# CAHIER D'ÉTUDES
# WORKING PAPER

# N° 172

# ARTIFICIAL NEURAL NETWORKS TO SOLVE DYNAMIC PROGRAMMING PROBLEMS: A BIAS-CORRECTED MONTE CARLO OPERATOR

JULIEN PASCAL

MARCH 2023

BANQUE CENTRALE DU LUXEMBOURG
EUROSYSTÈME

# ARTIFICIAL NEURAL NETWORKS TO SOLVE DYNAMIC PROGRAMMING PROBLEMS: A BIAS-CORRECTED MONTE CARLO OPERATOR

JULIEN PASCAL

Abstract. Artificial Neural Networks (ANNs) are powerful tools that can solve dynamic programming problems arising in economics. In this context, estimating ANN parameters involves minimizing a loss function based on the model's stochastic functional equations. In general, the expectations appearing in the loss function admit no closed-form solution, so numerical approximation techniques must be used. In this paper, I analyze a bias-corrected Monte Carlo operator (bc-MC) that approximates expectations by Monte Carlo. I show that the bc-MC operator is a generalization of the all-in-one expectation operator, already proposed in the literature. I propose a method to optimally set the hyperparameters defining the bc-MC operator and illustrate the findings numerically with well-known economic models. I also demonstrate that the bc-MC operator can scale to high-dimensional models. With just a few minutes of computing time, I find a global solution to an economic model with a kink in the decision function and more than 100 dimensions.

Keywords: Dynamic programming, Artificial Neural Network, Machine Learning, Monte Carlo.

---

March 2023. Julien Pascal: Banque centrale du Luxembourg, Département Économie et Recherche, 2 boulevard Royal, L-2983 Luxembourg (julien.pascal@bcl.lu). I thank Pablo Garcia-Sanchez, Alban Moura, Paolo Guarda and other BCL colleagues for useful comments. The results in this paper/presentation are preliminary materials circulated to stimulate discussion and critical comment. References in publications should be cleared with the authors. This paper/presentation should not be reported as representing the views of the BCL or the Eurosystem. The views expressed are those of the authors and may not be shared by other research staff or policymakers in the BCL or the Eurosystem.

## Résumé Non Technique

La transmission de la politique monétaire peut varier avec l'hétérogénéité des agents économiques. Par exemple, les consommateurs ayant différents niveaux de richesse peuvent répondre différemment aux changements de taux d'intérêt. D'autre part, les consommateurs appartenant à différentes catégories d'âge peuvent avoir des horizons de planification différents. Les entreprises avec moins de chiffre d'affaires ou de capital peuvent se trouver confrontées à des contraintes de crédit lorsqu'elles souhaitent financer de nouveaux investissements. Enfin, les banques avec moins de capital ou de liquidité peuvent être caractérisées par une offre de crédit qui est moins sensible aux changements de politique monétaire.

Les économistes ont développé des modèles théoriques de plus en plus complexes pour prendre en considération ces différences entre ménages, entreprises ou banques. Les modèles économiques de grande dimension apparaissent par exemple lorsqu'on considère plusieurs secteurs ou plusieurs pays, des modèles d'économie de l'environnement et des ressources naturelles, ou des modèles avec différents types d'actifs ou différents marchés. Parmi les exemples les plus courants, on peut citer les modèles dans lesquels les agents diffèrent par leur âge, tels que les modèles à générations imbriquées (OLG), ou les modèles dans lesquels les agents diffèrent selon leur richesse, tels que les modèles néo-keynésiens à agents hétérogènes (HANK).

La résolution de ces modèles de grande dimension requiert de nouvelles approches. Cet article présente une nouvelle méthode pour utiliser les réseaux de neurones artificiels (RNAs) pour résoudre des modèles économiques. Les RNAs sont des outils puissants pour l'analyse économique, car ils permettent d'approximer efficacement les solutions de modèles économiques complexes de grande dimension, pour lesquels les méthodes plus traditionnelles échouent souvent ou sont impossibles. Ce papier contribue de trois manières différentes à la littérature : il (i) généralise une approche déjà existante, tout en lui offrant une nouvelle base théorique (ii) illustre la nouvelle méthode proposée en résolvant des modèles économiques bien connus (iii) compare la performance des RNAs par rapport aux autres méthodes traditionnelles en considérant les conditions qui peuvent favoriser une approche par rapport aux autres.

## Non-Technical Summary

Central banks are aware that the transmission of monetary policy depends on heterogeneity across agents. Consumers with different wealth levels may respond differently to changes in interest rates. Consumers of different ages may have different planning horizons. Firms with lower turnover or capital may have more limited access to credit to finance new investments. Finally, banks with lower levels of capital or liquidity may supply less credit.

Economists have developed increasingly complex theoretical models to allow for these differences across households, firms or banks. For instance, high-dimension models arise when considering multiple sectors or multiple countries, environmental and natural resource economics, or multiple asset types or multiple markets. Some leading examples include models in which agents differ by their age, as in the overlapping generations (OLG) models, or models in which agents differ by their wealth, as in Heterogeneous Agent New Keynesian (HANK) models.

Solving these high-dimensional models requires new methods. This paper presents a new method to apply artificial neural networks (ANNs) to solve economic models. ANNs are attractive tools for economists because they can very efficiently approximate the solutions to complex high-dimensional economic models, where more traditional methods often fail or are not feasible. This paper makes three contributions: it (i) generalizes an existing ANN method to solve economic models, providing it with new theoretical foundations (ii) illustrates how to use this new method to solve well-known economic models (iii) discusses when ANNs perform better than more traditional methods.

## 1. Introduction

This paper belongs to the burgeoning field that applies machine learning tools to solve economic models. In particular, Artificial Neural Networks (ANNs) have three main advantages for economists. First, ANNs have been shown to be universal function approximators, in the sense that any Borel measurable function from one finite dimensional space to another can be approximated by an ANN (Hornik, Stinchcombe, and White, 1989).[1] In almost all cases, the value and policy functions arising from economic models are within the set of functions that can be approximated by ANNs (Blackwell, 1965). Second, ANNs are resilient to the curse of dimensionality, which usually restricts economists to work with low-dimensional models or to rely on linearization around a steady-state (Barron, 1993). Third, ANNs are usually estimated with the backpropagation algorithm that allows them to learn efficiently and quickly the underlying model (Rumelhart, Hinton, and Williams, 1986).[2]

The fact that ANNs are efficient universal function approximators in high-dimensional spaces was illustrated when a deep artificial neural network defeated the world leader at the game of Go, which has a state-space of $3^{361}$ (Silver et al., 2016).[3] While state-of-the-art economic models have more modest state-spaces, recent developments in economics have led the number of dimensions to explode. Models with partially insurable risk have a state-space which often includes the distribution of agents across some dimension, usually wealth (Aiyagari, 1994 and Krusell and Smith, 1998). Heterogeneity of households is a key ingredient of modern macroeconomic HANK models, because it affects the transmission mechanism of monetary policy (Kaplan, Moll, and Violante, 2018). High-dimensional state-spaces also arise from models with overlapping generations (Krueger and Kubler, 2006), from studies of firm dynamics (Khan and Thomas, 2008), or from multi-country models (Backus, Kehoe, and Kydland, 1992). Such models are usually solved by linearization around the non-stochastic steady-state. However, global methods

[1] In this paper, I use the term ANN to denote the standard multi-layer feedforward network (see for instance Schmidhuber (2015)).

[2] The backpropagation algorithm can be very efficiently executed on specialized units such as GPUs, which can reduce computing times by orders of magnitude. While alternative methodologies such as Value Function Iteration or Time Iteration may be amenable to GPUs, careful case-specific implementations may be warranted.

