Valaitis, Vytautas; Villa, Alessandro T.

Working Paper

A machine learning projection method for macro-finance models

Working Paper, No. WP 2022-19

Provided in Cooperation with:
Federal Reserve Bank of Chicago

Suggested Citation: Valaitis, Vytautas; Villa, Alessandro T. (2022) : A machine learning projection method for macro-finance models, Working Paper, No. WP 2022-19, Federal Reserve Bank of Chicago, Chicago, IL, https://doi.org/10.21033/wp-2022-19

This Version is available at:
https://hdl.handle.net/10419/264333

Federal Reserve Bank of Chicago

A Machine Learning Projection Method for Macro-Finance Models

Vytautas Valaitis and Alessandro T. Villa

November 18, 2021

WP 2022-19

https://doi.org/10.21033/wp-2022-19

A Machine Learning Projection Method for Macro-Finance Models*

Vytautas Valaitis
EUI†

Alessandro T. Villa
FRBC‡

November 18, 2021

Abstract

This paper develops a simulation-based solution method to solve large state space macro-finance models using machine learning. We use a neural network (NN) to approximate the expectations in the optimality conditions in the spirit of the stochastic parameterized expectations algorithm (PEA). Because our method can process the entire information set at once, it is scalable and can handle models with large and multicollinear state spaces. We demonstrate the computational gains by extending the optimal government debt management problem studied by Faraglia et al. (2019) from two to three maturities. We find that the optimal policy prescribes an active role for the newly added medium-term maturity, enabling the planner to raise financial income without increasing its total borrowing in response to expenditure shocks. Through this mechanism the government effectively subsidizes the private sector in recessions, resulting in a welfare gain of 2.38% when the number of available maturities increases from two to three.

Keywords: Machine Learning, Incomplete Markets, Projection Methods, Optimal Fiscal Policy, Maturity Management.

JEL classification: C63, D52, E32, E37, E62, G12.

*We are thankful to Andrea Lanteri, Lukas Schmid and Matthias Kehrig for their encouragement. The paper received the Student Award of the Society of Computational Economics and benefited from comments by Albert Marcet, Serguei Maliar, Lilia Maliar, Swapnil Singh, and seminar participants at the Duke Macro Breakfast, the 2018 Baltic Economic Conference, the Society for Computational Economics 24th International Conference, the 2018 Econometric Society Summer European Meeting, and the 2019 Econometric Society African meeting. Disclaimer: The views expressed in this paper do not represent the views of the Federal Reserve Bank of Chicago or the Federal Reserve System. Declaration of conflicts of interest: none.

†European University Institute, Via della Badia dei Roccettini 9, Fiesole, IT 50014. Email: vytautas.valaitis@eui.eu

‡Federal Reserve Bank of Chicago, 230 S. LaSalle St. Chicago, IL, USA 60604. Email: alessandro.villa@chi.frb.org

1 Introduction

This paper introduces a new stochastic simulation method to solve DSGE models. In particular, we use a neural network (NN) to approximate the expectation terms contained in the optimality conditions of a DSGE model, in the spirit of the Parameterized Expectations Algorithm (PEA) introduced by den Haan and Marcet (1990). In general, machine learning has been used successfully as a computational tool for optimization to approximate and interpolate multidimensional functions. Our method leverages the stochastic simulation mechanism in a similar fashion to the PEA. On the one hand, stochastic simulation methods allow us to tackle problems with a large state space, since they calculate solutions only in the states that are visited in equilibrium (i.e., the ergodic set). On the other hand, when the set of state variables is generated by a stochastic simulation, it is likely to suffer from multicollinearity. The contribution of this paper is to show that an NN-based Expectations Algorithm can deal efficiently with multicollinearity and alleviate the curse of dimensionality. As a result, it allows us to explore the solution of models of increased complexity that feature a large multicollinear state space and non-linearities in the decision rules (e.g., caused by occasionally binding constraints). We demonstrate the computational gains of the NN-based Expectations Algorithm by extending the optimal debt management problem studied by Faraglia et al. (2019) to three maturities, and we investigate how the hedging benefits provided by additional maturity contribute to households' welfare.

We consider this application particularly challenging for four reasons. First, the number of state variables increases in the number and length of maturities available. Second, this class of problems includes forward-looking constraints, and the problem can be made recursive at the cost of adding even more state variables. Following Marcet and Marimon (2019), we formulate the recursive Lagrangian to solve for the time-inconsistent optimal contract under full commitment with three maturities. When markets are incomplete, the Ramsey planner needs to keep track of all promises made in the previous periods. Because of these reasons, optimal maturity management problems suffer from the curse of dimensionality (see Bellman 1961). For example, the optimal debt management problem with three maturities considered in section 4 features 27 state variables.[1] Third, the state space includes lagged values of the same variables, such as outstanding bonds and recursive Lagrange multipliers. This feature makes the stochastic simulation of the state

[1] In the model we use borrowing and lending constraints on each maturity and on the total portfolio and use the values of the multipliers associated with these constraints as part of the state space. Because of this, effectively we solve the model using 61 state variables.

space multicollinear.[2] Fourth, in this class of problems the optimal government debt level tends to converge to a long-run limit, as documented in Aiyagari et al. (2002), and tends to frequently hit the borrowing and lending constraints. Such properties make it hard to solve the model by perturbing the system around the steady state.[3]

In section 4, we use our method to study the welfare effects of government debt management by allowing the Ramsey planner to issue an increasing number of debt instruments. Typically, the prices of longer maturities are more responsive to shocks than prices of shorter maturities. This differential response creates opportunities for hedging by borrowing in long-term and saving in short-term bonds. In this case, the value of liabilities falls by more than the value of assets in response to negative shocks (see Angeletos 2002, Buera and Nicolini 2004 and Faraglia et al. 2019). Additionally, the fact that short bond prices are not as responsive to shocks allows the planner to smooth the price of new debt issuance by rebalancing the portfolio toward the longer maturities in economic booms and toward the shorter maturities in recessions. We find that the planner actively uses additional maturities to exploit both the hedging and the price smoothing opportunities. It holds leveraged positions and rebalances the portfolio with more emphasis on the shorter maturities in recessions. As a consequence, we find that as the number of available maturities increases from one to three, the total debt becomes procyclical. The additional maturities allow the government to respond to expenditure shocks by raising financial income without increasing the total outstanding debt. Through this mechanism, the government effectively subsidizes the private sector in recessions, resulting in a welfare gain of 2.38% when the number of available maturities increases from two to three.

Literature Review Our method builds on the seminal work of den Haan and Marcet (1990), who introduced PEA. PEA has been extended more recently (see Faraglia et al. 2014 and Faraglia et al. 2019) to deal with multicollinearity (Condensed PEA) and over-identification (Forward-States PEA). The main contribution of our paper is to introduce an NN-based Expectations Algorithm, allowing for machine learning to reduce the state space endogenously and handling multicollinearity

[2] Multicollinearity in the state space might prevent standard regression based algorithms from converging because the estimated regression coefficients may never stabilize due to high estimation variance and because misspecification of the true policy function under multicollinearity may lead to severe prediction bias, as we show in section 2.6. Alternatively, people have used the stochastic simulation based on regularization, see Judd et al. (2011), or have extended the PEA algorithm to Condensed PEA, see Faraglia et al. (2019). In section 5 we discuss how the NN-based Expectations Algorithm improves upon these methods.

[3] Bhandari et al. (2017) propose a method that allows one to approximate a system around a current level of government debt, and Lustig et al. (2008) on the other hand, solve the optimal fiscal policy problem in incomplete markets with seven maturities up to 7 periods using value function iteration on a sparse grid.

effectively when a stochastic simulation approach is adopted. In contrast, Condensed PEA achieves this result by introducing an external loop that tests a subset of the state space as a candidate to solve the model. Other papers that use neural networks to solve economic models include Scheidegger and Bilionis (2019), Azinovic et al. (2021), Fernández-Villaverde et al. (2020), Maliar et al. (2021) and Duarte (2018). Fernández-Villaverde et al. (2020) use deep neural networks to approximate the aggregate laws of motion in a heterogeneous agents model featuring strong non-linearities and aggregate shocks. Duarte (2018) casts the economic model in continuous time and uses neural networks to approximate the Bellman equation. Maliar et al. (2021) and Azinovic et al. (2021) approximate all the model equilibrium conditions using neural networks and use the simulated data to train them. Maliar et al. (2021) illustrates the method in the Krusell and Smith (1998) model, and Azinovic et al. (2021) solve the life-cycle model with borrowing constraints, aggregate shocks, and financial frictions using unsupervised machine learning. The main difference of our paper is to leverage on supervised machine learning to deal effectively with the problem of multicollinearity typical of stochastic simulation approaches. In this context, we show how our algorithm can alleviate the curse of dimensionality, allowing us to explore the problem of the optimal maturity structure of government debt in a more realistic environment.

On the one hand, simulation-based methods allow us to tackle problems with a larger state space, since they allow us to calculate solutions only in a subset of the state space. On the other hand, when the set of state variables is generated by a stochastic simulation, they are likely to be jointly multicollinear. In the spirit of PEA, the literature tackles this problem by introducing the Condensed PEA (Faraglia et al., 2019), which requires an iterative procedure that looks for an orthogonal set of regressors. It is conceptually similar to principal component extraction. However, in contrast to principal component extraction, Condensed PEA features a number of factors determined endogenously. The main contribution of this paper is to show that the NN-based Expectations Algorithm can approximate these expectations, digesting all information set at once, allowing for machine learning to reduce the state space endogenously. PEA can potentially be used in combination with other standard econometric techniques that tackle the problem of multicollinearity, as in Judd et al. (2011). Similar to our paper, Judd et al. (2011) adopt a stochastic simulation approach and show how already established methods in econometrics can be used to alleviate the multicollinearity problem using a multi-country neoclassical growth model. We discuss the relation between our method and the methods of Faraglia et al. (2019) and Judd et al. (2011) in greater detail in section 5.

Our application also contributes to the strand of literature in optimal fiscal policy. In particular, it is relevant to the literature on optimal maturity structure of government debt.[4] Lustig et al. (2008) find that the optimal policy prescribes an almost exclusive role to the longest maturity in a model with lending constraints and where fiscal and monetary policies interact. Bhandari et al. (2017) study the optimal maturity structure in an open economy with two types of bonds available, and Bigio et al. (2021) allow for an arbitrary number of bonds in a closed economy where they study an equilibrium in which aggregate shocks are anticipated but not realized. Faraglia et al. (2019) is the closest paper to ours and studies the role of frictions in a closed economy with two types of bonds. Solving the Ramsey problem considered in this paper is particularly challenging, as the dimension of its state space increases significantly in function of the length of the maturities and the number of bonds. Moreover, this class of problem includes forward-looking constraints, so the commonly used recursive representation can not be adopted. Marcet and Marimon (2019) provide an alternative formulation to solve for the time-inconsistent optimal contract under full commitment: a recursive Lagrangian or saddle-point functional equation. The solution involves adding even more state variables to the original problem. These additional state variables, necessary to recursify the problem, create history dependence. In this context, we use our methodology to extend the literature to study optimal debt management with three maturities in a closed economy. We find that the optimal policy prescribes an active role for the medium-term bond. This additional maturity enables the planner to reduce the total debt portfolio while at the same time raising financial revenue in response to expenditure shocks. We show that the use of additional maturities leads to significant welfare gains.

The paper is organized as follows. Section 2 is a user guide that introduces the reader to PEA, machine learning, and how to combine them in a simple Neoclassical Growth Model example. Section 3 introduces the reader to the problem of multicollinearity using a one-bond economy studied in Aiyagari et al. (2002) and describes the details of the NN-based Expectations Algorithm using a general model with N maturities. Section 4 presents and discusses the calibration and the quantitative results for the extended model with three maturities. Section 5 discusses and compares the NN-based Expectations Algorithm to other state-of-the-art methods. Section 6 concludes.

[4] Aiyagari et al. (2002), Angeletos (2002), Buera and Nicolini (2004), Lustig et al. (2008), Faraglia et al. (2019), Bhandari et al. (2017), and Bigio et al. (2021).

2 User guide: Machine Learning and PEA

This section serves as a user guide to introduce the reader to the basics of machine learning and how to use it to solve a dynamic economic model in a similar fashion to PEA. Hence, the purpose of this section is solely to introduce the methodology in a simple environment. The method allows us to investigate more realistic models of increased complexity. Its benefits are highlighted in the application presented in section 3 and arise from the ability of the algorithm to approximate generic non-linear policy functions in the presence of a large and multicollinear state space.

2.1 Environment

The typical dynamic model contains Euler equations and laws of motion

$$
\begin{aligned}
f(c_t, X_t) &= \mathbb{E}\left[g(c_{t+1}, X_{t+1})|X_t\right], \\
X_{t+1} &= h(X_t, c_t, \xi_{t+1}),
\end{aligned}
$$

where $c_t \in \mathbb{R}^C$ is a vector of C controls, $X_t \in \mathbb{R}^S$ is a vector of endogenous and exogenous state variables, $f : \mathbb{R}^C \times \mathbb{R}^S \to \mathbb{R}^C$, $g : \mathbb{R}^C \times \mathbb{R}^S \to \mathbb{R}^C$, and ξ_{t+1} is a vector of innovation shocks. For example, in the stochastic neoclassical growth model c_t is consumption, f is marginal utility, $g = f(c_{t+1})\left(z_{t+1}K_{t+1}^{\alpha-1} + 1 - \delta\right)$, $X_t = \{K_t, z_t\}$ is a vector that contains capital stock and TFP, and $h(X_t, c_t, \xi_{t+1})$ is a function that describes the laws of motion for capital stock, given by the resource constraint $K_{t+1} = (1-\delta)K_t - c_t z_t K_t^{\alpha}$ and the TFP Markov process. The typical PEA approximates the conditional expectations in the Euler equations as polynomial functions of the state space X_t

$$
\mathbb{E}\left[g(c_{t+1}, X_{t+1})|X_t\right] \simeq P_n(X_t; \eta).
$$

The polynomial typically used in the PEA is

$$
P_n(X_t; \eta) = \exp\left(a_0 + \sum_{p=1}^{P}\sum_{s=1}^{S}\left[a_{p,s} \cdot (\ln X_{s,t})^p\right]\right),
$$

where $\eta = [a_0, a_{1,1}, \cdots, a_{1,S}, \dots]$. For a given sequence of exogenous aggregate shocks $\{\xi_t\}_{t=1}^T$, an initial guess of the polynomials' parameters η^1, the standard stochastic PEA (described in Algorithm 1) aims to find parameters η^n that solve all Euler equations and all laws of motion.

When $X \equiv \{X_t\}_{t=T_0}^T$ is generated by a stochastic simulation as in Algorithm 1, the matrix $X^T X$

is often ill-conditioned.[5] Hence, with a finite-precision computer, the inverse of $X^T X$ cannot be computed reliably and it is challenging to compute the linear regression in line 9 of Algorithm 1. This problem potentially leads to jumps in the regression coefficients and failure to converge.

Algorithm 1 Stochastic (simulations) PEA

Precondition: initial state X_0, sequence $\{\xi_t\}_{t=0}^T$, initial guess η_n^1, and dampening $0 < w < 1$

1: **while** η_n^i converges **do**
2: **for** $t \leftarrow 0$ to T **do** ▷ Generate $X \equiv \{X_t\}_{t=0}^T$
3: $c_t \leftarrow f^{-1}(P_n(X_t; \eta_n))$
4: $X_{t+1} \leftarrow h(X_t, c_t, \xi_{t+1})$
5: **end for**
6: **for** $t \leftarrow 0$ to T **do** ▷ Generate $Y \equiv \{y_{t+1}\}_{t=0}^T$
7: $y_{t+1} \leftarrow g(c_{t+1}, X_{t+1})$
8: **end for**
9: $\hat{\eta}_n^i \leftarrow (X^T X)^{-1} X^T Y$ ▷ Regress to find new weights
10: $\eta_n^{i+1} \leftarrow w \cdot \hat{\eta}_n^i + (1 - w) \cdot \eta_n^i$ ▷ Update with dampening
11: **end while**

Moreover, in the simple illustrative case of the neoclassical growth model a first order polynomial ($P = 1$) is enough to approximate the expectation term in the Euler equation. Generically speaking, richer models that feature a larger state space and non-linearities require the use of higher order approximation ($P \gg 1$) and/or cross-state terms. These circumstances further aggravate the multicollinearity problem as the matrix $\hat{X}^T \hat{X}$, with $\hat{X} \equiv \left\{ X_t, X_t^2, \cdots \right\}_{t=0}^T$, is even more ill-conditioned.

