A Machine Learning Projection Method for Macro-Finance Models*

Vytautas Valaitis
University of Surrey†

Alessandro T. Villa
FRBC‡

September 6, 2023

Abstract

We use supervised machine learning to approximate the expectations typically contained in the optimality conditions of an economic model in the spirit of the parameterized expectations algorithm (PEA) with stochastic simulation. When the set of state variables is generated by a stochastic simulation, it is likely to suffer from multicollinearity. We show that a neural network-based expectations algorithm can deal efficiently with multicollinearity by extending the optimal debt management problem studied by Faraglia et al. (2019) to four maturities. We find that the optimal policy prescribes an active role for the newly added medium-term maturities, 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 during recessions.

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.

†University of Surrey, Stag Hill, University Campus, Guildford GU2 7XH, UK. Email: v.valaitis@surrey.ac.uk.

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

1 Introduction

In this paper, we exploit the computational gains that derive from the robustness to multicollinearity of neural networks to extend the optimal debt management problem studied by Faraglia et al. (2019) to four maturities. The hedging benefits provided by 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.

We use a neural network (NN) in a supervised machine learning fashion to approximate the expectation terms typically contained in the optimality conditions of an economic model, in the spirit of the Parameterized Expectations Algorithm (PEA) with stochastic simulation, introduced by den Haan and Marcet (1990) and in a similar fashion to Duffy and McNelis (2001). On the one hand, stochastic simulation methods allow us to tackle problems with a high number of state variables, 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. In this context, this paper makes two contributions. First, we show that an NN-based Expectations Algorithm can deal efficiently with multicollinearity by extending the optimal debt management problem studied by Faraglia et al. (2019) to four maturities. Second, we show that the optimal debt management policy prescribes an active role for the medium-term maturities, enabling the planner to raise financial income without increasing its total borrowing in response to expenditure shocks. We consider this problem a particularly interesting economic application that also poses significant computational challenges 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 policy under full commitment with multiple 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 four maturities considered in section 4 features 46 state variables. Third, because of the maturities, many of these state variables are multicollinear when

the model is solved by using a stochastic simulation approach.[1] Fourth, this class of problems does not have a stochastic steady state, as documented in Aiyagari et al. (2002), and tends to frequently hit the borrowing and lending constraints. Such properties render the model particularly hard to solve using perturbation methods around a particular point.[2]

In section 4, we use the methodology to study the optimal government debt management policy when the Ramsey planner can issue an increasing number of debt instruments with different maturities. Intuitively, the prices of longer maturities are typically 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 the additional medium-term maturities to exploit both the hedging and the price smoothing benefits. The government holds leveraged positions in all bonds and rebalances the portfolio with more emphasis on the shorter maturities in recessions. We find that, when the number of available maturities increases from two to three (and four), the total amount of outstanding 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 higher leisure and less volatile labor taxes.

Literature Review This paper contributes to two strands of literature: (i) numerical methods in economics and (ii) optimal fiscal policy.

In terms of methods, this paper builds on the seminal work of den Haan and Marcet (1990), who introduced PEA. The idea of using neural networks to parameterize decision rules in a similar

[1] The state space includes lagged values of the same variables (e.g., lagged values of outstanding bonds and Lagrange multipliers). 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 2.5. 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.

[2] Bhandari et al. (2017*a*) 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.

fashion to PEA goes back to Duffy and McNelis (2001). Our paper contributes to this literature by showing that an NN-based Expectations Algorithm can deal efficiently with multicollinearity by extending the optimal debt management problem studied by Faraglia et al. (2019) to more than two maturities. In particular, we exploit the computational gains to study the optimal government debt management problem of Faraglia et al. 2019 with three and four maturities, which yields new economic insights. Note that 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). Our methodology builds on Condensed PEA and Forward-States PEA, in the context of optimal fiscal policy, allowing for machine learning to reduce the state space endogenously and handling multicollinearity 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.

In contemporaneous work, Maliar et al. (2021) and Maliar and Maliar (2022) discuss how neural networks can handle multicollinearity. In particular, they do so in the context of the Krusell and Smith (1998) model. We complement their work by demonstrating the robustness to multicollinearity in the context of optimal fiscal policy. Additionally, we show that the interaction between the capability of a neural network to deal with multicollinearity and its flexibility in approximating generic policy functions plays an important role in generating unbiased predictions. In section 2.5, we show that if a researcher pre-commits to approximate the policy functions with polynomials that are misspecified then, under multicollinearity among the state variables, the predictions will be biased. Thanks to the flexible non-parametric nature of a neural network, which does not require making ex-ante assumptions about the functional form of the policy functions, this problem disappears.

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.

Other papers that use machine learning to solve economic models include Scheidegger and Bilionis (2019), Azinovic et al. (2021), Fernández-Villaverde et al. (2020), 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. 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.

Our application also contributes to the strand of literature on optimal fiscal policy. In particular, it is relevant to the literature on the optimal maturity structure of government debt.[3] Lustig et al. (2008) find that the optimal policy prescribes an almost exclusive role to the longest maturity in a model with no-lending constraints and a New-Keynesian model where bonds are nominal. In our setting we allow for government lending and study the hedging benefits of a choice between multiple maturities of real bonds. Bhandari et al. (2017*a*) study the optimal maturity structure in an open economy with two maturities, and Bigio et al. (2022) allow for a finite number of maturities in an economy with liquidity costs of issuing debt, where liquidity costs differ by maturity. 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 and four maturities in a closed economy. We find that the optimal policy prescribes an active role for the medium-term bonds. The additional maturities enable the planner to raise financial revenue without increasing

[3] Aiyagari et al. (2002), Angeletos (2002), Buera and Nicolini (2004), Lustig et al. (2008), Faraglia et al. (2019), Bhandari et al. (2017*a*), and Bigio et al. (2022).

the total outstanding debt, in response to a positive expenditure shock. We show that, through this mechanism, the government uses the additional maturities to effectively subsidize the private sector in recessions, resulting in more leisure and less volatile labor taxes.

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 Investment 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 and four maturities. Section 5 discusses and compares the NN-based Expectations Algorithm to other state-of-the-art methods. Section 6 concludes.

2 User guide: Machine Learning and PEA

This section serves as an introduction to supervised machine learning. Specifically, it focuses on how to use it to solve a dynamic economic model in a similar fashion to PEA with stochastic simulation.[4] 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 non-linear policy functions in the presence of a large and multicollinear state space.

2.1 Environment

The typical dynamic model contains intertemporal Euler equations, intratemporal Euler equations, and laws of motion

$$
\begin{aligned}
f^{\text{Inter}}(c_t, X_t) &= \beta \mathbb{E}\left[g(c_{t+1}, X_{t+1})|X_t\right], \\
f^{\text{Intra}}(c_t, X_t) &= 0, \\
X_{t+1} &= h(X_t, c_t, \xi_{t+1}),
\end{aligned}
$$

[4] For a general introduction to machine learning, the reader can refer to Hastie et al. (2009). For a course tailored to economists, the reader can refer to the lecture notes by Jesús Fernández-Villaverde available here: https://www.sas.upenn.edu/~jesusfv/teaching.html.

where $c_t \in \mathbb{R}^C$ is a vector of C controls (with E dynamic choices and $C - E$ static choices), $X_t \in \mathbb{R}^S$ is a vector of endogenous and exogenous state variables, $\beta \in (0,1)$ is a time-discount factor, $f^{\text{Inter}} : \mathbb{R}^C \times \mathbb{R}^S \to \mathbb{R}^E$, $f^{\text{Intra}} : \mathbb{R}^C \times \mathbb{R}^S \to \mathbb{R}^{C-E}$, $g : \mathbb{R}^C \times \mathbb{R}^S \to \mathbb{R}^E$, and $\xi_{t+1} \in \mathbb{R}^I$ is a vector of innovation shocks. For example, in the stochastic neoclassical investment model, c_t corresponds to consumption, f^{Inter} corresponds to the marginal utility of consumption, f^{Intra} does not apply if the model does not include intratemporal choices (e.g, labor), $g \equiv f(c_{t+1}) \left(z_{t+1} \alpha K_{t+1}^{\alpha-1} + 1 - \delta \right)$, $X_t \equiv \{K_t, z_t\}$ 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, i.e.

$$X_{t+1} = h(X_t, c_t, \xi_{t+1}) = \begin{pmatrix} K_{t+1} \\ \log z_{t+1} \end{pmatrix} = \begin{pmatrix} (1-\delta)K_t - c_t + z_t K_t^{\alpha} \\ \rho \log z_t + \xi_{t+1} \end{pmatrix}.$$

The typical PEA approximates the conditional expectations in the intertemporal Euler equations as polynomial functions of the state space X_t

$$\forall e \in [1, E] : \mathbb{E}\left[g_e(c_{t+1}, X_{t+1}) | X_t \right] \simeq P_n(X_t; \eta_e).$$

The polynomial typically used in the PEA is

$$P_n(X_t; \eta_e) = \exp\left(\eta_{e,0} + \sum_{p=1}^{P} \sum_{s=1}^{S} \left[\eta_{e,p,s} \cdot (\ln X_{s,t})^p \right] \right),$$

where $\eta_e = [\eta_{e,0}, \eta_{e,1,1}, \cdots, \eta_{e,1,S}, \dots]$. 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 $\eta^n = \{\eta_1^n, \cdots, \eta_E^n\}$ that solve all Euler equations and all laws of motion.

