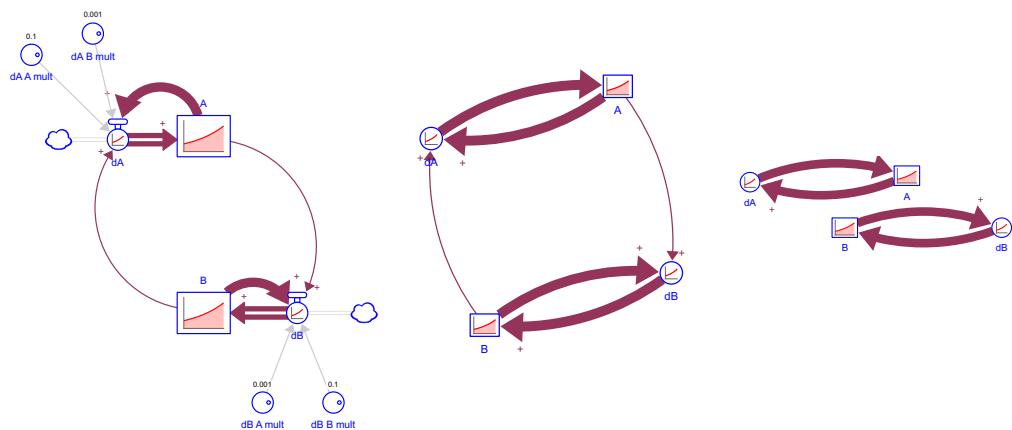


Figure 2: Demonstration of a weakly coupled system which when simplified turns into two disconnected systems. The far left is the Stock Flow Diagram for this system. The middle is a CLD showing all feedback relationships in the model. The far right shows the most important feedback relationships in the model.

Link Thickness and Polarity Markings

The second challenge in the visualization of simplified CLDs choosing the thickness and polarity for a simplified link. These links may represent several different causal pathways with different strengths, and potentially polarities. This is similar to the problem of representing composite link scores in macros, but has a different solution. In this case, we use the causal pathway that has the largest path score magnitude averaged across time. The reasoning behind this selection is that the simplified CLD is based on selections of links and loops using their average strength, so it is logical to use a consistent measure for the representation of the link. If a variable is included because it has a strong link into it, then the strength of that link should be presented.

Similarly, if a stock or flow is included because of a strong loop, the strength around that loop should be represented. This is a satisfying argument for links which represent causal pathways of the same polarity. For simplified links which have been over-abstracted, where they represent pathways of both reinforcing and balancing polarities, this solution produces simplified CLDs which may be misleading, and we have developed a measure to highlight that situation.

For all simplified links, a confidence value is generated using Equation 3 below, where r is the sum of the single highest magnitude instantaneous reinforcing pathway scores across the entire simulation and b is the sum of the single highest magnitude instantaneous balancing pathway scores across the entire simulation.

Equation 3: Equation for computing the confidence that a simplified link has a single and consistent polarity

$$\text{confidence} = \frac{|(r - |b|)|}{(r + |b|)}$$

This confidence value makes it very clear when a simplified link is representing two pathways of different polarities (it is 1 if either r or b is 0). A confidence value of 0.99 (or lower) was chosen as the cutoff point which is used to change the color representing the polarity of the simplified link to gray (to represent mixed polarity) which makes it abundantly clear that the simplified CLD is over-simplified. The gray color represents that the two variables are related, but that a key portion of the relationship is omitted preventing a proper polarity of the relationship from being displayed. Figure 3 demonstrates this problem in a simplified CLD representing the dynamics of the base case of Forrester's (1968) market growth model.