2.2 Supervised Machine Learning

In this paper, we use machine learning as a tool to learn how to represent the function that maps from the set of simulated state variables $\{X_t\}_{t=0}^T$ to the set of simulated terms $\{y_{t+1}\}_{t=0}^T$. For example, in the neoclassical growth model this would serve the purpose to represent the function

$$P(K_t, z_t) = \mathbb{E}\left[c_{t+1}^{-1} \left(z_{t+1} K_{t+1}^{\alpha - 1} + 1 - \delta \right) | K_t, z_t \right].$$

Machine learning proposes a flexible structure for the function P and infers a function from the generated data $\{X_t\}_{t=0}^T$ (which we label *training data*) to the set of generated examples $\{y_{t+1}\}_{t=0}^T$ (which we label *training examples*). This particular task of using machine learning to learn a

[5] Let $\lambda = \lambda_1, \cdots, \lambda_S$ be the vector of eigenvalues of the matrix $X^T X$, such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_S \geq 0$. Ill-conditioning refers to the fact that the ratio λ_1 / λ_n is large, implying the matrix is close to being singular.

function that maps from inputs to outputs based on training data and examples is referred to the literature as *supervised learning*. A powerful class of universal approximators able to deal with strong non-linearities is neural networks.

2.3 Artificial Neural Networks

Neural networks can be used for both regression and classification purposes. They are typically composed of three types of layers: (i) input, (ii) hidden and (iii) output. They can contain multiple hidden layers but, for regression purposes, typically one or two hidden layers are sufficient. The *universal approximation theorem* (see Cybenko, 1988 and Hornik et al., 1989) ensures that every bounded continuous function can be approximated with arbitrarily small error, by a network with one hidden layer. Moreover, any function can be approximated to arbitrary accuracy by a neural network with two hidden layers (Cybenko, 1988). In our application, the input layer takes as input the state space $X_t \in \mathbb{R}^S$. The hidden layer performs an intermediate transformation of the state space. The output layer predicts the expectation terms contained in the model optimality conditions $\mathbb{E}\left[g(c_{t+1}, X_{t+1})|X_t\right] \simeq f(X_t; w, \beta) \in \mathbb{R}^E$. See figure 7 in appendix B for a graphical illustration. If the problem requires approximation of E expectations terms, a neural network with one hidden layer has the following functional form

$$\tilde{X}_m = H\left(\sum_{s=0}^{S} w_{m,s} \cdot X_{s,t}\right), \quad m = 1, \cdots, M,$$
$$f_e(X_t; w, \beta) = \beta_{0,e} + \sum_{m=1}^{M} \beta_{m,e} \cdot \tilde{X}_m, \quad e = 1, \cdots, E.$$

The hidden layer transforms the state space $X_t \in \mathbb{R}^S$ through M linear combinations of the state variables, further transformed through an activation function $H(x)$. The activation function is typically a sigmoid

$$H(x; \alpha) = \frac{1}{1 + \exp(-\alpha \cdot x)},$$

where α is a parameter that regulates the activation rate. Intuitively, the larger is α the more $H(x; \alpha)$ resembles to a step function as shown in figure 1.

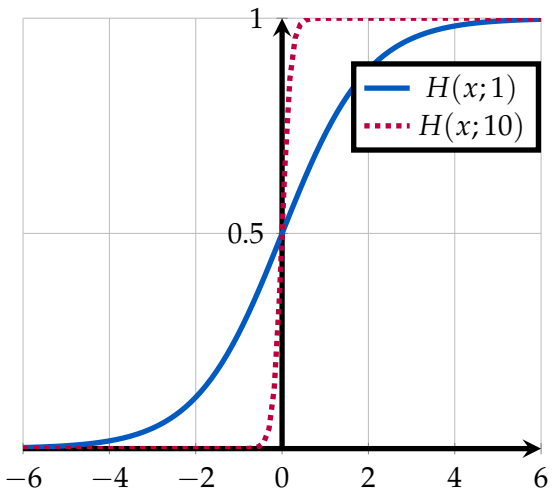

Figure 1: Sigmoid Function

Notes: The plot of the sigmoid function $H(x;\alpha)$, typically used in the hidden layer of a neural network. Solid blue line: Plot of the sigmoid function when $\alpha = 1$. Dashed purple line: Plot of the sigmoid function when $\alpha = 10$. The higher is α the more the sigmoid function acquires the shape of a step function.

2.4 Fitting Neural Networks

A neural network is characterized by unknown weights $\{w, \beta\}$. The objective of the *training phase* is to seek weights such that the neural network fits the samples $\{X_t, y_t\}_{t=0}^{T}$. More precisely, the problem is to find

$$\{w_{0,m}, w_m;\ m = 1, 2, \cdots, M\}, \{\beta_{0,e}, \beta_e;\ e = 1, 2, \cdots, E\},$$

such that the sum of squares

$$R(w, \beta) = \sum_{e=1}^{E} \sum_{t=0}^{T} (y_{t,e} - f_e(X_t; w, \beta))^2$$

is minimized. There are two important aspects to keep in mind compared to the standard PEA polynomial regression approach: (i) training a neural network typically does not require seeking a global minimizer for $R(w, \beta)$ since that solution is likely an overfit, and (ii) unlike OLS the network is trained using a gradient iterative procedure (e.g. gradient descendant). The gradient can be derived using the chain rule for differentiation. An iteration n of gradient descendant updates the

weight according to

$$w_m^{(n+1)} = w_m^{(n)} - \gamma_r \sum_{k=1}^{K} \frac{\partial R_k(w)}{\partial w_m},$$
$$\beta_e^{(n+1)} = \beta_e^{(n)} - \gamma_r \sum_{k=1}^{K} \frac{\partial R_k(w)}{\partial \beta_e}.$$

Back-Propagation The partial derivatives $\frac{\partial R_k(w)}{\partial w_m}$ and $\frac{\partial R_k(w)}{\partial \beta_e}$ can be efficiently computed through back-propagation (Rumelhart et al., 1986). Back-propagation is a two-pass algorithm that applies the chain-rule sequentially, iterating from the output layer to the input layer. Each neuron in the hidden layer receives and dispatches information only from and to neurons that are directly connected. For this reason this process can be efficiently parallelized. When the back-propagation algorithm is applied to a single-layer neural network, it is known as the delta rule (Widrow and Hoff, 1960).

Training Epoch Completing a training epoch means that all training samples have had a chance to update the model parameters. Batch (or offline) learning builds the model digesting the entire training set at once, whereas online training allows the network to update the weights as new observations come in. The former is typically implemented by batch gradient descent, when the latter can typically handle larger training sets and is implemented by stochastic gradient descent.

Learning Rate The learning rate γ_r is similar in spirit to a dampening parameter. It refers to the speed at which the model changes when the weights are updated. Intuitively, it represents how quickly the model "learns". It can either be a constant (for batch learning) or optimized dynamically at each update by minimizing the error function.

Initial Weights Initial neural network's weights are chosen as near zero random values. Figure 1 suggests that when the weight α is close to zero, the sigmoid approaches a linear function. This choice of initial weights allows the model to adapt to non-linearities starting from the linear case.

Overfitting The model should not overfit the data. Since stochastic simulation methods such as PEA only explore the ergodic set of state variables, it is particularly important to optimize the model for out-of-sample predictions. We split the simulated data randomly in training set

(in-sample) and validation set (out-of-sample) with a 70-30 proportion, respectively. The number of epochs is determined by maximizing the neural network's performance on the validation set.

Inputs Normalization All inputs are normalized to have mean zero and unitary standard deviation. This procedure ensures that all inputs have a comparable magnitude. If some inputs were of a bigger order of magnitude, the weights linked to those inputs would experience a faster update speed. This could potentially impair the learning process and lead to a slower convergence, or worst mean squared prediction errors.

Number of Neurons The choice of the number of neurons in the hidden layer should be guided by the trade-off between in-sample fit and out-of-sample performance, as illustrated in figure 2 (the figure refers to the neoclassical growth model that we present as an illustrative example in the next section). Increasing the number of hidden units tends to increase the in-sample fit but leads to over-fitting. We select the number of units by minimizing the mean squared prediction error calculated on the validation set.

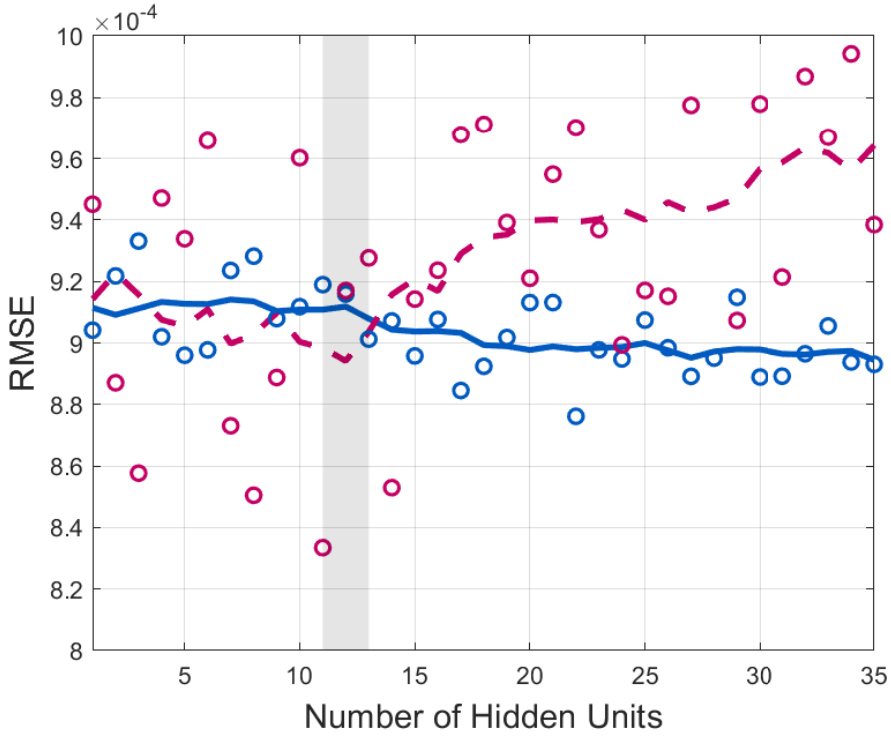

Figure 2: Root Mean Squared Error (RMSE) in function of the number of neurons in the hidden layer

Notes: The figure shows the relation between the number of hidden units and neural network performance in the neoclassical growth model. Solid blue line - network performance on the training set. Dashed purple line - network performance on the validation set. Circles show network performance for a specific number of units. Lines represent the moving averages.

2.5 Example: Neural Networks and PEA Applied to the Neoclassical Growth Model

This section describes the implementation of the NN-based Expectations Algorithm applied to the neoclassical growth model. We use Matlab and we leverage on the *Statistics and Machine Learning Toolbox*. The illustrative example code, together with the comparison with other methods and the procedure that selects the optimal number of neurons is publicly available.[6] The purpose of using Matlab and disseminating this application is to facilitate the adoption of machine learning in economics with well known tools in an easy-to-adopt package. In this example we use a single layer neural network with 12 neurons (this number of neurons minimize the mean squared prediction error out-of-sample as shown in figure 2).

We first calculate the steady-state, which is particularly useful to build a guess to initialize the neural network weights. The command *feedforwardnet(12)* creates a neural network with one hidden

[6]Downloadable from https://www.alessandrotenzinvilla.com/research.html.

Algorithm 2 NN-Based Expectations Algorithm applied to the Neoclassical Growth Model

Precondition: parameters β, α, δ, ρ, σ^{ϵ}, k_1, and ϵ; utility functions $u(c) = log(c)$, $u_1(c) = c^{-1}$.

▷ *Simulate log AR(1) process and an initial guess for k and c*
$\log(z_{t+1}) \leftarrow \rho \cdot \log(z_t) + \epsilon_{t+1}$
$k_{I,t} \leftarrow [(1 - \beta(1 - \delta))/(\beta \cdot \alpha \cdot z_t)]^{1/(\alpha-1)}$
$c_{I,t} \leftarrow z_t \cdot k_{I,t}^{\alpha} - \delta \cdot k_{I,t}$

▷ *Create and train the NN using the initial k_I and z alongside the RHS_t*
$\text{Net} \leftarrow \text{feedforwardnet}(12)$
$\text{RHS}_t \leftarrow u_1(c_{I,t+1})[\alpha \cdot z_{t+1} \cdot k_{I,t+1}^{\alpha-1} + 1 - \delta]$
$\text{Net} \leftarrow \text{train}(\text{Net}, [k_{I,t}, z_t], \text{RHS}_t)$
$k_{\text{old}} \leftarrow k_I$

▷ *Solve the model*
while error $> \epsilon$ **do**
    ▷ *Generate $\{c_t\}_{t=1}^{T}$ and $\{k_t\}_{t=1}^{T}$*
    for $t \leftarrow 1$ to T **do**
        $\mathbb{E}_t[\text{RHS}_{t+1}] \leftarrow \text{Net}([k_t; z_t])$
        $c_t \leftarrow u_1^{-1}(\beta \cdot \mathbb{E}_t[\text{RHS}_{t+1}])$
        $k_{t+1} \leftarrow z_t \cdot k_t^{\alpha} + (1 - \delta) \cdot k_t - c_t$
    end for
    ▷ *Train the NN using the new k and z alongside the above RHS*
    $\text{RHS}_t \leftarrow u_1(c_{t+1}) \cdot (\alpha \cdot z_{t+1} \cdot k_{t+1}^{\alpha-1} + 1 - \delta)$
    $\text{Net} \leftarrow \text{train}(\text{Net}, [k_{I,t}, z_t], \text{RHS}_t)$
    ▷ *Checking convergence and updating k_{old}*
    $\text{error} \leftarrow \max(|k_{\text{old},t} - k_t|)$
    $k_{\text{old},t} \leftarrow k_t$
end while

layer that contains 12 neurons. By default, this neural network is trained (through the function *train*) with Levenberg-Marquardt backpropagation, and has a maximum number of epochs set to 1000. We generate an initial dataset using the deterministic steady-state and substituting the value of the shock.

We then proceed to solve the model using the equivalent of Algorithm 1, except we use the neural network to approximate the expectation contained in the optimal condition of the model. We call this the NN-based Expectations Algorithm and a detailed description of the code is laid out in algorithm 2.

In a more complex environment with a large state space - where a stochastic simulation approach is desirable - the advantages of this method lie in the interaction between a satisfactory approximation of the model non-linearities and the degree of multicollinearity among the simulated states. In PEA the choice of polynomials is quite arbitrary as the policies' functional forms are ex-ante unknown (one has to rely on an ex-post accuracy test to make sure that the approximation is satisfactory). If the functional form of the chosen approximator cannot satisfactorily approximate the equilibrium policies, the presence of multicollinearity can lead to bias in the parameter estimates. The neural network does not suffer from this problem as it is a universal approximator. In the next section we conclude the user's guide illustrating this point.

2.6 Neural Networks and Multicollinearity

One general problem is that the functional form, not just the parameters, that links the state variables and the approximated terms is ex-ante unknown. A standard practice is to make these approximations using polynomials with order and cross-terms typically chosen through trial and error. When the policies are correctly specified, multicollinearity leads to consistent, yet noisy parameter estimates. However, if the chosen functional forms are not suitable to approximate the true policy functions, multicollinearity can potentially lead to severely biased and less precise predictions as we show in the following simple example.

