Measuring Eigenvalues' Sensitivities via Kernel Canonical Correlation Analysis

Ahmed F. Yehia [a], Mohamed Saleh [a], Ayman Taha [b] and Hisham El-Shishiny [c]

a Operations Research & Decision Support Department, Faculty of Computers and Information, Cairo University

b Information System Department, Faculty of Computers and Information, Cairo University

c Advanced Technology and Center for Advanced Studies, IBM Cairo Technology Development Center, Cairo – Egypt.

Abstract:

This paper is a continuation to our previous work [Yehia, Saleh et al., System Dynamics Conf. (2014)], where we developed a many to one statistical sensitivity analysis method based on the multivariate maximal information coefficient (MMIC). The two main critiques to this method (which we previously developed), were that the complexity of the algorithm increases exponentially with the number of input variables, and that it cannot handle many (inputs) to many (outputs) relations. To overcome these two critiques, in this paper, we propose a statistical sensitivity analysis based on the Kernel Canonical Correlation method. This kernel-based method is designed to handle many to many relations, and the complexity of the algorithm depends only on the number of samples – whatever the number of input variables is. We postulate that this kernel based sensitivity analysis represents a solid foundation to study the multivariate complex nonlinear non-monotonic relationships between behavior modes – expressed by eigenvalues – and the model parameters. The experiments conducted corroborate our postulation.

Keywords: *Sensitivity Analysis, Eigenvalue Analysis, Canonical Correlation and Kernel*

1. Introduction and Theoretical Background:

Eigenvalue analysis studies the sensitivity of behavior modes (eigenvalues) – or their associated weights – to the gains of feedback loops. In general, the gains of feedback loops are functions of the gains of links; which in turn are functions of parameters, in the model. Hence one can link parameters (input variables) to certain eigenvalues (output variables). Eigenvalues are computed from the Jacobian matrix **J**. Moreover, for each eigenvalue, $\boldsymbol{\lambda_i}$, there is a distinct sensitivity matrix, $\mathbf{S_i}$, which is equal to the product of the left-eigenvector and the transpose of the right-eigenvector.

$$S_i = \begin{bmatrix} \partial\lambda_i / \partial J(1,1) & \cdots & \partial\lambda_i / \partial J(1,n) \\ \vdots & \cdots & \vdots \\ \partial\lambda_i / \partial J(n,1) & \cdots & \partial\lambda_i / \partial J(n,n) \end{bmatrix}$$

The majority of previous works in this area (Forrester 1983, Güneralp 2006, Kampmann and Oliva 2008, Gonçalves 2009, Saleh, Oliva et al. 2010) utilized the traditional univariate sensitivity (or elasticity) measure, which is based on the first order partial derivative of the eigenvalue with respect to an independent variable. This univariate measure represents only the marginal contribution of the independent variable – assuming that all other independent variables are constants. However, as Sterman put it: "*In nonlinear systems, the sensitivity of a system to variations in multiple parameters is not a simple combination of the response to the parameters varied alone*" (Sterman 2000).

Some scholars adopted multivariate linear regression analysis (Hekimoğlu and Barlas 2010, Tøndel, Vik et al. 2013). Regression analysis is under the umbrella of multivariate analysis; i.e. can study simultaneous changes in more than one parameter (Esbensen, Guyot et al. 2002). Linear regression analysis aims to find a linear relationship between the dependent variable and independent variables. The core algorithm of linear regression is the least squares errors algorithm, which is used in data fitting (Rencher and Christensen 2012). Yet, the main limitation here is the assumption of linear relationship between the dependent variable and independent variables.

Nonlinear regression analysis can be used when linear regression fails. However, the analyst must try different functions; e.g. exponential, logarithmic, etc. In addition initial values for the coefficients are needed; and in general, the solution changes according to initial settings (Draper, Smith et al. 1966, Hair, Black et al. 2006). That is, there is no guarantee to reach the global minimum (least squares errors); as there might be several local minima. Moreover the solution of complex nonlinear regression equations might not converge. Finally, nonlinear multivariate regression is not well suited to handle non-monotonic functions (Rencher and Christensen 2012).

