EUROPEAN CENTRAL BANK
EUROSYSTEM

Working Paper Series

Massimo Ferrari Minesso, Carla Frenzel

Sequential solution for DSGE models with deep neural networks

No 3236

Abstract

This paper develops a sequential deep learning algorithm for solving dynamic stochastic general equilibrium (DSGE) models. The algorithm trains a deep neural network to approximate the model's policy functions across four progressive phases: steady-state anchoring, exploration around the steady state, simulation on the ergodic set, and Monte Carlo integration of stochastic expectations. Training requires no pre-computed starting approximation: the network initialises from the analytically known steady state and constructs its training data endogenously, resolving the circularity between the training distribution and the solution. A systematic comparison across network architectures shows that shallow, moderately wide networks with an intermediate steady-state penalty consistently deliver the best accuracy at the lowest computational cost. We apply the method to a two-country open-economy model and show that large tariff shocks generate non-linearities that local methods cannot reproduce even at higher orders.

Keywords: DSGE models; deep neural networks; solution methods; policy function approximation; non-linear solution.

JEL Codes: C45; C63; C68; E13; F13.

Non-technical summary

Macroeconomic models are central tools for both academic research and policy analysis. They allow economists to ask quantitative questions about how the economy reacts to specific events, such as a change in interest rates, a productivity boom, or a tariff. To answer these questions, the models must be solved: that is, the rules describing how households, firms, and policymakers will behave in every possible state of the economy must be computed. Solving even a moderately sized model is technically demanding, and the methods commonly used in practice each have important limitations. The most widely employed techniques rely on a local approximation of these decision rules around a stable point known as the steady state. They are fast and scalable, but lose accuracy when the model's economy is hit by large shocks or when the rules are strongly non-linear, as is increasingly the case in the questions central banks are asked to address: the effective lower bound on interest rates, financial crises, energy price spikes, and large changes in trade policy.

This paper develops a new sequencing to solve macroeconomic models that overcomes these limitations using deep neural networks — the same class of statistical tools that underpin recent advances in artificial intelligence. A neural network is a flexible function that learns by adjusting its internal parameters to fit data. In our application, the network is trained to satisfy the equations of the economic model directly, rather than to fit observed data: at the end of training, it produces a globally accurate description of how the economy responds to shocks of any size, anywhere in the relevant range of conditions. This makes neural networks a natural complement to the existing toolkit when the questions of interest involve unusually large shocks or markedly non-linear dynamics.

The paper makes four contributions. The first is a new training algorithm. Existing methods typically require an initial guess of the solution, often borrowed from a simpler approximation, before the neural network can begin to learn. We propose a sequential procedure that does not need any such starting point: training begins from the long-run equilibrium of the model, which can always be computed analytically, and then progressively introduces dynamics and uncertainty in four steps. This is important in practice because it removes a substantial barrier to entry for researchers wishing to adopt the methodology, and because it eliminates the risk of inheriting biases from the initial

approximation.

The second contribution is to resolve a circularity that has held back the application of deep learning to economic models: the data on which the network is trained should describe the way the economy behaves under the very rules that the network is trying to learn. Our procedure addresses this by allowing the network to construct its own training data progressively, starting from the steady state and refining it as the approximation improves. The result is a self-contained method that requires no auxiliary model.

The third contribution is a systematic study of how the choices made in designing a neural network — the number of layers, the number of neurons per layer, the strength of the anchoring to the steady state — affect the quality and the cost of the solution. The exercise is conducted on the canonical real business cycle model, a small benchmark widely used in the field. The conclusion is practical: shallow networks of two or three layers, with a moderate number of neurons and an intermediate steady-state anchor, consistently deliver the most accurate solution at the lowest computational cost. Deeper or wider networks bring no systematic gain and can prevent the algorithm from converging.

The fourth contribution is an application that illustrates why the methodological investment is worthwhile. We use the algorithm to solve a two-country open-economy model and to study the effects of tariffs of empirically realistic size. The exercise reveals behaviour that is fundamentally non-linear: as the size of the tariff increases, the responses of consumption, inflation and the exchange rate do not simply scale up but can change sign. In particular, the currency of the country imposing the tariff appreciates after a small shock — the textbook result — but depreciates after a large one, as the contractionary effects of the tariff begin to dominate and prompt a more accommodative monetary policy response. This pattern aligns with the empirical evidence on the response of the US dollar to recent tariff announcements when retaliation was anticipated. A second-order perturbation method, the standard alternative for capturing some non-linearities in DSGE models, partially detects the change in magnitude but misses the change in direction of the exchange-rate response entirely. This result demonstrates that the global solution method is necessary, not merely refined: large trade-policy shocks of the kind currently debated in the public discussion cannot be analysed accurately with standard local techniques.

1 Introduction

This paper develops a deep learning method for solving dynamic stochastic general equilibrium (DSGE) models by training deep neural networks to approximate the policy functions that map state variables into controls. Deep learning offers a powerful alternative to traditional numerical methods because it can deliver globally accurate solutions even when the state space is high-dimensional or when shocks are large and nonlinear — settings where conventional approaches either break down or become computationally infeasible. We contribute to this growing literature along four dimensions: i) propose steady state anchoring for the neural network solution; ii) address practical barriers in training models without known solutions; iii) explore how network architecture affects the solution; iv) apply the neural network to solve a tariff model.

At its core, solving a DSGE model amounts to finding unknown functions — policy rules or value functions — that map the current state of the economy into agents' optimal decisions, prices, and allocations. These functions must satisfy a system of intertemporal optimality conditions, such as Euler equations, budget constraints, and market clearing conditions, for all possible realizations of the state variables. Because closed-form solutions are generally unavailable, researchers rely on numerical approximation. In a deep learning approach, the unknown policy functions are parameterized as deep neural networks, and the network weights are chosen so as to minimize the residuals of the model's equilibrium conditions evaluated over a set of points in the state space. The key insight, emphasized by Fernández-Villaverde (2025), is that a neural network does not merely approximate the policy function directly as a function of the state variables X; rather, it first transforms the states into a more efficient internal representation $\phi(X)$ through its hidden layers, and then applies a simple (nearly linear) mapping to this transformed representation. This automatic discovery of informative representations is what allows neural networks to handle high-dimensional and highly nonlinear environments where the researcher's own economic intuition may not suffice to engineer the appropriate transformation by hand.

In practice, the algorithm proceeds as follows. The policy functions are parameterized as neural networks $\hat{d}(X;\theta)$, where θ denotes the vector of network weights. A loss function is then constructed from the model's equilibrium conditions — for instance, the squared Euler equation residuals or a combination thereof — and evaluated at a set of

points drawn from the state space. The weights θ are updated via the Adam optimizer (Kingma and Ba, 2014) to minimize this loss. Once training is complete, the neural network provides a global approximation to the policy function that can be evaluated at any point in the state space, yielding impulse responses, simulated moments, or welfare calculations without further computation. This formulation builds directly on Maliar et al. (2021), who showed how to cast lifetime reward functions, Bellman equations, and Euler equations into objective functions suitable for deep learning platforms such as TensorFlow and PyTorch, and on whose foundations we build.

A crucial complication, noted by Fernández-Villaverde (2025), is what one might call the *equilibrium loop*: the data needed to train the neural network — namely, the points in the state space at which the loss function is evaluated — are themselves generated by simulating the model under the current (imperfect) approximation of the policy function. The training data and the object being estimated are thus jointly determined. One natural response is to simulate training data from a linearised version of the model, but this approach is unsatisfactory when nonlinear methods are warranted: the ergodic set implied by the linear solution may differ substantially from the true one, and any nonlinearity in the decision rule must be inferred from data generated under the assumption that none exists. Restricting training to a fixed rectangular grid avoids this problem but wastes computational resources on regions of the state space never visited in equilibrium and becomes prohibitively costly as the dimension of the state space grows (Judd et al., 2011).

In this paper, we make four contributions to the literature on solving DSGE models with deep neural networks.

Our first contribution is methodological. We propose a sequential training algorithm composed of four phases, each building on the previous one. The key innovation is to begin by anchoring the network at the analytically known deterministic steady state of the model, requiring no prior knowledge of the dynamic solution whatsoever. This step ensures that the network starts from an economically plausible configuration before any dynamic training begins. We then progressively introduce dynamics: first through random exploration around the steady state, then by training on the model's own simulated trajectories, and finally by incorporating Monte Carlo integration over future shocks to enforce the stochastic Euler equations. A steady state penalty, maintained throughout all

phases, plays a stabilizing role analogous to regularization (Fernández-Villaverde, 2025): it prevents the optimizer from drifting toward solutions that, while locally minimizing the Euler equation residuals, are inconsistent with the model's long-run equilibrium. The training-on-simulated-data component of our approach is related to the parameterized expectations algorithm of Den Haan and Marcet (1990) and to the framework of Maliar et al. (2021); our contribution is the sequential structure that resolves the circularity between training data and the solution without relying on any auxiliary model.

Our second contribution addresses a practical barrier: how to train a deep neural network to solve a DSGE model when one has no pre-computed approximation to use as a starting point. Existing implementations typically initialise training from a perturbation solution or from the ergodic distribution implied by the linear model, which presupposes knowledge of the object one is trying to improve upon. Our staggered algorithm needs only the steady state values of the endogenous variables, which are always available analytically. The network then constructs its own increasingly representative training data as it improves, concentrating computational effort on the model's true ergodic set in the spirit of Judd et al. (2011), without inheriting any bias from a linear approximation.[1]

Our third contribution is a systematic exploration of the network architecture and hyperparameter space. Using the canonical real business cycle model of Kydland and Prescott (1982) as a controlled testing environment, we vary the number of neurons per layer (32, 64, 128), the number of hidden layers (2, 3, 5, 7), and the weight on the steady state penalty (0.5, 1, 5). This exercise complements the analysis of Beck et al. (2024), who study architecture choices in a single-phase setting; our specific contribution is to document how these design dimensions interact with the sequential training protocol. The results support a clear practical recommendation: shallow networks of two to three layers, moderate width of 32 neurons, and an intermediate steady state penalty weight of $\lambda_{ss} = 1$ consistently deliver the best accuracy at the lowest computational cost.

Our fourth contribution is an application to international macroeconomics. We apply the method to a two-country open-economy model with Rotemberg pricing and use it to simulate the effects of tariff shocks of empirically realistic magnitudes — a setting in which perturbation methods are unreliable because the shocks are large relative to the steady state, and perfect-foresight algorithms cannot account for the uncertainty surrounding

[1]The steady state is a standard starting point for business cycle models. Models that do not have a steady state could be trained to retrieve the desired initial conditions if no shocks are present.

future trade policy that is first-order for households and firms making forward-looking decisions. The exercise reveals strongly non-linear propagation that local methods cannot reproduce: as the shock size increases, the exchange rate switches from an appreciation under a small tariff to a sizeable depreciation under a large one. This sign reversal aligns with the empirical evidence of Ostry et al. (2025) on the response of the dollar to tariff announcements once retaliation is anticipated, and is missed by a second-order perturbation solution that, by construction, scales the small-shock appreciation up rather than reversing it. Recovering the direction of the exchange-rate response observed in the data for large tariff events is, in our view, the central methodological pay-off of the global solution approach for applied counterfactual policy analysis.

To understand why deep learning constitutes an important advance, it is useful to contrast it with the methods that have dominated the field (Judd, 1998; Fernández-Villaverde et al., 2016). Perturbation methods approximate policy functions via Taylor expansions around the steady state and remain the workhorse of applied macroeconomics owing to their speed and scalability, but are inherently local and fail when shocks are large or constraints occasionally bind (Fernández-Villaverde, 2025). Global methods based on value function iteration or projection with Chebyshev polynomials deliver higher accuracy over the full state space but suffer from the curse of dimensionality (Bellman, 1957): computational cost grows exponentially with the number of state variables. Sparse grid methods (Brumm and Scheidegger, 2017) and Gaussian process regression (Scheidegger and Bilionis, 2019) have extended the frontier, but the fundamental scaling constraint persists. Perfect-foresight algorithms (Adjemian et al., 2026) are useful for nonlinear transition paths but do not account for uncertainty about future shocks and provide a solution only along a single path. Deep learning overcomes many of these limitations: networks are universal approximators (Cybenko, 1989; Hornik et al., 1989), the loss is evaluated on randomly sampled mini-batches rather than a grid, and the trained network delivers a global policy function everywhere in the state space.

Related literature. This paper contributes to several strands of the literature on numerical methods for dynamic economic models. The foundational techniques for solving DSGE models are surveyed in Judd (1998) and Fernández-Villaverde et al. (2016). Within this toolkit, perturbation methods remain the workhorse of applied macroeco-

nomics (Fernández-Villaverde et al., 2016; Adjemian et al., 2026). Global methods based on value function iteration or projection suffer from the curse of dimensionality; sparse grid techniques (Brumm and Scheidegger, 2017) and Gaussian process regression (Scheidegger and Bilionis, 2019) have emerged as alternatives. For heterogeneous-agent models, Krusell and Smith (1998) proposes replacing the distribution of agents with a small set of moments; refinements include perturbation around the aggregate steady state (Reiter, 2010; Bayer and Luetticke, 2020), model reduction (Winberry, 2018), and sequence-space methods (Auclert et al., 2021).

The application of neural networks to economic models has a long history. Duffy and McNelis (2001) provided an early demonstration that neural networks could approximate policy functions in the real business cycle model, and Den Haan and Marcet (1990) developed the parameterized expectations algorithm, which shares the core idea of fitting decision rules on simulated data. Duarte (2018) and Duarte et al. (2024) show how neural networks can solve continuous-time asset pricing and growth models. Maliar et al. (2021) provided a unified framework casting Euler equation and Bellman formulations into objective functions amenable to stochastic gradient descent on simulated data, on which we build directly. Azinovic et al. (2022) developed deep equilibrium nets for overlapping generations economies with aggregate risk. Fernández-Villaverde et al. (2023) applied neural networks to track the wealth distribution in a model with financial frictions, while Villaverde et al. (2024) extended this approach to study inequality at the zero lower bound. On the practical side, Beck et al. (2024) provide a detailed guide to design choices that affect the accuracy of deep learning solutions, and Hall-Hoffarth (2023) propose hard-constraint methods that enforce economic restrictions directly in the network architecture rather than through penalty terms. Fernández-Villaverde (2025) offers a comprehensive survey of these developments.

Our paper builds on and extends this literature in three directions. We share with Maliar et al. (2021) the idea of training on simulated data and minimizing Euler equation residuals, but introduce a sequential training protocol that explicitly anchors the network at the deterministic steady state and resolves the circularity between training data and the solution without any auxiliary model. Our systematic exploration of the architecture space complements Beck et al. (2024) by documenting how network depth, width, and steady state penalty weight interact within our sequential protocol — a di-

mension their analysis does not address. Finally, our application to large tariff shocks in a multi-country model extends the class of problems tackled by deep learning methods in international macroeconomics, providing a global nonlinear solution in a setting where perturbation methods are unreliable and perfect-foresight algorithms cannot account for policy uncertainty.

This paper also contributes to the debate on the macroeconomic consequences of tariffs. Krugman (1982) provided one of the earliest formal analyses, emphasising that the effects of import protection depend critically on the prevailing exchange-rate regime. The modern New Keynesian open-economy literature has revisited the question in general-equilibrium monetary settings building on Galíand Monacelli (2005); Lindé and Pescatori (2019) reassess the Lerner symmetry documenting quantitatively important deviations under complete asset markets, slow exchange-rate adjustment, and direct tariff pass-through into prices. The recent resurgence of trade protectionism has also prompted a renewed effort to characterise tariff transmission and to identify the appropriate monetary response. Bergin and Corsetti (2023) analyse the optimal monetary stabilisation of import and export tariffs and emphasise that, under dominant currency pricing, the optimal policy stance can be expansionary in the imposing country. Bianchi and Coulibaly reach a related conclusion in an environment with incomplete international financial markets, where the tariff generates a fiscal externality that an expansionary monetary policy partially offsets, and Monacelli (2025) compares alternative implementable monetary regimes and shows that under CPI inflation targeting tariffs are unambiguously contractionary, while exchange-rate pegs and PPI targeting yield qualitatively different transmission.[2] On the empirical side, Furceri et al. (2022) document, using a panel of 151 countries, that tariff increases lead in the medium term to declines in output and productivity and to a real exchange-rate appreciation. The exchange-rate response itself remains contested: while the Mundell-Fleming benchmark predicts an appreciation of the imposing country's currency, dynamic open-economy models can deliver a depreciation when import substitution is low (Ostry, 1991), when domestic interest rates fall (Krugman, 1982), or when monetary policy responds optimally to the shock (Bergin and Corsetti, 2023; Bianchi and Coulibaly). The recent econometric evidence in Ostry et al. (2025) reconciles these

[2]Other contributions include Erceg et al. (2023) on the equivalence between trade policies and fiscal devaluations, Auclert et al. (2025) on the implications for the trade balance, and Werning et al. (2025) who interpret tariffs as cost-push shocks shifting the Phillips curve.

views by showing that the sign of the dollar response depends systematically on whether retaliation is anticipated, with depreciation prevailing under retaliation, in line with the predictions of dominant-currency-pricing models. Across this literature, however, the dominant solution method remains either perturbation around a steady state or perfect-foresight algorithms — neither of which is well suited to the analysis of large tariff shocks of empirically realistic magnitudes. We show instead that a reverse of the exchange rate response — that is a depreciation following of the currency of the tariff-imposing economy — can be reached in a Neo-Keynesian model under non-linear solutions if the shock is large enough. In this case, the contractionary effect dominate those on trade balance and the supply and demand of currency.

The remainder of the paper is organized as follows. Section 2 presents an introduction to deep neural networks and their application to dynamic economic models. Section 3 describes the sequential training algorithm and its implementation in `PyTorch`. Section 4 presents the results for the RBC model, including the systematic comparison of network specifications. Section 5 develops the international model and analyzes the effects of tariff shocks. Section 6 concludes.

2 An introduction to deep neural networks

This section provides an overview of neural networks, the algorithms used to train them and how they can be used to solve general equilibrium models. While the terminology originates in computer science, the underlying ideas are closely related to familiar concepts in econometrics: function approximation, non-linear regression, and gradient-based optimization.[3]

Neural networks are a class of parametric function approximators inspired by the architecture of biological nervous systems, though the analogy should not be taken too literally. Originally developed in the 1940s and 1950s (McCulloch and Pitts, 1943; Rosenblatt, 1958), they remained largely dormant until advances in hardware and software reignited interest in the 2010s. The breakthrough of Krizhevsky et al. (2012), who demonstrate that deep neural networks could outperform existing methods in image classification, marked

[3]For a comprehensive textbook treatments of neural networks, see Goodfellow et al. (2016) and Prince (2023); for a discussion aimed specifically at economists, see Fernández-Villaverde (2025) and Fernández-Villaverde et al. (2024).

the beginning of the current revolution. Since then, neural networks have achieved remarkable success across a wide range of domains: Mnih et al. (2015) show that deep reinforcement learning could surpass human performance in complex control tasks, while the attention mechanism of Vaswani et al. (2017) laid the foundation for the large language models that now power modern generative artificial intelligence. The common thread across these applications is that neural networks excel at learning unknown mappings from high-dimensional inputs to outputs, precisely the task that arises when solving for the policy functions of a dynamic economic model. This connection has motivated a growing literature applying deep learning to equilibrium computation in economics.

