5-2018

Why Hammerstein-Type Block Models Are So Efficient: Case Study of Financial Econometrics

Thongchai Dumrongpokaphan
Chiang Mai University, tcd43@hotmail.com

Afshin Gholamy
The University of Texas at El Paso, afshingholamy@gmail.com

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Nguyen Hoang Phuong
Thang Long University, nhphuong2008@gmail.com

Comments:
Technical Report: UTEP-CS-18-45
Dumrongpokaphan T., Gholamy A., Kreinovich V., Nguyen H.P. (2019) Why Hammerstein-Type Block Models Are so Efficient: Case Study of Financial Econometrics. In: Kreinovich V., Thach N., Trung N., Van Thanh D. (eds) Beyond Traditional Probabilistic Methods in Economics. ECONVN 2019. Studies in Computational Intelligence, vol 809. Springer, Cham
https://doi.org/10.1007/978-3-030-04200-4_9

Why Hammerstein-Type Block Models Are So Efficient: Case Study of Financial Econometrics

Thongchai Dumrongpokaphan, Afshin Gholamy, Vladik Kreinovich, and Hoang Phuong Nguyen

Abstract In the first approximation, many economic phenomena can be described by linear systems. However, many economic processes are non-linear. So, to get a more accurate description of economic phenomena, it is necessary to take this non-linearity into account. In many economic problems, among many different ways to describe non-linear dynamics, the most efficient turned out to be Hammerstein-type block models, in which the transition from one moment of time to the next consists of several consequent blocks: linear dynamic blocks and blocks describing static non-linear transformations. In this paper, we explain why such models are so efficient in econometrics.

1 Formulation of the Problem

Linear models and need to go beyond them. In the first approximation, the dynamics of an economic system can be often well described by a linear model, in which the values $y_1(t), \ldots, y_n(t)$ of the desired quantities at the current moment of time linearly depend:

- on the values of these quantities at the previous moments of time, and
- on the values of related quantities $x_1(t), \ldots, x_m(t)$ at the current and previous moments of time:

Thongchai Dumrongpokaphan
Department of Mathematics, Faculty of Science, Chiang Mai University, Thailand
e-mail: tcd43@hotmail.com

Afshin Gholamy and Vladik Kreinovich
University of Texas at El Paso, El Paso, Texas 79968, USA
e-mail: afshingholamy@gmail.com, vladik@utep.edu

Hoang Phuong Nguyen
Division Informatics, Math-Informatics Faculty, Thang Long University, Nghiem Xuan Yem Road
Hoang Mai District, Hanoi, Vietnam, e-mail: nhphuong2008@gmail.com

$$y_i(t) = \sum_{j=1}^{n} \sum_{s=1}^{S} C_{ijs} \cdot y_j(t-s) + \sum_{p=1}^{m} \sum_{s=0}^{S} D_{ips} \cdot x_p(t-s) + y_{i0}. \qquad (1)$$

In practice, however, many real-life processes are non-linear. To get a more accurate description of real-life economic processes, it is therefore desirable to take this non-linearity into account.

Hammerstein-type block models for nonlinear dynamics are very efficient in econometrics. There are many different ways to describe non-linearity. In many econometric applications, the most accurate and the most efficient models turned out to be models which in control theory are known as *Hammerstein-type block models*, i.e., models that combine linear dynamic equations like (1) with non-linear static transformations; see, e.g., [5, 9, 10].

To be more precise, in such models, the transition from the state at one moment of time to the state at the next moment of time consists of several sequential transformations:

- some of which are linear dynamical transformations of the type (1), and
- some correspond to static non-linear transformations, i.e., nonlinear transformations that take into account only the current values of the corresponding quantities.

A toy example of a block model. To illustrate the idea of a Hammerstein-type block model, let us consider the simplest case, when:

- the state of the system is described by a single quantity y_1,
- the state $y_1(t)$ at the current moment of time is uniquely determined only by its previous state $y_1(t-1)$ (so there is no need to take into account earlier values like $y_1(t-2)$), and
- no other quantities affect the dynamics.