Applying the Multivariate Maximal Information Coefficient (MMIC) to Eigenvalue Analysis outperforms the traditional sensitivity analysis methods (Yehia, Saleh et al. 2014). MMIC has several advantages; as it can capture any non-linear relationship between a set of inputs and a single output variable; moreover it measures the total contribution of each input – rather than its marginal contribution (Reshef, Reshef et al. 2011, Yehia, Saleh et al. 2014). On the other hand, there are limitations to using MMIC. The computational time of MMIC increases exponentially with the number of input variables. Moreover, it cannot handle multiple output variables.

The solution proposed in this paper is a "many to many" kernel-based sensitivity analysis method, which preserves all MMIC advantages; in addition to overcoming its limitations. This solution is illustrated in the next section.

The rest of this paper is organized as follows: In section 2, we explain the proposed solution. In section 3, we illustrate the experiments conducted to test the proposed solution. Finally, we conclude, in section 4.

2. Proposed Solution

This section is divided into three subsections: Description of Inputs and Outputs, Linear Canonical Correlation Analysis, and Kernel Canonical Correlation Analysis.

2.1 Description of Inputs and Outputs

The goal of our study is to link parameters in the model (input variables) to the most dominant eigenvalues (output variables). Recall that the eigenvalues are computed via the Jacobian matrix (which can be considered a condensed representation of the model structure).

Let matrix **X** represents the data associated with the multiple input variables. **X** is a data matrix of n-by-d1 size; with d1 variables in columns, and n data points (samples) in rows. Then the d1-by-d1 covariance matrix is given by $\mathbf{X^T X}$ [1] (multiplied by a scalar value). At the same time, one can compute the standard Gram matrix $\mathbf{X X^T}$ of the n-by-n size.

Let matrix **Y** represents the data associated with the multiple output variables -- i.e. the most dominant eigenvalues. **Y** is a data matrix of n-by-d2 size; with d2 variables in columns, and n data points (samples) in rows. Then the d2-by-d2 covariance matrix is given by $\mathbf{Y^T Y}$ (multiplied by a

[1] Superscript T refers to the transpose operation

scalar value). At the same time, one can compute the standard Gram matrix $\mathbf{YY^T}$ of the n-by-n size.

Note that for mathematical clarity, we assume that all data (**X** &**Y**) are centered (i.e. zero mean). In general, one can easily remove the mean by shifting the data. Remark that, in this paper, matrices are labeled by bold and upper-case letters; while vectors are labeled by bold and lower-case letters.

2.2 Linear Canonical Correlation Analysis

Linear Canonical Correlation analysis (LCCA) measures the linear relationship between two sets of variables (**X** set & **Y** set). One can consider the LCCA as an extension to the standard Pearson correlation coefficient (r). Recall that the standard correlation coefficient r measures the extent to which two variables are linearly related. The core of LCCA is based on the standard correlation coefficient r; In fact, LCCA maximizes the standard correlation coefficient r between two vectors (as shown in the figure below). The first vector is the canonical variate of the **X** – denoted by the **u** vector. This vector represents the best linear combinations of the multiple independent variables. The second vector is the canonical variate of the **Y** – denoted by the **v** vector. This vector represents the best linear combinations of the multiple dependent variables (Johnson and Wichern 1992, Bach and Jordan 2005).