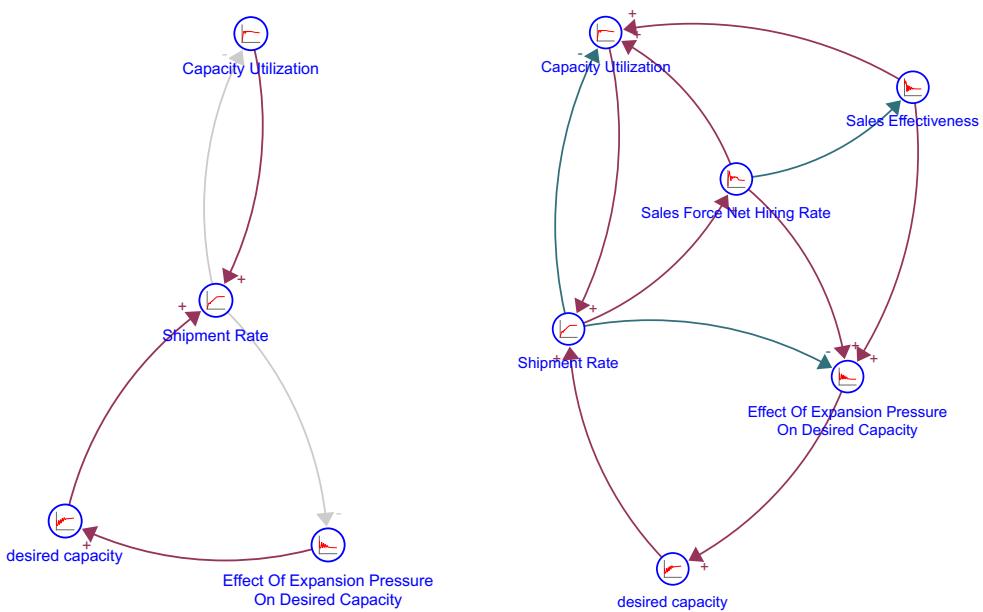


Figure 3: On the left is an oversimplified version of Forrester's 1968 market growth model with a gray link, which clearly communicates that the link does not have a determinable polarity. The simplified CLD on the right is after adjusting the link inclusion threshold down by one one-hundredth of a percent, it demonstrates how an over oversimplified link is expanded to its key constituent pathways

Loops with Unknown Polarity

Some models contain links, and therefore loops, that change polarity during a simulation. For example, in the yeast alcohol model examined by Schoenberg et. al (2019) there is a link from yeast concentration to growth that is at first positive, then negative. While such links are rare, only occurring in equations that compound multiple effects, they still occur with enough frequency that they have to be dealt with in a production ready system. The problem with these links is that they make interpreting the polarity of a loop across the full simulation time technically impossible because any loop including them most likely has expressed both positive and negative polarities at different points in the simulation. Attempting to discover the origins of behavior in a model where one of these links is along a key causal pathway, shared by many of the key loops in the model, is very difficult unless there is some way to classify the actual behavior of the loops these links are a part of. To

solve this problem a predominate polarity needs to be established if possible. Loops are labeled according to the following scheme, Rx, Bx, Rux, Bux, Ux, where x is the index of the loop of that type. R for reinforcing, B for balancing and U for unknown polarity. Ru means unknown polarity, predominantly reinforcing, and Bu means unknown polarity predominantly balancing. The Ru and Bu designations are assigned when the confidence value for a loop polarity is above .99, as calculated using Equation 3. This cutoff allows for a well-reasoned factual interpretation of full and simplified CLDs including these links, where the polarity changing nature of these links is not important over the course of the simulation.

An Additional Option to Simplify CLDs

The final improvement to the visualization of simplified CLDs is targeted at models with many stocks, and long loops. The original definition of the link inclusion and loop inclusion threshold state that anytime a stock is kept, so are its flows, regardless of the relative link scores of those flows. In large models with large numbers of stocks, especially where the feedback loops tend to be long, we get extra flows, which do not necessarily add anything to understanding. Therefore, a third simplification parameter, a boolean, was introduced allowing the user to control if flows are automatically kept if a stock is kept.

Demonstration of the Power of the Approach

Building intuitive understanding of the patterns of structural dominance in practitioner-built models is straightforward using the tools that are now part of Stella. To demonstrate this, we show three examples. This first is pedagogical, and shows how loop identification and highlighting on the stock and flow diagram can help students understand simple models. The second example is from a relatively complex model that shows the ability of the approach to give insight into larger models. The final example is from a model using a conveyor and a stock with a non-negativity constraint and demonstrates the robustness of the approach to nonlinearity and special conditions.

Pedagogy

All of the loop dominance work, including Loops that Matter, was inspired by the desire to understand things that weren't simply obvious. Simply obvious, however, means something very different to an individual who has never built a model than an experienced System Dynamics modeler. After implementing these tools in Stella, and then just using the software in day to day activities with the tools (by default) turned on, it became clear they could help with even very little things. Things that are obvious, but take too many words to explain to those for who would disagree with that claim.

Start with the simplest population model – Population, births and a birth rate as shown in Figure 4. There is one reinforcing loop, it accounts for all the behavior, and the model displays exponential growth. Student can certainly play with the growth rate at this point to get a feel for exponential growth.

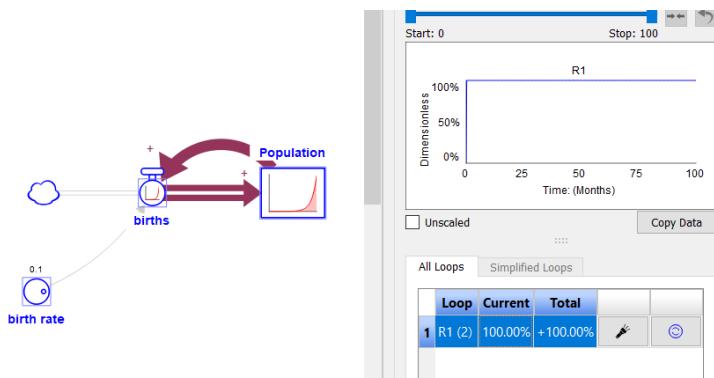


Figure 4: A simple population model with only Births.