Imagine that we would like to approximate a policy function with the true functional form

$$y(x_1, x_2) = \alpha x_1 + \beta x_2 + \gamma x_2^2, \tag{1}$$

and true parameters $\alpha = 2$, $\beta = 3$, and $\gamma = 1$. Imagine also that we use the optimality conditions of the model to generate, through stochastic simulation, equilibrium sequences $\{X_t, y_t\}_{t=0}^{\infty}$, where

$X_t = [x_{1,t} \quad x_{2,t}]$ is a vector that contains the two state variables. The objective is to use $\{X_t, y_t\}_{t=0}^{\infty}$ in order to infer: (i) the functional form of equation 1 and (ii) the true parameters α, β, and γ. When $X \equiv \{X_t\}_{t=0}^{T}$ is generated by a stochastic simulation as in Algorithm 1, the matrix $X^T X$ is often ill-conditioned. We simulate different degrees of multicollinearity by randomly generating sequences $\{x_{1,t}\}_{t=0}^{T}$ and $\{x_{2,t}\}_{t=0}^{T}$ with different degrees of correlation, and we calculate the associated $\{y_t\}_{t=0}^{T}$ using equation 1. We evaluate the success of the prediction in function of different degrees of multicollinearity using a (i) linear polynomial and a (ii) neural network. Note that on purpose we incorrectly assume that the mapping between state variables and policy is linear $y_t = \beta_1 x_{1,t} + \beta_2 x_{2,t}$.[7] Also note that the neural network has a flexible non-parametric nature and, therefore, does not require making ex-ante assumptions about the functional form of the actual data generating process. The success of the prediction is assessed using the mean squared prediction error (MSPE), which is the average prediction error at time t over many training samples. The error can be decomposed in bias and variance terms

$$\text{MSPE}_t = \mathbb{E}\left[(y_t - \hat{y}_t)^2\right] = \underbrace{\left[y_t - \mathbb{E}(\hat{y}_t)\right]^2}_{Bias_t^2} + \underbrace{\mathbb{E}\left[\hat{y}_t - \mathbb{E}(\hat{y}_t)\right]^2}_{Variance_t}. \tag{2}$$

Figure 3 reports the average MSPE for the entire validation set in function of the correlation between $\{x_{1,t}\}_{t=0}^{T}$ and $\{x_{2,t}\}_{t=0}^{T}$. Note that the higher the correlation, the higher the multicollinearity between $\{x_{1,t}\}_{t=0}^{T}$ and $\{x_{2,t}\}_{t=0}^{T}$.

[7] We purposely choose a polynomial that cannot correctly approximate the true policy functions, since often the true functional form is ex-ante unknown. We check that the results are robust to many types of misspecification. However, the purpose of the following example is to simply illustrate the possibility that misspecification under multicollineatity can lead to biased predictions. This problem would not arise with a universal approximator, such as neural networks, because they do not require prespecification of the functional form.

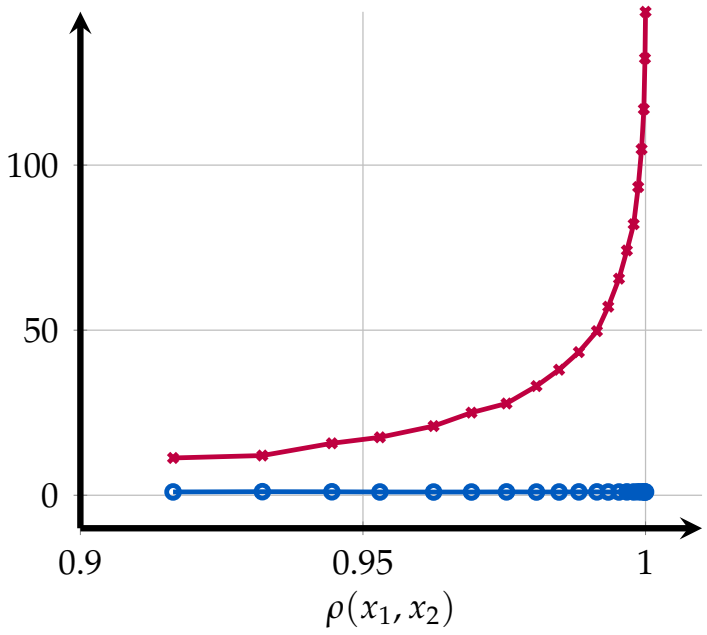

Figure 3: Mean squared prediction error with neural network and polynomial

Notes: The figure shows the means squared prediction error $1/n \sum_{t=1}^{n} [y_t - \mathbb{E}(\hat{y}_t)]^2$ in function of the correlation between x_1 and x_2. Blue line with circles - NN, purple line with crosses - polynomial regression.

The higher the correlation between the state variables, the higher the inaccuracy of the polynomial regression model. Moreover, if we decompose the MSPE using equation 2, we find that most of the prediction error comes from the bias-square term (see appendix B figures 8 and 9). Because of its non-parametric nature, the neural network adapts to the shape of the function to approximate without having to guess the functional form ex-ante. This experiment suggests that the non-parametric nature of a neural network is particular handy in solving economic models characterized by policy functions with functional forms that are ex-ante unknown and that potentially contain significant non-linearities and whose domain presents multicollinear states.[8] In the next section we illustrate the use of neural networks in a model that contains such features.

3 Model and Solution Method

The model we work with is an extension of the one bond economy analyzed in Aiyagari et al. (2002) and extended to two bonds in Faraglia et al. (2019). We work with this model for two reasons. First, it is a difficult computational problem that features a large multicollinear state-space with non-linearities difficult to approximate with a parametric approach (i.e., borrowing and lending

[8]Note that another option would be to specify a rich polynomial structure with many higher order and cross terms. One problem with such approach is that higher order terms of the same variable are extremely multicollinear.

constraints).[9] Second, extending Faraglia et al. (2019) to more than two maturities is a relevant economic problem since it helps in determining the optimal maturity structure of government debt. We start by introducing the reader to a one-bond economy with a single maturity of N periods. We then present our methodology in a general model with N maturities. The numerical advantages of our methodology allow us to explore the optimal maturity structure of government debt with three bonds. Quantitative results are presented in section 4.

3.1 Illustrative Model: One-Bond Economy

The economy is populated by a representative household with preferences over consumption c and leisure l. The representative household chooses sequences of consumption $\{c_t\}_{t=0}^{\infty}$ and leisure $\{l_t\}_{t=0}^{\infty}$ to maximize its time-0 expected lifetime utility

$$\mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t [u(c_t) + v(l_t)],$$

subject to the budget constraint

$$p_t^N b_t^N + c_t = (1 - \tau_t)(1 - l_t) + p_t^{N-1} b_{t-1}^N,$$

where b_t^N indicates an N-periods maturity bond and p_t^N is its corresponding price.[10] The only source of aggregate risk in the economy is an exogenous stream of government expenditures $\{g_t\}_{t=0}^{\infty}$. In each period, the government can finance g_t by: (i) levying a proportional labor tax τ_t and (ii) by issuing a non-state contingent bond with maturity of N periods. Hence, the government's budget constraint is:

$$g_t + p_t^{N-1} b_{t-1}^N = \tau_t(1 - l_t) + p_t^N b_t^N.$$

The aggregate resource constraint of the economy is $c_t + g_t = 1 - l_t$, where $1 - l_t$ is the period's GDP. We assume the government can buy back and reissue the entire stock of outstanding debt in each period. The government sets taxes and issues debt to solve a Ramsey taxation problem. We adopt the primal approach and assume the government's ability to borrow and lend is bounded.

[9] The non-parametric nature of a neural network makes it suited to approximating policy functions with strong non-linearities, which would be harder to capture with polynomials.

[10] In principle, households are able to trade government securities in the secondary market. However, since we assume households are identical, there is no trade in equilibrium and, for ease of notation, we omit these trades from the household's budget constraint.

Under these conditions, the government's problem is

$$\max_{\{c_t\}_{t=0}^{\infty},\{b_t\}_{t=0}^{\infty}} \mathbb{E}_0 \sum_t \beta^t \left[u(c_t) + v(1 - c_t - g_t)\right],$$

subject to a sequence of measurability constraints[11]

$$b_t^N \beta^N \mathbb{E}_t \left[u_{c,t+N}\right] - b_{t-1}^N \beta^{N-1} \mathbb{E}_t \left[u_{c,t-1+N}\right] - g_t u_{c,t} + (u_{c,t} - v_{l,t})(g_t + c_t) = 0,$$

with borrowing and lending limits[12]

$$\bar{M} \geq b_t^N, \quad \underline{M} \leq b_t^N.$$

The government's optimality conditions are

$$u_{c,t} - v_{l,t} + \mu_t (u_{cc,t} c_t + u_{c,t} + v_{ll,t}(c_t + g_t) - v_{l,t}) + u_{cc,t}(\mu_{t-N} - \mu_{t-N+1}) b_{t-N}^N = 0, \tag{3}$$

$$\mu_t = \mathbb{E}_t (u_{c,t+N})^{-1} \left[\mathbb{E}_t (u_{c,t+N} \mu_{t+1}) + \frac{\xi_{U,t}}{\beta^N} - \frac{\xi_{L,t}}{\beta^N}\right], \tag{4}$$

$$b_t^N \beta^N \mathbb{E}_t (u_{c,t+N}) = b_{t-1}^N \beta^{N-1} \mathbb{E}_t (u_{c,t+N-1}) - g_t u_{c,t} - (u_{c,t} - v_{l,t})(g_t + c_t), \tag{5}$$

where μ_t is the Lagrange multiplier on the time t measurability constraint, $\xi_{U,t}$ and $\xi_{L,t}$ are the Lagrange multipliers on the upper and the lower bounds, respectively. By issuing debt at time t, the government commits to increasing taxes and/or to reissuing debt at time $t + N$. When the government sets taxes between time t and time $t + N$, it needs to take into account its past actions in the form of all lags of the state variables up to N. More formally, the Ramsey planner's state space X_t is

$$X_t = \left\{g_t, \{\mu_{t-i}\}_{i=1}^N, \{b_{t-i}^i\}_{i=1}^N\right\}.$$

The state space contains $2N + 1$ variables, with many lags of the same state variable (e.g., μ), which tend to be highly correlated with each other. Moreover, equation 4 reveals that the Lagrange multiplier on the implementability constraint μ_t follows a random walk, creating an additional source of multicollinearity between the state variables. We solve the model with maturity $N = 10$, and we report in figure 4 the autocorrelation function of the simulated equilibrium bond's sequence $\{b_t^N\}$. It is clear that the previous 10 lags of the same variable, which are all part of the state space,

[11] See AMSS (2002) for details on how to use the recursive Lagrangian approach in this context.

[12] $\bar{M}_N \geq b_t^N$ is the government saving constraint, which is equivalent to a household's borrowing constraint.

are highly correlated with each other in the simulated sequence.

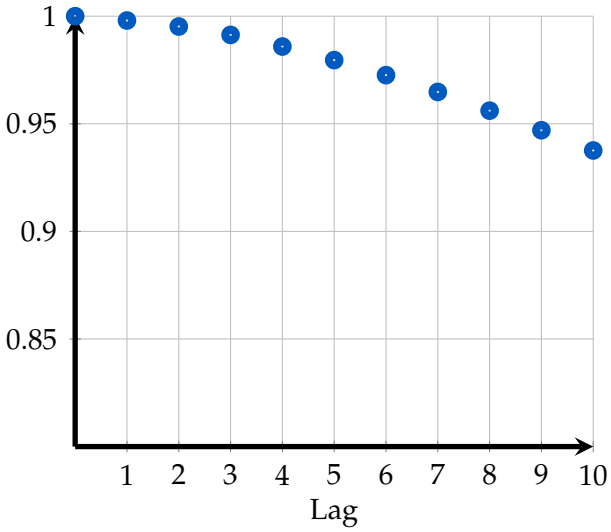

Figure 4: Autocorrelation function of the equilibrium bond sequence

Notes: The figure shows the autocorrelation function of b_t^N. The numbers are obtained after simulating the model equilibrium dynamics for T=800.

For this reason, the model is hardly solvable using PEA (algorithm 1). The multicollinearity problem poses a challenge and requires approximating the expected values in equations 3, 4, and 5 using functions of a subset X_t^C of the state space X_t (X_t^C is also called the core set). These approximations are $\mathbb{E}_t(u_{c,t+N}) \simeq P_1(X_t^C;\eta_1)$, $\mathbb{E}_t(u_{c,t+N-1}) \simeq P_2(X_t^C;\eta_2)$ and $\mathbb{E}_t(u_{c,t+N}\mu_{t+1}) \simeq P_3(X_t^C;\eta_3)$, where both the functions and the core set (including its cardinality) are ex-ante unknown. The subset X^C of the information set X is selected through an iterative procedure called Condensed PEA. In essence, this method adds an additional loop to PEA and keeps extracting orthogonal components from the state space, similarly to the Principle Component Analysis (PCA), but the number of factors does not have to be chosen ex-ante. A more detailed description of the procedure can be found in section 5 algorithm 4, where we compare our methodology to existing ones in the literature. In the next section, we present our methodology in a model with N maturities. Due to the presence of multiple lagged bonds, the multicollinearity problem is further accentuated.

3.2 Optimal Maturity Management with N Bonds

The economy is populated by a representative household with preferences over consumption c and leisure l. The representative household chooses sequences of consumption $\{c_t\}_{t=0}^{\infty}$ and leisure

$\{l_t\}_{t=0}^{\infty}$ to maximize its time-0 expected lifetime utility:

$$\mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \left[u(c_t) + v(l_t)\right],$$

subject to the budget constraint:

$$p_t^N \sum_{i=1}^{N} b_t^i + c_t = (1-\tau_t)(1-l_t) + \sum_{i=1}^{N} p_t^{i-1} b_{t-1}^i,$$

where b_t^i indicates an i-periods maturity bond and p_t^i is its corresponding price. The only source of aggregate risk in the economy is an exogenous stream of government expenditures $\{g_t\}_{t=0}^{\infty}$. In each period, the government can finance g_t by: (i) levying a proportional labor tax τ_t and (ii) by issuing non-state contingent bonds with maturity $1, \cdots, N$. The government's budget constraint reads

$$\sum_{i=1}^{N} p_{i-1,t} b_t^i = \tau_t h_t - g_t + \sum_{i=1}^{N} p_{i,t} b_{t+1}^i.$$

Sequential Formulation of the Ramsey Problem Combining the technology constraint, $c_t + g_t = h_t$, with the household's labor optimality condition, $1 - \tau_t = u_{l,t}/u_{c,t}$, yields an expression for surplus

$$s_t \equiv \tau_t h_t - g_t = c_t - (1-\tau_t) h_t = c_t - \frac{u_{l,t}}{u_{c,t}}(c_t + g_t).$$

Substitute bonds prices $p_{i,t}$, pinned down by the household's Euler equations, to get

$$\sum_{i=1}^{N} b_t^i \mathbb{E}_t \left[\beta^{i-1} \frac{u_{c,t+i-1}}{u_{c,t}}\right] = s_t + \sum_{i=1}^{N} b_{t+1}^i \mathbb{E}_t \left[\beta^i \frac{u_{c,t+i}}{u_{c,t}}\right],$$

with borrowing and lending limits[13]

$$\forall i: \quad \bar{M} \geq b_t^i, \quad \underline{M} \leq b_t^i, \quad \bar{M}_{\text{total}} \geq \sum_{i=1}^{N} b_t^i, \quad \underline{M}_{\text{total}} \leq \sum_{i=1}^{N} b_t^i.$$

[13] $\bar{M}_N \geq b_t^N$ is the government saving constraint, which is equivalent to a household's borrowing constraint.

The optimality conditions are

$$c_t : u_{c,t} - v_{l,t} + \mu_t \left[u_{c,t} - v_{l,t} + u_{cc,t} c + v_{ll,t}(c_t + g_t) \right] + \sum_{i=1}^{N} (\mu_{t-i} - \mu_{t-i+1}) b_{t-i}^i u_{cc,t} = 0,$$

$$\forall i,\ b_{t+1}^i : \mu_t = \left[\mathbb{E}_t u_{c,t+i} \right]^{-1} \left[\mathbb{E}_t \mu_{t+1} u_{c,t+i} + \frac{\xi_{U,t}^i}{\beta^i} - \frac{\xi_{L,t}^i}{\beta^i} + \frac{\xi_{U,t}^{\text{Total}}}{\beta^i} - \frac{\xi_{L,t}^{\text{Total}}}{\beta^i} \right],$$

$$\mu_t : \sum_{i=1}^{N} b_t^i \mathbb{E}_t \left[\beta^{i-1} \frac{u_{c,t+i-1}}{u_{c,t}} \right] = s_t + \sum_{i=1}^{N} b_{t+1}^i \mathbb{E}_t \left[\beta^i \frac{u_{c,t+i}}{u_{c,t}} \right],$$

where $\xi_{U,t}$ and $\xi_{L,t}$ are the Lagrange multipliers on the upper and the lower bounds, respectively, and $\xi_{U,t}^{\text{Total}}$ and $\xi_{L,t}^{\text{Total}}$ are the Lagrange multipliers on the upper and the lower bounds on the total bond portfolio. In the following section we describe in detail our computational strategy. Details on the implementation and results using Epstein-Zin preferences can be found in appendix A.