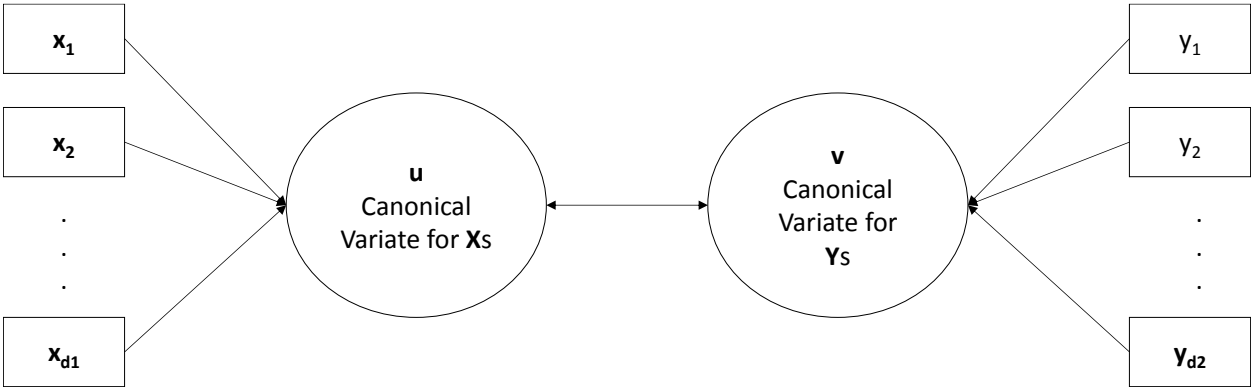

Fig 1: Linear Canonical Correlation Analysis (Clark 2009).

In other words, LCCA finds linear combinations of the columns of **X** and columns of **Y**, which have maximum correlation with each other. Specifically, the objective is to maximize the standard correlation coefficient r, specified as follows:

$$r = \frac{\mathbf{u}^{\mathrm{T}}\mathbf{v}}{\|\mathbf{u}\| \ \|\mathbf{v}\|}$$

Assuming that all data are centered (i.e. zero mean).

Where the **u** vector (size n) is the Canonical Variate of the **X** (data matrix of n-by-d1 size); i.e.

$$\mathbf{u} = \mathbf{Xa}$$

And the **v** vector (size n) is the Canonical Variate of the **Y** (data matrix of n-by-d2 size); i.e.

$$\mathbf{v} = \mathbf{Yb}$$

The elements of **a** and **b** vectors are the decisions variables that we want to optimize, in order to maximize r. Note that the size of vector **a** is d1; and the size of vector **b** is d2.

The above fractional programming problem can be transformed to the bi-objective optimization problem of simultaneously maximizing the numerator ($\mathbf{u}^T\mathbf{v}$), and minimizing the dominator ($\|\mathbf{u}\| \ \|\mathbf{v}\|$). An efficient solution of this bi-objective optimization problem can be obtained via maximizing the numerator subject to an inequality constraint that forces the dominator to less than or equal to a certain fixed value. In our case, one can decompose the dominator inequality constraint into the following two inequality constraints:

$$\|\mathbf{u}\| \leq 1$$

$$\|\mathbf{v}\| \leq 1$$

I.e. the Euclidean length of both the **u** vector and the **v** vector must be less than or equal to unity. In other words, we want both the **u** and **v** vectors to be just direction vectors.

The above two constraints are equivalent to the following two constraints:

$$\mathbf{u}^T\mathbf{u} \leq 1$$

$$\mathbf{v}^T\mathbf{v} \leq 1$$

Hence, the optimization can be summarized as follows:

Max. $\mathbf{u}^T\mathbf{v}$
Subject to:

$$\mathbf{u}^T\mathbf{u} \leq 1$$

$$\mathbf{v}^T\mathbf{v} \leq 1$$

Substituting the equations of **u** and **v** into the optimization problem yields:
Max. $\mathbf{a}^T\mathbf{X}^T\mathbf{Yb}$
Subject to:

$$\mathbf{a}^T\mathbf{X}^T\mathbf{Xa} \leq 1$$

$$\mathbf{b}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\mathbf{b} \leq 1$$

In our research, we used the "canoncorr" MATLAB function (in the MATLAB Statistics Toolbox) to solve the above optimization problem. The inputs to the function are the **X** and **Y** matrices, and the output is the optimal value for r. Note that, in the case of a single output variable, the optimal value for r is equal to the square root of the coefficient of determination, R^2, for the corresponding linear multiple regression model (Everitt 2002).