[3] The game of Go with a $19 \times 19$ grid has 361 placement spaces. Each space can be unoccupied or contain either colored stone, or no stone.

are sometimes preferable because (i) linearization may eliminate interesting amplification mechanisms[4] (ii) a non-stochastic steady-state may not exist in the first place.[5]

One peculiarity of using ANNs in the present context is that there is a potentially infinite number of observations that can be used for training.[6] In fact, the Bellman or Euler equations characterizing a model must hold on the entire continuous domain on which the problem is defined. Thus, there are no restrictions on the level of refinement a researcher may select when sampling values from the state-space. This differs from more common machine learning applications in which one attempts to predict an outcome using a finite sample (Goodfellow, Bengio, and Courville, 2016). As in Maliar, Maliar, and Winant (2021), I define a loss function that must hold in expectation over the state variables. I approximate the expectation over state variables using Monte Carlo integration, so that the method can solve models with high-dimensional state variables. Most economic models also involve a second type of expectation: Bellman or Euler equations often include an expectation with respect to the value of next period's shocks. I also use Monte Carlo integration to approximate the expectation over shock variables, so that the methodology developed in this paper can also be used to solve models with high-dimensional shock variables.

For a given value of the state vector, the Bellman or Euler equations characterizing the model define a system of $J$ stochastic functional equations that must be equal to zero in expectation. To measure the distance from zero, I use the $L^2$ norm, which gives rise to a loss function containing the square of expectations. It is well known that using the square of the empirical mean to estimate the square of the population mean leads to a small-sample bias (see Das, 1975). To correct for this bias, I use the minimum variance unbiased estimator (MVUE) of the square of the mean, which contains a bias-correction term. One key result of the paper is that using the MVUE to approximate the square of expectations transforms a problem with two nested expectations into a

[4]Linearized solutions imply certainty equivalence, which makes it difficult to talk about the welfare effects of uncertainty (Fernández-Villaverde, Rubio-Ramírez, and Schorfheide, 2016).

[5]See Coeurdacier, Rey, and Winant (2011) for a discussion. For example, a non-stochastic steady-state may not exist for models of small open economies (Schmitt-Grohé and Uribe, 2003) or models of portfolio choices (Devereux and Sutherland, 2011).

[6]The term "training" in the machine learning literature refers to the estimation of ANN parameters.

problem involving a single expectation. I call this non-nested integral a bias-corrected Monte Carlo (bc-MC) operator.[7]

I show that the all-in-one operator of Maliar, Maliar, and Winant (2021) is a special case of the bc-MC operator. The former only uses two independent shocks for each realization of the state variable, while the bc-MC operator may use $N$ independent shocks for each realization of the state variable. I propose a criterion to optimally choose the number of independent shocks, which minimizes the integration error of the loss function. Because the bc-MC operator is an unbiased estimator of the loss function, the integration error is equal to the variance of the loss, which can be easily estimated by simulation. I illustrate the performance of our bc-MC operator using the stochastic neoclassical growth model and a model with a borrowing constraint. I also compare the performance the methodology based on the bc-MC operator to the Time Iteration (TI) algorithm and show that the former has superior scaling properties. When the dimension of the problem gets large, the bc-MC operator clearly outperforms the TI algorithm when considering both speed and accuracy. Indeed, I solve a model with a borrowing constraint using a global method with more than 100 dimensions on a consumer-grade laptop in approximately 4 minutes.

This paper proceeds as follows. I first discuss how the method fits within the existing literature. I then formally introduce the bc-MC operator and discuss the optimal choice of hyperparameters. The next section illustrates the performance of the bc-MC operator using standard models from the economic literature. The final section concludes.

## 2. Related literature

This paper is related to the literature that develops global methods to solve high-dimensional economic models for which linearization techniques are ill-suited. When linearization around a steady-state is not appropriate, traditional global methods based on

[7]It would be more precise to name it the "bias-corrected Monte Carlo estimator", as an estimator is a function used to estimate an unknown population parameter based on a sample of data. In this paper, I am using an estimator to calculate an estimate of the model's loss function based on simulated data. The expectation operator is a mathematical operation that maps a random variable $X$ to its expected value denoted as $\mathbb{E}(X)$. However, Maliar, Maliar, and Winant (2021) use the term "all-in-one expectation operator" when developing their method for solving economic models with ANN. I will adopt their terminology to align my contribution with their work.

dense grids very rapidly lose tractability because of the curse of dimensionality. As noted by Bellman (1961), the number of grid points grows exponentially with the dimension of the problem. Solutions have been developed to deal with the curse of dimensionality. Krueger and Kubler (2004) and Judd et al. (2014) advocate the use of methodologies based on Smolyak's algorithm to compute sparse grids (Smolyak, 1963). Brumm and Scheidegger (2017) develop a method based on adaptive sparse grids, for which grid points are only added where they are most needed, for instance in regions with steep gradients or kinks. Scheidegger and Bilionis (2019) develop a computational framework that can compute global solutions to high-dimensional economic models using Gaussian processes. As discussed in Azinovic, Gaegauf, and Scheidegger (2022) none of the approaches can satisfactorily solve an economic model with (a) stochasticity, (b) a very high-dimensional state space, (c) strong non-linearities or kinks, (d) irregular geometries for the ergodic set of state variables.

In this paper, I build upon the literature that uses machine learning methods, especially Artificial Neural Networks (ANNs), to generate computational methods that can cope with the combination of (a)-(d). Machine learning methods have already been successfully applied to solve large scale differential equations arising from physical systems (Raissi and Karniadakis, 2018 and Raissi, Perdikaris, and Karniadakis, 2019). They have also been applied in economics. In a model with heterogeneous agents, Fernández-Villaverde, Hurtado, and Nuno (2019) use an ANN to approximate agents' perceived law of motion, instead of using a log-linear function, as it in most applications of the Krusell and Smith algorithm. However, the authors rely on a traditional finite-difference scheme to solve for the Hamilton-Jacobi-Bellman equation characterizing the model.

This paper is more closely related to the work of Azinovic, Gaegauf, and Scheidegger (2022), who solve a large-scale OLG model by approximating the policy function with an ANN and minimizing a loss function that takes into account the stochastic difference equations characterizing the model. However, the authors assume discrete shocks and can therefore exactly evaluate the expectation with respect to next's period shocks. The authors note that continuous shocks would require numerical integration methods, but they observe that the computational cost of approximating the expectation operator grows linearly with the number of quadrature nodes. In independent work, Maliar, Maliar, and Winant (2021) provide an elegant solution to this numerical integration

problem by using Monte Carlo integration to approximate the expectation with respect to next period's shocks. However, the resulting loss function contains two nested integrals, which are both slow to evaluate and usually inefficient (see Rainforth et al., 2018). The authors make a technical contribution by showing how two nested integrals can be transformed into a single integral using two series of independent shocks following the same distribution, which they call the all-in-one operator.[8] In this paper, I build upon this "unnesting" idea and extend it.

I make three main contributions to the existing literature. First, I introduce a new estimator that generalizes the all-in-one operator proposed by Maliar, Maliar, and Winant (2021). Instead of using two series of independent shocks, one may use $N$ series of independent shocks. In doing so, I provide a new theoretical derivation for the all-in-one operator and its generalization. I show that the all-in-one operator is the result of using an unbiased estimator for the square of the mean when using Monte Carlo integration to approximate expectations. This is why I call this estimator a bias-corrected Monte Carlo operator (bc-MC). Second, I derive theoretical properties characterizing the bc-MC operator. This allows me to provide actionable recommendations regarding the optimal choice of the hyperparameters characterizing the bc-MC operator, taking into consideration the fact that ANNs are trained by stochastic gradient descent or its variants. I illustrate these recommendations with two different economic models. Third, I solve a high-dimensional model using the bc-MC operator methodology and compare the results to the ones obtained using the Time Iteration (TI) algorithm with a dense grid. In a low-dimensional setting, the TI algorithm outperforms the bc-MC methodology, but in a high-dimensional setting the bc-MC operator outperforms the TI algorithm by several order of magnitudes. Hence, I provide new insights on when it is preferable to use machine learning methods to solve economic models.

## 3. The Bias-Corrected Monte Carlo Operator

In this section, I discuss how solving a general economic model can be reformulated as the problem of finding the value of a parameter vector $\theta$ that minimizes a loss function derived from the equations in the model. I focus on the case when $\theta$ is the parameter vector of an ANN, which can be trained by gradient descent or its variants. I then

[8]This approach has also been used in the non-economic literature by Goda (2017) and Rainforth et al. (2018).

introduce the bc-MC operator, which uses Monte Carlo techniques to approximate expectations and correct for a small-sample bias. I then derive some key properties of the bc-MC operator, which allows me to discuss how best to choose the hyperparameters of the bc-MC operator.