3.3 NN-based Expectations Algorithm

In this section we describe the main algorithm, which is an extension of the basic idea illustrated in section 2.5, applied to an optimal fiscal policy model with incomplete markets and multiple maturities. Here we present the key steps, while implementation details can be found in appendix C. There are N bonds available with maturities from 1 to N periods. The state space at time t is $\mathcal{I}_t = \{g_t, \{\{b_{t-k}^i\}_{k=0}^{N-1}\}_{i=1}^{N}, \{\mu_{t-k}\}_{k=1}^{N}\}$. The neural network needs to approximate $\mathbb{E}_t\left[u_{c,t+i}\right]$, $\mathbb{E}_t\left[\mu_{t+i} u_{c,t+i}\right]$ and $\mathbb{E}_t\left[u_{c,t+i-1}\right]$ in function of $\mathcal{I}_t$. We model these relationships using one single-layer neural network $\mathcal{ANN}(\mathcal{I}_t)$. In particular, if the long maturity is $N > 1$, then the terms to approximate are

$$\mathcal{ANN}_1^i(\mathcal{I}_t) = \mathbb{E}\left[u_{c,t+i} | \mathcal{I}_t\right] \quad \text{for } i = [1...N],$$
$$\mathcal{ANN}_2^i(\mathcal{I}_t) = \mathbb{E}\left[\mu_{t+i} u_{c,t+i} | \mathcal{I}_t\right] \quad \text{for } i = [1...N],$$
$$\mathcal{ANN}_3^i(\mathcal{I}_t) = \mathbb{E}\left[u_{c,t+i-1} | \mathcal{I}_t\right] \quad \text{for } i = [1...N].$$

For example, in the two-bond case there are six terms to approximate and, if the short bond has 1 period maturity, they reduce to five.[14] Given starting values for μ_t and $\{b_t^i\}_{i=1}^N$ and initial weights for $\mathcal{ANN}$, simulate a sequence of $\{c_t\}_{t=1}^T$, $\{\mu_t\}_{t=1}^T$ and $\{\{b_t^i\}_{i=1}^N\}_{t=2}^{T+1}$ as follows.[15]

[14] The six terms are $\mathbb{E}_t(u_{c,t+N})$, $\mathbb{E}_t(u_{c,t+N-1})$, $\mathbb{E}_t(u_{c,t+N-1}\mu_{t+1})$, $\mathbb{E}_t(u_{c,t+S})$, $\mathbb{E}_t(u_{c,t+S-1})$, $\mathbb{E}_t(u_{c,t+S}\mu_{t+1})$, and the term that does not require approximation in the latter case is $\mathbb{E}_t(u_{c,t+S-1})$ is just $u_{c,t}$. S and N denote bond maturities.

[15] The network can be initially trained using an educated guess for $\{b_t^i\}_{i=1}^N, c_t, \mu_t$. It is important that the initial training sequence is not constant. More details can be found in appendix C.

1. Impose the Maliar moving bounds on all debts instruments, see Maliar and Maliar (2003). These bounds are particularly important and need to be tight and open slowly. Proper penalty functions are used instead of the ξ terms to avoid out of bound solutions.[16] Since μ_t is identified by the first order condition for b_t^i, it is over-identified if the number of available maturities is greater than one.

$$\forall i: \quad \mu_t = \mathcal{ANN}_1^i(\mathcal{I}_t)^{-1}\left[\mathcal{ANN}_2^i(\mathcal{I}_t) + \frac{\xi_{U,t}^i}{\beta^i} - \frac{\xi_{L,t}^i}{\beta^i} + \frac{\xi_{U,t}^{\text{Total}}}{\beta^i} - \frac{\xi_{L,t}^{\text{Total}}}{\beta^i}\right].$$

We tackle this problem by using the forward-states approach described in Faraglia et al. (2019). This involves approximating the expected value terms at time $t+i$ with a function of the state variables that are relevant at $t+1$ instead of t and invoking the law of iterated expectations, such that we use $\mathbb{E}_t\mathcal{ANN}_1^i(\mathcal{I}_{t+1})$ instead of $\mathcal{ANN}_1^i(\mathcal{I}_t)$.[17]

2. To perform the stochastic simulation, choose T big enough and find $\{c_t\}_{t=0}^T$, $\{\mu_t\}_{t=0}^T$ and $\{\{b_{t+1}^i\}_{i=1}^N\}_{t=0}^T$ that solve the following system of $(N+2)T$ equations:

$$\begin{cases} \forall i: \quad \mu_t = \left[\mathbb{E}_t\mathcal{ANN}_1^i(\mathcal{I}_{t+1})\right]^{-1}\left[\mathbb{E}_t\mathcal{ANN}_2^i(\mathcal{I}_{t+1}) + \frac{\xi_{U,t}^i}{\beta^i} - \frac{\xi_{L,t}^i}{\beta^i} + \frac{\xi_{U,t}^{\text{Total}}}{\beta^i} - \frac{\xi_{L,t}^{\text{Total}}}{\beta^i}\right], \\ u_{c,t} - v_{l,t} + \mu_t\left[u_{c,t} - v_{l,t} + u_{cc,t}c + v_{ll,t}(c_t+g_t)\right] + \sum_{i=1}^N(\mu_{t-i} - \mu_{t-i+1})b_{t-i}^i u_{cc,t} = 0, \\ \sum_{i=1}^N b_t^i\beta^{i-1}\mathbb{E}_t\mathcal{ANN}_3^i(\mathcal{I}_{t+1}) = U_{c,t}s_t + \sum_{i=1}^N b_{t+1}^i\beta^i\mathbb{E}_t\mathcal{ANN}_1^i(\mathcal{I}_{t+1}). \end{cases} \tag{6}$$

3. If the solution error in the stochastic simulation is large, or a reliable solution could not be found, the algorithm automatically restores the previous period neural network and performs the stochastic simulation with a reduced Maliar bound.[18]

4. If the solution calculated shrinking the bound at iteration $i-1$ is not satisfactory, the algorithm does not go back another iteration but uses the same neural network and tries to lower the $Bound_{i-1}$ again towards $Bound_{i-2}$. Once a reliable solution is found, the algorithm proceeds to calculate the solution for iteration i again, but with $Bound_i = Bound_{i-1} + (Bound_{i-1} - Bound_{i-2})$. In this way, if an error is detected multiple times we guarantee that both $Bound_i$

[16] We also find that including ξ terms explicitly in the training set improves prediction accuracy. More details can be found in appendix C.

[17] A more detailed explanation of forward-states approach can be found in appendix C.

[18] If the unreliable solution has been detected in iteration i the algorithm restores iteration $i-1$ environment and performs the stochastic simulation with $Bound_{i-1} = \alpha Bound_{i-1} + (1-\alpha)Bound_{i-2}$.

and $Bound_{i-1}$ keep shrinking toward $Bound_{i-2}$ and there must exist a point close enough to $Bound_{i-2}$ such that the system can be reliably solved with both $Bound_{i-1}$ and $Bound_i$.

5. If the solution found at iteration i is satisfactory, the neural network enters the learning phase supervised by the implied model dynamics, the Maliar bounds are increased, and a new iteration starts again.

Keep repeating until the neural network predictions converge and the simulated sequences of $\{b_t^i\}_{i=1}^N$, and c_t do not change.[19] Algorithm 3 describes the algorithm in greater detail.

4 Numerical Results

In this section we use the NN-based Expectations Algorithm to study the welfare effects of additional maturities in a government debt management problem presented in section 3. Specifically, we are interested in the welfare effects arising from the additional hedging opportunities. We first present the calibration and then inspect how additional maturities influence household welfare and government finances.

4.1 Calibration

We calibrate the model following the strategy of Faraglia et al. (2019). Specifically, we use additively separable utility in consumption and leisure

$$U(c,l) = \frac{c^{1-\gamma}}{1-\gamma} + \chi \frac{l^{1-\eta_l}}{1-\eta_l},$$

with $\gamma = 1.5$ and $\eta_l = 1.8$, respectively. We calibrate χ such that households spend on average 2/3 of their time endowment on leisure in the steady state, which gives the value of 2.87.

We set β to 0.96 and for the sake of comparison we follow the calibration strategy for g_t from Faraglia et al. (2019). We assume that g_t follows an AR(1) process $g_t = \mu_g + \rho_g g_{t-1} + \epsilon_t$, $\epsilon_t \sim N(0, \sigma_g^2)$ with ρ_h equal to 0.95. Then we look for the value of μ_g such that government expenditure is on average equal to 25% of GDP. This gives the value of 0.0042. Lastly, we set the value for σ_g such that g_t is always at least 15% and at most 35% of GDP in a simulated sample of 10 thousand periods, which gives the value of 0.0031. Note that such parameterization is also broadly

[19] There is no need to check μ_t, which can be backed out analytically from the first order condition for c_t.

Algorithm 3 NN-Based Expectations Algorithm applied to Optimal Maturity Management

Precondition: parameters from table 1; utility functions $u(c) = \frac{c^{1-\gamma}}{1-\gamma}$, $u_c(c) = c^{-\gamma}$, $i = 0$, $\text{Bound}(0) = 0$.

1: ▷ *Simulate log AR(1) process*
2: $\log(g_{t+1}) \leftarrow \mu_g + \rho_g \cdot \log(g_t) + \epsilon_{g+1}$

3: ▷ *Create and train the NN using initial conditions*
4: $\text{Net} \leftarrow \text{feedforwardnet}(\text{Num. Neurons})$

5: ▷ *Solve the model*
6: **while** $\text{Bound}(i) < B_{max} \quad OR \quad \text{OutofBoundIter} < \text{NumOutofBound}$ **do**
7: ▷ *Generate* $\{c_t\}_{t=1}^{T}$, $\{\mu_t\}_{t=1}^{T}$, *and* $\{\{b_{t+1}^{i}\}_{i=1}^{N}\}_{t=1}^{T}$
8: **for** $t \leftarrow 1$ to T **do**
9: **for** r $\leftarrow 1$ to maxrep **do**
10: $x_{\text{guess}} \leftarrow \{c(r)_{\text{guess}}, b(r)_{\text{guess}}^{1}, \cdots, b(r)_{\text{guess}}^{N}\}$
11: $\{c_t(r), \mu_t(r), \{b_{t+1}^{i}(r)\}_{i=1}^{N}, \text{residuals}(r)\} \leftarrow$ Solve system 6 given $\mathcal{ANN}(\mathcal{I}_{t+1})$, $\text{Bound}(i)$ and x_{guess}
12: **end for**
13: $r^* \leftarrow \min_r \quad \text{residuals}(r)$
14: $\{c_t, \mu_t, \{b_{t+1}^{i}\}_{i=1}^{N}\} \leftarrow \{c_t(r^*), \mu_t(r^*), \{b_{t+1}^{i}(r^*)\}_{i=1}^{N}\}$
15: **end for**
16: **if** $\text{residuals}(r^*) >$ threshold **then** Restart from line 7 with a smaller bound
17: **end if**
18: ▷ *Train the NN using the new simulated sequences*
19: $\mathcal{I}_t \leftarrow \{g_t, \{\{b_{t-k}^{i}\}_{k=0}^{N-1}\}_{i=1}^{N}, \{\mu_{t-k}\}_{k=1}^{N}\}$
20: $\text{RHS}_{1,t}^{i} \leftarrow u_{c,t+i} \quad \text{for } i = [1...N]$
21: $\text{RHS}_{2,t}^{i} \leftarrow \mu_{t+i} u_{c,t+i} \quad \text{for } i = [1...N]$
22: $\text{RHS}_{3,t}^{i} \leftarrow u_{c,t+i-1} \quad \text{for } i = [1...N]$
23: $\text{Net} \leftarrow \text{train}(\text{Net}, \mathcal{I}_{t+1}, \text{RHS}_t)$
24: ▷ *Checking convergence and updating* $\{b_{old,t}^{i}\}_{i=1}^{N}$ and $c_{\text{old},t}$
25: $\text{error}_b \leftarrow \max(|\{b_{\text{old},t}^{i}\}_{i=1}^{N} - \{b_t^{i}\}_{i=1}^{N}|)$
26: $\text{error}_c \leftarrow \max(|c_{\text{old},t} - c_t|)$
27: **if** $\max(\text{error}_b, \text{error}_c) < \epsilon$ **then** Break
28: **end if**
29: $\{b_{\text{old},t}^{i}\}_{i=1}^{N} \leftarrow \{b_t^{i}\}_{i=1}^{N}$
30: $c_{\text{old},t} \leftarrow c_t$
31: $\text{Bound}(i) \leftarrow \text{Bound}(i) + \text{BoundStep}$
32: **if** $\text{Bound}(i) > \bar{M}$ **then**
33: $\text{Bound}(i) \leftarrow \bar{M}$
34: $\text{OutofBoundIter} \leftarrow \text{OutofBoundIter} + 1$
35: **end if**
36: $i \leftarrow i + 1$
37: **end while**

aligned with the estimates from the data.[20]

The government has three debt instruments at its disposal. We set maturities to S=1, M=5, and N=10 and denote them as short, medium, and long bonds, respectively. In addition to debt limits on individual bonds, we introduce a total debt limit of ±125% of GDP both in our benchmark model with only short and long bonds and in our calibration with three bonds. A fixed limit on total debt allows us to make a fair comparison and isolate the effects of the hedging benefits of the additional bond on the household's welfare. Table 1 summarizes the parameter values.

		Parameter	Value
Preferences	Discount factor	β	0.96
	Risk aversion	γ	1.5
	Labor disutility	χ	2.87
	Leisure curvature	η_l	1.8
Government	Average g_t	μ_g	0.0042
	Volatility of g_t	σ_g	0.0031
	Autocorr. of g_t	ρ_g	0.95
	Debt limits	$\bar{M}, \underline{M}, \bar{M}_{\text{total}}, \underline{M}_{\text{total}}$	± 125% of GDP

Table 1: Calibrated parameters

Before proceeding, it is worth noting that we tested our methodology with the two-bond case. We replicated the results of Faraglia et al. (2019), where the optimal debt portfolio includes a negative short bond position and a positive long bond position, as shown in table 2. Moreover, as shown in table 3, the dynamics of the two bonds are highly negatively correlated and, like in Buera and Nicolini (2004) the positions are large and volatile. It is optimal for the government to borrow using the long-term bond and lend using the short-term bond as in Angeletos (2002).

4.2 Optimal Debt Management with Three Bonds

Tables 2 reports the equilibrium outstanding debt-to-GDP ratio for each maturity and for each model with an increasing number of bonds. Moments are calculated given a sequence of government expenditure shocks with persistence and volatility specified in table 1.

[20] We obtain very similar estimates using the sum of government consumption and gross investment from the NIPA tables.

Model	$\mathbb{E}(b^S/GDP)$	$\mathbb{E}(b^M/GDP)$	$\mathbb{E}(b^L/GDP)$	$\sigma(b^S/GDP)$	$\sigma(b^M/GDP)$	$\sigma(b^L/GDP)$
1 Bond	0.000	-	-	0.564	-	-
2 Bonds	-0.026	-	0.353	0.1	-	0.124
3 Bonds	-0.784	1.01	0.894	0.476	0.283	0.425

Table 2: Selected Bond Moments: Means and Variances

Notes: The table shows the average outstanding debt for each maturity. Moreover, the table also reports the standard deviations of each outstanding position.

Model	$\rho(g_t, b_t^S)$	$\rho(g_t, b_t^M)$	$\rho(g_t, b_t^L)$	$\rho(b_t^S, b_t^M)$	$\rho(b_t^S, b_t^L)$	$\rho(b_t^L, b_t^M)$
1 Bond	0.431	-	-	-	-	-
2 Bonds	0.686	-	-0.598	-	-0.918	-
3 Bonds	0.487	-0.325	-0.5	-0.962	-0.994	0.954

Table 3: Selected Bond Moments: Correlations

Notes: The table shows the correlations between each maturity of outstanding debt and government expenditure. Moreover, the table also reports the cross-correlations among the bonds.

As shown in tables 2 and 3, the optimal policy includes a positive amount of medium-term bonds, which is also positively correlated with the long bonds. When the number of bonds increases from two to three, long and short bonds become even more negatively correlated. As shown in table 4, the additional hedging benefits of the third bond are reflected in an increase in welfare of more than 2%, compared to the two-bond model.

1 Bond	2 Bonds	3 Bonds
0%	1.49%	3.91%

Table 4: Welfare gains

Notes: The table shows the welfare gains in two and three bonds models calculated in terms of a consumption equivalent with respect to the one-bond economy.

In terms of allocations, table 5 reveals that the additional increase in welfare is linked to a higher average leisure and a lower consumption volatility, while the economy sustains a lower average consumption. Labor tax volatility and autocorrelation also decrease significantly, while the average level rises.