2.3 Kernel Canonical Correlation Analysis

The Kernel Canonical Correlation Analysis (KCCA) is an extension to the LCCA. KCCA measures the non-linear relationship between two sets of variables (i.e. the **X** set and the **Y** set). In the kernel approach, we consider a transfer function that maps each data point to a higher dimension space. The idea of KCCA is to perform the standard linear canonical correlation in this new space; i.e. the mapping into this new space transforms the non-linear relationship (between the two sets of variables) into a linear one. The dimensionality of this new space is usually very large (infinite in some cases). Nevertheless, we can always compute the so-called kernel Gram matrix. Elements of this kernel Gram matrix are computed via a kernel function, which is based on the inner product of the transfer function. In practice, one does not need to compute the transfer function, but only the associated kernel function. This is called the kernel trick (Hardoon, Szedmak et al. 2004, Drineas and Mahoney 2005, Huang, Lee et al. 2006, Welling 2011). Moreover, in practice, one can construct a kernel Gram matrix, without even knowing the dimensionality of the new space (or anything about the new space). The size of the kernel Gram matrix will always be n-by-n. In general the kernel Gram matrix is valid and useful, if it satisfies certain conditions – e.g. must be symmetric (i.e. the matrix is equal to its transpose).

In an analogy to the linear case, the kernel optimization problem can be formulated as follows (Hardoon, Szedmak et al. 2004).

Max. $\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}_x\mathbf{K}_y\boldsymbol{\beta}$
Subject to:

$$\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}_x{}^2\boldsymbol{\alpha} \leq 1$$
$$\boldsymbol{\beta}^{\mathrm{T}}\mathbf{K}_y{}^2\boldsymbol{\beta} \leq 1$$

The elements of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ vectors are the new decisions variables that we want to optimize. Note that, in the kernel case, both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ vectors are of size n. As Welling put it: "…$\boldsymbol{\alpha}$ is a vector in a different N-dimensional space than e.g. **a** which lives in a D-dimensional space…" (Welling 2011, p.71).

Similar to the linear case, we can use the "canoncorr" MATLAB function; yet, in this non-linear case, the inputs to the Matlab function are the $\mathbf{K_x}$ and $\mathbf{K_y}$ matrices (instead of the **X** and **Y** matrices). For simplicity, in this paper, we will use the polynomial kernel function, which is defined as follows (Chang, Hsieh et al. 2010).

$$\mathbf{K}_{\boldsymbol{x}} = \left(\mathbf{X}\mathbf{X}^{\mathbf{T}} + \mathrm{c}\right)^{.\wedge \mathrm{p}}$$

Where the operator ".^" donates an element by element power, p is the degree of the polynomial, and c is a constant that scales the influence of higher-order terms (in the polynomial) relative to the lower-order terms. Note that the same function form is applicable to $\mathbf{K_y}$.

In the next section, we will apply the polynomial KCCA on a simple model, in order to compute the optimal nonlinear correlation coefficient between any set of parameters (in the model) and any set of (dominant) eigenvalues. This will enable us to rank parameters according to their influences on any set of eigenvalues.

3. Experiments

This section shows the experiments conducted in order to illustrate the application of LCCA and KCCA to eigenvalue analysis. The dynamic model used in the experiments is the simple hypothetical dynamic model presented in our last year's paper (Yehia, Saleh et al. 2014). The stock and flow diagram of the model is shown in the following figure. The model consists of two stocks: S_1 and S_2. R_1 and R_2 are the inflows of S_1 and S_2 respectively. Moreover, there are three auxiliary variables: g_{11}, g_{12}, g_{21}. The equations of the model are as follows:

- X_1, X_2 & X_3 are uniform random variables [0,1] . These are the input variables (parameters)
- $g_{11} = X_1 * X_2 * X_3$
- $g_{12} = X_1 * X_2$
- $g_{21} = X_1 * X_3$
- $R_1 = g_{11}*S_1 + g_{12}*S_2 + 1$
- $R_2 = S_1*g_{21}$

