ESTIMATING CAUSAL RELATIONS OF DYNAMIC MODELS FROM REAL-LIFE DATA

by

Nefel Tellioğlu

B.S., Industrial Engineering, Boğaziçi University, 2016

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2019

ACKNOWLEDGEMENTS

First of all, I want to express my gratitude to Prof. Yaman Barlas for his endless guidance and support. It has always been a privilege to be a student of such an excellent professor. I am also grateful for every conversation we had; his passion on learning and sharing have helped me gain an invaluable perspective.

I would also like to thank Assoc. Prof. Gönenç Yücel for all his help in this journey. He generously gave many valuable advices on my thesis.

As members of my thesis committee, I would like to thank Assoc. Prof. Hakan Yaşarcan and Assist. Prof. Uzay Çetin for their valuable comments.

I would like to thank my lifelong friend, Gizem, for always being there with her sharp intelligence and her kind actions. I feel very lucky that there is someone whom I have shared and will share every aspect of my life. In addition, I want to thank Oylum and Pınar for always being there whenever I needed them.

SESDYN has always been a precious part of this journey. I thank İpek for being a great study buddy and friend. I also thank Feyyaz, Gizem, and Cansu who have made my last year a lot easier than I thought.

I want to thank my beloved family whose support and contributions I feel in every aspect of my life.

Last but not least, I want to express my gratitude to Baran for always being there with his endless support. He made my journey a lot cheerful with his sense of humor, encouragement, and love.

ABSTRACT

ESTIMATING CAUSAL RELATIONS OF DYNAMIC MODELS FROM REAL-LIFE DATA

In System Dynamics method, one tries to use data as much as possible, since model construction time and subjectivity can be reduced by the data analysis. In this research, our focus is the use of data analysis in (1) determining the polarity of causal effects and (2) discovering stock variables in a model. For determining the polarity of causal effects, we propose an algorithm, *discoverpolarity*, which is tested with seven data-sets. Then, the results are compared with Spearman's correlation analysis. The results show that *discoverpolarity* outperforms correlation analysis and is capable of obtaining useful and meaningful results when the input variables are properly selected and data-set comprises enough representative points in the causal domain. However, when the data only consists of all perfectly correlated data points, *discoverpolarity* may return multiple possible polarities instead of a unique solution. In addition, the modeler must determine the proper threshold values used in the algorithm. In further research, we plan to make *discoverpolarity* more robust to the input parameters. After enough tests with synthetic data, the algorithm must be tested with real data before it can be used in real-life modeling. Finally, the mathematical forms of the causal formulations can be estimated in further research, by extending the proposed algorithm. For the second thesis purpose, discovering stock variables, *curvefitting* algorithm is created and simulation-generated 'synthetic' data is analyzed in this algorithm to be able to evaluate the validity of the results obtained. The method is applied to three cases. We conclude that only in certain conditions, the algorithm may discover correct stock variables. In further research, we aim to categorize the monotonic relations where algorithm can find the correct stocks. In addition, we plan to focus on extending the *curvefitting* algorithm so that it can also analyze cases with multiple cause variables.

ÖZET

GERÇEK VERİLER KULLANILARAK DİNAMİK MODELLERDE NEDEN-SONUÇ İLİŞKİLERİNİN ÇIKARIMI

Sistem dinamiği yönteminde model kurma zamanını ve modelin öznelliğini azaltacağı için veri analizi olabildiğince uygulanmaya çalışılmaktadır. Bu çalışmada, modellerdeki (1) nedensel etkilerin işaretlerinin ve (2) stok değişkenlerinin belirlenmesi incelenmiştir. Nedensel etkilerin işaretlerinin belirlenmesi için *discoverpolarity* adlı bir algoritma oluşturulmuş ve yedi veri kümesi ile test edilmiştir. Sonuçlar Spearman'ın korelasyon analizi ile karşılaştırılmıştır. Algoritmadaki katsayılar uygun bir şekilde seçildiğinde ve veri kümesi yeterince zengin olduğunda, *discoverpolarity* korelasyon analizinden daha iyi sonuçlar bulmuş ve anlamlı ve pratik sonuçlar döndürmüştür. Ancak, veri kümesi tamamen benzer-ilintili verilerden oluştuğunda, *discoverpolarity* birden fazla sonuç döndürmektedir. Buna ek olarak, model kurucu algoritma için uygun eşik değerlerine kendisi karar vermelidir. İleriki çalışmalarda, algoritmayı girdi değişkenlerine ve varsayımlarına daha az hassas hale getirmeyi amaçlamaktayız. Sentetik veri ile yeterince analiz yaptıktan sonra gerçek verilerle algoritmayı çalıştırmayı hedeflemekteyiz. Son olarak, algoritmayı geliştirerek nedensel ilişkilerin matematiksel şekillerini de bulmayı planlamaktayız. Tezin ikinci, stok değişkenlerinin belirlenmesi amacına yönelik ise, *curvefitting* adlı algoritma oluşturulduktan sonra sonuçlarını doğrulamak için sentetik veri ile test edilmiştir. Bu yöntem üç farklı örnekte uygulanmış ve sınırlı koşullarda *curvefitting*'in doğru stokları bulduğu gözlemlenmiştir. İleriki çalışmalarda, hangi monoton ilişkilerde algoritmanın sonuç döndürdüğünü kategorize etmeyi planlamaktayız. Ek olarak, algoritmayı geliştirerek çoklu sebep değişkenleriyle de analiz yapabilmeyi hedeflemekteyiz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

CLD	Causal Loop Diagram
GAM	Generalized Additive Models
GLM	General Linear Models
MAPE	Mean Absolute Percentage Error
RMSE	Root Mean Square Error
RT	Room Temperature
SD	System Dynamics
SEM	Structural Equation Modeling
SFD	Stock Flow Diagram
TC	Temperature Change
X	Cause Variable
Y	Effect Variable

1. INTRODUCTION

System dynamics (SD), as a complex system methodology, aims to address non-linear dynamic feedback problems and analyze dynamic behaviours of complex systems. The focus of this methodology is to study how relations among parts of a system results in a collective behaviour of that system.

The systems anaylsed in the SD method refer to any structure which has circular, time delayed, and non-linear relationships among variables. In SD, a descriptive causal model is developed first to explain the complex structure behind a problem. In this step, researchers can use real-life experiences, data, or scientific literature to construct causal relationships [1].

Thanks to the progress of data collected worldwide and data analysis tools in recent years, the possibility to ground model construction in data has increased. The usage of data analysis tools can strengthen system dynamics practice [2]. A model construction process mostly relies on data analysis is one of the interesting and promising intersections of data science and the SD method [3]. In that respect, we attempt to offer appropriate data analysis algorithms and study their limitations to automate the model construction.

In the following chapters, we first present four SD model construction stages where data analysis algorithms can be used. For the selected two stages, we first describe the problem and then, we propose a data analysis algorithm for each stage. Following the detailed description of each proposed algorithm, experiment results are given to show the effectiveness of the suggested algorithms.

2. PROBLEM DEFINITION

2.1. Problem Statement

Model building is the most significant task in system dynamics methodology. Causal models help researchers to understand and control systems behind problematic behaviours. The constructed model reflects the causal structure that creates the problem [4]. That is why it is crucial to have a comprehensive understanding of the modeling process.

A typical SD modeling procedure follows some standard steps. First of all, a dynamic problem which has a feedback nature is precisely identified. Then, a conceptual model, namely a causal loop diagram (CLD), is constructed. This diagram shows the causes and the relations behind the problematic structure. Afterwards, stock and flow variables are identified. Stock variables represent accumulations over time such as a population or an inventory. Flow variables represent the rate of change of the identified stocks. As the following step, a formal model of the system, namely stock-flow diagram (SFD) is constructed by using CLD and stock-flow variables. Then, the mathematical representation of each effect variable through its causal variables is identified. In this step, mathematical forms (additive, multiplicative, or hybrid) of the relations and the shapes of effect functions (linear, exponential, or s-shaped) are determined.

When the initial model structure is built, the validation procedure begins with structural and behavioural tests such as dimension consistency test, extreme condition test, and behavioural pattern comparison. When the modeler is convinced by the validity test results that the model is a useful representation of the real world with respect to the studied problem, the model is analyzed by simulation experiments [1].

The development of the SD causal models is not an easy task. Researchers may face some challenges in this process. First of all, they need to spend a lot of effort on quantitative representation of each causal link. Each causal link represents a *ceteris*

paribus relation: "other things being equal" the causality is expected to hold the given effect. Since there are many other varying factors in reality, the real-life data-sets are usually *non-ceteris paribus*. Therefore, using such *non-ceteris paribus* data to extract *ceteris paribus* relations may result in wrong conclusions.

Moreover, the formation of the model only with real-life experience and scientific literature is a time-consuming process. Richardson [5] indicates the need of automatic procedures to be implemented in model construction and validation. In addition, Homer and Oliva [6] express that computer modeling can be time-consuming when a quick action is needed or stakes are low. Another problem related to model construction is the model subjectivity. Modeler needs to build the SD model by their real-life knowledge (*a priori*) when there is not any *ceteris paribus* data-set or scientific knowledge. A model constructed only with *a priori* knowledge or *non-ceteris paribus* data-set may encounter the problem of being subjective [7–9].

In this study, the focus is on the partial automation of SD model building by using real-life data. By the automated analysis of real-life data of model variables, it is aimed to reduce "the possibility of reaching wrong conclusions", "model construction time" and "model subjectivity".

2.2. Research Objective

SD model construction process can be divided into four possible stages where computational causality inference from data can be used. (1) Discovering causal relations (links): We may want to discover whether or not there is a direct causality between system variables (see Figure 2.1.a.). This process can be also considered as a (partial) causal loop diagram discovery. If which variables are stocks is already known, then this process can be considered as a (partial) stock-flow diagram discovery. (2) Discovering the polarity of relations: Under the assumption that we assume/know all the cause variables affecting the effect variable, we aim to find whether the links are positive or negative between variables (see Figure 2.1.b.). (3) Discovering stock variables: Under the assumption that we assume/know the CLD, we aim to find which ones are

stocks among variables. Therefore, this stage can be considered as stock-flow diagram discovery from CLD, using data tools (see Figure 2.1.c.). (4) Discovering mathematical expression of relations: This fourth inference stage can be under the assumption of a known structure, CLD or SFD. The aim is to discover the mathematical expressions that represent the relations between variables. This fourth category can be divided into two, as discovering mathematical forms of the relations and discovering unknown parameters, functional shapes, in these mathematical forms (see Figure 2.1.d.)

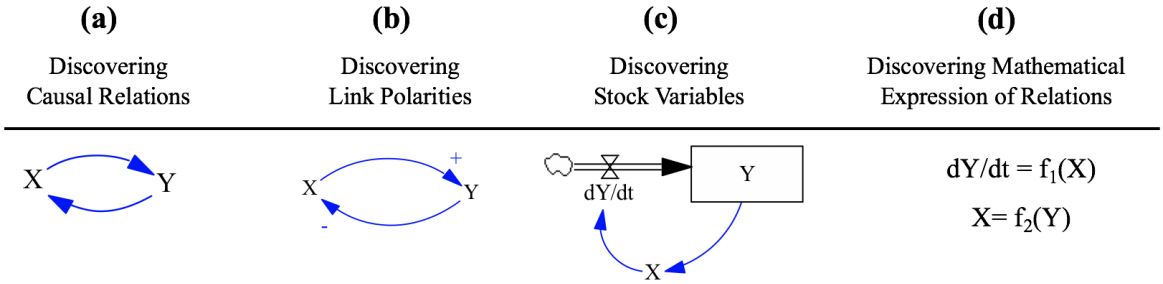

Figure 2.1. Causality Inferences in Model Construction.

Discovering causal relations stage (1) without *a priori* knowledge or scientific literature is nearly impossible because we cannot decide on the direction of causal arrows by using data. Any mathematical expression of "$x \rightarrow y$" relation obtained by real-life data can be easily expressed by the opposite relation: "$y \rightarrow x$". In addition, discovering mathematical expression of relations (4) may create misleading results, especially for policy analysis, if the collected data range is not sufficient enough to represent the overall causal relations. Due to above difficulties behind these two stages, we focus on the possibility to extract polarities of relations (2) and discovering stock variables (3) from a given real-life data-set in this research. To achieve this, we develop different algorithms for each purpose which are explained in Chapter 4 & 5.

3. LITERATURE REVIEW

In this section, data analysis tools applied in SD model construction are classified according to four causal inference stages mentioned in the previous section. In addition, some other data analysis tools that are not used in SD but have a potential to be applied in SD are presented.

For discovering causal relations (stage 1), Abdelbari and Shafi [10] propose echo state neural network-based methodology to obtain CLD from observations of system variables. They do not claim to find "one of the representative models" behind the problem, but their proposed approach aims to provide modelers with several probable model structures to start with. In addition, Medina-Borja and Pasupathy [11] applies Classification and Regression Trees and Chi-Square Automatic Interaction Detection methods to discover CLD and applies structural equation modeling (SEM) to find mathematical expressions behind the relations. Outside of the SD context, Bühlmann *et al.* [12] propose causal additive models to discover the causal structure behind the data of given variables. They try to estimate a structural model which only includes non-linear additive equations (monotonic or non-monotonic).

To extract the polarities of known causal links (stage 2), no data analysis tool exists in the literature. However, the correlation analysis, structural equation modeling (SEM), general linear models (GLM), and general additive models (GAM) can be utilized to extract polarities, since these methods are commonly used to calculate strength and polarity of association between variables [13–15]. However, when it comes to the applicability of the methods in SD causal polarity analysis, the assumptions of all the methods raise difficulties: The correlation analysis fails when there is multicollinearity or inter-dependency, among variables [16]. Cause variables are mostly interdependent in real-life problems, hence, collected data-sets have multicollinearity [17]. Therefore, it is not mostly suitable to use correlation analysis in real-life data-sets which are typical in SD modeling [18].

In case of multicollinearity, SEM can be applied by adding relationships between variables [19]. However, it has other drawbacks. Hovmand and Chalise [20] explicitly describe limitations of SEM applications in SD methodology. First of all, SEM is applicable when there is a solution for the system of differential equations. If there are non-linearities in the system, it may not be possible to find a solution so that SEM may not converge to a solution. Secondly, to apply SEM, system of equations should be already identified. Therefore, if not only the parameters but also the equation forms of the system -such as additive or multiplicative form- are unknown, SEM cannot be used. To use GLM, we need to make assumptions about functional form (additive, multiplicative, etc.) and shapes of effect functions [21]. Compared to GLM, GAM seems to be more appropriate method for discovering polarities since we do not have to assume shapes of effect functions but only functional forms [22]. However, GAM is not an appropriate tool for multiplicative forms where multiplied variables both have non-linear effect functions.

To extract a stock-flow diagram from CLD (stage 3), no data analysis tools exist, but Takahashi [23] suggests that CLD can be transformed into a stock-flow diagram with a Prolog code by using logic rules for different causal relations. In this work, the transformation is performed through "language", not through observed data. Therefore, there is a gap in the literature about the usage of real-life data to extract SFD.

For extracting mathematical forms and parameters of a given CLD (stage 4), Drobek *et al.* [24] propose neural network evolution for automated equation formulation. Authors find that, their proposed approach is not capable of finding formulations in some cases and long time frames. As further research, they consider evaluating their method by analysing whether it can accurately predict formulations for such time frames. Estimating unknown parameters of mathematical formulation used in an SD model is the most studied intersection of data science and SD methodology. For discovering unknown parameters, there are data analysis tools such as methods of bootstapping [25], simulated moments [26], and indirect inference [27]. These methods give promising results under different assumptions.

In the above literature, there are not enough discussions about the extent of data usage in the discovery of link polarities and stock variables. In other words, it is not clear under which conditions, analysing data can produce practical information for link polarity and stock-flow discovery. Hereby, we focus on the possibility to extract the polarity of relations and discovering stock variables by using historical data of the variables.

4. DISCOVERING THE POLARITY OF RELATIONS

4.1. Methodology

In the case of non-experimental real-life data, automated polarity discovery becomes a nontrivial, sometimes impossible problem. Our research aim is to find whether the links are positive or negative between variables by using real-life data under the assumption that we assume/know all the cause variables that have a significant influence on the effect variable (see Figure 4.1). We also assume that all the causal relations (functions) are either monotonically increasing or decreasing, which is typically an accepted best practice in SD formulations [28].

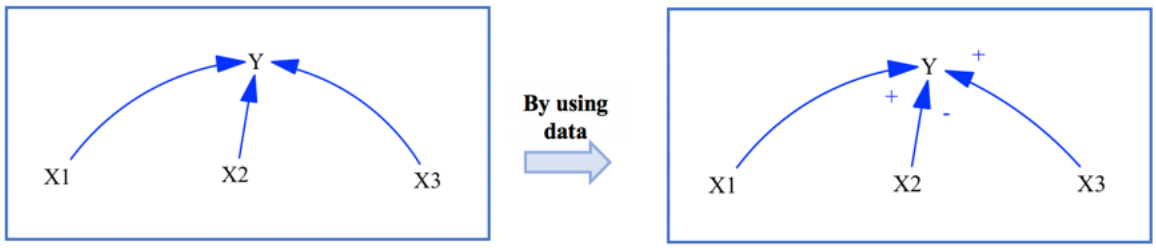

Figure 4.1. Discovering the Polarity of Relations.

Our aim is a low dimensional machine learning problem since a variable significantly affected from more than 4-5 variables is a rare case in causal descriptive models. For this purpose, we developed an algorithm that we call *discoverpolarity* (see Appendix A). In this study, instead of actual real-life data, 'synthetic' data is used to be able to evaluate the validity of the results obtained from our algorithm. Both algorithm and data-sets are generated in R programming language.

Firstly, we specified an underlying structure for each experiment and generated a data-set by using that structure. Then, we assumed that we did not have the information of the link polarities. By means of known causal relations and generated data-set, we tried to (re)discover the unknown link polarities with the developed algorithm (see Figure 4.2). Spearman correlation analysis results are used as a benchmark.

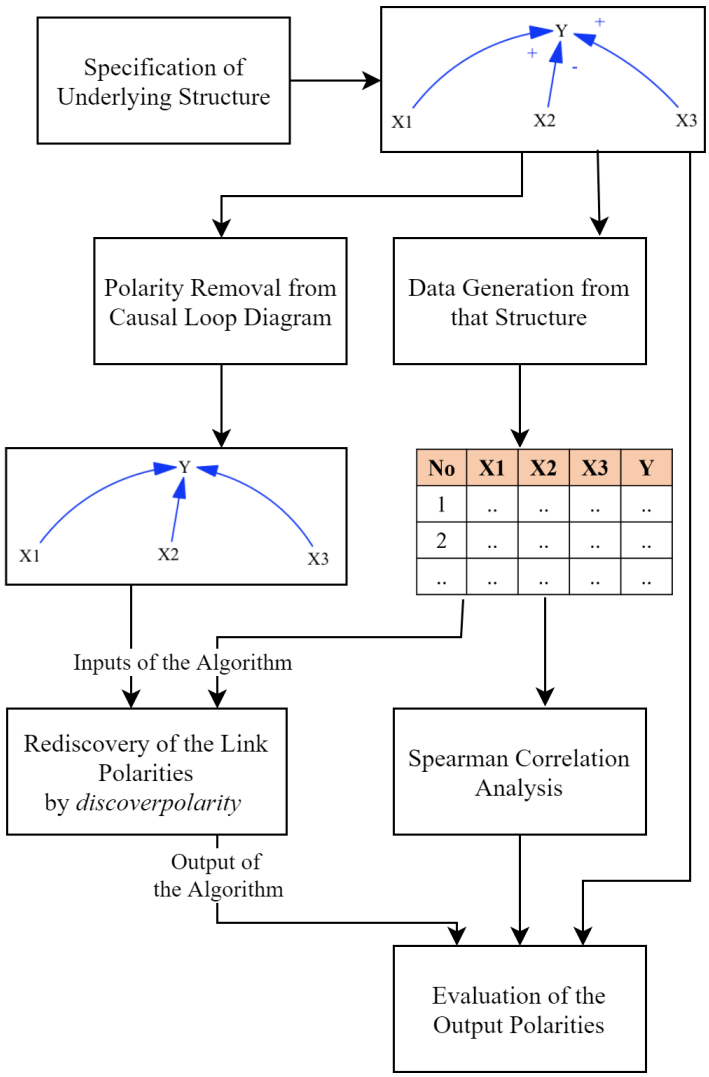

Figure 4.2. Research Design of Discovering Link Polarities.

4.1.1. Background

In a good SD practice, the effect functions are assumed to be monotonic. An effect variable (Y_t) is affected by variables ($X_{1t}, ..., X_{nt}$) through a multiplicative, additive, or hybrid combination of non-linear monotonic functions ($f_1(X_1), ..., f_n(X_n)$) [28]. Possible values of cause variables form the causal domain of a given effect variable. Each cause variable is considered to be one dimension of the causal domain. For instance, a variable affected by two cause variables is assumed to have a two-dimensional causal domain. In a causal domain, the coordinates of corners are named by signs in-

stead of values. In this notation, corner values, take (-) or (+) for each cause variable. Therefore, each corner represents a possible combination of polarity links (see Figure 4.3). Since each $f_i(X_i)$ is either non-decreasing or non-increasing function of X_i values, we encounter the maximum value of Y at one of the corners of domain of the cause variables (see 4.4).

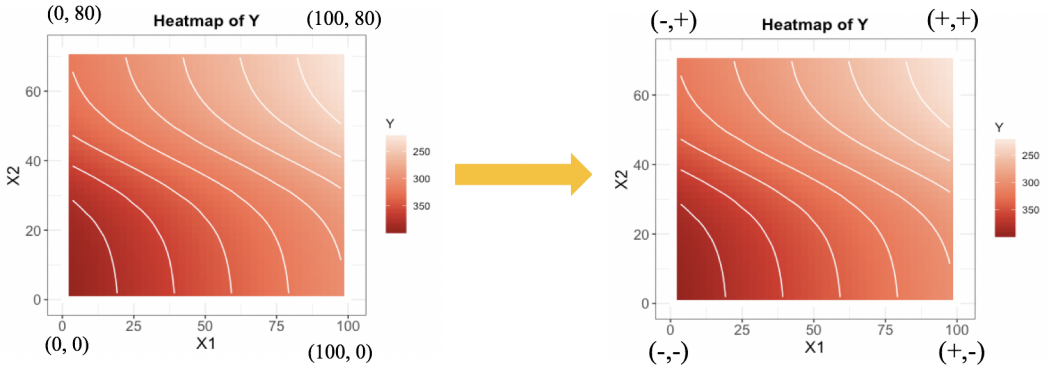

Figure 4.3. Representation of Causal Domain Corners.

In Figure 4.4, some examples are given where Y variable is affected by two cause variables (X_1 and X_2) through functions of either $Y = Yref + f_1(X_1) + f_2(X_2)$ or $Y = Yref * f_1(X_1) * f_2(X_2)$. The Y values in Figure 4.4 are reflected as colors from red (maximum) to white (minimum). It can be seen that the maximum value of Y is achieved at one of the corners of the causal domain where the cause variables take either their minimum or maximum values. Therefore, if we know in which corner the maximum value of Y is observed, then, we can derive the causal polarity of each effect function, since they are assumed to be monotonic. For example, in Figure 4.4.a, we can see that maximum value is at the corner of (-, +). This means that X_1 has a non-increasing causal effect on Y while X_2 has a non-decreasing effect. However, in Figure 4.4.c, we can see that maximum value is at the corner of (-, -). This means that both X_1 and X_2 have non-increasing effects on Y. Therefore, the question of finding the polarities of each causal relation reduces to finding the corner where Y is at its maximum (or minimum).

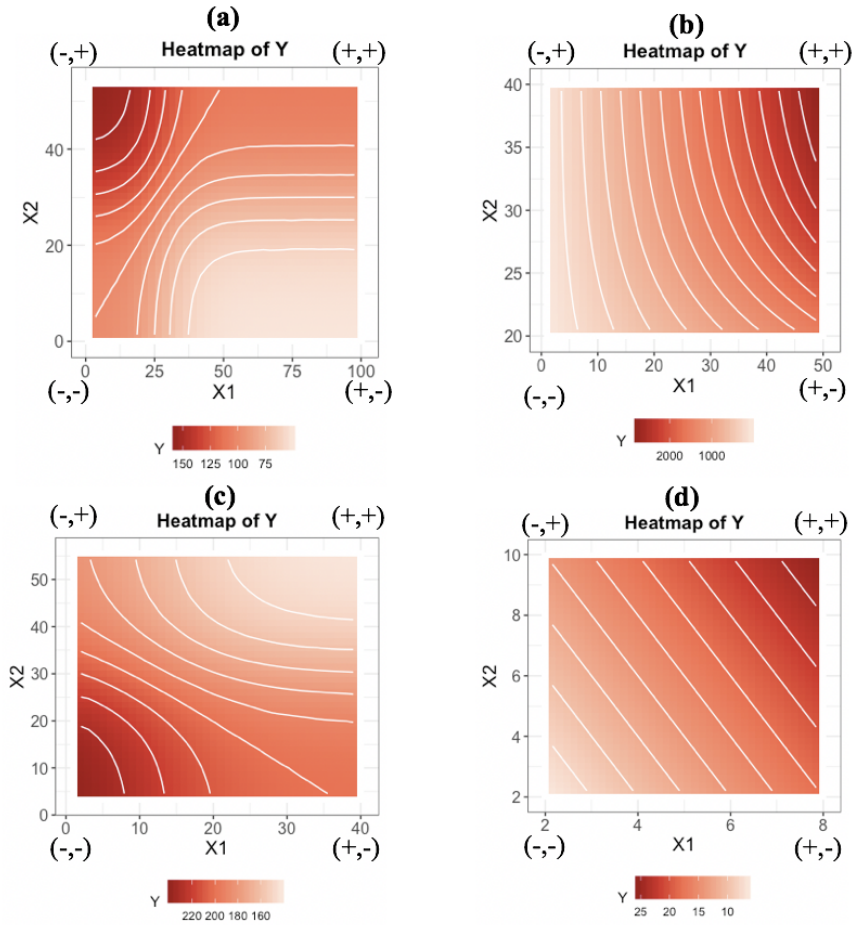

Figure 4.4. Heat-maps of Different Combinations of Monotonic Non-linear Functions.

Unfortunately, in most real-life data, we do not have all the data points that cover the causal domain. We may only have a few points in the middle of the causal domain. Therefore, we may not be able to directly compare the corner points to extract the polarities. In this research, we try to extract the polarities of causal effects by analysing imperfect datasets, just like real-life data, do not span the entire causal domain.

4.1.2. Assumptions

The challenge of polarity discovery differs essentially depending on the nature of data and the number of cause variables. The data can be collected with a controlled experiment, where data can be made -more or less- *ceteris paribus* (i), or it can be

collected from real-life, where data is *non-ceteris paribus* (ii). The causal equations in SD are based on *ceteris paribus* relations, which means that if data were gathered from a controlled experiment (i) they can be directly used in deriving the polarity of a causal relation. In other words, the polarity discovery of causal relations is a trivial question in the presence of a perfectly controlled experimental data where the effects of other variables are eliminated. However, in field (real-life) data (ii), there can be other variables affecting the variables that we are interested in. Therefore, direct use of a cross-sectional *non-ceteris paribus* data may yield misleading results (even reversed causalities) when there is more than one cause variable. In this study, we assume that there are real-life (non-experimental) real-life data about multiple system variables influencing a given effect variable.

Moreover, it is assumed that the scope of the variables, the boundary of the system, is well-chosen for the problem. Therefore, when one applies the method, she does not have to consider the possibility of a non-included variable affecting the system significantly. Moreover, it is assumed that data is noise-eliminated and has no missing values.

In addition, we assume that we know the range of cause variables in which $df_i(X_i)/dX_i$ is significantly different than 0. Therefore, we can separate the range of X_i where $f_i(X_i)$ is increasing (decreasing) with respect to X_i from the X_i domain where $f_i(X_i)$ is a (approximately) constant function. For example, a significant change in cause variable X_i, which has an S-shaped effect function, may not create any impact on the effect variable after some maximum and minimum limit of X_i, since $df_i(X_i)/dX_i$ becomes zero. To apply the algorithm, we need to know the ranges of cause variables where the variables significantly affect Y variable. For some cause variables, $df_i(X_i)/dX_i$ may be always different than zero such as linear or certain exponential functions (See Figure 4.5.a). In such cases, we do not have to consider the ranges where $df_i(X_i)/dX_i$ is almost zero and where it is significantly different than zero.