3.1. **Structure of economic models.** Many economic models are characterized by a stochastic functional equation of the form:[9]

$$\mathbb{E}_\varepsilon \Big( f(s, \epsilon) \Big) = 0 \quad \text{for } \forall s \in S \tag{1}$$

where $s$ is vector of state variables and $\varepsilon$ a vector of i.i.d. shocks occurring after decisions are made. Equation (1) is usually a Bellman or an Euler equation. A parametric approximation of the solution $f(s, \epsilon|\theta)$ can be obtained by minimizing a loss function of the form:

$$\mathcal{L}(\theta) = \mathbb{E}_s \left[ \mathbb{E}_\varepsilon \Big( f(s, \epsilon|\theta) \Big)^2 \right] \tag{2}$$

The inner integral is the usual integration with respect to the value of next period's shock. The inner integral is squared, which corresponds to finding the value of $\theta$ that minimizes the distance of $\mathbb{E}_\varepsilon \Big( f(s, \epsilon|\theta) \Big)$ from 0 using the $L^2$ norm.

The outer integral ensures that the stochastic functional equation holds for all values of the state vector $s$ in its domain $S$. Treating the vector $s$ as random vector allows us to use Monte Carlo integration, which is tractable even in high-dimensional settings. An alternative would consist in selecting a grid for $s$ and finding the value for $\theta$ such that (2) is minimized across these grid points. However, the number of grid points grows exponentially with the number of dimensions.

Since ANNs are universal function approximators (Hornik, Stinchcombe, and White, 1989) that work well in high-dimensional settings (Barron, 1993), I focus on the case in which $\theta$ is the parameter vector of an ANN. In this context, solving an economic model

[9]To simplify the presentation, I focus on the case in which the model is characterized by a single stochastic functional equation. For the case where the economic system is characterized by a system of $J$ stochastic equations, see section A of the Appendix.

consists in finding the value for $\theta$ that minimizes the loss function (2).[10] ANNs are generally trained using the stochastic gradient algorithm or its variants. The stochastic gradient algorithm is an iterative procedure in which the new guess for $\theta$ is updated using information from the gradient of the loss function:

$$\theta_{i+1} = \theta_i - \gamma \nabla_\theta \mathcal{L}(\theta_i) \tag{3}$$

where $\gamma$ is a parameter called the learning rate and $\nabla_\theta \mathcal{L}(\theta_i)$ denotes the gradient of loss function evaluated at the current guess $\theta_i$. One difficulty with (3) is that closed-form solutions for (2) and its gradient are generally unavailable. Instead, one must use an approximation for the loss function.

3.2. **Bias-corrected Monte Carlo operator.** In general, there are no closed-form solutions for the expectations appearing in (2). To approximate these expectations, one may use Monte Carlo integration separately for the two integrals ($\mathbb{E}_s$ and $\mathbb{E}_\varepsilon$):

$$\mathcal{L}^M_{M,N}(\theta) = \frac{1}{M} \sum_{m=1}^{M} \Big[ \frac{1}{N} \sum_{n=1}^{N} f(s_m, \epsilon_n | \theta) \Big]^2 \tag{4}$$

The next proposition shows that the estimator introduced in equation (4) has a small-sample bias.

**Proposition 1.** *The bias of* $\Big[ \frac{1}{N} \sum_{n=1}^{N} f(s_m, \epsilon_n | \theta) \Big]^2$ *is equal to* $\frac{\sigma^2_{f,s_m}}{N}$*, with* $\sigma^2_{f,s_m}$ *the variance of* $f$ *conditional on the value* $s = s_m$*.*

*Proof.* Using the equality $\mathrm{Var}[f(x)] = \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2$ and the fact that the variance of the sum of independent variables is equal to the variance of the sum, one finds:

$$\begin{aligned} &\mathbb{E}_\epsilon[\Big(\frac{1}{N} \sum_{n=1}^{N} f(s_m, \epsilon_n | \theta)\Big)^2] = \Big(\mathbb{E}_\epsilon[\frac{1}{N} \sum_{n=1}^{N} f(s_m, \epsilon_n | \theta)]\Big)^2 + \mathrm{Var}_\epsilon(\frac{1}{N} \sum_{n=1}^{N} f(s_m, \epsilon_n | \theta)) \\ &\mathbb{E}_\epsilon[\Big(\frac{1}{N} \sum_{n=1}^{N} f(s_m, \epsilon_n | \theta)\Big)^2] = \mu^2_{s_m} + \frac{\sigma^2_{f,s_m}}{N} \end{aligned} \tag{5}$$

where $\mu^2_{s_m} \equiv \mathbb{E}_\epsilon[f(s_m, \epsilon|\theta)]^2$ and $\sigma^2_{f,s_m} \equiv \mathrm{Var}_\epsilon(f(s_m, \epsilon|\theta))$. □

[10] Multi-layer feedforward networks usually have highly non-convex loss functions, with many local minima. However, while local minima are numerous, they lead to very similar predicted values (Choromanska et al., 2015).

Because the sample variance is an unbiased estimator of the population variance, proposition 1 suggests the following bias-corrected Monte Carlo operator, defined for $N \geq 2$:[11]

$$\mathcal{L}^{U}_{M,N}(\theta) = \frac{1}{M}\sum_{m=1}^{M}\left\{\Big[\frac{1}{N}\sum_{n=1}^{N} f(s_m, \epsilon_n|\theta)\Big]^2 - \frac{S^2_{m,n}}{N}\right\} \tag{6}$$

with $S^2_{m,n} \equiv \frac{1}{N-1}\sum_{n=1}^{N}\Big(f(s_m, \epsilon_n|\theta) - \hat{\mu}_{s_m}\Big)^2$ and $\hat{\mu}_{s_m} \equiv \frac{1}{N}\sum_{n=1}^{N} f(s_m, \epsilon_n|\theta)$.

The operator in equation (6) relies on solid theoretical foundations, because in the class of unbiased estimators of the square of the population mean $\mu^2_{s_m}$, the one with the smallest variance is exactly given by $\hat{\mu}^2_{s_m} - \frac{S^2_{m,n}}{N}$. In statistical terms, $\hat{\mu}^2_{s_m} - \frac{S^2_{m,n}}{N}$ is the minimum variance unbiased estimator (MVUE) of $\mu^2_{s_m}$ (see for instance Das (1975)).

The next proposition shows that the bias-corrected Monte Carlo operator defined in equation (6) is a generalization of an estimator that has already been proposed in the literature.

**Proposition 2.** *(1) The bias-corrected Monte Carlo operator* (6) *can be expressed as*

$$\mathcal{L}^{U}_{M,N}(\theta) = \frac{2}{MN(N-1)}\sum_{m=1}^{M}\sum_{1\leq i<j}^{N} f(s_m, \epsilon^i_m|\theta) f(s_m, \epsilon^j_m|\theta) \tag{7}$$

*where $\epsilon^i$ and $\epsilon^j$ are i.i.d. shocks with the same distribution as $\epsilon$.*

*(2) In the special case with $N = 2$*

$$\mathcal{L}^{U}_{M,2}(\theta) = \frac{1}{M}\sum_{m=1}^{M} f(s_m, \epsilon^1_m|\theta) f(s_m, \epsilon^2_m|\theta)$$

*Proof.* See Appendix B. □

Proposition 2 shows that using the MVUE to approximate the square of the inner expectation has the virtue of transforming the two nested integrals into a single non-nested integral. This matters because nesting Monte Carlo estimators generally leads to efficiency losses (see Rainforth et al., 2018).