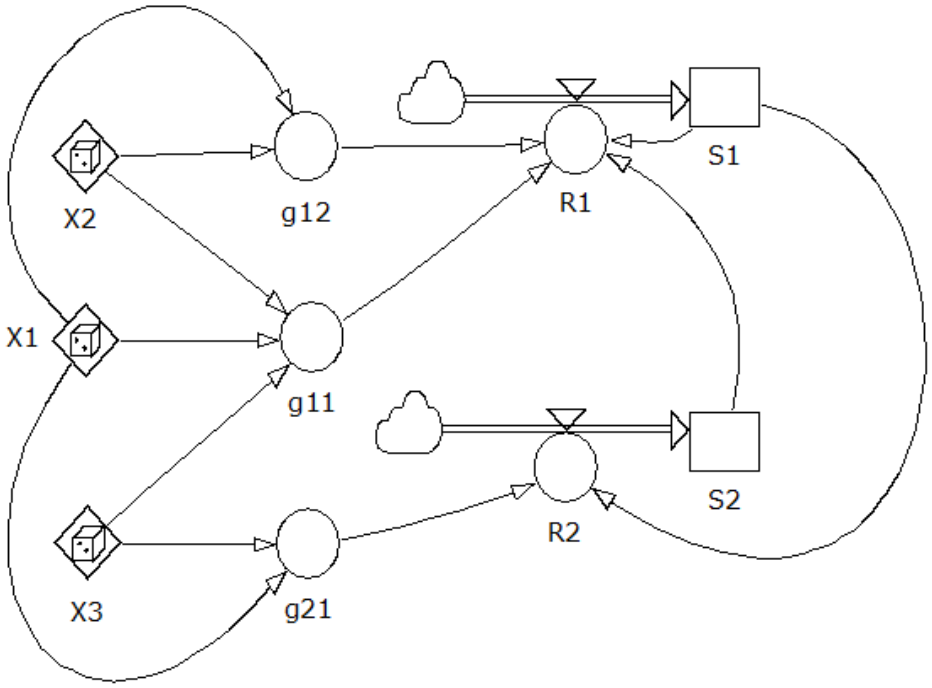

Fig 2: The Stock & Flow Diagram of the Model used in the Eigenvalue Experiments.

The results of the experiments are shown in tables 1 and 2. Table 1 shows the results associated with a single output variable, which is the most dominant eigenvalue. While, table 2 shows the results associated with two output variables, which are the two eigenvalues, in the model. Note that since there are two stocks, in the model, then there are only two eigenvalues. Recall that the eigenvalues are computed via the Jacobian matrix (which can be considered a condensed representation of the model structure). Note that the eigenvalues of this model are always real numbers (i.e. not complex numbers) -- whatever the values of the parameters.

To facilitate the replications of the results, the Matlab code used to conduct the experiments is shown in Appendix A. In the LCCA case (i.e. the first row in both tables), the Matlab code computes the optimal linear correlation coefficient (i.e. the association level) between any set of parameters (in the model) and any set of (dominant) eigenvalues. While, in the KCCA case (i.e. the remaining rows in both tables), the Matlab code computes the optimal nonlinear correlation coefficient (i.e. the association level) between any set of parameters and any set of (dominant) eigenvalues. In both tables, the second row shows the results associated with applying polynomial KCCA of degree 1 (for various sets of inputs); and the third row shows the results associated with applying polynomial KCCA of degree 2. We stopped at the second degree, because for this degree, we reached full association; i.e. approximately the value of "1" for the nonlinear correlation coefficient relating all input variables to the output variables(s). In more complex models, we will need to increase further the degree of the polynomial, until we reach the full association.

Note that, for the results shown below, we had set the number of samples "n" to 5000, and we had set "c" (the coefficient in the polynomial kernel equation) to 1. After conducting many experiments, we reached the conclusion that 5000 is more than enough for the sample size (i.e. there are no significant changes, in the results, when we increase the sample size above 5000; or when we change the seed for the random number generator). We also tried several values for 'c', and the results did not change significantly; except when we set c equal to zero (as in this case, we totally ignore the linear term in the polynomial).