Before proceeding with the description of how deep neural networks are applied to solving structural model, it is perhaps useful to fix terminology. A *neuron* (also called a node or unit) is the elementary building block of a neural network: it takes a vector of inputs, computes a weighted sum plus a constant (the *bias*), and passes the result through a non-linear *activation function* to produce a scalar output. Common activation functions include the hyperbolic tangent, the sigmoid, and the rectified linear unit (ReLU), each introducing a specific type of non-linearity while remaining computationally inexpensive to evaluate and differentiate. A collection of neurons that share the same inputs and operate in parallel is called a *layer*; the number of neurons in a layer is its *width*. A neural network is a composition of several layers: the first is the *input layer*, which receives the raw state variables (e.g. capital and shocks); the last is the *output layer*, which produces the network's predictions (e.g., consumption and labor); and any layers in between are *hidden layers*, where the network constructs increasingly refined internal representations of the data. The number of layers is the network's *depth* — a network with more than one hidden layer is called *deep*, hence the term deep learning. The full collection of weights and biases across all layers constitutes the parameter vector θ, which is determined during training; the choices of depth, width, and activation functions define the network's *architecture*.[4]

2.1 From linear approximation to neural networks

Solving a DSGE model requires approximating functions that do not have a closed form solution— decision rules, value functions, or pricing functions — that map state variables

[4]For a comprehensive treatment of these concepts, see Goodfellow et al. (2016) and Prince (2023).

$X = (x_1, \ldots, x_N)$ into equilibrium outcomes y (Fernández-Villaverde et al., 2016). The simplest and most common approximation ($g_\theta(\bullet)$) is linear:

$$y \approx g_\theta^{LA}(X) = \theta_0 + \sum_{n=1}^{N} \theta_n x_n \tag{1}$$

this works well when the target function is close to linear, but fails to capture the non-linearities that are part of most modern economic models: kinks from occasionally binding constraints, concavity of value functions, or threshold effects in policy rules.

A natural extension is to include polynomial terms, such as Chebyshev polynomials, but these suffer from the so-called curse of dimensionality: the number of terms grows exponentially with the dimension N of the state space (Judd, 1998). A neural network addresses this problem from a different angle. Rather than constructing an elaborate non-linear function of X directly, it first transforms the inputs into a new, more informative representation $\varphi(X)$, and then applies a simple (nearly linear) mapping to this representation (Fernández-Villaverde, 2025):

$$y \approx g_\theta(\varphi(X)) \tag{2}$$

the network learns the transformation φ automatically during training. This is the key insight: rather than relying on the researcher to engineer the right variable transformations (e.g., taking logs, detrending), the neural network discovers them from the data.

2.2 Architecture of a neural network

A feedforward neural network constructs the transformation φ through a sequence of simple operations organized in layers (see Figure 1). The construction proceeds in three steps.

Pre-activations. For each of M neurons in a given layer, compute a weighted sum of the inputs plus a constant (the bias):

$$a_m = \theta_{0,m} + \sum_{n=1}^{N} \theta_{n,m}\, x_n, \quad m = 1, \ldots, M \tag{3}$$

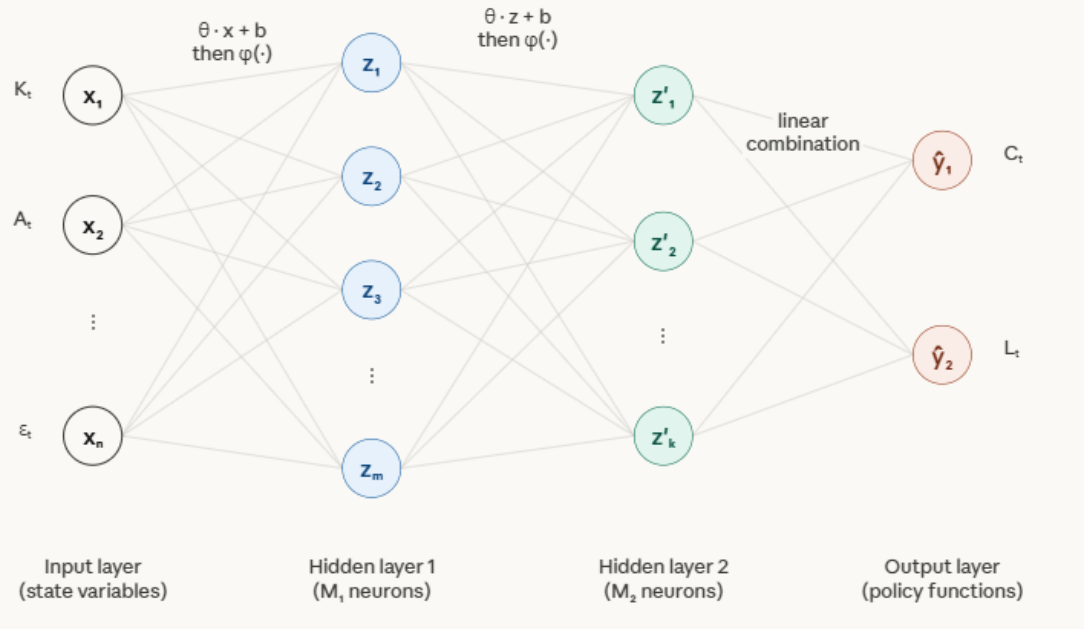

Figure 1: A deep neural network architecture applied to a DSGE model.
Notes: the figure shows a deep neural network applied to solve a DSGE model. Inputs are levels of the state variables and the shock (K_t, A_t, ε_t) while outputs are controls. The model features an input layer, an output layer and two intermediate (also called hidden) layers. Inputs are transformed by activation function. The first layers has $M_1 \in [1, m]$ neurons while the second hidden layer model has $M_2 \in [1, k]$ neurons. Each neuron performs an operation on the inputs from the previous layer through the activation function depicted as blue lines.

this is an affine transformation — the same operation as in a linear regression, applied M times with different weights.

Activation. Apply a non-linear function $\varphi(\cdot)$, called the activation function, to each pre-activation:

$$z_m = \varphi(a_m) \tag{4}$$

Equation (4) is the only non-linear step. Common choices include the hyperbolic tangent $\tanh(a)$, the sigmoid $\sigma(a) = 1/(1 + e^{-a})$, and the rectified linear unit (ReLU) $\max(0, a)$. The hyperbolic tangent tends to perform well when the target function is smooth and concave, as is typical of value and policy functions in standard DSGE models; the ReLU is a safer default when less is known about the shape of the target (Fernández-Villaverde, 2025; Beck et al., 2024).

Output. Combine the neuron outputs linearly to produce the network's prediction:

$$\hat{y} = \theta_0 + \sum_{m=1}^{M} \theta_m \, z_m = \theta_0 + \sum_{m=1}^{M} \theta_m \, \varphi\left(\theta_{0,m} + \sum_{n=1}^{N} \theta_{n,m} \, x_n\right) \tag{5}$$

this defines a single-layer (or shallow) neural network. Two classical results make this architecture attractive in principle: the universal approximation theorem (Cybenko, 1989; Hornik et al., 1989), which guarantees that a single-layer network with enough neurons can approximate any continuous function to arbitrary accuracy; and the result of Barron (1994), which shows that the approximation error decreases at rate $O(1/M)$ regardless of the input dimension N, in contrast to polynomial approximations whose convergence deteriorates exponentially with N.

A *deep* neural network stacks multiple such layers: the outputs z_m of one layer become the inputs to the next. With J hidden layers, the network applies J successive rounds of affine transformation and non-linear activation before producing the final output. Depth is valuable because it allows the network to build hierarchical representations: early layers capture simple patterns, while deeper layers combine them into progressively more complex features. With piecewise-linear activations such as ReLU, the expressiveness of the network grows exponentially with depth while computational cost increases only linearly (Fernández-Villaverde, 2025). In practice, a few layers (two to five) with a moderate number of neurons per layer (32 to 128) suffice for most DSGE applications.

2.3 Training: determining the weights

The vector θ collecting all weights across all layers must be chosen so that the network's output closely approximates the target function. Define a loss function that measures how far the network's predictions are from the desired outputs:

$$\mathcal{L}(\theta) = \frac{1}{L}\sum_{l=1}^{L} \|y_l - g_\theta(\varphi(X_l, y_l))\|^2, \tag{6}$$

where $\{(X_l, y_l)\}_{l=1}^{L}$ is a set of training data. In the context of solving DSGE models, y_l is typically zero (the target residual of the Euler equations) and X_l are points in the state space drawn either from a grid, from a distribution around the steady state, or from a simulation of the model itself (Maliar et al., 2021).

Training proceeds by *stochastic gradient descent* (SGD) (Robbins and Monro, 1951). Starting from a random initialization $\theta^{(0)}$, the weights are updated iteratively:

$$\theta^{(k+1)} = \theta^{(k)} - \eta\, \nabla_\theta \mathcal{L}(\theta^{(k)}) \tag{7}$$

where $\eta > 0$ is the learning rate, controlling the size of each step, and $\nabla_\theta \mathcal{L}$ is the gradient of the loss function with respect to the weights. The gradient is computed efficiently via *back propagation*, which applies the chain rule recursively through the layers of the network (Rumelhart et al., 1986). In practice, the gradient is not evaluated on the full dataset but on a small random subset (a mini-batch), which significantly reduces the computational cost per iteration. This is the same principle underlying stochastic approximation in econometrics: most of the information about the gradient direction is already contained in a small subsample. The noise introduced by mini-batch sampling is not merely a computational shortcut — it also helps the optimizer escape shallow local minima, improving generalization (Fernández-Villaverde, 2025).

Modern implementations typically use ADAM (Kingma and Ba, 2014), a variant of SGD that adapts the learning rate for each parameter based on past gradients. ADAM combines momentum (an exponential moving average of past gradients, which accelerates progress across flat regions of the loss landscape) with adaptive step sizes (which rescale the gradient for each weight individually). This makes training more robust to the choice of the initial learning rate and generally speeds up convergence.[5] A typical training algorithm is described in Figure 2

3 Solution algorithm

We propose a new deep learning algorithm to solve DSGE models. At its core, the algorithm trains a deep neural network (DNN) to approximate the model's decision rules—that is, the mapping from the current state of the economy to the optimal values of the control variables—by minimising the residuals of the model's equilibrium conditions over a stochastically simulated state space. The procedure is divided into four sequential training phases, each designed to progressively improve the accuracy of the approximation: an initialisation phase anchored at the deterministic steady state, a random exploration phase around the steady state, a simulation-based phase that focuses training on the ergodic set, and a final phase that incorporates Monte Carlo integration over future shocks to enforce the forward-looking Euler equations.

[5]The entire training procedure — forward propagation of inputs through the network, evaluation of the loss, back propagation of gradients, and weight updates — is automated by open-source libraries such as PyTorch and JAX, allowing the researcher to focus on the economic model rather than the numerical implementation.

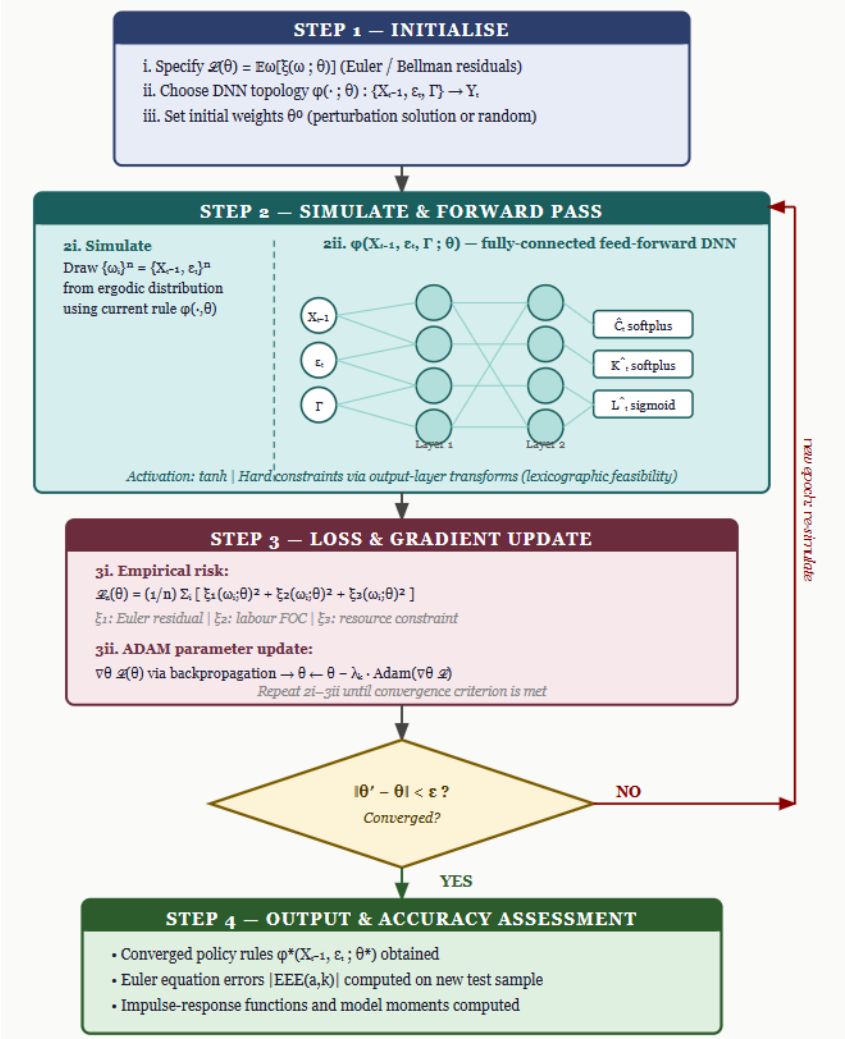

Figure 2: Standard training algorithm.

3.1 The Deep Neural Network

Consider a DSGE model with Y endogenous variables, of which X are state variables and $Y - X$ are control (jump) variables. The model is hit by a vector of structural shocks $\boldsymbol{\varepsilon}_t \in \mathbb{R}^{n_\varepsilon}$. Let $\mathbf{s}_t \in \mathbb{R}^X$ denote the vector of state variables at time t, which includes both endogenous states (e.g. the capital stock) and exogenous states (e.g. total factor productivity following a Markov process). The policy rule that maps states into controls is:

$$\mathbf{x}_t = \bar{\varphi}(\mathbf{s}_t) \tag{8}$$

where $\mathbf{x}_t \in \mathbb{R}^{Y-X}$ is the vector of control variables. The policy rule $\bar{\varphi}$ is in general a highly nonlinear function of the state vector and generally cannot be obtained in closed form.

We approximate $\bar{\varphi}$ with a deep neural network $\varphi(\cdot\,;\boldsymbol{\theta})$ parametrised by a vector of weights and biases $\boldsymbol{\theta}$. The DNN takes the state vector $\mathbf{s}_t$ as input and returns the controls $\mathbf{x}_t$ as output. This mapping from states to controls is the natural object to approximate in the deep learning literature: the universal approximation theorem guarantees that a sufficiently wide feedforward network can represent any continuous function on a compact set to arbitrary precision (Hornik et al., 1989),[6] and in high-dimensional settings deep networks scale more efficiently than tensor-product polynomial bases while being intrinsically robust to multicollinearity (Maliar et al., 2021). We further train the network on simulated data to concentrate computational effort on the ergodic set—the region of the state space actually visited in equilibrium—rather than on a fixed rectangular grid, thereby avoiding the curse of dimensionality that affects conventional projection methods (Judd et al., 2011; Fernández-Villaverde, 2025).

The network architecture is a fully connected feedforward network with L hidden layers, each containing H neurons, and `tanh` activation functions. Formally, with input $\mathbf{s} \in \mathbb{R}^X$, the ℓ-th hidden layer computes

$$\mathbf{h}^{(\ell)} = \tanh\!\left(\mathbf{W}^{(\ell)}\mathbf{h}^{(\ell-1)} + \mathbf{b}^{(\ell)}\right), \quad \ell = 1, \ldots, L, \tag{11}$$

with $\mathbf{h}^{(0)} = \mathbf{s}$ and the output layer returning $\widehat{\mathbf{x}} = \mathbf{W}^{(L+1)}\mathbf{h}^{(L)} + \mathbf{b}^{(L+1)}$. Monotone output transformations (`softplus` for strictly positive controls and `sigmoid` for bounded controls) are applied to enforce economic domain constraints without imposing explicit inequality constraints during optimisation.[7]

[6] The theorem states that given $\sigma : \mathbb{R} \to \mathbb{R}$ be a non-constant, bounded, and continuous activation function. For any $\varepsilon > 0$ and any continuous target function $f \in C(\mathcal{K})$ defined on a compact set $\mathcal{K} \subset \mathbb{R}^n$, there exists a single-hidden-layer feedforward network $\hat{f} : \mathbb{R}^n \to \mathbb{R}$ of the form

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{H} \alpha_j \, \sigma\!\left(\mathbf{w}_j^{\top}\mathbf{x} + b_j\right), \tag{9}$$

with weights $\mathbf{w}_j \in \mathbb{R}^n$, biases $b_j \in \mathbb{R}$, and output coefficients $\alpha_j \in \mathbb{R}$, such that

$$\sup_{\mathbf{x}\in\mathcal{K}} \left| f(\mathbf{x}) - \hat{f}(\mathbf{x}) \right| < \varepsilon. \tag{10}$$

[7] This is a standard practice which helps computationally to avoid impossible values for the endogenous variables.