In Figure 4.5, we see the effect of X_1 on Y, the effect of X_2 on Y, and heat-map of Y in the causal variables domain. In this example, Y has an additive functional

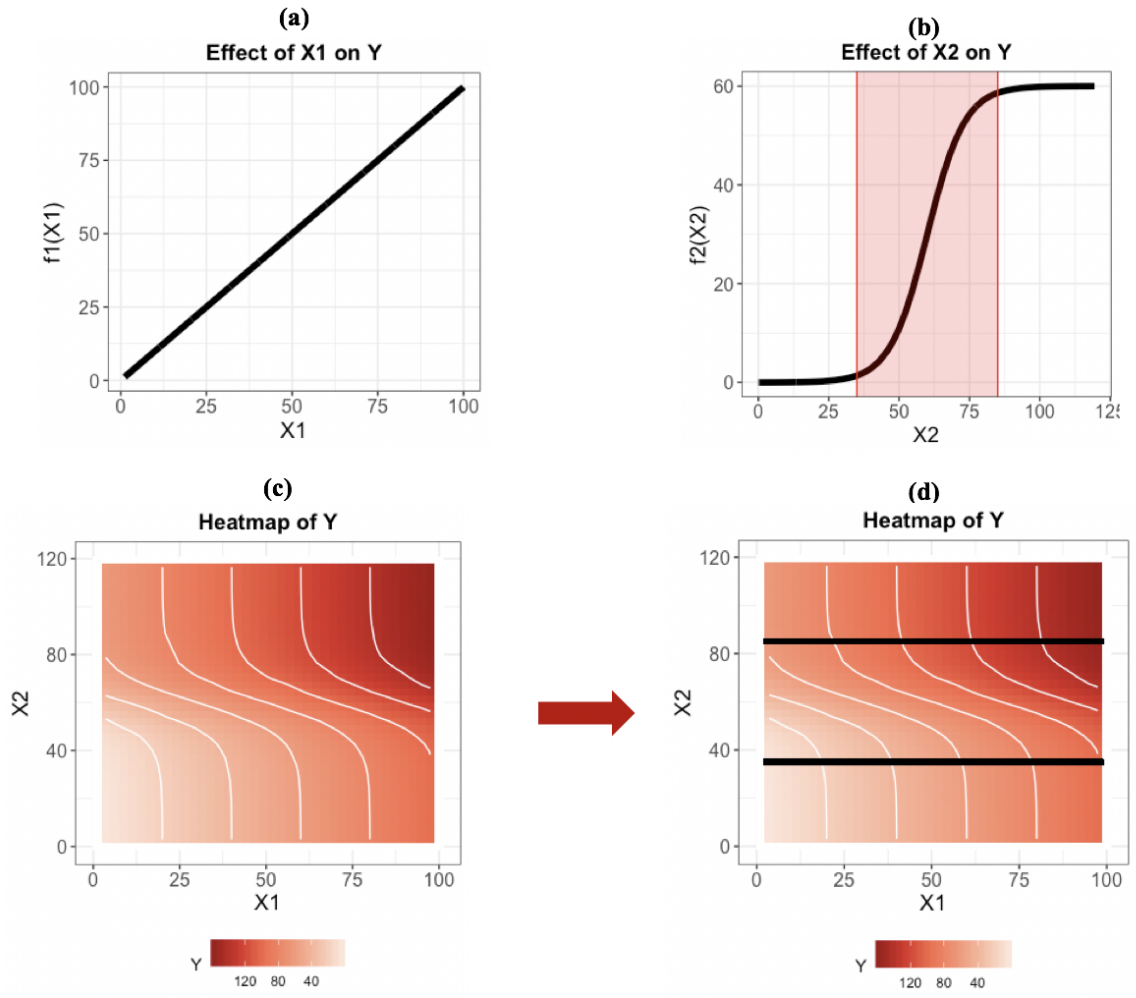

Figure 4.5. Effect Functions of Cause Variables on Y and the Heatmap of Y.

form: $Y = Yref + f_1(X_1) + f_2(X_2)$. As we see from Figure 4.5.b, X_2 variable does not affect the Y variable significantly out of the red range of $f_2(X_2)$. The limits of the red range is shown as black limits in Figure 4.5.d. The out of the limits parts are where $X_2 < 35$ and $X_2 > 85$. There is no such limits for the effect of X_1 variable in Figure 4.5.d since X_1 variable linearly affects Y variable.

4.1.3. Correlation Analysis

Correlation analysis is chosen to compare with the results of our algorithm. It is a fact that finding correlations between variables does not prove causation. However, in our case, we do not extract causation from correlation. Here, under the assumption

of the presence of causal relations, the polarities of the causalities are estimated. To do this, Spearman (as opposed to Pearson's) correlation coefficient is selected which is an appropriate choice for causality analysis in SD, because: Firstly, Spearman correlation analysis is non-parametric which means that there is no assumption for underlying distributions of variables. Secondly, just like in SD, it assumes monotonic relationship among variables [29]. To calculate Spearman coefficient (ρ), which is between [-1, 1], data (X, Y) are ranked and assigned values 1,2, ... so on. For tied scores, a calculated mean rank is given. Then, for each X and Y pairs, the difference between ranks of two values is calculated. Using these differences Spearman coefficient is obtained by the following formula.

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

With the presence of another cause variable, correlation analysis between variables can be misleading. Therefore, a partial correlation analysis must be conducted. Partial correlation is the correlation between two variables after removing the effect of other existing variables. The following formula is for the partial correlation coefficient between variable 1 and 2 in the case of one additional variable 3.

$$\rho_{12.3} = \frac{\rho_{12} - \rho_{13}\rho_{23}}{\sqrt{(1 - \rho_{13}^2)(1 - \rho_{23}^2)}}$$

4.2. Algorithm: *Discoverpolarity*

4.2.1. The Basic Process (First Phase) of the Algorithm

Our algorithm works with proof by contradiction. It compares changes in the effect variable while moving among observed data points in the causal domain. Then, it eliminates the corner points that cannot have the maximum value of the effect variable. In other words, the algorithm takes the "causal variable" and respective "effect variable" values as inputs. Then, it creates the "difference" data-set by differencing

all the pairwise data points. In the differenced data-set, it only checks the signs of differences. By considering differences signs, it eliminates the impossible cases (corners) from *all the possible polarities list* (names of corners of the causal domain) which has 2 to the power of the 'number of causal variables' possibilities. *All the possible polarities list* has 4 values when there are two cause variables: (+, +), (+, -), (-, +), (-, -). The process steps are given in Figure 4.6.

This process can be better understood with a simple example in Figure 4.7. A data-set for two cause variables (X_1, X_2) and one effect variable (Y) is given for four data points where the true underlying (generating) function is $Y = 2X_1 + X_2$. From the underlying function, we see that polarities of causal effects are both positive (+, +). But we do not know the underlying function and try to discover the causal polarity, (+, +).

For the example in Figure 4.7, the algorithm takes the observed data points in tabular form (see Table 4.1.). Then, it takes the differences of variables for each pair of rows (see Table 4.2). The basic process of the algorithm eliminates the corner with the same polarity of the causal variable movement while Y value is decreasing, and it eliminates the corner with the opposite polarities of the causal variable movement while Y value is increasing. Therefore, we analyze all of the paired combinations of the data points we have. In total, we check the sign of changes in $n * (n-1)/2$ points for a data-set composed of n points.

Table 4.1. Observed Data.

No	**X1**	**X2**	**Y**
1	5	2	12
2	6	3	15
3	8	1	17
4	2	10	14

From Table 4.2, the algorithm only needs the information of "increase/decrease"

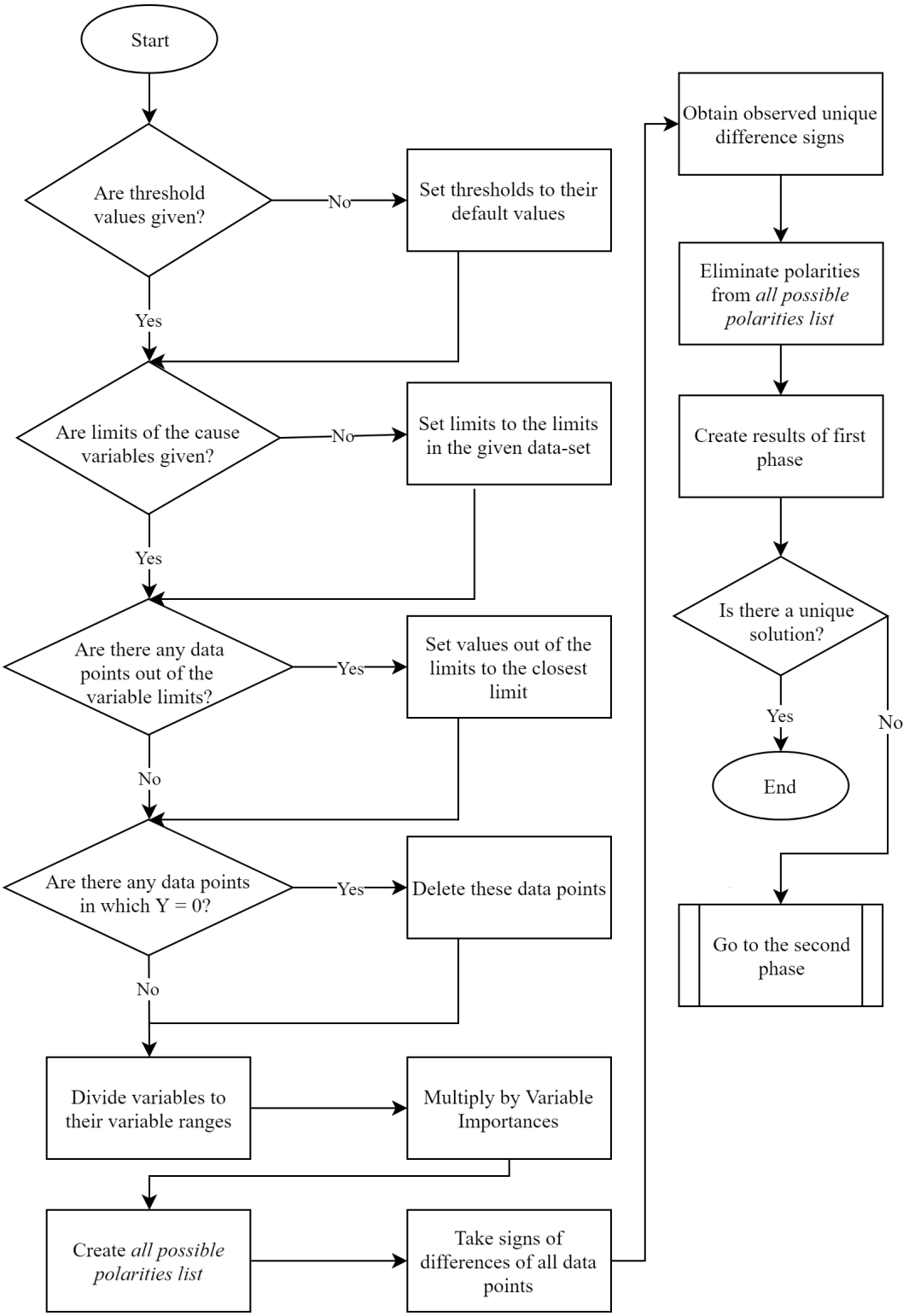

Figure 4.6. Flow Chart of the First Phase.

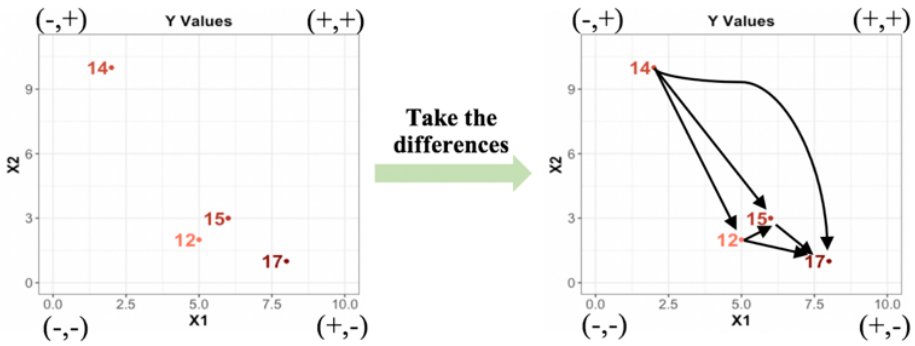

Figure 4.7. A Simple Example where $Y = 2X_1 + X_2$.

for each variable. Therefore, the algorithm creates another table by only considering "signs of change" of variables (see Table 4.3).

Table 4.2. All Differences in the Observed Data.

diff(No)	**diff(X1)**	**diff(X2)**	**diff(Y)**
2-1	1	1	3
3-2	2	-2	2
4-3	-6	9	-3
3-1	3	-1	5
4-2	-4	7	-1
4-1	-3	8	2

As it is seen in Table 4.3, there are repeated rows and the rows which have exact opposite signs for each variable (see rows 3 & 5, and rows 4 & 5). Such rows have the same information. For example, rows 3 and 5 have the information of "Y decreases while X1 decreases and X2 increases". After considering the information in row 3, we do not have to check row 5. Therefore, it is enough to check one of the repeated rows. Moreover, the rows include opposite signs for each variable have the same information, too. For example, rows 3 has the information of "Y decreases while X1 decreases and X2 increases" while row 4 has the information of "Y increases while X1 increases and X2 decreases". When we have one of these rows (statements), we can automatically

Table 4.3. Change Signs of Variables.

diff(No)	**Polarity of X1 Change**	**Polarity of X2 Change**	**Polarity of Y Change**
2-1	+	+	+
3-2	+	-	+
4-3	-	+	-
3-1	+	-	+
4-2	-	+	-
4-1	-	+	+

obtain the opposite information. Therefore, it is enough to check only one of the opposite rows. To get rid of the repetitive information, algorithm only considers rows with unique statements by excluding repeated and opposite rows (see Table 4.4).

Table 4.4. Unique Observed Change Signs of Variables.

Observed Change Signs		
Polarity of X1 Change	**Polarity of X2 Change**	**Polarity of Y Change**
+	+	+
+	-	+
-	+	+

In Table 4.4, it is easy to interpret each row. For instance, in the first row we have the following information: "Y increases when X1 and X2 are increased". When we observe an increase in Y when X1 and X2 are increased, we can say that X1 and X2 cannot both have a decreasing effect on Y variable. Hence, we eliminate (-, -) from all the possible polarities list. In generic form, the algorithm eliminates all impossible polarities from *all the possible polarities list* after obtaining the observed change signs in differences. As the geometric interpretation, what we do is to compare (-, -) corner

and (+, +) corner (see Figure 4.7). It is seen that when we move from the corner (-, -) to corner (+, +), we observe an increase in Y. Therefore, corner (-, -) should be eliminated. However, by only looking at the first row, we cannot comment on the corner points (+, -) and (-, +). After the elimination of (-, -), second and third rows in Table 4.4 eliminate corners of (-, +) and (+, -) respectively. Therefore, (+, +) becomes the only possibility which represents the correct polarities for this example (see Table 4.5).

Table 4.5. The Eliminated and Left (bold) Options.

All the Possible Effect Polarities of X1 and X2 on Y (f1(X1), f2(X2))	**Comments**
(+,+)	**(+,+) is not eliminated, therefore it can be the polarities of causal functions**
(+,-)	(+,-) is eliminated since d(Y)=(+) is observed while (dX1, dX2)= (-,+)
(-,+)	(-,+) is eliminated since d(Y)=(-) is observed while (dX1, dX2)= (-,+)
(-,-)	(-,-) is eliminated since d(Y)=(+) is observed while (dX1, dX2)= (+-,+)

Apart from the above example, the algorithm also handles "no change" in cause (1) and effect variables (2). For no change in cause variables (1), algorithm applies the following procedure: When there is no change in some of the causal variables, the algorithm only considers the changes in the other causal variables. For instance, if we observe changes in the cause variables as (dX1, dX2, dX3) = (+, +, 0) while d(Y)= (+), the algorithm eliminates 2 corners: (f1(X1), f2(X2), f3(X3)) = (-, -, +) and (f1(X1), f2(X2), f3(X3)) = (-, -, -). As another example, if we observe changes in the cause variables as (dX1, dX2, dX3) = (+, 0, 0) while d(Y)= (+), the algorithm eliminates 4 corners: (f1(X1), f2(X2), f3(X3)) = (-, +, +), (f1(X1), f2(X2), f3(X3)) = (-, +, -), (f1(X1), f2(X2), f3(X3)) = (-, -, +), (f1(X1), f2(X2), f3(X3)) = (-, -, -). The

elimination of four corners leaves only the corners with f1(X1) = (+) by dropping all the corners with f1(X1) = (-).

When we observe no change in the Y variable while a causal variable significantly changes (2), the algorithm eliminates both corners with the same and the opposite polarities of the causal variable movement. For example, if we observe changes in the cause variables as (dX1, dX2, dX3) = (+, +, +) while d(Y)= (0), the algorithm eliminates 2 corners: (f1(X1), f2(X2), f3(X3)) = (+, +, +) and (f1(X1), f2(X2), f3(X3)) = (-, -, -).

When a value of a cause variable which is out of its range in which $df_i(X_i)/dX_i$ is significantly different than 0, is observed, this value is set as the closest range limit of that variable. Therefore, a difference between two points out of the effective range is considered as zero even if the variable changes significantly out of its range. In addition, the points where the effect variable is zero are eliminated to prevent a misleading conclusion that may result from a multiplicative causal relationship. Because when the differences of two points in a multiplicative causal relationship where effect variable takes value of zero is calculated, zero change in the effect variable is obtained. Hence, *discoverpolarity* would eliminate the signs of the observed differences in cause variables. However, this action may eliminate true corners since no change in the effect variable is actually caused just by its zero value.

4.2.2. The Second Phase of the Algorithm

In some data, there may not be enough information to compare and eliminate many corners. Therefore, because of multicollinearity or insufficient data, we may end up with multiple possibilities instead of a unique solution.

When we end up with multiple solutions, *discoverpolarity* moves into the second phase. At this point, the algorithm has a default assumption: in the range of the given limits of the variables, the maximum *ceteris paribus* changes created by all X_i's on Y are (approximately) the same. In other words, influence importances of X_i's on Y

are assumed the same. However, if the modeler knows the maximum *ceteris paribus* importance of each variable which is significantly different than each other, then she must give these variable importances as an input, *varImp*.

The flow chart of the second phase given in Figure 4.8. In the second phase, *discoverpolarity* takes increasing or decreasing amounts of change of cause and effect variables into account. We already know that a significant change in a cause variable must produce a change (ceteris paribus) in the effect variable. Therefore, when a cause variable X_i does not change much but other cause variables $X_{j,j\neq i}$ change significantly, then we can say that the change in the effect variable results from $X_{j,j\neq i}$.

In this step, *discoverpolarity* subtracts the minimum limit of the variable from the variable value and then divides the variables by their given limits (the limit of the effect variable is its maximum value). By doing so, it obtains relative changes over a range of (0, 1). Then, if variable importances are different, the algorithm multiplies relative importances (minimum is set as 1) with the difference values. In other words, we increase the importance of some variables according to the given values.

In the new causal domain, *discoverpolarity* takes the absolute value of the differences of each data point to obtain the relative absolute difference values for each causal variable. Then, *discoverpolarity* searches for points close to each other with respect to each cause variable. The points that have relative difference values smaller than *threshold1* (default value is set as 0.15) are considered as 'close' relative to a cause variable and are considered as first candidate indexes. Then, for each first candidate index, *discoverpolarity* checks the absolute difference values of other cause variables. If the ratio of *"the absolute difference value of candidate cause variable/sum of the absolute difference value of all causal variables in the candidate index"* is smaller than *threshold2* (default value is set equal to *threshold1*), then *discoverpolarity* sets the difference value of the candidate cause variable at that point to zero. In other words, it assumes that the change in Y results from the other cause variables. By doing so, it eliminates more corners which do not include the maximum Y value.

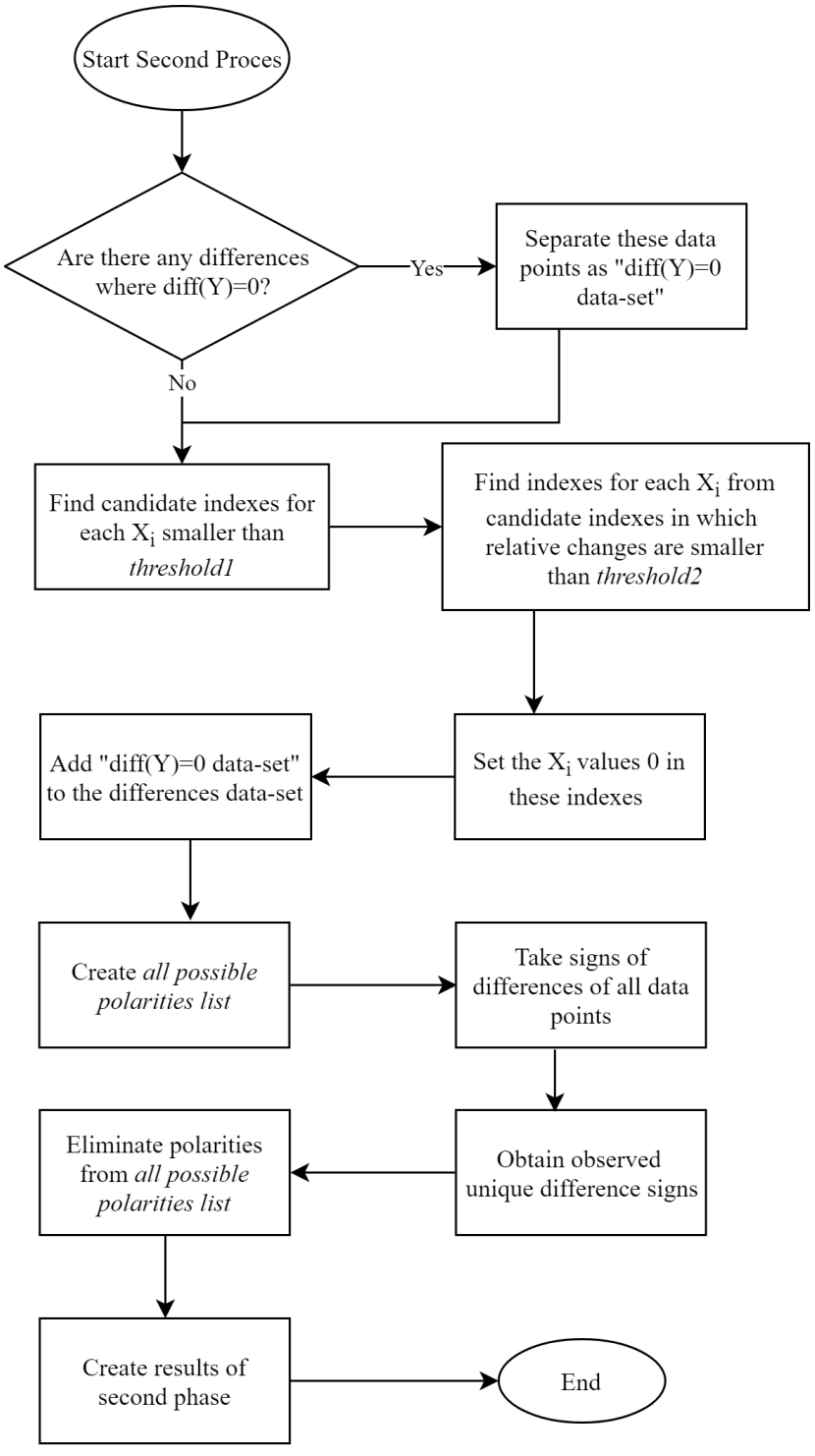

Figure 4.8. Flow Chart of the Second Phase.

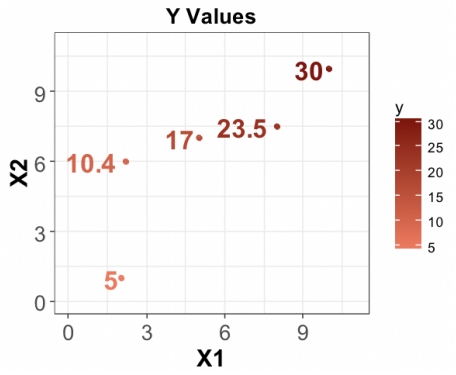

Figure 4.9. A Difficult Example where $Y = 2X_1 + X_2$.

Let's consider the example in Figure 4.9. The tabular version of the collected data is also given in Table 4.6. A collected data for two cause variables (X1, X2) and one effect variable (Y) are given for five data points where the underlying function is $Y = 2 * X1 + X2$. From the underlying function, we see that polarities of causal effects are both positive (+, +). When we apply the first phase of the algorithm to this data, we can only eliminate the corner (-, -) since what we only observe in the data is an increase in Y while X1 and X2 are increased.

Table 4.6. A Difficult Example where $Y = 2X_1 + X_2$ (in Tabular Form).

No	**X1**	**X2**	**Y**
1	2	1	5
2	2.2	6	10.4
3	5	7	17
4	8	7.5	23.5
5	10	10	30

In Figure 4.10.a, the second phase of *discoverpolarity* which is applied to the data-set in Figure 4.9 is given. After the first phase, we end up with the possibilities (-, +), (+, -), and (+, +) for this data, since what we only observe is an increase in Y value while cause variables are increasing together. We start with the second phase by

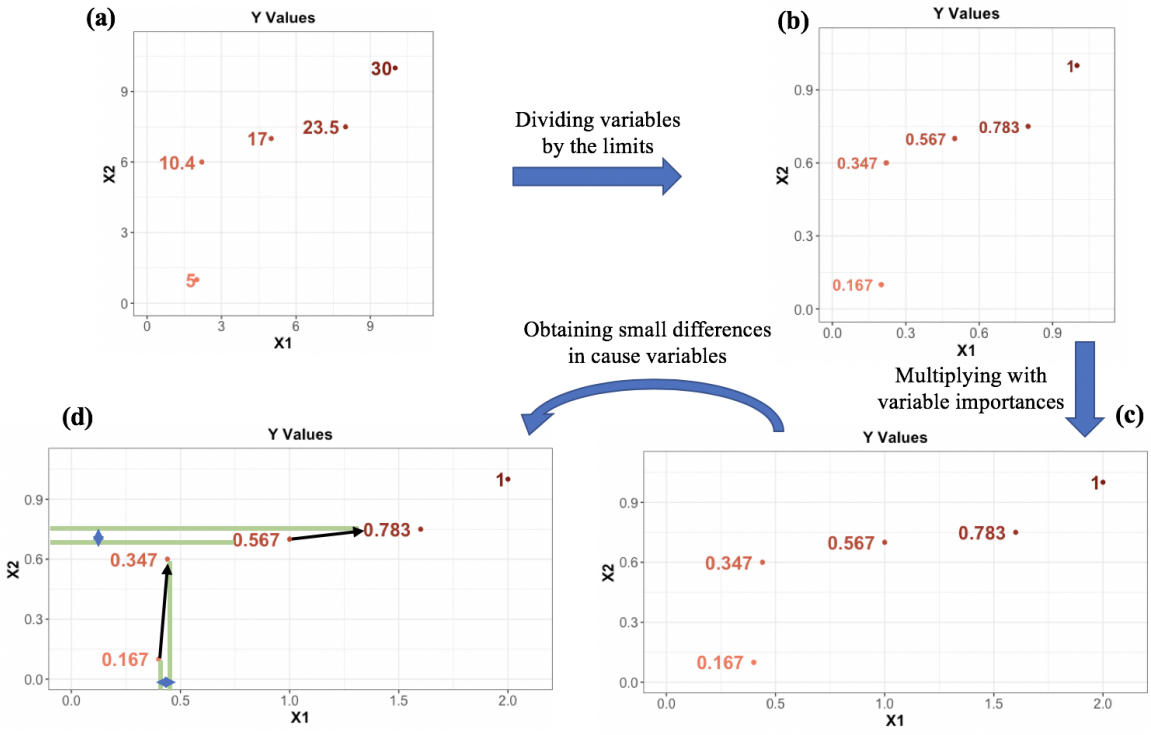

Figure 4.10. An Example for the Second Phase of *Discoverpolarity*.

obtaining relative values (see Figure 4.10.b). Then, we check the variable importances. For the observed data ranges of cause variables, X1 is twice effective than X2. By assuming that the modeler knows the variable importances, we multiply X1 domain by two (see Figure 4.10.c). Then, we find first candidate indexes where some of the cause variables change less than *threshold1* of their limits (see Figure 4.10.d).