Table 1: The association level between any set of input variables and the dominant eigenvalue

	X1	X2	X3	X1 & X2	X1 & X3	X2 & X3	X1, X2 & X3
LCCA	0.63	0.46	0.46	0.77	0.77	0.65	0.90
KCCA degree: 1	0.63	0.46	0.46	0.77	0.77	0.65	0.90
KCCA degree: 2	0.69	0.47	0.46	0.83	0.84	0.69	0.99

Table 2: The association level between any set of input variables and the two eigenvalues

	X1	X2	X3	X1 & X2	X1 & X3	X2 & X3	X1, X2 & X3
LCCA	0.88	0.52	0.52	0.89	0.89	0.74	0.93
KCCA degree: 1	0.88	0.52	0.52	0.89	0.89	0.74	0.93
KCCA degree: 2	0.92	0.60	0.59	0.94	0.94	0.94	0.99

From the above two tables we can conclude the following:

1. LCCA is equivalent to polynomial KCCA of the first degree (i.e. in both tables, any result shown in the first row is equal to the corresponding result in the second row).
2. The association level increases as we increase the degree of the polynomial in the KCCA.
3. The association level increases as we include more input variables.
4. The eigenvalues are sensitive to all input parameters. However, X1 has the highest impact on the eigenvalues. This is consistent with the equations of the model.

Remark that, in the case of a single output variable the optimal value of r associated with LCCA -- i.e. any result shown in the first row (or the second row) of table 1 -- must equal to the square root of the coefficient of determination, R^2, for the corresponding linear multiple regression model. I.e. conceptually (in this case), the linear multiple regression process is equivalent to the LCCA, which in turn is equivalent to the polynomial KCCA of degree 1. In our experiments, we empirically verified the above fact.

Our final remark, in this section, is that computing the association measure via the KCCA analysis only takes few seconds (for n=5000). This is because the computational time of KCCA depends on the number of samples "n", rather than the number of input or output variables. In fact, KCCA has a polynomial time complexity of $O(n^3)$ (Rasiwasia, Mahajan et al. 2014); unlike the exponential time complexity of the MMIC method.

5. Conclusion

In this paper, we presented a many to many nonlinear sensitivity method for eigenvalues, which is based on Kernel Canonical Correlation analysis. This sensitivity method overcomes the research gaps associated with rival methods. Specifically, the proposed method not only preserves the advantages of the multivariate maximal information coefficient method, but also has polynomial time complexity. Moreover, there is no need to simulate the underline model. The proposed solution only interacts with a condensed matrix representation of the model; i.e. the Jacobian matrix. In each run, the process, which takes time, is the computation of eigenvalues from the Jacobian matrix; and there are algorithms that compute eigenvalues very fast. Via this method, decision-makers can rank policy parameters according to their impacts on the dominant eigenvalues.

In the near future, we will continue the experimental work, and test the method on complex models. In addition, we plan to develop a wrapper based parameter selection method to facilitate the automatic ranking of parameters. Finally, in many nonlinear dynamic models, eigenvalues depend on the current state of the model. For these cases, we plan to devise an innovative framework that links the time trajectory of the dominant eigenvalues with parameters.

ACKNOWLEDGEMENT

This work is part of an R&D project funded by an IBM Faculty Award.

References

Bach, F. R. and M. I. Jordan (2005). A probabilistic interpretation of canonical correlation analysis. Department of Statistics, University of California, Berkeley.

Chang, Y.-W., C.-J. Hsieh, K.-W. Chang, M. Ringgaard and C.-J. Lin (2010). "Training and testing low-degree polynomial data mappings via linear SVM." The Journal of Machine Learning Research 11: 1471-1490.

Clark, M. (2009). "Canonical Correlation." from www.unt.edu/rss/class/mike/6810/Cancorr.pdf.

Draper, N. R., H. Smith and E. Pownell (1966). Applied regression analysis, Wiley New York.

Drineas, P. and M. W. Mahoney (2005). "On the Nyström method for approximating a Gram matrix for improved kernel-based learning." The Journal of Machine Learning Research 6: 2153-2175.