3.2 Equilibrium Conditions and Loss Function

The equilibrium conditions are used to train the weights of the model so to minimise residuals. As an example, consider a model with two two equilibrium conditions. The first is a set of static conditions—such as intratemporal optimality conditions—that hold exactly period by period. The second is the Euler equation, which embeds an expectation over next-period outcomes. For a state $\mathbf{s}_t$ and controls $\mathbf{x}_t = \varphi(\mathbf{s}_t; \boldsymbol{\theta})$, the two residuals are:

$$r_{1,t}(\boldsymbol{\theta}) = f_1\big(\mathbf{s}_t, \mathbf{x}_t(\boldsymbol{\theta})\big), \tag{12}$$

$$r_{2,t}(\boldsymbol{\theta}) = \mathbf{s}_t^{\text{endo}} - \beta\, \mathbb{E}_t\big[g\big(\mathbf{s}_t, \mathbf{x}_t(\boldsymbol{\theta}), \mathbf{s}_{t+1}, \mathbf{x}_{t+1}(\boldsymbol{\theta})\big)\big], \tag{13}$$

where $g(\cdot)$ collects the terms of the Euler equation, and β is the subjective discount factor. The conditional expectation in Equation (13) is approximated by Monte Carlo integration: for each state $\mathbf{s}_t$ in the mini-batch, D draws of the innovation vector $\boldsymbol{\varepsilon}_{t+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ are used to simulate next-period states $\mathbf{s}_{t+1}^{(d)}$ according to the law of motion, and the network is evaluated at each draw to obtain $\mathbf{x}_{t+1}^{(d)} = \varphi(\mathbf{s}_{t+1}^{(d)}; \boldsymbol{\theta})$. The Monte Carlo estimator of the expectation is:

$$\widehat{\mathbb{E}}_t[g(\cdot)] = \frac{1}{D} \sum_{d=1}^{D} g\big(\mathbf{s}_t, \mathbf{x}_t, \mathbf{s}_{t+1}^{(d)}, \mathbf{x}_{t+1}^{(d)}\big). \tag{14}$$

The training objective is the *empirical loss*:

$$\mathcal{L}^n(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \Big[r_{1,i}(\boldsymbol{\theta})^2 + r_{2,i}(\boldsymbol{\theta})^2\Big] + \lambda_{\text{ss}}\, \mathcal{L}_{\text{ss}}(\boldsymbol{\theta}), \tag{15}$$

where the second term penalises deviations of the network predictions from the known deterministic steady state in case there are no shocks and the starting point is the steady state (with weight $\lambda_{\text{ss}} > 0$). This element of the loss is important, as we want a solution that:

$$\mathbf{x}_{ss} = \bar{\varphi}(\mathbf{s}_{ss}) \tag{16}$$

that is if the DNN is fed with the steady state state variables and no shocks (s_{ss}) returns the steady state values of the controls (x_{ss}). This is imprtant to ensure that solutions are coherent both dynamically and in equilibrium.

3.3 Training Procedure

The network weights θ are updated at each iteration k via the Adam optimiser (Kingma and Ba, 2014), a stochastic variant of gradient descent:

$$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k - \lambda_k \nabla_{\boldsymbol{\theta}} \mathcal{L}^n(\boldsymbol{\theta}_k), \tag{17}$$

where λ_k is a schedule-adjusted learning rate and the gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}^n$ is computed by automatic differentiation through the network and through the model equations.

A fundamental challenge in applying deep learning to DSGE models is that the network must be trained on data generated by the model's decision rule, yet that decision rule is precisely the object one seeks to approximate. One natural response is to simulate data from a linearised version of the model, but this approach is unsatisfactory when nonlinear methods are warranted: the ergodic set implied by the linear solution may differ substantially from the true one, and any non-linearity in the decision rule must be inferred from data generated under the assumption that none exists. Restricting training to a fixed rectangular grid avoids this problem but wastes computational resources on regions of the state space never visited in equilibrium and becomes prohibitively costly as the dimension of $\mathbf{s}_t$ grows (Judd et al., 2011).

Our approach resolves this circularity through a self-referential procedure that constructs its own training data progressively. The network is first anchored at the deterministic steady state, whose values (s_{ss} and x_{ss}) are available analytically, requiring no prior knowledge of the solution. In the next pahses the partial solution obtained in the preceding phase to simulate increasingly representative trajectories of the model: each iteration of the network produces a candidate decision rule, which is used to simulate the state process, which in turn provides the mini-batches for the next iteration. Because the simulated trajectories converge to the model's ergodic set as the network improves, the training distribution automatically adapts to concentrate effort where the solution is relevant. Finally, expectations through Monte Carlo integration over future shocks, using the current network to compute next-period controls at each draw. The algorithm is deemed to have converged when the equilibrium residuals $\mathcal{L}^n(\boldsymbol{\theta})$ fall below the prescribed tolerance ε and successive updates to $\boldsymbol{\theta}$ cease to alter the simulated trajectories, so that both the parameter vector and the induced ergodic distribution are jointly stable.

Training follows four sequential phases described in Algorithm A, each building on the solution obtained in the previous one. The staggered setting trains the network on increasingly complexity from discovering the steady state solution to learn dynamics when expectations matter. The four steps are the following, reported also in Figure 3.

Phase 1 — steady state anchoring. The network is first trained to reproduce the deterministic steady state. States are kep at their steady state level (s_{ss}) and no exogenous shocks are used. The loss penalises only deviations of the network's output from the known steady state controls x_{ss}, with a large penalty weight λ_{ss}. This initialisation ensures that the weights encode economically plausible behaviour before any dynamic training begins, preventing gradient descent from exploring regions of the state space that are never visited in equilibrium. It also does not require prior knowledge on the dynamic approximation of the decision rule $g(\bullet)$ as the steady state can be computed analytically.

Phase 2 — Random exploration. The network is exposed to a wider distribution of states, drawn randomly around the steady state. The full equilibrium loss is now active, including both the static and the Euler equation residuals, but next-period states are computed deterministically by setting future shocks to zero. This phase teaches the network the local dynamics of the model without yet requiring integration over the shock distribution. It is a natural starting point for exploring dynamics, albeit limited to a neighborood of the deterministic steady state.

Phase 3 — Simulation on the ergodic set. Mini-batches are no longer drawn from an arbitrary distribution but are instead subsampled from trajectories simulated with the current network under random shocks. Because simulated paths converge to the ergodic set, training effort is automatically concentrated on the region of the state space that is relevant in equilibrium, in the spirit of Judd et al. (2011). Next-period states continue to be evaluated deterministically. This allows the models to learn the dynamics excluding uncertainty over future shocks, which is a substantially easier task. The network trained without uncertainty is then given future shocks to learn expectations.

Phase 4 — Monte Carlo integration of expectations. The final phase retains the simulation-based sampling of Phase 3 but replaces the deterministic next-period evaluation with a Monte Carlo approximation of the conditional expectation operator. For each state in the mini-batch, D independent draws of the future shock vector $\boldsymbol{\varepsilon}_{t+1}$ are

used to evaluate the network at D distinct next-period states, and the Euler residual is averaged across draws before entering the loss. Gradient norms are clipped to ensure numerical stability. This phase enforces the forward-looking equilibrium conditions with full stochastic integration and constitutes the main training step of the algorithm. After this step the model has learned to approximate dynamics under future uncertainty.

The progressive structure of the four phases serves three purposes. First, it allows to train the model even when the solution is not known a priory, converging from the initial steady state to the final dynamic solution under uncertainty. Second, anchoring the network at the steady state in Phase 1 prevents the gradient descent from exploring economically implausible regions of the state space during early iterations, a well-known source of instability in the training of physics-informed and economics-informed neural networks (Maliar et al., 2021). Third, by progressively shifting the training distribution from a fixed neighbourhood of the steady state (Phase 2) to draws from the model's ergodic set (Phases 3 and 4), the algorithm follows the principle advocated by Judd et al. (2011): computational effort is concentrated where the solution is economically relevant, rather than uniformly spread over a predefined rectangular domain. The inclusion of Monte Carlo integration in Phase 4 ensures that the Euler equation residuals accurately reflect the conditional expectation operator, an approach closely related to the all-in-one (AiO) integration operator of Maliar et al. (2021) and to the simulation-based methods discussed in Fernández-Villaverde (2025).

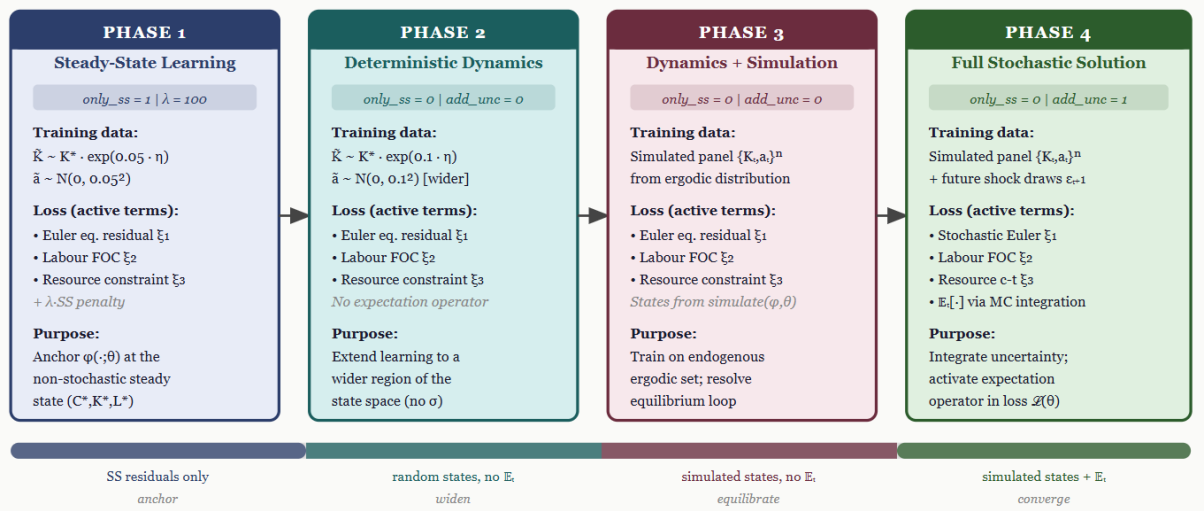

Figure 3: Four-step training algorithm.

4 Testing the solution

This section tests the deep neural network solution under alternative configurations of the networks using a simple, standard DSGE model as testing ground.

4.1 A simple DSGE model

The model is the canonical real business cycle economy of Kydland and Prescott (1982), as implemented in the benchmark example of `Dynare` (Adjemian et al., 2026). A representative household maximises expected lifetime utility:

$$\mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \left[\log C_t - \theta L_t\right], \tag{18}$$

where C_t denotes consumption, L_t denotes hours worked, $\beta \in (0,1)$ is the subjective discount factor, and $\theta > 0$ governs the disutility of labour. Output is produced according to a Cobb–Douglas technology:

$$Y_t = e^{a_t} K_{t-1}^{\alpha} L_t^{1-\alpha}, \tag{19}$$

where K_{t-1} is the capital stock carried into period t, $\alpha \in (0,1)$ is the capital share, and a_t is total factor productivity (TFP). Capital accumulates according to:

$$K_t = Y_t - C_t + (1-\delta) K_{t-1}, \tag{20}$$

with depreciation rate $\delta \in (0,1)$, and TFP follows an AR(1) process:

$$a_t = \rho\, a_{t-1} + \sigma_\varepsilon\, \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0,1), \tag{21}$$

with persistence $\rho \in (0,1)$ and shock standard deviation σ_ε. The equilibrium is characterised by two optimality conditions: the intratemporal labour supply condition:

$$\theta C_t = (1-\alpha)\frac{Y_t}{L_t}, \tag{22}$$

and the intertemporal Euler equation for capital:

$$K_t = \beta \, \mathbb{E}_t \left[\frac{C_t}{C_{t+1}} \left(\alpha \frac{Y_{t+1}}{K_t} + 1 - \delta \right) K_t \right]. \tag{23}$$

The model has $Y = 5$ endogenous variables $\{C_t, L_t, Y_t, K_t, a_t\}$, of which $X = 2$ are state variables $\{K, a\}$, and is driven by a single structural shock ε_t. The parameter values are set to their standard calibration: $\alpha = 0.36$, $\beta = 0.99$, $\delta = 0.025$, $\theta = 2.95$, $\rho = 0.95$, and $\sigma_\varepsilon = 0.01$.

4.2 Testing the network

We first test our proposed algorithm on a simple and relatively parsimonious deep neural network.

The policy function is approximated by a fully connected feedforward neural network (`PolicyNet`) that maps three state variables — lagged capital K_{t-1}, the lagged technology level a_{t-1}, and the current shock s_t — to onsumption C_t and labor L_t. The architecture consists of four hidden layers of 64 neurons each, with hyperbolic tangent activation functions throughout. Economic constraints are enforced directly at the output layer via `softplus` for consumption and `sigmoid` for labor, following the hard-constraint approach of Hall-Hoffarth (2023). The network is trained using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of $\lambda = 10^{-3}$ and mini-batches of 512 observations, with gradient norms clipped at unity to prevent destructive updates from extreme draws. The learning rate lies in the intermediate range identified as optimal for Adam-based training by Beck et al. (2024), who show that rates that are too high generate oscillations while rates that are too low prevent efficient convergence.[8]

Evolution of the training. Figure 4 reports the evolution of the training loss, on a logarithmic scale, across the four phases of the algorithm. Several features are worth noting. Vertical lines indicates moving from one phase to another of the algorithm.

In Phase 1 (Figure C.1 in the Appendix), the network is trained to replicate the deterministic steady state. The loss falls rapidly from above 10^1 to 10^{-6} within approximately 130 epochs, confirming that the steady state target is an easily learnable anchor for the

[8] Each step of the algorithm terminates if either the loss is below 10^{-8} or 50,000 drows are evaluated. Replication codes are available from the authors upon request.

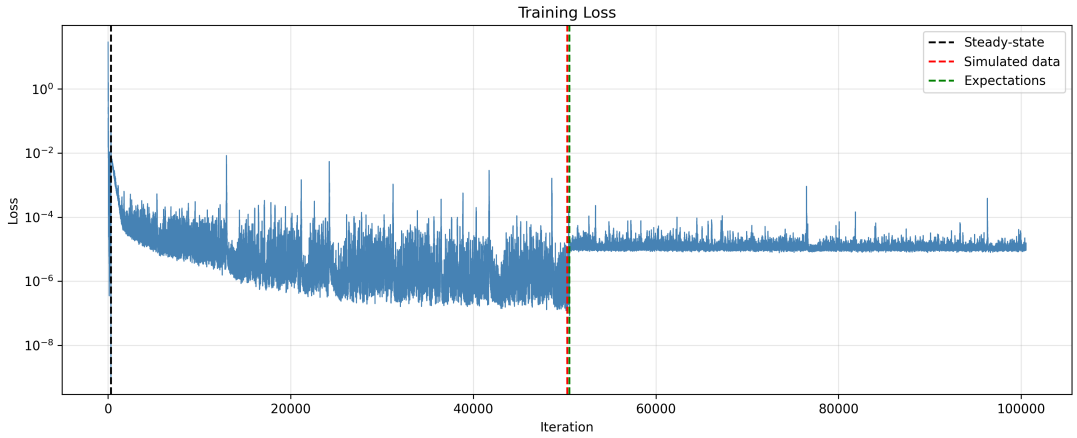

Figure 4: Training loss over the execution of the algorithm.
Notes: the figure shows the training loss over steps (Epochs) of the algorithm. Vertical lines indicates the shift from one phase to another of the algorithm.

optimizer. A transient upward jump around epoch 150 — likely reflecting a particularly unfavourable random draw of the mini-batch — is quickly corrected, and the loss ultimately reaches levels below 10^{-8} by the end of the phase, well within the convergence tolerance.

Phase 2 introduces stochastic dynamics by training on random draws around the steady state over 50,000 epochs (Figure C.2 in the Appendix). The loss drops by roughly two orders of magnitude in the first few thousand iterations and subsequently stabilises in the $[10^{-7}, 10^{-6}]$ range. The numerous sharp upward spikes visible throughout the training are a characteristic feature of stochastic gradient methods (Kingma and Ba, 2014): they reflect individual mini-batches that happen to draw state combinations far from the ergodic distribution, producing temporarily large residuals that are absorbed by the subsequent gradient steps without disrupting the overall downward trend. Gradient clipping limits the damage from these outlier draws.

Phase 3 transitions to simulated data (Figure C.3 in the Appendix). The network enters this phase already well trained, as evidenced by the loss initialising near 10^{-7}. The pronounced hump around epoch 150, where the loss temporarily rises to 10^{-4}, reflects the additional difficulty of matching policy functions along simulated trajectories, which explore regions of the state space not visited under pure random sampling. This temporary deterioration is a natural consequence of the distribution shift between training phases and is consistent with findings in Beck et al. (2024), who document that accuracy can temporarily worsen when the support of the training distribution is expanded. The

loss recovers and ends near 10^{-7}, indicating successful adaptation.

Phase 4, which incorporates rational expectations, produces the most stable loss profile (Figure C.4 in the Appendix). The loss initialises immediately at 10^{-5} and remains tightly concentrated around that level for the entire 50,000 epochs, with only occasional spikes driven by extreme shock realisations in the simulated paths. The absence of any visible downward trend confirms that the network has effectively converged at the start of this phase, and that the expectations step serves primarily to consolidate and refine the solution rather than to identify it from scratch.

The higher loss level observed in Phase 4 relative to the preceding phases is expected and does not indicate a deterioration of the solution. In the earlier phases, the loss is computed over fixed or randomly drawn state vectors, so the only source of residuals is the network's failure to satisfy the equilibrium conditions at those points. In Phase 4, by contrast, the loss incorporates expectations of future policy functions evaluated along simulated trajectories, introducing an additional layer of approximation error that is irreducible given a finite number of simulated paths. This Monte Carlo approximation of the conditional expectation operator is a fundamental source of numerical noise that sets a lower bound on the achievable loss, as discussed in Maliar et al. (2021). The stabilisation of the loss around 10^{-5} should therefore be interpreted as evidence of convergence to the best approximation attainable under the chosen simulation design, rather than as a sign of underfitting.

Comparison of the solutions. But how good is the solution approximated by the deep neural network? In our testing example, the model can be solved with standard solution packages, like Dynare. We use then the Dynare solution to benchmark the accuracy of the deep neural network in producing impulse responses, the most frequent object of interest of DSGE mdoels.

Figure 5 reports the impulse responses of capital, consumption, labor, and output to a one standard deviation TFP shock, comparing the Dynare (Adjemian et al., 2026) perfect-foresight solution (red dots) with the deep neural network solution from Phase 3, which abstracts from uncertainty (black dashed line), and the full deep neural network solution from Phase 4, which incorporates rational expectations over future shocks (blue solid line).

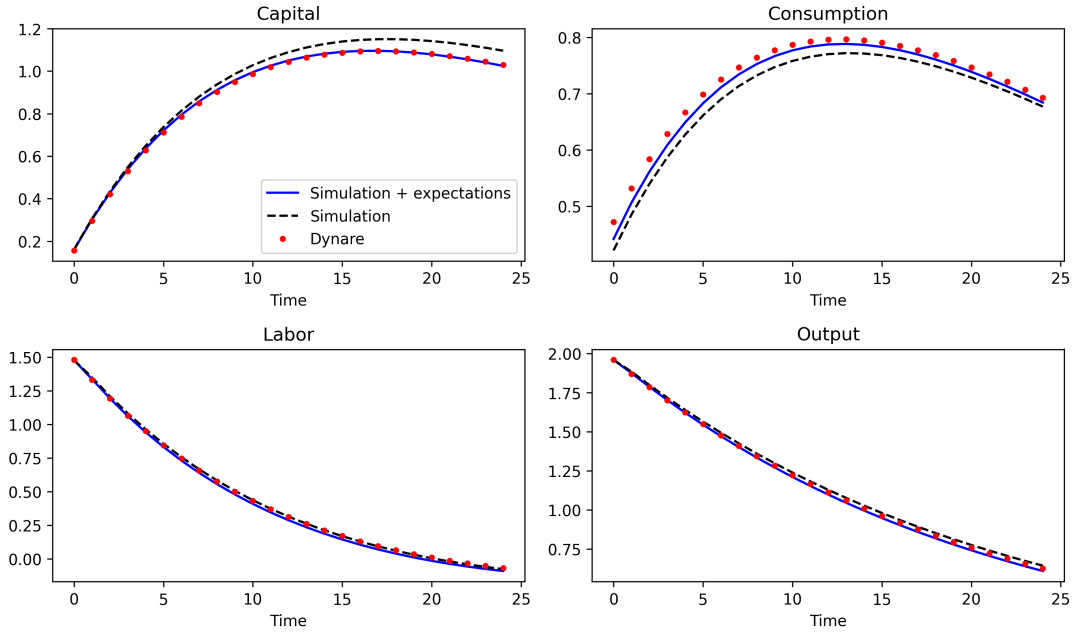

Figure 5: Impulse responses to a one standard deviation TFP shock.
Notes: the figure shows the impulse responses of endogenous variables to 1 standard deviation TFP shocks. The red dots show impulse responses computed under Dynare's perfect foresight algorithm, the black dashed line are responses computed with the deep neural network trained in Phase 3 and the blue line are responses under the fully trained neural network, Phase 4. Impulse responses are reported in percent deviation from the deterministic steady state.