When $X \equiv \{X_t\}_{t=T_0}^{T-1}$ 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.

Moreover, in the simple illustrative case of the neoclassical investment model, a first order

[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.

Algorithm 1 Stochastic (simulations) PEA

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

while η_n^i converges **do**
  for $t \leftarrow 0$ to T **do** ▷ Generate $X \equiv \{X_t\}_{t=0}^{T}$
    $c_t \leftarrow$ Solve $f^{\text{Intra}}(c_t, X_t) = 0$ and $f^{\text{Inter}}(c_t, X_t) = \beta P_n(X_t; \eta_n)$
    $X_{t+1} \leftarrow h(X_t, c_t, \xi_{t+1})$
  end for
  for $t \leftarrow 0$ to $T-1$ **do** ▷ Generate $Y \equiv \{y_t\}_{t=0}^{T-1}$
    $y_t \leftarrow g(c_{t+1}, X_{t+1})$
  end for
  $\hat{\eta}_n^i \leftarrow (X^T X)^{-1} X^T Y$ ▷ Regress to find new weights
  $\eta_n^{i+1} \leftarrow w \cdot \hat{\eta}_n^i + (1-w) \cdot \eta_n^i$ ▷ Update with dampening
end while

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 $\{y_t\}_{t=0}^{T-1}$. For example, in the neoclassical investment model with log-utility over consumption, this would serve the purpose of representing the function

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

Machine learning proposes a flexible structure for the function P and infers a function from the generated data $\{X_t\}_{t=T_0}^{T-1}$ (which we label *training data*) to the set of generated examples $\{y_t\}_{t=T_0}^{T-1}$ (which we label *training examples*). This particular task of using machine learning to learn a function that maps from inputs to outputs based on training data and examples is referred in the literature as *supervised learning*. And neural networks are a powerful class of universal approximators able to deal with strong non-linearities.

2.3 Fitting Neural Networks

In the NN-based Expectations Algorithm, the equivalent of the *regression phase* is called the *training phase*. As described in appendix C, a neural network is characterized by unknown weights $\{w, \psi\}$.[6] Similarly to a regression, the objective is to seek weights such that the neural network fits the samples $\{X_t, y_t\}_{t=0}^{T-1}$. More precisely, the problem is to find

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

such that the sum of squares

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

is minimized. In a standard linear regression setting, typically (but not necessarily) this problem is solved analytically. This problem could also be solved using a gradient iterative procedure (e.g. gradient descent). This approach is typically more robust to multicollinearity since it does not require inverting the matrix $\hat{X}^T \hat{X}$. An iteration n of gradient descent updates the weights of the neural network according to

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

$$\psi_e^{(n+1)} = \psi_e^{(n)} - \gamma_r \sum_{k=1}^{K} \frac{\partial R_k(w)}{\partial \psi_e}, \tag{2}$$

where the gradient can be derived using the chain rule for differentiation. More specifically, the partial derivatives $\frac{\partial R_k(w)}{\partial w_m}$ and $\frac{\partial R_k(w)}{\partial \beta_e}$ in equations (1) and (2) can be efficiently computed through a two-pass algorithm called back-propagation (Rumelhart et al., 1986). Back-propagation 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). One cycle through the full training samples is called a training epoch. In other words, 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

[6] Appendix C contains details about the neural network structure used in this section.

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. When the neural network weights are updated, the speed at which the model changes can be updated through the parameters γ_r in equations (1) and (2). The parameter γ_r is called the learning rate and it is similar in spirit to a dampening parameter. 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.

Other aspects that can affect the fitting of the neural network are: (i) the initial weights, (ii) the problem of overfitting, (iii) inputs normalization, and (iv) the number of neurons. Initial neural network weights are chosen as near zero random values. Figure 9 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.[7] In practical terms, we solve the model by first initializing the neural network to a simplified version of the model. For example, before solving the neoclassical investment model as described in algorithm 2, it is possible to solve the model analytically (in this particularly case, by setting $\delta = 1$), simulate an equilibrium sequence with the analytical solution, and use it to train the neural network. In a more complicated scenario, such as the optimal maturity management problem presented in this paper, we first solve the model without debt (i.e., the government only uses the income from taxes to finance government expenditure) to first initialize the neural network. The model should not overfit the data. Since stochastic simulation methods (such as PEA) only explore a subset of the ergodic set of state variables (i.e., those combinations of state variables simulated in the equilibrium), we optimize the model for out-of-sample predictions. We split the simulated data randomly in the training set (in-sample) and validation set (out-of-sample) with a 70-30 proportion, respectively.[8] The number of epochs is determined by maximizing the neural network's performance on the validation set. 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.[9] This could potentially

[7] Substantial research effort has been put into choosing the initial weights depending on the specific neural network architecture (for example, see Glorot and Bengio, 2010 for deep neural networks).

[8] We did not find a significant difference in allocations and forecast errors when we changed the training set to be 50% and 90% of the sample.

[9] In the context of deep neural networks, the distribution of each layer's inputs varies during the training phase, since the weights of the previous layers change as well. Typically, this requires adopting lower learning rates and carefully choosing initial parameters. This problem is known as internal covariate shift. In this context, batch normalization can

impair the learning process and lead to a slower convergence, or worse, mean squared prediction errors. 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 1 (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.

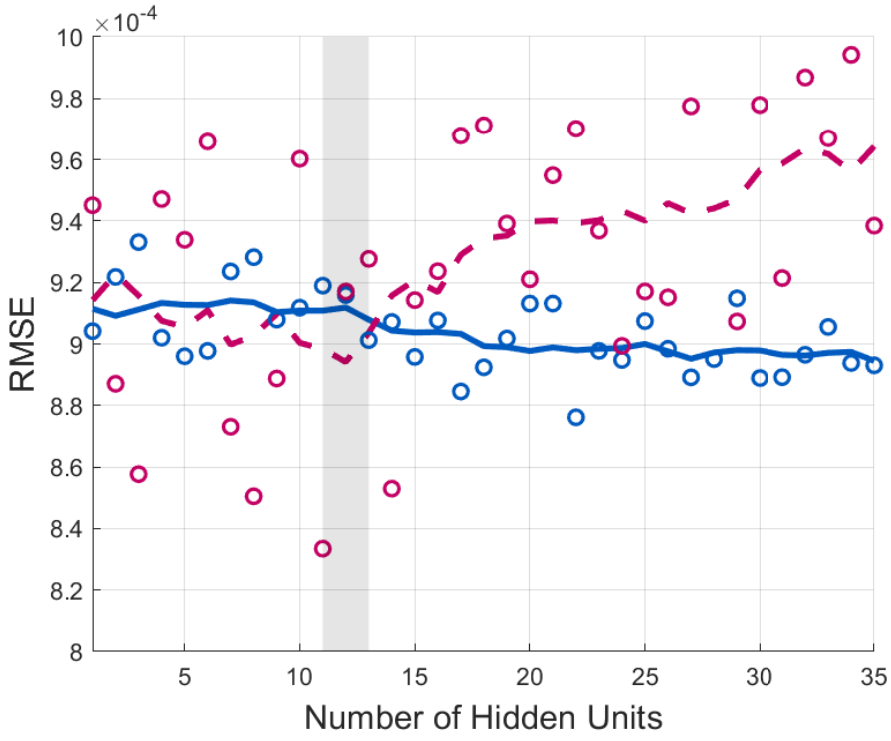

Figure 1: Root Mean Squared Error (RMSE) in function of the number of neurons in the hidden layer

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.