Model	$\mathbb{E}(c_t)$	$\mathbb{E}(l_t)$	$\sigma(c_t)$	$\sigma(l_t)$	$\sigma(\tau_t)$	$\rho(\tau_t, \tau_{t-1})$	$\mathbb{E}(\tau_t)$
1 Bond	0.252	0.666	0.008	0.006	0.037	0.992	0.246
2 Bonds	0.25	0.667	0.007	0.003	0.026	0.923	0.257
3 Bonds	0.245	0.672	0.006	0.004	0.021	0.891	0.286

Table 5: Allocations and Policies

Notes: The table shows the effects of the optimal policy on consumption and leisure as the number of bonds increases.

Next, we inspect the economic mechanism that links the hedging benefit provided by the additional bond to the increase in the household's welfare. As known since Angeletos (2002), differences in long and short bond prices provide a tool to hedge against shocks by borrowing in long bonds and accumulating assets in the short term. Since long prices are more volatile than short prices, when a negative shock hits, the value of government liabilities falls more than the value of government assets, thus providing insurance against negative shocks. In addition to decreasing the government liability, the differential response of long and short prices also affects the terms of issuing new debt. Since long prices fall more than the shorter ones, it becomes cheaper for the planner to obtain funds by issuing shorter debt. This is why we observe portfolio rebalancing and a negative correlation between the long and short bonds.

Table 6 shows how optimal debt management affects government finances as we increase the number of debt instruments.

	1 Bond	2 Bonds	3 Bonds
$\rho(\text{Debt/GDP}, g_t)$	0.429	-0.15	-0.34
$\rho(\text{Net Financial Income}, g_t)$	0.162	0.342	0.706
$\rho(\text{Net Financial Income (constant price)}, g_t)$	0.037	0.067	-0.025
Av. Net Financial Income	0.151	-0.843	-3.449
Av. Labor Tax Income	24.717	25.846	28.801

Table 6: Government Income and Borrowing

Notes: The table shows selected moments from the models with one, two, and three maturities. First row shows the correlation between the outstanding debt/GDP ratio and expenditure shocks. Rows two and three show the correlation between government financial income and expenditure shocks. The last two rows show the average net financial income and the average labor tax income. Net Financial Income is defined as the inflow from issuing new debt at the net of the cost of buying back the outstanding debt. Net Financial Income (constant price) is the counterfactual and it corresponds to Net Financial Income holding bond prices fixed at their average values.

First, we decompose government income into labor tax income and net financial income, which is the inflow from issuing new bonds minus the outflow due to outstanding debt. Most importantly, as the number of maturities increases, the correlation between total debt and government expenditures changes sign, as shown in the first row of table 6.

In the one- and two-bond economy, the government borrows from the private sector to finance expenditure shocks. In the three-bond economy, the government reduces its total debt to subsidize the private sector and smooth its consumption. At the same time, net financial income becomes even more positively correlated with g_t and allows for smoother labor taxes, despite a falling total debt in bad times. Reduction of total debt together with rising financial income is achieved precisely because the planner holds leveraged positions and responds to expenditure shocks by

substituting to short bonds.

As further evidence of this mechanism, we construct a counterfactual measure of net financial income assuming that bond prices were fixed at their mean values. The counterfactual correlation is reported in the third row of table 6. As the number of optimally managed bonds increases, the correlations in the third row decrease. This suggests that the co-movement between net financial income and government expenditures is achieved by exploiting the differential response of short, medium, and long prices.

Looking at the averages in rows four and five, we see that as the number of maturities increases, the government becomes a net payer to the private sector and collects a larger share of its income in labor taxes. This happens because the increase in labor taxes outweights the decrease in average labor supply. Although average household labor income falls, the household gets compensated for holding government debt.

5 Comparison with Alternative Methods

There are other simulation-based numerical methods designed to address the issue of multicollinearity among state variables. In this section we discuss and compare our method to the two most prominent ones: the Condensed PEA used in Faraglia et al. (2019) and GSSA, described in Judd et al. (2011).

5.1 Relation to Condensed PEA

This method extracts orthogonal components from the information set, similarly to the Principle Component Analysis (PCA), but the number of factors does not have to be chosen ex-ante. As a brief reminder to the reader, here is a high level description of the algorithm, see Faraglia et al. (2014) for more details.

1. Parameterize the expectations in the optimality conditions as functions of a subset of the state space (called the core set), given an initial guess of the polynomial parameters.[21]

    - Set the bounds for the bonds (Maliar and Maliar, 2003).
    - Simulate the model given the parameters.
    - Use the simulated dynamics to update the parameters' values.

[21] Initial parameters can be given by a simulated sequence with $\{b_t\}_{t=0}^{T}$.

- Iterate and stop when the predictions and simulated sequences converge.

2. Regress the remaining state variables on the core set and save the residuals. Then regress the realized values of the approximated expectations on the core set and the saved residuals. Add these residuals, multiplied by the estimated coefficients, to the core set and go back to point 1 till convergence on the path of debt is reached.

Algorithm 4 reports the pseudo-code of the algorithm and highlights with colors the part of Condensed PEA that changes when the NN-based Expectations algorithm is implemented. In particular, we remove the external loop (lines 1, 13, 14, 15, 16, and 17) as the neural network digests the information set at once.

Algorithm 4 From Condensed PEA to NN-based EA

Precondition: initial state X_0, initial $X^{C,1}$ and X^{Out}, sequence $\{\xi_t\}_{t=0}^{T}$, initial guess η_n^1, dampening $0 < w < 1$

1: **while** core set $X^{C,k}$ converges **do**
2: **while** η_n^i converges **do**
3: **for** $t \leftarrow 0$ to T **do** ▷ Generate $X \equiv \{X_t\}_{t=0}^{T}$
4: $c_t \leftarrow f^{-1}(P_n(X_t^{C,k}; \eta_n))$
5: $X_{t+1} \leftarrow h(X_t, c_t, \xi_{t+1})$
6: **end for**
7: **for** $t \leftarrow 0$ to T **do** ▷ Generate $Y \equiv \{y_{t+1}\}_{t=0}^{T}$
8: $y_{t+1} \leftarrow g(c_{t+1}, X_{t+1})$
9: **end for**
10: $\hat{\eta}_n^i \leftarrow \underset{}{\text{argmin}}\ (Y - P_n(X^{C,k}; \eta_n^i)^2$ ▷ Regress to find new weights
11: $\eta_n^{i+1} \leftarrow w \cdot \hat{\eta}_n^i + (1 - w) \cdot \eta_n^i$ ▷ Update with dampening
12: **end while**
13: $\omega \leftarrow \text{argmin}\ X^{\text{Out}} - \omega X^{C,k}$ ▷ Update core set
14: $X^{\text{Res}} \leftarrow X^{\text{Out}} - \omega X^{C,k}$
15: $\lambda \leftarrow \text{argmin}\ Y - P_n(X^{C,k}; \eta_n^i) - \lambda X^{\text{Res}}$
16: $X^{C,k+1} \leftarrow X^{C,k} \cup \lambda X^{\text{Res}}$
17: **end while**

On the one side, eliminating the external loop (line 1) reduces the time complexity of the algorithm significantly, since Condensed PEA requires testing an unknown number of combinations of core regressors. On the other side, our algorithm requires substituting OLS (lines 10 and 11) with a neural network training algorithm, which has a higher time complexity.

Recall that S also corresponds to the number of neurons in the input layer, M is the number of neurons in the hidden layer, E is the number of expectations to approximate (which also corresponds to the number of neurons in the output layer), and T is the number of training examples.

Time Complexity of Condensed PEA In a least squares regression, the matrix multiplication ($X^T X$) dominates asymptotically the Cholesky factorization of $X^T X$.[22] Hence, the time complexity to approximate E expectations terms with OLS is $\mathcal{O}(E \cdot S^2 \cdot T)$. In the condensed PEA, this operation is repeated for an unknown number of iterations n_{CPEA}. We estimate the time complexity of the Condensed PEA as

$$\mathcal{O}(n_{\text{CPEA}} \cdot n_{\text{InternalLoop}} \cdot (\kappa(E, S, T) + E \cdot S^2 \cdot T)),$$

where $\kappa(E, S, T)$ indicates the time complexity to compute lines 3-9 and $n_{\text{InternalLoop}}$ is the unbounded number of iterations required for model convergence (while-loop in line 2).

Time Complexity of NN-based Expectations algorithm The time complexity of a single iteration of back-propagation to train a neural network that has 3 layers (with S, M, and E nodes) is $\mathcal{O}(T \cdot (S \cdot M + M \cdot E))$. Assuming there are more neurons in the hidden layer than the number of expectations to approximate ($M > E$) and n_{NN} is the number of epochs (which is in principle unbounded), we estimate the time complexity to train a neural network as $\mathcal{O}(n_{NN} \cdot T \cdot S \cdot M)$. We estimate the time complexity of the NN-based Expectations algorithm as

$$\mathcal{O}(n_{\text{InternalLoop}} \cdot (\kappa(E, S, T) + n_{NN} \cdot T \cdot S \cdot M)).$$

Given our estimates, our algorithm has a better time complexity when[23]

$$n_{\text{InternalLoop}} \cdot (\kappa(E, S, T) + n_{NN} \cdot T \cdot S \cdot M) < n_{\text{CPEA}} \cdot n_{\text{InternalLoop}} \cdot (\kappa(E, S, T) + E \cdot S^2 \cdot T).$$

After rearranging, we get a simplified expression

$$\underbrace{n_{NN} \cdot T \cdot S \cdot M}_{\text{NN training}} < (n_{\text{CPEA}} - 1) \cdot \kappa(E, S, T) + \underbrace{n_{CPEA} \cdot E \cdot S^2 \cdot T}_{\text{CPEA regression}}. \tag{7}$$

The left hand side captures the time complexity of training the neural network. The right hand side contains the term that captures the time complexity of the regressions involved in the Condensed PEA and an additional term $(n_{\text{CPEA}} - 1) \cdot \kappa(E, S, T)$ that captures the difference between the complexity of the stochastic simulation as computed with Condensed PEA and NN-based Expectations algorithm. While $\kappa(E, S, T)$ is unknown, we can still compare the $n_{NN} \cdot T \cdot S \cdot M$ and

[22] We assume that the number of features (state variables) is smaller than the number of samples. That is, $T > S$.
[23] We assume that $n_{\text{InternalLoop}}$ and $\kappa(E, S, T)$ are the same in both methods.

$n_{CPEA} \cdot E \cdot S^2 \cdot T$ terms, which further reduces to comparing $n_{NN} \cdot M$ with $n_{CPEA} \cdot E \cdot S$. In our application $S = 27$, $E = 8$, $M = 10$, and $n_{NN} = 20$ on average. Given these numbers, the two algorithms have comparable complexity when $n_{CPEA} = 1$.[24] When $n_{CPEA} > 1$, the inequality in 7 clearly holds since $(n_{\text{CPEA}} - 1) \cdot \kappa(E, S, T) > 0$.

5.2 Relation to GSSA

Judd et al. (2011) propose a related method called generalized stochastic simulation algorithm (GSSA) to deliver high accuracy predictions as well as resolving the multicollinearity problem. Judd et al. (2011) resolve the multicollinearity problem using standard econometric techniques, such as single value decomposition (SVD), principal components or ridge regression. At the same time, high accuracy is achieved by approximating the policy functions and integrating them using Gauss-Hermite quadrature, instead of approximating the whole expectation terms in the optimality conditions. While the GSSA method has multiple advantages and has been successfully applied in different contexts, we find that its application to the Ramsey problem analyzed in this paper poses challenges.

First, in this application it is particularly challenging to approximate the policy functions directly, because the expectations are over N periods ahead. In the context of the model presented in section 2 of this paper, the evaluation of $\mathbb{E}[u'(c_{t+1})]$ requires knowing K_{t+2}, which is a function of K_{t+1}, which itself is a function of K_t and z_t, $K_{t+1} = \phi(K_t, z_t)$. That is, $c_{t+1} = z_{t+1}\phi(K_t, z_t)^\alpha + (1-\delta)\phi(K_t, z_t) - \phi(\phi(K_t, z_t), z_{t+1})$. When the expectation is N periods ahead, the evaluation of $\mathbb{E}[u'(c_{t+N})]$ using the GSSA approach would require iterating on the approximated policy functions and the budget constraint N times to have the value for K_{t+N+1}, which would result in an imprecise evaluation of the expectations and lower stability of the algorithm compared to an application where the forecast is one period ahead.

Second, methods such as ridge regression impose a penalty on the size of the coefficients, providing stability but causing them to be downward biased. The choice of this penalty parameter is the source of instability. It is particularly hard to choose the penalty parameter in the context of solving the model, since the regression is performed on simulated data which are not fixed and not exogenous from the choice of the penalty. Simulated data depend on the coefficients and penalties obtained in the previous iteration of the algorithm. Typically the choice of penalty parameter

[24] Note that when $n_{CPEA} = 1$, the Condensed PEA is essentially a standard PEA algorithm. It is possible that the Condensed PEA converges in one iteration but that requires a good guess of the initial set of core state variables, which needs to be found through trial and error.

requires adding an additional loop and solving the model with multiple values. However, it is not obvious which penalty parameter is optimal because the optimal penalty is different at every step of the PEA iteration as the Maliar bounds become increasingly open.

In order to compare our method with GSSA and other standard econometric techniques, in this section we solve the one-bond model from section 3, with short-term debt $N = 1$. We solve it with standard PEA, except that we use ridge regression. In order to pick the penalty parameter we use a cross-validation approach, where we change the penalty dynamically at each step of the fixed point iteration in the regression stage. We select the penalty parameter that minimizes the mean squared prediction error between predicted and simulated sequences.[25] Equation 8 illustrates the penalty selection procedure.

$$\begin{aligned} &\min_{\kappa} ||Y - X\hat{\boldsymbol{\beta}}||_2^2 \quad \text{s. t.} \\ &\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||Y - X\boldsymbol{\beta}||_2^2 + \kappa||\boldsymbol{\beta}||_2^2 \end{aligned} \tag{8}$$

We use ridge regression as opposed to SVD or a principal component analysis for two reasons. First, ridge regression is supposed to work better than SVD when the multicollinearity is severe, as is the case in the model of section 4. Second, we do not use principal component analysis, since the Condensed PEA effectively does the same extraction of orthogonal components, just iteratively.

From our numerical experiments, we discover that PEA combined with ridge regression converges only under specific conditions. Note that throughout the paper we have been using debt limits. This effectively introduces an occasionally binding constraint, which makes the multicollinearity problem even more severe if the debt sequence visits the constrained region frequently. Besides, the algorithm requires using Maliar bounds, which potentially cause even more instability since the borrowing constraint changes as the bounds progressively open. We find this to be crucially important. For illustration we consider the one-bond model with tight ($\bar{M}, \underline{M}$ at $\pm 100\%$ of GDP) and loose ($\bar{M}, \underline{M}$ at $\pm 200\%$) borrowing constraints. At every iteration we compute the maximum difference between the ridge regression coefficients at the current and the previous iteration. When the borrowing constraint is loose, the regression coefficients stabilize after the Maliar bounds are wide enough and the borrowing constraint stops binding. In contrast, in the specification with tight constraints, the constraint binds more often, requiring the use of a large

[25] Similarly, Judd et al. (2011) choose the smallest penalty that ensures numerical stability of the fixed point iteration, which also provides a high accuracy solution.

penalty parameter, which prevents the algorithm from converging.

	Tight Constraint	Loose Constraint
$\mathbb{E}(\Delta\beta)$	3.81	.08
% constraint binds	38.2%	0

Table 7: Solution using ridge regression

Notes: Table shows selected statistics from model specifications with tight and loose borrowing constraints when solving the model using PEA with Ridge regression. Tight refers to the case when $\bar{M}, \underline{M}$ at $\pm 100\%$ and loose refers to the case when $\bar{M}, \underline{M}$ at $\pm 200\%$. First row shows the average change in the Ridge coefficients in the last five iterations. Second row shows the percentage of time the bond hits the constraint. The specification with loose constraints converges in 198 iterations. The specification with tight constraints never converges, therefore, we stop the code at iteration 198.

Table 7 illustrates this point. The first row shows the average change in ridge coefficients across consecutive PEA iterations for the last 50 iterations. The second row shows the percentage of time that debt visits the constraint in the last iteration. In the specification with tight borrowing constraint, the bond stays around 43% of the time close to the constraint and ridge coefficients never stabilize. Alternatively, this can be seen in figure 5, which plots the total model prediction error across the PEA iterations as the Maliar bound is being opened. In the specification with loose constraints, the forecast errors begin to stabilize when the Maliar bound stops changing and the algorithm slowly converges. In contrast, when constraints are tight, the ridge penalty parameter keeps changing and the forecast errors never stabilize and remain large (see table 8).

