Neural-Network Approximation of Reduced Forms for CGE Models Explained by Elementary Examples

CoPS Working Paper No. G-348, November 2024

Peter B. Dixon,
Maureen T. Rimmer
And
Florian Schiffmann
Centre of Policy Studies,
Victoria University

ISSN 1 921654 02 3

ISBN 978-1-921654-57-2

About us

Researchers at the Centre of Policy Studies have a 45-year history of continuous achievement in the development, application and dissemination of large-scale economic models. Our models and software are used around the world to analyse a diverse range of economic issues. CoPS' funders include: Australian federal and state government departments; private firms and universities in many parts of the world; central government agencies such as finance and trade ministries in many countries; and international development organisations. The Centre's GEMPACK software, used for solving large economic models, is used at more than 700 sites in over 95 countries.

Citation

Dixon, Peter B., Maureen T. Rimmer and Florian Schiffmann (2024), "Neural-Network approximation of reduced forms for CGE models explained by elementary examples" Centre of Policy Studies Working Paper No. G-348, Victoria University, November 2024.

Neural-Network approximation of reduced forms for CGE models explained by elementary examples

by

Peter Dixon, Maureen Rimmer, Florian Schiffmann

Centre of Policy Studies, Victoria University, Melbourne

November 12, 2024

Abstract:

Neural Network (NN) theory provides a powerful method for approximating the reduced form of a large-scale multi-regional CGE model. However, NN methods are relatively unknown by CGE modellers. We set out the theory of the NN approximation method and demonstrate how it works with simple examples.

The paper is motivated by a project for a client with limited in-house CGE capabilities but requiring the ability to obtain CGE solutions at short notice in a confidential environment. We describe how an NN approximation meets the client's needs. The NN approximation is more accurate and broadly applicable than earlier approaches that CGE modellers have used based on regression equations and matrices of elasticities.

JEL codes

C45, C68

Key words:

Neural network method explained; Neural network approximations to reduced forms; Multi-regional computable general equilibrium models

Contents

Neural-Network approximation of reduced forms for CGE models explained by elementary examples

by

Peter Dixon, Maureen Rimmer, Florian Schiffmann

Centre of Policy Studies, Victoria University, Melbourne

November 12, 2024

1. Introduction

Florian Schiffmann suggested using a Neural Network (NN) approach to approximate the reduced form of a CGE model. He then implemented it in the application motivating this paper. Peter Dixon and Maureen Rimmer, with advice from Florian, drafted the paper. Their aim was to teach themselves about what Florian was doing and, perhaps in the process, help to demystify NN ideas in the minds of other CGE modellers.[1]

Section 2 explains the role of reduced forms in transferring CGE capabilities to the clients of CGE modeling groups. As an example, we introduce the destructive events tool (DET).

Section 3 sets out the formal mathematics of the NN approach. In section 4 we give the mathematics intuitive substance by working through elementary examples. In the examples, we specify highly simplified reduced forms and show how these can be approximated by an NN search.

Section 5 is a brief report on our experience in using NN to construct DET. The construction of DET is part of a larger project. Further information on NN will be included in a report on the larger project which we are preparing with other colleagues at the Centre of Policy Studies.

Section 6 contains concluding remarks.

2. Motivation

Computable general equilibrium (CGE) modelling offers insights on policies for the environment, agriculture, trade, and other areas. However, policy-relevant CGE models are complex. Successful application of CGE models requires people with considerable training and experience. Organizations without CGE expertise can go to outside providers, but this is an unattractive option for quick-turnaround projects, especially if the results must be confidential.

Rose *et al.* (2017) and Dixon *et al.* (2019) developed mimic tools that clients use to approximate CGE solutions relevant to their interests. These tools were formed by conducting a limited number of CGE simulations. From the simulations, they were able to fit regression equations or estimate elasticities that encapsulate CGE relationships between endogenous and exogenous variables. These relationships became the basis of mimic tools. The attractive feature of these tools is that they can be applied easily in-house with negligible

[1] The only CGE application of NN in CGE modelling of which we are aware is Britz *et al*. (2021) who used NN for sensitivity analysis. In the Britz *et al.* paper, readers are assumed to know how NN works.

turn-around time by clients without CGE experience. On the other hand, the number of exogenous variables included in these mimic tools and the range of applicable shocks is limited, and non-linearities present in the CGE model are generally missed.

This paper describes an approach based on Neural Network (NN) theory for creating mimic tools. This approach overcomes the limitations of the earlier regression/elasticity methods.

To motivate the explanation of the NN approach we briefly describe a destructive events tool (DET). This was built for a client who wanted to equip itself with the ability to assess at short notice, in a secure environment, the likely regional and national economic effects of an adverse event with any given destruction/death/evacuation characteristics occurring at any location.

The underlying economic model for DET is a version of TERM[2]. TERM is a dynamic, CGE modelling system. It has been implemented for single countries disaggregated into sub-national regions and for multi-country models with subnational regions. In the application motivating this paper, TERM was formulated for a single national economy divided into 29 sub-national regions and 23 industries. The model can receive shocks representing capital destruction in each industry and sub-national region, and deaths in and evacuations from each sub-national region. The output from the model includes national and sub-national economic variables such as GDP and output by industry. In reduced form the model can be visualized as:

$$\text{gdp}(q) = G_q\left(k(r,j); D(r); E(r) \text{ for all } r \in \text{REG},\ j \in \text{IND}\right) \quad q \in \{\text{nation, 29 regions}\} \tag{2.1}$$

$$\text{output}(q,i) = H_{q,i}\left(k(r,j); D(r); E(r) \text{ for all } r \in \text{REG},\ j \in \text{IND}\right) \quad \text{for } q \in \{\text{nation, 29 regions}\},\ \text{and } i \in \{\text{23 industries}\} \tag{2.2}$$

where the exogenous variables are

- k(r,j), the percentage of the capital stock in industry j in region r that is destroyed or made unusable by the adverse event;
- D(r), the number of deaths in region r; and
- E(r), the number of evacuations from region r;

and the endogenous variables are

- gdp(q), the percentage change in GDP in region q (national and sub-national) caused by the adverse event; and
- output(q,i), the percentage change in the output of industry i in region q (national and sub-national).

For any given values of the exogenous variables, we can obtain values for the endogenous variables by solving the model. But that option is not available to our client. What the client needs are functional forms for G_q and $H_{q,i}$. Then by inserting into (2.1) and (2.2) values for the exogenous variables suitable for the event under consideration, the client can evaluate the endogenous variables.

[2] **T**he **E**normous **R**egional **M**odel. The TERM methodology was pioneered by Horridge *et al* (2005). In recent years, it has been developed and extended by Wittwer and other colleagues at the Centre of Policy Studies, see for example Wittwer (2017 and 2024) and Wittwer and Horridge (2018).