Across all variables, the three solutions are broadly consistent, confirming that the network successfully captures the qualitative dynamics of the model. The most economically meaningful discrepancy, however, emerges in consumption. The Dynare perfect-foresight algorithm computes the deterministic transition path conditional on a known, fixed shock sequence: although forward-looking, agents face no future uncertainty by assumption, and therefore have no precautionary motive, making it a deterministic benchmark rather than a stochastic rational-expectations equilibrium. The DNN solution with expectations, by contrast, trains the network to satisfy the stochastic Euler equation, integrating over the distribution of future shocks at each point in time. As a result, forward-looking households internalize the risk of adverse future realisations and optimally choose to consume less and save more at each date, producing the lower consumption path visible in the blue line relative to the red dots. This wedge is precisely the precautionary savings premium induced by uncertainty, (i.e. Carroll (1997)), and its presence validates the importance of Phase 4: a solution that ignores expectations, such as the Phase 3 black dashed line, fails to correctly account for this channel. The differences in capital, labor, and output are comparatively minor and reflect the indirect general equilibrium consequences of the

consumption adjustment.

4.3 Sensitivity to the neural network's parameters

To assess the sensitivity of the network to architectural and training choices, we re-train the model under a grid of alternative specifications, varying three key design parameters: the number of hidden layers, the number of neurons per layer, and the weight assigned to the steady state loss during the dynamic training phases. The number of hidden layers — set to 2, 3, 5, or 7 — governs the expressive capacity of the network and its ability to approximate complex, non-linear policy functions; deeper architectures can in principle capture richer dynamics but are also more prone to vanishing gradients and overfitting, and may require more epochs to converge (Fernández-Villaverde, 2025). The number of neurons per layer — 32, 64, or 128 — determines the width of the network and controls the trade-off between approximation accuracy and computational cost, with wider networks offering greater flexibility at the expense of a larger parameter space to optimise. Finally, the steady state penalty weight — set to 0.5, 1, or 5 — regulates how strongly the network is anchored to the deterministic steady state during the learning of dynamics: a weight that is too low allows the network to drift away from a well-understood reference point, potentially destabilising training, while a weight that is too high may prevent the optimizer from exploring the stochastic dynamics of the model sufficiently, biasing the solution towards the non-stochastic equilibrium (Maliar et al., 2021).

Effect of network depth. Table 1 reveals a non-monotonic and, at the extreme, catastrophic relationship between the number of hidden layers and solution accuracy. Among 32-neuron networks trained with a penalty weight of unity, the shallowest architecture (2 layers) achieves the lowest final expectations loss of 9×10^{-6}, while the 3-layer network reaches 1.1×10^{-5} and the 5-layer architecture produces the highest loss of 1.5×10^{-5}, exhausting the full 50,000-epoch budget in the random dynamics phase without converging. The introduction of a 7-layer architecture sharpens this finding. Among 64-neuron networks, the two lowest penalty weights fail to produce any convergent solution at all, while the weight-5 specification yields a final loss of 13.3 — several orders of magnitude larger than any other configuration — indicating a complete breakdown of training rather than a marginal deterioration. Several 128-neuron, 7-layer specifications similarly

Table 1: Comparison across network specifications

Neurons	steady state weight	Epochs Phase 1	Epochs Phase 2	Epochs Phase 3	Epochs Phase 4	steady state ate loss	Final loss
2 hidden layers architecture							
32	0.5	129	38978	1	50000	8.95E-08	1.15E-05
32	1	129	49652	1	50000	8.95E-08	9.43E-06
32	5	129	50000	121	50000	8.95E-08	1.04E-05
64	0.5	159	36239	136	50000	1.96E-08	1.15E-05
64	1	159	43178	1	50000	1.96E-08	1.2E-05
64	5	159	50000	66	50000	1.96E-08	1.51E-05
128	0.5	1667	26375	1	212	8.58E-09	9.55E-06
128	1	1667	29619	1	1	8.58E-09	9.8E-06
128	5	1667	44728	1	2	8.58E-09	9.39E-06
3 hidden layers architecture							
32	0.5	142	35448	1	50000	7.68E-08	1.24E-05
32	1	142	39328	199	50000	7.68E-08	1.1E-05
32	5	142	50000	155	50000	7.68E-08	1.09E-05
64	0.5	144	38702	1	50000	8.33E-08	1.28E-05
64	1	144	47871	1	50000	8.33E-08	1.1E-05
64	5	144	50000	62	50000	8.33E-08	1.1E-05
128	0.5	632	21171	1	1	7.22E-09	9.23E-06
128	1	632	30224	1	2	7.22E-09	9.32E-06
128	5	632	50000	181	1	7.22E-09	9.99E-06
5 hidden layers architecture							
32	0.5	147	49923	1	50000	3.08E-08	1.12E-05
32	1	147	50000	166	50000	3.08E-08	1.47E-05
32	5	147	50000	257	50000	3.08E-08	9.79E-06
64	0.5	140	42623	1	50000	1.87E-08	1.24E-05
64	1	140	50000	394	50000	1.87E-08	1.14E-05
64	5	140	50000	345	50000	1.87E-08	1.09E-05
128	0.5	335	34222	1	3	8.41E-08	9.74E-06
128	1	335	48133	1	1	8.41E-08	8.01E-06
128	5	335	50000	179	1	8.41E-08	9.96E-06
7 hidden layers architecture							
32	0.5	163	47921	1	1	5.1E-08	9.63E-06
32	1	163	50000	214	1	5.1E-08	9.96E-06
32	5	163	50000	352	3	5.1E-08	8.81E-06
64	0.5						
64	1						
64	5	326	50000	53	4	6.22E-08	13.31288
128	0.5	396	50000	74	0	7.66E-08	
128	1	396	50000	232	3	7.66E-08	9.96E-06
128	5	396	50000	244	1	7.66E-08	9.86E-06

Notes: The table reports initial and final losses for alternative network configurations in terms of hidden layers, neurons and steady state loss weight. Empty cells indicate that the algorithm has not converged. All other parameters are kept as in Section 4.2.

fail to converge. This result strongly reinforces the observation of Beck et al. (2024) that, for models of moderate complexity, deeper architectures do not necessarily outperform shallower ones and may instead introduce optimisation difficulties that slow, impair, or entirely prevent convergence.

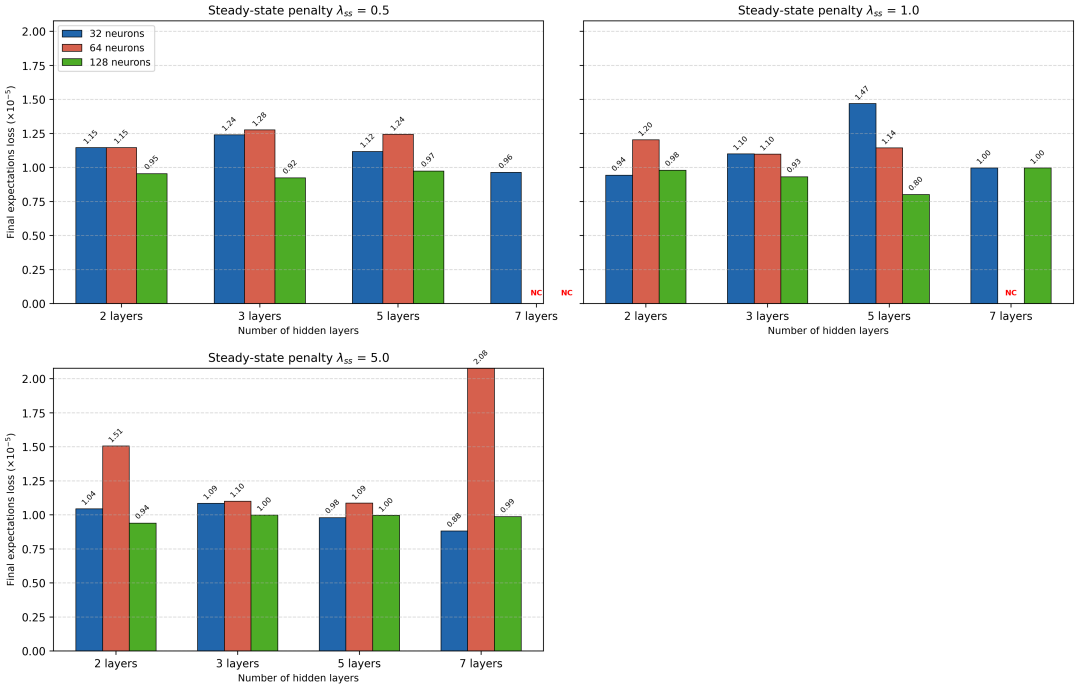

Figure 6: Comparison of penalty across number of neurons.
Notes: Each panel reports the final expectations loss (scaled by 10^{-5}) across network architectures for a given steady state penalty weight $\lambda_{ss} \in \{0.5, 1.0, 5.0\}$; within each panel, bars are grouped by the number of hidden layers and colours distinguish networks with 32 (blue), 64 (red) or 128 (green) neurons per layer. A lower value indicates a more accurate approximation of the policy function at the end of Phase 4 training.

Effect of network width. Increasing the number of neurons per layer from 32 to 64 has a comparatively modest effect on final solution accuracy, but expanding to 128 neurons reveals a more nuanced picture. Across all layer-weight combinations, the expectations loss is broadly similar between 32 and 64 neurons. The 128-neuron networks achieve comparable or marginally better final losses in several configurations — including the overall best result in the grid, 8×10^{-6}, obtained with 5 layers and $\lambda_{ss} = 1$ — but at a substantial cost to Phase 1 training: convergence to the steady state requires up to 1,667 epochs for 128-neuron, 2-layer networks, compared to 129 epochs for their 32-neuron counterparts, reflecting the much larger parameter space the optimizer must navigate before dynamic training can begin. Notably, once Phase 1 is complete, wider networks tend to reach convergence in Phases 3 and 4 in very few epochs, suggesting that the additional capacity accelerates the learning of dynamics once the steady state anchor is established, but imposes a heavy upfront cost. These findings echo Beck et al. (2024), who document that gains from increasing network width are marginal beyond a modest threshold for standard DSGE models. Figure 6 illustrates this by comparing final losses

across neuron counts, grouped by number of layers.

Effect of the steady state penalty weight. The penalty weight λ_{ss} assigned to the steady state loss during the dynamic training phases has a meaningful impact on the speed and stability of convergence, though its effect on the final expectations loss is more contained for well-behaved architectures. Fixing the architecture at 2 layers and 32 neurons, the intermediate weight $\lambda_{ss} = 1$ achieves the best final loss (9×10^{-6}) and allows the random dynamics phase to progress efficiently. A low weight of 0.5 accelerates convergence in the random phase (completing in approximately 39,000 epochs) but yields a marginally higher final loss, suggesting the network drifts away from the steady state anchor too quickly. A high weight of 5 systematically exhausts the maximum epoch budget in the random phase across nearly all architectures, indicating that over-penalising deviations from steady state prevents the optimizer from exploring the stochastic dynamics of the model with sufficient freedom. The interaction between the penalty weight and network depth becomes especially consequential for the 7-layer architecture: at this depth, low and intermediate penalty weights produce convergence failures across multiple neuron counts, while even the high-penalty specification can yield catastrophically large final losses or non-convergence, suggesting that the steady state anchor is insufficient to stabilise training when the network is excessively deep. These results suggest that $\lambda_{ss} = 1$ represents a robust default choice for architectures of moderate depth. Figure 6 presents a heatmap of final expectations loss over the grid of layer counts and penalty weights, separately for each neuron count.

The robustness analysis points to a clear set of recommendations for network design. On depth, shallower architectures are preferable: two or three hidden layers consistently deliver the lowest expectations loss, while five-layer networks offer no systematic accuracy gain and seven-layer networks frequently fail to converge entirely, with some configurations producing catastrophically large residuals. Additional depth introduces optimisation difficulties that outweigh any representational benefit for models of this complexity. On width, 32 neurons per layer perform at least as well as 64 neurons at lower computational cost; 128 neurons can marginally improve final accuracy in some configurations but impose a substantially larger overhead in Phase 1 training, making them unlikely to be cost-effective in practice. On the steady state penalty, an intermediate weight of $\lambda_{ss} = 1$ is

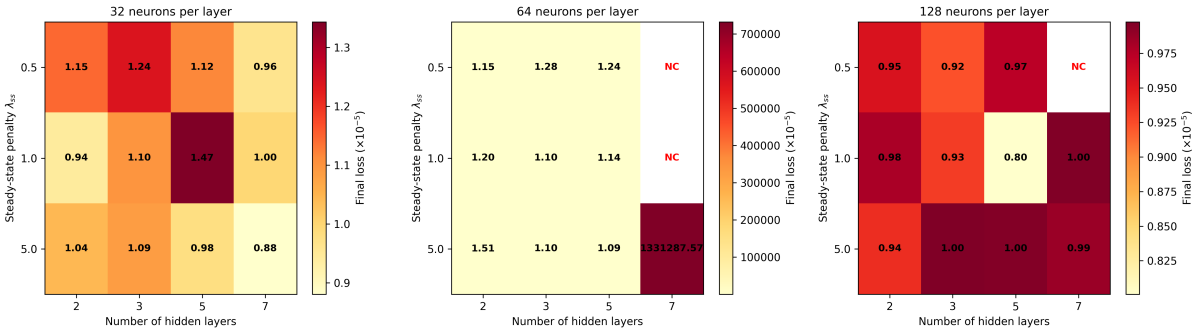

Figure 7: Comparison of penalty across weight for the steady state loss.
Notes: Each panel reports the final expectations loss (scaled by 10^{-5}) over the grid of hidden layers (x-axis) and steady state penalty weights (y-axis) for networks with 32, 64 and 128 neurons per layer respectively; cells are colour-coded from yellow (low loss) to red (high loss), with numeric values annotated within each cell. Darker shading identifies architecture-penalty combinations associated with higher approximation error at the end of Phase 4 training. Empty cells denote configurations that failed to converge.

the recommended default: it provides sufficient anchoring without over-constraining the optimizer, a risk that becomes more severe as network depth increases. Taken together, these findings confirm that a parsimonious architecture — shallow, moderately wide, and moderately anchored to the steady state — dominates more complex alternatives on both accuracy and computational grounds.

5 Application to tariffs in an open economy model

This section applies the deep learning algorithm to solve a two-country New Keynesian model with large tariff shocks, illustrating both the economic relevance and the methodological advantages of the approach. Tariff shocks of the magnitude observed in recent episodes of trade policy disruption are inherently large and generate non-linear interactions between relative prices, demand composition, and monetary policy that local perturbation methods cannot capture, as Taylor series approximations deteriorate rapidly away from the steady state (Fernández-Villaverde, 2025). The asymmetric and state-dependent responses that are central to the policy debate — including the differential impact on the tariff-imposing versus the recipient country — are precisely the features that linearisation obscures. Global projection methods can in principle recover these non-linearities, but suffer from the curse of dimensionality and become computationally prohibitive for models of the scale required for quantitative policy analysis (Hall-Hoffarth, 2023). The deep learning approach discussed in Section 3 resolves this tension by training a neural

network to satisfy the equilibrium conditions globally over the ergodic distribution of states, combining the accuracy of projection methods with the scalability of simulation-based techniques.[9] Our application to a medium-scale open-economy model with capital accumulation, nominal rigidities, and endogenous monetary policy further demonstrates that the method extends naturally beyond the stylised environments in which it has most commonly been demonstrated.

The model we use builds on Benigno (2009) and consists of two symmetric countries of equal size, which we refer to as Home and Foreign. Each country is populated by a representative household that maximises lifetime utility over consumption and labour, subject to an intertemporal budget constraint. Households save through holdings of domestic bonds and invest in physical capital, which is subject to convex capital adjustment costs. Departing from the original formulation, we adopt a capital-based specification for adjustment costs following Hayashi (1982), whereby the cost is proportional to the deviation of the investment-to-capital ratio from its steady state level. Under this specification, Tobin's q can be expressed as an analytical function of current investment and the existing capital stock, thereby eliminating the need to solve a separate forward-looking optimality condition for the shadow price of capital and reducing the dimensionality of the problem.[10] In each country, a continuum of monopolistically competitive firms produces differentiated intermediate goods using capital and labour under a Cobb–Douglas technology subject to country-specific total factor productivity shocks, and sets prices subject to quadratic Rotemberg adjustment costs, giving rise to a standard New Keynesian Phillips curve. Households have home bias in consumption and investment, with the price index in each country aggregating domestic and imported goods through a constant elasticity of substitution bundle. International financial markets are complete in the sense that the uncovered interest parity condition holds, so that the real exchange rate is

[9] In this case the implementation of the algorithm departs from Section 3 in three minor respects. First, during the steady state learning phase we sample states progressively further from the steady state, starting within a 5% neighbourhood and gradually widening to 20%, which prevents the network from overfitting a narrow region before it has learned the correct steady state values. Second, we allow up to 200,000 training epochs to ensure adequate convergence in a model with a larger state space than those typically considered in the literature. Third, we employ a learning rate scheduler that halves the learning rate whenever the loss falls below 10^{-7} and fails to improve over 2,000 consecutive epochs, which accelerates convergence in the final stages of training when coarse gradient steps would otherwise cause the loss to oscillate rather than descend. We use 5 layers, 64 neurons, `Tanh` activation function, `sigmoind` for labor and `softplus` for other variables regularisation functions are applied to enforce economic sensible constrains on the variables.

[10] This avoids the lengthy formulation of the Tobin's Q under standard quadratic adjustment costs for investment that may generate computational issues when training the model.

determined by the ratio of the marginal utilities of consumption across countries. Each country's monetary authority follows an inertial Taylor rule that responds to domestic inflation and output relative to their respective steady state values. A key departure from Benigno (2009) is the introduction of an import tariff levied by the Home country on goods imported from the Foreign country, which drives a wedge between the border price and the consumer price of imported goods and alters the composition of domestic demand. The tariff follows an exogenous autoregressive process, allowing us to trace out the non-linear general equilibrium effects of tariff shocks on output, inflation, and the real exchange rate in both countries.[11] Losses from the four stages are reported in Figure C.5 in the Appendix. After converging relatively slow to a dynamic solution in Phase 2, the model learns quicker to solve under simulated data and uncertainty.