Now add deaths simply assuming an average lifetime of 20. The resulting model is shown in Figure 5 and still exhibits exponential growth, but at a much slower rate. The reinforcing loop has a 67% contribution, the balancing loop 35%. If the students shorten the lifetime they will see the percentages move toward 50/-50. If they go less than 10 the graph will show decline and the balancing loop will be dominant.

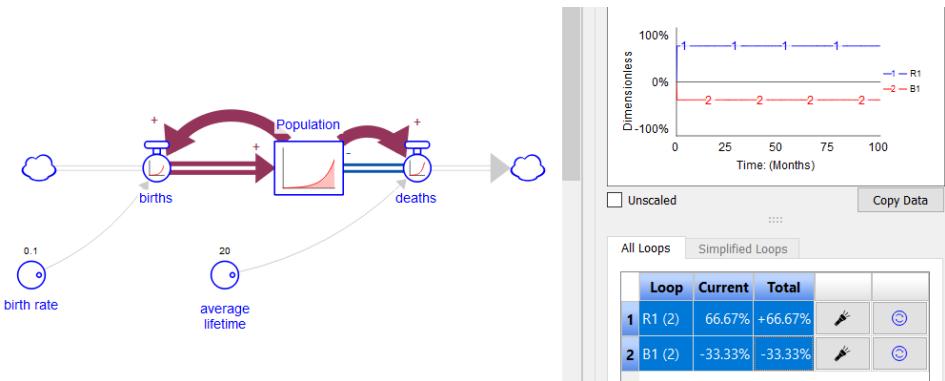


Figure 5: Population model showing both births and deaths

Clicking on the for B1 is also a good way to highlight a loop that does not look like a loop on a stock and flow diagram as shown in Figure 6. This kind of loop, where the outflow from the stock does not have an arrowhead, is harder to students to see.

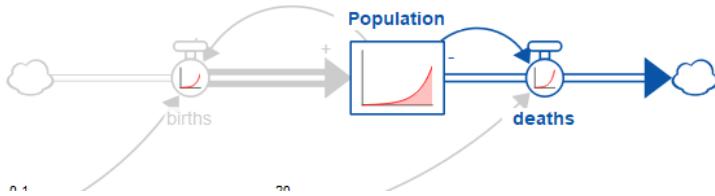


Figure 6 Highlighting the negative loop involving 'deaths'.

If the students set ‘average lifetime’ to exactly 10, then nothing is changing and there are no loops reported. This is a learning moment, both for the students and the teacher. On the one hand the lack of any loops being reported is simply an artifact of the way the Loops that Matter method works. On the other hand, the lack of change is the result of two opposing loops just happening to match. That is either cosmic coincidence, or there is a reason for it, and this allows the introduction of the concept of carrying capacity as shown in Figure 7

This dramatically shows the difference between a fragile equilibrium, and one that is the result of shifting loop dominance. At the end there is one reinforcing loop with a

score of 50%, and two balancing loops with scores that add to -50%. The capacity constraint loop (B1) is at first inactive, but then comes to be the bigger of the two balancing loops. The question of whether B1 is changing behavior, or is simply making B2 strong enough to balance R1 should make for good class discussion, though it seems unlikely the class will reach a conclusion.

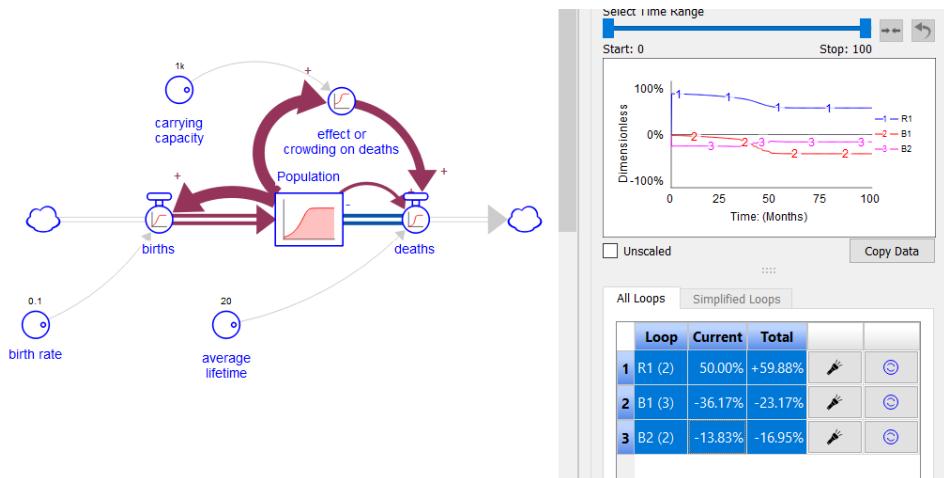


Figure 7: Population model with the addition of 'carrying capacity'