Thus, the challenge for us was to derive explicit functional forms for G_q and $H_{q,i}$ that could be used by the client.

3. Using Neural Networks to derive explicit forms for G_q and $H_{q,i}$ in (2.1) and (2.2)

In developing explicit forms for G_q and $H_{q,i}$ we ran about 150,000 TERM simulations with different sets of values for the exogenous variables. This gave us a dataset with 150,000 observations for the vectors of exogenous variables and the corresponding values for the endogenous variables. Then we derived G_q and $H_{q,i}$ by fitting Neural Network (NN) equations to 135,000 observations and used the remaining 15,000 observations for out-of-sample testing.

We start the explanation of how the NN method uses the data by specifying in (3.1) to (3.5) the mathematical form of the NN optimization problem. In explaining the notation, we introduce various NN concepts: layer, input layer, output layer, hidden layer, node, weights, bias, edge and activation function. Before reviewing the mathematics in detail, it is useful to note that:

(a) N1 in (3.2) is the number of exogenous variables;
(b) NJ in (3.5) is the number of endogenous variables;
(c) $V_r^{Exog}(t)$ for t = 1, …, 135,000 is the data for the r^{th} exogenous variable, and $V_k^{Endo}(t)$ is the data for k^{th} endogenous variable;
(d) $V_{Jk}(t)$, for k = 1, …, NJ, is the fitted value for k^{th} the endogenous variable when the exogenous variables are from observation t; and
(e) the Fs are pre-specified functions.

The NN problem takes the form:

choose values for Ws to minimize

$$\sum_{k=1}^{NJ} \sum_{t \in OBS} \left(V_k^{Endo}(t) - V_{Jk}(t) \right)^2 \tag{3.1}$$

subject to

$$V_{1r}(t) = V_r^{Exog}(t), \quad \text{for all t and } r = 1, 2, \ldots, N1, \tag{3.2}$$

$$V_{2r}(t) = F_{2r}\left(\sum_{\ell=1}^{N1} W_{1\ell,2r} * V_{1\ell}(t) + W_{b,2r} \right), \quad \text{for all t and } r = 1, 2, \ldots, N2 \tag{3.3}$$

… … …

$$V_{(J-1)r}(t) = F_{(J-1)r}\left(\sum_{\ell=1}^{N(J-2)} W_{(J-2)\ell,(J-1)r} * V_{(J-2)\ell}(t) + W_{b,(J-1)r} \right), \quad \text{for all t and } r = 1, 2, \ldots, N(J-1) \tag{3.4}$$

and

$$V_{Jk}(t) = \sum_{\ell=1}^{N(J-1)} W_{(J-1)\ell,Jk} * V_{(J-1)\ell}(t) + W_{b,Jk} \quad \text{for all t and } k = 1, , 2, \ldots NJ \tag{3.5}$$

With given values for the Ws (often referred to as *weights* although they can positive or negative and need not sum to one), we could work through (3.2) to (3.5) to obtain $V_{Jk}(t)$. The objective is to set the Ws so that the resulting fitted values for $V_{Jk}(t)$, k = 1, …, NJ minimize (3.1).

The constraints in the optimization problem are set out in what is referred to in NN as *layers*, each of which contains *nodes*. The node values in one layer are transmitted to the next layer by transmission lines known as *edges*.

The first layer, known as the *input layer*, is specified by (3.2). This layer has a node for each exogenous variable. The values at these nodes [$V_{1r}(t)$] are supplied by the observations for the exogenous variables.

The final layer (layer J), known as the *output layer*, is specified by (3.5). This layer has a node for each endogenous variable. The value at the k^{th} node [$V_{Jk}(t)$] is the fitted value for endogenous variable k and is formed as a linear combination of the node values in the second-last layer (layer J-1) plus an intercept term. The weights used in transferring the value in layer J-1 at node ℓ to layer J at node k is denoted by $W_{(J-1)\ell,Jk}$. The intercept term (often referred to as *bias*) introduced in layer J at node k is denoted by $W_{b,Jk}$.

If there are no layers between the input layer and the output layer, the NN problem reduces to ordinary least squares.

The layers between the input and output layers are known as *hidden layers*. These layers can have any number of nodes. The value at a node [$V_{mr}(t)$, m = 2, .., J-1 and r = 1, …Nm] in a hidden layer is formed by applying an *activation function*, the Fs in (3.3) – (3.4). The input to an F function is a linear combination of node values in the previous layer *plus* an intercept term. The weights used in transferring the value in layer m-1, node ℓ to layer m, node r is denoted by $W_{(m-1)\ell,mr}$. The intercept term in layer m at node r is denoted by $W_{b,mr}$.

Figure 1 illustrates a 3-layer NN computation in which there are two exogenous variables and one endogenous variable, and the second layer has two nodes.

While (3.1) – (3.5) provides a formal definition of the process by which the reduced-form equations in (2.1) – (2.2) can be approximated by NN fitting, it leaves important questions unanswered. How do we determine the number of hidden layers and the number of nodes in each of them? What form should the activation functions take? Finally, does the method give a close approximation to the G_q and $H_{q,i}$ functions?

The answer to the last question is yes, but the answers to the other questions are indefinite. All we can do is report our experience in section 5 and show that the method worked. However, we can say that the activation functions chosen in NN analyses are typically simple. For example, in our application, we used the ReLU form[3] in generating the node values in all hidden layers, that is we evaluated the nodes in the hidden layers according to:

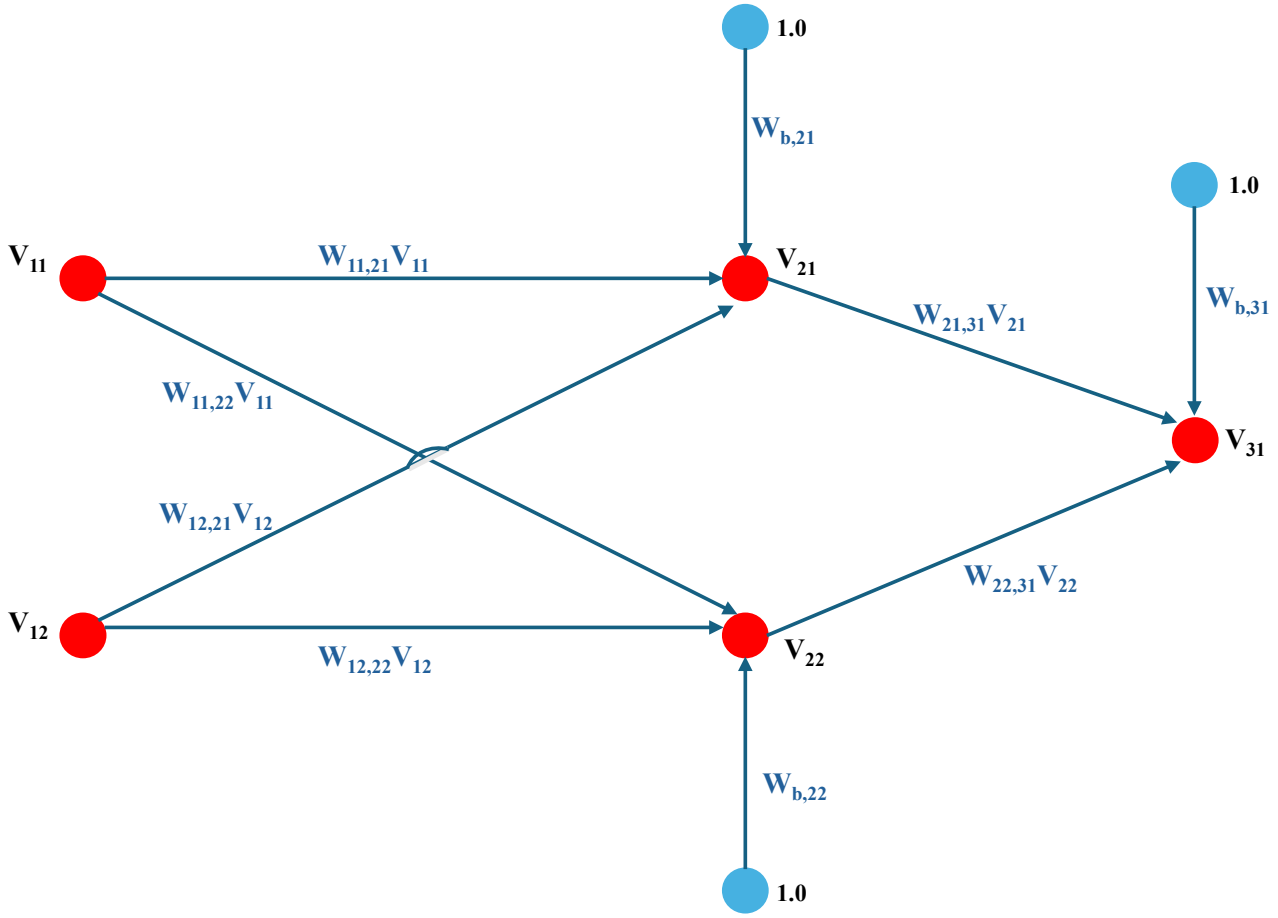
$$V_{mr}(t) = \text{Max}\left(0, \sum_{\ell=1}^{N(m-1)} W_{(m-1)\ell,mr} * V_{(m-1)\ell}(t) + W_{b,mr}\right),$$

$$\text{for all t, m= 2, …, J-1, and r= 1, 2,…, Nm} \qquad (3.6)$$

In the next section, we illustrate the surprising flexibility of (3.6) in elementary examples of NN computations.

[3] This is the rectified linear unit (ReLU) function.

Figure 1. An NN approximation of the value of the endogenous variable in a 3-layer computation with 2 exogenous variables and 1 endogenous variable



Input layer *Hidden layer* *Output layer*

4. NN computations in elementary examples

We present three examples in which there are two exogenous variables and just one endogenous variable. Then we discuss the NN method for a case in which there is more than one endogenous variable.

Example 1: a linear model

What happens if the true reduced form for our model is linear, and not knowing this, we use an NN computation to approximate the reduced form? Will the hidden layers get in the way of finding the true linear reduced form?

To answer this question, we assume that the endogenous variable is generated by a linear equation with two exogenous variables:

$$V^{Endo}(t) = V_1^{Exog}(t) + V_2^{Exog}(t), \quad \text{for all t}, \tag{4.1}$$

How well can we approximate (4.1) with a 3-layer NN search in which the activation functions are ReLU and second layer has two nodes? [The first layer has two nodes (number of exogenous variables and the third layer has 1 node (the fitted value of the single endogenous variable)].

With this setup we look for W values to minimize

$$\sum_{t\in OBS} \left(V^{Endo}(t) - V_{31}(t)\right)^2 \tag{4.2}$$

subject to

$$V_{1r}(t) = V_r^{Exog}(t) \quad \text{for all t and r= 1, 2}, \tag{4.3}$$

$$V_{2r}(t) = Max\left(0, \sum_{\ell=1}^{2} W_{1\ell,2r} * V_{1\ell}(t) + W_{b,2r}\right) \quad \text{for all t and r= 1, 2} \tag{4.4}$$

and

$$V_{31}(t) = \sum_{\ell=1}^{2} W_{2\ell,31} * V_{2\ell}(t) + W_{b,31} \quad \text{for all t} \tag{4.5}$$

After a little experimenting, we found that NN could exactly reproduce the linear model with the following weights:

$W_{11,21} = W_{12,21} = 1$ (weights for transmitting node values from layer 1 to 1st node in layer 2)

$W_{11,22} = W_{12,22} = -1$ (weights for transmitting node values from layer 1 to 2nd node in layer 2)

$W_{b,2r} = 0$ for r = 1, 2 (intercept terms for the nodes in layer 2)

$W_{21,31} = 1$, $W_{22,31} = -1$ (weights for transmitting node values from 2 to the single node at 3)

$W_{b,31} = 0$ (intercept term for the node in layer 3)

(4.6)

With these weights, the node values in layer 2 are given by

$$V_{21}(t) = Max\left(0, V_{11}(t) + V_{12}(t)\right) = \begin{cases} V_{11}(t) + V_{12}(t) & \text{if } V_{11}(t) + V_{12}(t) \geq 0 \\ 0 \text{ otherwise} \end{cases} \tag{4.7}$$

and

$$V_{22}(t) = \text{Max}\left(0, -V_{11}(t) - V_{12}(t)\right) = \begin{cases} -V_{11}(t) - V_{12}(t) & \text{if } V_{11}(t) + V_{12}(t) < 0 \\ 0 \text{ otherwise} \end{cases} \tag{4.8}$$

At layer 3, we have

$$V_{31} = V_{21} - V_{22}\,. \tag{4.9}$$

If $V_{11}(t) + V_{12}(t) \geq 0$ then substituting from (4.7) and (4.8) into (4.9) gives

$$V_{31}(t) = V_{11}(t) + V_{12}(t) \tag{4.10}$$

If $V_{11}(t) + V_{12}(t) < 0$ then substituting from (4.7) and (4.8) into (4.9) again gives

$$V_{31}(t) = V_{11}(t) + V_{12}(t) \tag{4.11}$$

Thus, under all circumstances, we see from (4.1) and (4.3) that

$$V_{31}(t) = V_1^{\text{Exog}}(t) + V_2^{\text{Exog}}(t) = V^{\text{Endo}}(t) \tag{4.12}$$

With the choice of weights in (4.6), the objective function (4.2) is optimized with value zero. This confirms that if the reduced form is linear then NN can exactly replicate it.