We set those changes to zero if the change of the candidate cause variable is less than *threshold2* of the sum of all the changes in cause variables (See Table 4.7 & Table 4.8). Then, we interpret the observed change signs. Since an increase in Y is observed with an increase in only X2, X2 has an increasing effect on Y. Similarly, since an increase in Y is observed with an increase in only X1, X1 has an increasing effect on Y. Therefore, we end up with only one possibility (+, +), which reflects the true causal polarities (instead of ending up with three options).

Table 4.7. Change Signs of Variables.

diff(No)	**Polarity of X1 Change**	**Polarity of X2 Change**	**Polarity of Y Change**
2-1	**0**	+	+
3-2	+	+	+
4-3	+	**0**	+
5-4	+	+	+
3-1	+	+	+
4-2	+	+	+
5-3	+	+	+
4-1	+	+	+
5-2	+	+	+
5-1	+	+	+

Table 4.8. Observed Change Signs After Second Phase.

Observed Change Signs		
Polarity of X1 Change	**Polarity of X2 Change**	**Polarity of Y Change**
0	+	+
+	0	+
+	+	+

The second phase is applied with the following rules:

- If all the cause variables are changing less than *threshold1* of their limits between two points, the algorithm eliminates that difference from further consideration.
- If only one cause variable is changing more than *threshold1* of its limit while the effect variable is changing less than *threshold3* (default value is set equal to *threshold1*) of its limit between two points, the algorithm eliminates that difference from further interpretation.
- If several ($j = 1, \ldots, n$) cause variables are changing less than *threshold1* of their limits between two points (i^{th} difference), the algorithm applies another rule for *threshold2*: If *the candidate causal variable (j) change/sum of all the causal variables change in the candidate index i* is smaller than "*threshold2*/n" or *"the sum of all candidate causal variables (j= 1,. . . ,n) change in index i/sum of all the causal variables change in the index i"* is smaller than "*threshold2*", the candidate cause variable (j) in that index (i) is set as zero.

4.2.3. Inputs and Outputs of *Discoverpolarity*

Inputs of *discoverpolarity* and their definitions are given in the below list. When *discoverpolarity* is applied, the modeler must decide on the threshold values used in the algorithm. This is the trickiest part of the algorithm since the proper values of the thresholds may differ from one data-set to another. Even though the algorithm has default values for the thresholds, for some data-sets, the algorithm may eliminate all the possible polarities including the true polarity when default threshold values are used. This may happen if the threshold values that are used in the algorithm are too high for these data-sets. In such cases, the modeler must decrease the threshold values to obtain a reasonable result. Conversely, when the algorithm returns multiple results, the modeler may prefer to increase the threshold values. However, the usage of thresholds higher than 0.15 is not recommended since it can cause the elimination of true polarity because of high non-linearity in the effect functions.

- *causes:* Cause variables which must be given as data.frame() including the cause variable names as the column names. Data must be free of missing values. The number of cause variables to be used (n) is not limited. However, if n is beyond 4-5, the chance of obtaining a unique solution significantly decreases. For too many causal variables, significantly more data points are needed to cover the causal domain.
- *effect:* Corresponding effect variable as a vector. The vector must be free of missing values.
- *threshold1:* The threshold to select first candidate indexes for each cause variable to set them to zero. The indexes that have a change less than *threshold1* of their limits are considered as the first candidate indexes. *Threshold1* must be given between 0 and 1. The default value is 0.15.
- *threshold2:* The threshold to select indexes from the first candidate indexes list to set their values to 0. In the indexes, the causal variables that have the value of (percentage change in the candidate cause variable/sum of all the percentage changes in all the cause variables at that difference) smaller than *threshold2* are set as 0. *Threshold2* must be given between 0 and 1. The default *threshold2* is the value of *threshold1*.
- *threshold3:* The threshold to select indexes for the effect variable to set the effect variable to zero. *Threshold3* must be given between 0 and 1. The default *threshold3* value is the value of *threshold1*.
- *varImp:* Variable Importances which are used specifically in the second phase. It must be given as a vector having importance value for each cause variable with the same order as they are given in the causes data frame. The minimum value must be set as 1 and the others must be specified relative to that variable. In the default setting, all the variables' varImp values are 1.
- *limits:* Limits that define the range of cause variables where the variables significantly affect Y variable. If a variable always has a significant impact on the Y variable, the limit can be set as the limits of the observed data of that variable.

Outputs of the algorithm are as the following: If the first phase of the algorithm finds a unique solution, it returns the solution (*All1*) and the eliminated possibilities

table (*EliminatedOptions1*) with the corresponding number of differences used to eliminate the corners (*NoofObservedDifferences*). The idea behind returning the eliminated possibilities is to show how many differences there are to support the elimination of the corners. In some cases, the researcher may want to check the differences and the points used to get these differences when a possibility is eliminated with very weak evidence (number of differences).

If the algorithm goes into the second phase, it returns results of both first and second phases (*All1, EliminatedOptions1, All2, EliminatedOptions2*). Therefore, the modeler can see the results before and after applying the second phase. The solution (*All2*), sometimes multiple ones, and eliminated possibilities table (*EliminatedOptions2*) are the outputs of the second phases just like the first one.

4.2.4. Specifications of the Parameters

In the above section, it is said that the parameters used in the algorithm must be carefully chosen. For parameter specification, it is needed to properly analyze sensitivities of the parameters to data-set assumptions. To that end, the sensitivity analysis of each parameter is carried on for both phases of the algorithm.

Neither threshold values nor variable importances are used in the first phase of the algorithm. Therefore, for the first phase, only "variable limits" (*limits*) is analyzed since it is the only parameter used in the initial phase.

As a strong assumption, the effective ranges of the cause variables must be known more or less by the modeler before applying the algorithm. In case of wrong variable range specification, the algorithm may return wrong results. An example in which the algorithm fails to find the correct result because of incorrect variable range specification is given in Figure 4.11.

Assume that we have a Y variable affected by X1 and X2 by the following formula: $Y = Y_{ref} * (f_1(X_1) + f_2(X_2))$. The effect functions are given in Figure 4.11. Let us say

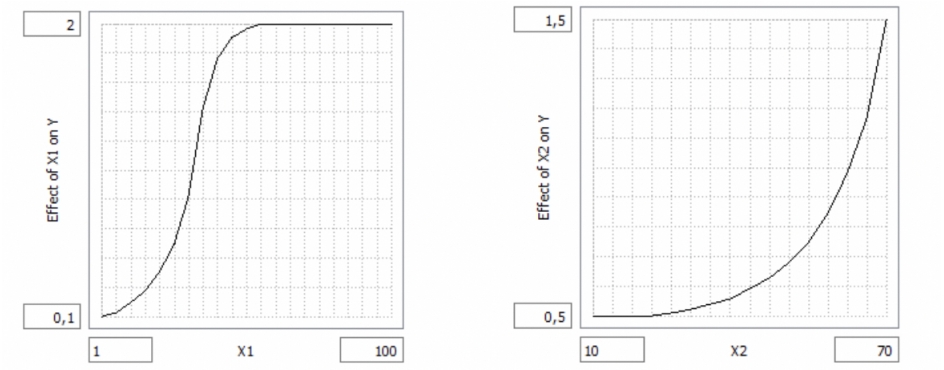

Figure 4.11. Effect Functions that Illustrate the Role of Variable Limits.

that we did not state the variable ranges in which the cause variables have significant influence on the effect variable. In other words, we claim that X1 and X2 have a significant impact on Y in their entire domain even though this is not true with the underlying effect functions. Let's suppose that we are given two data points where (X1, X2, Y) = (60, 10, 100) and (X1, X2, Y) = (100, 20, 100). In these two points, effects of X1 and X2 do not change even though the X1 and X2 change significantly. It is observed that effect variable does not change while cause variables increase significantly. Therefore, the algorithm concludes that "effect polarities of X1 and X2 cannot be (+, +) and (-, -)" and eliminates both possibilities, including the true corner of (+, +). In this scenario, the algorithm may return wrong results because of incorrect variable range specification.

This example represents the case where the first phase eliminates the correct possibility: Wrong variable limit specification. There must be two data points where all the cause variables are out of their range without a correct specification of their limits. Moreover, they must all move in their correct polarity direction or the opposite direction at the same time. In such a case, algorithm eliminates the correct possibility in the first phase, therefore, also in the second phase. When a modeler is not sure about the variable limits, she must subtract the data point differences where the effect variable does not change at all in further analysis.

The possibility of the second phase of the algorithm returning the correct solution is dependent on the selection of not only the variable ranges but also the threshold values and variable importances.

An appropriate selection of *threshold1* parameter is related to the cause variable limits (1), cause variable importances (2), and the non-linearity of the effect functions (3). When the modeler suspects about the true variable limits or the true variable importances or a high non-linearity in effect functions, she must consider decreasing the default *threshold1* value.

On the subject of variable limits, the second phase of the algorithm is more sensitive than the first phase. When we set a variable change as zero since it is "a quite small change that can be ignored" regarding to *threshold1* as other variables have considerable changes -but, they are out of their range-, the algorithm may eliminate the correct possibility. For instance, suppose that we have a Y variable affected by X1 and X2 by the following formula: $Y = Y_{ref} * (f_1(X_1) + f_2(X_2))$. The effect functions are given in Figure 4.11. Let us say that we did not state the variable limits in which the cause variables have significant effect on the effect variable. We claim that X1 and X2 have a significant impact on Y in their entire domains even though this is not true with the underlying effect functions. In our data-set, we observe X1 and X2 values between (1,100) and (10, 70), respectively. Let's suppose that we are given two data points where (X1, X2, Y) = (60, 66, 233.4) and (X1, X2, Y) = (100, 58, 169.2) where the *threshold1* value is set as 0.15. In the first phase,the algorithm only eliminates the corner of (+, -). However, in the second phase, the algorithm sets the change in X2 variables as zero since (absolute(58 - 66))/(70 - 10) = 0.13 < 0.15 = *threshold1* value. Therefore, the algorithm considers the change as (dX1, dX2, dY) = (+, 0, -) instead of (dX1, dX2, dY) = (+, -, -) and eliminates not only the corner (+, -) but also the corner (+, +) which is the correct one. Therefore, the *threshold1* value must be carefully chosen considering the variable limits.

The *threshold1* value is also sensitive to the selection of variable importances. If the modeler is not sure about the correct selection of variable importances, she must

consider decreasing *threshold1* value. Let us consider an example where the second phase eliminates correct result because of wrong selection of variable importances: suppose that we have a Y variable affected by X1 and X2 by the following formula: $Y = Y_{ref} * (f_1(X_1) + f_2(X_2))$. The effect functions are given in Figure 4.12. In the causal domains, X1 creates 0.2 * Yref at most while X2 creates 3 * Yref. Therefore, X2 must have a much higher variable importance, a value such as 10 or 20. However, assume that the modeler sets variable importances as the same. Let us suppose that we are given two data points where (X1, X2, Y) = (10, 62, 342) and (X1, X2, Y) = (40, 66, 386). In our data-set, we observe X1 and X2 values between (1,100) and (10, 70), respectively. In the first phase, the algorithm eliminates only the corner of (-, -) since (dX1, dX2, dY) = (+, +, +) is observed. However, in the second phase, the algorithm sets the change in X2 variables as zero since (absolute(62- 66))/(70 - 10) = 0.13 < 0.15 = *threshold1* value. Therefore, the algorithm considers the change as (dX1, dX2, dY) = (+, 0, +) instead of (dX1, dX2, dY) = (+, +, +) and eliminates not only the corner (-, -) but also the corner (-, +) which is the correct one. Therefore, the *threshold1* value must be carefully chosen considering the variable importances. The correct corner is eliminated with these two data points when a high importance value is given to X2 variable.

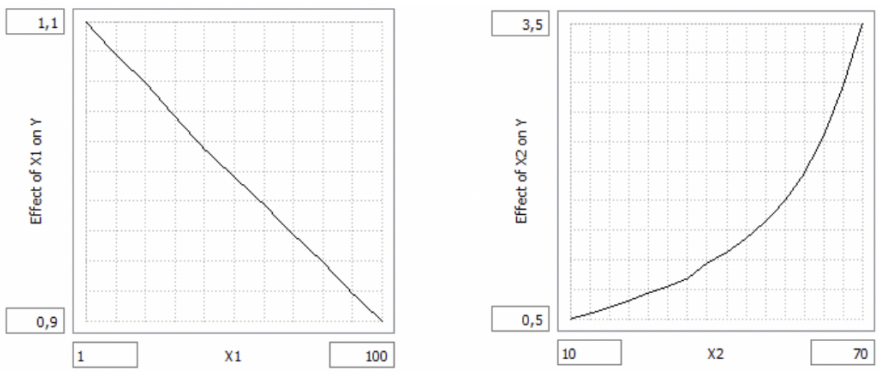

Figure 4.12. Effect Functions that Illustrate the Role of Variable Importances.

The *threshold1* and *threshold2* values are also sensitive to data-sets with high non-linearities. Here, non-linearity refers to the effect function shapes not the formulation forms as multiplicative or hybrid. If the modeler suspects that there are high non-

linearities in the effect function shapes, she must consider decreasing *threshold1* value. Let us consider an example where the second phase eliminates correct result because of ignored high non-linearities in the effect functions: suppose that we have a Y variable affected by X1 and X2 by the following formula: Y= Yref * (f1(X1) + f2(X2)). The effect functions are given in Figure 4.13. Let us suppose that we are given two data points where (X1, X2, Y) = (50, 1, 165) and (X1, X2, Y) = (45, 30, 132). In our data-set, we observe X1 and X2 values between (1,100) and (1, 100), respectively. In the first phase, the algorithm eliminates only the corner (-, +) since (dX1, dX2, dY) = (-, +, -) is observed. However, in the second phase, the algorithm sets the change in X1 variables as zero since (absolute(62- 66))/(70 - 10) = 0.13 < 0.15 = *threshold1* value and relative change in X1 = (0.05/0.34) = 0.147 < 0.15 = *threshold2* value. Therefore, the algorithm considers the change as (dX1, dX2, dY) = (+, 0, -) instead of (dX1, dX2, dY) = (+, +, -) and eliminates not only the corner (-, +) but also the corner (+, +) which is the correct one. Therefore, the *threshold1* value must be carefully chosen considering the non-linearities.

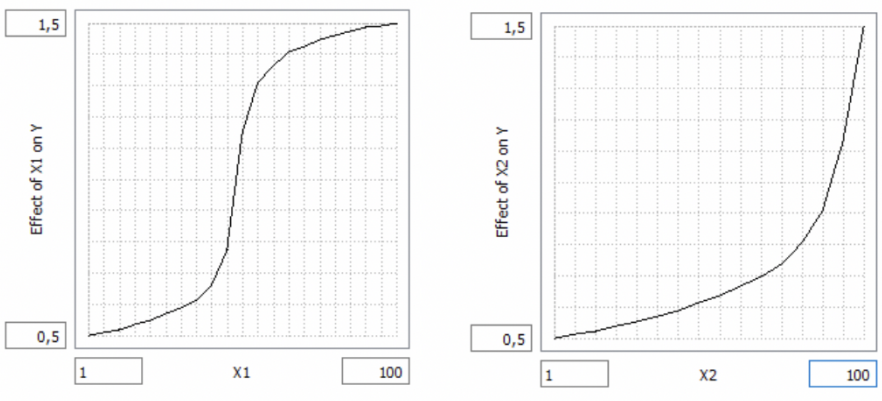

Figure 4.13. Effect Functions that Illustrate the Role of Non-linearity.

The *threshold2* value is also sensitive to the number of cause variables since it is the division of a cause variable change to the total sum of variable changes in their domain. In a formulation where there are many cause variables, a lower *threshold2* must be considered. Assume that we have two cause variables (X1 and X2) which have changes of 15% and 30% in their domain. And, assume *threshold1* and *threshold2* variables are set as 0.15. Then, the algorithm does not set the change of X1 to zero

since $0.15/(0.15 + 0.30) = 0.33 > 0.15 =$*threshold2*. Let us consider another cause domain which has 5 variables: X1, X2, X3, X4, and X5. Assume that between two data points, we observe cause variable changes as (X1, X2, X3, X4, X5) = (15%, 30%, 30%, 30%, 30%) in their domain. In that case, the algorithm set the change of X1 to zero: $0.15/(0.15 + 0.30 + 0.30 + 0.30 + 0.30) = 0.125 < 0.15 =$ *threshold2*.

The *threshold3* variable is sensitive to variable limits too. If the modeler is not sure about an appropriate specification of cause variable limits, she must consider lowering *threshold3* value to be able to not eliminate correct solution.

Lastly, determination of variable importance is sensitive to the functional form: additive, multiplicative, or hybrid. If the underlying effects have an additive formulation, the variable importance can be selected according to their effect ranges. If the form is multiplicative, all variables have the same importance. However, in that case, specification of lower threshold values becomes a better choice. Since a small change in a variable can have a bigger influence with a multiplicative formulation.

4.3. Experiments and Results in Discovering The Polarity Of Relations

In a truely randomized data-set, Spearman correlation analysis most probably returns the correct polarities of causal links. However, most real-life data-sets have collinearity among variables which make polarity discovery difficult. Because of multicollinearity, correlation analysis may return wrong results in real-life data-sets. Our algorithm is tested, and the results are reported for seven different data-sets in which there are multicollinearities (see Appendix B). It is important to note that samples are carefully selected to be able to represent real-life data-sets that consist of mostly dependent data points typical in system dynamic problems [17]. In other words, there are high correlations among cause variables in these data-sets: The data points are not randomly selected, hence they do not mostly cover the total causal domain.

For each data-set, the underlying structure is chosen as a different typical SD formulation (see Table 4.9). First and the second data-set have two; data-sets 3, 4,

and 5 have four; data-sets 6 and 7 have three cause variables. Data-set 7 represents a poor data-set where the algorithm fails to give a useful insight.

Table 4.9. Data-Sets and Underlying Structure Formulations.

Data-set	**Formulation**
Data-set 1	$Y = Y_{ref} + f_1(X_1) + f_2(X_2)$
Data-set 2	$Y = Y_{ref} * f_1(X_1) * f_2(X_2)$
Data-set 3	$Y = Y_{ref} + f_1(X_1) + f_2(X_2) + f_3(X_3) + f_4(X_4)$
Data-set 4	$Y = Y_{ref} + f_1(X_1) + f_2(X_2) + f_3(X_3) + f_4(X_4)$
Data-set 5	$Y = Y_{ref} * f_4(X_4) * (f_1(X_1) + f_2(X_2) + f_3(X_3))$
Data-set 6	$Y = Y_{ref} * (f_1(X_1) * f_2(X_2) * f_3(X_3))$
Data-set 7	$Y = Y_{ref} * (f_1(X_1) + f_2(X_2) + f_3(X_3))$

4.3.1. Data-set 1: Additive Formulation with Two Cause Variables

The first data-set has Y and X variables with the following generating function:

$$Y = Y_{ref} + f_1(X_1) + f_2(X_2)$$

Each true effect function and the heat-map of Y in the causal domain is given in the Figure 4.14. The data-set that we have with 20 samples is shown in Figure 4.15. The correlation matrix of data-set 1 is given in Table 4.10. The cause variables are almost perfectly correlated.

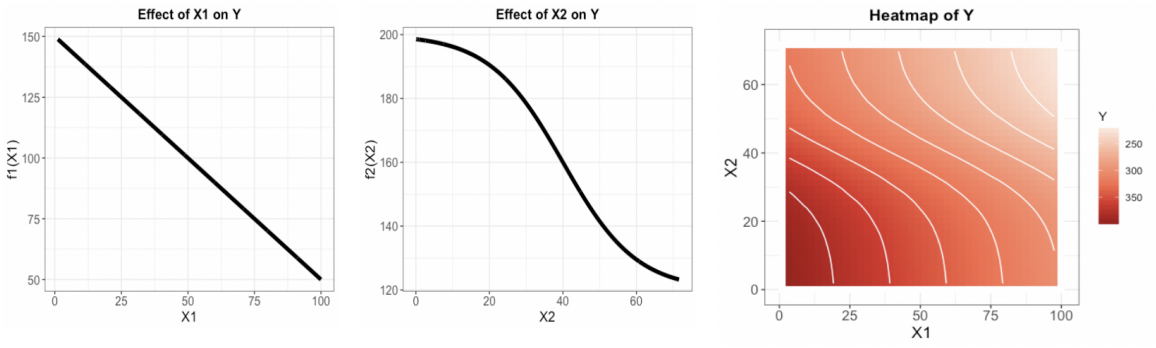

Figure 4.14. Underlying Functions and Heat-map of the Effect Variable in Data-set 1.

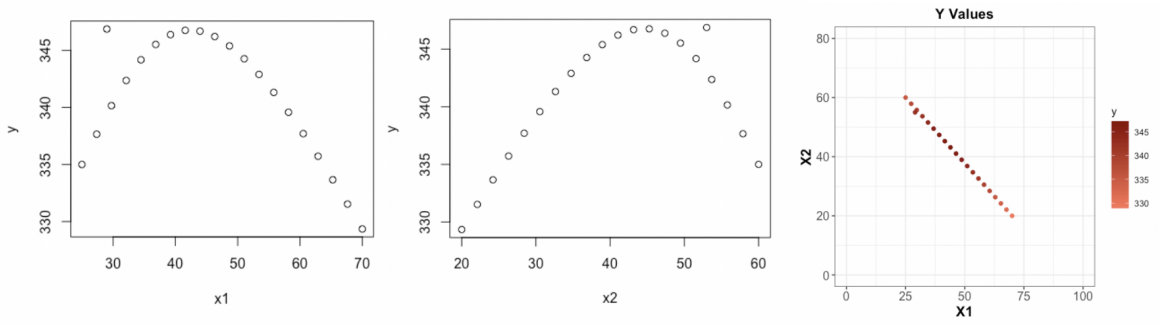

Figure 4.15. Data-set 1.

Table 4.10. Spearman Correlation Matrix of Data-set 1.

	x1	x2	y
x1	1.0000	-0.9987	-0.3662
x2	-0.9987	1.0000	0.3623
y	-0.3662	0.3623	1.0000

When we analyze the data-set 1 with *discoverpolarity* algorithm, we get the unique result in the first phase (see Table 4.11). In this analysis, variable importances are given as one (default value of *varImp*) and no variable limit is specified. Since the algorithm obtains a unique solution in the first phase, it does not move into the second phase. We see that the only possibility left at the end is (-, -) which means that both cause variables have a non-increasing effect on Y variable which is the correct conclusion. Through partial Spearman correlation, we obtain the following correlation coefficients:

$$(\rho_{X1Y.X2}, \rho_{X2Y.X1}) = (-0.09, -0.07)$$

These coefficients represent correct polarity signs with respective P-values of 0.70 and 0.76. Therefore, both *discoverpolarity* and correlation analysis are capable of returning correct polarities in simple additive formulations only include a few cause variables.

Table 4.11. Results of the Algorithm in Data-set 1.

Result(s) (X1, X2)	Eliminated Options	Number of Observed Differences to Support the Elimination
(-,-)	(+,+)	1
	(+,-)	135
	(-,+)	74

Note that we only have one difference cell to support the elimination of (+, +) corner. Even though *discoverpolarity* has the assumption of no-noise, the researcher may consider not to eliminate the (+, +) corner or may consider obtaining more historical data. As it is indicated for data-set specification at the beginning of Chapter 4, the data-sets are -on purpose- determined to have high multicollinearity. However, with more and mostly randomly selected data, the result of the algorithm may become more robust. Let us look at the data-set in Figure 4.16 which combines data-set 1

and ten additional uniformly distributed data points. The decrease in the correlation between cause variables can be seen in Table 4.13. This means that the data-set have more information to compare the corners. In this "extended data-set 1", extra 125 point differences support the elimination of (+, +) corner. We have strong evidence to eliminate corner (+, +) (see Table 4.12). Spearman correlation analysis returns correct polarities for the "extended data-set 1".

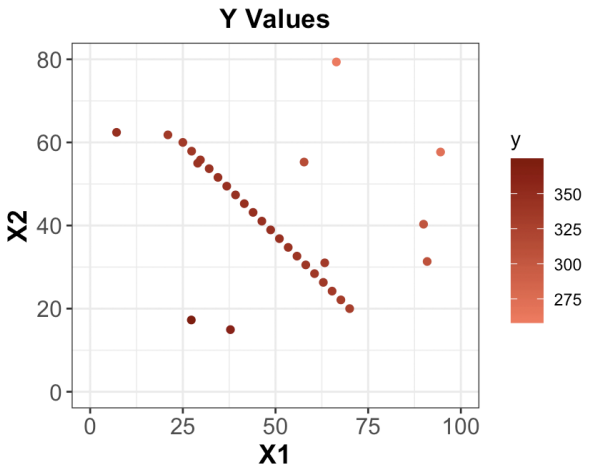

Figure 4.16. Data-set 1 with Uniformly Distributed 10 Additional Points.

Table 4.12. Results of the Algorithm in Data-set 1 with Additional Points.

Result(s) (X1, X2)	Eliminated Options	Number of Observed Differences to Support the Elimination
(-,-)	(+,+)	126
	(+,-)	222
	(-,+)	117

Table 4.13. Spearman Correlation Matrix of Extended Data-set 1.

	x1	x2	y
x1	1.0000	-0.4444	-0.6512
x2	-0.4444	1.0000	-0.0944
y	-0.6512	-0.0944	1.0000

4.3.2. Data-set 2: Multiplicative Formulation with Two Cause Variables

The second data-set has Y variable with the following underlying function:

$$Y = Y_{ref} * f_1(X_1) * f_2(X_2)$$

Each effect function and the heat-map of Y in the causal domain is given in the Figure 4.17. The data-set that we have with 26 samples is shown in Figure 4.18. The correlation matrix of data-set 2 is given in Table 4.14. The ranks of cause variables are perfectly correlated.

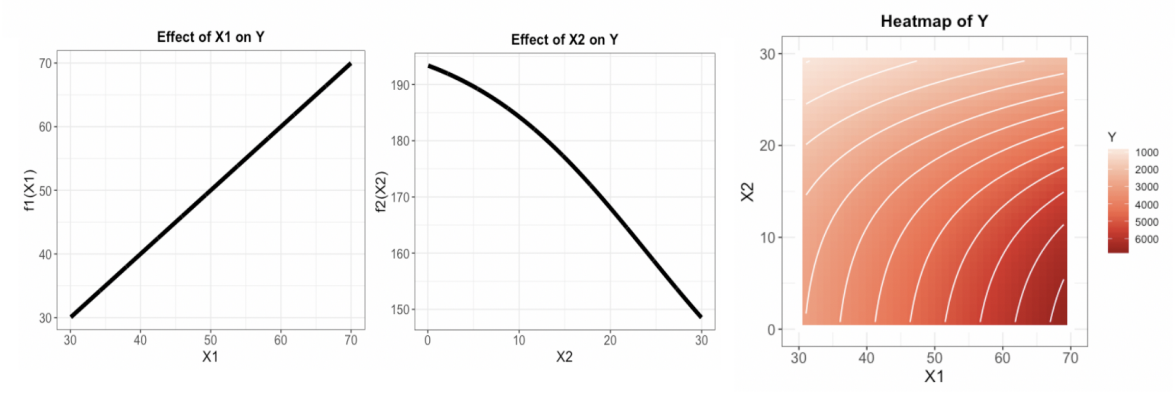

Figure 4.17. Underlying Functions and Heat-map of the Effect Variable in Data-set 2.