Esbensen, K. H., D. Guyot, F. Westad and L. P. Houmoller (2002). Multivariate data analysis: in practice: an introduction to multivariate data analysis and experimental design, Multivariate Data Analysis.

Everitt, B. S. (2002). The Cambridge Dictionary of Statistics. 2nd edition. . Cambridge, UK. , Cambridge University Press.

Forrester, N. (1983). Eigenvalue analysis of dominant feedback loops. Plenary Session Papers Proceedings of the 1st International System Dynamics Society Conference.

Gonçalves, P. (2009). "Behavior modes, pathways and overall trajectories: eigenvector and eigenvalue analysis of dynamic systems." System dynamics review 25(1): 35-62.

Güneralp, B. (2006). "Towards coherent loop dominance analysis: progress in eigenvalue elasticity analysis." System dynamics review 22(3): 263-289.

Hair, J. F., W. C. Black, B. J. Babin, R. E. Anderson and R. L. Tatham (2006). Multivariate data analysis, Pearson Prentice Hall Upper Saddle River, NJ.

Hardoon, D., S. Szedmak and J. Shawe-Taylor (2004). "Canonical correlation analysis: An overview with application to learning methods." Neural computation 16(12): 2639-2664.

Hekimoğlu, M. and Y. Barlas (2010). "Sensitivity Analysis of System Dynamics Models by Behavior Pattern Measures."

Huang, S.-Y., M.-H. Lee and C. K. Hsiao (2006). "Kernel canonical correlation analysis and its applications to nonlinear measures of association and test of independence." Unpublished manuscript. http://[www.stat.sinica.edu.tw/syhuang/].

Johnson, R. A. and D. W. Wichern (1992). Applied multivariate statistical analysis, Prentice hall Englewood Cliffs, NJ.

Kampmann, C. E. and R. Oliva (2008). "Structural dominance analysis and theory building in system dynamics." Systems Research and Behavioral Science 25(4): 505-519.

Rasiwasia, N., D. Mahajan, V. Mahadevan and G. Aggarwal (2014). Cluster Canonical Correlation Analysis. Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics.

Rencher, A. C. and W. F. Christensen (2012). Methods of multivariate analysis, John Wiley & Sons.

Reshef, D. N., Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher and P. C. Sabeti (2011). "Detecting novel associations in large data sets." science 334(6062): 1518-1524.

Saleh, M., R. Oliva, C. E. Kampmann and P. I. Davidsen (2010). "A comprehensive analytical approach for policy analysis of system dynamics models." European Journal of Operational Research 203(3): 673-683.

Sterman, J. (2000). Business dynamics, Irwin-McGraw-Hill.

Tøndel, K., J. O. Vik, H. Martens, U. G. Indahl, N. Smith and S. W. Omholt (2013). "Hierarchical multivariate regression-based sensitivity analysis reveals complex parameter interaction patterns in dynamic models." Chemometrics and Intelligent Laboratory Systems 120: 25-41.

Welling, M. (2011). Kernel Canonical Correlation Analysis. "A first encounter with Machine Learning." Irvine, CA. University of California (2011): 1-93.

Yehia, A., M. Saleh, A. Taha and H. El-Shishiny (2014). A Novel Sensitivity Analysis for Dynamic Models. The 32rd International Conference of the System Dynamics Society. Delft, Netherlands.