2.4 Example: Neural Networks and PEA Applied to the Neoclassical Investment Model

This section describes the implementation of the NN-based Expectations Algorithm applied to the neoclassical investment 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

achieve higher a learning rate (see Ioffe and Szegedy, 2015) by reducing the problem of internal covariate shift.

and the procedure that selects the optimal number of neurons, is publicly available.[10] 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 minimizes the mean squared prediction error out-of-sample as shown in figure 1).

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 $u(c) = \log(c)$, $u_c(c) = c^{-1}$.

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

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

▷ *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**
        ▷ *The output of the function Net is the expectation object, given the current state*
        $\mathbb{E}_t[\text{RHS}_{t+1}] \leftarrow \text{Net}([k_t; z_t])$
        $c_t \leftarrow u_1^{-1}(\beta \cdot \mathbb{E}_t[\text{RHS}_{t+1}])$
        $k_{t+1} \leftarrow z_t \cdot k_t^{\alpha} + (1 - \delta) \cdot k_t - c_t$
    end for
    ▷ *Train the NN using the new k and z alongside the above RHS*
    $\text{RHS}_t \leftarrow u_c(c_{t+1}) \cdot (\alpha \cdot z_{t+1} \cdot k_{t+1}^{\alpha-1} + 1 - \delta)$
    $\text{Net} \leftarrow \text{train}(\text{Net}, [k_t, z_t], \text{RHS}_t)$
    ▷ *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

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

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

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 optimality conditions 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 guide illustrating this point.

2.5 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.

For simplicity, imagine that we would like to approximate the policy function of the neoclassical investment model with two TFP shocks, full depreciation $\delta = 1$, and log-utility $u(c) = \log c$. The true functional form of the policy function for the capital choice (in this simple case it can be solved analytically) is

$$k_{t+1}(k_t, z_{1,t}, z_{2,t}) = \beta\alpha \exp(z_{1,t}) \exp(z_{2,t}) k_t^{\alpha}, \tag{3}$$

where the true parameters are $\alpha = .36$, $\beta = .95$, and $z_{1,t}$ is a log-AR(1) process with persistence .8 and standard deviation of the innovation shock .0224. Moreover, z_2 is determined using the

following formula

$$z_2 = \lambda \hat{z}_2 + (1 - \lambda) z_1.$$

where $\hat{z}_2$ is a log-AR(1) process with persistence .8 and standard deviation of the innovation shock .0224.

We generate, through stochastic simulation, equilibrium sequences $\{X_t, y_t\}_{t=0}^T$ where $y_t = k_{t+1}$, and $X_t = [k_t \quad z_{1,t} \quad z_{2,t}]$ is a vector that contains the three state variables. The objective is to use $\{X_t, y_t\}_{t=0}^T$ in order to infer the functional form of equation 3. 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 (i.e., different values of $\lambda \in [0, 1]$), and we calculate the associated $\{y_t\}_{t=0}^T$ using equation 3. 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 $y_t = \beta_1 x_{1,t} + \beta_2 x_{2,t} + \beta_3 x_{3,t}$.[11] 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 policy functions. 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

$$\text{MSPE}_t = \mathbb{E}\left[(y_t - \hat{y}_t)^2\right] = \underbrace{\left[y_t - \mathbb{E}(\hat{y}_t)\right]^2}_{Bias_t^2} + \underbrace{\mathbb{E}\left[\hat{y}_t - \mathbb{E}(\hat{y}_t)\right]^2}_{Variance_t}. \tag{4}$$

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

[11] 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 simply to 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.

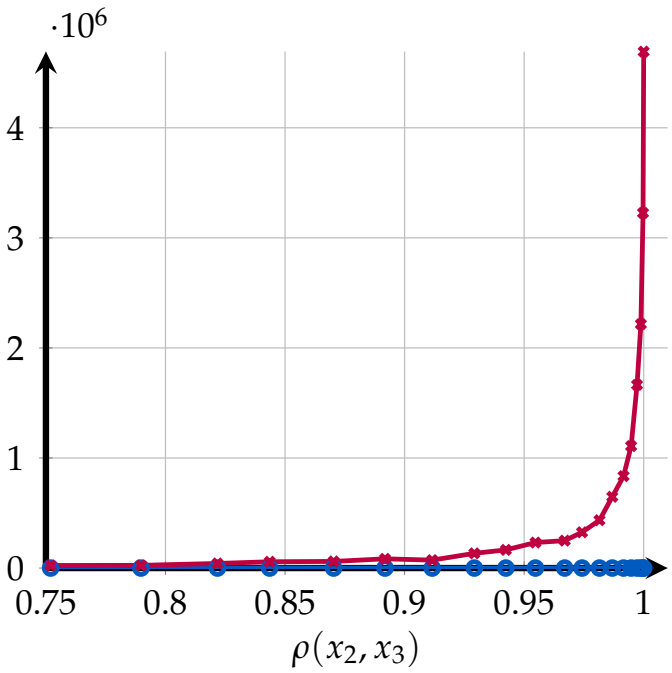

Figure 2: Mean squared prediction error with a neural network and a polynomial

Notes: The figure shows the mean squared prediction error $1/n \sum_{t=1}^{n} [y_t - \mathbb{E}(\hat{y}_t)]^2$ in function of the correlation between x_2 and x_3. 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 4, we find that most of the prediction error comes from the bias squared term as shown in figures 6 and 7 in appendix B.4. 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 particularly 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.[12] 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

[12]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.

non-linearities difficult to approximate with a parametric approach (i.e., borrowing and lending constraints).[13] 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

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

where b_t^N indicates an N-periods maturity bond and p_t^N is its corresponding price.[14] 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:

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

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.

[13] 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.

[14] 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[15]

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

with borrowing and lending limits[16]

$$\bar{M} \geq b_{t+1}^N, \quad \underline{M} \leq b_{t+1}^N.$$

The government's optimality conditions are

$$u_{c,t} - v_{l,t} + \mu_t(u_{cc,t}c_t + u_{c,t} + v_{ll,t}(c_t + g_t) - v_{l,t}) + u_{cc,t}(\mu_{t-N} - \mu_{t-N+1})b_{t-N+1}^N = 0, \tag{5}$$

$$\mu_t = \mathbb{E}_t(u_{c,t+N})^{-1}\left[\mathbb{E}_t(u_{c,t+N}\mu_{t+1}) + \frac{\xi_{U,t}}{\beta^N} - \frac{\xi_{L,t}}{\beta^N}\right], \tag{6}$$

$$b_{t+1}^N \beta^N \mathbb{E}_t(u_{c,t+N}) = b_t^N \beta^{N-1} \mathbb{E}_t(u_{c,t+N-1}) - g_t u_{c,t} - (u_{c,t} - v_{l,t})(g_t + c_t), \tag{7}$$

where μ_t is the Lagrange multiplier on the time t measurability constraint, and $\xi_{U,t}$ 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

$$X_t = \left\{g_t, \{\mu_{t-i}\}_{i=1}^N, \{b_{t-i}^N\}_{i=0}^{N-1}\right\}.$$

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 6 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 3 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,

[15] See Aiyagari et al. (2002) for details on how to use the recursive Lagrangian approach in this context.

[16] $\bar{M}_N \geq b_{t+1}^N$ is the government saving constraint, which is equivalent to a household's borrowing constraint.

are highly correlated with each other in the simulated sequence.

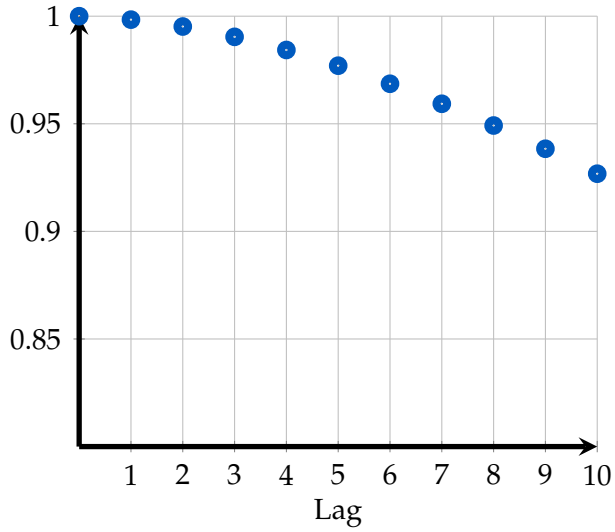

Figure 3: Autocorrelation function of the equilibrium bond sequence

Notes: The figure shows the autocorrelation function of b_t^N. The numbers are obtained after simulating the model equilibrium dynamics for T=5000.