Proposition 2 also shows that the bias-corrected Monte Carlo operator generalizes the all-in-one expectation operator of Maliar, Maliar, and Winant (2021). Indeed, given a

[11] Note that $\mathcal{L}^{U}_{M,N}(\theta)$ is defined for $N \geq 2$, because the calculation of the sample variance $S^2_{m,n}$ is needed.

desired number of function evaluations required to calculate the loss function equal to $2T$ with $T \equiv \frac{MN}{2}$, one may use two function calls for each value of $s_m$, which is exactly the all-in-one expectation operator. Alternatively, one may use $T$ function calls for each value of $s_m$, which increases the accuracy of the expectation $\mathbb{E}_\varepsilon$, at the detriment of the accuracy with respect to $\mathbb{E}_s$, because only two function calls are left for each value of $\epsilon_m^i$.

For a given $T$, proposition 2 does not determine the optimal choice of $M$ and $N$. A natural criterion consists in selecting the combination of $M$ and $N$ that minimizes the integration error given by the distance of $\mathcal{L}_{M,N}^U(\theta)$ from its true value $\mathcal{L}(\theta)$. Because I use Monte Carlo integration, the integration error is a random variable, so I choose $M$ and $N$ to minimize the mean squared integration error. The next proposition shows that the mean squared integration error is equal to the variance of the loss function.

**Proposition 3.** *Let $e_{M,N}(f|\theta)$ denote the integration error, defined as*

$$e_{M,N}(f|\theta) \equiv \mathbb{E}_s \left[ \mathbb{E}_\varepsilon \left( f(s, \epsilon|\theta) \right)^2 \right] - \mathcal{L}_{M,N}^U(\theta) \tag{8}$$

*The mean squared integration error is equal to*

$$\mathbb{E}\left[ e_{M,N}(f|\theta)^2 \right] = \mathrm{Var}(\mathcal{L}_{M,N}^U(\theta)) \tag{9}$$

*Proof.* See Appendix C. □

The mean squared error of an estimator can be written as the sum of the variance of the estimator and the square of its bias. The fact that the mean squared integration error is equal to the variance of the loss function is a direct consequence of the fact that the bc-MC operator is an unbiased estimator of the loss function, independent of the choice of $M$ and $N$.

Choosing $M$ and $N$ to minimize the variance of the loss function can be further justified by noting that ANNs are trained by stochastic gradient descent (or its variants). At each step of gradient descent, the value of the loss function changes for two reasons: (i) because the new value for $\theta$ leads to a value of $\mathcal{L}_{M,N}^U(\theta)$ that is closer to the true value $\mathcal{L}(\theta)$ (ii) because the values of $s$ and $\epsilon$ used to approximate the expectation have changed. The smaller the noise caused by Monte Carlo integration, the more accurate is the gradient

descent step, and the faster the algorithm converges (see for instance Katharopoulos and Fleuret, 2018).

Intuitively, in a model the role of uncertainty is limited[12], one would like to have a high value for $M$ and a small value for $N$. One would also expect that models that are more non-linear with respect to the state space would require a higher $M$. The next proposition formalizes this intuition.

**Proposition 4.** *(1) The variance of the loss function can be expressed as*

$$\begin{aligned}\mathrm{Var}(\mathcal{L}^U_{M,N}(\theta)) = \frac{1}{T(N-1)}\Big[ &\mathrm{Var}\left(f(s_m,\varepsilon^1_m|\theta)f(s_m,\varepsilon^2_m|\theta)\right)\\ &+2\big(N-2\big)\,\mathrm{Cov}(f(s_m,\varepsilon^1_m|\theta)f(s_m,\varepsilon^2_m|\theta),f(s_m,\varepsilon^1_m|\theta)f(s_m,\varepsilon^3_m|\theta))\\ &+2\Big(\frac{N(N-1)}{4}-N+\frac{3}{2}\Big)\,\mathrm{Cov}(f(s_m,\varepsilon^1_m|\theta)f(s_m,\varepsilon^2_m|\theta),f(s_m,\varepsilon^3_m|\theta)f(s_m,\varepsilon^4_m|\theta))\Big]\end{aligned} \tag{10}$$

*where $\varepsilon^1$, $\varepsilon^2$, $\varepsilon^3$ and $\varepsilon^4$ are four independent random variables, with the same distribution as $\varepsilon$.*

*(2) If $f(s_m,\varepsilon_m|\theta)=f(\varepsilon_m|\theta), \quad \forall s\in S$:*

$$\mathrm{Var}(\mathcal{L}^U_{M,N}(\theta)) = \frac{1}{T(N-1)}\,\mathrm{Var}\left(f(s_m,\varepsilon^1_m|\theta)\right)^2+\frac{2}{T}\,\mathbb{E}\left[f(s_m,\varepsilon^1_m|\theta)\right]^2\mathrm{Var}\left(f(s_m,\varepsilon^1_m|\theta)\right) \tag{11}$$

*(3) If $f(s_m,\varepsilon_m|\theta)=f(s_m|\theta), \quad \forall \varepsilon_m\in\mathcal{E}$:*

$$\mathrm{Var}(\mathcal{L}^U_{M,N}(\theta)) = \frac{1}{M}\,\mathrm{Var}\left(f(s_m,\varepsilon^1_m|\theta)^2\right) \tag{12}$$

*Proof.* See Appendix D. □

The first part of proposition 4 shows that, for whatever the choice of $M$ and $N$, the variance of the loss function can be decreased by increasing $T$, which corresponds to an increase in the total number of function evaluations required for each evaluation

[12]In a single-innovation process model, an increase in economic uncertainty is commonly characterized as an increase in the variance of the innovation process. In the following numerical section, I will explore this form of uncertainty. Other perspectives are that uncertainty increases when the number of innovation processes increases, or when the likelihood of rare events rises.

of the loss function. One drawback of increasing $T$ is the corresponding increase in computational time.

The second and third parts of proposition 4 suggest ways to reduce the variance of the loss function, for a constant value of $T$. On the one hand, in the limit case in which the state vector does not contribute to the variance of the loss function, the optimal choice is the maximum possible value for $N$: $M = 2$ and $N = T$ (proposition 4.2). On the other hand, in the limit case in which the shock vector does not contribute to the variance of the loss function, the optimal choice consists is the maximum possible value for $M$: $M = T$ and $N = 2$ (proposition 4.3).

To summarize, the bigger the role of shocks in the model, the more $\varepsilon$ contributes to the overall variance of the loss function, and the more likely the optimal choice is $N = T$. The less smooth is the state-space (for instance because of borrowing constraints), the more $s$ contributes to the overall variance of the loss function, and the more likely the optimal choice is $N = 2$. I illustrate these properties in the next section.[13]

## 4. Numerical illustration

In this section I solve three economic models using the bc-MC methodology. I first solve a simple stochastic growth model, which illustrates the role of uncertainty in the optimal choice of hyperparameters $M$ and $N$. I then solve an optimal consumption-saving model with a borrowing constraint, which highlights how curvature and non-smoothness in the state space affects $M$ and $N$. In the third numerical exercise, I increase the dimension of the consumption-saving model with a borrowing constraint and see how it affects the performance of the bc-MC methodology.

### 4.1. **Stochastic growth model.**

Consider an agent maximizing her inter-temporal discounted utility

$$\max_{\{c_t\}_{t=0}^{\infty}} \mathbb{E}\Big[\sum_{t=0}^{\infty} \beta^t u(c_t)\Big] \tag{13}$$

[13]Note that proposition 4 is not a statement on the variance of the state or shocks vectors, but rather a statement on the variance of $f(s_m, \varepsilon_m|\theta)$.

with $y_0$ given.[14] She enters period $t$ with income $y_t$ and chooses her consumption level $c_t$ subject to the constraints $0 \leq c_t \leq y_t$. What is saved in the current period is used for production to generate income next period. Income next period is stochastic and takes the following form: $y_{t+1} = g(y_t - c_t)\eta_{t+1}$ with $\eta$ denoting i.i.d. productivity shocks. Let us assume that preferences can be represented by a logarithmic utility function $u(c) = \log(c)$, the production function is Cobb-Douglas $g(k) = k^\alpha$, the discount factor is positive but strictly less than one, $\beta \in (0, 1)$, and that productivity shocks are lognormally distributed $\eta_t \equiv \eta(\nu_t) = \exp(\mu + \sigma_\nu \nu_t)$, with $\nu$ a standard Normal random variable.