Example 2: a quadratic model

In presenting this example, we simplify the notation by omitting the "t" arguments.

We assume that the endogenous variable is generated by a quadratic equation with two exogenous variables:

$$V^{\text{Endo}} = \left(V_1^{\text{Exog}} + V_2^{\text{Exog}}\right)^2 \quad , \tag{4.13}$$

where V_1^{Exog} and V_2^{Exog} are drawn randomly and independently from rectangular distributions each with range [-0.5, 0.5]. The dots in Figure 2 show values of $V_1^{\text{Exog}} + V_2^{\text{Exog}}$ with the corresponding value of V^{Endo} for 100 observations of $(V_1^{\text{Exog}}, V_2^{\text{Exog}})$.

How well can we approximate (4.13) with a 3-layer NN computation in which the second layer has 4 nodes and the activation functions are ReLU?

We won't answer this question directly. But we will show that such an NN computation does at least as well as the fit obtained by the 4-piece linear approximation indicated by 4 straight lines in Figure 2 and defined by:

$$V^{\text{Endo}} = \begin{cases} 1.5*(V_1^{\text{Exog}} + V_2^{\text{Exog}}) - 0.5 & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} \geq 0.5 \\ 0.5*\left(V_1^{\text{Exog}} + V_2^{\text{Exog}}\right) & \text{if } 0 \leq V_1^{\text{Exog}} + V_2^{\text{Exog}} \leq 0.5 \\ -0.5*\left(V_1^{\text{Exog}} + V_2^{\text{Exog}}\right) & \text{if } 0 \geq V_1^{\text{Exog}} + V_2^{\text{Exog}} \geq -0.5 \\ -1.5*\left(V_1^{\text{Exog}} + V_2^{\text{Exog}}\right) - 0.5 & \text{if } -0.5 \geq V_1^{\text{Exog}} + V_2^{\text{Exog}} \end{cases} \tag{4.14}$$

We will do this by showing that the 4-piece linear approximation is delivered by the following weights applied to generate node values in the NN layers:

$W_{11,21} = 0.5$; $W_{12,21} = 0.5$; $W_{b,21} = 0$ [weights to generate value at layer 2, node 1]

$W_{11,22} = 1$; $W_{12,22} = 1$; $W_{b,22} = -0.5$ [weights to generate value at layer 2, node 2]

$$W_{11,23} = -0.5;\ W_{12,23} = -0.5;\ W_{b,23} = 0 \quad \text{[weights to generate value at layer 2, node 3]}$$
$$W_{11,24} = -1;\ W_{12,24} = -1;\ W_{b,24} = -0.5 \quad \text{[weights to generate value at layer 2, node 4]}$$
$$W_{21,31} = 1;\ W_{22,31} = 1;\ W_{23,31} = 1;\ W_{24,31} = 1;\ W_{b,31} = 0 \quad \text{[weights at layer 3, node 1]} \tag{4.15}$$

Demonstration that the weights specified in (4.15) deliver (4.14)

Working with (3.2) – (3.3) and using the weights in (4.15), we obtain node values in layer 2:

$$V_{21} = \text{Max}\left[0, 0.5*V_{11} + 0.5*V_{12}\right] = \begin{cases} 0.5*(V_{11}+V_{12}) & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} \geq 0 \\ 0 & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} < 0 \end{cases} \tag{4.16}$$

$$V_{22} = \text{Max}\left[0, V_{11} + V_{12} - 0.5\right] = \begin{cases} V_{11}+V_{12}-0.5 & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} \geq 0.5 \\ 0 & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} < 0.5 \end{cases} \tag{4.17}$$

$$V_{23} = \text{Max}\left[0, -0.5*V_{11} - 0.5*V_{12}\right] = \begin{cases} -0.5*(V_{11}+V_{12}) & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} \leq 0 \\ 0 & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} > 0 \end{cases} \tag{4.18}$$

$$V_{24} = \text{Max}\left[0, -V_{11} - V_{12} - 0.5\right] = \begin{cases} -(V_{11}+V_{12})-0.5 & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} \leq -0.5 \\ 0 & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} > -0.5 \end{cases} \tag{4.19}$$

Using the layer-3 weights from (4.15) in (3.5), we obtain the fitted value, V_{31}, for the endogenous variable as:

$$V_{31} = V_{21} + V_{22} + V_{23} + V_{24} \tag{4.20}$$

Consistent with (4.14) we find that the fitted value, V_{31}, of V^{Endo} is determined as follows:

if $V_1^{\text{Exog}} + V_2^{\text{Exog}} \geq 0.5$ then,

$$V_{31} = 0.5*(V_{11}+V_{12}) + (V_{11}+V_{12}) - 0.5 = 1.5*(V_{11}+V_{12}) - 0.5 \tag{4.21}$$