For this reason, the model is hardly solvable using PEA (algorithm 1). In the literature, this problem has been tackled by an algorithm called Condensed PEA. Condensed PEA proposes to approximate the expected values in equations 5, 6, and 7 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:

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

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

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

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

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

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[17]

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

[17] $\bar{M}_N \geq b_{t+1}^N$ is the government saving constraint, which is equivalent to a household's borrowing constraint.

The optimality conditions are

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

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

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

where $\xi_{U,t}$ and $\xi_{L,t}$ are the Lagrange multipliers on the upper and the lower bounds, respectively, and $\xi^{\text{Total}}_{U,t}$ and $\xi^{\text{Total}}_{L,t}$ are the Lagrange multipliers on the upper and the lower bounds on the total bond portfolio. In the following section, we describe our computational strategy in detail. 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 2.4, 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 B.1. There are N bonds available with maturities from 1 to N periods. The state space at time t is $\mathcal{I}_t = \{g_t, \{\{b^i_{t-k}\}_{k=0}^{N-1}\}_{i=1}^{N}, \{\mu_{t-k}\}_{k=1}^{N}\}$. 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

$$\mathcal{ANN}^i_1(\mathcal{I}_t) = \mathbb{E}\left[u_{c,t+i} | \mathcal{I}_t\right] \quad \text{for } i = [1, ..., N],$$
$$\mathcal{ANN}^i_2(\mathcal{I}_t) = \mathbb{E}\left[\mu_{t+1} u_{c,t+i} | \mathcal{I}_t\right] \quad \text{for } i = [1, ..., N],$$
$$\mathcal{ANN}^i_3(\mathcal{I}_t) = \mathbb{E}\left[u_{c,t+i-1} | \mathcal{I}_t\right] \quad \text{for } i = [1, ..., N].$$

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.[18] Given starting values for μ_t and $\{b^i_t\}_{i=1}^{N}$ and initial weights

[18] Use S and N to denote short- and long-bond maturities, respectively. The six terms are $\mathbb{E}_t(u_{c,t+N})$, $\mathbb{E}_t(u_{c,t+N-1})$, $\mathbb{E}_t(u_{c,t+N-1}\mu_{t+1})$, $\mathbb{E}_t(u_{c,t+S})$, $\mathbb{E}_t(u_{c,t+S-1})$, and $\mathbb{E}_t(u_{c,t+S}\mu_{t+1})$. The term that does not require approximation in the latter case is $\mathbb{E}_t(u_{c,t+S-1})$, which becomes just $u_{c,t}$ when $S = 1$.

for $\mathcal{ANN}$, simulate a sequence of $\{c_t\}_{t=1}^T$, $\{\mu_t\}_{t=1}^T$ and $\{\{b_{t+1}^i\}_{i=1}^N\}_{t=1}^T$ as follows.[19]

1. As suggested by Maliar and Maliar (2003), we initially restrict the solution artificially within tight bounds on all debt instruments, and refine the solution gradually while we open the bounds slowly. These bounds are particularly important and initially 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. Additionally, we use penalty functions instead of the ξ-terms to avoid out of bound solutions.[20] 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 functions of the state variables that are relevant at $t+1$ instead of t and invoking the law of iterated expectations, such that we calculate $\mathbb{E}_t\mathcal{ANN}^i(\mathcal{I}_{t+1})$ instead of $\mathcal{ANN}^i(\mathcal{I}_t)$. 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 $\mathbb{E}_t(u_{c,t+i})$, $\mathbb{E}_t(u_{c,t+i-1})$, and $\mathbb{E}_t(u_{c,t+i}\mu_{t+1})$, we use the information set $\mathcal{I}_{t+1}$ to approximate $\mathbb{E}_{t+1}(u_{c,t+i})$, $\mathbb{E}_{t+1}(u_{c,t+i-1})$, and $\mathbb{E}_{t+1}(u_{c,t+i}\mu_{t+1})$. Then, we use Gaussian quadrature to calculate the conditional expectations of the neural network evaluated at $\mathcal{I}_{t+1}$.

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

$$\begin{cases} \forall i: & \mu_t = \left[\mathbb{E}_t\mathcal{ANN}_1^i(\mathcal{I}_{t+1})\right]^{-1}\left[\mathbb{E}_t\mathcal{ANN}_2^i(\mathcal{I}_{t+1}) + \frac{\xi_{U,t}^i}{\beta^i} - \frac{\xi_{L,t}^i}{\beta^i} + \frac{\xi_{U,t}^{\text{Total}}}{\beta^i} - \frac{\xi_{L,t}^{\text{Total}}}{\beta^i}\right], \\ & u_{c,t} - v_{l,t} + \mu_t\left[u_{c,t} - v_{l,t} + u_{cc,t}c + v_{ll,t}(c_t+g_t)\right] + \sum_{i=1}^N(\mu_{t-i} - \mu_{t-i+1})b_{t-i+1}^i u_{cc,t} = 0, \\ & \sum_{i=1}^N b_t^i\beta^{i-1}\mathbb{E}_t\mathcal{ANN}_3^i(\mathcal{I}_{t+1}) = u_{c,t}s_t + \sum_{i=1}^N b_{t+1}^i\beta^i\mathbb{E}_t\mathcal{ANN}_1^i(\mathcal{I}_{t+1}). \end{cases} \tag{8}$$

The system of equations (8) contains multiple Lagrange multipliers (arising from the in-

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

[20] We also find that including ξ terms explicitly in the training set improves prediction accuracy. More details can be found in appendix B.1.

equality constraints). This 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 $b^N_{t+1} = \bar{M}^N$ and find the associated values for consumption and leisure. In a multiple-bond model this is challenging because after setting $b^N_{t+1} = \bar{M}^N$, one needs to check if other constraints do not bind in the recomputed solution and, if they do, enforce them and recalculate the solution again,d and so on and on. To overcome this challenge, we augment the objective function with the following differentiable penalty function:

$$\forall i: \quad \Xi(b^i_{t+1}) = \begin{cases} \frac{\phi}{2} \cdot (b^i_{t+1} - \bar{M}^i)^2 + \int \log(1 + \phi \cdot (b^i_{t+1} - \bar{M}^i))\, \mathrm{d}b^i_{t+1}, & \text{if } b^i_{t+1} > \bar{M}^i \\ 0, \quad \text{if } \underline{M}^i \leq b^i_{t+1} \leq \bar{M}^i & \\ \frac{\phi}{2} \cdot (\underline{M}^i - b^i_{t+1})^2 + \int \log(1 + \phi \cdot (\bar{M}^i - b^i_{t+1}))\, \mathrm{d}b^i_{t+1}, & \text{if } b^i_{t+1} < \underline{M}^i \end{cases},$$

where ϕ controls the severity of the penalty. More details can be found in appendix B. The system of equations (8) can be re-derived after including the aforementioned penalty function. We solve the system of equations (8) using the Levenberg-Marquardt algorithm. Since this is a local solver, there is no guarantee that the system is solved globally 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 t. If the solution errors are not below our specified threshold, we pick the solution with the lowest error.

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 bound. More specifically, if an unreliable solution has been detected in iteration i, the algorithm restores the iteration $i-1$'s environment and performs the stochastic simulation with

$$\text{Bound}_{i-1} = \alpha \cdot \text{Bound}_{i-1} + (1 - \alpha) \cdot \text{Bound}_{i-2}.$$

4. If the solution calculated shrinking the bound at iteration $i-1$ is still 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 toward Bound_{i-2}. Once a reliable solution is found, the algorithm

proceeds to calculate the solution for iteration i again, but with

$$\text{Bound}_i = \text{Bound}_{i-1} + (\text{Bound}_{i-1} - \text{Bound}_{i-2}).$$

In this way, if an error is detected multiple times we guarantee that both Bound_i and Bound_{i-1} keep shrinking toward Bound_{i-2}, and there should 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 bounds are increased, and a new iteration starts.

We repeat this procedure until the neural network predictions converge and the simulated sequences of $\{b_t^i\}_{i=1}^N$ and c_t do not change.[21] Algorithm 3 describes the algorithm in greater detail and appendix B.1 contains more details.