Appendix A: The Matlab Code of the Experiments

To run our code you need the following two external m files:

1. Kernel.m which can be download from http://is.gd/icofit
2. canoncorr.m which can be found in the Matlab Statistics Toolbox

Our code consists of the folloing two m files:

1. Main.m (which is the main script)
2. ourcanoncorr.m (which is a simple wrapper function for the original canoncorr function)

Main.m

```
clc; clear all; close all; rng(0);

N = 5000; % Total number of Samples

X1 = rand(N, 1); % N samples for X1
X2 = rand(N, 1); % N samples for X2
X3 = rand(N, 1); % N samples for X3


for i=1:N
    j11 = X1(i)*X2(i)*X3(i);
    j12 = X1(i)*X2(i);
    j21 = X1(i)*X3(i);

    J =[j11 j12;j21 0];           % Jacobian matrix for sample i
    Yboth(i,:) = eig(J)';         % All eigenvalues for sample i
    Ymax(i,1) = max(Yboth(i,:)); % The Dominant eigenvalue for sample i

end

% Setting Y = Ymax produces the results in table# 1 in paper
% Setting Y = Yboth produces the results in table# 2 in paper
Y = Ymax;
% Y = Yboth;


c = 1; % The coefficient in the polynomial kernel function

% Computations of the various polynomial kernels for the input variables
kx1_p1= kernel(X1',X1','polynomial',1,c);
kx1_p2= kernel(X1',X1','polynomial',2,c);

kx2_p1= kernel(X2',X2','polynomial',1,c);
kx2_p2= kernel(X2',X2','polynomial',2,c);

kx3_p1= kernel(X3',X3','polynomial',1,c);
```

```
kx3_p2= kernel(X3',X3','polynomial',2,c);

kx12_p1= kernel([X1,X2]',[X1,X2]','polynomial',1,c);
kx12_p2= kernel([X1,X2]',[X1,X2]','polynomial',2,c);

kx13_p1= kernel([X1,X3]',[X1,X3]','polynomial',1,c);
kx13_p2= kernel([X1,X3]',[X1,X3]','polynomial',2,c);

kx23_p1= kernel([X2,X3]',[X2,X3]','polynomial',1,c);
kx23_p2= kernel([X2,X3]',[X2,X3]','polynomial',2,c);

kx123_p1= kernel([X1,X2,X3]',[X1,X2,X3]','polynomial',1,c);
kx123_p2= kernel([X1,X2,X3]',[X1,X2,X3]','polynomial',2,c);

% Computations of the various polynomial kernels for the output variables
ky_p1= kernel(Y',Y','polynomial',1,c);
ky_p2= kernel(Y',Y','polynomial',2,c);


Table = zeros (3,7); % Stores the results presented in table#1 or table #2

% Line 1 : LCCA
Table(1,1) = ourcanoncorr(X1,Y);
Table(1,2) = ourcanoncorr(X2,Y);
Table(1,3) = ourcanoncorr(X3,Y);
Table(1,4) = ourcanoncorr([X1,X2],Y);
Table(1,5) = ourcanoncorr([X1,X3],Y);
Table(1,6) = ourcanoncorr([X2,X3],Y);
Table(1,7) = ourcanoncorr([X1,X2,X3],Y);

% Line 2 : KCCA -- degree 1
Table(2,1) = ourcanoncorr(kx1_p1,ky_p1);
Table(2,2) = ourcanoncorr(kx2_p1,ky_p1);
Table(2,3) = ourcanoncorr(kx3_p1,ky_p1);
Table(2,4) = ourcanoncorr(kx12_p1,ky_p1);
Table(2,5) = ourcanoncorr(kx13_p1,ky_p1);
Table(2,6) = ourcanoncorr(kx23_p1,ky_p1);
Table(2,7) = ourcanoncorr(kx123_p1,ky_p1);

%Line 3 : KCCA -- degree 2
Table(3,1) = ourcanoncorr(kx1_p2,ky_p2);
Table(3,2) = ourcanoncorr(kx2_p2,ky_p2);
Table(3,3) = ourcanoncorr(kx3_p2,ky_p2);
Table(3,4) = ourcanoncorr(kx12_p2,ky_p2);
Table(3,5) = ourcanoncorr(kx13_p2,ky_p2);
Table(3,6) = ourcanoncorr(kx23_p2,ky_p2);
Table(3,7) = ourcanoncorr(kx123_p2,ky_p2);

% EOF
```

ourcanoncorr.m

```
function output = ourcanoncorr(X,Y)

      [A,B,r] = canoncorr(X,Y);
      output = r(1);

end
```