Solving the model involves finding the value of parameter vector $\theta$ such that the consumption function $c(y|\theta)$ satisfies the Euler equation characterizing the model:

$$\mathbb{E}_\nu \left[ u'\Big(c(y|\theta)\Big) - \beta u'\Big(c\Big(g\big(y - c(y|\theta)\big)\eta(\nu)\Big|\theta\Big) g'\big(y - c(y|\theta)\big)\eta(\nu)\Big)\right] = 0 \tag{14}$$

Equation (14) is one instance of equation (1), with the state variable $s = y$, the shock variable $\varepsilon = \nu$, and the function $f(s, \varepsilon) = u'\Big(c(s|\theta)\Big) - \beta u'\Big(c\Big(g\big(s - c(s|\theta)\big)\eta(\varepsilon)\Big|\theta\Big) g'\big(s - c(s|\theta)\big)\eta(\varepsilon)\Big)$. It is easy to verify that the optimal behavior for the agent is to consume a fixed fraction of her cash on hand $y$. More specifically, the optimal consumption policy function is given by $c^*(y) = (1 - \alpha\beta)y$.

To illustrate how the hyperparameters $M$ and $N$ in the loss function (7) affect the accuracy of the approximation, I solve the model using the bias-corrected Monte Carlo operator for several choices of $M$ and $N$. For the architecture of the ANN, I use a sigmoid function for the final activation function, which ensures that the budget constraint is respected. Hence, the ANN outputs a number between 0 and 1, which I interpret as the fraction of income that is consumed, denoted by $\phi(y|\theta)$. Thus, the consumption function parameterized by an ANN has the following form $c(y|\theta) = \phi(y|\theta)y$. I use a single node

[14] This model is a special case of the model developed by Brock and Mirman (1972), with full depreciation of capital, which admits a closed-form solution. For textbook treatments of this model, refer to Stokey (1989) or Ljungqvist and Sargent (2018).

because we know that the solution is extremely simple: the ANN must learn to output a constant.[15]

In terms of parametrization, I set the production parameter $\alpha = 0.4$ and the discount factor $\beta = 0.96$. Regarding the uncertainty parameters, I set $\mu = 0$ and try two values for $\sigma_\nu$, 0.5 in the low-uncertainty parametrization and 1.5 in the high-uncertainty parametrization. In both cases, I solve the model using the loss functions defined by the bias-corrected Monte Carlo operator (7). For the optimization algorithm, I use the stochastic gradient algorithm with a learning rate $\gamma = 1.10^{-1}$. Training is stopped after 1000 steps of the stochastic gradient algorithm. Because the loss function defined by the bias-corrected Monte Carlo operator (7) is stochastic, the final value of the ANN parameter $\theta$ may differ differ between two full training runs. To account for this uncertainty, I train $K$ different ANNs and report average values.

Key results are presented in Figure 1. The left panels show the standard deviation of the loss function defined in equation (7) for several choices of $M$ and $N$, holding the hyperparameter $T$ constant ($T = 100$). In the low-uncertainty parametrization ($\sigma_\nu = 0.5$), the top-left panel shows that the standard deviation of the loss function is much smaller for $M = 100$ and $N = 2$ than for $M = 2$ and $N = 100$. This lower standard deviation for $M = 100$ results in more efficient training of the ANN, as illustrated in the top-middle panel. With $M = 100$, the final value of the loss function is on average 30% smaller and the dispersion of the loss function is approximately one order of magnitude smaller. The top-right panel shows that differences in loss functions translate directly into differences in the accuracy of the estimated policy function: the policy function $c(y|\theta)$ estimated is much more accurate with $M = 100$ than with $M = 2$.

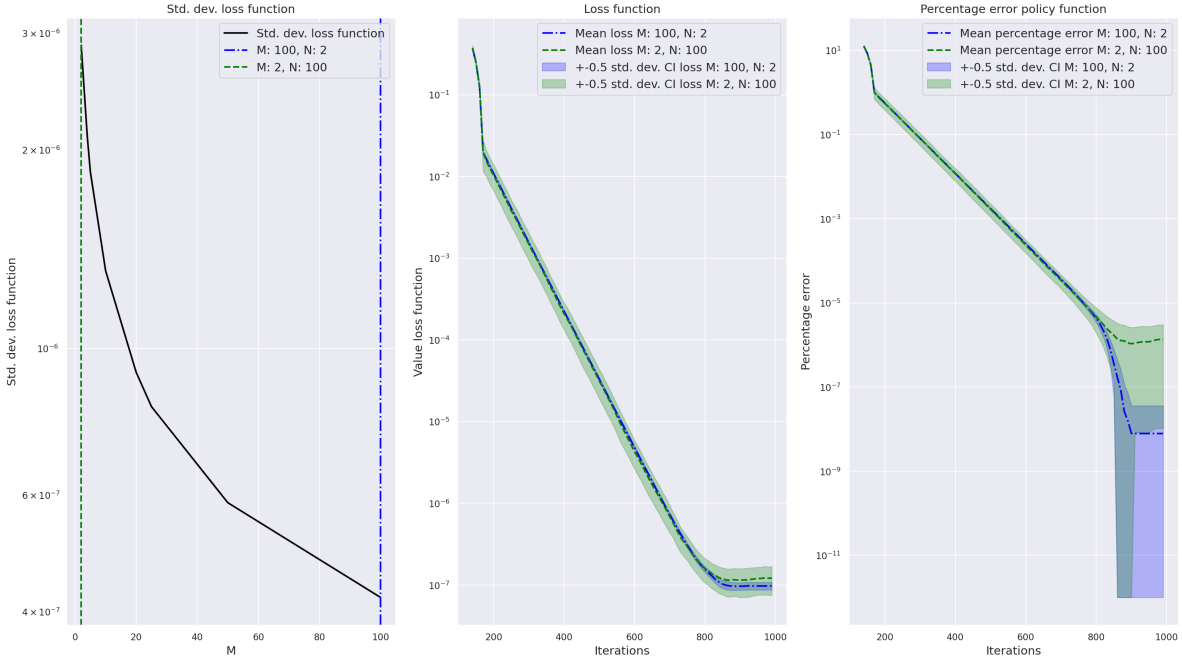
In the high-uncertainty parametrization ($\sigma_\nu = 1.5$), the situation is reversed. The bottom-left panel shows that the standard deviation of the loss function is smaller for $M = 2$ and $N = 100$ than for $M = 100$ and $N = 2$. The bottom-middle panel indicates that the lower standard deviation for $M = 2$ results in more efficient training of the ANN. The final value of the loss function is almost two orders of magnitudes smaller with $M = 2$

[15] Our choice of architecture implies that $c(y|\theta) = S(\theta_0 + \theta_1 y)y$, with $S$ the sigmoid function. Training the ANN consists in finding $\theta_0$ and $\theta_1$ such that $S(\theta_0 + \theta_1 y) = (1 - \alpha\beta)$. One solution is $\theta_1 = 0$ and $\theta_0 = \log(\frac{1-\alpha\beta}{\alpha\beta})$.