This is an extremely simple example, and likely one that many of the readers have used. It demonstrates the extra cues that the Loops that Matter techniques add to the conversation. This has the potential to make the discussion faster, more informative, and better remembered.

Understanding Economic Cycles

Mass' 1975 Economic Cycles model provides a good example of a practitioner developed model with a high level of complexity. The model contains 163 variables with 17 stocks, producing 494 feedback loops⁷. Figure 8 shows a machine generated CLD of this model whose 9 simplified feedback loops represent the combined effects of 21 full feedback loops. The 9 simplified feedback loops of Figure 4 explain 59.7%

⁷ There are 4 other two variable, stock/flow balancing feedback loops, each in their own cycle partition which do not affect the behavior of the model.

of the behavior for this model across its entire simulation. Of the 40.3% of behavior not represented in Figure 8, 31.2% of it comes from 469 relatively unimportant feedback loops which individually produce less than 2% of the cumulative model behavior. This leaves 4 remaining feedback loops which are not included. Those 4 loops contribute the remaining 8.9% of cumulative behavior. Those 4 have not been included because they consist of two sets of paired feedback loops (one balancing and one reinforcing) which perfectly cancel each other at all time points, therefore making all 4 of those loops irrelevant to the observed behavior of the model. This overview confirms that Figure 8 is a reasonable simplification of the important dynamics within the Economic Cycles model.

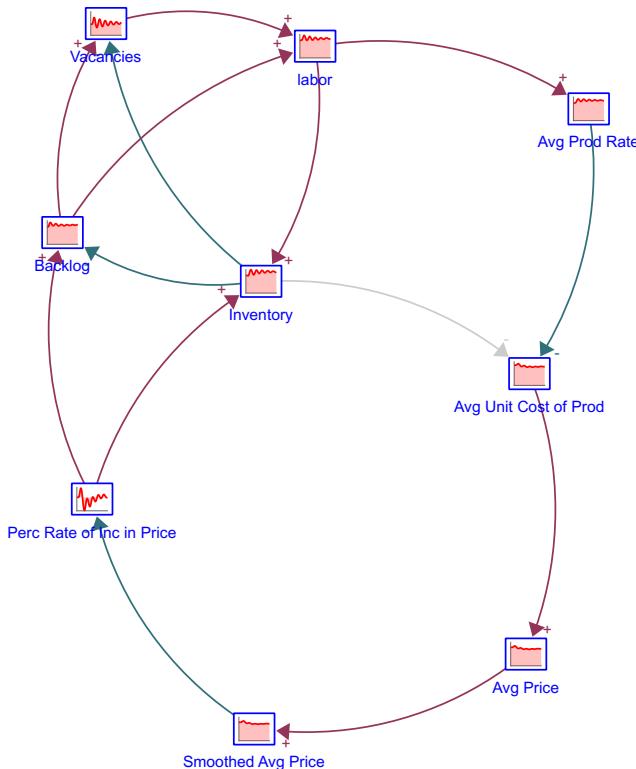


Figure 8: Automatically generated simplified CLD of Mass' 1975 Economic Cycles model with a link inclusion threshold of over 100%, a loop inclusion threshold of 2.4% and without automatically keeping flows with stocks.

Table 1: List of all feedback loops shown in Figure 4 with cumulative composite link scores. Total loops aggregated describes the number of full feedback loops which are represented by the simplified feedback loop. The total behavior explained by Figure 4 is 59.7% of the total model behavior. B5, B6, R2, and U1 are artifact loops brought forth by the specific combination of feedback loops selected. They were not directly selected for.