4 Numerical Results

In this section, we exploit the computational gains that derive from the robustness to multicollinearity of the NN-based Expectations Algorithm to study the optimal maturity management problem of section 3 with four maturities of 1, 5, 10, and 15 periods. Specifically, we are interested in the effects on policy and allocations arising from the additional hedging opportunities with respect to a portfolio with only a short and a long maturity. We first present the calibration and then our numerical results.

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

$$u(c) = \frac{c^{1-\gamma}}{1-\gamma}, \quad v(l) = \chi \frac{l^{1-\eta_l}}{1-\eta_l}$$

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 a value of 2.87.

[21] 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 AR(1) process*
2: $g_{t+1} \leftarrow \mu_g + \rho_g \cdot g_t + \epsilon_{t+1}$

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 8 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**

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_g 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 a 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 ten thousand periods, which gives a value of 0.0031. Note that such parameterization is also broadly aligned with the estimates from the data.[22]

The government has four debt instruments at its disposal. We set maturities to 1, 5, 10, and 15 years and denote b^1, b^5, b^{10}, b^{15} as short, medium, and long and very long bonds, respectively. In addition to debt limits on individual bonds, we introduce a total debt limit of $\pm 100\%$ of GDP both in our benchmark model with only short and long bonds and in our calibration with four 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	$\bar{M}, \underline{M}, \bar{M}_{\text{total}}, \underline{M}_{\text{total}}$	$\pm$ 100% of GDP

Table 1: Calibrated parameters

Before proceeding, it is worth noting that we tested our methodology with the two-bond case. Our results in a two-bond model confirms the findings of Angeletos (2002) and 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 also shown in table 2, the bond portfolio positions are large and volatile as in Buera and Nicolini (2004).

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

4.2 Optimal Debt Management with Three and Four Bonds

Tables 2 and 3 summarize 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.

Model	$\mathbb{E}(b^1/GDP)$	$\mathbb{E}(b^5/GDP)$	$\mathbb{E}(b^{10}/GDP)$	$\mathbb{E}(b^{15}/GDP)$
1 Bond	0.017	-	-	-
2 Bonds	-0.03	-	0.343	-
3 Bonds	-0.555	0.704	0.632	-
4 Bonds	-0.63	0.884	0.908	-0.173
Model	$\sigma(b^1/GDP)$	$\sigma(b^5/GDP)$	$\sigma(b^{10}/GDP)$	$\sigma(b^{15}/GDP)$
1 Bond	0.243	-	-	-
2 Bonds	0.1	-	0.122	-
3 Bonds	0.591	0.34	0.533	-
4 Bonds	0.218	0.266	0.27	0.374

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	$\rho(g_t, b_t^1)$	$\rho(g_t, b_t^5)$	$\rho(g_t, b_t^{10})$	$\rho(g_t, b_t^{15})$		
1 Bond	0.549	-	-	-		
2 Bonds	0.707	-	-0.482	-		
3 Bonds	0.35	-0.181	-0.302	-		
4 Bonds	0.762	-0.094	-0.212	-0.22		
Model	$\rho(b_t^1, b_t^5)$	$\rho(b_t^1, b_t^{10})$	$\rho(b_t^{10}, b_t^5)$	$\rho(b_t^{15}, b_t^1)$	$\rho(b_t^{15}, b_t^5)$	$\rho(b_t^{15}, b_t^{10})$
1 Bond	-	-	-	-	-	-
2 Bonds	-	-0.796	-	-	-	-
3 Bonds	-0.944	-0.985	0.931	-	-	-
4 Bonds	-0.458	-0.565	0.918	0.047	-0.877	-0.82

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 an active use of all available maturities. Table 2 shows that the average position of each maturity is significantly different from zero and that bond positions are volatile, suggesting their active use responding to expenditure shocks. Table 3 show the correlations of all the maturities with government expenditure and among themselves. First, it shows the position of short maturity is positively correlated with expenditure shocks while the other maturities are negatively correlated. Second, short maturity is negatively correlated with all other maturities. These two together suggest that, in addition to holding a leveraged portfolio

on average, it is optimal to rebalance the portfolio toward shorter maturities. As shown in table 4, the additional hedging benefits of the additional maturities are reflected in 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	$\mathbb{E}(c_t)$	$\sigma(\ln(c_t))$	$\mathbb{E}(l_t)$	$\sigma(\ln(l_t))$	$\mathbb{E}(\tau_t)$	$\sigma(\ln(\tau_t))$	$\rho(\ln(\tau_t), \ln(\tau_{t-1}))$
1 Bond	0.252	0.029	0.666	0.006	0.247	0.121	0.971
2 Bonds	0.250	0.029	0.668	0.004	0.255	0.106	0.929
3 Bonds	0.248	0.028	0.670	0.005	0.27	0.10	0.914
4 Bonds	0.247	0.027	0.671	0.006	0.274	0.091	0.841

Table 4: 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 of how hedging benefits provided by the additional maturities affect household allocations and taxes. 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's liabilities, the differential response of long and short prices also affects the terms of issuing new debt. Since long prices fall more than 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 5 shows how optimal debt management affects government finances as we increase the number of debt instruments.

Description	Moment	1 Bond	2 Bonds	3 Bonds	4 Bonds
Corr. Debt/GDP and g_t	$\rho\left(\frac{\sum_i b_t^i}{y_t}, g_t\right)$	0.547	0.136	-0.079	-0.131
Corr. Net Financial Income and g_t	$\rho(\sum_i (p_t^i b_{t+1}^i - p_t^{i-1} b_t^i), g_t)$	0.186	0.405	0.416	0.511
Corr. Net Financial Income (constant price) and g_t	$\rho(\sum_i (\mathbb{E}(p_t^i) b_{t+1}^i - \mathbb{E}(p_t^{i-1}) b_t^i), g_t)$	0.078	0.11	-0.103	0.019
Av. Net Financial Income (%)	$\mathbb{E}\left(\frac{\sum_i p_t^i b_{t+1}^i - p_t^{i-1} b_t^i}{y_t}\right)$	-0.142	-0.845	-2.213	-2.569
Av. Labor Tax Income (%)	$\mathbb{E}\left(\frac{\tau_t(1-l_t)}{y_t}\right)$	24.7	25.5	27.0	27.4

Table 5: Government Income and Borrowing

Notes: The table shows selected moments from the models with one, two, three, and four maturities. The 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 corresponds to Net Financial Income holding bond prices fixed at their average values.

To inspect how this rebalancing matters for the government's budget, 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 5.

In the one- and two-bond economy, the government borrows from the private sector to finance expenditure shocks. In the three- and four-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. The 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 5. The low correlation here 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. This indicates that if prices were constant, portfolio rebalancing would have little effect on the cyclicality of financial income and the government's

budget.

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 is 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 (C. PEA) used in Faraglia et al. (2019) and the generalized stochastic simulation algorithm (GSSA) described in Judd et al. (2011).

5.1 Relation to Condensed PEA

This method extracts orthogonal components from the information set. The method is similar to the Principle Component Analysis (PCA), except that the number of factors does not have to be chosen ex-ante. Algorithm 4 reports the pseudo-code of Condensed PEA and highlights with colors the part of Condensed PEA that changes when the NN-based Expectations Algorithm (NN EA) is implemented. In this section, we solve the model with a short one-period bond and a long ten-period bond using both algorithms. The model is solved with individual bonds bounds $\bar{M}, \underline{M}$ set at $\pm 100\%$. As reported in table 6, the two algorithms reach a similar outcome, but the NN-based Expectations Algorithm is significantly faster. The speed gains mainly come from the removal of the external loop (lines 1, 13, 14, 15, 16, and 17 in Condensed PEA) 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: $\hat{\eta}_n^i \leftarrow \text{argmin } (Y - P_n(X^{C,k}; \eta_n^i)^2$ ▷ 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 hand, eliminating the external loop (line 1) reduces the complexity of the algorithm significantly, since Condensed PEA requires testing an unknown number of combinations of core regressors. On the other hand, our algorithm requires substituting OLS (lines 10 and 11) with a neural network training algorithm, which has higher complexity. Note that the computation time of one entire simulation from 1 to T (lines 3-9) takes significantly more time under Condensed PEA.[23] In total, the Condensed PEA needed to cycle 4 times before the core set $X^{C,k}$ converged. This means that Condensed PEA required approximately four times more iterations than NN EA.