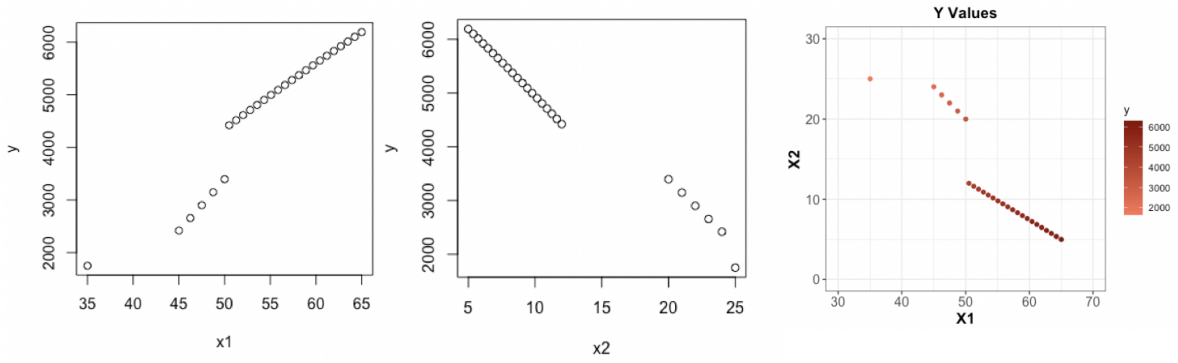

Figure 4.18. Data-set 2.

Table 4.14. Spearman Correlation Matrix of Data-set 2.

	x1	x2	y
x1	1.0000	-1.0000	1.0000
x2	-1.0000	1.0000	-1.0000
y	1.0000	-1.0000	1.0000

When we analyze the data-set 2 with *discoverpolarity* algorithm, we get the results in Table 4.15 and Table 4.16. Table 4.15 and 4.16 show the results of the first phase and second phase, respectively. In this analysis, variable importances are given as one (default value of *varImp*) and no variable limit is specified. Moreover, the default threshold values are used. In the first phase, the algorithm only eliminates the corner of (-, +). In the second phase, the algorithm is able to eliminate two other corners: (-, -) and (+, +). Therefore, the only left option is (+, -) which means that X1 has a non-decreasing effect while X2 has a non-increasing effect on Y variable. From Figure 4.17, it can be seen that this is the correct conclusion. However, the modeler may prefer to check the observed differences to support the elimination of the corners (-, -) and (+, +) since number of observed differences to support these eliminations are only one and four, respectively.

Table 4.15. Results of the First Phase of the Algorithm in Data-set 2.

Result(s) (X1, X2)	Eliminated Options	Number of Observed Differences to Support the Elimination
(-,-)	(-,+)	325
(+,-)		
(+,+)		

Table 4.16. Results of the Second Phase of the Algorithm in Data-set 2.

Result(s) (X1, X2)	Eliminated Options (X1, X2)	Number of Observed Differences to Support the Elimination
(+,-)	(-,-)	1
	(-,+)	325
	(+,+)	4

Spearman correlation analysis cannot produce any result for data-set 2 because of perfect "ranked" multicollinearity. Because of the perfect correlation of the ranks of cause variables, variance-covariance matrix of the rank values has a determinant of zero.

Additional data points can support the elimination of (-, -) and (+, +) corners. Let us look at the data-set in Figure 4.19 which combines data-set 2 and five additional uniformly distributed data points. With the additional points, the correlation of cause variables decreases from 1.00 to 0.73 (see Table 4.17). In this "extended data-set 2", there are extra 24 and 28 point differences that support the elimination of (-, -) and (+, +) corners, respectively. Thanks to these extra five points, the algorithm finds the correct unique solution in the first phase. We can confidently eliminate corners of (-, -) and (+, +)(see Table 4.18). It is important to note that the additional points in extended data reduce the multicollinearity of data, hence, Spearman correlation

analysis also returns the correct conclusion.

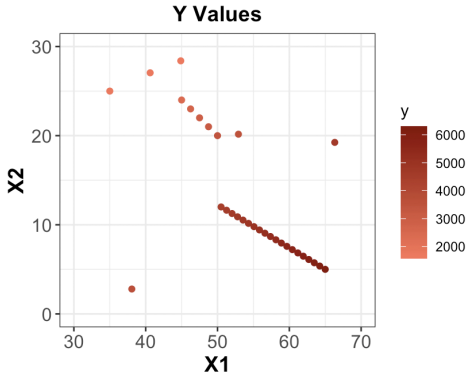

Figure 4.19. Data-set 2 with Uniformly Distributed 5 Additional Points.

Table 4.17. Spearman Correlation Matrix of Extended Data-set 2.

	x1	x2	y
x1	1.0000	-0.7298	0.9153
x2	-0.7298	1.0000	-0.9040
y	0.9153	-0.9040	1.0000

4.3.3. Data-set 3: Additive Formulation with Four Cause Variables

The third data-set has Y variable with the following underlying function:

$$Y = Y_{ref} + f_1(X_1) + f_2(X_2) + f_3(X_3) + f_4(X_4)$$

Effects of causal variables on Y are given in the Figure 4.20. From the effect functions, it is observed that X_1, X_3, and X_4 have positive while X_2 has negative effect on Y variable. Therefore, the true corner where the maximum Y variable is located is (+, -, +, +). The data-set that is used to discover the causal polarities has 100 samples as it is shown in Figure 4.21 and 4.22. The correlation matrix of data-set 3 is given in Table 4.19.

Table 4.18. Results of the Algorithm in Data-set 2 with Additional Points.

Result(s) (X1, X2)	Eliminated Options	Number of Observed Differences to Support the Elimination
(+,-)	(-, -)	25
	(-, +)	408
	(+,+)	32

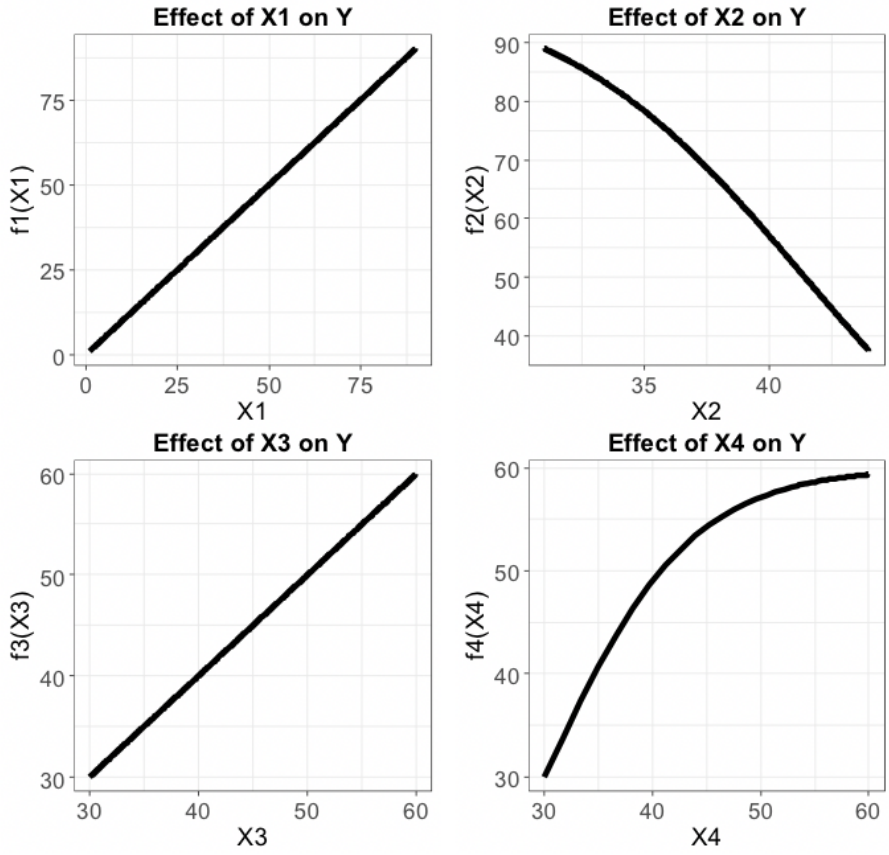

Figure 4.20. Effect Functions of Cause Variables on Y in Data-set 3.

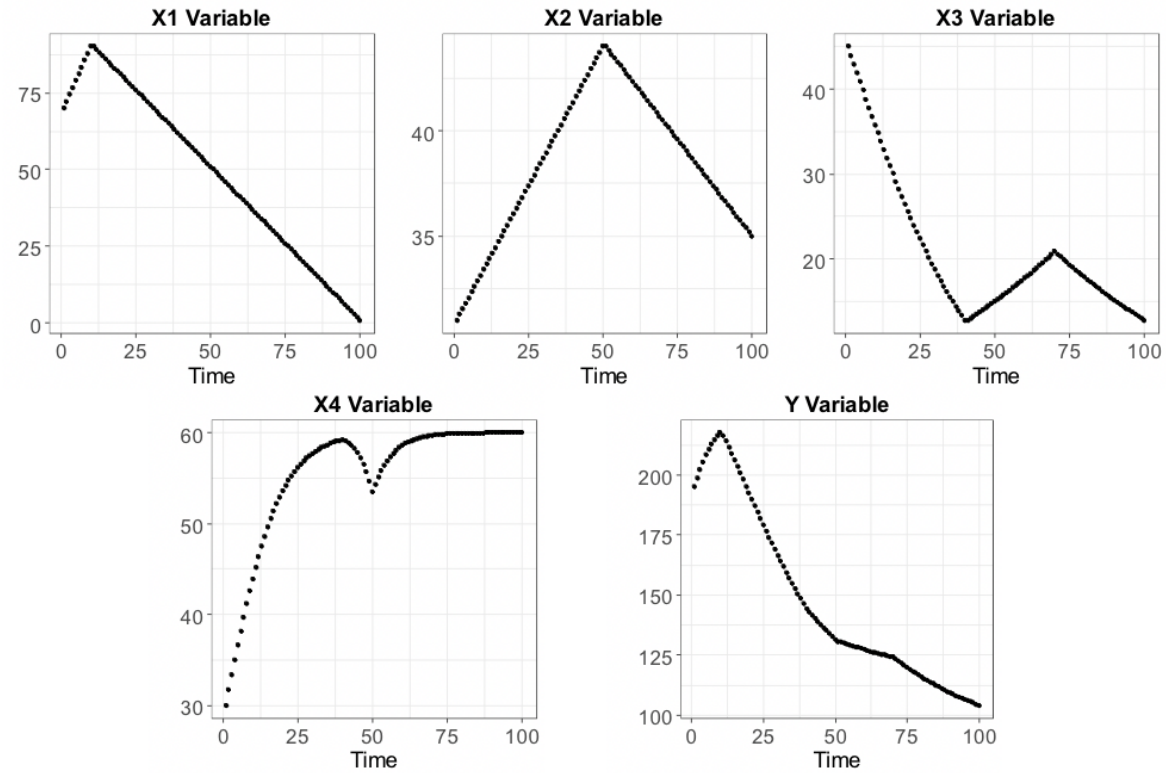

Figure 4.21. Data-set 3.

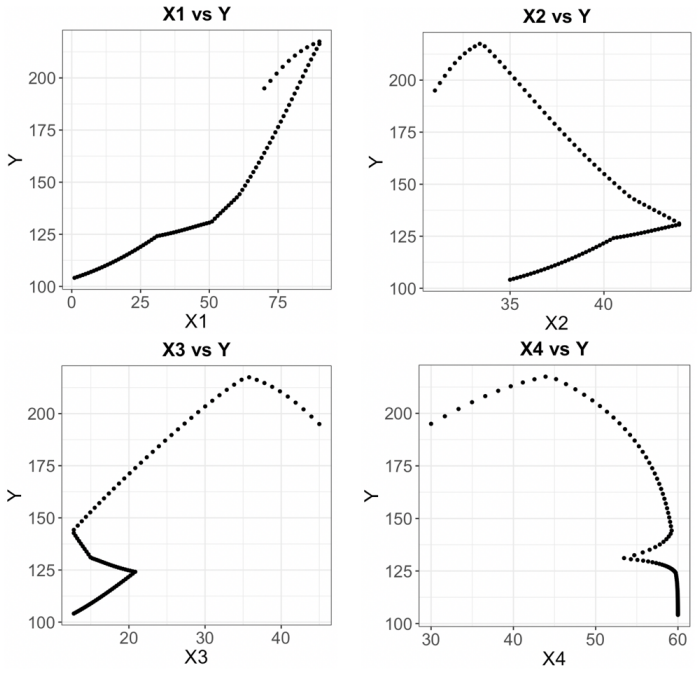

Figure 4.22. Cross-Sectional Graphs of Data-set 3.

Table 4.19. Spearman Correlation Matrix of Data-set 3.

	x1	x2	x3	x4	y
x1	1.0000	-0.2552	0.5985	-0.9159	0.9949
x2	-0.2552	1.0000	-0.5132	0.1710	-0.2736
x3	0.5985	-0.5132	1.0000	-0.6195	0.6120
x4	-0.9159	0.1710	-0.6195	1.0000	-0.9308
y	0.9949	-0.2736	0.6120	-0.9308	1.0000

The result of the analysis of the data-set 3 with *discoverpolarity* algorithm is shown in Table 4.20 and Table 4.21. Table 4.20 and 4.21 show the results of the first phase and second phase, respectively. In this analysis, variable importances are given as one (default value of *varImp*) and only the upper limit of X4 variable is specified (as 55). In the first phase, the algorithm eliminates nine corners. In the second phase, the algorithm eliminates eight other corners. Therefore, the only left option is (+, -, +, +). From the effect functions in Figure 4.20, it can be seen that this is the

correct conclusion. However, the modeler may prefer to check the observed differences to support the elimination of the corners (+, -, -, -) and (+, -, -, +) since number of observed differences to support these eliminations are only one and three, respectively. It is critical to mention that threshold values are chosen as 0.05 for this data-set. When the default value of 0.15 is used, the algorithm eliminates all the corners which means that the default threshold has a very high value for this data-set.

Table 4.20. Results of the First Phase of the Algorithm in Data-set 3.

Result(s) (X1, X2, X3, X4)	Eliminated Options (X1, X2, X3, X4)	Number of Observed Differences to Support the Elimination
(+, -, -, -)	(-,-, -, -)	1104
(+, +, -, -)	(-,+, -, -)	593
(+, -, +, -)	(-,-, +, -)	901
(+, +, +, -)	(-,+, +, -)	352
(+, -, -, +)	(-,-, -, +)	1167
(+, -, +, +)	(-,+, -, +)	2308
(+, +, +, +)	(+,+, -, +)	68
	(-,-, +, +)	935
	(-,+, +, +)	314

Through partial Spearman correlation, we obtain the following correlation coefficients with respective P-values which are close to zero:

$$(\rho_{X1Y.X2X3X4}, \rho_{X2Y.X1X3X4}, \rho_{X2Y.X1X3X4}, \rho_{X2Y.X1X3X4}) = (0.99, -0.25, 0.63, -0.91)$$

The Spearman correlation coefficient does not return the true polarity of causal relations. Our algorithm gives correct results where the correlation coefficient fails to return the correct polarity of causal relations.

Table 4.21. Results of the Second Phase of the Algorithm in Data-set 3.

Result(s) (X1, X2, X3, X4)	Eliminated Options (X1, X2, X3, X4)	Number of Observed Differences to Support the Elimination
(+, -, +, +)	(-,-, -, -)	1306
	(-,+, -, -)	787
	(-,-, +, -)	1147
	(-,+, +, -)	428
	(-,-, -, +)	1363
	(-,+, -, +)	2447
	(+,+, -, +)	268
	(-,-, +, +)	1122
	(-,+, +, +)	391
	(+, -, -, -)	1
	(+, +, -, -)	166
	(+, -, +, -)	10
	(+, +, +, -)	108
	(+, -, -, +)	3
	(+, +, +, +)	108

In data-set 3, we do not have strong evidence to eliminate (+, -, -, -) and (+, -, -, +) corners. Additional data points can support the elimination of these corners. Let us assume that we are given 50 additional data points which are uniformly distributed in the causal domain. In Figure 4.23, cross-sectional graphs of variables in data-set 3 with additional data points are given. With the additional points, the correlation of cause variables decreases (see Table 4.22). In the analysis of this "extended data-set 3", there are extra 233 and 93 point differences that support the elimination of (+, -, -, -) and (+, -, -, +) corners. Thanks to the additional 50 points, the algorithm finds the correct unique solution in the first phase. We can confidently eliminate corners of (+, -, -, -) and (+, -, -, +) (see Table 4.23). It is important to note that Spearman correlation analysis does not return the correct solution in this case even with the additional data

points.

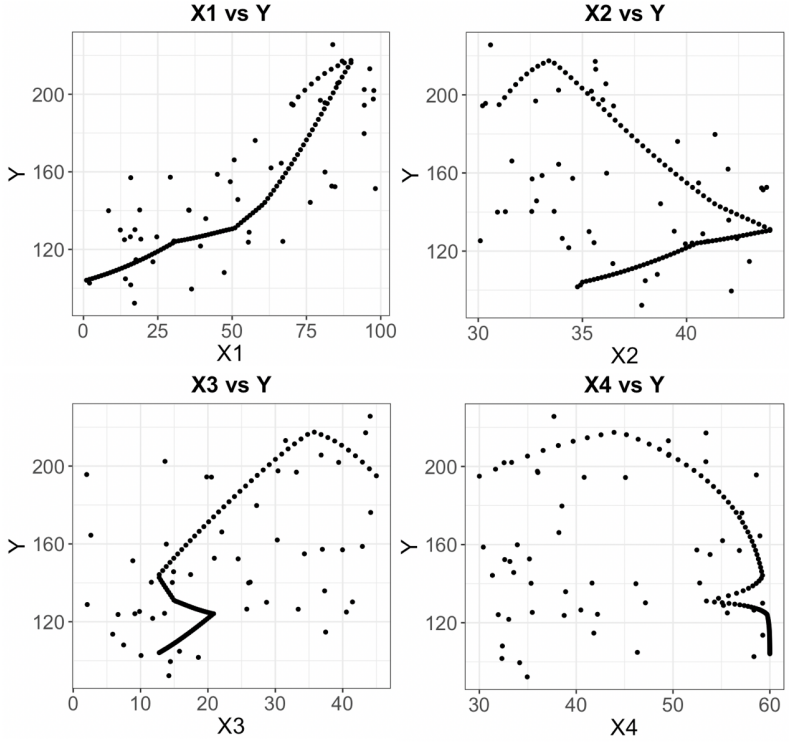

Figure 4.23. Data-set 3 with Uniformly Distributed 50 Additional Points.

Table 4.22. Spearman Correlation Matrix of Extended Data-set 3.

	x1	x2	x3	x4	y
x1	1.0000	-0.1129	0.3610	-0.5857	0.8993
x2	-0.1129	1.0000	-0.2842	0.2858	-0.3343
x3	0.3610	-0.2842	1.0000	-0.3695	0.5650
x4	-0.5857	0.2858	-0.3695	1.0000	-0.5666
y	0.8993	-0.3343	0.5650	-0.5666	1.0000

Table 4.23. Results of the First Phase in Data-set 3 with Additional Points.

Result(s) (X1, X2, X3, X4)	Eliminated Options (X1, X2, X3, X4)	Number of Observed Differences to Support the Elimination
(+, -, +, +)	(-,-, -, -)	1991
	(-,+, -, -)	1080
	(-,-, +, -)	1360
	(-,+, +, -)	837
	(-,-, -, +)	1985
	(-,+, -, +)	3717
	(+,+, -, +)	536
	(-,-, +, +)	1310
	(-,+, +, +)	1097
	(+, -, -, -)	234
	(+, +, -, -)	380
	(+, -, +, -)	30
	(+, +, +, -)	111
	(+, -, -, +)	96
	(+, +, +, +)	83

4.3.4. Data-set 4: Additive Formulation with Four Cause Variables

Let us consider another data-set where Y variable has the following function

$$Y = Y_{ref} + f_1(X_1) + f_2(X_2) + f_3(X_3) + f_4(X_4)$$

Effects of causal variables on Y are given in the Figure 4.24. From the effect functions, it is observed that X_1, X_2, and X_3 have negative while X_4 has positive effects on Y variable. Therefore, the true corner where the maximum Y variable is located is (-, -, -, +). The data-set that is used to discover the causal polarities has 100 samples as it is shown in Figure 4.25 and 4.26. The correlation matrix of data-set 4 is given in Table 4.24.

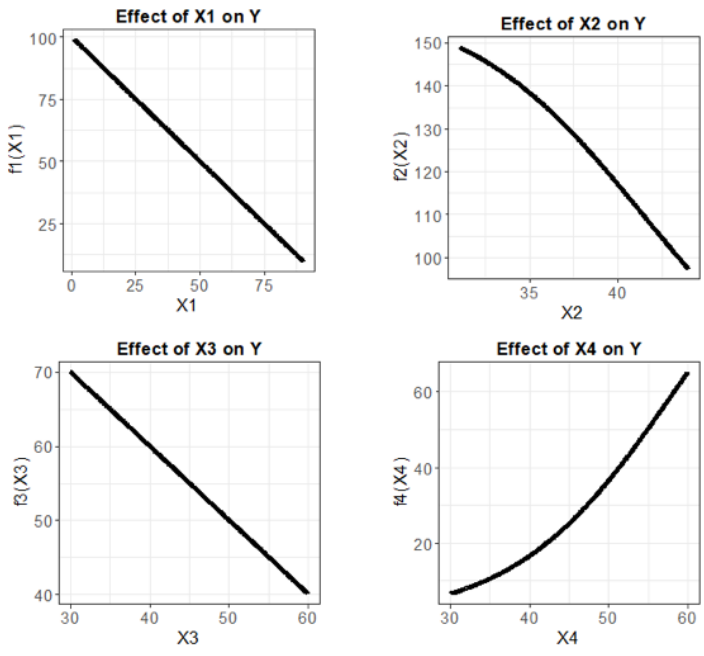

Figure 4.24. Effect Functions of Cause Variables on Y in Data-set 4.

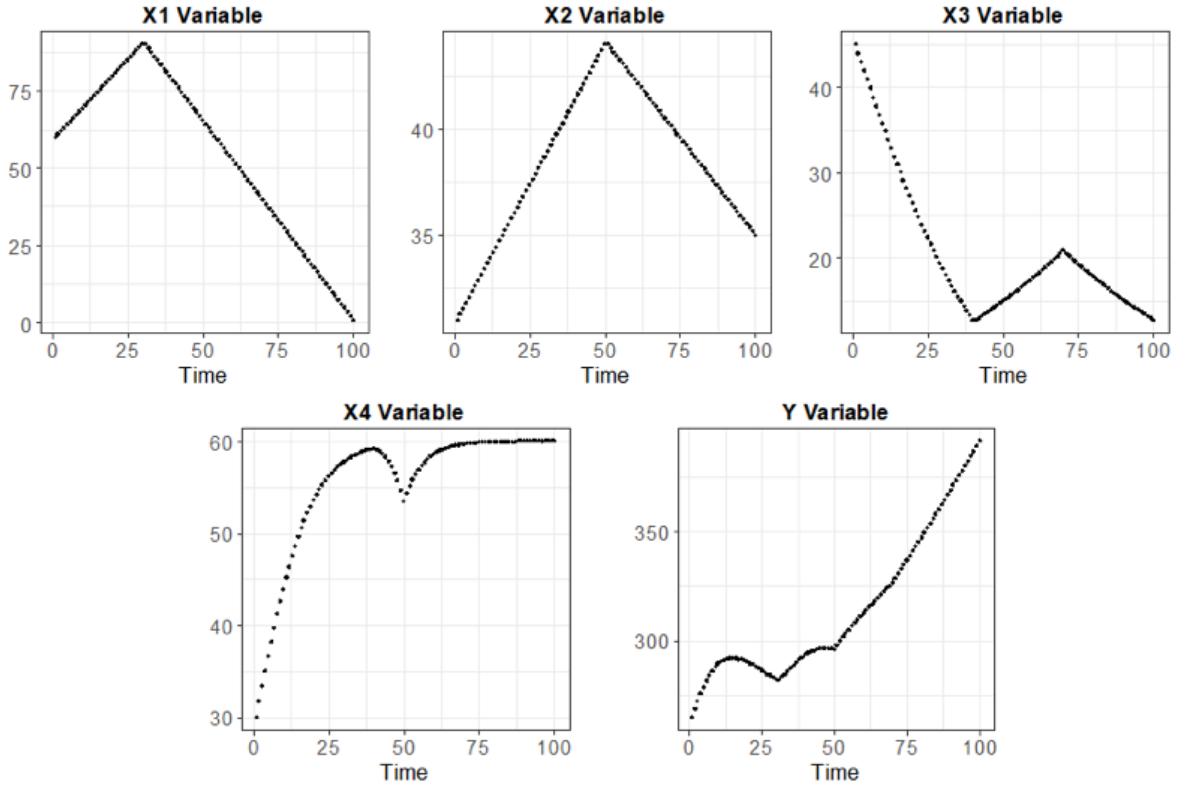

Figure 4.25. Data-set 4.

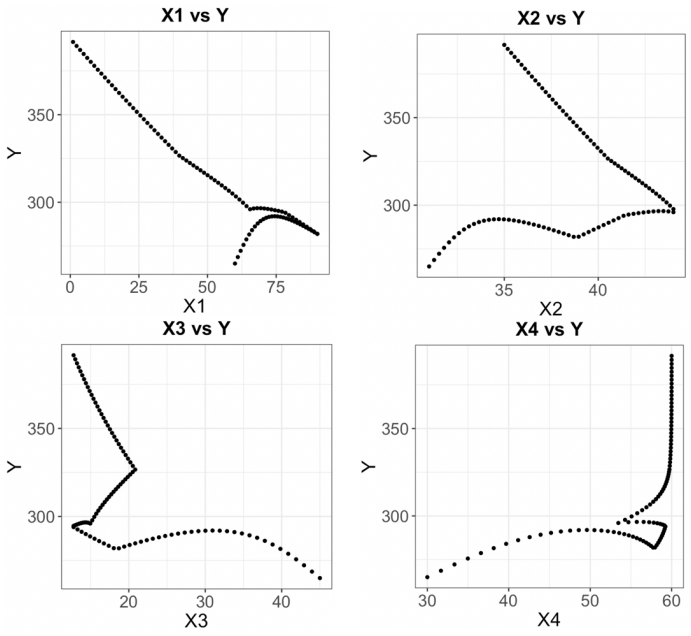

Figure 4.26. Cross-Sectional Graphs of Data-set 4.

Table 4.24. Spearman Correlation Matrix of Data-set 4.

	x1	x2	x3	x4	y
x1	1.0000	0.0709	0.2626	-0.6982	-0.8863
x2	0.0709	1.0000	-0.5132	0.1710	0.1769
x3	0.2626	-0.5132	1.0000	-0.6195	-0.5201
x4	-0.6982	0.1710	-0.6195	1.0000	0.8579
y	-0.8863	0.1769	-0.5201	0.8579	1.0000

The result of the analysis of the data-set 4 with *discoverpolarity* algorithm is shown in Table 4.25 and Table 4.26. Table 4.25 and 4.26 show the results of the first phase and second phase, respectively. In this analysis, variable importances are given as one (default value of *varImp*);no variable limit is specified; and the default threshold values are used. In the first phase, the algorithm eliminates nine corners. In the second phase, the algorithm eliminates four other corners. Therefore, three corners are left as the possible polarities: (-, -, -, -), (-, -, -, +), and (-, +, +, +). From the effect functions in Figure 4.24, it can be seen that the correct signs are (-, -, -, +) which is one of the three left options (The default threshold values are used). The modeler must choose one of the three possibilities for this data-set. She can either use her a priori knowledge or she can try to obtain more data to eliminate two incorrect possibilities.

Through partial Spearman correlation, we obtain the following correlation coefficients with respective P-values which are close to zero:

$$(\rho_{X1Y.X2X3X4}, \rho_{X2Y.X1X3X4}, \rho_{X2Y.X1X3X4}, \rho_{X2Y.X1X3X4}) = (0.89, -0.18, -0.54, 0.83)$$

The Spearman correlation coefficient does not return the true polarity of causal relations. Even though our algorithm gives multiple results, the true polarity is at least one of them. When the number of cause variables increases, our algorithm tends to return more possibilities since it becomes unlikely for data to cover all the causal

domain. However, when the number of cause variables is increases, the Spearman correlation analysis tends to return wrong polarities because of the strong linearity assumption among the ranked values of the variables.