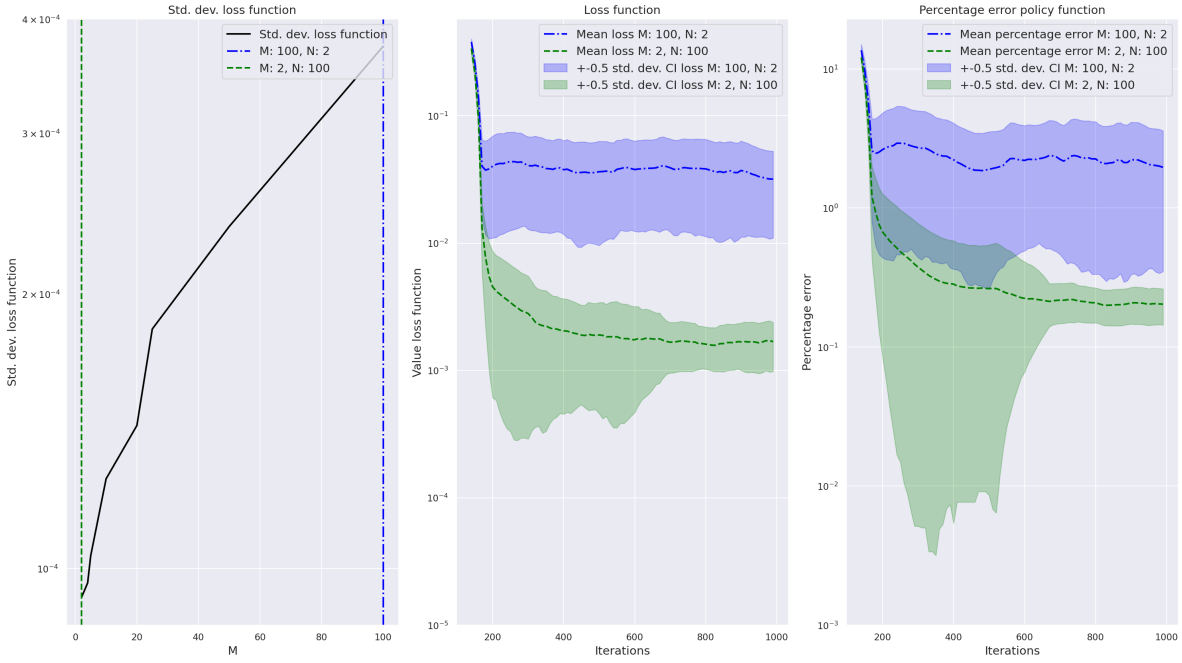
than with $M = 100$. The bottom-right panel indicates that the policy function $c(y|\theta)$ is one order of magnitude more accurate with $M = 2$ than with $M = 100$.

Results presented in this section confirm that choosing $M$ and $N$ in order to minimize the mean squared integration error of $\mathcal{L}^M_{M,N}(\theta)$, which is equal to the variance of the loss function, results in more efficient training of ANNs via the stochastic gradient descent algorithm. They also illustrate the role that uncertainty has on the optimal choice of the hyperparameters $M$ and $N$. I find that the higher the uncertainty created by the shocks $\varepsilon$, the larger the optimal value for $N$, as already suggested by proposition 4.

FIGURE 1. Stochastic neoclassical growth model solved with the bc-MC operator




(A) Low-uncertainty parametrization ($\sigma_\nu = 0.5$)




(B) High-uncertainty parametrization ($\sigma_\nu = 1.5$)

*Notes.* The left panels show the standard deviation of the loss function defined in equation (7) for several choices of $M$ and $N$, holding the hyperparameter $T$ constant ($T = 100$). The middle panels plot the loss function against the number of gradient descent steps. The right panels display the percentage error of the policy function, defined as $100|\frac{c(y|\theta)-c^*(y)}{c^*(y)}|$, with $c^*(y) = (1 - \alpha\beta)y$. For the middle and right panels, the solid line represents the average value across $K = 50$ independently trained ANNs. The shaded areas are confidence intervals using 0.5 standard deviations. The blue line is for $M = 100$ and $N = 2$, while the red line is for $M = 2$ and $N = 100$.

4.2. **Optimal consumption with a borrowing constraint.** Consider an agent maximizing her inter-temporal discounted utility

$$\max_{\{c_t\}_{t=0}^{\infty}} \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t u(c_t)\exp(\delta_t)\right] \tag{15}$$

subject to the constraints $0 \leq c_t \leq w_t + b$, $w_{t+1} = (w_t - c_t)\bar{r}\exp(r_{t+1}) + \exp(y_{t+1})$ and with $w_0$ given. The variable $c_t$ is consumption, $w_t$ is cash on hand at the beginning of the period, $\beta \in (0,1)$ is the subjective discount factor, $\bar{r} \in (0, \frac{1}{\beta})$ is the gross constant interest rate. Total income $y_t$ is assumed to be stochastic, with $y_t = \exp(p_t + q_t)$. The four exogenous state variables are assumed to follow AR(1) processes:

$$\begin{aligned} p_{t+1} &= \rho_p p_t + \sigma_p \varepsilon^p_{t+1} \\ q_{t+1} &= \rho_q q_t + \sigma_q \varepsilon^q_{t+1} \\ r_{t+1} &= \rho_r r_t + \sigma_r \varepsilon^r_{t+1} \\ \delta_{t+1} &= \rho_\delta \delta_t + \sigma_\delta \varepsilon^\delta_{t+1} \end{aligned} \tag{16}$$

where $\varepsilon^i$ are i.i.d. standard Normal random variables. I set the value of the variance parameters $\sigma_i$ to 0.01 and the persistence parameters $\rho_i$ to 0.9. I use a CRRA utility function $u(c) = \frac{c^{1-\gamma}}{1-\gamma}$, with $\gamma = 2$. I use $\beta = 0.9$ and $\bar{r} = 1.04$. The parameter $b$ is the borrowing constraint. When $b = 0$, the agent cannot borrow; when $b > 0$ the agent can borrow up to $b$.[16]

The solution to this dynamic programming problem is characterized by the following Kuhn-Tucker conditions:

$$\begin{aligned} A \geq 0, H \geq 0, AH = 0 \\ A \equiv w + b - c \\ H \equiv u'(c) - \beta\bar{r}\,\mathbb{E}_\varepsilon\left[u'(c')\exp(\delta' + \delta - r')\right] \end{aligned} \tag{17}$$

When the borrowing constraint is not binding ($A > 0$), the usual Euler condition applies. When the borrowing constraint is binding ($H > 0$), the agent consumes all her

[16] Maliar, Maliar, and Winant (2021) solve this consumption-saving problem model with their all-in-one expectation operator with $b = 0$.

cash on hand $w$ and borrows up to the limit $b$. For better performance, I use rescaled versions of $A$ and $H$: $a \equiv 1 - \frac{c-b}{w}$, $h \equiv 1 - \beta\bar{r}\,\mathbb{E}_\varepsilon\left[\frac{u'(c')}{u'(c)}\exp(\delta' + \delta - r')\right]$.

Following Maliar, Maliar, and Winant (2021), I note that the Kuhn-Tucker conditions can be smoothly approximated, making the problem differentiable. I use the Fischer-Burmeister (FB) function, defined by $\Psi(x, y) \equiv x + y - \sqrt{x^2 + y^2}$. The FB function is such that when it is equal to 0, the Kuhn-Tucker conditions are satisfied: $\Psi(x, y) = 0 \Leftrightarrow x \geq 0, y \geq 0, xy = 0$ (see for instance Chen, Chen, and Kanzow, 2000). As in Maliar, Maliar, and Winant (2021), I note that the model can be solved by minimizing a loss function of the form:

$$\begin{aligned}\mathcal{L}(\theta) = \mu_1\,\mathbb{E}_s\left[\Big(\mathbb{E}_\varepsilon\left[\beta\bar{r}\frac{u'(c(s'|\theta))}{u'(c(s|\theta))}\exp(\delta' + \delta - r') - h(s|\theta)\right]\Big)^2\right] \\ + \mu_2\Big(\mathbb{E}_s\left[\Psi\Big(1 - \frac{c(s|\theta) - b}{w}, 1 - h(s|\theta)\Big)^2\right]\Big)\end{aligned} \tag{18}$$

where $c(s|\theta)$ and $h(s|\theta)$ are the output of an ANN parametrized by $\theta$, while $\mu_1$ and $\mu_2$ are subjective weights. The state is a 5-dimensional vector $s = \big(w, p, q, r, \delta\big)$ and the shock is a 4-dimensional vector $\varepsilon = \big(\varepsilon^p, \varepsilon^q, \varepsilon^r, \varepsilon^\delta\big)$.