if $0 \leq V_1^{\text{Exog}} + V_2^{\text{Exog}} \leq 0.5$ then,

$$V_{31} = 0.5*(V_{11}+V_{12}) \tag{4.22}$$

if $0 \geq V_1^{\text{Exog}} + V_2^{\text{Exog}} \geq -0.5$ then,

$$V_{31} = -0.5*(V_{11}+V_{12}) \tag{4.23}$$

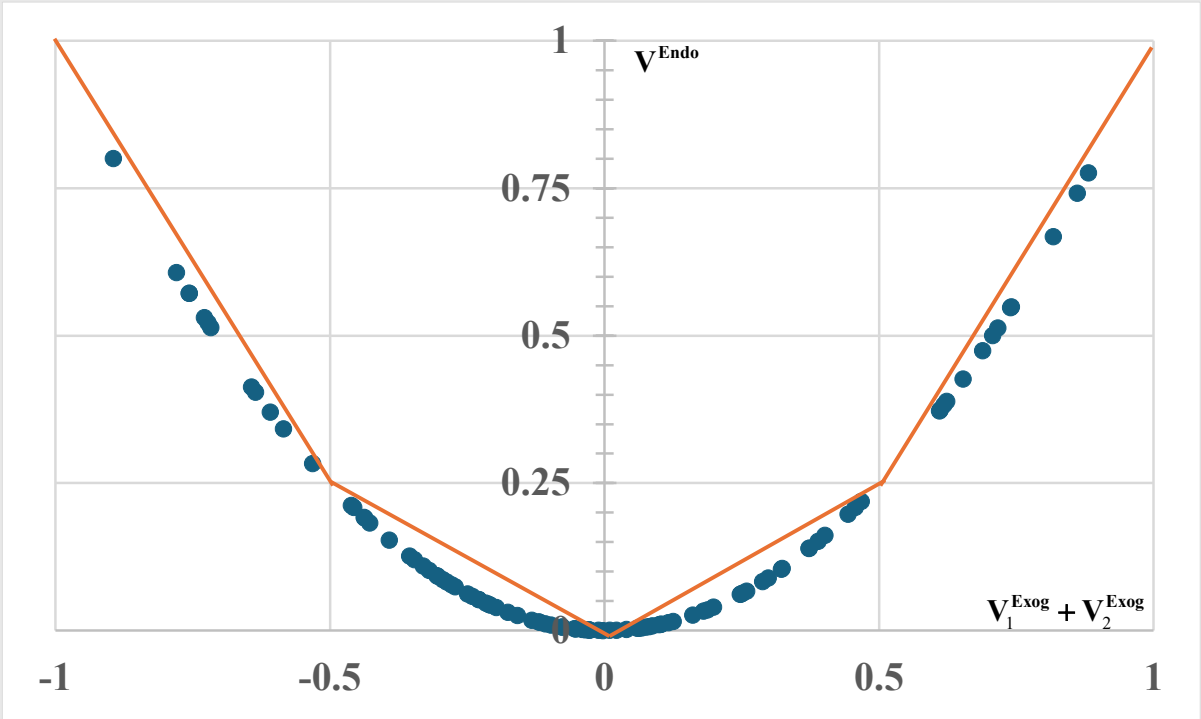
and if $-0.5 \geq V_1^{\text{Exog}} + V_2^{\text{Exog}}$ then,

$$V_{31} = -0.5*(V_{11}+V_{12}) - (V_{11}+V_{12}) - 0.5 \quad = -1.5*(V_{11}+V_{12}) - 0.5 \tag{4.24}$$

Interpretation

The forecasting R-squared for the piecewise approximation defined by (4.14) and illustrated in Figure 2 is 0.95. This has been achieved without optimizing. We can conclude that an NN computation with ReLU activation functions and one hidden layer containing 4 nodes would do even better than this. It is also apparent that with more nodes in the hidden layer, the approximation function would have more pieces and give any desired level of accuracy.

Figure 2. Quadratic model approximated by 4-segment piecewise linear function



We chose the illustrative model (4.13) because it could be represented easily in a 2-dimensional diagram. It is clear that NN piecewise approximations could be used to handle more general polynomial forms.

Example 3: A Leontief function

In this example, we assume that the endogenous variable is generated by

$$V^{Endo} = \text{Max}\left(V_1^{Exog}, V_2^{Exog}\right) \quad , \tag{4.25}$$

A reduced form along the lines of (4.25) might arise if V_1^{Exog} and V_2^{Exog} refer to production facilities for a key product and V^{Endo} is GDP. The effect on GDP might be severe if there are large negative shocks to both production facilities but relatively mild if just one facility is destroyed, allowing users of the critical commodity to be supplied from the other facility.

We show that (4.25) can be reproduced exactly with a 3-layer NN computation in which the second layer has 4 nodes and the activation functions are ReLU.

Consider the following settings for the weights:

$W_{11,21} = 0.5;\ W_{12,21} = 0.5; W_{b,21} = 0$ [weights to generate value at layer 2, node 1]

$W_{11,22} = -0.5;\ W_{12,22} = -0.5; W_{b,22} = 0$ [weights to generate value at layer 2, node 2]

$W_{11,23} = 0.5;\ W_{12,23} = -0.5; W_{b,23} = 0$ [weights to generate value at layer 2, node 3]

$W_{11,24} = -0.5;\ W_{12,24} = 0.5; W_{b,24} = 0$ [weights to generate value at layer 2, node 4]

$W_{21,31} = 1;\ W_{22,31} = -1; W_{23,31} = 1;\ W_{24,31} = 1; W_{b,31} = 0$ [weights at layer 3, node 1]

(4.26)

Demonstration that the weights specified in (4.26) deliver (4.25)

Working with (3.2) – (3.3) and using the weights in (4.26), we obtain node values in layer 2:

$$V_{21} = \text{Max}\left[0, 0.5 * V_{11} + 0.5 * V_{12}\right] = \begin{cases} 0.5*(V_{11}+V_{12}) & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} \geq 0 \\ 0 & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} < 0 \end{cases} \tag{4.27}$$

$$V_{22} = \text{Max}\left[0, -0.5 * V_{11} - 0.5 * V_{12}\right] = \begin{cases} -0.5*(V_{11}+V_{12}) & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} \leq 0 \\ 0 & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} > 0 \end{cases} \tag{4.28}$$

$$V_{23} = \text{Max}\left[0, 0.5 * V_{11} - 0.5 * V_{12}\right] = \begin{cases} 0.5*(V_{11}-V_{12}) & \text{if } V_1^{\text{Exog}} - V_2^{\text{Exog}} \geq 0 \\ 0 & \text{if } V_1^{\text{Exog}} + V_2^{\text{Exog}} < 0 \end{cases} \tag{4.29}$$

$$V_{24} = \text{Max}\left[0, -0.5 * V_{11} + 0.5 * V_{12}\right] = \begin{cases} -0.5*(V_{11}-V_{12}) & \text{if } V_1^{\text{Exog}} - V_2^{\text{Exog}} \leq 0 \\ 0 & \text{if } V_1^{\text{Exog}} - V_2^{\text{Exog}} > 0 \end{cases} \tag{4.30}$$

Using the layer-3 weights from (4.26) in (3.5), we obtain the fitted value, V_{31}, for the endogenous variable as:

$$V_{31} = V_{21} - V_{22} + V_{23} + V_{24} \tag{4.31}$$

From (4.27) and (4.28) we see that under all circumstances,

$$V_{21} - V_{22} = 0.5*(V_{11} + V_{12}) \tag{4.32}$$

From (4.29) and (4.30) we see that under all circumstances,

$$V_{23} + V_{24} = 0.5*\left|V_{11} - V_{12}\right| \tag{4.33}$$

Then using (4.31), (4.32) and (4.33) we obtain

$$V_{31} = \text{Max}\left(V_{11}, V_{12}\right) \tag{4.34}$$

This confirms that the Leontief function can be reproduced by an NN-computation with ReLU activation functions.

Example 4. Dealing with more than one endogenous variable

If there is more than one endogenous variable, then we could perform a separate NN computation for each one. However, by solving a single NN problem encompassing multiple endogenous variables, as in (3.1) to (3.5), we can economize on the dimension of the NN optimization problem.