Table 4.25. Results of the First Phase of the Algorithm in Data-set 4.

Result(s) (X1, X2, X3, X4)	Eliminated Options (X1, X2, X3, X4)	Number of Observed Differences to Support the Elimination
(-, -, -, -)	(+,-, -, -)	149
(-, +, -, -)	(+,+, -, -)	935
(-, -, -, +)	(-,-, +, -)	479
(-, +, -, +)	(+,-, +, -)	1555
(-, -, + +)	(-,+, +, -)	8
(-, +, +, +)	(+,+, +, -)	1167
(+, +, +, +)	(+,-, -, +)	197
	(+,+, -, +)	322
	(+,-, +, +)	147

Table 4.26. Results of the Second Phase of the Algorithm in Data-set 4.

Result(s) (X1, X2, X3, X4)	Eliminated Options (X1, X2, X3, X4)	Number of Observed Differences to Support the Elimination
(-, -, -, -)	(+, +, +, +)	962
(-, -, -, +)	(+, -, -, +)	411
(-, +, +, +)	(+, -, +, +)	326
	(- +, -, -)	104
	(-, -, +, -)	774
	(-, +, +, -)	60
	(-, +, -, +)	328
	(+, +, -, +)	1420
	(-, -, +, +)	28
	(+, -, -, -)	353
	(+, +, -, -)	1365
	(+, -, +, -)	1786
	(+, +, +, -)	1635

The algorithm does not return a unique solution for data-set 4. But with the help of additional data points it can achieve this goal. Let us assume that we are given 20 additional data points which are uniformly distributed in the causal domain. In Figure 4.27, cross-sectional graphs of variables in data-set 4 with additional data points are given. With the additional points, the correlation of cause variables decreases (see Table 4.27). In the analysis of this "extended data-set 4", the algorithm finds the correct unique solution in the first phase (see Table 4.28). It is important to note that Spearman correlation analysis does not return the correct solution even with the additional data points.

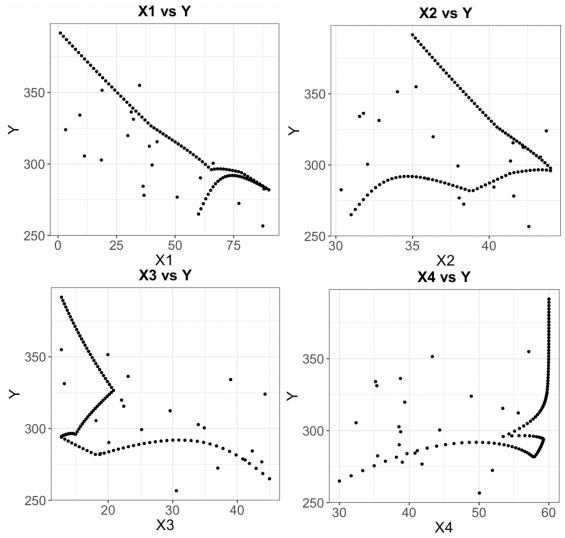

Figure 4.27. Data-set 4 with Uniformly Distributed 20 Additional Points.

Table 4.27. Spearman Correlation Matrix of Extended Data-set 4.

	x1	x2	x3	x4	y
x1	1.0000	0.1216	0.2005	-0.4476	-0.8429
x2	0.1216	1.0000	-0.4270	0.2600	0.0421
x3	0.2005	-0.4270	1.0000	-0.6309	-0.5246
x4	-0.4476	0.2600	-0.6309	1.0000	0.7165
y	-0.8429	0.0421	-0.5246	0.7165	1.0000

Table 4.28. Results of the First Phase of the Algorithm in Data-set 4 with Additional Points.

Result(s) (X1, X2, X3, X4)	Eliminated Options (X1, X2, X3, X4)	Number of Observed Differences to Support the Elimination
(-, -, -, +)	(-,-, -, -)	13
	(+, -, -, -)	243
	(-,+, -, -)	29
	(+,+, -, -)	1035
	(-,-, +, -)	711
	(+, -, +, -)	2025
	(-,+, +, -)	177
	(+, +, +, -)	1468
	(+,-, -, +)	255
	(-, +, -, +)	10
	(+, +, -, +)	691
	(-, -, +, +)	2
	(+, -, +, +)	222
	(-, +, +, +)	19
	(+, +, +, +)	249

4.3.5. Data-set 5: Hybrid Formulation with Four Cause Variables

Let us consider another data-set where Y variable has the following function:

$$Y = Y_{ref} * f_4(X_4) * (f_1(X_1) + f_2(X_2) + f_3(X_3))$$

Effects of causal variables on Y are given in the Figure 4.28. From the effect functions, it is observed that X_1, X_2, and X_3 have negative while X_4 has positive effect on Y variable. Therefore, the true corner where the maximum Y variable is located is

(-, -, -, +). The data-set that is used to discover the causal polarities has 100 samples as it is shown in Figure 4.29 and 4.30. The correlation matrix of data-set 5 is given in Table 4.29.

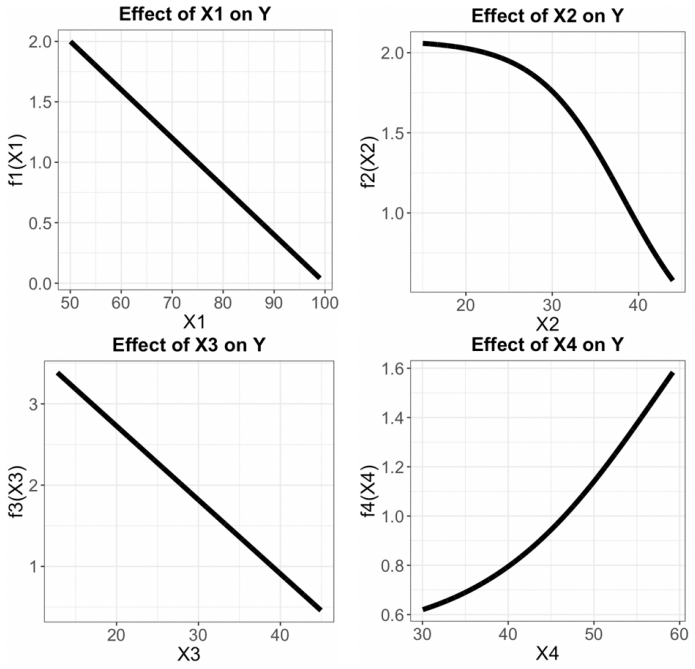

Figure 4.28. Effect Functions of Cause Variables on Y in Data-set 5.

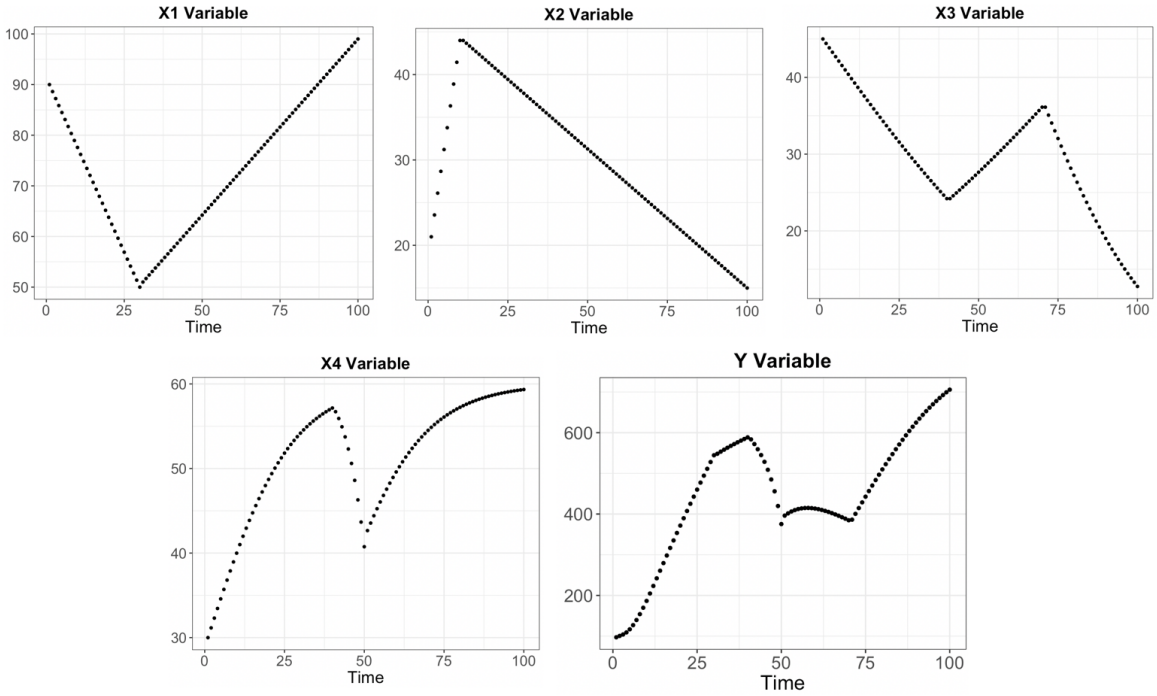

Figure 4.29. Data-set 5.

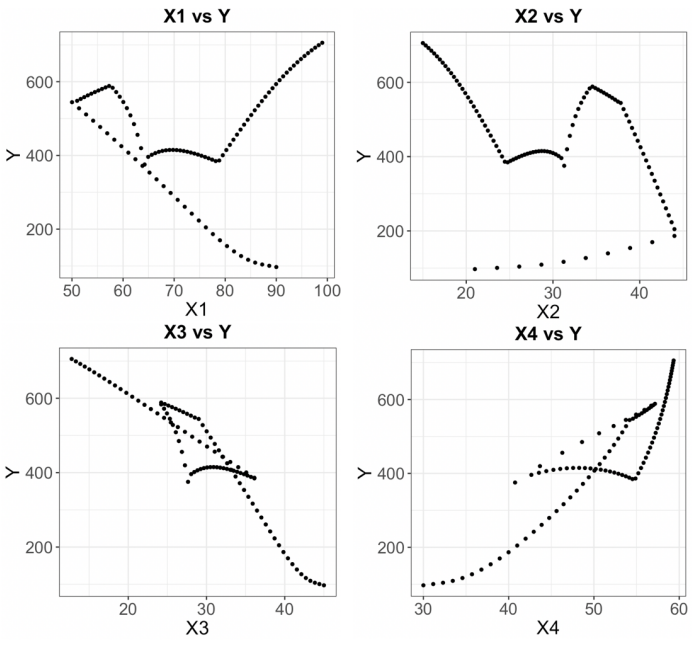

Figure 4.30. Cross-Sectional Graphs of Data-set 5.

Table 4.29. Spearman Correlation Matrix of Data-set 5.

	x1	x2	x3	x4	y
x1	1.0000	-0.7644	-0.1085	0.3109	0.0723
x2	-0.7644	1.0000	0.5017	-0.6266	-0.4687
x3	-0.1085	0.5017	1.0000	-0.7949	-0.9477
x4	0.3109	-0.6266	-0.7949	1.0000	0.8875
y	0.0723	-0.4687	-0.9477	0.8875	1.0000

The result of the analysis of the data-set 5 with *discoverpolarity* algorithm is shown in Table 4.30 and Table 4.31. Table 4.30 and 4.31 show the results of the first phase and second phase, respectively. In this analysis, variable importances are given as one (default value of *varImp*) and the default threshold values are used. Moreover, upper limits of X2 and X4 variables are set as 35 and 55, respectively. In the first phase, the algorithm eliminates twelve corners. In the second phase, the algorithm eliminates three other corners. Therefore, the only left corner as the possible polarity is (-, -, -, +). From the effect functions in Figure 4.28, it can be seen that the correct signs are (-, -, -, +) which is the only left option. Thus, it can be said that *discoverpolarity* is capable of obtaining correct polarities even for complex hybrid formulations when the inputs of the algorithm are properly selected.

Table 4.30. Results of the First Phase of the Algorithm in Data-set 5.

Result(s) (X1, X2, X3, X4)	Eliminated Options (X1, X2, X3, X4)	Number of Observed Differences to Support the Elimination
(-, -, -, -)	(+, -, -, -)	171
(+, +, -, -)	(-,+, -, -)	183
(-, -, -, +)	(-, -, +, -)	52
(-, +, -, +)	(+, -, +, -)	1373
(+, +, -, +)	(-, +, +, -)	2254
	(+, +, +, -)	887
	(+, -, -, +)	68
	(-, -, +, +)	15
	(+, -, +, +)	511
	(-, +, +, +)	653
	(+, +, +, +)	41

Table 4.31. Results of the Second Phase of the Algorithm in Data-set 5.

Result(s) (X1, X2, X3, X4)	Eliminated Options (X1, X2, X3, X4)	Number of Observed Differences to Support the Elimination
(-, -, -, +)	(-, -, -, -)	15
(+, +, -, +)	(+,-, -, -)	4455
	(-, +, -, -)	543
	(+, +, -, -)	23
	(-, -, +, -)	83
	(+, -, +, -)	1641
	(-, +, +, -)	2491
	(+, +, +, -)	1124
	(+, -, -, +)	412
	(-, +, -, +)	81
	(-, -, +, +)	15
	(+, -, +, +)	776
	(-, +, +, +)	1118
	(+, +, +, +)	67

Through partial Spearman correlation, we obtain the following correlation coefficients with respective P-values which are close to zero. The Spearman correlation coefficient does not return the true polarity of causal relations.

$$(\rho_{X1Y.X2X3X4}, \rho_{X2Y.X1X3X4}, \rho_{X2Y.X1X3X4}, \rho_{X2Y.X1X3X4}) = (0.04, -0.47, -0.94, 0.88)$$

4.3.6. Data-set 6: Multiplicative Formulation with Three Cause Variables

Let us consider another data-set where Y variable has the following function:

$$Y = Y_{ref} * (f_1(X_1) * f_2(X_2) * f_3(X_3))$$

Effects of causal variables on Y are given in the Figure 4.31. From the effect functions, it is observed that X_1 and X_2 have positive while X_3 has negative effect on Y variable. Therefore, the true corner where the maximum Y variable is located is (+, +, -). The data-set that is used to discover the causal polarities has 100 samples as it is shown in Figure 4.32 and 4.33. The correlation matrix of data-set 6 is given in Table 4.32.

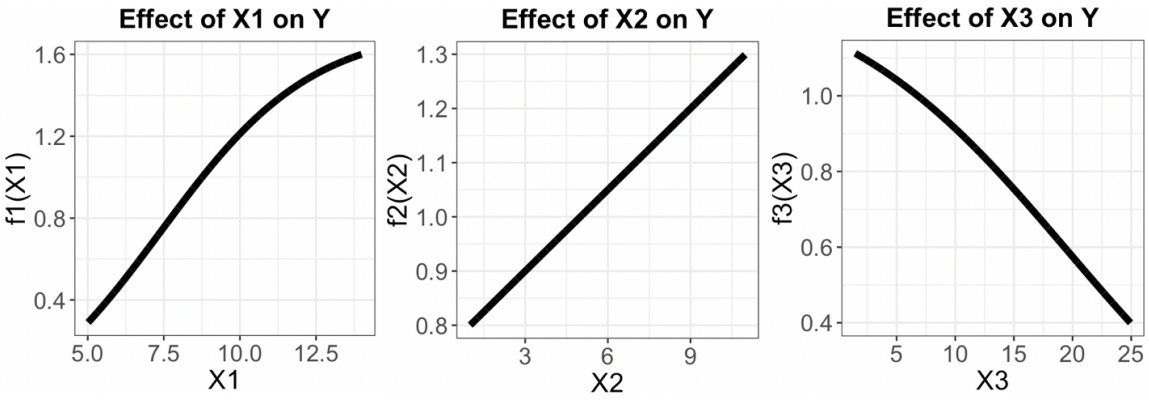

Figure 4.31. Effect Functions of Cause Variables on Y in Data-set 6.

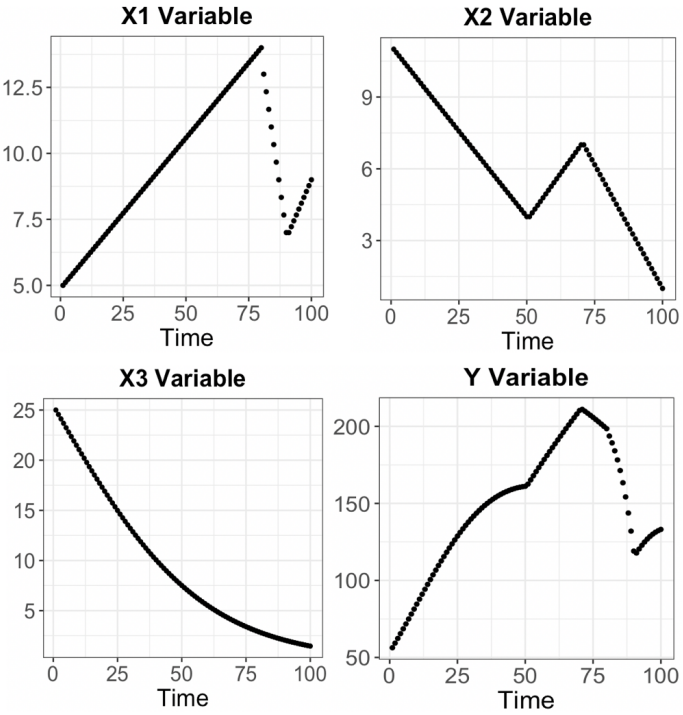

Figure 4.32. Data-set 6.

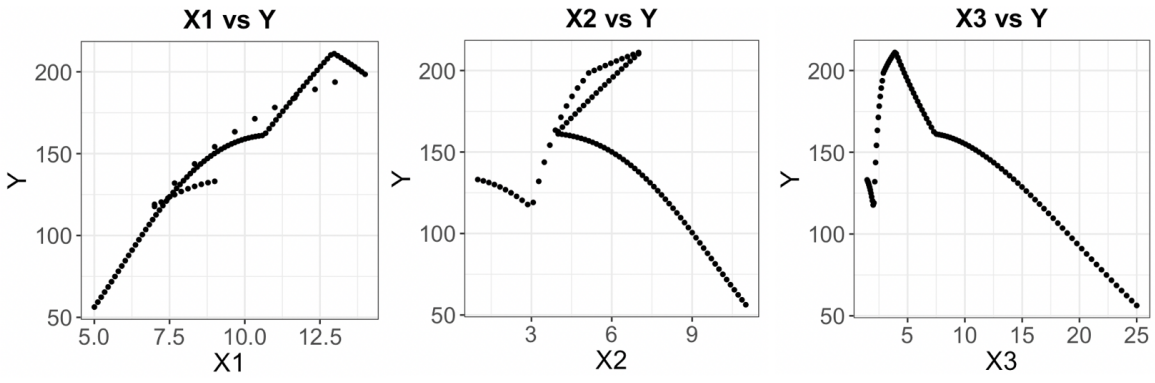

Figure 4.33. Cross-Sectional Graphs of Data-set 6.

Table 4.32. Spearman Correlation Matrix of Data-set 6.

	x1	x2	x3	y
x1	1.0000	-0.4137	-0.6037	0.9894
x2	-0.4137	1.0000	0.8291	-0.3769
x3	-0.6037	0.8291	1.0000	-0.5843
y	0.9894	-0.3769	-0.5843	1.0000

The result of the analysis of the data-set 6 with *discoverpolarity* algorithm is shown in Table 4.33 and Table 4.34. Table 4.33 and 4.34 show the results of the first phase and second phase, respectively. In this analysis, variable importances are given as one (default value of *varImp*); only the upper limit of X1 variable is set as 12.5; and the threshold values are set as 0.05. From the first phase, the algorithm eliminates six corners. In the second phase, the algorithm eliminates one more corner. Therefore, the only left corner as the possible polarity is (+, +, -). From the effect functions in Figure 4.31, it can be seen that the correct signs are (+, +, -) which is the only left option. It is critical to mention that threshold values are chosen as 0.05 for this data-set. When the default value of 0.15 is used, the algorithm eliminates all the corners which means that the default threshold is a very high value for this data-set. The results show that *discoverpolarity* is capable of obtaining meaningful results for multiplicative formulations when the input variables are properly selected.

Table 4.33. Results of the First Phase of the Algorithm in Data-set 6.

Result(s) (X1, X2, X3)	Eliminated Options (X1, X2, X3)	Number of Observed Differences to Support the Elimination
(-, +, -)	(-, -, -)	1055
(+, +, -)	(+, -, -)	141
	(-, -, +)	735
	(+, -, +)	15
	(-, +, +)	3078
	(+, +, +)	40

Table 4.34. Results of the Second Phase of the Algorithm in Data-set 6.

Result(s) (X1, X2, X3)	Eliminated Options (X1, X2, X3)	Number of Observed Differences to Support the Elimination
(+, +, -)	(-, -, -)	1095
	(+, -, -)	302
	(-, +, -)	13
	(-, -, +)	910
	(+, -, +)	16
	(-, +, +)	3191
	(+, +, +)	104

Through partial Spearman correlation, we obtain the following correlation coefficients with respective P-values which are close to zero. The Spearman correlation coefficient does not return the true polarity of causal relations.

$$(\rho_{X1Y.X2X3}, \rho_{X2Y.X1X3}, \rho_{X2Y.X1X3}) = (0.99, -0.37, -0.61)$$

4.3.7. A Special Data-set 7: Additive Formulation with Three Cause Variables

Let us consider another data-set where Y variable has the following function:

$$Y = Y_{ref} * (f_1(X_1) + f_2(X_2) + f_3(X_3))$$

Effects of causal variables on Y are given in the Figure 4.34. From the effect functions, it is observed that X_1 and X_2 have positive while X_3 has negative effect on Y variable. Therefore, the true corner where the maximum Y variable is located is (+, +, -). The data-set that is used to discover the causal polarities has 40 samples as it is shown in Figure 4.35 and 4.36. As it is seen in data-set 7 graphs and Table 4.35, there is a perfect correlation in the data-set. Even the effect function shapes are close to each other. This data-set represents a poor data-set where algorithm concludes many possible results instead of a unique one. In such cases where the variables are perfectly correlated, the data-set does not give sufficient information to algorithm to extract effect polarities.

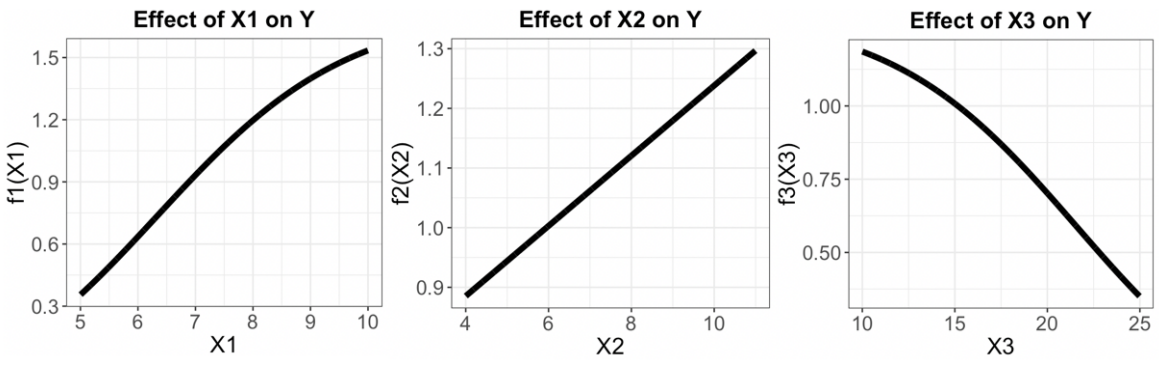

Figure 4.34. Effect Functions of Cause Variables on Y in Data-set 7.

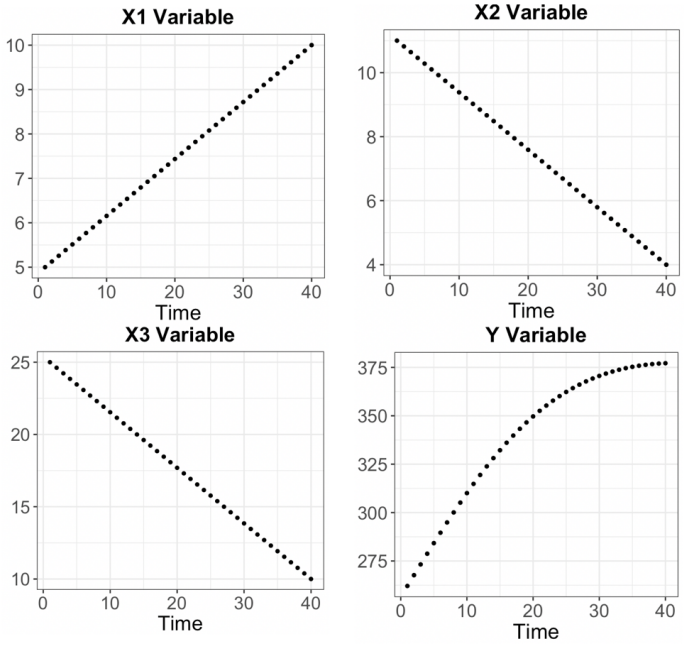

Figure 4.35. Data-set 7.

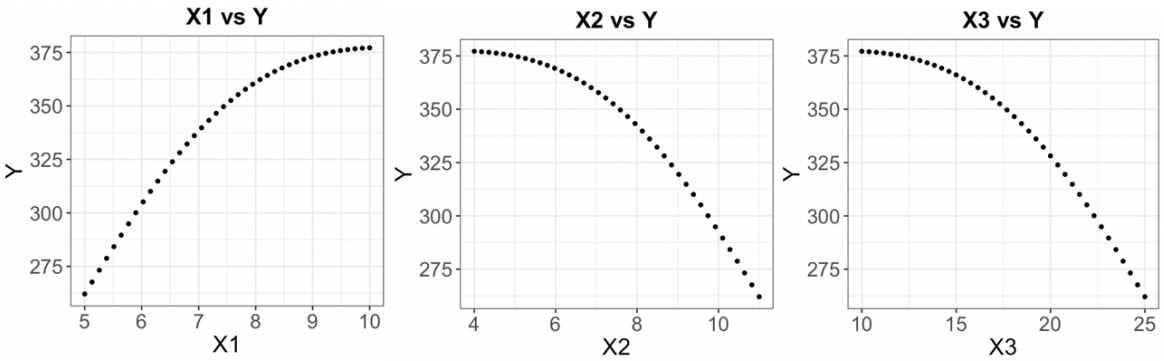

Figure 4.36. Cross-Sectional Graphs of Data-set 7.

Table 4.35. Spearman Correlation Matrix of Data-set 7.

	x1	x2	x3	y
x1	1.0000	-1.0000	-1.0000	1.0000
x2	-1.0000	1.0000	1.0000	-1.0000
x3	-1.0000	1.0000	1.0000	-1.0000
y	1.0000	-1.0000	-1.0000	1.0000

The result of the analysis of the data-set 7 with *discoverpolarity* algorithm is shown in Table 4.36 and Table 4.37. Table 4.36 and 4.37 show the results of the first phase and second phase, respectively. In this analysis, variable importances are given as one (default value of *varImp*); no variable limit is specified; and the default threshold values are used. From the first phase, the algorithm eliminates only one corner. In the second phase, the algorithm eliminates only one more corner because of the perfect collinearity among variables. Therefore, the algorithm gives us six of the eight possible options. The algorithm does not return useful information for this data-set because the data do not cover many independent points of the causal domain. The data must include at least a few data points that are not interdependent so that algorithm can move in different directions in the causal domain to eliminate corner points.

Through partial Spearman correlation, we cannot obtain correlation coefficients because of multicollinearity.

Table 4.36. Results of the First Phase of the Algorithm in Data-set 7.

Result(s) (X1, X2, X3)	**Eliminated Options (X1, X2, X3)**	**Number of Observed Differences to Support the Elimination**
(-, -, -)	(-, +, +)	780
(+, -, -)		
(-, +, -)		
(+, +, -)		
(-, -, +)		
(+, -, +)		
(+, +, +)		

Table 4.37. Results of the Second Phase of the Algorithm in Data-set 7.