Loop	Total Contrib.	Total Loops Aggregated	Links Included
B1	38.12%	1	Vacancies→labor→Inventory→Backlog
B2	6.60%	1	labor→Inventory→Backlog
B3	5.11%	3	Vacancies→labor→Avg Prod Rate→Avg Unit Cost of Prod→Avg Price→Smoothed Avg Price→Perc Rate of Inc in Price→Inventory
R1	5.11%	3	Vacancies→labor→Avg Prod Rate→Avg Unit Cost of Prod→Avg Price→Smoothed Avg Price→Perc Rate of Inc in Price→Backlog
B4	4.15%	1	Vacancies→labor→Inventory→Avg Unit Cost of Prod→Avg Price→Smoothed Avg Price→Perc Rate of Inc in Price→Backlog
B5	0.37%	1	Vacancies→labor→Inventory
B6	0.13%	1	Avg Unit Cost of Prod→Avg Price→Smoothed Avg Price→Perc Rate of Inc in Price→Backlog→labor→Inventory
R2	0.10	3	Avg Unit Cost of Prod→Avg Price→Smoothed Avg Price→Perc Rate of Inc in Price→Backlog→labor→Avg Prod Rate
U1	0.02	7	Avg Unit Cost of Prod→Avg Price→Smoothed Avg Price→Perc Rate of Inc in Price→Inventory

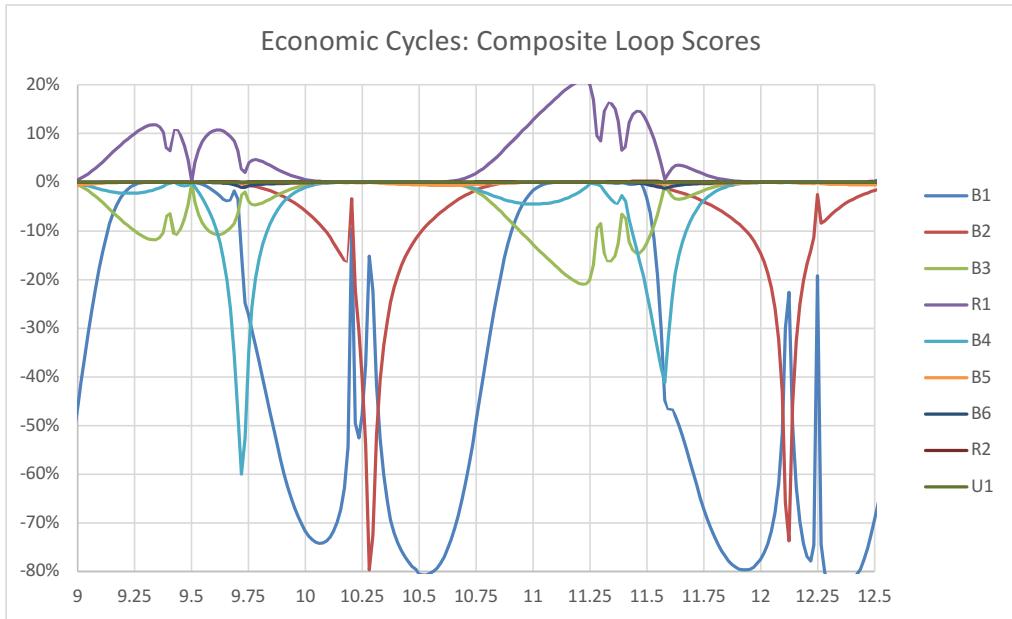


Figure 9: Composite relative loop scores demonstrating the overtime impact of each of the simplified loops shown in Figure 8 and Table 1. A full explanation of this figure can be seen below.

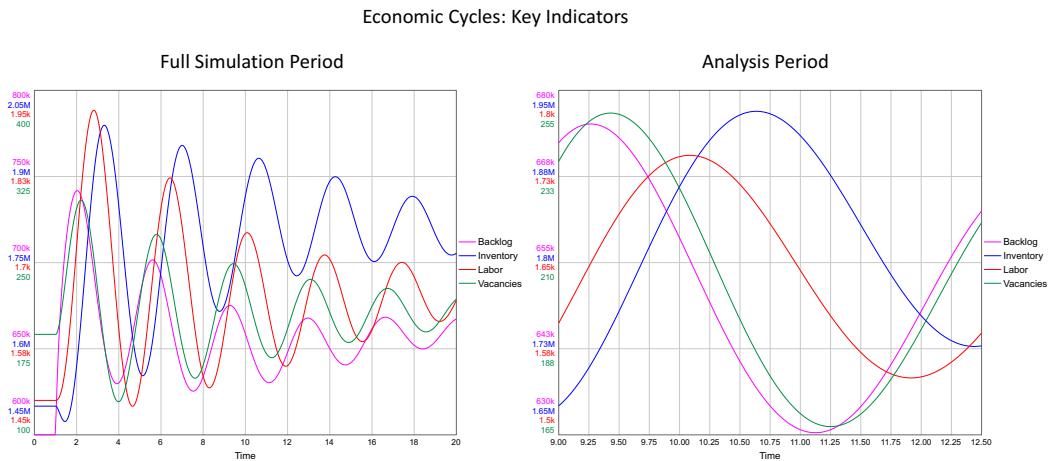


Figure 10: Plot of the key indicator stocks from Mass's 1975 Economic Cycles model showing the damped oscillation in economic systems.