Figure 8 reports the impulse responses to a one-percent increase in the effective tariff rate imposed by the Home country on imports from the Foreign country. The tariff follows an autoregressive process with persistence parameter $\rho_\tau = 0.5$, and as shown in the upper left panel it dissipates within roughly ten quarters. Despite the limited persistence of the shock itself, its general equilibrium effects are quantitatively significant and asymmetric across countries — a transmission pattern central to the current policy debate and one that a linearised solution would be ill-equipped to capture.

The upper right panel shows that the nominal exchange rate of the Home country appreciates against the Foreign currency on impact, by approximately 1.5 percent, with the appreciation moderating temporarily over the first year before progressively deepening to nearly 2.7 percent by the end of the simulation horizon. This pattern aligns with the standard open-economy New Keynesian prediction analysed by Lindé and Pescatori (2019) that an import tariff appreciates both the real and nominal exchange rates in the imposing country, and with the panel evidence on real appreciation following tariff increases in Furceri et al. (2022). Ostry et al. (2025) provide complementary event-study evidence that the US dollar appreciates around unilateral tariff announcements, with the sign of the response reversing only when retaliation is anticipated. The non-monotonic profile in the figure — a sharp initial appreciation, partial reversal, and renewed appreciation as adjustment unfolds — reflects the interaction between the immediate financial-market response and the slower endogenous repricing of imported goods, and is itself a non-linear

[11] The model is described in more details in Section B which also reports the calibration used.

feature that a first-order perturbation solution would miss by construction.

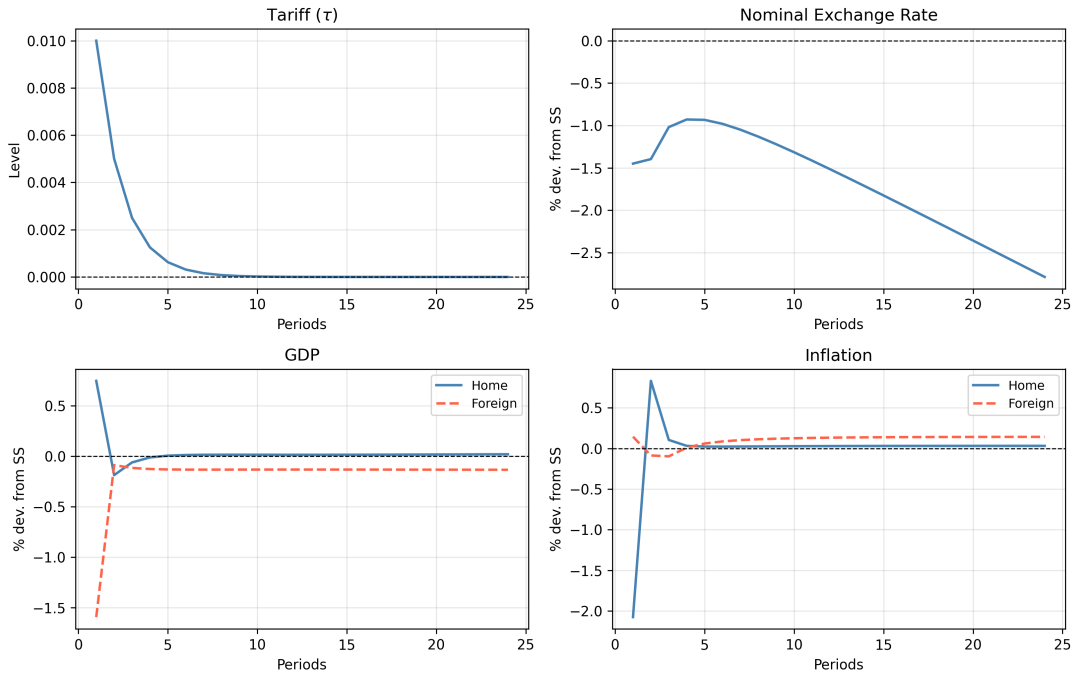

Figure 8: Impulse response to a one-percent tariff shock.
Notes: The chart shows the impulse responses to a one-percent increase in the effective tariff rate imposed by the Home country on imports from the Foreign country. A negative deviation of the nominal exchange rate denotes an appreciation of the Home currency. The model is solved with the algorithm described in Section 3.

The lower left panel reveals a stark asymmetry in the GDP responses across the two countries. Home output expands sharply on impact, by approximately 0.75 percent, as domestic households substitute away from now-tariffed Foreign goods toward Home-produced varieties. The expansion is short-lived: by the second quarter the substitution channel is dominated by the general equilibrium contraction of aggregate demand, and Home output dips slightly below steady state before recovering gradually toward zero. The Foreign country bears the larger contractionary burden, with output falling by approximately 1.6 percent on impact and stabilising around 0.15 percent below steady state for the remainder of the horizon. The Foreign decline reflects the loss of export demand to the Home market driven by the tariff wedge, which is only partially cushioned by the contemporaneous depreciation of the Foreign currency that makes Foreign varieties more competitive once exchange-rate pass-through takes hold. This medium-term contraction is broadly consistent with the cross-country empirical evidence on the output costs of tariffs in Furceri et al. (2022) and with the price-rigidity-driven dynamics emphasised theoretically by Lindé and Pescatori (2019).

The lower right panel illustrates the inflation dynamics, which are particularly informative about the role of nominal rigidities in this environment. Home inflation falls sharply on impact, by approximately two percent, before reversing to roughly 0.8 percent in the second quarter and stabilising near steady state thereafter. The initial deflation reflects the interaction of price stickiness with the strong exchange-rate appreciation: imported goods become cheaper in Home currency, and because posted prices cannot fully adjust contemporaneously, this exchange-rate channel temporarily dominates the direct cost-push effect of the tariff and pulls the consumer price index below steady state. As stickiness wears off in the following quarter, prices update to incorporate the direct pass-through of the tariff into the import component of the consumption basket and inflation jumps to a positive value. Foreign inflation, by contrast, exhibits only a mild and persistent rise of roughly 0.2 percent, consistent with the gradual pass-through of the Foreign currency depreciation into Foreign tradable prices. The interplay between sticky prices, exchange-rate adjustment, and the tariff wedge is closely analogous to the mechanism stressed by Lindé and Pescatori (2019) in their analysis of border-adjustment shocks.

Taken together, these responses highlight several intrinsically non-linear features that local solution methods would mischaracterise. The non-monotonic exchange-rate path, the sign reversal in Home output, and the abrupt swing between deflation and inflation in the impact period all reflect interactions between substitution effects, price rigidities, and exchange-rate adjustment that depend non-linearly on the size of the shock. These properties jointly motivate the global, deep-learning solution approach of Maliar et al. (2021), which approximates the policy functions over the full ergodic distribution of the state space without imposing the symmetry and proportionality of a perturbation solution.

5.1 Small versus large tariff shocks

Figure 9 compares the impulse responses of the Home economy to a small and a large tariff shock — a one-percent increase in the effective tariff rate (the "1σ" line) and a ten-percent increase (the "10σ" line) — under the same autoregressive process for the tariff. If the equilibrium conditions of the model were linear, the responses of every endogenous variable would simply be proportional, and the two lines would lie on top of each other up to a factor of ten. The figure makes clear that this is far from being the case: not only

do magnitudes diverge sharply across the two scales, but several variables change sign between the two scenarios. These qualitative differences are the signature of non-linear propagation, and they motivate the global solution method adopted in this paper.

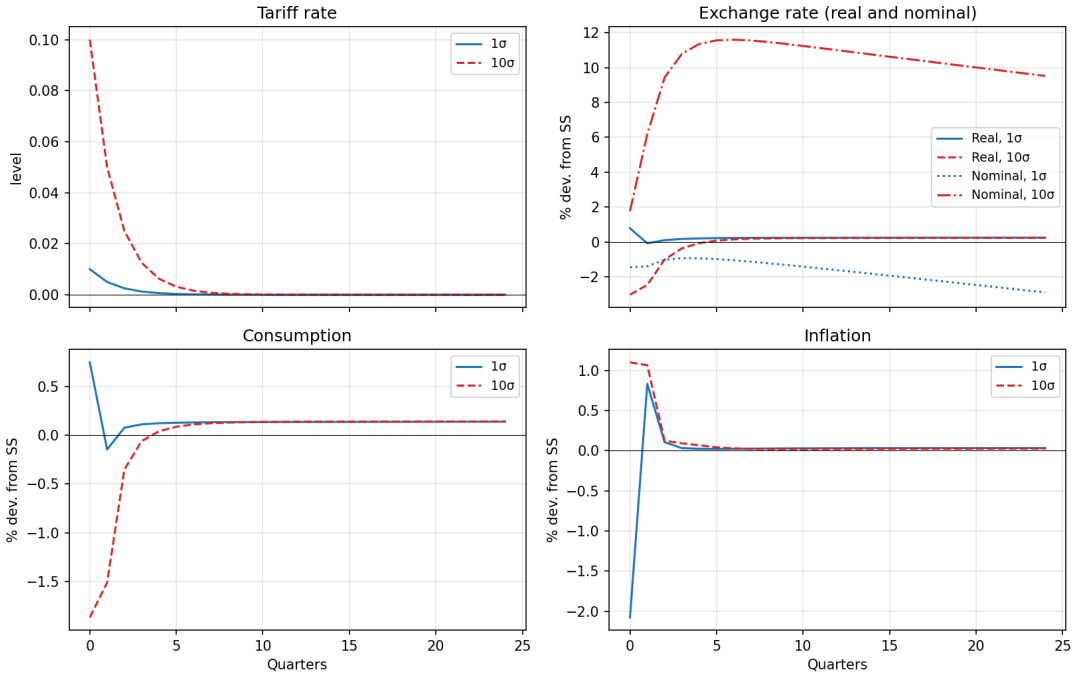

Figure 9: Impulse response to a one-percent and ten-percent tariff shocks.
Notes: The chart shows the impulse responses to a one-percent increase in the effective tariff rate imposed by the Home country on imports from the Foreign country. A negative deviation of the nominal exchange rate denotes an appreciation of the Home currency. The model is solved with the algorithm described in Section 3.

The sign reversal is most striking in the upper right panel, which plots both the nominal and real exchange rates. Following the small shock, the nominal exchange rate behaves in line with textbook intuition: it appreciates on impact by approximately 1.5 percent and deepens its appreciation over the simulation horizon, consistent with Lindé and Pescatori (2019) and Furceri et al. (2022). Following the large shock, by contrast, the nominal exchange rate *depreciates* sharply, by roughly two percent on impact and by nearly twelve percent at the peak around the second year, before gradually unwinding. The real exchange rate exhibits an analogous but more muted reversal, briefly appreciating on impact under the large shock before settling close to steady state. This sign flip mirrors the empirical evidence of Ostry et al. (2025), who document that the US dollar depreciates around tariff announcements when retaliation is anticipated. They rationalise the result with the mechanism of Bergin and Corsetti (2023). In our model, this channel takes the form of a strongly counter-cyclical Taylor-rule response to the

deep contraction in domestic consumption discussed below; under the small shock, the substitution channel and the standard demand-shift channel dominate and the currency appreciates, whereas under the large shock the contractionary effects of the tariff and the resulting monetary-easing channel become quantitatively pre-eminent and the currency depreciates.

The lower left panel reveals an analogous sign reversal in domestic consumption. Under the small shock, consumption rises on impact by about 0.75 percent as households substitute toward Home-produced varieties, briefly dips below steady state in the second quarter, and converges to a modestly positive long-run level of approximately 0.13 percent. Under the large shock, the substitution motive is overwhelmed by the income and relative-price losses associated with the tariff: consumption falls by roughly 1.85 percent on impact and recovers only gradually toward the same long-run level. The fact that the long-run endpoint is essentially identical under the two scenarios — while the impact response differs in sign and is more than two orders of magnitude larger in absolute value under the large shock — is a prototypical non-linearity that no first-order solution could reproduce.

The lower right panel shows the corresponding reversal in inflation. Under the small shock, the appreciation of the nominal exchange rate combined with sticky prices generates an initial deflation of approximately two percent before inflation rebounds in the following quarter. Under the large shock, by contrast, the dominant force is the direct cost-push transmission of the tariff itself together with the nominal depreciation, and inflation rises on impact by roughly 1.1 percent before converging back toward steady state. As with consumption, the sign of the impact response flips between the two scenarios. This asymmetry is also a natural feature of Rotemberg-style price adjustment: as the desired-price gap widens, the marginal benefit of repricing grows while the marginal cost remains bounded by the convex adjustment penalty, so that foreign exporters find it optimal to update their prices on impact rather than gradually whenever the shock is large enough — generating the immediate pass-through of the tariff into Home inflation that is absent under the small shock.

These three reversals would be invisible under a perturbation solution, which would by construction generate proportional and sign-preserving responses in the size of the shock. They are precisely the type of state-dependent and shock-size-dependent dynamics that

the deep-learning approximation of Maliar et al. (2021) is designed to capture, and they constitute one of the main practical benefits of the methodology when used to evaluate trade policy counterfactuals of empirically realistic magnitudes.

5.2 Comparison with a second-order perturbation solution

A natural benchmark against which to assess the value added of the deep-learning solution is a perturbation method computed at second order. Second-order solution capture some degrees of non-linearities through squared and interaction terms, while being substantially less computational demanding than global methods. It makes therefore sense to compare accuracy gains by the deep neural network against this substantially simpler solution moethod. We solve the same model in `Dynare` (Adjemian et al., 2026) using the algorithm of Schmitt-Grohé and Uribe (2004), which approximates the equilibrium policy functions by a second-order Taylor expansion around the deterministic steady state. At first order, the policy functions are linear in the state variables and the impulse responses are exactly proportional to the size of the shock; the second-order term is the minimum non-trivial enrichment of this benchmark, introducing curvature in the policy functions and a wedge between the deterministic and the stochastic steady state. The approximation is therefore able in principle to deliver shock-size dependence, but only of a kind generated by the leading quadratic term: it is by construction local, and its accuracy deteriorates rapidly as the state moves away from the steady state.

Figure 10 reports the difference between the impulse response under a ten-standard-deviation tariff shock and a one-standard-deviation shock — henceforth the "large-versus-small gap" — for consumption, inflation, and the real exchange rate, under both solution methods. Plotting the gap directly, rather than the two impulse responses separately, isolates the part of the dynamics that is driven by the curvature of the policy function: under a fully linear solution the gap would simply equal nine times the small-shock response, and any deviation from this scaling reflects the non-linearity that each method is able to detect.

For the three variables shown in the figure, the picture that emerges is qualitatively reassuring but quantitatively striking. The two methods agree on the sign and on the broad temporal pattern of the gap: in both cases consumption and the real exchange rate display a substantially amplified contraction at the larger shock, and inflation a markedly

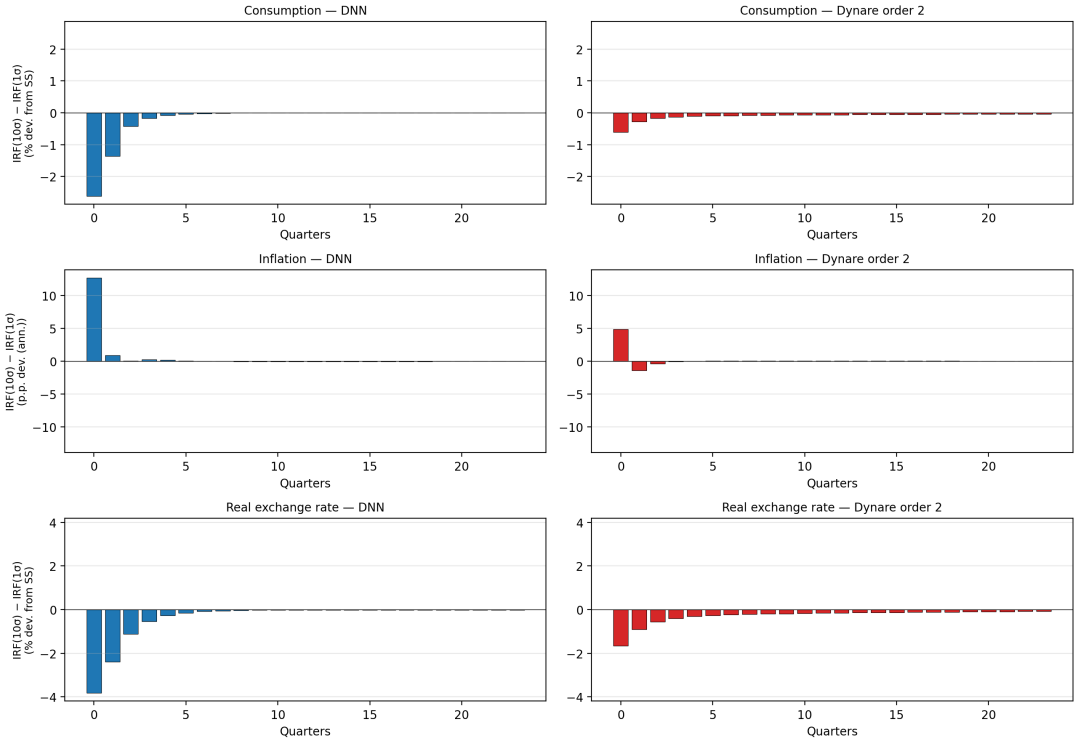

Figure 10: Comparison between deep neural network and Dynare second-order solutions. *Notes*: The chart shows the difference in the impulse responses of selected variables between the deep neural network solution and the second order perturbation solution of Dynare. Bars show the difference in impulse responses between a 1 standard deviation and a 10 stanard deviation shock. The model is solved with the algorithm described in Section 3.

stronger response on impact. The second-order perturbation, however, systematically and severely understates the magnitude of the non-linearity. Across the three variables the impact gap implied by the deep neural network is between roughly two and three times larger in absolute value than the one delivered by perturbation, and the discrepancy persists for several quarters. For these variables, then, the second-order solution preserves the qualitative direction of the curvature in the model but compresses its quantitative significance.

This relatively benign characterisation, however, does not extend to all margins of adjustment. As shown in Figure C.6 in the Appendix, the non-linearity in the nominal exchange rate response is qualitatively beyond the reach of the second-order solution. The deep neural network delivers the sign reversal documented in Section 5.1: the nominal exchange rate appreciates under a small tariff shock but *depreciates* sharply under a large one, in line with the mechanism of Bergin and Corsetti (2023) and the empirical evidence in Ostry et al. (2025). The second-order perturbation, by contrast, simply scales up the appreciation observed at the small shock without reversing its sign on impact. This is a

failure not of magnitude but of direction, and it is precisely the kind of state-dependent regime change that a fixed-order Taylor expansion around the deterministic steady state cannot accommodate.