Method	Time	Forecast Error	$\mathbb{E}(c_t)$	$\mathbb{E}(l_t)$	$\mathbb{E}(\tau_t)$	$\sigma(\ln(c_t))$	$\sigma(\ln(l_t))$	$\sigma(\ln(\tau_t))$	$\rho(\tilde{S}_t, \tilde{S}_{t-1})$
C. PEA	203810s	0.373	0.256	0.663	0.219	0.024	0.007	0.116	0.798
NN EA	23744s	0.391	0.256	0.664	0.222	0.024	0.007	0.123	0.869

Table 6: Moments Condensed PEA versus NN EA

Notes: The table shows the equilibrium moments calculated with Condensed PEA and the NN Expectations Algorithm (NN EA). The forecast error is calculated as $\sum_{t=1}^{T}\sum_{e=1}^{E} |y_{e,t} - \hat{\mathbb{E}}(y_{e,t})|$, where $y_{e,t}$ here is used to indicate the realized value correspondent to each expectation $\hat{\mathbb{E}}(y_t)$ predicted by either Condensed PEA or NN EA. Bond bounds are set as $\bar{M}, \underline{M}$ at $\pm 100\%$. We follow Faraglia et al. (2019) and calculate $\tilde{S}_t = \frac{\tilde{b}_t^1}{\tilde{b}_t^1 + \tilde{b}_t^{10}}$ as the ratio between the market value of short-term debt $\tilde{b}_t^1$ and the market value of the total outstanding debt $\tilde{b}_t^1 + \tilde{b}_t^{10}$. The computation times are obtained using MATLAB 2023a and a computer with an Intel(R) Core(TM) i7-8750H CPU (9M Cache, up to 4.10 GHz) with RAM 16 GB.

[23] Lines 3-9 of algorithm 4 take on average 34s for Condensed PEA, whereas the neural network training phase took around 15s. Lines 13-16 of algorithm 4 take on average 0.003s for Condensed PEA, whereas the neural network training phase took around 0.21s.

Note that, as the number of maturities increases, the Condensed PEA requires testing a much higher number of combinations of core regressors. In this sense, NN EA is a more scalable approach. On a related note, in appendix B.2, we also present a comparison between Condensed PEA and our algorithm based on time complexity. Our calculations show that NN EA has a lower time complexity than Condensed PEA, if Condensed PEA requires more than one loop on the core set of state variables to converge.

5.2 Relation to GSSA

Next, we compare our methodology to the GSSA method proposed by Judd et al. (2011). GSSA resolves the multicollinearity problem using standard econometric techniques (i.e., single value decomposition, principal components, and ridge regression), combined with stochastic simulation. We solve the government debt management problem with one maturity using GSSA.[24] In particular, we use ridge regression since, as noted in Judd et al. (2011), it works best under severe multicollinearity.[25] In our application, ridge regression combined with stochastic simulation works when bond constraints are loose, whereas when the constraints are tight, ridge regression coefficients fail to converge. As we illustrate in details in appendix B.3, the reason lies in the fact that ridge regression requires choosing penalty parameters. This choice presents non-trivial challenges. In our application, the simulated data changes in each iteration since we are solving the model with an iterative procedure. In principle, this should require updating the penalty parameters in each iteration. However, in our experience, changing penalty parameters at each iteration also creates instability as the simulated data is endogenous to the penalty choice in the previous iterations. For this reason, we fix the penalty parameters (see appendix B.3 for more details on how we choose the penalty parameters) during the whole procedure and, since the simulated debt sequence tends to change significantly with each iteration when the bond constraints are tight, the algorithm fails to converge. Recall that, as explained in section 3.3, our algorithm initially restricts the solution artificially within tight bounds on all debt instruments, and refines the solution gradually while it opens the bounds slowly. To conclude, the challenges in choosing (or fixing) the penalty parameters, combined with the frequent changes in debt sequences in each iteration induced by initially tight bounds, render ridge regression with stochastic simulation hard to scale in our application. Further details can be found in appendix B.3.

[24] We also attempted to solve the model with two maturities but we were not successful.

[25] Our application features a very ill-conditioned matrix $X^T X$, especially when the number of maturities increases.

6 Conclusion

In this paper, we exploit the computational gains that derive from the robustness to multicollinearity of neural networks to extend the optimal debt management problem studied by Faraglia et al. (2019) to four maturities. The hedging benefits provided by the additional maturities allow the government to respond to positive expenditure shocks by raising financial income without increasing the total outstanding debt. We show that, through this mechanism, the government uses the additional maturities to effectively subsidize the private sector in recessions, resulting in more leisure and less volatile labor taxes.

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*a*. "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. 2017*b*. "The Optimal Maturity of Government Debt." *Working Paper*.

Bigio, Saki, Galo Nuño, and Juan Passadore. 2022. "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*.

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

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*.

Glorot, Xavier, and Yoshua Bengio. 2010. "Understanding the difficulty of training deep feedforward neural networks." Vol. 9 of *Proceedings of Machine Learning Research*, 249–256. Chia Laguna Resort, Sardinia, Italy:PMLR.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Vol. Second Edition, Springer.

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

Ioffe, Sergey, and Christian Szegedy. 2015. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *CoRR*, abs/1502.03167.

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, and Serguei Maliar. 2022. "Deep learning classification: Modeling discrete labor choice." *Journal of Economic Dynamics and Control*, 135: 104295.

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." *American 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*.

ONLINE APPENDIX

A Machine Learning Projection Method for Macro-Finance Models

VYTAUTAS VALAITIS AND ALESSANDRO T. VILLA

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). The parameter ρ controls the substitutability in time and the parameter γ controls the attitude towards risk.[26] When $\rho = \gamma$, preferences collapse to CRRA, where consumption and leisure are additively separable. Preferences are

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

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

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

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

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

[26] When the 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.

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

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

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

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

The optimality condition for labor supply (FOC_h) is

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

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 p_t is the expected value of the stochastic discount factor

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

This problem can be generalized to N maturities in a similar fashion to section 3.2. This yields one Euler equation for each maturity $i = 1, \cdots, N$ that reads:

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

Ramsey Problem

The Ramsey problem consists of finding $\{c_t, \{b_{t+1}^i\}_{i=1}^N, \mu_t, V_t\}_{t=0}^\infty$ 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

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

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[27]

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

where λ_t^V 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.[28] The first order condition with respect to b_{t+1}^i yields the following inter-temporal expression for the promise keeping Lagrange multiplier μ

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

The first order condition with respect to V_t is

$$\beta^{t-i}\sum_{i=1}^{N}\pi(g^{t-i}|g^0)\beta^i\mu_{t-i}\pi(g^t|g^{t-i})b_{t-i+1}^iU_{c,t}U_t^{-\rho}\frac{\partial\mathcal{M}_{t-i}(V_t)}{\partial V_t(g^t)}-$$
$$\beta^{t-i+1}\sum_{i=1}^{N}\pi(g^{t-i+1}|g^0)\beta^{i-1}\mu_{t-i+1}\pi(g^t|g^{t-i+1})b_{t-i+1}^iU_{c,t}U_t^{-\rho}\frac{\partial\mathcal{M}_{t-i+1}(V_t)}{\partial V_t(g^t)}-$$
$$\lambda_t^V\beta^t\pi(g^t|g^0)+\beta^{t-1}\pi(g_{t-1}|g^0)\lambda_{t-1}^V\beta V_{t-1}^{\rho}\mathcal{R}_{t-1}(V_t)^{-\rho}\mathcal{M}_{t-1}(V_t)^{\frac{-\gamma}{\rho-\gamma}}\pi(g_t|g^{t-1})=0,$$

which, after rearranging, yields the following recursion for λ_t^V[29]

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

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

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

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

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

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

If $j < 0$:

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

If $j = 0$:

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

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.[30] 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:

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

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

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

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. (2017*b*) 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 $l_t = z_t - h_t$.[31] Figure 4 shows that in this setting 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 TFP shocks, which are orthogonal to government expenditure shocks, makes it risky to hold a highly leveraged position. This risk is magnified by Epstein-Zin preferences. Bhandari et al. (2017*b*) 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.