The loss function (18) is a slight generalization of equation (2), because it has two elements. However, I can still use the key idea of section 3. Namely, I use the MVUE estimator for the first term involving nested expectations, while I use a simple sample average for the second term:

$$\begin{aligned}\mathcal{L}^U_{M,N}(\theta) = \mu_1\frac{2}{MN(N-1)}\sum_{m=1}^{M}\sum_{1\leq i<j}^{N} f(s_m, \epsilon^i_m|\theta)f(s_m, \epsilon^j_m|\theta) \\ + \mu_2\frac{1}{M}\sum_{m=1}^{M}\Psi\Big(1 - \frac{c(s_m|\theta) - b}{w_m}, 1 - h(s_m|\theta)\Big)^2\end{aligned} \tag{19}$$

with $f(s, \epsilon|\theta) \equiv \beta\bar{r}\frac{u'(c(s'|\theta))}{u'(c(s|\theta))}\exp(\delta' + \delta - r') - h(s|\theta)$.

For the architecture of the ANN, I use a single hidden layers of 32 nodes with ReLU activation functions. The ANN outputs the share consumed out of cash on hand plus

borrowing $\phi \equiv \frac{c(s|\theta)}{w+b}$ and the value $h$. I use the logistic function to enforce the budget constraint ($\phi \in [0,1]$). I also use an exponential function to ensure that $h > 0$.

To measure accuracy, I use the following unit-less metric, denoted by $E_c$, which is equal to 0 at the true solution $c(s|\theta) = c^*(s)$:

$$E_c \equiv \left| 1 - \frac{1}{c(s|\theta)} (u')^{-1} \Big[ \max \Big( \beta \overline{r} E_\epsilon \left[ u' \left( c(s'|\theta) \right) \exp \left( \delta' - \delta + r' \right), u'(w+b) \Big) \right] \right| \quad (20)$$

This accuracy metric is similar to the one discussed by Judd and Guu (1997) and Barillas and Fernández-Villaverde (2007). It accounts for the fact that when the borrowing constraint is binding, the agent consumes all her available cash on hand $w$ and borrows up to the limit $b$. When the borrowing constraint is not binding, consumption obeys the usual Euler equation defined by $H$ in equation (17). When $E_c$ is equal to 0.01, on average agents make a 1 dollar mistake for every dollar they invest.

As in section 4.1, I solve the model for two different parametrizations, to analyze how the optimal choice of $M$ and $N$ is influenced by the underlying characteristics of the model. In the first parametrization, I set $b = 0$, in which case the borrowing constraint is binding. In the second parametrization, I set $b = 1$, in which case the borrowing constraint does not bind, as explained below.

The consumption rules implied by two parametrizations are visible in Figure 2. For the bc-MC methodology, I train the ANN for 10000 gradient descent steps (left panel) by minimizing loss function (19) with the ADAM optimization algorithm using a learning rate $\gamma = 1.10^{-3}$. As a robustness check, I also solve the model using the Time Iteration algorithm (right panel). A comparison of the right and left panels of Figure 2 indicates that both methodologies yield similar consumption functions, with minimal differences at around the kink point ($w \approx 1$) when $b = 0$. Figure 2 also shows that when $b = 0$ the budget constraint is binding, which is not the case when $b = 1$.

Key results are presented in Figure 3. In the binding parametrization ($b = 0$), the variance of the loss function is minimized for the choice $N = 2$ and $M = 100$ (top-left panel). This optimal choice of $N$ and $M$ leads to more efficient training (middle-top panel), which results in a more accurate estimate of the policy function $c(s|\theta)$ (top-right panel). Compared to a sub-optimal choice ($N = 100$, $M = 2$), the optimal choice reduces

the loss function by approximately 60% and the accuracy metric $E_c$ by approximately 55%. In the non-binding parametrization ($b = 1$), the choice $N = 2$ and $M = 100$ is no longer optimal. The variance of the loss function is minimized for $N = 20$ and $M = 10$ (bottom-left panel). This choice of hyperparameters leads to more efficient training (bottom-middle panel) and a more accurate estimate policy function (bottom-right panel).

Comparing the binding and non-binding parametrizations indicates that the optimal choice of hyperparameters $M$ and $N$ is not only linked to the amount of randomness in the model (section 4.1), but also to the curvature of the decision rule in the state space. When $b = 0$, there exists a kink around $w \approx 0$. To correctly capture the kink in the consumption rule, a large number of draws in the state space is required (a large value for $M$). For a given total number of function calls for each evaluation of the loss function (captured by $T$), this requires minimizing the number of independent shocks used in the loss function (19) (a small value for $N$).

4.3. **Large Scale Application.** In this section, I compare the bc-MC operator to more traditional techniques, especially when the dimension of the problem gets large. With this objective in mind, I slightly modify the model from the previous section in order to easily increase or decrease the dimension of the problem. As in the previous section, let us consider an agent maximizing her inter-temporal discounted utility

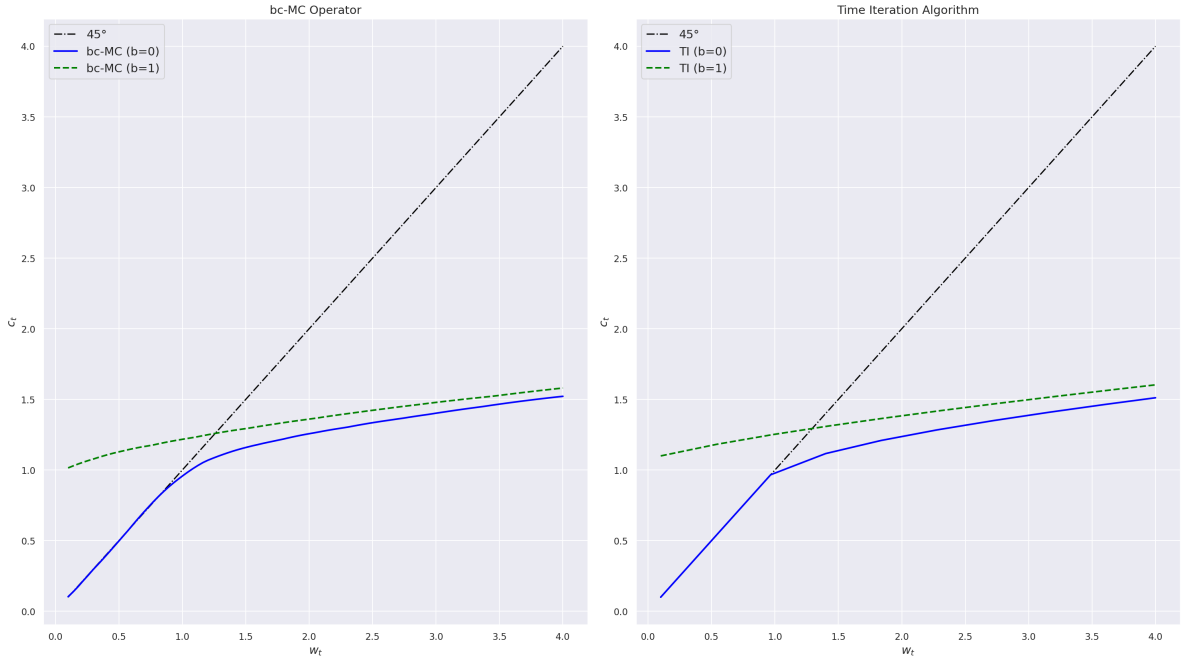
$$\max_{\{c_t\}_{t=0}^{\infty}} \mathbb{E}\Big[\sum_{t=0}^{\infty} \beta^t u(c_t) \exp(\delta_t)\Big]$$

subject to the same constraints as in section 4.2. In this section, I set the borrowing constraint $b = 0$, which creates a kink in the decision rule. Total income $y_t$ is assumed to be stochastic, with $y_t = \exp(\sum_{i=1}^{l} p_{i,t})$. The model has $2 + l$ exogenous variables, which follow AR(1) processes:

$$\begin{aligned} p_{i,t+1} &= \rho_{i,p} p_{i,t} + \sigma_{i,p} \varepsilon^p_{i,t+1}, \quad \forall i \in 1, 2, ..., l \\ r_{t+1} &= \rho_r r_t + \sigma_r \varepsilon^r_{t+1} \\ \delta_{t+1} &= \rho_\delta \delta_t + \sigma_\delta \varepsilon^\delta_{t+1} \end{aligned} \tag{21}$$