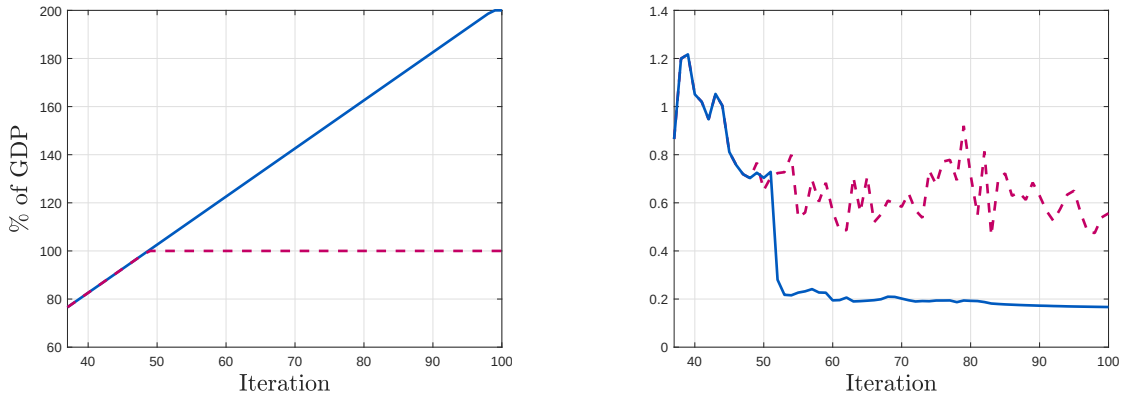

Figure 5: Convergence of model forecast errors using ridge regression

Notes: The figure shows the convergence of the model using ridge regression. Left panel shows the value of Maliar bound in function of the algorithm iteration. Right panel shows the total model forecast error in function of the algorithm iteration. Solid blue line - loose borrowing constraint. Dashed purple line - tight borrowing constraint. Tight constraint refers to the case when $\bar{M}, \underline{M}$ at $\pm 100\%$ and loose constraint refers to the case when $\bar{M}, \underline{M}$ at $\pm 200\%$.

	$\mathbb{E}_t(u_{c,t+N}\mu_{t+1})$	$\mathbb{E}_t(u_{c,t+N})$	$\mathbb{E}_t(u_{c,t+N-1})$
Tight Constraint	0.1634	0.2786	0.2664
Loose Constraint	0.0117	0.0679	0.0668

Table 8: Prediction accuracy using ridge regression

Notes: The table shows the average absolute forecast errors from model specifications tight and loose borrowing constraint when solving them using PEA with Ridge regression. Tight refers to the case when $\bar{M}, \underline{M}$ at $\pm 100\%$ and loose refers to the case when $\bar{M}, \underline{M}$ at $\pm 200\%$. First row shows the average change in the Ridge coefficients in the last five iterations. The specification with loose constraints converges in 198 iterations. The specification with tight constraints never converges, therefore, we stop the code at iteration 198. We define the average absolute forecast error as $\frac{1}{T}\sum |Y_{t+N} - \hat{E}_t(Y_{t+N})|$.

We have also attempted to solve the two-bond model using ridge regression but in this case, the problem of instability becomes even more severe as the long and the short bonds are negatively correlated and highly volatile. As a result, bonds hit the constraint very frequently and the algorithm fails to converge even before the Maliar bounds are open.[26]

6 Conclusion

This paper introduces a new stochastic simulation method to solve large state space macro-finance models. In particular, we use a neural network to approximate the expectation terms contained in the optimality conditions of a model in the spirit of the stochastic Parameterized Expectations Algorithm, introduced by den Haan and Marcet (1990). We call the method the NN-based Expectations Algorithm. Its benefits are highlighted in the application presented in section 3 and arise from the ability of the algorithm to approximate generic non-linear policy functions, whose functional forms are ex-ante unknown, in presence of a large and multicollinear state space generated by solving the model through a stochastic simulation approach. In particular, we demonstrate the computational gains of our methodology by extending the optimal government debt problem studied by Faraglia et al. (2019) from two to three maturities and investigating how the hedging benefits provided by the additional bond contribute to the household's welfare. We find that optimal debt management with three maturities allows the government to respond to expenditure shocks by raising financial income and, at the same time, reducing total outstanding debt. Through this mechanism, the government effectively subsidizes the private sector. In our calibration it results in a welfare gain of 2.38%, mainly driven by a lower consumption volatility and increased leisure.

[26] In the one bond code ridge penalty is a scalar. When solving the 2 bonds model we allow coefficients to have different penalties, without any noticeable improvement.

References

Aiyagari, S. Rao, Albert Marcet, Thomas J. Sargent, and Juha Sappala. 2002. "Optimal Taxation without State-Contingent Debt." *Journal of Political Economy*, 110(6): 1220–1254.

Angeletos, George-Marios. 2002. "Fiscal Policy with Noncontingent Debt and the Optimal Maturity Structure." *Quarterly Journal of Economics*, 117(3): 1105–1131.

Azinovic, Marlon, Luca Gaegauf, and Simon Scheidegger. 2021. "Deep Equilibrium Nets." *Working Paper*.

Bellman, Richard Ernest. 1961. "Adaptive control processes: a guided tour." *Princeton University Press*.

Bhandari, Anmol, David Evans, Mikhail Golosov, and Thomas J. Sargent. 2017. "Fiscal Policy and Debt Management with Incomplete Markets." *The Quarterly Journal of Economics*, 132: 617–663.

Bhandari, Anmol, David Evans, Mikhail Golosov, and Thomas Sargent. 2021. "Managing Public Portfolios." *Working Paper*.

Bigio, Saki, Galo Nuño, and Juan Passadore. 2021. "Debt-Maturity Management with Liquidity Costs." *Working Paper*.

Buera, Francisco, and Juan Pablo Nicolini. 2004. "Optimal maturity of government debt without state contingent bonds." *Journal of Monetary Economics*, 51: 531–554.

Cybenko, G. 1988. "Approximation by superpositions of a sigmoidal function." *Mathematics of Control, Signals and Systems volume 2, pages 303–314*.

den Haan, Wouter, and Albert Marcet. 1990. "Solving the Stochastic Growth Model by Parameterizing Expectations." *Journal of Business and Economic Statistics*, 8(1): 31–34.

Duarte, Victor. 2018. "Machine Learning for Continuous-Time Finance." *Working Paper*.

Faraglia, Elisa, Albert Marcet, Rigas Oikonomou, and Andrew Scott. 2014. "Optimal Fiscal Policy Problems Under Complete and Incomplete Financial Markets: A Numerical Toolkit." *Working Paper*.

Faraglia, Elisa, Albert Marcet, Rigas Oikonomou, and Andrew Scott. 2019. "Government Debt Management: the Long and the Short of It." *Review of Economic Studies*, 86: 2554–2604.

Fernández-Villaverde, Jesús, Samuel Hurtado, and Galo Nuño. 2020. "Financial Frictions and the Wealth Distribution." *Working Paper*.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. 1989. "Multilayer feedforward networks are universal approximators." *Neural Networks Volume 2, Issue 5, 1989, Pages 359-366*.

Judd, Kenneth, Lilia Maliar, and Serguei Maliar. 2011. "Numerically stable and accurate stochastic simulation approaches for solving dynamic economic models." *Quantitative Economics, 2, 173-210*, 2: 173–210.

Karantounias, Anastasios G. 2018. "Optimal fiscal policy with recursive preferences." *Review of Economic Studies*, 85: 2283–2317.

Krusell, Per, and Anthony A Jr. Smith. 1998. "Income and Wealth Heterogeneity in the Macroeconomy." *Journal of Political Economy*, 106(5).

Lustig, Hanno, Christopher Sleet, and Sevin Yeltekin. 2008. "Fiscal hedging with nominal assets." *Journal of Monetary Economics*, 55: 710–727.

Maliar, Lilia, and Serguei Maliar. 2003. "Parameterized Expectations Algorithm and the Moving Bounds." *Journal of Business and Economic Statistics*, 21(1): 88–92.

Maliar, Lilia, Serguei Maliar, and Pablo Winant. 2021. "Deep learning for solving dynamic economic models." *Journal of Monetary Economics*, 122: 76–101.

Marcet, Albert, and Ramon Marimon. 2019. "Recursive Contracts." *Econometrica,*, 87: 1589–1631.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. 1986. "Learning representations by back-propagating errors." *Nature 323, 533–536*.

Scheidegger, Simon, and Ilias Bilionis. 2019. "Machine Learning for High-Dimensional Dynamic Stochastic Economies." *Journal of Computational Science*, 33: 68–82.

Swanson, Eric T. 2012. "Risk Aversion and the Labor Margin in Dynamic Equilibrium Models." *American Economic Review*, 102(4): 1663–1691.

Swanson, Eric T., and Glenn Rudebusch. 2012. "The Bond Premium in a DSGE Model with Long-Run Real and Nominal Risk." *merican Economic Journal: Macroeconomics*, 4: 105–144.

Widrow, B., and Jr. Hoff, M. E. 1960. "Learning representations by back-propagating errors." *Adaptive switching circuits, IRE WESCON Convention Record, Part 4, New York: IRE, pp. 96–104*.

A Appendix A - Optimal Fiscal Policy with Epstein-Zin Preferences

This appendix presents the general model with N bonds and Epstein-Zin preferences, describes the computation algorithm, and presents the equilibrium dynamics of the Epstein-Zin model with two bonds.

The Household's Problem Households have Epstein-Zin preferences where the instantaneous utility comes from consumption (c_t) and leisure (l_t). Parameter ρ controls the substitutability in time and parameter γ controls the attitude towards risk.[27] When $\rho = \gamma$, preferences collapse to CRRA, where consumption and leisure are additively separable. Preferences are

$$V_t = [(1-\beta)U(c_t, l_t)^{1-\rho} + \beta(\mathbb{E}_t V_{t+1}^{1-\gamma})^{\frac{1-\rho}{1-\gamma}}]^{\frac{1}{1-\rho}}.$$

Time endowment is equal to 1, therefore hours worked are $h_t = 1 - l_t$. Household cash-on-hand consists of after-tax labor income and current bond holdings. This can be either consumed or spent on the purchase of new bonds. The budget constraint (BC) is

$$c_t + q_t b_{t+1} = b_t + (1-\tau_t)h_t.$$

Defining $W_t = b_t$ and $R_{t+1} = W_{t+1}/q_t b_{t+1} = 1/q_t$, the BC can be rewritten as

$$\begin{aligned} c_t + q_t W_{t+1} &= W_t + (1-\tau_t)h_t \\ \implies W_{t+1} &= R_{t+1}(W_t - c_t + (1-\tau_t)h_t). \end{aligned}$$

Given the redefinition of the BC, the household's problem can be rewritten as

$$\begin{aligned} V_t(W_t) &= \max_{c_t, h_t} [(1-\beta)U(c_t, 1-h_t)^{1-\rho} + \beta(\mathbb{E}_t V_{t+1}(W_{t+1})^{1-\gamma})^{\frac{1-\rho}{1-\gamma}}]^{\frac{1}{1-\rho}}, \\ W_{t+1} &= R_{t+1}(W_t - c_t + (1-\tau_t)h_t). \end{aligned}$$

[27] When instantaneous utility includes leisure, the relative risk aversion is not γ but $1-(1-\gamma)(1-\rho)$, see Swanson (2012) or Swanson and Rudebusch (2012) for a detailed explanation.

Define the certainty equivalent (CE) as $\mathcal{R}_t(V_{t+1}) \equiv (\mathbb{E}_t V_{t+1}^{1-\gamma})^{\frac{1}{1-\gamma}}$. The optimality condition for consumption (FOC_c) is

$$\begin{aligned} & V_t^{\rho}\left((1-\beta)(1-\rho)U_t^{-\rho}U_{c,t} - \beta(1-\rho)(\mathbb{E}_t V_{t+1}^{1-\gamma})^{\frac{\gamma-\rho}{1-\gamma}}\mathbb{E}_t\left[V_{t+1}^{-\gamma}R_{t+1}V_{W,t+1}\right]\right) = 0 \\ & \implies (1-\beta)U_t^{-\rho}U_{c,t} = \beta\mathcal{R}_t^{\gamma-\rho}\mathbb{E}_t\left[V_{t+1}^{-\gamma}R_{t+1}V_{W,t+1}\right]. \end{aligned}$$

The optimality condition for labor supply (FOC_h) is

$$\begin{aligned} & V_t^{\rho}\left(-(1-\beta)(1-\rho)U_t^{-\rho}U_{l,t} + \beta(1-\rho)(\mathbb{E}_t V_{t+1}^{1-\gamma})^{\frac{\gamma-\rho}{1-\gamma}}\mathbb{E}_t\left[V_{t+1}^{-\gamma}(1-\tau_t)R_{t+1}V_{W,t+1}\right]\right) = 0 \\ & \implies (1-\beta)U_t^{-\rho}U_{l,t} = (1-\tau_t)\beta\mathcal{R}_t^{\gamma-\rho}\mathbb{E}_t\left[V_{t+1}^{-\gamma}R_{t+1}V_{W,t+1}\right]. \end{aligned}$$

The envelope condition is

$$V_{W,t} = V_t^{\rho}\beta\mathcal{R}_t^{\gamma-\rho}\mathbb{E}_t V_{t+1}^{-\gamma}R_{t+1}V_{W,t+1}.$$

Combine FOC_c with FOC_h to get

$$\frac{U_{l,t}}{U_{c,t}} = 1-\tau_t.$$

Combine FOC_c with the envelope condition to get

$$V_{W,t} = V_t^{\rho}(1-\beta)U_t^{-\rho}U_{c,t} \implies V_{W,t+1} = V_{t+1}^{\rho}(1-\beta)U_{t+1}^{-\rho}U_{c,t+1},$$

which implies

$$(1-\beta)U_t^{-\rho}U_{c,t} = \beta\mathcal{R}_t^{\gamma-\rho}\mathbb{E}_t\left[V_{t+1}^{-\gamma}R_{t+1}V_{t+1}^{\rho}(1-\beta)U_{t+1}^{-\rho}U_{c,t+1}\right].$$

Plugging this back in the FOC_c, rearranging and simplifying leads to the following inter-temporal Euler equation

$$1 = \beta\mathbb{E}_t\left[\mathcal{M}_t(V_{t+1})\left(\frac{U_{t+1}}{U_t}\right)^{-\rho}\frac{U_{c,t+1}}{U_{c,t}}R_{t+1}\right],$$

where $\mathcal{M}_t(V_{t+1}) \equiv \left(\frac{V_{t+1}}{\mathcal{R}_t(V_{t+1})}\right)^{\rho-\gamma}$. The bond's price q_t is the expected value of the stochastic discount factor

$$q_t = \beta \mathbb{E}_t \left[\mathcal{M}_t(V_{t+1}) \left(\frac{U_{t+1}}{U_t} \right)^{-\rho} \frac{U_{c,t+1}}{U_{c,t}} \right].$$

Ramsey Problem

The Ramsey problem consists of finding $\{c_t, b^i_{t+1}, \mu_t, V_t\}_{t=0}^{\infty}$ in order to maximize the household's welfare taking household intra- and inter-temporal first order conditions as constraints. Hence, the Lagrangian of the problem is

$$\mathcal{L} = V_0 + \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \Bigg\{ \mu_t \Big(U_t^{-\rho} U_{c,t} s_t + \sum_{i=1}^{N} \mathbb{E}_t \beta^i b^i_{t+1} \mathcal{M}_t(V_{t+i}) U_{t+i}^{-\rho} U_{c,t+i} - \\ \sum_{i=1}^{N} \mathbb{E}_t \beta^{i-1} b^i_t U_{t+i-1}^{-\rho} U_{c,t+i-1} \mathcal{M}_t(V_{t+i-1}) \Big) + \sum_{i=1}^{N} \xi^i_{U,t}(B^U - b^i_{t+1}) + \sum_{i=1}^{N} \xi^i_{L,t}(b^i_{t+1} - B^L) \Bigg\}.$$

In addition, the Ramsey planner needs to respect the recursivity constraint for V_t

$$V_t = [(1-\beta) U(c_t, 1 - c_t - g_t)^{1-\rho} + \beta (\mathbb{E}_t V_{t+1}^{1-\gamma})^{\frac{1-\rho}{1-\gamma}}]^{\frac{1}{1-\rho}}.$$

Optimality Conditions

In order to calculate the first order condition with respect to c_t, it is necessary to calculate an expression for the derivative of welfare V_0 with respect to c_t. Note that V_0 contains all the consumption path from 0 throughout ∞, which we derive in subsection A. Knowing $\frac{\partial V_0}{\partial c_t(g^t)}$ the first order condition with respect to consumption is[28]

$$V_0^{\rho}(1-\beta) \mathcal{X}_{0,t} U_t^{-\rho} \frac{\partial U_t}{\partial c_t(g^t)} + \mu_t \left(\frac{\partial U_t^{-\rho} U_{c,t}}{\partial c_t(g^t)} s_t + \frac{\partial s_t}{\partial c_t} U_t^{-\rho} U_{c,t} \right) + \\ \frac{\partial U_t^{-\rho} U_{c,t}}{\partial c_t(g^t)} \sum_{i=1}^{N} \left(\mu_{t-i} \mathcal{M}_{t-i}(V_t) - \mu_{t-i+1} \mathcal{M}_{t-i+1}(V_t) \right) b^i_{t-i+1} + \lambda^V_t V_t^{-\rho} (1-\beta) U_t^{-\rho} \frac{\partial U_t}{c_t(g^t)} = 0,$$

where λ^V_t is the time-t Lagrange multiplier associated with the recursive constraint and $\mathcal{X}_{t_1,t_2} \equiv \prod_{k=1}^{t_2-t_1} \mathcal{M}_{t_1+k-1}(V_{t_1+k})$ with $\mathcal{X}_{t_1,t_2} \equiv 1, \forall t_2 \leq t_1$. Note that $\mathcal{X}$ admits a recursive representation.[29]

[28] With $\frac{\partial V_0}{\partial c_t(g^t)} = V_0^{\rho} \beta^t (1-\beta) \mathcal{X}_{0,t} \pi(g^t|g^0) U_t^{-\rho} \frac{\partial U_t}{\partial c_t(g^t)}$.