To gain intuition on how this works, we combine examples 1 and 2. We assume that there are two endogenous variables generated by 2 exogenous variables according to:

$$V_1^{\text{Endo}} = \left(V_1^{\text{Exog}} + V_2^{\text{Exog}}\right)^2 \tag{4.35}$$

$$V_2^{\text{Endo}} = V_1^{\text{Exog}} + V_2^{\text{Exog}} \tag{4.36}$$

How well can we approximate (4.35) and (4.36) with a 3-layer NN computation in which the second layer has 4 nodes and the activation functions are ReLU? As in example 2, we don't answer this question directly. We specify weights and show that they generate good

approximations to the true reduced-form equations, thereby demonstrating that an NN optimization procedure would generate good approximations.

The weights we choose to determine the values of the nodes in the second layer and the value of the first node in the third layer are those given in example 2 by (4.15). The weights we choose to determine the value of the second node in the third are

$$W_{21,32} = 2;\ W_{22,32} = 0;\ W_{23,32} = 2;\ W_{24,32} = 0;\ W_{b,32} = 0 \qquad (4.37)$$

With this choice of weights, the values at the nodes in the second layer are given by (4.27) to (4.30), and the value at the first node of the third layer is given by (4.21) to (4.24). Consequently, with our chosen weights the NN approximation for V_1^{Endo} is the same as the 4-piece linear approximation illustrated in Figure 2 for example 2. What about the approximation for V_2^{Endo}? By working through Figure 3, we can see that the NN fit for V_2^{Endo} is exact.

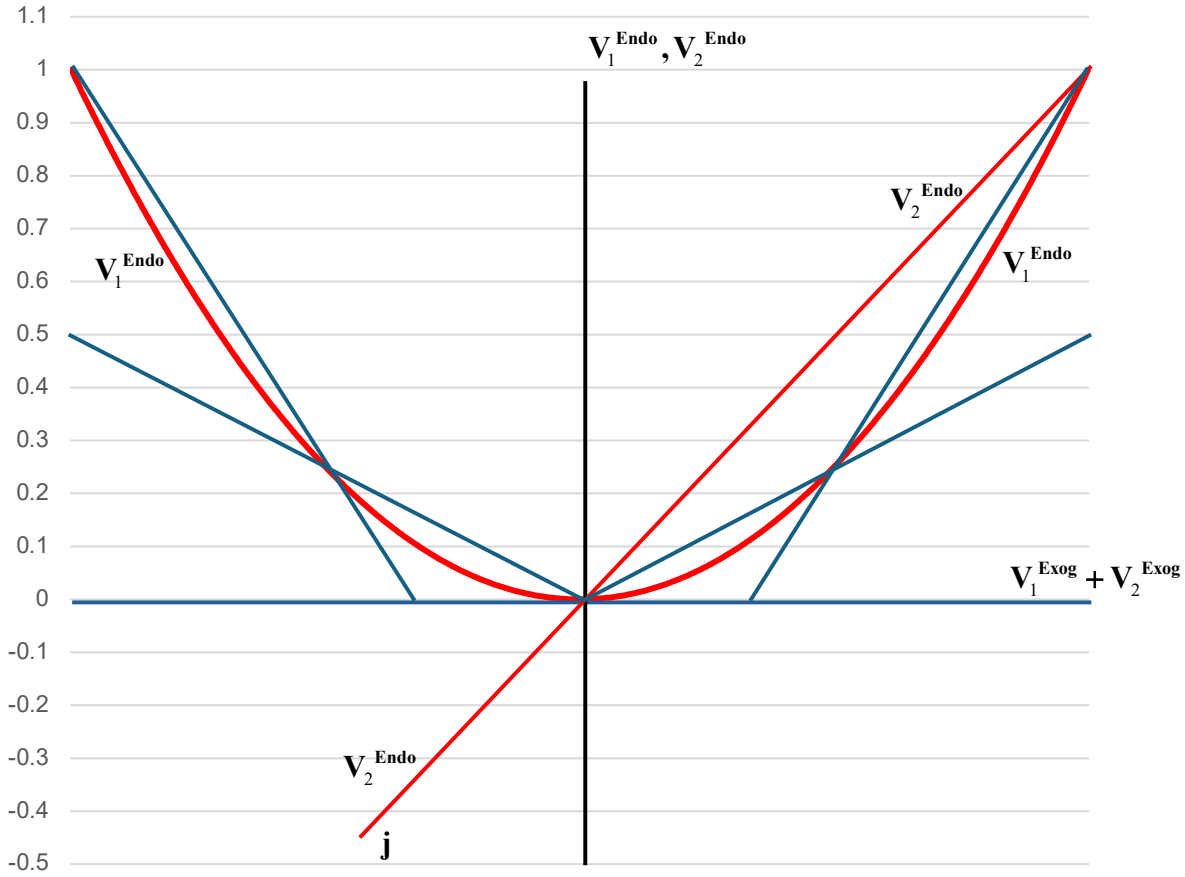
In this example we have lost no accuracy by combining the NN computations for the 2 endogenous variables into a single computation. At the same time, we have reduced the number of weights that need to be determined. In two separate computations we have a total of 34 weights: 17 for each computation made up of 8 weights to transmit information from the first layer to the second layer *plus* 4 bias weights at the second layer *plus* 4 weights to transmit information from the second layer to the third layer *plus* 1 bias weight at the third layer. In a combined computation there are 22 weights: 8 weights to transmit information from the first layer to the second layer *plus* 4 bias weights at the second layer *plus* 8 weights to transmit information from the second layer to the third layer *plus* 2 bias weights at the third layer.

More generally, the determination of approximation functions for NJ endogenous variables in NJ separate neural network problems with N1 exogenous variables and N2 nodes in 1 hidden layer requires the estimation of NJ*[N1*N2+N2+N2+1] weights. This is reduced to [N1*N2+N2+ N2*NJ +NJ] weights in the combined problem. As we will see in section 5, this is a huge saving when there are large numbers of exogenous and endogenous variables. The advantages of the combined approach are accentuated in problems with more than one hidden layer.

Our example in which there was no loss of accuracy in the combined problem as we go from one endogenous variable to two is rather a special case, facilitated by the second reduced form equation being linear. Nevertheless, it illustrates a broadly applicable idea.

If we have successfully implemented a neural network approximation of reduced forms for NJ endogenous variables, then it is likely that we can extend this to additional endogenous variables in an expanded neural network optimization problem with extra weights only in the last layer. These are necessary to obtain the approximation functions for the additional endogenous variables. The number of extra weights is $\Delta NJ * [N(J-1)+1]$ where ΔNJ is the number of additional endogenous variables and N(J-1) is the number of nodes in the last hidden layer. Linear combinations of the functions at the last hidden layer can give considerable flexibility in approximating the reduced forms for the additional endogenous variables. This is illustrated in Figure 3 in which the functions used in approximating V_1^{Endo} are combined with new weights in approximating V_2^{Endo}.