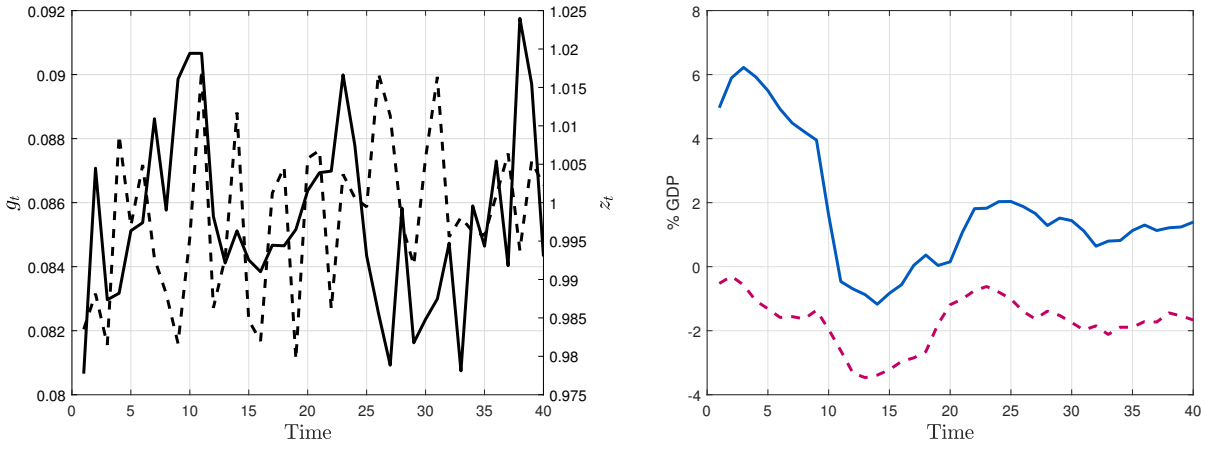

Figure 4: Simulated series with 2 bonds and Epstein-Zin preferences

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.

[31] z_t is independent of g_t and follows an AR(1) process $z_t = \mu_z + \rho_z z_{t-1} + \epsilon_t^z$ with $\mu_z = 0.9$, $\rho_z = 0.1$, $\epsilon \sim N(0, 0.0001)$.

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	$\phi_1 = 0.95$
constant in AR(1) process of in g_t	$c = 0.00416$
Variance of the disturbances to g_t	$\sigma^2_\epsilon = 0.00001$
AR(1) parameter in z_t	$\phi_z = 0.1$
constant in AR(1) process of in z_t	$c = 0.9$
Variance of the disturbances to z_t	$\sigma^2_{\epsilon_z} = 0.0001$
Borrowing limits	$\bar{M}^N, \bar{M}^S = 100\%$ of GDP $\underline{M}^N, \underline{M}^S = -100\%$ of GDP

Table 7: Parameter Values used in the model with Epstein-Zin preferences

B Implementation Details

This appendix includes three subsections: (i) additional algorithm details, (ii) an in-depth discussion about the relationship with GSSA and (iii) additional figures.

B.1 Algorithm Details

In this section we describe the practical details of the algorithm used to solve the model described in section 3.3. For explanation purposes, and as an example, we use the case with three bonds. In particular, we use three maturities $S = 1$, $M = 5$, and $L = 10$.

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 $b^N_{t+1} = \bar{M}^N$ and find the associated values for consumption and leisure. In a multiple-bond model this is challenging because after setting $b^N_{t+1} = \bar{M}^N$, 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: $\xi^i_{L,t} = \phi(\underline{M}^i - b^i_{t+1}) + \log(1 + \phi(\bar{M}^i - b^i_{t+1}))$ if $b^i_{t+1} < \underline{M}^i$ and $\xi^i_{U,t} = \phi(b^i_{t+1} - \bar{M}^i) + \log(1 + \phi(b^i_{t+1} - \bar{M}^i))$ if $b^i_{t+1} > \bar{M}^i$, 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

$$\mathcal{I}_t = \{g_t, \{b^S_{t-k}\}_{k=0}^{S-1}, \{b^M_{t-k}\}_{k=0}^{M-1}, \{b^L_{t-k}\}_{k=0}^{L-1}, \{\mu_{t-k}\}_{k=1}^{L}, \{\{\xi^S_{i,t-k}\}_{k=0}^{S-1}\}_{i=L,U}, \{\{\xi^M_{i,t-k}\}_{k=0}^{M-1}\}_{i=L,U}, \{\{\xi^L_{i,t-k}\}_{k=0}^{L-1}\}_{i=L,U}, \{\xi^{\text{Total}}_{i,t-1}\}_{i=L,U}\}.$$

With $S = 1$, $M = 5$, and $L = 10$ the 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.

$$\forall i: \quad \mu_t = \mathcal{ANN}^i_1(\mathcal{I}_t)^{-1}\left[\mathcal{ANN}^i_2(\mathcal{I}_t) + \frac{\xi^i_{U,t}}{\beta^i} - \frac{\xi^i_{L,t}}{\beta^i} + \frac{\xi^{\text{Total}}_{U,t}}{\beta^i} - \frac{\xi^{\text{Total}}_{L,t}}{\beta^i}\right]. \tag{9}$$

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 $\mathcal{I}_{t+1}$ to approximate $\mathbb{E}_{t+1}(u_c(c_{t+i}))$, $\mathbb{E}_{t+1}(u_c(c_{t+i-1}))$ and $\mathbb{E}_{t+1}(u_c(c_{t+i}\mu_{t+1}))$ for $i = S, M, L$. 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 $b^L_{t+1} = g_t/2 - \mathbb{E}(g_t/2); b^S_{t+1} = -b^L_t$ and $b^M_{t+1} = \sqrt{g_t/10} - \mathbb{E}(g_t/10)$. 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 $\xi^i_{L,t}$, $\xi^i_{U,t}$, $\xi^{\text{Total}}_{L,t}$, and $\xi^{\text{Total}}_{U,t}$. 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 run the stochastic simulation from the starting point where $b^S_1 = b^M_1 = b^N_1 = 0$ 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^L_{t+1}, b^M_{t+1}, and b^S_{t+1}. 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 2.3. We train the network using gradient descent with an adaptive learning rate. Table 8 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 8: NN hyperparameters

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

B.2 Relation to Condensed PEA

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 $X^T X$.[32] 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[33]

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

After rearranging, we get a simplified expression

$$\underbrace{n_{NN} \cdot T \cdot S \cdot M}_{\text{NN training}} < (n_{\text{CPEA}} - 1) \cdot \kappa(E, S, T) + \underbrace{n_{CPEA} \cdot E \cdot S^2 \cdot T}_{\text{CPEA regression}}. \tag{10}$$

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

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 $n_{CPEA} \cdot E \cdot S^2 \cdot T$ 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 $n_{CPEA} = 1$.[34] When $n_{CPEA} > 1$, the inequality in 10 clearly holds since $(n_{\text{CPEA}} - 1) \cdot \kappa(E, S, T) > 0$.

B.3 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. 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

[34] 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.

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 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. Hence, we do not use Montecarlo integration as in GSSA because that would require to integrate N-periods ahead. Instead, standard PEA does not require to perform any integration since it approximates the expectation terms directly.[35] 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.[36] Equation 11 illustrates the penalty selection procedure.

$$\begin{aligned} &\min_{\kappa} ||Y - X\hat{\boldsymbol{\beta}}||_2^2 \quad \text{s. t.} \\ &\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||Y - X\boldsymbol{\beta}||_2^2 + \kappa||\boldsymbol{\beta}||_2^2 \end{aligned} \tag{11}$$

[35] Note that when we use more than one maturity, we use Forward-States PEA and perform one step of integration using Gaussian quadrature as explained in section 3.3.

[36] 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.

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 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 9: 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 9 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 10).

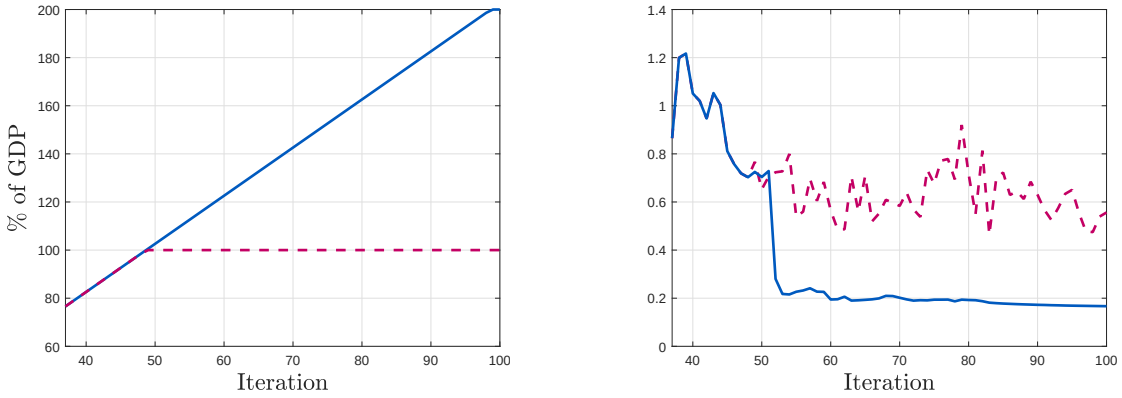

Figure 5: Convergence of model forecast errors using ridge regression

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 10: 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 $\frac{1}{T}\sum \left|Y_{t+N} - \hat{\mathbb{E}}_t(Y_{t+N})\right|$.