[29] $\mathcal{X}_{t_1,t_2} \equiv \prod_{k=1}^{t_2-t_1} \mathcal{M}_{t_1+k-1}(V_{t_1+k}) = \mathcal{M}_{t_2-1}(V_{t_2}) \prod_{k=1}^{t_2-t_1-1} \mathcal{M}_{t_1+k-1}(V_{t_1+k}) = \mathcal{M}_{t_2-1}(V_{t_2}) \mathcal{X}_{t_1,t_2-1}$.

The first order condition with respect to b^i_{t+1} yields the following inter-temporal expression for the promise keeping Lagrange multiplier μ

$$\mu_t = \left[\mathbb{E}_t \mathcal{M}_t(V_{t+i}) U^{-\rho}_{t+i} U_{c,t+i}\right]^{-1} \left[\mathbb{E}_t \mu_{t+1} \mathcal{M}_{t+1}(V_{t+i}) U^{-\rho}_{t+i} U_{c,t+i} + \frac{\xi^U_t}{\beta^i} - \frac{\xi^L_t}{\beta^i}\right].$$

The first order condition with respect to V_t is

$$\begin{aligned}
&\beta^{t-i} \sum_{i=1}^{N} \pi(g^{t-i}|g^0) \beta^i \mu_{t-i} \pi(g^t|g^{t-i}) b^i_{t-i+1} U_{c,t} U_t^{-\rho} \frac{\partial \mathcal{M}_{t-i}(V_t)}{\partial V_t(g^t)} - \\
&\beta^{t-i+1} \sum_{i=1}^{N} \pi(g^{t-i+1}|g^0) \beta^{i-1} \mu_{t-i+1} \pi(g^t|g^{t-i+1}) b^i_{t-i+1} U_{c,t} U_t^{-\rho} \frac{\partial \mathcal{M}_{t-i+1}(V_t)}{\partial V_t(g^t)} - \\
&\lambda^V_t \beta^t \pi(g^t|g^0) + \beta^{t-1} \pi(g_{t-1}|g^0) \lambda^V_{t-1} \beta V^{\rho}_{t-1} \mathcal{R}_{t-1}(V_t)^{-\rho} \mathcal{M}_{t-1}(V_t)^{\frac{-\gamma}{\rho-\gamma}} \pi(g_t|g^{t-1}) = 0,
\end{aligned}$$

which, after rearranging, yields the following recursion for λ^V_t[30]

$$\lambda^V_t = \sum_{i=1}^{N} \left(\mu_{t-i} \frac{\partial \mathcal{M}_{t-i}(V_t)}{\partial V_t(g^t)} - \mu_{t-i+1} \frac{\partial \mathcal{M}_{t-i+1}(V_t)}{\partial V_t(g^t)}\right) b^i_{t-i+1} U_{c,t} U_t^{-\rho} + \lambda^V_{t-1} \left(\frac{V_{t-1}}{V_t}\right)^{\rho} \mathcal{M}_{t-1}(V_t).$$

The remaining first order condition with respect to μ_t just gives back the inter-temporal government implementability constraint.

1. Derivation of $\partial V_t / \partial c_{t+j}$

If $j < 0$:

$$\partial V_t / \partial c_{t+j} = 0.$$

If $j = 0$:

$$\frac{\partial V_t}{\partial c_t} = (1-\beta) V_t^{\rho} U_t^{-\rho} \frac{\partial U_t}{\partial c_t}.$$

[30] Where: $\frac{\partial \mathcal{M}_{t-i}(V_t)}{\partial V_t} = (\rho - \gamma) \frac{\mathcal{M}_{t-i}(V_t)}{V_t} \left[1 - \mathcal{M}_{t-i}(V_t)^{\frac{1-\gamma}{\rho-\gamma}} \pi(g_t|g^{t-i})\right]$.

If $j = 1$:

$$\begin{aligned}\frac{\partial V_t}{\partial c_{t+1}(g^{t+1})} &= V_t^{\rho}\beta\mathcal{R}_t(V_{t+1})^{\gamma-\rho}\pi(g_{t+1}|g^t)V_{t+1}^{-\gamma}\frac{\partial V_{t+1}}{\partial c_{t+1}(g^{t+1})} \\ &= V_t^{\rho}\beta\mathcal{R}_t(V_{t+1})^{\gamma-\rho}\pi(g_{t+1}|g^t)V_{t+1}^{-\gamma}\left((1-\beta)V_{t+1}^{\rho}U_{t+1}^{-\rho}\frac{\partial U_{t+1}}{\partial c_{t+1}(g^{t+1})}\right) \\ &= V_t^{\rho}\beta(1-\beta)\mathcal{M}_t(V_{t+1})\pi(g_{t+1}|g^t)U_{t+1}^{-\rho}\frac{\partial U_{t+1}}{\partial c_{t+1}(g^{t+1})}.\end{aligned}$$

If $j = 2$:

$$\begin{aligned}\frac{\partial V_t}{\partial c_{t+2}(g^{t+2})} &= V_t^{\rho}\beta\mathcal{R}_t(V_{t+1})^{\gamma-\rho}\pi(g_{t+1}|g^t)V_{t+1}^{-\gamma}\frac{\partial V_{t+1}}{\partial c_{t+2}} \\ &= V_t^{\rho}\beta\mathcal{R}_t(V_{t+1})^{\gamma-\rho}\pi(g_{t+1}|g^t)V_{t+1}^{-\gamma}\left(V_{t+1}^{\rho}\beta(1-\beta)\mathcal{R}_{t+1}(V_{t+2})^{\gamma-\rho}\pi(g_{t+2}|g^{t+1})V_{t+2}^{\rho-\gamma}U_{t+2}^{-\rho}\frac{\partial U_{t+2}}{\partial c_{t+2}}\right) \\ &= V_t^{\rho}\beta^2(1-\beta)\prod_{k=1}^{2}\mathcal{M}_{t+k-1}(V_{t+k})\prod_{k=1}^{2}\pi(g_{t+k}|g^{t+k-1})U_{t+2}^{-\rho}\frac{\partial U_{t+2}}{\partial c_{t+2}(g^{t+2})}.\end{aligned}$$

For a generic $j \geq 0$:

$$\frac{\partial V_t}{\partial c_{t+j}(g^{t+j})} = V_t^{\rho}\beta^j(1-\beta)\mathcal{X}_{t,t+j}\pi(g^{t+j}|g^t)U_{t+j}^{-\rho}\frac{\partial U_{t+j}}{\partial c_{t+j}(g^{t+j})}.$$

2. Derivation of $\frac{\partial\mathcal{M}_{t-1}(V_t)}{\partial V_t}$

$$\begin{aligned}\frac{\partial\mathcal{M}_{t-1}(V_t)}{\partial V_t} &= (\rho-\gamma)\frac{\mathcal{M}_{t-1}(V_t)^{\frac{\rho-\gamma-1}{\rho-\gamma}}}{\mathcal{R}_{t-1}(V_t)^2}\left[\mathcal{R}_{t-1}(V_t) - V_t\underbrace{\mathcal{M}_{t-1}(V_t)^{\frac{-\gamma}{\rho-\gamma}}\pi(g_t|g^{t-1})}_{\frac{\partial\mathcal{R}_{t-1}(V_t)}{\partial V_t}}\right] \\ &= (\rho-\gamma)\frac{\mathcal{M}_{t-1}(V_t)^{\frac{\rho-\gamma-1}{\rho-\gamma}}}{\left(\frac{V_t}{\mathcal{M}_{t-1}(V_t)^{\frac{1}{\rho-\gamma}}}\right)^2}\left[\mathcal{R}_{t-1}(V_t) - V_t\mathcal{M}_{t-1}(V_t)^{\frac{-\gamma}{\rho-\gamma}}\pi(g_t|g^{t-1})\right] \\ &= (\rho-\gamma)\frac{\mathcal{M}_{t-1}(V_t)^{\frac{\rho-\gamma+1}{\rho-\gamma}}}{V_t^2}\left[\mathcal{M}_{t-1}(V_t)^{\frac{-1}{\rho-\gamma}}V_t - V_t\mathcal{M}_{t-1}(V_t)^{\frac{-\gamma}{\rho-\gamma}}\pi(g_t|g^{t-1})\right] \\ &= (\rho-\gamma)\frac{\mathcal{M}_{t-1}(V_t)^{\frac{\rho-\gamma+1}{\rho-\gamma}}}{V_t}\left[\mathcal{M}_{t-1}(V_t)^{\frac{-1}{\rho-\gamma}} - \mathcal{M}_{t-1}(V_t)^{\frac{-\gamma}{\rho-\gamma}}\pi(g_t|g^{t-1})\right] \\ &= (\rho-\gamma)\frac{\mathcal{M}_{t-1}(V_t)}{V_t}\left[1 - \mathcal{M}_{t-1}(V_t)^{\frac{1-\gamma}{\rho-\gamma}}\pi(g_t|g^{t-1})\right].\end{aligned}$$

Algorithm to Solve the Model with Epstein-Zin Preferences

Here we describe an algorithm to solve the model with Epstein-Zin preferences. Generally, it is very similar to the one used to solve the model with CRRA preferences, with the exception that there are additional state variables and additional terms that the neural network has to approximate. At every instant t, the information set is $\mathcal{I}_t = \{g_t, \{\{b^i_{t-k}\}_{k=0}^{N-1}\}_{i=1}^{N}, \{\mu_{t-k}\}_{k=1}^{N}, \{\lambda^V_{t-k}\}_{k=1}^{N}\}$. Consider projections of $\mathcal{R}_{t-i}(V_t)$, $\mathbb{E}_t\mathcal{M}_t(V_{t+i})U^{-\rho}_{t+i}U_{c,t+i}$, $\mathbb{E}_t\mu_{t+i}\mathcal{M}_{t+1}(V_{t+i})U^{-\rho}_{t+i}U_{c,t+i}$ and $\mathbb{E}_t\mathcal{M}_t(V_{t+i-1})U^{-\rho}_{t+i-1}U_{c,t+i-1}$ on $\mathcal{I}_t$. We model these relationships using one single-layer artificial neural network $\mathcal{ANN}(\mathcal{I}_t)$. For example, with two bonds we would have $4N+1$ inputs and 8 outputs.[31] In particular, use the following notations for each output:

$$
\begin{aligned}
\mathcal{ANN}^i_1 &= \mathcal{R}_{t-i}(V_t) \quad \text{for } i = \{1, N-1, N\},\\
\mathcal{ANN}^i_2 &= \mathbb{E}_t\mathcal{M}_t(V_{t+i})U^{-\rho}_{t+i}U_{c,t+i} \quad \text{for } i = \{1, N\},\\
\mathcal{ANN}^i_3 &= \mathbb{E}_t\mu_{t+i}\mathcal{M}_{t+1}(V_{t+i})U^{-\rho}_{t+i}U_{c,t+i} \quad \text{for } i = \{1, N\},\\
\mathcal{ANN}^i_4 &= \mathbb{E}_t\mathcal{M}_t(V_{t+i-1})U^{-\rho}_{t+i-1}U_{c,t+i-1} \quad \text{for } i = \{N\}.
\end{aligned}
$$

Given starting values $\mu_{t-1} = \lambda^V_{-1} = 0$ and initial weights for $\mathcal{ANN}$, simulate a sequence of $\{c_t\}$, $\{\lambda^V_t\}$, $\{\mu_t\}$ as follows:

1. Impose the Maliar moving bounds on all debts instruments, see Maliar and Maliar (2003). These bounds are particularly important and need to be tight and open slowly, since the neural network at the beginning can only make accurate predictions around zero debt - that is our initialization point. Proper penalty functions are used instead of the ξ terms to avoid out of bound solutions, see Faraglia et al. (2014) for more details.

2. Use forward-states on the following i equations

$$
\forall i: \quad \mu_t = \left[\mathbb{E}_t\mathcal{ANN}^i_1(\mathcal{I}_{t+1})\right]^{-1}\left[\mathbb{E}_t\mathcal{ANN}^i_2(\mathcal{I}_{t+1}) + \frac{\xi^i_{U,t}}{\beta^i} - \frac{\xi^i_{L,t}}{\beta^i}\right].
$$

[31] One with maturity 1 and the other with maturity N.

3. Find λ_t^V, μ_t, c_t and $\{b_{t+1}^i\}_{i=1}^N$ that solve the following system of equations:

i. $$\lambda_t^V = \sum_{i=1}^{N} \left(\mu_{t-i} \frac{\partial \mathcal{M}_{t-i}(V_t)}{\partial V_t(g^t)} - \mu_{t-i+1} \frac{\partial \mathcal{M}_{t-i+1}(V_t)}{\partial V_t(g^t)} \right) b_{t-i+1}^i U_{c,t} U_t^{-\rho} + \lambda_{t-1}^V \left(\frac{V_{t-1}}{V_t} \right)^{\rho} \left(\frac{V_t}{\mathcal{ANN}_1^1(\mathcal{I}_{t+1})} \right)^{\rho-\gamma},$$

ii. $$V_0^{\rho}(1-\beta)\mathcal{X}_{0,t} U_t^{-\rho} \frac{\partial U_t}{\partial c_t(g^t)} + \mu_t \left(\frac{\partial U_t^{-\rho} U_{c,t}}{\partial c_t(g^t)} s_t + \frac{\partial s_t}{\partial c_t} U_t^{-\rho} U_{c,t} \right) +$$

$$\frac{\partial U_t^{-\rho} U_{c,t}}{\partial c_t(g^t)} \sum_{i=1}^{N} \left(\mu_{t-i} \left(\frac{V_t}{\mathcal{ANN}_1^i(\mathcal{I}_{t+1})} \right)^{\rho-\gamma} - \mu_{t-i+1} \left(\frac{V_t}{\mathcal{ANN}_1^{i-1}(\mathcal{I}_{t+1})} \right)^{\rho-\gamma} \right) b_{t-i+1}^i + \lambda_t^V V_t^{-\rho} (1-\beta) U_t^{-\rho} \frac{\partial U_t}{c_t(g^t)} = 0,$$

iii. $$\sum_{i=1}^{N} \beta^{i-1} b_t^i \mathcal{ANN}_4^i(\mathcal{I}_{t+1}) = s_t U_c^{-\rho} U_{c,t} + \sum_{i=1}^{N} \beta^i b_{t+1}^i \mathcal{ANN}_2^i(\mathcal{I}_{t+1}),$$

iv. $$\forall i: \quad \mu_t = \left[\mathbb{E}_t \mathcal{ANN}_1^i(\mathcal{I}_{t+1}) \right]^{-1} \left[\mathbb{E}_t \mathcal{ANN}_2^i(\mathcal{I}_{t+1}) + \frac{\xi_{U,t}^i}{\beta^i} - \frac{\xi_{L,t}^i}{\beta^i} \right],$$

where

$$\frac{\partial \mathcal{M}_{t-i}(V_t)}{\partial V_t} = (\rho - \gamma) \frac{\left(\frac{V_t}{\mathcal{ANN}_1^i} \right)^{\rho-\gamma}}{V_t} \left[1 - \left(\frac{V_t}{\mathcal{ANN}_1^i} \right)^{1-\gamma} f_{g_t}(g_t | g_{t-i}) \right],$$

$$V_t = [(1-\beta) U(c_t, 1-c_t-g_t)^{1-\rho} + \beta \mathcal{ANN}_1^1(\mathcal{I}_{t+1})^{1-\rho}]^{\frac{1}{1-\rho}},$$

and

$$\frac{\partial U_t}{\partial c_t} = U_{c,t} - U_{l,t}.$$

Note that $f_{g_t}(g_t|g_{t-1})$ is the conditional probability density of the exogenous g process.