Figure 9 demonstrates from a feedback perspective the dominant relationships at each point in time during a complete single cycle of the model. Figure 10 shows the patterns of behavior for the key stocks in this model over the entire simulation time

period, and the chosen period of analysis of Figure 9. The time period of 9 to 12.5 was chosen as the basis for the analysis because it represents one complete cycle of all the stocks in the model. Specifically, this time period tracks one complete wave in Inventory from trough to trough. Within this time period the loop dominance pattern of the Economic Cycle model repeats twice, beginning at time 9 and again at time 10.6. The first walk through this pattern represents the growth of labor, the second, the decline of labor.

The loop dominance pattern within the Economic Cycles model starts with B1 being nearly completely dominant. B1 describes the major drivers of the hiring of labor based upon vacancies created by the backlog driven directly by inventory due to the changes in labor. As the direct hiring process (B1) wanes, a series of loops (R1 and B3) which are both long delays also related to hiring of labor through the changes in vacancies become active. These loops are an example of the perfect destructive interference of feedback loops and cancel each other out perfectly. R1 and B3 cancel each other out because their only difference is the specific route, they take to influencing vacancies from the perceived rate of increase in price. Those two differing pathways taken by R1 and R3 have the same magnitude contribution to vacancies but have opposite signs. R1 represents the effect of the perceived rate of increase in price on backlog which reinforces vacancies, while B3 represents the effect of the same perceived rate of increase in price on inventory which balances vacancies. During this same time period, two other pairs of perfectly destructive feedback loops are also active but are not pictured in Figure 8.

The next major change in the feedback loop dominance of the Economic Cycles model occurs when B4 becomes important. B4, like B1 describes the changes in labor due to vacancies, but it does so through a long delay from the perception of price, due to changes in inventory affecting unit costs, rather than directly from changes in inventory to changes in backlog. After the effect of the delayed price adjustment on vacancies (B4), the direct hiring process becomes dominant again. What is different during this period is that B2 is now active, although not yet dominant. B2 represents the direct effects of the changes in backlog on changes in

labor through termination. Next B2 becomes the dominant loop, driving the changes in labor through termination at the inflection points in either the growth or decline of labor. B2 then yields again to B1 which starts the cycle of shifting feedback loop dominance anew. Every other progression through this cycle of feedback loop dominance changes whether labor is growing or shrinking (as B1 is an oscillatory loop), and this progression continues over the entire course of the simulation period. Even as the oscillations dampen, this same progression of feedback loop dominance is still happening.

This cogent explanation for Mass' Economic Cycles model demonstrates the power of the toolset to dramatically simplify the process of model understanding, while providing objective clarity on the causes of behavior in complex systems.

Demonstrating the application to a discrete system

Another interesting case is in a much simpler, discrete model, which is ideal for demonstrating the power of Loops that Matter to explain the origins of behavior in discrete systems. Pictured below in Figure 11 is a simple workforce model which uses a conveyor (which can be thought of as a pipeline delay) to model the training process by which apprentices are turned into workers. This system also uses a non-negative stock on purpose to limit the number of employees leaving the system. The equations for this model appear in Table 2.

Table 2: Equations for the sample workforce system. The two cases analyzed are exactly the same except case 1 used a time to adjust of 5, case 2 used a time to adjust of 2.

$Apprentices = CONVEYOR(hiring -$	People
<i>finishing training, training time)</i>	
$Workers = NONNEGATIVE(finishing$	People
<i>training - leaving)</i>	
$hiring = adjustment + leaving$	People/Time
$leaving = 100 + STEP(50, 5)$	People/Time
$finishing training = f(Apprentices)$	People/Time
$adjustment = (target workers -$	People/Time
<i>workers)/time to adjust</i>	
$training time = 5$	Time
$target workers = 500$	People
$Initial Apprentices = 5 * hiring$	People
$Initial Workers = target workers$	People
$time to adjust = 5, 2$	Time

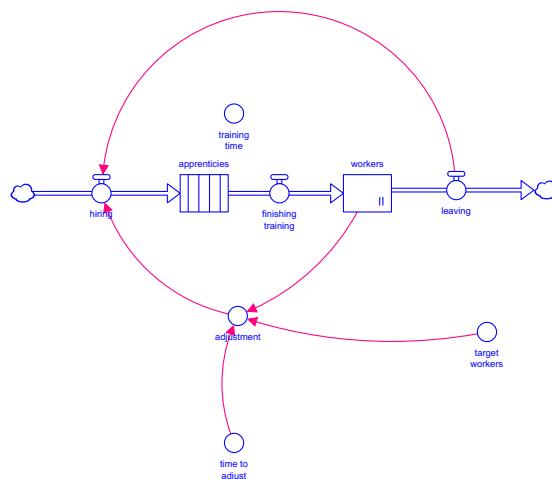


Figure 11: Sample workforce training model containing discrete elements, and non-negative stocks