Figure 3. Model with 2 exogenous and 2 endogenous variables approximated by an NN computation with a single 4-node hidden layer



The solid red curve is the true reduced form for V_1^{Endo} and the red straight line is the true reduced form for V_2^{Endo}. The 2-piece linear line aob is the V_{21} function defined in (4.16). The 2-piece linear line hcb is the V_{22} function defined in (4.17). The 2-piece linear line eod is the V_{23} function defined in (4.18). The 2-piece linear line gfd is the V_{24} function defined in (4.19). With the weights defined in (4.15) for the first node in the third layer, we obtain, as in example 2, the approximation for V_1^{Endo} given by the four linear pieces, hn, no, om and mg. In determining the approximation for V_2^{Endo}, the weights for the second node in the third layer defined by (4.37) eliminate the V_{22} and V_{24} functions. At the same time, they tilt the ao and the eo pieces of the V_{21} and V_{23} functions so that they become coincident with the V_2^{Endo} line in the north-east and south-west quadrants.

5. Using NN to develop the Destructive Events Tool (DET)

This section provides a brief description of our experience in constructing DET to approximate the reduced form for the model described by (2.1) – (2.2).

As mentioned earlier, we created a database for DET by conducting 150,000 TERM simulations. The values for the 725 exogenous variables (capital destruction in 23 industries and 29 regions, and deaths and evacuations in 29 regions) were chosen via a process involving random choice of location and other characteristics of the event. We recorded results for 864 endogenous variables. These included GDP and employment by region, and output by industry and region.

Each TERM simulation took about 20 seconds on our 32 core Ryzen machine. The simulations were conducted with GEMPACK version 12.2. We used the Runge-Kutta Dormand Prince method with adaptive step size and an epsTolerance value of 0.1 (see Horridge *et al.*, 2018, GEMPACK Manual v. 12.2). Conducting the 150,000 simulations took about 3 days using 16 parallel jobs on a 32 core machine.

From the simulations, we extracted 135,000 data points, with each data point consisting of values for 725 exogenous variables and 864 endogenous variables. Using these data, we constructed ("trained") an NN containing an input layer with 725 nodes (one for each exogenous variable) and an output layer with 864 nodes (one for each endogenous variable). Between the input and output layers we inserted 4 hidden layers each containing 725 nodes. The activation functions were ReLU. The loss function used in optimizing the network was mean squared error.

The total number of weights in the resulting network was 2,732,664 [=4*(725*725 +725)+725*864+864)]. The large number of hidden layers was required to appropriately represent the non-linearities in the CGE model and allow for accurate prediction of the effects of both small-scale and large-scale shocks. The optimization was done using the Adam optimizer as implemented in a tensorflow package (see https://www.tensorflow.org/). We chose to terminate after 1000 optimisation steps, by which time the value of the loss function for the training data did not change anymore. The optimization of the neural network took approximately 1 hour on a standard desktop machine.

Using a single NN calculation rather than separate calculations for each endogenous variable sharply reduced the number of weights for which we had to find optimal values. If we had attempted to solve 864 separate NN problem with 4-hidden layers each having 725 nodes, then the total number of weights would have been close to 2 billion [={4*(725*725 +725)+725+1)}*864].

We supplied the client with the NN approximation functions for 864 endogenous variables together with a program for inputting shocks to 725 exogenous variables.

Validation tests

Having used the 135,000 data points to create the NN reduced-form approximations, we used the remaining 15,000 out-of-sample data points to assess the accuracy of the NN reduced-form approximations. Figures 4 and 5 compare results generated by TERM with those generated by our NN reduced forms.

Figure 4. % effects on national GDP generated by TERM and the NN approximation equation in 15,000 out-of-sample tests

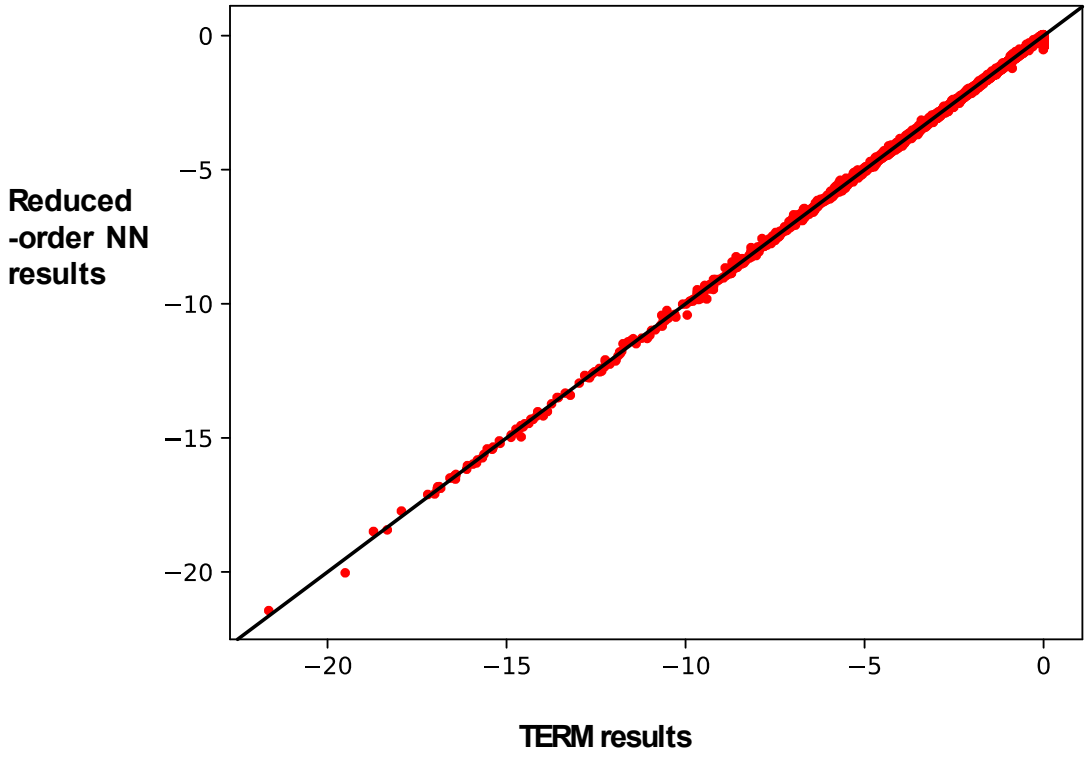
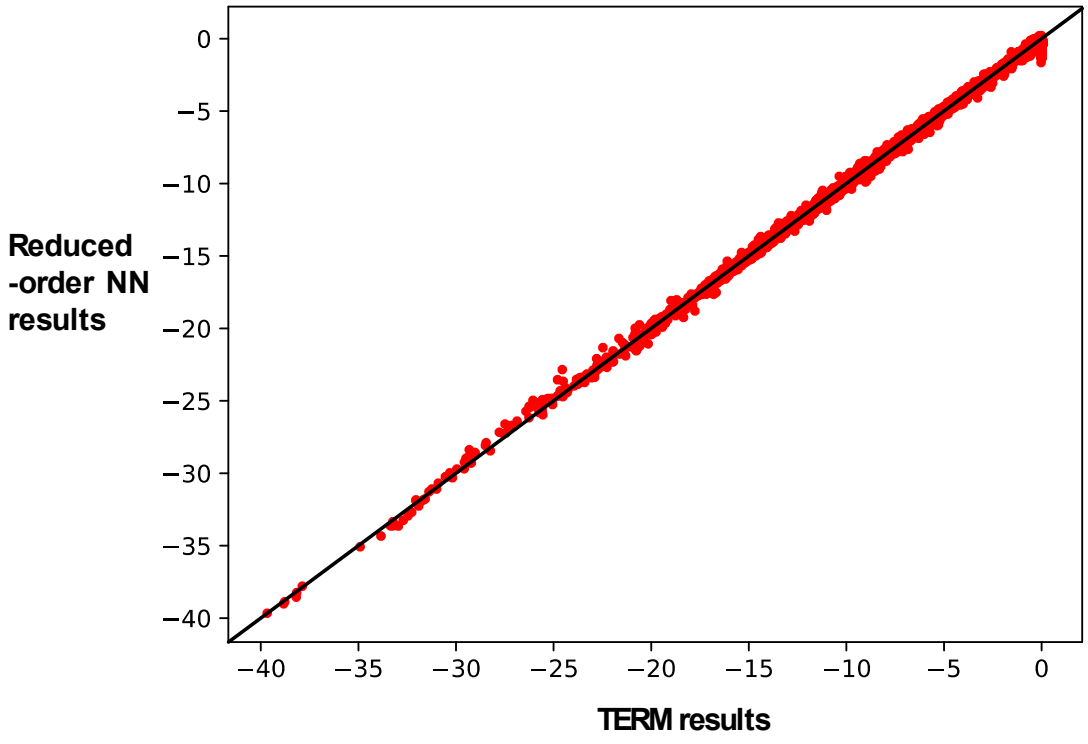