In the linear approximation, the dynamics of such a system is described by a linear dynamic equation

$$y_1(t) = C_{111} \cdot y_1(t-1) + y_{10}.$$

The simplest possible non-linearity here will be an additional term which is quadratic in $y_1(t)$:

$$y_1(t) = C_{111} \cdot y_1(t-1) + c \cdot (y_1(t-1))^2 + y_{10}.$$

The resulting non-linear system can be naturally reformulated in Hammerstein-type block terms if we introduce an auxiliary variable $s(t) \stackrel{\text{def}}{=} (y_1(t))^2$. In terms of this auxiliary variable, the above system can be described in terms of two blocks:

- a linear dynamical block described by a linear dynamic equation

$$y_1(t) = C_{111} \cdot y_1(t-1) + c \cdot s(t-1) + y_{10}, \text{ and}$$

- a nonlinear block described by the following non-linear static transformation

$$s(t) = (y(t))^2.$$

Comment. In this simple case, we use a quadratic non-linear transformation. In econometrics, other non-linear transformations are often used: e.g., logarithms and exponential functions that transform a multiplicative relation $z = x \cdot y$ between quantities into a linear relation between their logarithms: $\ln(z) = \ln(x) + \ln(y)$.

Formulation of the problem. The above example shows that in many cases, a non-linear dynamical system can indeed be represented in the Hammerstein-type block form, but the question remains why necessarily such models often work the best in econometrics – while there are many other techniques for describing non-linear dynamical systems (see, e.g., [1, 7]), such as:

- Wiener models, in which the values $y_i(t)$ are described as Taylor series in terms of $y_j(t-s)$ and $x_p(t-s)$,
- models that describe the dynamics of wavelet coefficients,
- models that formulate the non-linear dynamics in terms of fuzzy rules, etc.

What we do in this paper. In this paper, we provide an explanation of why such block models are indeed empirically efficient in econometrics, especially in financial econometrics.

2 Analysis of the Problem and the Resulting Explanation

Specifics of computations related to econometrics, especially to financial econometrics. In many economics-related problems, it is important not only to predict future values of the corresponding quantities, but also to predict them as fast as possible. This need for speed is easy to explain.

For example, an investor who is the first to finish computation of the future stock price will have an advantage of knowing in what direction this price will go. If his or her computations show that the price will go up, the investor will buy the stock at the current price, before everyone else realizes that this price will go up – and thus gain a lot. Similarly, if the investor's computations show that the price will go down, the investor will sell his/her stock at the current price and thus avoid losing money.

Similarly, an investor who is the first to predict the change in the ratio of two currencies will gain a lot. In all these cases, fast computations are extremely important. Thus, the nonlinear models that we use in these predictions must be appropriate for the fastest possible computations.

How can we speed up computations: need for parallel computations. If a task takes a lot of time for a single person, a natural way to speed it up is to have someone else help, so that several people can perform this task in parallel. Similarly, if a task

takes too much time on a single computer processor, a natural way to speed it up is to have several processors work in parallel on different parts of this general task.

Need to consider the simplest possible computational tasks for each processor. For a massively parallel computation, the overall computation time is determined by the time during which each processor finishes its task. Thus, to make the overall computations as fast as possible, it is necessary to make the elementary tasks assigned to each processor as fast – and thus, as simple – as possible.

Each computational task involves processing numbers. Since we are talking about the transition from linear to nonlinear models, it makes sense to consider linear versus nonlinear transformations. Clearly, linear transformations are much faster than nonlinear ones.

However, if we only use linear transformations, then we only get linear models. To take nonlinearity into account, we need to have some nonlinear transformations as well. A nonlinear transformation can mean:

- having one single input number and transforming it into another,
- it can mean having two input numbers and applying a nonlinear transformation to these two numbers,
- it can mean having three input numbers, etc.

Clearly, in general, the less numbers we process, the faster the data processing. Thus, to make computations as fast as possible, it is desirable to restrict ourselves to the fastest possible nonlinear transformations: namely, the transformations of one number into one number.

Thus, to make computations as fast as possible, it is desirable to make sure that on each computation stage, each processor performs one of the fastest possible transformations:

- either a linear transformation
- or the simplest possible nonlinear transformation $y = f(x)$.

Need to minimize the number of computational stages. Now that we agreed how to minimize the computation time needed to perform each computation stage, the overall computation time is determined by the number of computational stages. To minimize the overall computation time, we thus need to minimize the overall number of such computational stages.

In principle, we can have all kinds of nonlinearities in economic systems. Thus, we need to select the smallest number of computational stages that would still allow us to consider all possible nonlinearities. How many stages do we need?

One stage is not sufficient. One stage is clearly not enough. Indeed, during one single stage, we can compute:

- either a linear function $Y = c_0 + \sum_{i=1}^{N} c_i \cdot X_i$ of the inputs $X_1, \ldots, X_N$,
- or a nonlinear function of one of these inputs $Y = f(X_i)$,
- but not, e.g., a simple nonlinear function of two inputs, such as $Y = X_1 \cdot X_2$.