The simplified CLD seen in Figure 12 highlights the ability of the Loops that Matter method to identify the unique causes of behavior in two different parameterizations of this system highlighting the complexity which is typically hidden in the SFDs of discrete systems. First, Figure 12 shows how Loops that Matter is able to identify the hidden feedback loop (from the perspective of the SFD) between Apprentices and finishing training. It shows how the conveyor is directly affecting its own output, and how changes in the output cause changes to the amount of material in the conveyor. When the time to adjust is equal to 2, the non-negative stock becomes active and constrains the outflow ‘leaving’, and those dynamics are captured in the CLD on the right in Figure 12. In addition, because ‘leaving’ now changes, and therefore it changes the number of people hired, it shows up as a concept unto itself in the right CLD. As a matter of fact, the feedback complexity of the model structure changes with its parameterization, the system on the left only has two balancing loops, the system on the right has the same two balancing loops, but it also contains an additional balancing loop (between leaving and workers), and an additional reinforcing loop (across the entirety of the mainchain to adjust hiring without passing through the adjustment variable). These two CLDs in Figure 8 demonstrate that Loops that Matter is fully capable of performing analyses on discrete systems without losing any of its ability to quickly generate insight.

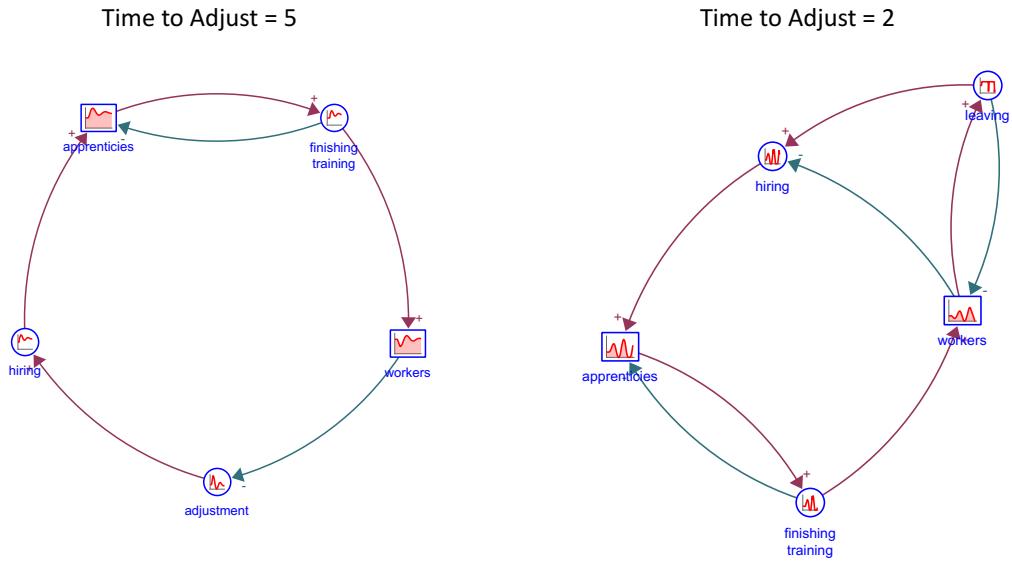


Figure 12: Causal loop diagrams for the sample workforce training model containing discrete elements under two different parameterizations. Notice how Loops that Matter is able to pick up on the additional feedback loops exposed by the changes in behavior of the discrete system based upon the parameterization of the model. These simplified CLDs communicate more of the feedback dynamics of the system than the SFD pictured in Figure 11.

Conclusions

The new Stella family of products put into the hands of all system dynamics practitioners the automated tools they need to quickly understand the origins of their model's behavior. It makes real the dream, of an objective, correct, useful and easily understandable analysis of the origins of model behavior in any system dynamics models. This paper has demonstrated the utility of the Loops that Matter method, and its visualization capabilities, including animated stock and flow diagrams, animated automatically generated causal loop diagrams, and automated simplification of CLD structure, on large and dynamically complex models such as Mass's Economic Cycles, as well as discrete feedback systems. This research has led us to the point where the only limitations of the Loops that Matter method are its inability to report on unobserved behavioral modes, and its inability to report on loop dominance during equilibrium. Neither of these limitations ought to prevent its acceptance and usage by the field of system dynamics and this should dramatically increase the quality of

the understanding of why models produce behavior, while increasing the speed at which these insights are discovered. The work presented here while, designed to fulfill the needs of the system dynamics community, is also directly applicable in a multitude of other fields such as data science, machine learning, and automated intelligence. Future research in these areas will help to broaden the appeal of structural dominance analysis to the wider universe of scientists cutting across fields and disciplines.