Result(s) (X1, X2, X3)	**Eliminated Options (X1, X2, X3)**	**Number of Observed Differences to Support the Elimination**
(-, -, -)	(+, -, -)	83
(-, +, -)	(-, +, +)	780
(+, +, -)		
(-, -, +)		
(+, -, +)		
(+, +, +)		

Collinearity spectrum of data-sets has two end points: "fully randomly selected" data-set and "perfectly correlated" data-set. Real-life data-sets are somewhere in between, close to "perfectly correlated" end point. In seven data-sets that have high collinearities, our algorithm results are compared with Spearman correlation analysis. For the simplest data-set with two cause variables, both *discoverpolarity* and correlation analysis return the true causal polarities. In the second multiplicative case, *discoverpolarity* returns the true polarities whereas Spearman correlation analysis fails

to return any conclusion because of multicollinearity among cause variables. In the third data-set with four additive effects, *discoverpolarity* returns the true polarities whereas Spearman correlation analysis returns wrong causal polarities. In the fourth case with four additive effects, *discoverpolarity* is able to reduce the possible polarity set to three, one of which corresponds to correct polarities. Correlation analysis on the other hand returns wrong causalities in this case. In the fifth data-set with four hybrid effects, *discoverpolarity* is able to reduce the possible polarity set to two, one of which corresponds to correct polarities. Correlation analysis on the other hand returns wrong causalities in this case. In the sixth case with three multiplicative effects, *discoverpolarity* returns the true polarities whereas Spearman correlation analysis returns wrong causal polarities. In the last data-set with three additive effects, there is perfect collinearity among variables. Therefore, the data-set is at the "perfectly correlated" end point. In this data-set, *discoverpolarity* is only able to reduce the possible polarity set to six from eight, one of which corresponds to correct polarities. It cannot eliminate many possibilities when the data-sets and effect functions have perfect multicollinearity. Correlation analysis on the other hand fails to return any conclusion because of multicollinearity among cause variables.

When we introduce new randomly selected data points in the data-sets above (see data-set 1, 2, 3, and 4), correlations among variables decrease. Therefore, the data-set approximates the "fully randomly selected" end point. In this case, *discoverpolarity* tends to return the unique correct conclusion with more confidence. However, obtaining correct answer by correlation analysis is still not guaranteed with such enhanced data, even though correlation analysis tends to return the correct conclusion.

5. DISCOVERING STOCK VARIABLES

5.1. Methodology

The focus of the section is discovering stock variables using a known CLD and historical data. In other words, the stock variables are identified by applying the curve fitting method to data about the variables in CLD (see Appendix C). The identification of stock variables can also be considered as discovering simplified SFD from a known CLD. The reason is that we define the simplified SFD by specifying stock variables of a CLD. Therefore, we automatically obtain the simplified SFD by identifying stock variables in a CLD.

Briefly, possible monotonic relations are fitted to "$x \rightarrow y$", "$x \rightarrow dy/dt$", and "$dx/dt \rightarrow dy/dt$" for each "$x \rightarrow y$" in CLD. To compare the fitted functions, root mean square error (RMSE) and mean absolute percentage error (MAPE) are selected as persormance measures since both measures return reasonable results for time-series data-sets [30]. The calculation of both measures are given below. In the formulas, A and F represent actual and forecast values, respectively. Since there are two error terms (MAPE and RMSE), our problem is multi-objective. Therefore, pareto analysis is applied to compare the results. The best fit is checked whether it is directly fitted to "$x \rightarrow y$", "$x \rightarrow dy/dt$", or "$dx/dt \rightarrow dy/dt$" for each "$x \rightarrow y$". The method is applied to three different models.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(A_i - F_i)^2}{n}}$$

$$MAPE = (\frac{100\%}{n}) \sum_{i=1}^{n} |\frac{(A_i - F_i)}{A_i}|$$

In Figure 5.1, the chart of research design can be seen. Firstly, we specify the SFD of a structure. Then, we generate a data-set from that structure and we specify the full

correct CLD of the structure. Afterwards, we fit monotonic relations to the generated data for each causal relation. Finally, we check whether the underlying structure is correctly identified or not.

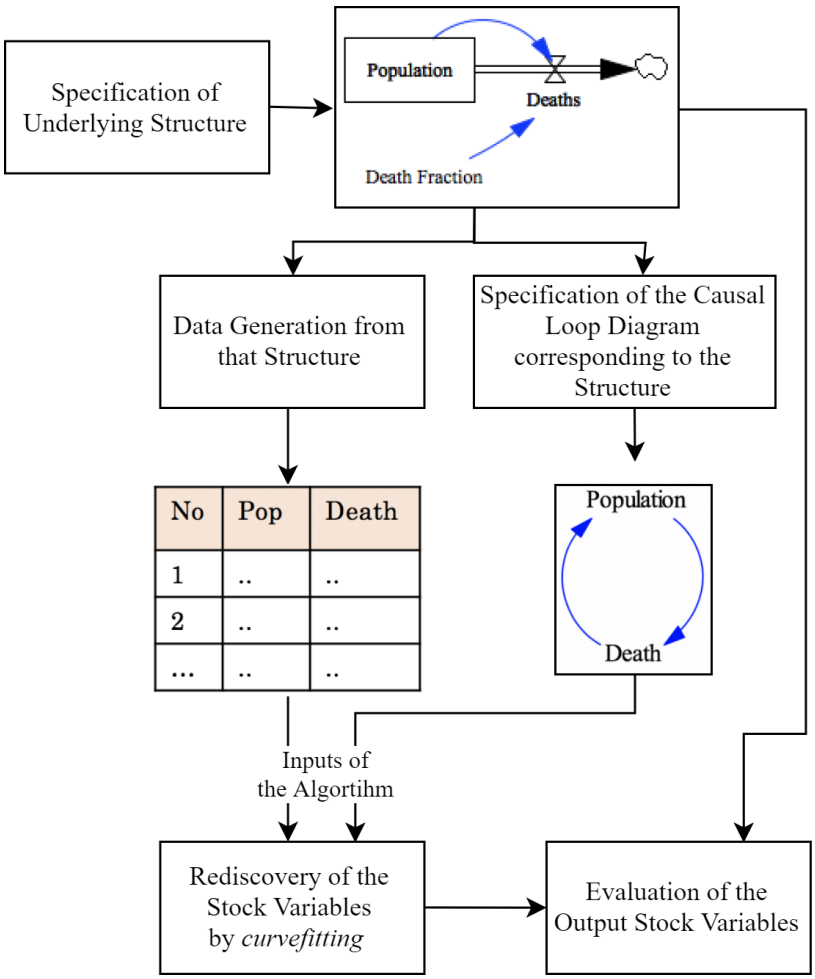

Figure 5.1. Research Design of Discovering Stock Variables.

The research rests on several assumptions. To discover the stock variables, we first assume that there is a known CLD where each effect variable is only influenced by one cause variable. Therefore, the analyzed effects are assumed *ceteris paribus*. For each feedback loop, we know that there must be at least one stock variable. Moreover, it is assumed that there is real-life dynamic data about all the system variables. However, simulation-generated 'synthetic' data is analyzed instead of actual real-life data to be able to evaluate the validity of the results obtained from alternative data tools. All data are generated by using Vensim model simulations after the specification of underlying

structures.

5.2. Algorithm: *Curvefitting*

5.2.1. The Basic Process of *Curvefitting*

Since the relationships are monotonic in SD models, the possible functional relations between two variables are limited: linear, exponential, and s-shaped relations. (see Table 5.1). Of course, there may be more complex monotonic relations, such as a piecewise combination of linear and exponential growth but they are excluded in this research. Limited number of functional shapes makes curve fitting easier to discover structure behind a behavior.

Briefly, data of the cause and effect variable is fitted to the functions in Table 5.1. To this aim, *lm, nls*, and *gnls* functions from packages *stats* and *nlme* used in R. Afterwards, for the fitted functions (not every function fits the data), performance measures are compared.

Table 5.1. Candidate Functional Forms Used in R for Curve Fitting.

Linear Relation	gnls(y ∼ a + b * x , start = c(a = 0, b = 1))
	lm(y ∼ x)
Exponential Relation	gnls(log(y) ∼ a + b * x, start = c(a = 0, b = 1))
S-shaped Relation	nls(y ∼ SSlogis(x, Asym, xmid, scal))
	gnls(y ∼ Asym/(1 + exp((xmid - x)/scal)), start =coef(above function))
	nls(y ∼ SSasymp(x, Asym, R0, lrc))
	gnls(y ∼ max(y)/(1 + exp((xmid - x)/scal)), start = c(xmid = 1,scal = 1))
	gnls(y ∼ max(y) - (max(y) - min(y))/(1 + exp((xmid - x)/scal)), start = c(xmid = 1, scal = 1))
	gnls(y ∼ min(y),+ (max(y) - min(y))/(1 + exp((xmid - x)/scal)), start = c(xmid = 1, scal = 1))

5.2.2. Inputs and Outputs of *Curvefitting*

Curvefitting algorithm only takes one cause variable, x as a vector, and one effect variable, y as a vector. As outputs, it returns *sigma (RMSE), mape, method, function, relation* , and parameter values in the fitted relation. *Sigma* and *mape* are the error terms which are later used in pareto analysis. *Method* gives the name of the fitted functional shape such as linear or exponential function. *Function* gives the exact form of fitted function such as "$y \sim a+b*x$" or "$log(y) \sim a+b*x$". *Relation* represents which relation the function is fitted to: "$x \rightarrow y$", "$x \rightarrow dy/dt$", or "$dx/dt \rightarrow dy/dt$". As parameter values, the algorithm returns values of the parameters used in the functional form such as *intercept (a)* and *slope (b)* parameters in linear regression.

5.3. Experiments and Results in Discovering Stock Variables

5.3.1. Model with One Stock and Linear Relations: Population - Death Model

The first analysis is applied to population death model. In the algorithm, both population and death variables are considered as possible stock variables (see Figure 5.2 and Table 5.2). Basically, the formulations of Population= f(Death) and dPopulation/dt =f(Death) are calculated for "Death→ Population" link. Then, the formulations of Death= f(Population)and dDeath/dt= f(Population) are calculated for "Population→ Death" link (see Table 5.2).

Table 5.2. Curve Fitting for Population-Death Model.

Curve Fitting from Death to Population	**Curve Fitting from Population to Death**
(1) Population = f(Death)	(3) Death = f(Population)
(2) dPopulation/dt = f(Death)	(4) dDeath/dt = f(Population)

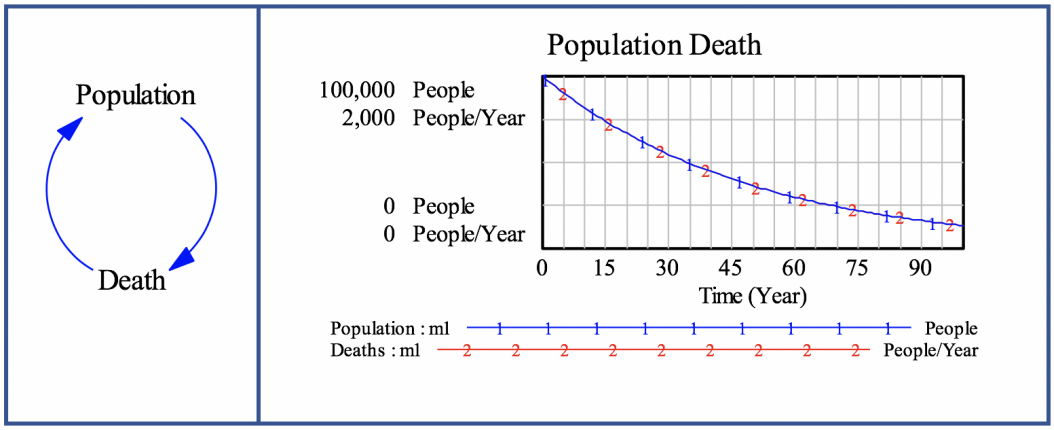

Figure 5.2. CLD and Behavior of Population-Death Model.

In Table 5.3 and Table 5.4, obtained results for the curve fitting from death to population and from population to death are given respectively.

Table 5.3. Curve Fitting Results from Death(x) to Population(y).

No	RMSE	MAPE	Method	Relation
1	**0.0339**	**0.0001**	**Linear Function**	**X to dY**
2	**0.0967**	**0.0001**	**Linear Function**	**X to Y**
3	2318.4037	3.2818	Exponential Function	X to dY
4	8422.5490	13.0940	Exponential Function	X to Y
5	15571.5583	4.6847	S-shaped Function	X to Y

Table 5.4. Curve Fitting Results from Population(x) to Death(y).

No	RMSE	MAPE	Method	Relation
1	**0.00118**	**0.00010**	**Linear Function**	**X to dY**
2	**0.00193**	**0.00014**	**Linear Function**	**X to Y**
3	46.368	3.2818	Exponential Function	X to dY
4	168.451	13.0940	Exponential Function	X to Y
5	311.431	4.6847	S-shaped Function	X to Y

According to performance measures, first two rows of curve fitting results in Table 5.3 and 5.4 can be counted as best performances since there is not much difference in RMSE and MAPE scores. The best curve fitting equations with their parameters are given in Table 5.5.

Table 5.5. Curve Fitting for Population-Death Model.

Curve Fitting from Death to Population	**Curve Fitting from Population to Death**
(1) Population = 50*Death	(3) Death = 0.02*Population
(2) dPopulation/dt = -1*Death	(4) dDeath/dt = -0.0004*Population

Curvefitting perfectly fit monotonic relations to "$x \rightarrow y$" and "$x \rightarrow dy/dt$" when there are linear relations between two variables, "$x \rightarrow y$". Therefore, it returns both possibilities with the same errors. By keeping at least one variable as stock, we obtain 3 very different SFDs from the set of equations in Table 5.5 (see Figure 5.3). All three stock-flow diagrams return nearly the same population and death data. Obviously, only one of these SFDs is the correct SFD that generated the used data, the other two SFDs have nothing to do with the true structure.

We reach the conclusion that many SFDs can be discovered by data analysis for CLDs based on one stock variable and only linear relations because different SFDs with

linear relations can produce the same behavior. Without checking the variable units and/or logic rules for each cause relations [23], the correct SFD cannot be obtained from a CLD for very simple models with linear relations.

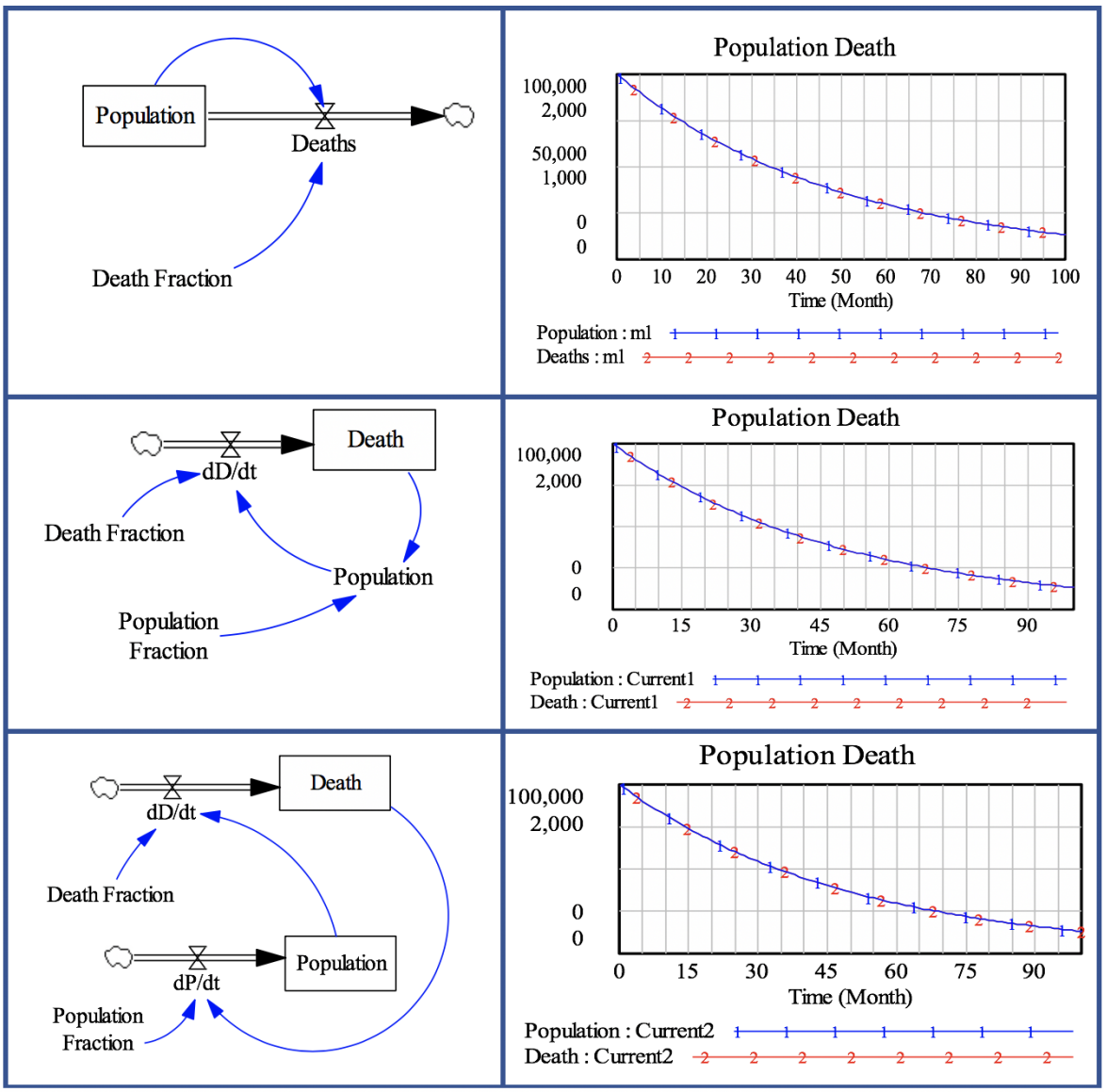

Figure 5.3. Alternative SFDs and Their Behaviors based on Population-Death Data.

After ending up with many stock-flow diagrams producing the same behaviour, the necessity of choosing the right stock variables may be questioned. In other words, if they produce the same behaviour, is it so crucial to decide which variable is the stock variable? Here, we do some analysis on the different SFDs to check the importance of choosing the correct stock variable.

The alternative SFD completely fails at the causality analysis. Actually, because of the irrational causality behind these SFDs, they can produce wrong behaviours in the further sensitivity analysis. For example, in the Alternative-1 Population Death SFD, Population is 50 * Death. It means that regardless of the past values of population, population is always be 50 times of death. The structure fails in when we make a sudden change in Death Fraction because of this illogical formula.

In Alternative-2 Population-Death SFD, population and death are both stock variables. The inaccuracy behind death being a stock variable is explained in the previous paragraph. In this structure, since only the rate of changes of variables are related to each other, the model can be run with irrational initial values. Moreover, death can have a higher value than population. This has no real-life equivalence since population cannot have negative values. Thus, wrong SFDs fail at logic tests and any possible further system analysis. Therefore, a proper selection of stock variables with respect to analyzed system is crucial for further system analysis.

5.3.2. Model with One Stock and Non-linear Relation

The second analysis is applied to the CLD in Figure 5.4 where both X and Y are candidates as possible stock variables (see Table 5.6). The SFD and the non-linear effect function are given in Figure 5.5. In this model, "y $\rightarrow$ x" relation is non-linear as it is seen in Figure 5.5 so that their behaviours are significantly different than each other.

Table 5.6. Curve Fitting for the Second Model.

Curve Fitting from X to Y	**Curve Fitting from Y to X**
(1) Y = f(X)	(3) X = f(Y)
(2) dY/dt = f(X)	(4) dX/dt = f(Y)

Curvefitting is applied for all the relations in Table 5.6 to discover stock variables. In other words, monotonic relations are fitted to each relation in Figure 5.6. For the

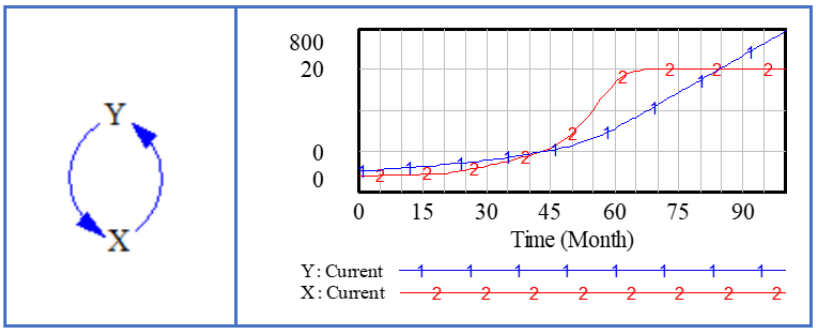

Figure 5.4. CLD and Behavior of the Second Model.

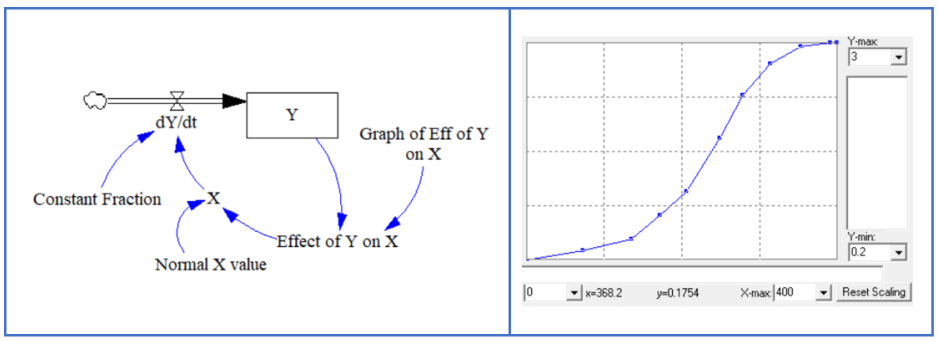

Figure 5.5. SFD and Non-linear Graphical Effect Function of Y on X.

causal link of x → y, *curvefitting* has the best fit to x → dy with MAPE = 0.119 and RMSE = 0.2802. The reason is that Figure 5.6.b is more "strongly monotonic" function than Figure 5.6.a. For the causal link of y → x, *curvefitting* has the best fit to y → x with MAPE = 6.8 and RMSE = 0.3736. The reason is that Figure 5.6.c is suitable for a monotonic function, but Figure 5.6.d is not.

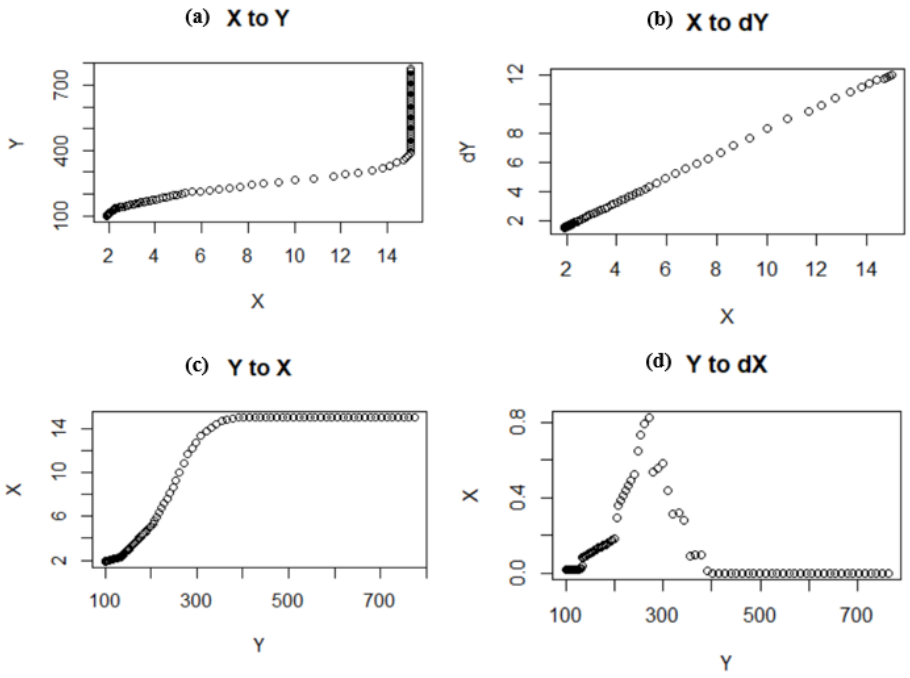

Figure 5.6. Relation Graphs in the Second Model.

Curvefitting results suggest that X variable affects Y variable through Y's rate of change whereas Y variable affects X variable directly. This means that there is only one stock variable, Y. The algorithm reaches the correct conclusion since only variable Y is the stock variable in the original model (Figure 5.5).

We reach the tentative conclusion that when at least one relation is non-linear in the system, the incorrect relations tend to become non-monotonic (see Figure 5.6.a and 5.6.d.). In such cases, *curvefitting* returns the correct relation as the best fit.

5.3.3. Model with Two Stocks and Linear Relations

The third analysis is applied to the CLD in Figure 5.7 where both X and Y are candidates as possible stock variables (see Table 5.7). In this model, both variables are actually stocks and the causal relations are linear. Here, we test whether multiple stock variables add more information so that *curvefitting* can find correct SFD.

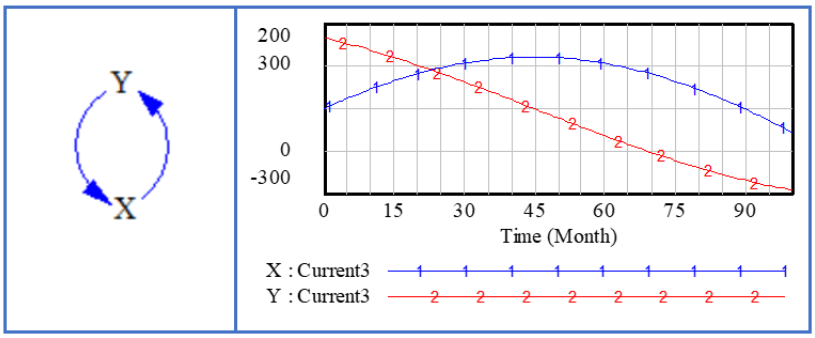

Figure 5.7. CLD and Behavior of the Third Model.

Table 5.7. Curve Fitting for the Third Model.

Curve Fitting from X to Y	**Curve Fitting from Y to X**
(1) Y = f(X)	(3) X = f(Y)
(2) dY/dt = f(X)	(4) dX/dt = f(Y)

Curvefitting is applied for all the relations in Table 5.7 to discover stock variables. In other words, monotonic relations are fitted to each relation in Figure 5.8. For the causal link of x → y, *curvefitting* has the best fit to x → dy with MAPE = 2.28 and RMSE = 0.78. The reason is that Figure 5.8.a does not have a monotonic shape whereas Figure 5.8.b has a monotonic shape. For the causal link of y → x, *curvefitting* has the best fit to y → dx with MAPE = 0.04 and RMSE = 0.058. The reason is that Figure 5.8.d is suitable for a monotonic function, but Figure 5.8.c is not.