This pattern is what the underlying mathematics leads one to expect. The non-linear features that drive the dynamics analysed in Section 5.1 — the kink in the Rotemberg pricing decision that arises when the desired-price gap becomes large enough to outweigh the convex adjustment cost, the strongly state-dependent reaction of monetary policy to deeper consumption contractions, and the non-linear interaction between exchange-rate adjustment and the tariff wedge — involve cumulative powers of the state variables higher than two and are therefore truncated away by a second-order Taylor expansion. The deep neural network, by contrast, does not commit to any particular polynomial order: it approximates the policy functions globally over the ergodic distribution of the state space, and so absorbs the relevant higher-order curvature automatically.

The practical implication is that perturbation solutions of standard order can be a useful but unreliable guide to the macroeconomic consequences of large trade shocks. For some variables they preserve the sign of the non-linearity at the cost of compressing its magnitude; for others, such as the nominal exchange rate, they miss the sign reversal entirely. This combination of quantitative under-statement and occasional qualitative failure is the central reason why the global, deep-learning solution method of Maliar et al. (2021) is the appropriate tool for the questions addressed in this paper, and it constitutes the main methodological pay-off of the approach for applied trade-policy analysis.

5.3 Practical considerations for larger models

The implementation of the two country model can be instructive in highlighting a number of practical considerations that become increasingly relevant as the dimension of the state space grows. We collect them here as concise guidance for researchers seeking to apply this class of methods to richer environments.

Network outputs in deviation form. While it is conceptually natural to have the network return the levels of the controls, we find that specifying the outputs as deviations from the deterministic steady state consistently improves convergence. Working in deviations centres the target around zero, reduces the heterogeneity in scale across controls,

and naturally embeds the steady state into the architecture: at the deterministic steady state the network is required to return zero, an anchor that is straightforward to impose and that does not require additional penalty terms once the early phases of training are completed.

Adequate length of the random-states phase. Simulation-based training in Phases 3 and 4 implicitly assumes that the network already provides a reasonable approximation to the policy functions, since the simulated trajectories are themselves generated by the network being trained. It is therefore important to allow Phase 2 to run long enough for the loss to stabilise under random-state sampling before moving on. Premature progression to simulated data feeds the network with trajectories that are still substantially mis-specified, amplifying approximation errors and often slowing, or destabilising, convergence in the subsequent phases.

Selective approximation of controls. Not every endogenous variable needs to be approximated by a dedicated output of the network. Many controls are deterministic convolutions of other variables: given the capital stock and hours worked, the production function pins down output; given output, capital and labour, factor prices can be recovered from the firms' first-order conditions. Limiting the network outputs to a parsimonious set of genuinely forward-looking variables, and recovering the remaining quantities analytically from the equilibrium conditions, tends to deliver a more accurate and computationally lighter solution. The selection is intrinsically an economic question, requiring an understanding of the structure of the model rather than a purely numerical criterion: the variables chosen as network outputs should be those whose dynamics are genuinely forward-looking, and not those that can be inferred contemporaneously from the state and from the other controls.

Numerical robustness of the loss function. Algebraically equivalent reformulations of the equilibrium conditions can behave very differently in numerical implementation. Divisions by endogenous variables that may take small values, negative powers of quantities not bounded away from zero, or logarithms of variables that occasionally cross into non-positive territory can each produce `NaN` or infinite values for an unlucky draw of the random states, particularly in the early phases of training when the network has not

yet learned to keep variables within their economically meaningful range. Such isolated breakdowns propagate through the gradient computation and can corrupt the parameter vector, leaving training stuck at a degenerate solution from which it cannot recover. We find it useful to rewrite the model equations in forms that are robust to near-zero realisations, to clip the gradient norm during backpropagation, and to incorporate safeguards that detect pathological draws and, when necessary, restore the network from the most recent checkpoint, in line with the design recommendations of Beck et al. (2024).

Training discipline in large state spaces. As the model size grows, the loss surface becomes noisier and the marginal return to additional gradient steps falls rapidly, so the training schedule needs to be calibrated with a sober view of the computational cost. The convergence threshold can profitably be relaxed in the more advanced training phases, since the accuracy attainable under steady state-only training is rarely matched once Monte Carlo sampling and rational expectations are introduced. The base learning rate should likewise be reduced in the later phases, possibly via a scheduler that halves the step size whenever the loss fails to improve over a long window of epochs, so that coarse gradient steps do not cause the loss to oscillate rather than descend. Finally, in models with many shocks it is prudent to checkpoint the network at the end of each phase, so that training can revert to the most recent stable state should an unlucky draw produce `NaN` or infinite values capable of corrupting the parameter vector.

6 Conclusion

This paper develops and evaluates a deep learning method for solving dynamic stochastic general equilibrium models. Our approach parameterizes the model's policy functions as deep neural networks and trains them to satisfy the equilibrium conditions of the model over a stochastically simulated state space. We contribute to this literature in four dimensions.

Our first and principal methodological contribution is a sequential training algorithm composed of four phases, each building on the solution obtained in the preceding one. The algorithm begins by anchoring the network at the deterministic steady state of the model — a fixed point that is available analytically and requires no prior knowledge of the dynamic solution. This anchoring step ensures that the network encodes economi-

cally plausible behaviour before any dynamic training begins, preventing the optimizer from exploring regions of the state space that are never visited in equilibrium. The algorithm then progressively introduces dynamics, first through random exploration around the steady state, then through simulation on the model's ergodic set, and finally through Monte Carlo integration over future shocks to enforce the forward-looking Euler equations. A steady state penalty, maintained throughout all phases with weight λ_{ss}, plays a stabilising role analogous to regularization: it prevents the optimizer from drifting toward solutions that, while locally minimising the Euler equation residuals, are inconsistent with the model's long-run equilibrium.

Our second contribution addresses a fundamental circularity in applying deep learning to DSGE models: the training data must be generated by simulating the model under the current approximation of the policy function, yet that policy function is precisely the object one seeks to estimate. Rather than initialising the network from a linearised solution — which would bias training toward the ergodic set implied by the linear approximation — or restricting training to a fixed rectangular grid, our staggered approach constructs its own training data progressively. The network begins from the analytically known steady state, requiring no prior knowledge of the dynamic solution whatsoever, and the simulated trajectories used for training converge to the model's true ergodic set as the approximation improves. This self-referential procedure ensures that computational effort is concentrated where the solution is economically relevant, without any auxiliary model or pre-computed starting values.

Our third contribution is a systematic empirical exploration of the network architecture and hyperparameter space. Using the canonical real business cycle model of Kydland and Prescott (1982) as a controlled testing environment, we vary the number of hidden layers (2, 3, 5, 7), the number of neurons per layer (32, 64, 128), and the weight on the steady state penalty (0.5, 1, 5), running the full four-phase algorithm for each specification. The results provide clear practical guidance for applied researchers: shallow architectures with two or three hidden layers, moderate width of 32 neurons per layer, and an intermediate steady state penalty weight of $\lambda_{ss} = 1$ consistently deliver the best accuracy at the lowest computational cost, with deeper or wider networks offering no systematic improvement and a high penalty weight impeding convergence by over-constraining the optimizer during the dynamic training phases. These recommendations

complement the analysis of Beck et al. (2024), who study architecture choices in a simpler single-phase setting, by documenting how network design interacts specifically with the sequential training protocol we propose.

Our fourth contribution is an application to international macroeconomics. We use the algorithm to solve a two-country open-economy model with Rotemberg pricing and to simulate the effects of tariff shocks of empirically realistic magnitudes. The exercise reveals strongly non-linear propagation that local methods cannot reproduce: as the shock size increases, the responses of some variables might change sign rather than simply scaling up. In our example model, for example, the nominal exchange rate switching from an appreciation under a small tariff to a sizeable depreciation under a large one. This sign reversal aligns with the empirical evidence of Ostry et al. (2025) on the response of the dollar to tariff announcements once retaliation is anticipated, and is rationalised theoretically by the dominant-currency-pricing mechanism of Bergin and Corsetti (2023). We also show how the deep neural network solution improve substantially over higher-order perturbation methods as applied in `Dynare` to account for non-linearities

Taken together, the results of this paper support a view of deep learning as a genuinely useful addition to the toolkit of quantitative macroeconomists, complementing rather than replacing existing methods. Perturbation remains indispensable for large-scale models where speed is paramount and shocks are moderate; perfect-foresight algorithms continue to be valuable for computing deterministic transition paths. Deep learning expands the frontier by providing globally accurate, stochastic solutions in settings where these alternatives fail, particularly when the state space is high-dimensional, shocks are large, or non-linearities are central to the economic question at hand (Fernández-Villaverde, 2025; Maliar et al., 2021). The sequential training protocol we propose reduces the barrier to entry by eliminating the need for a pre-computed starting solution, making deep learning methods accessible to a wider range of models and applications.

References

Adjemian, S., Juillard, M., Karamé, F., Mutschler, W., Pfeifer, J., Ratto, M., Rion, N., and Villemot, S. "Dynare: Reference Manual, Version 7". Dynare Working Papers 87, CEPREMAP, Mar. 2026.

Auclert, A., Bardóczy, B., Rognlie, M., and Straub, L. "Using the sequence-space Jacobian to solve and estimate heterogeneous-agent models". *Econometrica*, 89(5):2375–2408, 2021.

Auclert, A., Rognlie, M., and Straub, L. "The macroeconomics of tariff shocks". NBER Working Paper 33726, National Bureau of Economic Research, 2025.

Azinovic, M., Gaegauf, L., and Scheidegger, S. "Deep equilibrium nets". *International Economic Review*, 63(4):1471–1525, 2022.

Barron, A. R. "Approximation and estimation bounds for artificial neural networks". *Machine learning*, 14(1):115–133, 1994.

Bayer, C. and Luetticke, R. "Solving discrete time heterogeneous agent models with aggregate risk and many idiosyncratic states by perturbation". *Quantitative Economics*, 11(4):1253–1288, 2020.

Beck, P., Sanchez, P. G., Moura, A., Pascal, J., and Pierrard, O. "Deep learning solutions of dsge models: a technical report". Technical report, Central Bank of Luxembourg, 2024.

Bellman, R. *Dynamic Programming*. Princeton University Press, 1957.

Benigno, P. "Price stability with imperfect financial integration". *Journal of Money, Credit and Banking*, 41(s1):121–149, 2009. doi: 10.1111/j.1538-4616.2008.00201.x.

Bergin, P. R. and Corsetti, G. "The macroeconomic stabilization of tariff shocks: What is the optimal monetary response?". *Journal of International Economics*, 143:103758, 2023.

Bianchi, J. and Coulibaly, L. "The optimal monetary policy response to tariffs". *NBER Working Paper*, (33560).

Brumm, J. and Scheidegger, S. "Using adaptive sparse grids to solve high-dimensional dynamic models". *Econometrica*, 85(5):1575–1612, 2017.

Carroll, C. D. "Buffer-stock saving and the life cycle/permanent income hypothesis". *Quarterly Journal of Economics*, 112(1):1–55, 1997.

Christiano, L. J., Eichenbaum, M., and Evans, C. L. "Nominal rigidities and the dynamic effects of a shock to monetary policy". *Journal of Political Economy*, 113(1):1–45, 2005. doi: 10.1086/426038.

Cybenko, G. "Approximation by superpositions of a sigmoidal function". *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

Den Haan, W. J. and Marcet, A. "Solving the stochastic growth model by parameterizing expectations". *Journal of Business & Economic Statistics*, 8(1):31–34, 1990.

Duarte, V. "Gradient-based structural estimation". 2018.

Duarte, V., Duarte, D., and Silva, D. H. "Machine learning for continuous-time finance". *The Review of Financial Studies*, 37(11):3217–3271, 2024.

Duffy, J. and McNelis, P. D. "Approximating and simulating the stochastic growth model: Parameterized expectations, neural networks, and the genetic algorithm". *Journal of Economic Dynamics and Control*, 25(9):1273–1303, 2001.

Erceg, C. J., Prestipino, A., and Raffo, A. "Trade policies and fiscal devaluations". *American Economic Journal: Macroeconomics*, 15(4):104–140, 2023.

Fernández-Villaverde, J. "Deep learning for solving economic models". Working Paper 34250, National Bureau of Economic Research, 2025.

Fernández-Villaverde, J., Rubio-Ramírez, J. F., and Schorfheide, F. "Solution and estimation methods for DSGE models". In *Handbook of Macroeconomics*, volume 2, pages 527–724. Elsevier, 2016.

Fernández-Villaverde, J., Hurtado, S., and Nuno, G. "Financial frictions and the wealth distribution". *Econometrica*, 91(3):869–901, 2023.

Fernández-Villaverde, J., Nuño, G., and Perla, J. "Taming the curse of dimensionality: quantitative economics with deep learning". *NBER Working Paper*, (33117), 2024.

Furceri, D., Hannan, S. A., Ostry, J. D., and Rose, A. K. "The macroeconomy after tariffs". *The World Bank Economic Review*, 36(2):361–381, 2022. doi: 10.1093/wber/lhab016.

Galí, J. and Monacelli, T. "Monetary policy and exchange rate volatility in a small open economy". *Review of Economic Studies*, 72(3):707–734, 2005.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Hall-Hoffarth, E. "Non-linear approximations of dsge models with neural-networks and hard-constraints". *arXiv preprint arXiv:2310.13436*, 2023.

Hayashi, F. "Tobin's marginal q and average q: A neoclassical interpretation". *Econometrica*, 50(1):213–224, 1982. doi: 10.2307/1912538.

Hornik, K., Stinchcombe, M., and White, H. "Multilayer feedforward networks are universal approximators". *Neural Networks*, 2(5):359–366, 1989.

Judd, K. L. *Numerical Methods in Economics*. MIT Press, 1998.

Judd, K. L., Maliar, L., and Maliar, S. "Numerically stable and accurate stochastic simulation approaches for solving dynamic economic models". *Quantitative Economics*, 2(2):173–210, 2011.

Kingma, D. P. and Ba, J. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. "Imagenet classification with deep convolutional neural networks". In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

Krugman, P. "The macroeconomics of protection with a floating exchange rate". *Carnegie-Rochester Conference Series on Public Policy*, 16:141–182, 1982.

Krusell, P. and Smith, A. A., Jr. "Income and wealth heterogeneity in the macroeconomy". *Journal of political Economy*, 106(5):867–896, 1998.

Kydland, F. E. and Prescott, E. C. "Time to build and aggregate fluctuations". *Econometrica*, 50(6):1345–1370, 1982. doi: 10.2307/1913386.

Lindé, J. and Pescatori, A. "The macroeconomic effects of trade tariffs: Revisiting the Lerner symmetry result". *Journal of International Money and Finance*, 95:52–69, 2019. doi: 10.1016/j.jimonfin.2019.01.019.

Maliar, L., Maliar, S., and Winant, P. "Deep learning for solving dynamic economic models". *Journal of Monetary Economics*, 122:76–101, 2021. doi: 10.1016/j.jmoneco.2021.07.004.

McCulloch, W. S. and Pitts, W. "A logical calculus of the ideas immanent in nervous activity". *The Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. "Human-level control through deep reinforcement learning". *Nature*, 518:529–533, 2015.

Monacelli, T. "Tariffs and monetary policy". Working paper, Bocconi University, IGIER and CEPR, 2025.

Ostry, D., Lloyd, S., and Corsetti, G. "Trading blows: The exchange-rate response to tariffs and retaliations". *Bank of England Staff Working Paper*, (1,139), 2025.

Ostry, J. D. "Tariffs, real exchange rates, and the trade balance in a two-country world". *European Economic Review*, 35:1127–1142, 1991.

Prince, S. J. *Understanding deep learning*. MIT press, 2023.

Reiter, M. "Approximate and almost-exact aggregation in dynamic stochastic heterogeneous-agent models". *IHS Working Paper*, (258), 2010.

Robbins, H. and Monro, S. "A stochastic approximation method". *The annals of mathematical statistics*, pages 400–407, 1951.

Rosenblatt, F. "The perceptron: A probabilistic model for information storage and organization in the brain". *Psychological Review*, 65(6):386–408, 1958.

Rotemberg, J. J. "Sticky prices in the United States". *Journal of Political Economy*, 90 (6):1187–1211, 1982. doi: 10.1086/261117.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. "Learning representations by back-propagating errors". *nature*, 323(6088):533–536, 1986.

Scheidegger, S. and Bilionis, I. "Machine learning for high-dimensional dynamic stochastic economies". *Journal of Computational Science*, 33:68–82, 2019.

Schmitt-Grohé, S. and Uribe, M. "Solving dynamic general equilibrium models using a second-order approximation to the policy function". *Journal of Economic Dynamics and Control*, 28(4):755–775, 2004.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. "Attention is all you need". In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Villaverde, J. F., Marbet, J., Nuño, G., and Rachedi, O. "Inequality and the zero lower bound". *Documentos de trabajo-Banco de España*, (7):1, 2024.

Werning, I., Lorenzoni, G., and Guerrieri, V. "Tariffs as cost-push shocks: Implications for optimal monetary policy". NBER Working Paper 33772, National Bureau of Economic Research, 2025.

Winberry, T. "A method for solving and estimating heterogeneous agent macro models". *Quantitative Economics*, 9(3):1123–1151, 2018.