4. Use the simulated sequence to train the $\mathcal{ANN}$ and re-start from point 1 till convergence of the predicted sequence over the realized one.

Numerical Results with Two Bonds and Epstein-Zin Preferences

Bhandari et al. (2021) and Karantounias (2018) convincingly demonstrate that implications for optimal portfolios changes once the model contains preferences that match the asset prices. In this section, we explore the implications for optimal debt management once we change preferences to Epstein-Zin and add a shock that is orthogonal to the government expenditure process. In

particular we add an endowment shock z_t, such that $l_t = z_t - h_t$.[32] Figure 6 shows that in this setting the optimal portfolio does not hold any short position, the allocation shares are around equal among different maturities, and little portfolio re-balancing happens in response to government shocks. The intuition is that the presence of a shock orthogonal to government expenditure makes it risky to hold a highly leveraged position, and this risk is magnified by Epstein-Zin preferences. Bhandari et al. (2021) solve a similar model using a perturbation method around the current level of government debt. Our results using our methodology are consistent with their intuition.

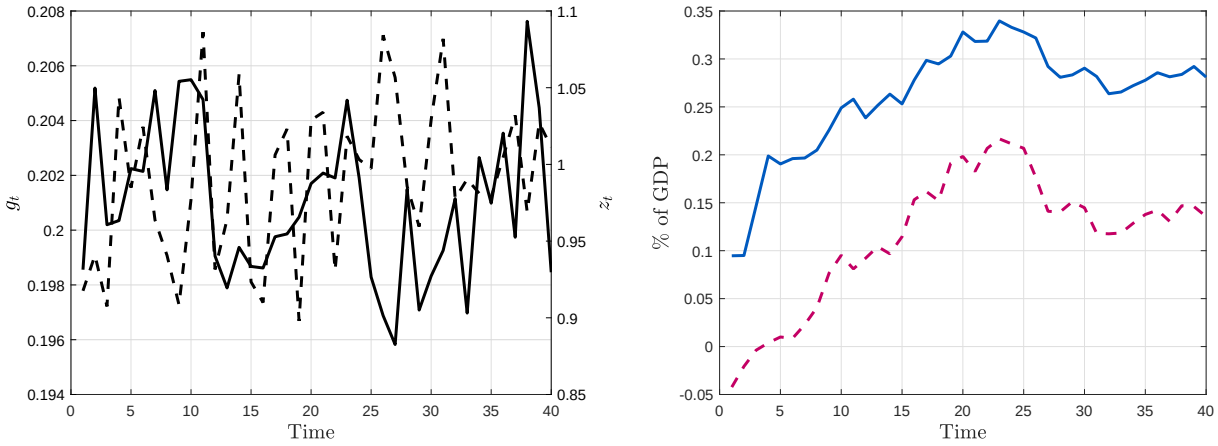

Figure 6: Simulated series with 2 bonds and Epstein-Zin preferences

Notes: The figure shows the equilibrium dynamics of the two-bond model with Epstein-Zin preferences. The left panel plots a sequence of exogenous shocks. Solid black - government expenditure, dashed black - productivity. The right panel plots the sequence of bonds. Solid blue - long bond with N=10. Dashed purple line - short bond with N=1.

Parameter	Value
Discount factor	$\beta = 0.96$
RRA	$\gamma = 1.5$
1/EIS	$\rho = 1.6$
Leisure utility parameter	$\eta = 1.8$
AR(1) parameter in g_t	$\phi_1 = 0.8$
constant in AR(1) process of in g_t	$c = 0.04$
Variance of the disturbances to g_t	$\sigma_\epsilon^2 = 0.00001$
AR(1) parameter in z_t	$\phi_z = 0.9$
constant in AR(1) process of in z_t	$c = 0.1$
Variance of the disturbances to z_t	$\sigma_{\epsilon_z}^2 = 0.003$
Borrowing limits	$\bar{M}^N, \bar{M}^S = 100\%$ of GDP $\underline{M}^N, \underline{M}^S = -100\%$ of GDP

Table 9: Parameter Values used in the model with Epstein-Zin preferences

[32] z_t is independent of g_t and follows an AR1 process $z_t = \mu_z \rho_z z_{t-1} + \epsilon_t^z$ with $\mu_z = 0.9$, $\rho_z = 0.1$, $\epsilon \sim N(0, 0.003)$.

B Appendix B: Additional Figures

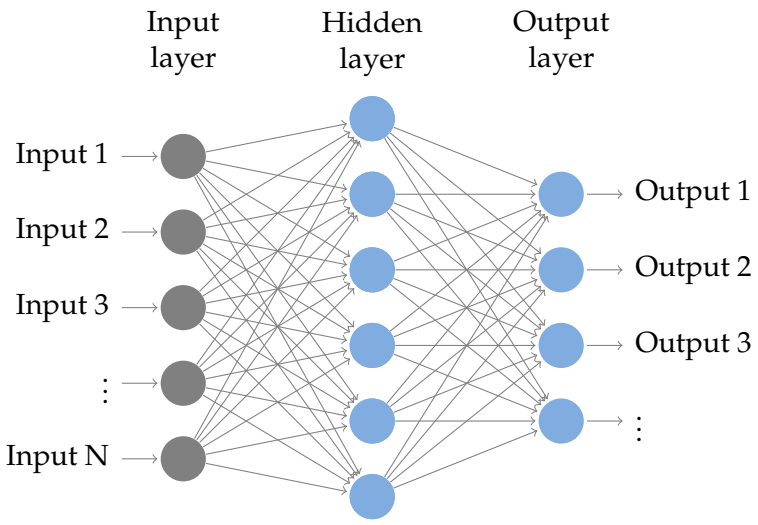

Figure 7: Artificial Neural Network Structure

Notes: The figure presents the structure of a single hidden layer artificial neural network. Each circle in the picture represents an artificial neuron, and the arrows point the direction of the information flow in the prediction process. Neurons in the hidden layer perform the non linear activation of inputs, which are combined linearly in the output layer.

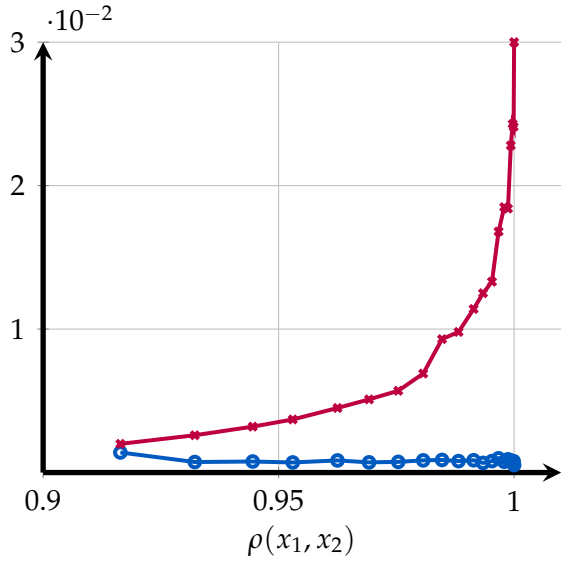

Figure 8: Variance term of MSPE with neural network and polynomial

Notes: The figure shows the variance part of the means squared prediction error $1/n \sum_{i=1}^{n} [y_i - \mathbb{E}(\hat{y}_i)]^2$ in function of the correlation between x_1 and x_2. Blue line with circles - NN, purple line with crosses - polynomial regression.

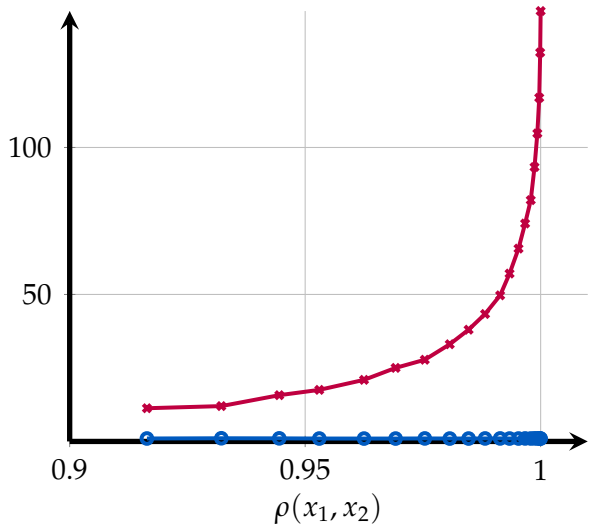

Figure 9: Variance term of MSPE with neural network and polynomial

Notes: The figure shows the bias squared part of the means squared prediction error $1/n \sum_{i=1}^{n} [y_i - \mathbb{E}(\hat{y}_i)]^2$ in function of the correlation between x_1 and x_2. Blue line with circles - NN, purple line with crosses - polynomial regression.

C Appendix C: Implementation Details

In this section we describe the practical details of the algorithm used to solve the model in section 4 and described in section 3.3.

Lagrange multipliers The system in step 2 contains multiple constraints, which poses a significant computational challenge. Ideally one would numerically solve the unconstrained model and then verify that the constraints do not bind and if, for example, M^N binds, set $b_t^N = \bar{M}^N$ and find the associated values for consumption and leisure. In a multiple-bond model this is challenging because after setting $b_t^N = \bar{M}^N$, one needs to check if the other constraints do not bind in the recomputed solution and, if they do, enforce them and recalculate the solution again and so on and on. To overcome this challenge we approximate Lagrange multipliers with the following function: $\xi_{L,t}^i = \phi(\underline{M}^i - b_t^i) + log(1 + \phi(\bar{M}^i - b_t^i))$ if $b_t^i < \underline{M}^i$ and $\xi_{U,t}^i = \phi(b_t^i - \bar{M}^i) + log(1 + \phi(b_t^i - \bar{M}^i))$ if $b_t^i > \bar{M}^i$, where ϕ controls the relative importance of the constraint. In our implementation we set $\phi = 90$. We also find that including these multipliers in the training set allows for different bond dynamics close and away from the constraints and improves prediction accuracy. As a result, in our implementation

$$\mathcal{I}_t = \{g_t, \{b_{t-k}^S\}_{k=0}^{S-1}, \{b_{t-k}^M\}_{k=0}^{M-1}, \{b_{t-k}^L\}_{k=0}^{L-1}, \{\mu_{t-k}\}_{k=1}^{L}, \{\{\xi_{i,t-k}^S\}_{k=0}^{S-1}\}_{i=L,U}, \{\{\xi_{i,t-k}^M\}_{k=0}^{M-1}\}_{i=L,U}, \{\{\xi_{i,t-k}^L\}_{k=0}^{L-1}\}_{i=L,U}, \{\xi_{i,t-1}^{\text{Total}}\}_{i=L,U}\}.$$

With $S = 1, N = 5, L = 10$ our state space includes 61 variables.

Forward-States PEA When the model contains more than one maturity, μ_t is over-identified. This is because optimality conditions for every maturity identify μ_t, as the information set I_t contains variables that are pre-determined at time t.

$$\forall i: \quad \mu_t = \mathcal{ANN}_1^i(\mathcal{I}_t)^{-1}\left[\mathcal{ANN}_2^i(\mathcal{I}_t) + \frac{\xi_{U,t}^i}{\beta^i} - \frac{\xi_{L,t}^i}{\beta^i} + \frac{\xi_{U,t}^{\text{Total}}}{\beta^i} - \frac{\xi_{L,t}^{\text{Total}}}{\beta^i}\right]. \tag{9}$$

We tackle this problem by using a Forward States PEA, introduced in Faraglia et al. (2019). It uses the current values of the state variables I_{t+1} combined with the law of iterated expectations. This is done in two steps. First, we replace the $\mathcal{ANN}^i(\mathcal{I}_t)$ terms in the optimality conditions with $\mathbb{E}_t\mathcal{ANN}^i(\mathcal{I}_{t+1})$ and, instead of approximating $\mathbb{E}_t(U_c(c_{t+i}))$, $\mathbb{E}_t(U_c(c_{t+i-1}))$ and $\mathbb{E}_t(U_c(c_{t+i}\mu_{t+1}))$, we use the information set $\mathcal{I}_t$ to approximate $\mathbb{E}_t(U_c(c_{t+i-1}))$, $\mathbb{E}_t(U_c(c_{t+i-1-1}))$ and $\mathbb{E}_t(U_c(c_{t+i-1}\mu_{t+1}))$ for $i = S, L, M$. Then, we use Gaussian quadrature to calculate the conditional expectation of the neural network evaluated at $\mathcal{I}_{t+1}$.

Neural Network Initialization In order to initialize the neural network weights, we need to make a guess for bond sequences. We make an educated guess that $b_{t+1}^L = g_t/2 - \mathbb{E}(g_t/2); b_{t+1}^S = -b_t^L$ and $b_{t+1}^M = \sqrt{g_t/10} - \mathbb{E}(g_t/10)$. Given these sequences, we use the government budget constraint and the first order condition for c_t to find the sequence for μ_t. Given the guess for bonds, we can calculate the initial multipliers $\xi_{L,t}^i$, $\xi_{U,t}^i$, $\xi_{L,t}^{\text{Total}}$, and $\xi_{U,t}^{\text{Total}}$. We calculate them setting the initial bond constraints equal to $\underline{M}, \bar{M} = \pm.005$. We then use these sequences to initialize the neural network. Generally, the initial guess can be any real sequences as long as it is not constant. Nevertheless, having a good guess helps the algorithm to converge faster.

Stochastic Simulation We solve the model using Matlab 2020a and build and train the neural network using functions from the Neural Network toolbox. We run the stochastic simulation from the starting point where $b_1^S = b_1^M = b_1^N = 0$ and $\mu_0 = \mathbb{E}(\mu)$. To solve the system of first order conditions at every period t of the simulation, we use the Levenberg-Marquardt algorithm and stop the solver when the first order condition errors are less than 10^{-12}. We set T=1000 and drop the first 150 periods in training the neural network. The algorithm converges when the sequences for bonds and neural network weights do not change between two consecutive stochastic simulations.

Solving the System At every period t of the stochastic simulation we need to solve the system of first order conditions to get values for c_t, b_t^L, b_t^M, and b_t^S. We solve it using the Levenberg-Marquardt

method. Since this is a local solver, there is no guarantee that the system is solved given a particular initial guess. In our implementation we attempt to solve the system for at most *maxrep* number of different starting points. If the solution errors are below our specified threshold, the algorithm proceeds with the solution and moves to the next period. If the solution errors are not below our specified threshold, we pick the solution with the lowest error. In practice we use 10^{-13} as the solution criteria and set *maxrep* to 50 and verify that solution errors are below it for every t in the stochastic simulation.

Neural Network Hyperparameters We set most of the hyperparameters to standard values used in the literature. We use one hidden layer to leverage the trade-off between approximation accuracy and training time. According to the universal approximation theorem, single hidden layer networks are able to approximate every continuous bounded function with an arbitrarily small error Cybenko (1988) and have smaller training times than multiple hidden layer networks. We then choose the number of neurons according to procedure described in section 2.4. We train the network using gradient descent with an adaptive learning rate. Table 10 summarizes the remaining hyperparameters.

Parameter	Value
Activation function	Hyperbolic tangent sigmoid
Training algorithm	Gradient descent with an adaptive learning rate backpropagation
Number of Hidden Layers	1
Number of Neurons	10
Learning Rate	0.0001
Learning Increase Factor	1.01
Learning Decrease Factor	0.9
Maximum Number of Epochs	1000
Performance Goal	0
Maximum Validation Checks	6
Maximum Performance Increase	1.04

Table 10: NN hyperparameters

Notes: The table summarizes the hyperparameters of the neural network used to solve the model in section 4.