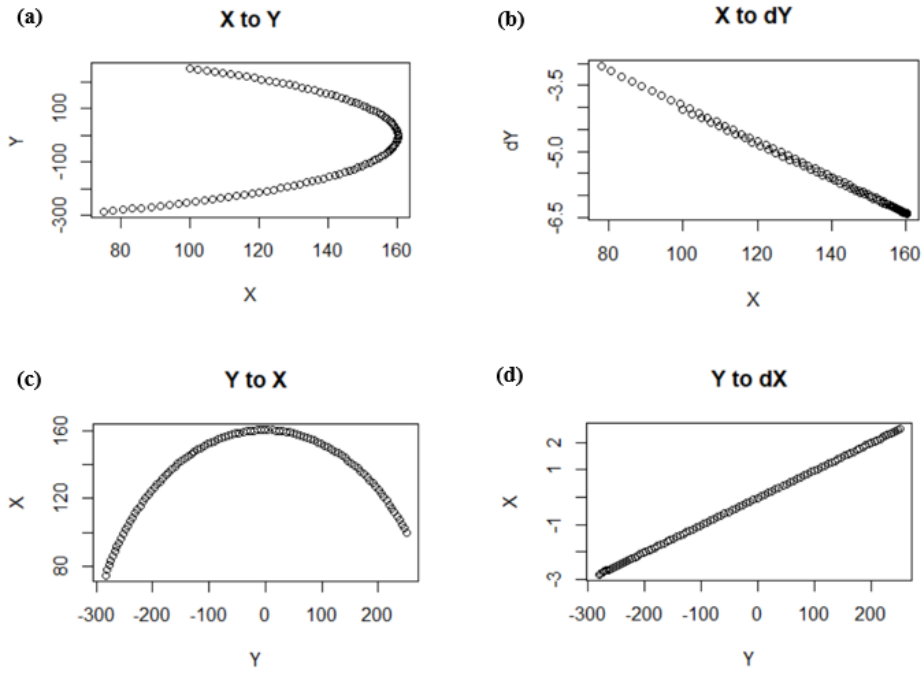

Figure 5.8. Relation Graphs in the Third Model.

Curvefitting results suggest that both variables affect each other through their rate of changes. This means that both variables are stocks. The algorithm reaches the correct conclusion since both variables are stocks in the original model (Figure 5.9).

We reach the conclusion that when there are multiple stocks in the system, the incorrect causal relations may become non-monotonic (see Figure 5.8.a and 5.8.c.). Therefore, *curvefitting* returns the correct relation as the best fit.

In this section, *curvefitting* algorithm is tested on three different models. For the simplest model with only one stock and linear relations, *curvefitting* fits perfect monotonic relations for all the relation possibilities. Hence, it cannot eliminate incorrect SFDs. For the second model with only one stock and non-linear relations, *curvefitting* fits monotonic relations to only one correct possibility for each relation. Therefore, it eliminates incorrect SFDs. For the third model with two stocks and linear relations, *curvefitting* fits monotonic relations to only one correct possibility for each relation. Therefore, it eliminates incorrect SFDs. As a conclusion, it can be said that when

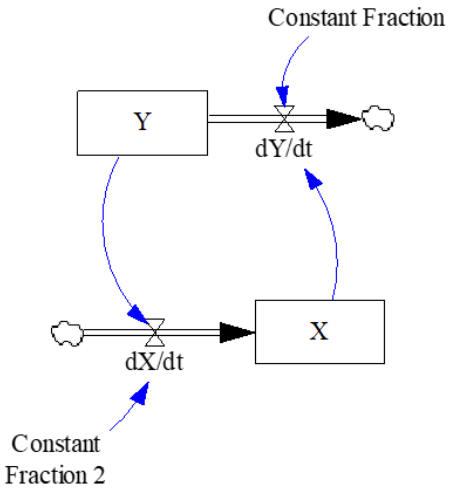

Figure 5.9. Correct SFD of the Third Model.

there are complexity elements in the original model such as non-linearity or multiple stocks, the algorithm has a better chance to discover correct stock variables. Because in such cases, the alternative wrong relations tend to be non-monotonic.

6. CONCLUSION

In system dynamics methodology, model building phase is the most significant task which takes considerable time of the modeler and can potentially create the problem of being subjective. Thanks to the progress in data science, model construction time and model subjectivity can both be reduced by formal analysis of historical data of model variables. The possibility of automated data analysis in the process of determining (1) the polarity of the causal effects and (2) discovering stock variables are two of the promising topics that are examined in this research.

For determining the polarity of causal effects, it is assumed that we have real-life (non-experimental) data about multiple system variables influencing a given effect variable. In case of multiple cause variables, the modeler needs to rely on *a priori* knowledge or expert opinions because extracting ceteris-paribus relation polarities from non-experimental (*non-ceteris paribus*) data is a nontrivial, sometimes impossible problem. To solve the problem, we propose an algorithm that we call *discoverpolarity*, code it in R language and test it with 7 different synthetic data-sets including highly collinear cause variables. Then, we compare the test results with Spearman correlation analysis.

The results show that under the assumptions of *discoverpolarity*, it outperforms Spearman correlation analysis when there is collinearity in the data-set. *Discoverpolarity* can obtain meaningful and useful results regardless of the underlying structure formulations. However, the algorithm is sensitive to the selection of the input variables and the data-set sampling. The algorithm gives an opportunity to modelers to strengthen their *a priori* knowledge on link polarities when there is a lack of *ceteris paribus*information or data. Moreover, modeler does not have to search for a *ceteris paribus* experimental data or information from the literature.

It is important to note that the algorithm may return multiple possible polarities instead of a unique solution when the data only consists of all perfectly correlated data

points. In such cases, the modeler must choose the most plausible of the returned polarities. In addition, modeler may need to check the link polarities that are eliminated in the second phase of algorithm if the number of the observed differences to support these eliminations are only a few. Another limitation of *discoverpolarity* is that the modeler must decide on the threshold values used in the algorithm. Even though *discoverpolarity* has default values for the thresholds, for some data-sets, the default may eliminate all the possible polarities including the true polarity. This may happen if the thresholds used in *discoverpolarity* are high relative to the nature of the data. In such cases, the modeler must decrease the threshold values to obtain a reasonable result. On the other hand if *discoverpolarity* returns multiple possible polarities, the modeler may prefer to increase the threshold values. However, usage of too high thresholds can cause the elimination of true polarity because of high non-linearity in the effect functions.

As further research, we plan to focus on decreasing the sensitivity of *discoverpolarity* to its threshold values and other input parameters to make it more robust. After enough tests with synthetic data, we also plan to test the algorithm with real data before it can be used in real-life modeling. Finally, not only using the signs of differences but also the magnitudes of these differences, one can try to estimate the mathematical forms (additive, multiplicative, or hybrid) of the causal formulations.

For the second goal, discovering stock variables, simulation-generated 'synthetic' data is analyzed to be able to evaluate the validity of the results obtained from the *curvefitting* algorithm. *Curvefitting* algorithm is tested on three different models. We conclude that when there are complexity elements in the original model such as non-linearity or multiple stocks, the algorithm is more likely to discover correct stock variables. Because in such cases, the alternative wrong relations are likely to be non-monotonic. Therefore, there is a promising possibility to discover stock variables by using CLD and historical data of variables. When we correctly identify stock variables, we automatically find simplified stock-flow diagram from the given causal loop diagram. Discovering stock variables has a potential to reduce the modeling time and it may eliminate the risk of wrong SFD usage.

In further research, we plan to categorize cases in which the algorithm can discover stocks from CLD. In addition, we plan to focus on extending the *curvefitting* algorithm so that it can analyze not only the cases with one cause variable but also cases with multiple cause variables.

REFERENCES

1. Barlas, Y., "System Dynamics: Systemic Feedback Modeling for Policy Analysis 1", *Knowledge for Sustainable Development: An Insight Into the Encyclopedia of Life Support Systems*, 2002.

2. Pruyt, E., "Integrating Systems Modelling and Data Science: The Joint Future of Simulation and 'Big Data' Science", *International Journal of System Dynamics Applications (IJSDA)*, 2016.

3. Houghton, J. and M. Siegel, "Advanced Data Analytics for System Dynamics Models using PySD", *Revolution*, Vol. 3, No. 4, 2015.

4. Martinez-Moyano, I. J. and G. P. Richardson, "Best Practices in System Dynamics Modeling", *System Dynamics Review*, Vol. 29, No. 2, pp. 102–123, 2013.

5. Richardson, G. P., "Problems for the Future of System Dynamics", *System Dynamics Review*, 1996.

6. Homer, J. and R. Oliva, "Maps and Models in System Dynamics: A Response to Coyle", *System Dynamics Review*, 2001.

7. Barlas, Y., "Formal Aspects of Model Validity and Validation in System Dynamics", *System Dynamics Review*, 1996.

8. Nordhaus, W. D., "World Dynamics: Measurement Without Data", *The Economic Journal*, Vol. 83, No. 332, pp. 1156–1183, 1973.

9. Lane, D. C., "Should System Dynamics be Described as a 'Hard' or 'Deterministic' Systems Approach?", *Systems Research and Behavioral Science*, 2000.

10. Abdelbari, H. and K. Shafi, "A Computational Intelligence-based Method to

'Learn' Causal Loop Diagram-like Structures from Observed Data", *System Dynamics Review*, Vol. 33, No. 1, pp. 3–33, 2017.

11. Medina-Borja, A. and K. S. Pasupathy, "Uncovering Complex Relationships in System Dynamics Modeling: Exploring the Use of CART , CHAID and SEM", *System Dynamics Society Conference*, pp. 1–24, 2007.

12. Bühlmann, P., J. Peters, J. Ernest and E. Zürich, "CAM: Causal Additive Models, High-Dimensional Order Search And Penalized Regression", *The Annals of Statistics*, Vol. 42, No. 6, pp. 2526–2556, 2014.

13. Tarka, P., "An overview of Structural Equation Modeling: Its Beginnings, Historical Development, Usefulness and Controversies in the Social Sciences", *Quality and Quantity*, Vol. 52, No. 1, pp. 313–354, 2018.

14. Hauke, J. and T. Kossowski, "Comparison of Values of Pearson's and Spearman's Correlation Coefficients on the Same Sets of Data", *Quaestiones Geographicae*, Vol. 30, No. 2, pp. 87–93, 2011.

15. Kroesen, M., E. Molin and B. Wee, "Determining the Direction of Causality Between Psychological Factors and Aircraft Noise Annoyance", *Noise and Health*, Vol. 12, No. 46, p. 17, 2010.

16. Farrar, D. E. and R. R. Glauber, "Multicollinearity in Regression Analysis: The Problem Revisited", *The Review of Economics and Statistics*, Vol. 49, No. 1, p. 92, feb 1967.

17. Dormann, C. F., J. Elith, S. Bacher, C. Buchmann, G. Carl, G. Carré, J. R. G. Marquéz, B. Gruber, B. Lafourcade, P. J. Leitão *et al.*, "Collinearity: a review of methods to deal with it and a simulation study evaluating their performance", *Ecography*, Vol. 36, No. 1, pp. 27–46, 2013.

18. Oliva, R., "Model Calibration as a Testing Strategy for System Dynamics models",

European Journal of Operational Research, Vol. 151, No. 3, pp. 552–568, 2003.

19. Malhotra, N. K., M. Peterson and S. B. Kleiser, "Marketing research: A State-of-the-art Review and Directions for the Twenty-First Century", *Journal of the Academy of Marketing Science*, 1999.

20. Hovmand, P. S. and N. Chalise, "Simultaneous Linear Estimation Using Structural Equation Modeling", H. Rahmandad, R. Oliva and N. D. Osgood (Editors), *Analytical Methods for Dynamic Modelers*, chap. 1, pp. 71–93, MIT Press, 1 edn., 2015.

21. Kiebel, S. and A. Holmes, "The General Linear Model", *Human Brain Function: Second Edition*, 2003.

22. Hastie, T. J. and R. J. Tibshirani, "Generalized Additive Models", *Monographs on Statistics and Applied Probability*, Vol. 43, p. 15, 1990.

23. Takahashi, Y., "Stock Flow Diagram Making with Incomplete Information about Time Properties of Variables", *Proceedings of the 2006 System Dynamics Conference*, 2006.

24. Drobek, M., W. Gilani and D. Soban, "A Data Driven and Tool Supported CLD Creation Approach", , 2014.

25. Struben, J., J. Sterman and D. Keith, "Parameter Estimation Through Maximum Likelihood and Bootstrapping Methods — Request PDF", H. Rahmandad, R. Oliva and N. D. Osgood (Editors), *Analytical Methods for Dynamic Modelers*, chap. 1, pp. 3–38, MIT Press, 1 edn., 2015.

26. Jalali, M. S., H. Rahmandad and H. Ghoddusi, "Using the Method of Simulated Moments for System Identification", H. Rahmandad, R. Oliva and N. D. Osgood (Editors), *Analytical Methods for Dynamic Modelers*, chap. 2, pp. 39–69, MIT Press, 1 edn., 2015.

27. Hosseinichimeh, N., H. Rahmandad, M. S. Jalali and A. K. Wittenborn, "Estimating the Parameters of System Dynamics Models Using Indirect Inference", *System Dynamics Review*, Vol. 32, No. 2, pp. 156–180, 2016.

28. Sterman, J. D., *Business Dynamics, System Thinking and Modeling for a Complex World*, HD30. 2 S7835, 2000.

29. Yue, S., P. Pilon and G. Cavadias, "Power of the Mann–Kendall and Spearman's Rho Tests for Detecting Monotonic Trends in Hydrological Series", *Journal of Hydrology*, Vol. 259, No. 1-4, pp. 254–271, 2002.

30. Saigal, S. and D. Mehrotra, "Performance Comparison of Time Series Data Using Predictive Data Mining Techniques", *Advances in Information Mining*, Vol. 4, pp. 57–66, 01 2012.

APPENDIX A: DISCOVERPOLARITY CODE

```
library(data.table)
library(Hmisc)
options(scipen = 999)

dealvaroutoflimits<-function(causevar, maxlimit, minlimit){

  big_indexes<- which(causevar> maxlimit )
  small_indexes<- which(causevar< minlimit)
  if(length(big_indexes)>0){
    causevar[big_indexes]<-unname(maxlimit)
  }
  if(length(small_indexes)>0){
    causevar[small_indexes]<-unname(minlimit)
  }
  causevar
}
diffofeveryrow<-function(causeseffect){
  h<-sapply(1:(nrow(causeseffect)-1), function(j)
    sapply(1:ncol(causeseffect),function(i) as.numeric(diff(causeseffect[,i],j))))

  t<-sapply(1:(nrow(causeseffect)-1), function(lag)
                  cbind(c((1+lag):nrow(causeseffect)),
                  c(1:(nrow(causeseffect)-lag))))
  t<-do.call(rbind,t)


  diff_causeseffect1 <-do.call(rbind,h)
  diff_causeseffect<-data.frame(cbind(t,diff_causeseffect1))
  diff_causeseffect
}
dividetolimits<-function(causeseffect, limits){
  maxdiff<-limits[,2]-limits[,1]    #maximum change in cause variables
  causeseffect<-sapply(1:ncol(causeseffect), function(y)
   (causeseffect[,y]-limits[y,1])/maxdiff[y])
  data.frame(causeseffect)
}
varImpmultiplication<-function(causeseffect,varImp){
  causeseffect[,1:(ncol(causeseffect)-1)]<-sapply(1:(ncol(causeseffect)-1),
    function(x) causeseffect[,x]*varImp[x])
  causeseffect
}
seperatez0dataset<-function(diff_causeseffect, threshold1,
 threshold3){

  output <- unlist(data.table(abs(diff_causeseffect[,
```

```
                        3:(ncol(diff_causeseffect)-1)]))<=threshold1)
  output1<-output*1
  indexes0z<- which(abs(diff_causeseffect[,
      ncol(diff_causeseffect)])<=(threshold3) &
       rowSums(output1)<((ncol(diff_causeseffect)-3)-1))
  z0<-FALSE
  if(length(indexes0z)>0){
    diff_causeseffect[indexes0z,ncol(diff_causeseffect)]<-0

    sign_ind3<-sapply(3:ncol(diff_causeseffect)
    , function(x) as.numeric(sign(diff_causeseffect[indexes0z,x])))
    if(length(indexes0z)==1){
      datdz0<-data.frame(cbind(diff_causeseffect[indexes0z,1]
                               ,diff_causeseffect[indexes0z,2],t(sign_ind3)))

    }else{
      datdz0<-data.frame(cbind(diff_causeseffect[indexes0z,1]
                               ,diff_causeseffect[indexes0z,2],sign_ind3))

    }
    diff_causeseffect<-diff_causeseffect[-indexes0z,]
    z0=TRUE
  }
  if(z0==TRUE){return(list(diff_causeseffect=diff_causeseffect,
  z0=z0, datdz0=datdz0))
    }else{return(list(diff_causeseffect=diff_causeseffect,z0=z0))}

}
findcandidateindexes<-function(diff_causeseffect, threshold1){

  candlistindexes<-as.list(NULL)
  for(i in 3:(ncol(diff_causeseffect)-1)){
    candlistindexes[[i-2]]<- which(abs(diff_causeseffect[,i])<=threshold1)
     }
  #reduce points where we do not move significantly in any direction
  deletedcommons<-Reduce(intersect, candlistindexes)

  if(length(deletedcommons)>0){
    for(i in 1:length(candlistindexes)){
      row_sub = sapply( 1:length(candlistindexes[[i]]),
       function(row) all(find.matches(candlistindexes[[i]][row],
       deletedcommons)$match==0))
      candlistindexes[[i]]<- candlistindexes[[i]][row_sub]

    }

    }
  return(list(diff_causeseffect=diff_causeseffect,
```

```
  candlistindexes=candlistindexes))
}
setcausevarzero<-function(diff_causeseffect,
candlistindexes,threshold1, threshold2){
  diff_causeseffect3<-diff_causeseffect
  bigratios<-NULL
  smallratios<-NULL
  for(i in 1:(ncol(diff_causeseffect)-3)){
    if(length(candlistindexes[[i]])!=0){

      for(j in candlistindexes[[i]]){
        ratioss<- abs(diff_causeseffect3[j,3:(ncol(diff_causeseffect)-1)])/
                  sum(abs(diff_causeseffect3[j,3:(ncol(diff_causeseffect)-1)]))

          sumofcand<- sum(ratioss[which((abs(diff_causeseffect3[j,
                          3:(ncol(diff_causeseffect)-1)]))<=threshold1)])
          howmanycand<- sum((abs(diff_causeseffect3[j,
                            3:(ncol(diff_causeseffect)-1)])<=threshold1)*1)
          if(unname(sumofcand<=threshold2 | ratioss[i]<=threshold2/howmanycand)){

            diff_causeseffect[j,i+2]<-0
            smallratios<-c(smallratios,unname(ratioss[i]))
          }else if (ratioss[i]>threshold2){
            bigratios<-c(bigratios,unname(ratioss[i]))
          }


      }
    }
  }
  return(list(diff_causeseffect=diff_causeseffect, smallratios=smallratios,
              bigratios=bigratios))
}
obtainallcorners<-function(sign_diff_causeseffect){
  dat1<-sign_diff_causeseffect
  candlistindexes2<-NULL

  for (i in 3:(ncol(dat1))){
    candlistindexes2<-which(dat1[,i]==0)
    if(length(candlistindexes2)>0){
      for(j in candlistindexes2){
        newline1<-dat1[j,]
        newline1[i]<-1
        newline2<-dat1[j,]
        newline2[i]<--1
        newlines<-rbind(newline1, newline2)
        dat1<-rbind(dat1,newlines)
      }
```

```
    }
    rownames(dat1)<-1:nrow(dat1)}
  #multiply rows with -1 where effect is 1
  dat1[which(dat1[,ncol(dat1)]==1),3:ncol(dat1)]<-dat1[which(dat1[,
  ncol(dat1)]==1) ,3:ncol(dat1)]*(-1)
  for(x in 3:ncol(dat1)){
    dat1[,x] <- factor(dat1[,x], levels = c("-1","0","1"))
  }

  factors<- list()
  factors[[1]]<- dat1[,3]
  for(i in 3:(ncol(dat1))){

    factors[[i-2]]<- dat1[,i]
  }

  a<-split(dat1,f=factors, drop=TRUE)
  values<-sapply(1:length(a), function(x) strsplit(names(a)[x],".", fixed=TRUE))
  signvalues<-NULL
  perc<- sapply(1:length(a), function(p) nrow(a[[p]]))

  for(i in 1:length(a)){
    signvalues<-rbind(signvalues,as.numeric(values[[i]]))
  }
  return(list(signvalues=signvalues, perc=perc))
}
createcorners<-function(n){
  alt<-c(-1,1)
  alt<-unname(data.frame(replicate(n,alt)))
  all<-expand.grid(alt)
  all1<-data.frame(all)
  return(all1)
}
eliminatecorners<-function(signvalues,all1){
  for(i in 1:nrow(signvalues))
  {
    if ( signvalues[i,ncol(signvalues)]==1){

      s<- (signvalues[i,-ncol(signvalues)]*(-1))
      s<-data.table(t(s))
      if(nrow(all1)!=0){
        m<-find.matches(all1,s)
        if(sum(m$matches)!=0){
          all1<-all1[-which(m$matches==1),]
        }
      }

    }else{
```

```
      s<- (signvalues[i,-ncol(signvalues)])
      s<-data.table(t(s))
      if(nrow(all1)!=0){
        m<-find.matches(all1,s)
        if(sum(m$matches)!=0){
          all1<-all1[-which(m$matches==1),]
        }
      }
    }
  }
  return(all1=all1)
}



discoverpolarity<-function(causes,effect,threshold1, threshold2=NA,
                           limits=NULL,
                           varImp=rep(1,ncol(causes)), threshold3=NA){

  if(is.na(threshold2)){threshold2<-threshold1}
  if(is.na(threshold3)){ threshold3<-threshold1}
  if(is.null(limits)){ limits<-t(sapply(1:ncol(causes),
                        function(x) c(Min=min(causes[,x]), Max=max(causes[,x])))))}
  #add maximum diff in observed effect values
  limits<-rbind(limits, c(min(effect),max(effect)))
  effect<-data.frame(effect)
  causes<-data.frame(causes)
  causesvarnames<- colnames(causes)  #names of cause variables


  #set the variables which are out of the limits as limits
  causes<- data.frame(sapply(1:ncol(causes),
                             function(k) dealvaroutoflimits(causes[,k], limits[k,2],
                                                            limits[k,1])))
  causeseffect<-data.frame(cbind(causes,effect))
  if(length(which(causeseffect[,ncol(causeseffect)]==0))!=0){
    causeseffect<-causeseffect[-which(causeseffect[,ncol(causeseffect)]==0),]
  }
  #delete rows where z=0
  if(length(which(causeseffect[,ncol(causeseffect)]==0))>0){
    causeseffect<-causeseffect[-which(causeseffect[,ncol(causeseffect)]==0),]
  }
  #divide each variable to its limits
  causeseffect<-dividetolimits(causeseffect, limits)
  #multiply variable importances
  causeseffect<-varImpmultiplication(causeseffect, varImp)
  #take differences of each point
  diff_causeseffect <-diffofeveryrow(causeseffect)
```

```
#create all the possible corners
all1<-createcorners((ncol(causeseffect)-1))


#APPLY FIRST PHASE
#take signs of diffirences
sign_diff_causeseffect<-sapply(3:ncol(diff_causeseffect)
, function(x) as.numeric(sign(diff_causeseffect[,x])))
sign_diff_causeseffect<-data.frame(cbind(diff_causeseffect[,1]
  ,diff_causeseffect[,2],sign_diff_causeseffect))
 #obtain overall observed corners
a<-obtainallcorners(sign_diff_causeseffect)
signvalues1<-a$signvalues
perc1<-a$perc


#eliminate corners
all1<-eliminatecorners(signvalues1,all1)

if(nrow(all1)<=1){
  #create result tables
  row_sub = apply(signvalues1, 1, function(row) all(row !=0 ))

  if(length(row_sub)==1){
    eliminated1proc<-data.frame(cbind(t(signvalues1[row_sub,
    1:(ncol(signvalues1)-1)]),perc1[row_sub]))
  }else{
    eliminated1proc<-data.frame(cbind(signvalues1[row_sub,
    1:(ncol(signvalues1)-1)],perc1[row_sub]))
  }
  colnames(eliminated1proc)<-c(causesvarnames, "NoofObservedDifferences")
  colnames(all1)<-causesvarnames

  if(nrow(all1)>0){rownames(all1)<-1:nrow(all1)}

  l1<-list(All1=all1,  EliminatedOptions1=eliminated1proc)
  return(l1)

}else{#APPLY SECOND PHASE
  #find z=0 values and seperate it as a new set
  k<-seperatez0dataset(diff_causeseffect,
                       threshold1, threshold3)
  diff_causeseffect<-k$diff_causeseffect
  z0<-k$z0
  datdz0<-k$datdz0

  #find candidate indexes
  a<-findcandidateindexes(diff_causeseffect, threshold1)
```

```
diff_causeseffect<-a$diff_causeseffect
candlistindexes<-a$candlistindexes
#set cause variables as 0
s<-setcausevarzero(diff_causeseffect, candlistindexes,
threshold1, threshold2)
diff_causeseffect<-s$diff_causeseffect
smallratios<-s$smallratios
bigratios<-s$bigratios

#take signs of diffirences
sign_diff_causeseffect<-sapply(3:ncol(diff_causeseffect)
, function(x) as.numeric(sign(diff_causeseffect[,x])))
sign_diff_causeseffect<-data.frame(cbind(diff_causeseffect[,1]
    ,diff_causeseffect[,2],sign_diff_causeseffect))

#add rows where z=0
if(z0){
  sign_diff_causeseffect<-rbind(sign_diff_causeseffect, datdz0)
}
sign_diff_causeseffect<-data.frame(sign_diff_causeseffect)

#obtain overall observed corners
a2<-obtainallcorners(sign_diff_causeseffect)
signvalues2<-a2$signvalues
perc2<-a2$perc

#create all the possible corners
all<-createcorners((ncol(causeseffect)-1))


#eliminate corners
all2<-eliminatecorners(signvalues2,all)

#create result tables
row_sub = apply(signvalues1, 1, function(row) all(row !=0 ))

if(length(row_sub)==1){
  eliminated1proc<-data.frame(cbind(t(signvalues1[row_sub,
  1:(ncol(signvalues1)-1)]),perc1[row_sub]))
}else{
  eliminated1proc<-data.frame(cbind(signvalues1[row_sub,
  1:(ncol(signvalues1)-1)],perc1[row_sub]))
}
colnames(eliminated1proc)<-c(causesvarnames, "NoofObservedDifferences")

row_sub2 = apply(signvalues2, 1, function(row) all(row !=0 ))

if(length(row_sub2)==1){
```

```
    eliminated2proc<-data.frame(cbind(t(signvalues2[row_sub2,
    1:(ncol(signvalues2)-1)]),perc2[row_sub2]))
  }else{
    eliminated2proc<-data.frame(cbind(signvalues2[row_sub2,
    1:(ncol(signvalues2)-1)],perc2[row_sub2]))
  }
  colnames(eliminated2proc)<-c(causesvarnames, "NoofObservedDifferences")
  colnames(all1)<-causesvarnames
  colnames(all2)<-causesvarnames

  if(nrow(all1)>0){rownames(all1)<-1:nrow(all1)}
  if(nrow(all2)>0){rownames(all2)<-1:nrow(all2)}

  l1<-list(All1=all1,   EliminatedOptions1=eliminated1proc,
           All2=all2,   EliminatedOptions2=eliminated2proc)
  return(l1)}

}
```