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.[37]

[37] 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.

B.4 Additional Figures

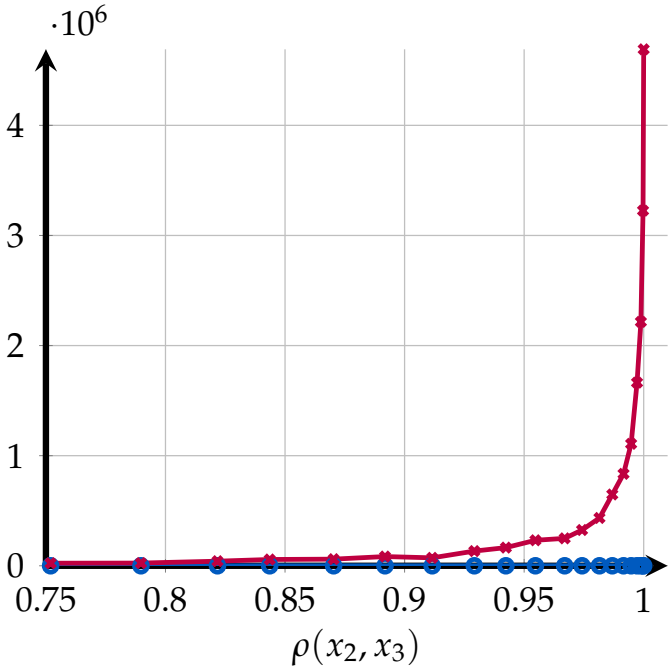

Figure 6: Bias squared of the mean squared prediction error with a neural network and a polynomial

Notes: The figure shows the bias squared of the mean squared prediction error $1/n \sum_{t=1}^{n} [y_t - \mathbb{E}(\hat{y}_t)]^2$ in function of the correlation between x_2 and x_3. Blue line with circles - NN, purple line with crosses - polynomial regression.

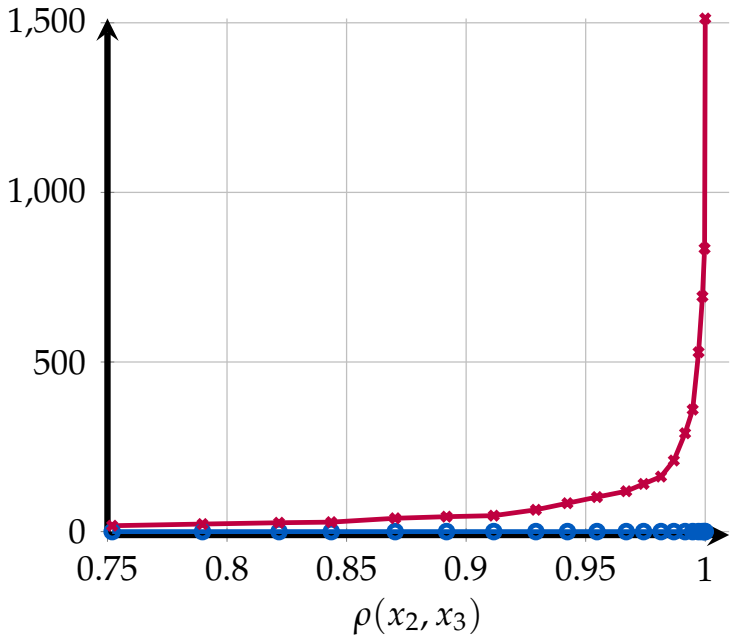

Figure 7: Variance term of the mean squared prediction error with a neural network and a polynomial

Notes: The figure shows the variance component of the mean squared prediction error $1/n \sum_{t=1}^{n} [y_t - \mathbb{E}(\hat{y}_t)]^2$ in function of the correlation between x_2 and x_3. Blue line with circles - NN, purple line with crosses - polynomial regression.

C Neural Network Details

This appendix includes further details about the type of neural network used in section 2.

C.1 Artificial Neural Networks

Neural networks can be used efficiently for both regression and classification purposes (in a supervised machine learning fashion) and clustering (in an unsupervised machine learning fashion). In the context of our application, we focus on the former. Neural networks 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.[38] 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 $\mathbb{E}\left[g_e(c_{t+1}, X_{t+1})|X_t\right] \simeq \mathcal{F}_e(X_t; w, \beta)$. See figure 8 for a graphical illustration.

[38] In our application, we consider a neural network with one hidden layer. The reader can refer to Goodfellow et al. (2016) for a general introduction to machine learning and deep learning in particular.

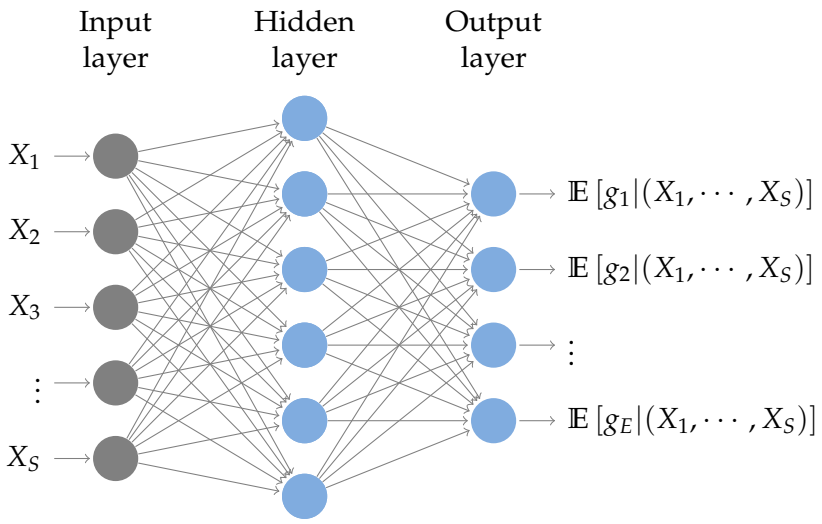

Figure 8: Artificial Neural Network Structure

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 in 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.

If the problem requires the approximation of E expectations terms, a neural network with one hidden layer has the following functional form

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

Note that 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. 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 $\mathcal{H}(x)$. The activation function is typically a sigmoid

$$\mathcal{H}(x) = \frac{1}{1 + \exp(-x)}.$$

In order to gain intuition about the behavior of this function, consider a more generic function $\tilde{\mathcal{H}}(x; \alpha) = \frac{1}{1+\exp(-\alpha \cdot x)}$, where α is a parameter that regulates the activation rate. Intuitively, the

larger is α, the more $\tilde{\mathcal{H}}(x;\alpha)$ resembles to a step function as shown in figure 9.[39] Note that neural networks are equivalent to linear regression if the activation function $\tilde{\mathcal{H}}$ is linear.[40]

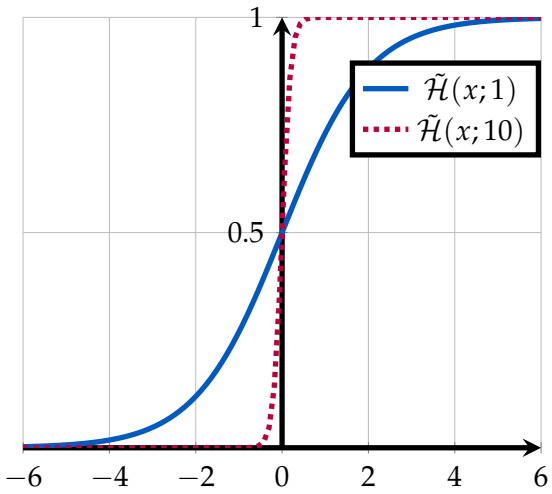

Figure 9: Sigmoid Function

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

[39] In the context of deep learning (in contrast to shallow neural networks), the de-facto standard is rather the rectifier (or ReLU) or swish activation functions.

[40] Setting $\tilde{\mathcal{H}}$ to a linear function would be equivalent to approximating the expectations with linear polynomials in the traditional PEA algorithm.