References

- Eberlein RL. 1984. Simplifying dynamic models by retaining selected behavior modes. PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Eberlein, R., & Schoenberg, W. (2020). Finding the Loops that Matter. *arXiv preprint arXiv:2006.08425*.
- Davidsen P. 1991. The Structure-Behavior Graph. The System Dynamics Group, MIT, Cambridge, MA.
- Ford D. 1999. A behavioral approach to feedback loop dominance analysis. *System Dynamics Review* **15**(1): 3–36.
- Forrester, J. W. (1968). *Market growth as influenced by capital investment*. Industrial Management Review.
- Forrester N. 1982. A Dynamic Synthesis of Basic Macroeconomic Theory: Implications for Stabilization Policy Analysis. PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Graham AK. 1977. Principles of the Relationship Between Structure and Behavior of Dynamic Systems. PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Gonçalves P. 2009. Behavior modes, pathways and overall trajectories: eigenvector and eigenvalue analysis of dynamic systems. *System Dynamics Review* **25**(1): 35–62.
- Güneralp, B. 2006. Towards coherent loop dominance analysis: progress in eigenvalue elasticity analysis. *System Dynamics Review*, **22**(3), 263-289.
- Hayward J, Boswell GP. 2014. Model behaviour and the concept of loop impact: a practical method. *System Dynamics Review* **30**(1): 29–57.^[L]_[SEP]
- Hayward, J, and Roach PA. 2017. Newton's laws as an interpretive framework in system dynamics. *System Dynamics Review* **33**(3-4): 183-218.

- Huang, J, Howley, E, & Duggan, J. (2012). Observations on the shortest independent loop set algorithm. *System Dynamics Review*, **28**(3), 276-280.
- Kampmann CE. 2012. Feedback loop gains and system behaviour (1996). *System Dynamics Review* **28**(4): 370–395.^{[[1](#)]}
- Kampmann CE, Oliva R. 2006: Loop eigenvalue elasticity analysis: Three case studies. *System Dynamics Review*. **22**(2): 141-162.
- Kampmann CE, Oliva R. 2009. Analytical methods for structural dominance analysis in system dynamics. In Meyers RA (ed.) *Encyclopedia of Complexity and Systems Science* Springer, New York NY, 8948–8967.
- Mass, NJ. 1975. Economic Cycles: An Analysis of Underlying Causes, Wright. Cambridge Massachusetts.
- Mojtahedzadeh MT. 1996. Structural Analysis of the URBAN1 model. University at Albany, SUNY, NY 12222, USA.
- Mojtahedzadeh M, Andersen D, and Richardson GP. 2004. Using Digest to implement the pathway participation method for detecting influential system structure. *System Dynamics Review* **20**(1): 1-20.
- Moxnes, E, and Davidsen, PI. 2016. Intuitive understanding of steady-state and transient behaviors. *System Dynamics Review* **32**(2): 128-153.
- Oliva, R. 2004. Model structure analysis through graph theory: partition heuristics and feedback structure decomposition. *System Dynamics Review*, **20**(4): 313-336.
- Oliva R. 2016. Structural dominance analysis of large and stochastic models. *System Dynamics Review* **32**(1): 26-51.
- Richardson GP. 1995. Loop polarity, loop dominance, and the concept of dominant polarity. *System Dynamics Review* **11**(1): 67–88.

- Sterman JD. 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Irwin/McGraw-Hill, Boston.
- Saleh MM. 2002. The Characterization of Model Behavior and its Causal Foundation. PhD Thesis, University of Bergen, Bergen, Norway.
- Saleh M, Oliva R, Kampmann CE, Davidsen P. 2010. A comprehensive analytical approach for policy analysis of system dynamics models. *European Journal of Operational Research* **203**(3): 673-683.
- Sato, JB. (2016). State space analysis of dominant structures in dynamic social systems. PhD Thesis, University of Washington, St. Louis, MO.
- Schoenberg, W., Davidsen, P., & Eberlein, R. (2019). Understanding model behavior using loops that matter. *arXiv preprint arXiv:1908.11434*.
- Schoenberg, W. (2019). LoopX: Visualizing and understanding the origins of dynamic model behavior. *arXiv preprint arXiv:1909.01138*.



Graphic design: Communication Division, UB / Print: Skjøres Kommunikasjon AS



uib.no

ISBN: 9788230850848 (print)
9788230866122 (PDF)