Figure 5. % effects on employment in a major exporting region generated by TERM and the NN approximation equation in 15,000 out-of-sample tests



The 15,000 dots in each figure show TERM results for the 15,000 additional simulations on the horizontal axis and NN results on the vertical axis. A perfect fit generates a dot on the 45-degree line.

The figures refer to just two variables, national GDP and employment in a major exporting region. The tight fit shown in these figures applies to the wide range of the macro and regional variables that we tested.

In conducting both the 135,000 training simulations and the additional 15,000 test simulations, we allowed variations in the exogenous variables over the full ranges of what was considered plausible for the destructive events under consideration. As can be seen from Figure 4, the test simulations produced national GDP results in the range zero to -22 per cent. For employment in the selected region, the range of TERM results shown in Figure 5 was even greater, from slightly positive to -40 per cent. Reassuringly, the NN approximations in both figures are accurate over the entire range of TERM results.

6. Concluding remarks

Only 10 years ago, the work described in this paper would not have been feasible for a relatively small research organization relying mainly on personal computers. The computational challenge and the investment of human time would have been too great. Two factors have changed that situation and made it possible to successfully complete this project.

The first is major improvements to GEMPACK software over this period. Solution times per integration step have been reduced by a factor of 4 or more for large models (https://www.copsmodels.com/gpfort.htm) by using a newly developed LU algorithm. Recently added Runge-Kutta integrators using adaptive step-size methods produce accurate solutions using only a fraction of the number of steps previously required. These facts combined with developments in computer hardware reduce the task of computing 150,000 CGE simulations from more than a month on multiple PC's to 3 days on a single machine.

The second factor is the availability of fast and easy-to-use libraries for NN development such as tensorflow (released in 2015). With these libraries, implementing a neural network as described here is a routine task, whereas before, expert knowledge and significant programming time would have been required.

The feasibility of NN methods dramatically increases the detail and accuracy that can be built into mimic equations for CGE reduced forms. An NN mimic system can include a large number of exogenous variables with wide ranges of shocks and can capture the non-linearities of the CGE model. The NN mimic system described in this paper produces results that closely match those from a large-scale CGE model for wide variations in 725 exogenous variables. In scope and tested accuracy, this is well beyond any previous mimic system for a CGE model.

References

Britz, W., J. Li and L. Shang (2021), "Combining large-scale sensitivity analysis in computable general equilibrium models with machine learning: an example application to policy supporting the bio-economy", 2021 GTAP Conference paper, available at https://www.gtap.agecon.purdue.edu/uploads/resources/download/10483.pdf .

Dixon, P.B., M. Jerie, M.T. Rimmer and G. Wittwer (2019), "Rapid assessments of the economic implications of terrorism events using a regional CGE model: creating GRAD-ECAT (**G**eneralized, **R**egional **A**nd **D**ynamic **E**conomic **C**onsequence **A**nalysis **T**ool)", chapter 6, pp. 121-161, in Okuyama, Y. and A. Rose. (eds.). *Advances in Spatial and Economic Modeling of Disaster Impacts*, Springer Nature, Switzerland.

Horridge, J.M., J. Madden and G Wittwer (2005), "Using a highly disaggregated multi-regional single country model to analyze the impacts of the 2002-03 drought on Australia", *Journal of Policy Modeling*, vol 27, pp. 285-308.

Horridge J.M., Jerie M., Mustakinov D. & Schiffmann F. (2018), *GEMPACK manual*, version 12.2, GEMPACK Software, ISBN 978-1-921654-34-3.

Wittwer, G. and Horridge, M. (2018), "Prefectural Representation of the Regions of China in a Bottom-up CGE Model: SinoTERM365", *Journal of Global Economic Analysis*, 3(2), pp. 178- 213.

Rose, A., F. Prager, Z. Chen, and S. Chatterjee (2017), *Economic Consequence Analysis of Disasters: The E-CAT Software Tool*, Singapore, Springer.

Wittwer, G., editor (2017), *Multi-regional dynamic general equilibrium modelling of the U.S. economy: USAGE-TERM Development and applications*, Springer, Switzerland.

Wittwer, G, (2024), "The economic impacts of a hypothetical foot and mouth disease outbreak in Australia"*, Australian Journal of Agricultural and Resource Economics* 68(1):23- 43, available at: https://onlinelibrary.wiley.com/doi/full/10.1111/1467-8489.12546 .