Appendix

A Algorithm

Require: Equilibrium residual functions $f_1(\cdot)$, $f_2(\cdot)$; steady-state values $\bar{\mathbf{s}}$, $\bar{\mathbf{x}}$; network depth L, width H; batch size n, simulation length T, burn-in τ; MC draws D, steady-state penalty λ_{ss}, learning rate η, tolerance ε, maximum epochs $K_{\max}$

Ensure: Trained policy network $\varphi(\,\cdot\,;\,\widehat{\boldsymbol{\theta}})$ such that $\mathbf{x}_t \approx \varphi(\mathbf{s}_t;\,\widehat{\boldsymbol{\theta}})$

Initialisation

Randomly initialise network weights: $\boldsymbol{\theta} \sim \mathcal{U}(-\delta, \delta)$

Initialise Adam optimiser with learning rate η

Phase 1 — Steady-State Anchoring ▷ $\lambda_{\text{ss}} \gg 1$, equilibrium residuals disabled

for $k = 1, \ldots, K_{\max}$ **do**

  Draw mini-batch of states: $\mathbf{s}_i \sim \mathcal{N}(\bar{\mathbf{s}},\, \sigma_1^2 \mathbf{I})$ for $i = 1, \ldots, n$

  Evaluate network: $\mathbf{x}_i \leftarrow \varphi(\mathbf{s}_i;\,\boldsymbol{\theta})$

  Compute steady-state loss: $\mathcal{L}^n(\boldsymbol{\theta}) \leftarrow \lambda_{\text{ss}} \dfrac{1}{n} \displaystyle\sum_{i=1}^{n} \left\| \mathbf{x}_i - \bar{\mathbf{x}} \right\|^2$

  $\boldsymbol{\theta} \leftarrow \text{Adam}(\boldsymbol{\theta},\, \nabla_{\boldsymbol{\theta}}\, \mathcal{L}^n(\boldsymbol{\theta}))$

  if $\mathcal{L}^n(\boldsymbol{\theta}) < \varepsilon$ **then**

    break

  end if

end for

Phase 2 — Random Exploration Around the Steady State ▷ $\lambda_{\text{ss}} = \lambda$, equilibrium residuals enabled, no MC integration

for $k = 1, \ldots, K_{\max}$ **do**

  Draw mini-batch: $\mathbf{s}_i \sim \mathcal{N}(\bar{\mathbf{s}},\, \sigma_2^2 \mathbf{I})$, $\sigma_2 > \sigma_1$, for $i = 1, \ldots, n$

  Evaluate network: $\mathbf{x}_i \leftarrow \varphi(\mathbf{s}_i;\,\boldsymbol{\theta})$

  Compute next-period states deterministically: $\mathbf{s}'_i \leftarrow \Gamma(\mathbf{s}_i,\, \mathbf{x}_i,\, \mathbf{0})$

  Evaluate network at next period: $\mathbf{x}'_i \leftarrow \varphi(\mathbf{s}'_i;\,\boldsymbol{\theta})$

Compute loss: $\mathcal{L}^n(\boldsymbol{\theta}) \leftarrow \frac{1}{n}\sum_{i=1}^{n}\Big[f_1(\mathbf{s}_i,\mathbf{x}_i)^2 + f_2(\mathbf{s}_i,\mathbf{x}_i,\mathbf{s}'_i,\mathbf{x}'_i)^2\Big] + \lambda_{\text{ss}}\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i-\bar{\mathbf{x}}\|^2$

$\boldsymbol{\theta} \leftarrow \text{Adam}(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}}\mathcal{L}^n(\boldsymbol{\theta}))$

if $\mathcal{L}^n(\boldsymbol{\theta}) < \varepsilon$ **then**

break

end if

end for

Phase 3 — Simulation on the Ergodic Set ▷ $\lambda_{\text{ss}} = \lambda$, no MC integration

for $k = 1, \ldots, K_{\max}$ **do**

Draw shock sequence: $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for $t = 1, \ldots, T$

Simulate trajectory with **no gradient tracking**:

$\{\mathbf{s}_t, \mathbf{x}_t\}_{t=1}^{T} \leftarrow \text{Simulate}\big(\varphi(\cdot\,;\boldsymbol{\theta}),\ \bar{\mathbf{s}},\ \{\boldsymbol{\varepsilon}_t\}_{t=1}^{T}\big)$

Subsample mini-batch (discard burn-in): $\{(\mathbf{s}_i,\mathbf{x}_i)\}_{i=1}^{n} \leftarrow \{\mathbf{s}_t,\mathbf{x}_t\}_{t=\tau+1}^{\tau+n}$

Re-evaluate network with gradient tracking: $\mathbf{x}_i \leftarrow \varphi(\mathbf{s}_i;\boldsymbol{\theta})$

Compute next-period states deterministically: $\mathbf{s}'_i \leftarrow \Gamma(\mathbf{s}_i, \mathbf{x}_i, \mathbf{0})$

Re-evaluate at next period: $\mathbf{x}'_i \leftarrow \varphi(\mathbf{s}'_i;\boldsymbol{\theta})$

Compute loss $\mathcal{L}^n(\boldsymbol{\theta})$ as in Phase 2

$\boldsymbol{\theta} \leftarrow \text{Adam}(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}}\mathcal{L}^n(\boldsymbol{\theta}))$

if $\mathcal{L}^n(\boldsymbol{\theta}) < \varepsilon$ **then**

break

end if

end for

Phase 4 — Simulation with Monte Carlo Expectations ▷ $\lambda_{\text{ss}} = \lambda$, full MC integration over future shocks

for $k = 1, \ldots, K_{\max}$ **do**

Simulate trajectory and subsample mini-batch as in Phase 3

Re-evaluate network: $\mathbf{x}_i \leftarrow \varphi(\mathbf{s}_i;\boldsymbol{\theta})$

for $d = 1, \ldots, D$ **do**

Draw future shock: $\boldsymbol{\varepsilon}^{(d)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Compute next-period state: $\mathbf{s}_i'^{(d)} \leftarrow \Gamma\left(\mathbf{s}_i,\, \mathbf{x}_i,\, \boldsymbol{\varepsilon}^{(d)}\right)$

Evaluate network at next period: $\mathbf{x}_i'^{(d)} \leftarrow \varphi\left(\mathbf{s}_i'^{(d)};\, \boldsymbol{\theta}\right)$

end for

Form MC expectation of the Euler residual: $\widehat{\mathbb{E}}_i[f_2] \leftarrow \frac{1}{D}\sum_{d=1}^{D} f_2\left(\mathbf{s}_i,\, \mathbf{x}_i,\, \mathbf{s}_i'^{(d)},\, \mathbf{x}_i'^{(d)}\right)$

Compute loss: $\mathcal{L}^n(\boldsymbol{\theta}) \leftarrow \frac{1}{n}\sum_{i=1}^{n}\left[f_1(\mathbf{s}_i, \mathbf{x}_i)^2 + \widehat{\mathbb{E}}_i[f_2]^2\right] + \lambda_{\text{ss}}\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - \bar{\mathbf{x}}\|^2$

Compute $\nabla_{\boldsymbol{\theta}}\mathcal{L}^n(\boldsymbol{\theta})$ via backpropagation

Clip gradients: $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}}\mathcal{L}^n$ **if** $\|\nabla_{\boldsymbol{\theta}}\mathcal{L}^n\| > 1$ **then** $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}}\mathcal{L}^n / \|\nabla_{\boldsymbol{\theta}}\mathcal{L}^n\|$

$\boldsymbol{\theta} \leftarrow \text{Adam}(\boldsymbol{\theta}, \mathbf{g})$

if $\mathcal{L}^n(\boldsymbol{\theta}) < \varepsilon$ **then**

break

end if

end for

return $\widehat{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$

B Two country model

Appendix A: Model Description and Equilibrium Conditions

B.1 Environment

The world economy consists of two countries, Home and Foreign, each of measure n and $1-n$ respectively, where $n = 1/2$. Each country is inhabited by a representative household, a continuum of monopolistically competitive intermediate goods firms, a competitive final goods producer, and a monetary authority. Variables pertaining to the Foreign country are denoted with a z subscript. The model is set in discrete time and all agents have rational expectations.

B.2 Households

Home household. The representative Home household maximises expected lifetime utility:

$$\mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \left[\frac{c_t^{1-\sigma}}{1-\sigma} - \kappa_L \frac{h_t^{1+\phi}}{1+\phi} \right], \tag{B.1}$$

where $\beta \in (0,1)$ is the subjective discount factor, c_t is consumption, h_t is hours worked, $\sigma > 0$ is the inverse elasticity of intertemporal substitution, $\phi \geq 0$ is the inverse Frisch elasticity, and $\kappa_L > 0$ is a labour disutility scaling parameter. The household accumulates physical capital k_t, supplies labour to domestic firms at the real wage w_t, receives rental income on capital at rate rk_t, holds domestic nominal bonds earning the gross nominal interest rate r_t, and receives profits from domestic firms. The flow budget constraint in real terms is:

$$c_t + i_t + \frac{b_t}{r_t} = w_t h_t + rk_t k_{t-1} + \frac{b_{t-1}}{\pi_t}, \tag{B.2}$$

where i_t denotes investment, b_t is the end-of-period real bond position, and π_t is the gross inflation rate of the Home consumer price index.

Following Hayashi (1982), capital adjustment costs are specified as a convex function

of the investment-to-capital ratio. Capital accumulates according to:

$$k_t = (1-\delta)k_{t-1} + i_t - \frac{\kappa_I}{2}\left(\frac{i_t}{k_{t-1}} - \delta\right)^2 k_{t-1}, \tag{B.3}$$

where the term $\frac{\kappa_I}{2}\left(\frac{i_t}{k_{t-1}} - \delta\right)^2 k_{t-1}$ represents resources lost to adjustment, and δ is the steady-state investment-to-capital ratio. A key advantage of this specification is that the first-order condition for investment yields Tobin's q as a purely static, analytical function of current investment and the existing capital stock:

$$q_t = \frac{1}{1 - \kappa_I\left(\frac{i_t}{k_{t-1}} - \delta\right)}, \tag{B.4}$$

so that $q^* = 1$ at the steady state, as required. Since q_{t+1} is likewise determined analytically from i_{t+1} and k_t, both of which are known at date t, the forward-looking investment Euler equation that arises under Christiano et al. (2005) costs is no longer needed. The capital Euler equation therefore reduces to a single condition:

$$q_t = \beta\,\mathbb{E}_t\left[\frac{u'(c_{t+1})}{u'(c_t)}\left(rk_{t+1} + (1-\delta)\,q_{t+1}\right)\right], \tag{B.5}$$

where both q_t and q_{t+1} are evaluated analytically rather than solved as separate network outputs. This reformulation reduces the number of equilibrium conditions that must be approximated by the neural network, and eliminates potential numerical instabilities that arise from the forward-looking investment Euler equation under the Christiano et al. (2005) specification. The capital Euler equation is:

$$q_t = \beta\mathbb{E}_t\left[\frac{u'(c_{t+1})}{u'(c_t)}\left(rk_{t+1} + (1-\delta)q_{t+1}\right)\right]. \tag{B.6}$$

The bond Euler equation and the intratemporal labour supply condition are:

$$1 = \beta\mathbb{E}_t\left[\frac{u'(c_{t+1})}{u'(c_t)}\frac{r_t}{\pi_{t+1}}\right], \tag{B.7}$$

$$w_t = \frac{\kappa_L h_t^{\phi}}{u'(c_t)}. \tag{B.8}$$

The Foreign household solves an isomorphic problem, replacing Home variables with their

Foreign counterparts.

International financial markets and the real exchange rate. In contrast to Benigno (2009), we assume that uncovered interest parity holds exactly, so that financial markets are complete at the international level. Complete markets imply the standard Backus–Smith condition:

$$s_t = \frac{u'(c_t)}{u'(c_{z,t})} = \frac{c_t^{-\sigma}}{c_{z,t}^{-\sigma}}, \tag{B.9}$$

where s_t denotes the real exchange rate, defined as the price of the Foreign consumption basket in terms of the Home basket. This condition equates, up to a constant pinned down by initial conditions, the ratio of marginal utilities of consumption across countries to the real exchange rate. As a result, the real exchange rate is determined endogenously by relative consumption levels and no separate equation for international bond holdings is required.

B.3 Firms

Production. Each country hosts a continuum of monopolistically competitive intermediate goods firms indexed on the unit interval. A representative Home firm produces output $y_{H,t}$ using capital and labour according to a Cobb–Douglas technology:

$$y_{H,t} = a_t k_{t-1}^{\alpha} h_t^{1-\alpha}, \tag{B.10}$$

where a_t is Home total factor productivity and $\alpha \in (0,1)$ is the capital share. TFP follows a stationary AR(1) process in logs:

$$\log a_t = (1-\rho_a)\log \bar{a} + \rho_a \log a_{t-1} + \sigma_a \varepsilon_{a,t}, \quad \varepsilon_{a,t} \sim \mathcal{N}(0,1), \tag{B.11}$$

where $\rho_a \in (0,1)$ and $\sigma_a > 0$. Cost minimisation by firms yields the standard factor demand conditions:

$$(1-\alpha) mc_t y_{H,t} = w_t h_t, \tag{B.12}$$

$$\alpha\, mc_t y_{H,t} = r k_t k_{t-1}, \tag{B.13}$$

where mc_t denotes the real marginal cost of production.

Price setting. Firms set prices subject to quadratic Rotemberg (1982) adjustment costs. The optimal price-setting decision yields a New Keynesian Phillips Curve (NKPC) for domestic inflation $\pi_{H,t}$:

$$(\pi_{H,t} - \bar{\pi})\pi_{H,t} = \beta\, \mathbb{E}_t \left[\frac{u'(c_{t+1})}{u'(c_t)} \frac{p_{H,t+1} y_{H,t+1}}{p_{H,t} y_{H,t}} \pi_{H,t+1}(\pi_{H,t+1} - \bar{\pi}) \right] + \frac{\varepsilon}{\kappa_P} \left(\frac{mc_t}{p_{H,t}} - \frac{\varepsilon - 1}{\varepsilon} \right), \tag{B.14}$$

where $\varepsilon > 1$ is the elasticity of substitution across varieties, $\kappa_P > 0$ governs the degree of price stickiness, and $\bar{\pi}$ is the steady-state inflation target. An analogous condition holds for Foreign firms with respect to $\pi_{Fz,t}$.

B.4 Price Indices and the Tariff

The Home consumer price index (CPI) aggregates domestic and imported goods through a CES bundle with trade share parameter $\gamma \in (0,1)$ and elasticity of substitution $\eta > 0$:

$$1 = (1-\gamma) p_{H,t}^{1-\eta} + \gamma \left[(1+\tau_t) p_{F,t} \right]^{1-\eta}, \tag{B.15}$$

where $p_{H,t}$ is the relative price of the Home good, $p_{F,t}$ is the relative price of the imported Foreign good, and τ_t is an *ad valorem* tariff levied by the Home country on imports from the Foreign country. The tariff drives a wedge between the border price of Foreign goods and the consumer price faced by Home households and firms, and its introduction represents the first key departure from Benigno (2009). The tariff follows an exogenous AR(1) process:

$$\tau_t = \rho_\tau \tau_{t-1} + \sigma_\tau \varepsilon_{\tau,t}, \quad \varepsilon_{\tau,t} \sim \mathcal{N}(0,1). \tag{B.16}$$

The Foreign CPI aggregates Foreign domestic goods and Home imports in an analogous fashion, but without a tariff:

$$1 = (1-\gamma_z) p_{Fz,t}^{1-\eta} + \gamma_z p_{Hz,t}^{1-\eta}. \tag{B.17}$$

The law of one price holds for each individual good, so that:

$$p_{H,t} = s_t p_{Hz,t}, \qquad p_{F,t} = s_t p_{Fz,t}. \tag{B.18}$$

Imported inflation rates are defined by:

$$\pi_{H,t} = \frac{p_{H,t}}{p_{H,t-1}}\pi_t, \qquad \pi_{Fz,t} = \frac{p_{Fz,t}}{p_{Fz,t-1}}\pi_{z,t}. \tag{B.19}$$

B.5 Market Clearing

Goods market clearing for the Home intermediate good requires that domestic production equals Home and Foreign demand, net of price adjustment costs:

$$y_{H,t} = (1-\gamma)p_{H,t}^{-\eta}(c_t + i_t) + \frac{1-n}{n}\gamma_z p_{Hz,t}^{-\eta}(c_{z,t} + i_{z,t}) + \frac{\kappa_P}{2}(\pi_{H,t} - \bar{\pi})^2 y_{H,t}. \tag{B.20}$$

An analogous condition holds for the Foreign good, where the tariff enters the Home demand for Foreign imports:

$$y_{Fz,t} = (1-\gamma_z)p_{Fz,t}^{-\eta}(c_{z,t} + i_{z,t}) + \frac{n}{1-n}\gamma\left[(1+\tau_t)p_{F,t}\right]^{-\eta}(c_t + i_t) + \frac{\kappa_P}{2}(\pi_{Fz,t} - \bar{\pi}_z)^2 y_{Fz,t}. \tag{B.21}$$

Home and Foreign GDP are defined as:

$$gdp_t = p_{H,t}y_{H,t}, \qquad gdp_{z,t} = p_{Fz,t}y_{Fz,t}. \tag{B.22}$$

B.6 Monetary Policy

Each country's central bank sets the short-term nominal interest rate according to an inertial Taylor rule. The Home rule is:

$$\frac{r_t}{r^*} = \left(\frac{r_{t-1}}{r^*}\right)^{\rho_m}\left[\left(\frac{\pi_t}{\bar{\pi}}\right)^{\phi_\pi}\left(\frac{gdp_t}{gdp^*}\right)^{\phi_y}\right]^{1-\rho_m}\exp(\sigma_r\varepsilon_{r,t}), \tag{B.23}$$

where r^* denotes the steady-state nominal interest rate, $\rho_m \in (0,1)$ is the interest rate smoothing parameter, $\phi_\pi > 1$ and $\phi_y \geq 0$ are the responses to inflation and output, and $\varepsilon_{r,t}$ is a monetary policy shock. The Foreign central bank follows an analogous rule.

B.7 Equilibrium

A competitive equilibrium consists of sequences of allocations $\{c_t, h_t, k_t, i_t, b_t, c_{z,t}, h_{z,t}, k_{z,t}, i_{z,t}\}_{t=0}^{\infty}$, prices $\{w_t, rk_t, w_{z,t}, rk_{z,t}, p_{H,t}, p_{F,t}, p_{Hz,t}, p_{Fz,t}, \pi_t, \pi_{z,t}, s_t\}_{t=0}^{\infty}$, and policy instruments $\{r_t, r_{z,t}, \tau_t\}_{t=0}^{\infty}$

such that: (i) households in both countries maximise utility taking prices as given; (ii) firms in both countries maximise profits subject to their price-setting constraints; (iii) all goods, labour, capital and financial markets clear; and (iv) the monetary authorities follow their respective Taylor rules and the tariff evolves according to its exogenous process.

B.8 Calibration

The discount factor is set to $\beta = 0.99$, implying a steady-state annual real interest rate of approximately four percent. The capital share in production is $\alpha = 0.33$ and the depreciation rate is $\delta = 0.025$, both standard values in the business cycle literature. The inverse elasticity of intertemporal substitution is $\sigma = 2$ and the inverse Frisch elasticity of labour supply is $\phi = 1$. Each country has equal size, $n = 0.5$, and the import share parameter is set to $\gamma = \gamma_z = 0.15$, consistent with a moderate degree of trade openness. The elasticity of substitution between domestic and imported goods is $\eta = 1.5$, and the elasticity of substitution across differentiated varieties is $\varepsilon = 6$, implying a steady-state markup of twenty percent. Price stickiness is governed by the Rotemberg adjustment cost parameter $\kappa_P = 28$, while the investment adjustment cost parameter is set to $\kappa_I = 2.48$ following Hayashi (1982). Monetary policy follows a Taylor rule with inflation response $\phi_\pi = 1.5$, output response $\phi_y = 0.125$, and interest rate smoothing $\rho_m = 0.8$. TFP shocks have persistence $\rho_a = 0.9$ and standard deviation $\sigma_a = 0.01$, while the tariff shock has persistence $\rho_\tau = 0.5$ and standard deviation $\sigma_\tau = 0.01$, so that a three standard deviation shock corresponds to a tariff rate of approximately three percent.