APPENDIX B: DISCOVERPOLARITY DATA-SETS

```
library(ggplot2)
library(akima)
library(data.table)
library(ppcor)

#Dataset1
{
  set.seed(1)
  x1<-c(seq(25,70, length.out = 20),29)
  x2<-c(seq(60,20, length.out = 20), 55)
  y<- 50+ 150-x1+200-80/(1+exp(-0.12*(x2-x2[1])))
  cor(x1,x2)
  causes<-data.frame(cbind(x1,x2))
  effect<-y
  discoverpolarity(causes,effect,threshold1=0.15)
}


#Dataset2
{
  set.seed(1)
  x1<- c(35,seq(45,50,length.out =5),seq(50.5,65,length.out =20) )
  x2<- c(25, seq(24,20,length.out =5),seq(12,5,length.out =20))
  cor(x1,x2)
  ind<-data.frame(cbind(x1,x2))
  y<- x1*(100-100/(1+exp(-0.15*(x2-x2[1]))))
  discoverpolarity(ind,y,threshold1=0.15)
}


#Dataset3
{
  x1<- c(seq(70,90,length.out =10), seq(90,1,length.out =90))
  x2<- c(seq(31,44,length.out =50),seq(44,35,length.out =50) )
  k<-c(seq(120,30,length.out =40),seq(30,60,length.out =30),
  seq(59,30,by=-1) )
  x3<-90/(1+exp(-0.02*(k-k[1])))
  x4<- c(seq(1,30,length.out =40), seq(29,15,length.out =10),
  seq(16,70,length.out =50))
  x4<-60/(1+exp(-0.15*(x4-x4[1])))
   ind<-data.frame(cbind(x1,x2,x3,x4))
  cor(ind)
  y<- x1+100-100/(1+exp(-0.2*(x2-x2[1])))+x3+
  60/(1+exp(-0.15*(x4-x4[1])))
  limits<-t(sapply(1:ncol(ind), function(x) c(Min=min(ind[,x]),
```

```
  Max=max(ind[,x])))))
  limits[4,2]<-55
  discoverpolarity(causes=ind,effect =y,threshold1=0.05,
                   limits=limits)
}


#Dataset4
{
  x1<- c(seq(60,90,length.out =30), seq(90,1,length.out =70))
  x2<- c(seq(31,44,length.out =50),seq(44,35,length.out =50) )
  k<-c(seq(120,30,length.out =40),seq(30,60,length.out =30),
  seq(59,30,by=-1) )
  x3<-90/(1+exp(-0.02*(k-k[1])))
  x4<- c(seq(1,30,length.out =40), seq(29,15,length.out =10),
  seq(16,70,length.out =50))
  x4<-60/(1+exp(-0.15*(x4-x4[1])))
  ind<-data.frame(cbind(x1,x2,x3,x4))
  cor(ind)
  y<- 100-x1+(160-100/(1+exp(-0.2*(x2-x2[1]))))+
  100-x3+120/(1+exp(-0.1*(x4-x4[1])))
  limits<-t(sapply(1:ncol(ind), function(x) c(Min=min(ind[,x]),
   Max=max(ind[,x]))))
  limits[4,2]<-55
  limits[2,2]<-42
  discoverpolarity(causes=ind,effect =y,threshold1=0.15)
 }


#Dataset5
{
  x1<- c(seq(90,50,length.out =30), seq(51,99,length.out =70))
  x2<- c(seq(21,44,length.out =10),seq(44,15,length.out =90) )
  k<-c(seq(120,70,length.out =40),seq(70,100,length.out =30),
  seq(100,30,length.out =30) )
  x3<-90/(1+exp(-0.02*(k-k[1])))
  x4<- c(seq(10,30,length.out =40), seq(29,15,length.out =10),
  seq(16,40,length.out =50))
  x4<-60/(1+exp(-0.15*(x4-x4[1])))
  ind<-data.frame(cbind(x1,x2,x3,x4))
  cor(ind)
  y<-100*(((120/(1+exp(-0.1*(x4-x4[1])))-30)/60)*
            ((100-x1)/25+(160-100/(1+exp(-0.2*(x2-x2[1]))))-55)/50.55+
               ((100-x3)-50)/11 ))
  limits<-t(sapply(1:ncol(ind), function(x) c(Min=min(ind[,x]),
  Max=max(ind[,x]))))
  limits[4,2]<-55
  limits[2,2]<-35
```

```
  discoverpolarity(causes=ind,effect =y,limits=limits,threshold1=0.15)
}


#Dataset6
{
  x1<- c(seq(5,14,length.out =80), seq(13,7,length.out =10),
  seq(7,9,length.out =10))
  x2<- c(seq(11,4,length.out =50),seq(4,7,length.out =20),
  seq(7,1,length.out =30) )
  k<-c(seq(60,10,length.out =100) )
  x3<-50/(1+exp(-0.07*(k-k[1])))
  ind<-data.frame(cbind(x1,x2,x3))
  cor(ind)
  y<- 100*(((15/(1+exp(-0.4*(x1-x1[1]))))-2)/7.5)*(x2/20+0.75)*
  ((14-(20/(1+exp(-0.1*(x3-x3[1])))))/13.77+0.3)
  limits<-t(sapply(1:ncol(ind), function(x) c(Min=min(ind[,x]),
   Max=max(ind[,x]))))
  limits[1,2]<-12.5
  discoverpolarity(causes=ind,effect =y,threshold1=0.05,limits=limits)
}

#Dataset7
{
  x1<- c(seq(5,10,length.out =40))
  x2<- c(seq(11,4,length.out =40))
  x3<-c(seq(25,10,length.out =40))
  ind<-data.frame(cbind(x1,x2,x3))
  cor(ind)
  y<- 100*(((15/(1+exp(-0.6*(x1-x1[1]))))-2)/7.5+
              (x2/17+0.65)+
              ((14-(20/(1+exp(-0.2*(x3-x3[1])))))/13.77+0.3) )
  limits<-t(sapply(1:ncol(ind), function(x) c(Min=min(ind[,x]),
  Max=max(ind[,x]))))
  discoverpolarity(causes=ind,effect =y,threshold1=0.15,limits=limits)
}
```

APPENDIX C: CURVEFITTING CODE

```
library(rPref)

curvefit<- function(x,y){
  results<-data.frame()
  predictions<-NULL

  #linear models
  m1<-NULL
  m1<- tryCatch(gnls(y~ a+ b*x ,
   start= c(a=0, b=1)), error=function(e) NULL)
  if(!is.null(m1)){results<- rbind.fill(results,
   data.frame(cbind(rbind(m1$coefficients),
   Mape=100*sum(abs((y-predict(m1))/y))/length(y),
   Sigma=sqrt(sum((y-predict(m1))^2)/length(y)),
    Method= "LinFunc", Function="y~ a+b*x", Model= "m1",
     Isnull=is.null(m1) ) ))
  predictions<- cbind(predictions, M1=predict(m1))}

  m2<-NULL
  m2<- tryCatch(lm(y~ x), error=function(e) NULL)
  summary(m2)
  if(!is.null(m2)){results<- rbind.fill(results,
  data.frame(cbind(rbind(m2$coefficients),
  Mape=100*sum(abs((y-predict(m2))/y))/length(y),
  Sigma=sqrt(sum((y-predict(m2))^2)/length(y)),
  Method= "LinFunc", Function="y~ a+b*x", Model= "m2",
  Isnull=is.null(m2) ) ))
  predictions<- cbind(predictions, M2=predict(m2))}


  #exponential and s-shaped models
  m3<- NULL
  logy<- log(y)
  m3<- tryCatch(gnls(logy~ a+b*x, start=c(a=0, b=1)),
  error=function(e) NULL)
  if(!is.null(m3)){results<- rbind.fill(results,
  data.frame(cbind(rbind(m3$coefficients),
  Mape=100*sum(abs((y-exp(predict(m3)))/y))/length(y),
  Sigma=sqrt(sum((y-exp(predict(m3)))^2)/length(y)),
  Method= "ExpFunc", Function="logy~a+b*x",
   Model= "m3", Isnull=is.null(m3)  ) ))
  predictions<- cbind(predictions, M3=exp(predict(m3)))}

  m7<- NULL
  m7<- tryCatch(nls(y~ SSlogis(x, Asym, xmid, scal)),
```

```
error=function(e) NULL)
if(!is.null(m7)){results<- rbind.fill(results,
data.frame(cbind(rbind(coef(m7)),
Mape=100*sum(abs((y-predict(m7))/y))/length(y),
Sigma=sqrt(deviance(m7)), Method= "SShaped",
Function="y~ SSlogis(x, Asym, xmid, scal)", Model= "m7",
Isnull=is.null(m7) ) ))
predictions<- cbind(predictions, M7=predict(m7))}

m8<- NULL
m8<- tryCatch(nls(y~ SSasymp(x, Asym, R0, lrc)),
error=function(e) NULL)
if(!is.null(m8)){results<- rbind.fill(results,
data.frame(cbind(rbind(coef(m8)),
Mape=100*sum(abs((y-predict(m8))/y))/length(y),
Sigma=sqrt(deviance(m8)), Method= "SShaped",
Function="y~ SSasymp(x, Asym, R0, lrc)", Model= "m8",
Isnull=is.null(m8) ) ))
predictions<- cbind(predictions, M8=predict(m8))}

m11<- NULL
logy<- log(-y)
m11<- tryCatch(gnls(logy~ a+b*x, start=c(a=0, b=1)),
error=function(e) NULL)
if(!is.null(m11)){results<- rbind.fill(results,
data.frame(cbind(rbind(m11$coefficients),
Mape=100*sum(abs((y+exp(predict(m11)))/y))/length(y),
Sigma=sqrt(sum((y+exp(predict(m11)))^2)/length(y)),
Method= "ExpFunc", Function="log-y~ a+b*x", Model= "m11",
Isnull=is.null(m11) ) ))
predictions<- cbind(predictions, M11=-exp(predict(m11)))}


if(!is.null(m7))
{
  m13<- NULL
  m13<- tryCatch( gnls(y ~ Asym/(1+exp((xmid-x)/scal)),
   start = coef(m7)), error=function(e) NULL)
  if(!is.null(m13)){results<- rbind.fill(results,
  data.frame(cbind(rbind(m13$coefficients),
  Mape=100*sum(abs((y-predict(m13))/y))/length(y),
  Sigma=sqrt(sum((y-predict(m13))^2)/length(y)),
  Method= "SShaped", Function="y ~ Asym/(1+exp((xmid-x)/scal)",
  Model= "m13", Isnull=is.null(m13) ) ))
  predictions<- cbind(predictions, M13=predict(m13))}
}

m16<- NULL
```

```
  m16<- tryCatch( gnls(y ~max(y)/(1+exp((xmid-x)/scal)),
  start = c(xmid=1, scal=1)), error=function(e) NULL)
  if(!is.null(m16)){results<- rbind.fill(results,
  data.frame(cbind(rbind(m16$coefficients),
  Mape=100*sum(abs((y-predict(m16))/y))/length(y),
  Sigma=sqrt(sum((y-predict(m16))^2)/length(y)),
  Method= "SShaped", Function="y ~ max(y)/(1+exp((xmid-x)/scal)",
  Model= "m16", Isnull=is.null(m16) ) ))
  predictions<- cbind(predictions, M16=predict(m16))}

  m17<- NULL
  m17<- tryCatch( gnls(y ~max(y)-(max(y)-min(y))/(1+exp((xmid-x)/scal)),
  start = c(xmid=1, scal=1)), error=function(e) NULL)
  if(!is.null(m17)){results<- rbind.fill(results,
  data.frame(cbind(rbind(m17$coefficients),
  Mape=100*sum(abs((y-predict(m17))/y))/length(y),
  Sigma=sqrt(sum((y-predict(m17))^2)/length(y)),
  Method= "SShaped",
  Function="y ~ max(y)-(max(y)-min(y))/(1+exp((xmid-x)/scal))",
  Model= "m17", Isnull=is.null(m17) ) ))
  predictions<- cbind(predictions, M17=predict(m17))}

  m18<- NULL
  m18<- tryCatch( gnls(y ~min(y) +(max(y)-min(y))/(1+
  exp((xmid-x)/scal)), start = c(xmid=1, scal=1)), error=function(e) NULL)
  if(!is.null(m18)){results<- rbind.fill(results,
  data.frame(cbind(rbind(m18$coefficients),Mape=100*
  sum(abs((y-predict(m18))/y))/length(y),
  Sigma=sqrt(sum((y-predict(m18))^2)/length(y)), Method= "SShaped",
  Function="y ~ min(y) +(max(y)-min(y))/(1+exp((xmid-x)/scal)",
  Model= "m18", Isnull=is.null(m18) ) ))
  predictions<- cbind(predictions, M18=predict(m18))}

  results<- results[which(results$Isnull == FALSE),]
  results1<-results
  predictions1<-predictions
  output1<-list(results1, predictions1)
  return (output1)
}



curvefitxdy<- function(x,y,dy){
  x<- x[-c( length(x))]
  y1<-y
  y<-dy
  results<-data.frame()
  predictions<-NULL
```

```
#Linear Models
m1<-NULL
m1<- tryCatch(gnls(y~ a+ b*x , start= c(a=0, b=1)), error=function(e) NULL)
if(!is.null(m1)){
  dyEst<- predict(m1)
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(m1$coefficients),
  Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
  Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "LinFunc",
  Function="dy~ a+ b*x", Model= "m1", Isnull=is.null(m1) ) ))
predictions<- cbind(predictions, M1=yEst)}

m2<-NULL
m2<- tryCatch(lm(y~ x), error=function(e) NULL)
if(!is.null(m2)){
  dyEst<- predict(m2)
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(m2$coefficients),
  Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
  Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "LinFunc",
  Function="dy~ a+ b*x", Model= "m2", Isnull=is.null(m2) ) ))
predictions<- cbind(predictions, M2=yEst)}

#Exponential and Sshaped Models
m3<- NULL
logy<- log(y)
m3<- tryCatch(gnls(logy~ a+b*x, start=c(a=0, b=1)), error=function(e) NULL)
if(!is.null(m3)){
  dyEst<- exp(predict(m3))
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(m3$coefficients),
  Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
  Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "ExpFunc",
  Function="logdy~ a+b*x", Model= "m3", Isnull=is.null(m3) ) ))
predictions<- cbind(predictions, M3=yEst)}

 m7<- NULL
m7<- tryCatch(nls(y~ SSlogis(x, Asym, xmid, scal)), error=function(e) NULL)
if(!is.null(m7)){
  dyEst<- predict(m7)
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
```

```
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(coef(m7)),
  Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
  Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "SShaped",
  Function="dy~ SSlogis(x, Asym, xmid, scal)", Model= "m7",
   Isnull=is.null(m7) ) ))
predictions<- cbind(predictions, M7=yEst)}

m8<- NULL
m8<- tryCatch(nls(y~ SSasymp(x, Asym, R0, lrc)), error=function(e) NULL)
if(!is.null(m8)){
  dyEst<- predict(m8)
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(coef(m8)),
  Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
  Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "SShaped",
  Function="dy~ SSasymp(x, Asym, R0, lrc)", Model= "m8",
  Isnull=is.null(m8) ) ))
predictions<- cbind(predictions, M8=yEst)}

m11<- NULL
logy<- log(-y)
m11<- tryCatch(gnls(logy~ a+b*x, start=c(a=0, b=1)), error=function(e) NULL)
if(!is.null(m11)){
  dyEst<- -exp(predict(m11))
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(m11$coefficients),
  Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
  Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "ExpFunc",
  Function="log-dy~ a+b*x", Model= "m11", Isnull=is.null(m11) ) ))
predictions<- cbind(predictions, M11=yEst)}

if(!is.null(m7))
{
  m13<- NULL
  m13<- tryCatch( gnls(y ~ Asym/(1+exp((xmid-x)/scal)),
  start = coef(m7)), error=function(e) NULL)
  if(!is.null(m13)){
    dyEst<- predict(m13)
    yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
    yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
    yEst<- c(y1[1],yEst)
    results<- rbind.fill(results, data.frame(cbind(rbind(m13$coefficients),
```

```
    Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
    Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "SShaped",
    Function="dy ~ Asym/(1+exp((xmid-x)/scal)", Model= "m13",
    Isnull=is.null(m13)  ) ))
  predictions<- cbind(predictions, M13=yEst)}
}

m16<- NULL
m16<- tryCatch( gnls(y ~max(y)/(1+exp((xmid-x)/scal)),
start = c(xmid=1, scal=1)), error=function(e) NULL)
if(!is.null(m16)){
  dyEst<- predict(m16)
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(m16$coefficients),
  Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
  Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "SShaped",
  Function="dy ~ max(dy)/(1+exp((xmid-x)/scal)", Model= "m16",
  Isnull=is.null(m16)  ) ))
predictions<- cbind(predictions, M16=yEst)}

m17<- NULL
m17<- tryCatch( gnls(y ~max(y)-(max(y)-min(y))/(1+exp((xmid-x)/scal)),
start = c(xmid=1, scal=1)), error=function(e) NULL)
if(!is.null(m17)){
  dyEst<- predict(m17)
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(m17$coefficients),
  Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
  Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "SShaped",
  Function="dy ~ max(dy)-(max(dy)-min(dy))/(1+exp((xmid-x)/scal))",
  Model= "m17", Isnull=is.null(m17)  ) ))
predictions<- cbind(predictions, M17=yEst)}

m18<- NULL
m18<- tryCatch( gnls(y ~min(y) +(max(y)-min(y))/(1+exp((xmid-x)/scal)),
start = c(xmid=1, scal=1)), error=function(e) NULL)
if(!is.null(m18)){
  dyEst<- predict(m18)
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(m18$coefficients),
  Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
  Sigma=sqrt(sum((y1-yEst)^2)/length(y1)),
```

```
    Method= "SShaped",
    Function="dy ~ min(dy) +(max(dy)-min(dy))/(1+exp((xmid-x)/scal)",
    Model= "m18", Isnull=is.null(m18) ) ))
  predictions<- cbind(predictions, M18=yEst)}

  results<- results[which(results$Isnull == FALSE),]
  results2<-results
  predictions2<-predictions
  output2<-list(results2, predictions2)
  return (output2)
}



curvefitdxdy<- function(dx,y,dy){
  x<- dx
  y1<-y
  y<-dy
  results<-data.frame()
  predictions<-NULL

  #Linear Models
  m1<-NULL
  m1<- tryCatch(gnls(y~ a+ b*x , start= c(a=0, b=1)), error=function(e) NULL)
  if(!is.null(m1)){
    dyEst<- predict(m1)
    yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
    yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
    yEst<- c(y1[1],yEst)
    results<- rbind.fill(results, data.frame(cbind(rbind(m1$coefficients),
    Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
    Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "LinFunc",
    Function="dy~ a+ b*x", Model= "m1", Isnull=is.null(m1) ) ))
    predictions<- cbind(predictions, M1=yEst)}

  m2<-NULL
  m2<- tryCatch(lm(y~ x), error=function(e) NULL)
  if(!is.null(m2)){
    dyEst<- predict(m2)
    yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
    yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
    yEst<- c(y1[1],yEst)
    results<- rbind.fill(results, data.frame(cbind(rbind(m2$coefficients),
    Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
    Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "LinFunc",
    Function="dy~ a+ b*x", Model= "m2", Isnull=is.null(m2) ) ))
    predictions<- cbind(predictions, M2=yEst)}
```

```
#Exponential and Sshaped Models
m3<- NULL
logy<- log(y)
m3<- tryCatch(gnls(logy~ a+b*x, start=c(a=0, b=1)), error=function(e) NULL)
if(!is.null(m3)){
  dyEst<- exp(predict(m3))
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(m3$coefficients),
  Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
   Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "ExpFunc",
   Function="logdy~ a+b*x", Model= "m3", Isnull=is.null(m3) ) ))
  predictions<- cbind(predictions, M3=yEst)}

m7<- NULL
m7<- tryCatch(nls(y~ SSlogis(x, Asym, xmid, scal)), error=function(e) NULL)
 if(!is.null(m7)){
  dyEst<- predict(m7)
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(coef(m7)),
  Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
  Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "SShaped",
   Function="dy~ SSlogis(x, Asym, xmid, scal)", Model= "m7",
   Isnull=is.null(m7) ) ))
  predictions<- cbind(predictions, M7=yEst)}

m8<- NULL
m8<-  tryCatch(nls(y~ SSasymp(x, Asym, R0, lrc)), error=function(e) NULL)
 if(!is.null(m8)){
  dyEst<- predict(m8)
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(coef(m8)),
  Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
   Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "SShaped",
    Function="dy~ SSasymp(x, Asym, R0, lrc)", Model= "m8",
     Isnull=is.null(m8) ) ))
  predictions<- cbind(predictions, M8=yEst)}


m11<- NULL
logy<- log(-y)
m11<- tryCatch(gnls(logy~ a+b*x, start=c(a=0, b=1)), error=function(e) NULL)
if(!is.null(m11)){
```

```
  dyEst<- -exp(predict(m11))
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(m11$coefficients),
  Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
  Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "ExpFunc",
  Function="log-dy~ a+b*x", Model= "m11", Isnull=is.null(m11) ) ))
  predictions<- cbind(predictions, M11=yEst)}

if(!is.null(m7))
{
  m13<- NULL
  m13<- tryCatch( gnls(y ~ Asym/(1+exp((xmid-x)/scal)),
  start = coef(m7)), error=function(e) NULL)
  if(!is.null(m13)){
    dyEst<- predict(m13)
    yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
    yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
    yEst<- c(y1[1],yEst)
    results<- rbind.fill(results, data.frame(cbind(rbind(m13$coefficients),
    Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
    Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "SShaped",
    Function="dy ~ Asym/(1+exp((xmid-x)/scal)", Model= "m13",
    Isnull=is.null(m13) ) ))
    predictions<- cbind(predictions, M13=yEst)}
}

m16<- NULL
m16<- tryCatch( gnls(y ~max(y)/(1+exp((xmid-x)/scal)),
start = c(xmid=1, scal=1)), error=function(e) NULL)
if(!is.null(m16)){
  dyEst<- predict(m16)
  yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
  yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
  yEst<- c(y1[1],yEst)
  results<- rbind.fill(results, data.frame(cbind(rbind(m16$coefficients),
   Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
   Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "SShaped",
   Function="dy ~ max(dy)/(1+exp((xmid-x)/scal)", Model= "m16",
    Isnull=is.null(m16) ) ))
  predictions<- cbind(predictions, M16=yEst)}

m17<- NULL
m17<- tryCatch( gnls(y ~max(y)-(max(y)-min(y))/(1+exp((xmid-x)/scal)),
 start = c(xmid=1, scal=1)), error=function(e) NULL)
if(!is.null(m17)){
  dyEst<- predict(m17)
```

```
    yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
    yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
    yEst<- c(y1[1],yEst)
    results<- rbind.fill(results, data.frame(cbind(rbind(m17$coefficients),
    Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
    Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "SShaped",
     Function="dy ~max(dy)-(max(dy)-min(dy))/(1+exp((xmid-x)/scal))",
     Model= "m17", Isnull=is.null(m17) ) ))
    predictions<- cbind(predictions, M17=yEst)}

  m18<- NULL
  m18<- tryCatch( gnls(y ~min(y) +(max(y)-min(y))/(1+exp((xmid-x)/scal)),
  start = c(xmid=1, scal=1)), error=function(e) NULL)
  if(!is.null(m18)){
    dyEst<- predict(m18)
    yEst<- lapply(1:(length(dyEst)), function(i) sum(dyEst[1:i]))
    yEst<- apply( cbind( yEst ) , 1 , unlist ) +y1[1]
    yEst<- c(y1[1],yEst)
    results<- rbind.fill(results, data.frame(cbind(rbind(m18$coefficients),
    Mape=100*sum(abs((y1-yEst)/y1))/length(y1),
    Sigma=sqrt(sum((y1-yEst)^2)/length(y1)), Method= "SShaped",
     Function="dy ~ min(dy) +(max(dy)-min(dy))/(1+exp((xmid-x)/scal)",
     Model= "m18", Isnull=is.null(m18) ) ))
    predictions<- cbind(predictions, M18=yEst)}



  results<- results[which(results$Isnull == FALSE),]
  results3<-results
  predictions3<-predictions
  output3<-list(results3, predictions3)
  return (output3)
}

library(nlme)
library(plyr)
library(data.table)
library(purrr)

checkRelation<- function(k){
   a<-grepl("dx",as.character(k$Function))
  if(a){
    k<- data.frame(k, Relation= "DXtoDY")
  }
  else{
    b<-grepl("dy",as.character(k$Function))
    if(b){
      k<- data.frame(k, Relation= "XtoDY")
```

```
    }
    else{
      k<- data.frame(k, Relation= "XtoY")
    }}
  return(k)
}
curveFitAll<- function(x1,y1,dx1=NA,dy1=NA){

  if(is.na(dx1)){
    dx1<- lapply(1:(length(x1)), function(i) (x1[i+1]-x1[i]))
    dx1<- dx1[-length(dx1)]
    dx1<- apply( cbind( dx1 ) , 1 , unlist )}

  if(is.na(dy1))
  { dy1<- lapply(1:(length(y1)), function(i) (y1[i+1]-y1[i]))
    dy1<- dy1[-length(dy1)]
    dy1<- apply( cbind( dy1 ) , 1 , unlist )}

  fitxy<- curvefit(x1,y1)
  fitxdy<- curvefitxdy(x1,y1,dy1)
  fitdxdy<- curvefitdxdy(dx1,y1,dy1)

  colnames(fitxdy[[2]])<- if(nrow(fitxdy[[1]])>0){paste0("XDY",
  colnames(fitxdy[[2]]) ) }
  colnames(fitdxdy[[2]])<- if(nrow(fitdxdy[[1]])>0){ paste0("DXDY",
  colnames(fitdxdy[[2]]) )}

  overallresults<- rbind.fill(fitxy[[1]],fitxdy[[1]],fitdxdy[[1]])
  listedresults<-lapply(1:nrow(overallresults),
  function(i) checkRelation(overallresults[i,]))
  overallresults<- apply( t(rbind.fill( listedresults )) , 1 , unlist )

  overallpredictions<- NULL
  if(nrow(fitxy[[1]])>0)
  {overallpredictions<-cbind(overallpredictions, fitxy[[2]])}
  if(nrow(fitxdy[[1]])>0)
  {overallpredictions<-cbind(overallpredictions, fitxdy[[2]])}
  if(nrow(fitdxdy[[1]])>0)
  {overallpredictions<-cbind(overallpredictions, fitdxdy[[2]])}

  outputall<- list(overallresults,overallpredictions)
  return(outputall)
}


#Applying Curve Fit
    {
      fit1<- curveFitAll(x,y)
```

```
fit2<- curveFitAll(x=y,y=x)
#FOR FIT1
{results<- as.data.frame(fit1[[1]])
  #sigmaCol<- results[, which(colnames(results)=="Sigma")]
  #sigmaCol<- as.numeric(levels(sigmaCol))[sigmaCol]
  #results[, which(colnames(results)=="Sigma")]<- round(sigmaCol,
   digits= 2)
  results$Sigma<- as.numeric(levels(results$Sigma))[results$Sigma]
  results$Sigma<- round(results$Sigma, digits= 4)
  results$Mape<- as.numeric(levels(results$Mape))[results$Mape]
  results$Mape<- round(results$Mape, digits= 4)

  Pareto1 <- psel(results, low(results$Sigma) * low(results$Mape),
   top=min(10,nrow(results)))
  Pareto1
  all<-data.frame(Sigma=Pareto1$Sigma,Mape=Pareto1$Mape,
  Method=Pareto1$Method,Function=Pareto1$Function,
  Relation=Pareto1$Relation)
  all
  # best<-data.frame(Sigma=Pareto1[1,]$Sigma,
  Mape=Pareto1[1,]$Mape,Method=Pareto1[1,]$Method,
  Function=Pareto1[1,]$Function, Relation=Pareto1[1,]$Relation)
  # best

  predictions<- as.data.frame(fit1[[2]])
  bestindex<-as.numeric(rownames(Pareto1[1,]))
  yestimate<-predictions[,as.numeric(rownames(Pareto1[1,]))]
  yestimate<-as.matrix(yestimate)
}
# PLOT for FIT1
{ plot(x,y, type="l", ylim=c(min(min(yestimate[,1]),min(y)),
max(max(yestimate[,1]),max(y))))
  lines(x,yestimate[,1], col=2)
}
#FOR FIT2
{ results<- as.data.frame(fit2[[1]])
  results$Sigma<- as.numeric(levels(results$Sigma))[results$Sigma]
  results$Sigma<- round(results$Sigma, digits= 5)
  results$Mape<- as.numeric(levels(results$Mape))[results$Mape]
  results$Mape<- round(results$Mape, digits= 5)

  Pareto2 <- psel(results, low(results$Sigma) *
  low(results$Mape), top=min(10,nrow(results)))
  Pareto2
  all<-data.frame(Sigma=Pareto2$Sigma,Mape=Pareto2$Mape,
  Method=Pareto2$Method,Function=Pareto2$Function,
  Relation=Pareto2$Relation)
  all
```

```
    # best<-data.frame(Sigma=Pareto2[1,]$Sigma,
    Mape=Pareto2[1,]$Mape,Method=Pareto2[1,]$Method,
    Function=Pareto2[1,]$Function,
    Relation=Pareto2[1,]$Relation)
    # best

    predictions<- as.data.frame(fit2[[2]])
    bestindex<-as.numeric(rownames(Pareto2[1,]))
    yestimate<-predictions[,as.numeric(rownames(Pareto2[1,]))]}
  # PLOT FOR FIT2
  {plot(y,x, type="l", ylim=c(min(min(yestimate),
  min(x)), max(max(yestimate),max(x))))
    lines(y,yestimate, col=2)
  }
}
```