What about two stages? Can we use two stages?

- If both stages are linear, all we get is a composition of two linear functions which is also linear.
- Similarly, if both stages are nonlinear, all we get is compositions of functions of one variable – which is also a function of one variable.

Thus, we need to consider two different stages.

If:

- on the first stage we use nonlinear transformations $Y_i = f_i(X_i)$, and
- on the second stage, we use a linear transformation $Y = \sum_{i=1}^{N} c_i \cdot Y_i + c_0$,

we get the expression

$$Y = \sum_{i=1}^{N} c_i \cdot f_i(X_i) + c_0.$$

For this expression, the partial derivative

$$\frac{\partial Y}{\partial X_1} = c_1 \cdot f_1'(X_1)$$

does not depend on X_2 and thus,

$$\frac{\partial^2 Y}{\partial X_1 \partial X_2} = 0,$$

which means that we cannot use such a scheme to describe the product $Y = X_1 \cdot X_2$ for which

$$\frac{\partial^2 Y}{\partial X_1 \partial X_2} = 1.$$

But what if:

- we use linear transformation on the first stage, getting

$$Z = \sum_{i=1}^{N} c_i \cdot X_i + c_0,$$

and then

- we apply a nonlinear transformation $Y = f(Z)$.

This would result in

$$Y(X_1, X_2, \ldots) = f\left(\sum_{i=1}^{N} c_i \cdot X_i + c_0\right).$$

In this case, the level set $\{(X_1, X_2, \ldots) : Y(X_1, X_2, \ldots) = \text{const}\}$ of thus computed function is described by the equation

$$\sum_{i=1}^{N} c_i \cdot X_i = \text{const},$$

and is, thus, a plane. In particular, in the 2-D case when $N = 2$, this level set is a straight line. Thus, a 2-stage function cannot describe or approximate multiplication $Y = X_1 \cdot X_2$, because for multiplication, the level sets are hyperbolas $X_1 \cdot X_2 = \text{const}$ – and not straight lines.

So, two computational stages are not sufficient, we need at least three.

Are three computational stages sufficient? The positive answer to this equation comes from the fact that an arbitrary function can be represented as a Fourier transform and thus, can be approximated, with any given accuracy, as a linear combination of trigonometric functions:

$$Y(X_1,\ldots,X_N) \approx \sum_k c_k \cdot \sin\left(\omega_{k1} \cdot X_1 + \ldots + \omega_{kN} \cdot X_N + \omega_{k0}\right).$$

The right-hand side expression can be easily computed in three simple computational stages of one of the above types:

- first, we have a linear stage where we compute the linear combinations

$$Z_k = \omega_{k1} \cdot X_1 + \ldots + \omega_{kN} \cdot X_N + \omega_{k0},$$

- then, we have a nonlinear stage at which we compute the values $Y_k = \sin(Z_k)$, and
- finally, we have another linear stage at which we combine the values Y_k into a single value $Y = \sum_k c_k \cdot Y_k$.

Thus, three stages are indeed sufficient – and so, in our computations, we should use three stages, e.g., linear-nonlinear-linear as above.

Relation to traditional 3-layer neural networks. The same three computational stages form the basis of the traditional 3-layer neural networks (see, e.g., [2, 4, 6, 8]):

- on the first stage, we compute a linear combination of the inputs

$$Z_k = \sum_{i=1}^{N} w_{ki} \cdot X_i - w_{k0};$$

- then, we apply a nonlinear transformation $Y_k = s_0(Z_k)$; the corresponding *activation function* $s_0(z)$ usually has either the form $s_0(z) = \dfrac{1}{1+\exp(-z)}$ or the rectified linear form $s_0(z) = \max(z, 0)$ [3, 6];
- finally, a linear combination of the values Y_k is computed: $Y = \sum_{k=1}^{K} W_k \cdot Y_k - W_0$.

Comments.

- It should be mentioned that in neural networks, the first two stages are usually merged into a single stage in which we compute the values

$$Y_k = s_0\left(\sum_{i=1}^{N} w_{ki}\cdot X_i - w_{k0}\right).$$

  The reason for this merger is that in the biological neural networks, these two stages are performed within the same neuron:

  – first, the signals X_i from different neurons come together, forming a linear combination $Z_k = \sum\limits_{i=1}^{N} w_{ki}\cdot X_i - w_{k0}$, and
  – then, within the same neuron, the nonlinear transformation $Y_k = s_0(Z_k)$ is applied.