Table B.1: Calibrated parameter values

Param.	Description	Value	Param.	Description	Value
Panel A — Preferences and technology					
β	Discount factor	0.99	σ	Relative risk aversion	2.00
φ	Inverse Frisch elasticity	1.00	α	Capital share	0.33
δ	Capital depreciation rate	0.025	ε	Elasticity of substitution, varieties	6.00
Panel B — Trade and openness					
η	Trade elasticity (Home–Foreign)	1.50	n	Relative country size	0.50
γ	Home import share	0.15	γ^*	Foreign import share	0.15
Panel C — Nominal rigidities and adjustment costs					
κ_P	Price adjustment (Rotemberg)	28.00	κ_L	Wage adjustment	7.77
κ_I	Investment adjustment	2.48			
Panel D — Monetary policy (Taylor rule)					
ϕ_π	Inflation reaction	1.50	ϕ_y	Output reaction	0.125
ϕ_e	Exchange-rate reaction	0.00	ρ_m	Interest-rate smoothing	0.80
Panel E — Shock processes					
ρ_a	TFP shock persistence	0.90	σ_a	TFP innovation s.d.	0.01
ρ_τ	Tariff shock persistence	0.50	σ_τ	Tariff innovation s.d.	0.01
σ_r	Monetary innovation s.d.	0.01			
Panel F — Steady states (non-normalised)					
$\bar{a}$	Home TFP	1.0584	$\bar{a}^*$	Foreign TFP	1.0584
$\bar{r}$	Home gross interest rate	1.0101	$\bar{r}^*$	Foreign gross interest rate	1.0101

Notes: All other steady states are normalised. Gross inflation rates $\bar{\pi} = \bar{\pi}^* = 1$; the home-good price $\bar{p}_H = 1$; foreign GDP $\overline{GDP}^* = 1$; government spending and bond holdings $\bar{g} = \bar{g}^* = \bar{b}_F = \bar{b}^*_H = 0$.

C Figures

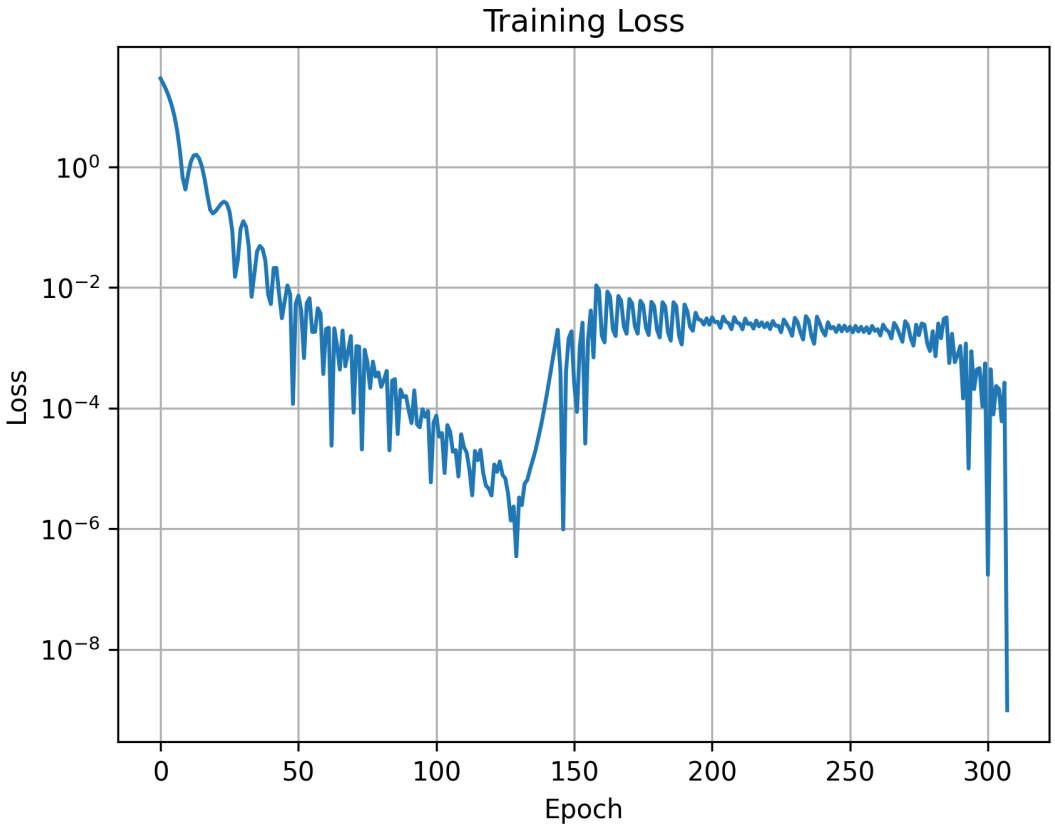

Figure C.1: Training loss over the execution of Phase 1 of the algorithm.
Notes: the figure shows the training loss over steps (Epochs) of the algorithm.

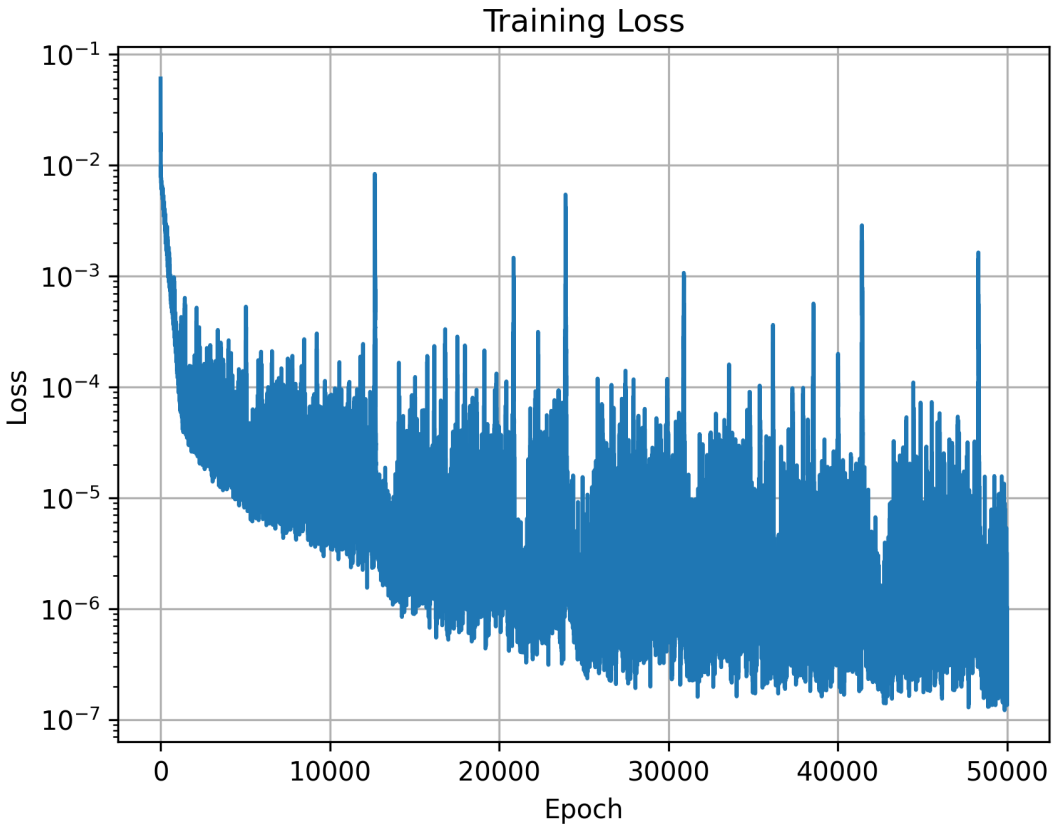

Figure C.2: Training loss over the execution of Phase 2 of the algorithm.
Notes: the figure shows the training loss over steps (Epochs) of the algorithm.

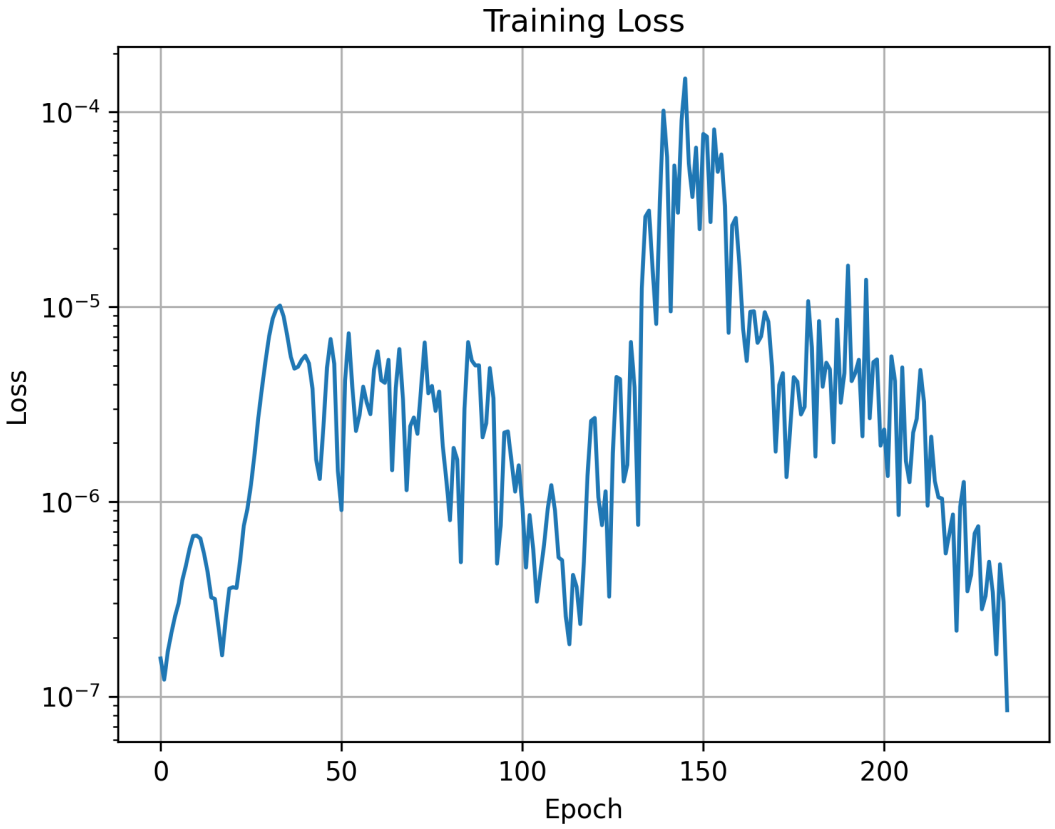

Figure C.3: Training loss over the execution of Phase 3 of the algorithm.
Notes: the figure shows the training loss over steps (Epochs) of the algorithm.

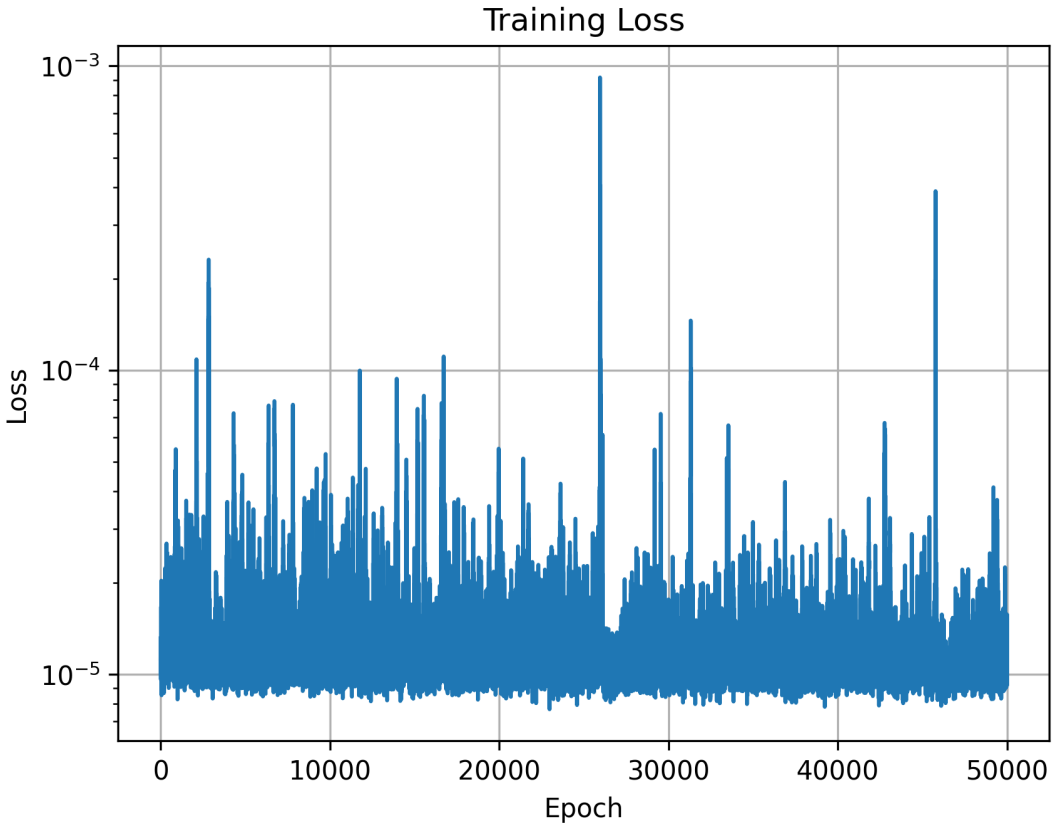

Figure C.4: Training loss over the execution of Phase 4 of the algorithm.
Notes: the figure shows the training loss over steps (Epochs) of the algorithm.

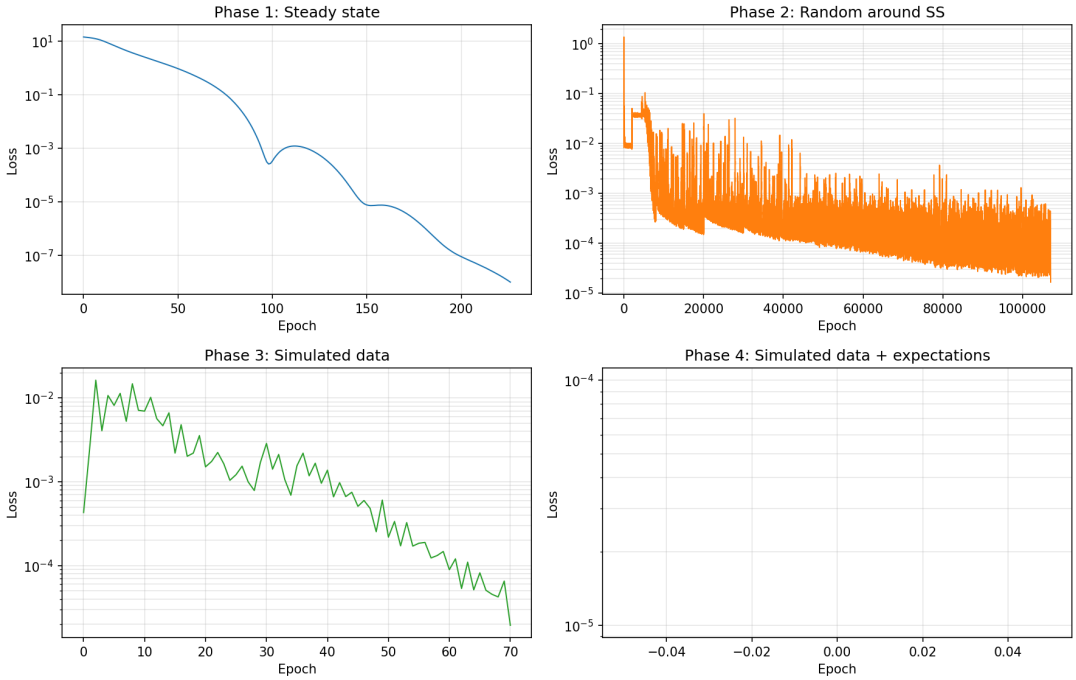

Figure C.5: Losses across the four phases of the training algorithm for the two-country model.
Notes: The figure shows the training loss over steps (Epochs) of the algorithm across the four phases of trainig for the two-country model with tariffs.

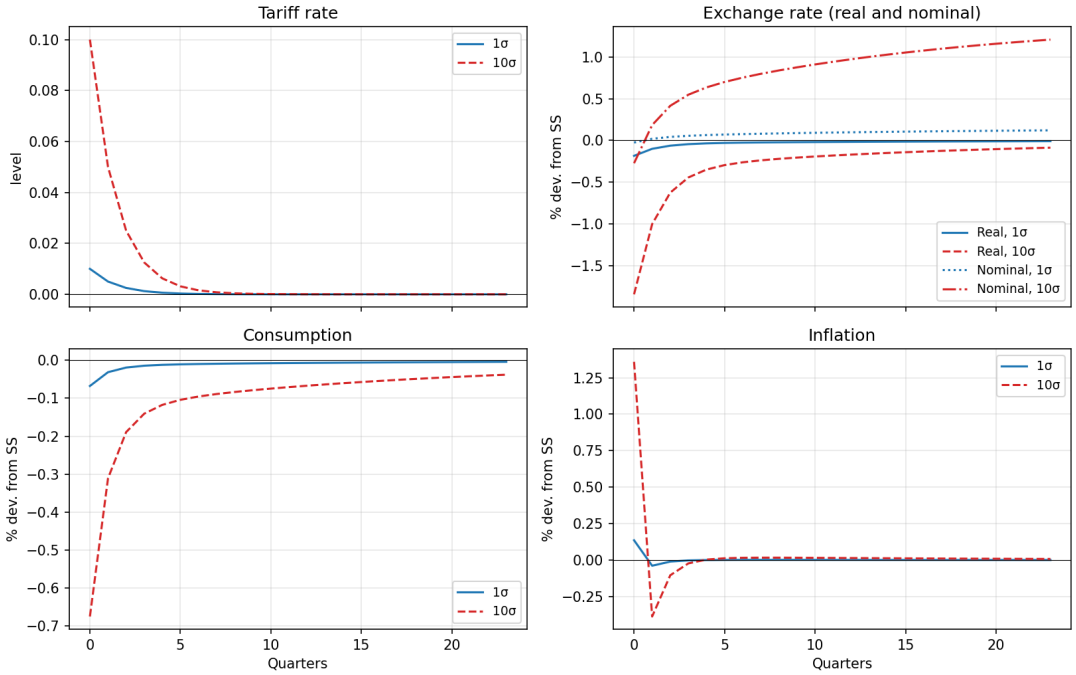

Figure C.6: Impulse response to a one-percent and ten-percent tariff shocks – second order perturbation.

Notes: The chart shows the impulse responses to a one-percent increase in the effective tariff rate imposed by the Home country on imports from the Foreign country. A negative deviation of the nominal exchange rate denotes an appreciation of the Home currency. The model is solved with a second order perturbation method as implemented by `Dynare`.

Acknowledgements

The authors are grateful to seminar participants at the ECB for useful comments.
The views expressed in this paper are those of the authors and do not necessarily represent the views of the European Central Bank or the Eurosystem.

Massimo Ferrari Minesso
European Central Bank, Frankfurt am Main, Germany; Complexity Lab in Economics, Università Cattolica del Sacro Cuore, Milan, Italy; email: massimo.ferrari_minesso@ecb.europa.eu

Carla Frenzel
European Central Bank, Frankfurt am Main, Germany; email: carla.frenzel@ecb.europa.eu

Postal address 60640 Frankfurt am Main, Germany
Telephone +49 69 1344 0
Website www.ecb.europa.eu

PDF ISBN 978-92-899-7895-8 ISSN 1725-2806 doi:10.2866/7455973 QB-01-26-137-EN-N