- Instead of using the same activation function $s_0(z)$ for all the neurons, it is sometimes beneficial to use different functions in different situations, i.e., take $Y_k = s_k(Z_k)$ for several different functions $s_k(z)$; see, e.g., [6] and references therein.

How all this applies to non-linear dynamics. In non-linear dynamics, as we have mentioned earlier, to predict each of the desired quantities $y_i(t)$, we need to take into account the previous values $y_j(t-s)$ of the quantities $y_1,\ldots,y_n$, and the current and previous values $x_p(t-s)$ of the related quantities $x_1,\ldots,x_m$. In line with the above-described 3-stage computation scheme, the corresponding prediction of each value $y_i(t)$ consists of the following three stages:

- first, there is a linear stage, at which we form appropriate linear combinations of all the inputs; we will denote the values of these linear combinations by $\ell_{ik}(t)$:

$$\ell_{ik}(t) = \sum_{j=1}^{n}\sum_{s=1}^{S} w_{ikjs}\cdot y_j(t-s) + \sum_{p=1}^{m}\sum_{s=0}^{S} v_{ikps}\cdot x_p(t-s) - w_{ik0}; \tag{2}$$

- then, there is a non-linear stage when we apply the appropriate nonlinear functions $s_{ik}(z)$ to the values ℓ_{ik}; the results of this application will be denoted by $a_{ik}(t)$:

$$a_{ik}(t) = s_{ik}(\ell_{ik}(t)); \tag{3}$$

- finally, we again apply a linear stage, at which we estimate $y_i(t)$ as a linear combination of the values $a_{ik}(t)$ computed on the second stage:

$$y_i(t) = \sum_{k=1}^{K} W_{ik}\cdot a_{ik}(t) - W_{i0}. \tag{4}$$

We thus have the desired Hammerstein-type block structure:

- a linear dynamical part (2) is combined with

- static transformations (3) and (4), in which we only process values corresponding to the same moment of time t.

Thus, *the desire to perform computations as fast as possible indeed leads to the Hammerstein-type block models.* We have therefore explained the efficiency of such models in econometrics.

Comment. Since, as we have mentioned, 3-layer models of the above type are universal approximators, we can conclude that:

- not only Hammesterin-type models compute as fast as possible,
- these models also allow us to approximate any possible nonlinear dynamics with as much accuracy as we want.

Acknowledgments

This work was supported by Chiang Mai University. It was also partially supported by the US National Science Foundation via grant HRD-1242122 (Cyber-ShARE Center of Excellence).

The authors are greatly thankful to Hung T. Nguyen for valuable discussions.

References

1. S. A. Billings, *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*, Wiley, Chichester, UK, 2013.
2. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.
3. O. Fuentes, J. Parra, E. Anthony, and V. Kreinovich, "Why Rectified Linear Neurons Are Efficient: A Possible Theoretical Explanations", In: O. Kosheleva, S. Shary, G. Xiang, and R. Zapatrin (eds.), *Beyond Traditional Probabilistic Data Processing Techniques: Interval, Fuzzy, etc. Methods and Their Applications*, Springer, Cham, Switzerland, to appear.
4. A. Gholamy, J. Parra, V. Kreinovich, O. Fuentes, and E. Anthony, "How to Best Apply Deep Neural Networks in Geosciences: Towards Optimal 'Averaging' in Dropout Training", In: J. Watada, S. C. Tan, P. Vasant, E. Padmanabhan, and L. C. Jain (eds.), *Smart Unconventional Modelling, Simulation and Optimization for Geosciences and Petroleum Engineering*, Springer Verlag, to appear.
5. F. Giri and E.-W. Bai (eds.), *Block-oriented Nonlinear System Identification*, Springer Lecture Notes in Control and Information Sciences, Vol. 404, Springer Verlag, Berlin, Heidelberg, 2010.
6. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Leaning*, MIT Press, Cambridge, Massachusetts, 2016.
7. O. Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*, Springer Verlag, Berlin, Heidelberg, 2010.
8. H. T. Nguyen and V. Kreinovich, *Applications of Continuous Mathematics to Computer Science*, Kluwer, Dordrecht, Netherlands, 1997.
9. S. Strmcnik and D. Juricic (eds.), *Case Studies in Control: Putting Theory to Work*, Springer Verlag, London, 2013.
10. W. van Drongelen, *Signal Processing for Neuroscientists*, London, UK, 2018.