The dimension of the model depends on the number of variables appearing in stochastic income $y_t$. More specifically, the state vector $s = \big(w, r, \delta, p_1, ..., p_l\big)$ has $d_s \equiv 3 + l$

FIGURE 2. Consumption rule with borrowing constraint $b = 0$ or $b = 1.0$

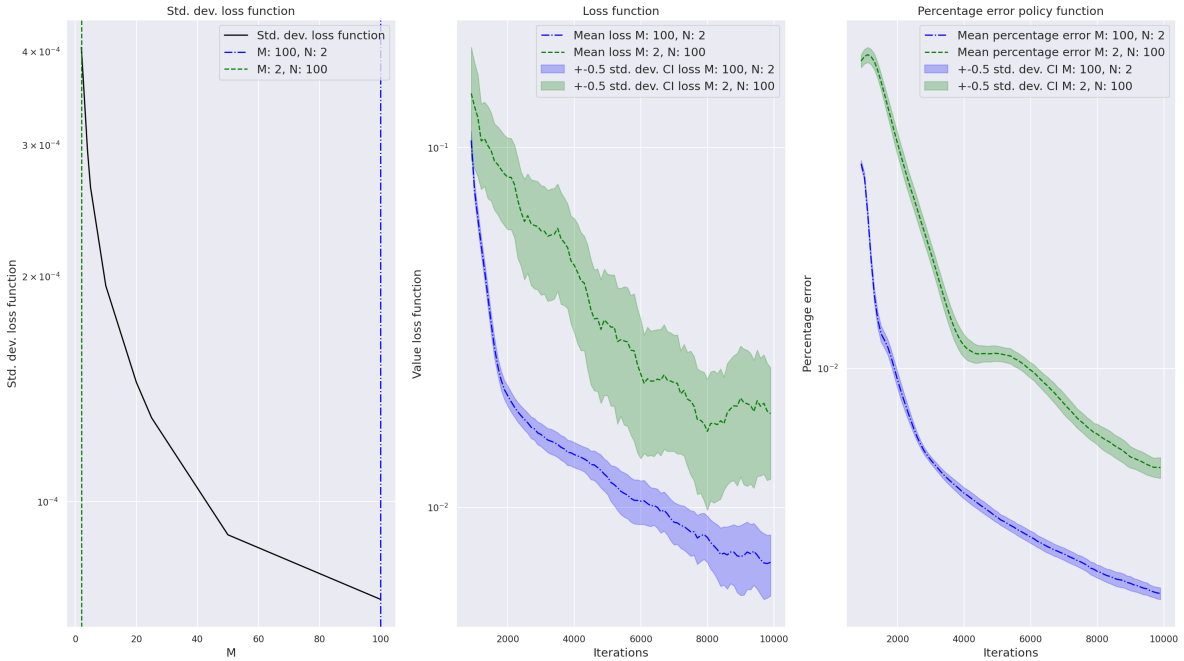



*Notes.* This Figure shows the consumption rules when $b = 0$ (blue solid line) and $b = 1$ (green dotted line). The left panel uses the bc-MC operator with $N = 2$ and $M = 100$. I train an ANN for 10000 gradient descent steps by minimizing the loss function (19) with the ADAM optimization algorithm using a learning rate $\gamma = 1.10^{-3}$. The right panel uses the Time Iteration (TI) algorithm. For the TI algorithm, I use a dense grid with equally spaced points. I use 10 points for $w$, and 3 for the other 4 state variables. To approximate expectations, I use a sparse Gaussian quadrature of order 1, which leads to 9 evaluations of $\varepsilon$ for each value of $s$.

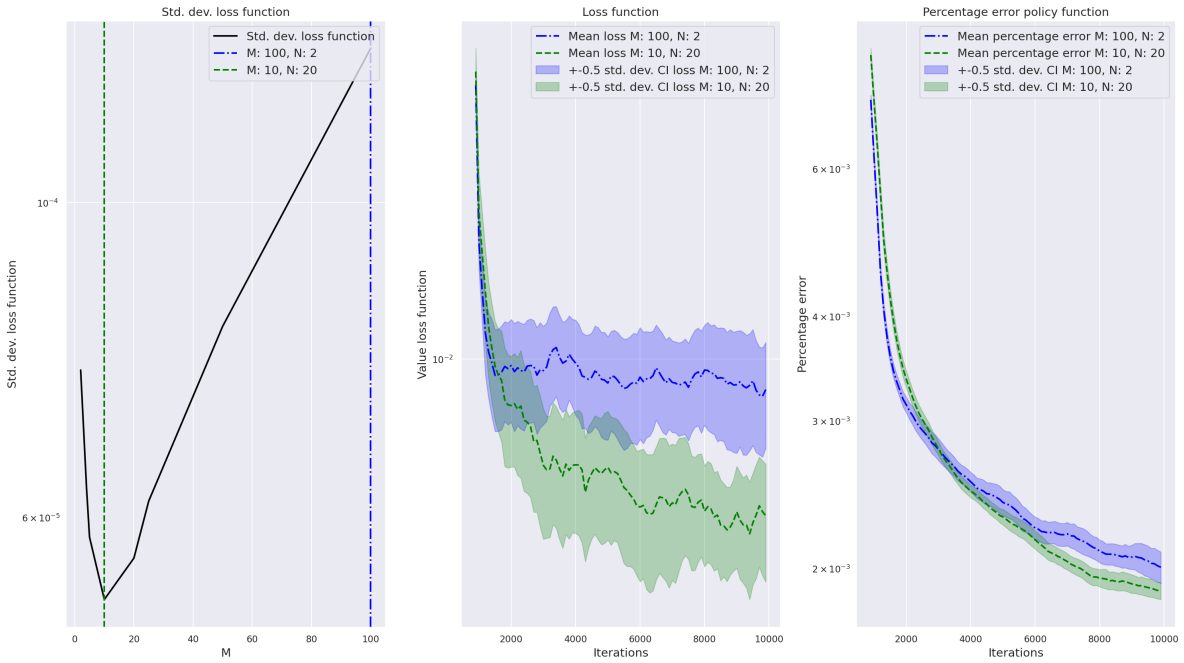
elements and the shock vector $\varepsilon = \left(\varepsilon^r, \varepsilon^\delta, \varepsilon_1^p, ..., \varepsilon_l^p\right)$ has $d_\varepsilon \equiv 2 + l$ elements. I use the same parameter values as in the previous section. For the $l$ values of $p_i$, I use $\rho_{i,p} = 0.9$ and a standard deviation $\sigma_{\varepsilon_i^p} = 0.01$.

I compare the performance of the bc-MC operator to the Time Iteration (TI) algorithm. For the TI algorithm, I use a dense grid with equally spaced points. I use 10 points for $w$, and 3 for the other $2 + l$ state variables. Hence, the number of grid points is equal to $10 \times 2^{2+l}$, which grows exponentially with $l$. I use linear interpolation to approximate values between grid points. To approximate expectations, I use a sparse

FIGURE 3. Model with a borrowing constraint solved with the bc-MC operator




(A) $b$ binding ($b = 0$)




(B) $b$ non-binding ($b = 1$)

*Notes.* The left panels show the standard deviation of the loss function defined in equation (7) for several choices of $M$ and $N$, holding hyperparameter $T$ constant ($T = 100$). The middle panels plot the loss function against the number of gradient descent steps. The right panels display the accuracy metrics $E_c$, defined in equation (20). For the middle and right panels, the solid line represents the average value across the $K = 10$ independently trained ANNs. The shaded areas are confidence intervals using 0.5 standard deviations.

Gaussian quadrature of order 1.[17] I stop the TI algorithm when the distance between the new guess for $c(s)$ and the previous guess is less than $10^{-5}$.[18]

For the bc-MC operator, I use $N = 2$ and $M = 100$, which is the optimal choice when $b = 0$ (see section 4.2). I use the same ANN architecture as in section 4.2 (a single hidden layers of 32 nodes with ReLU activation functions). I also use the same optimization algorithm and training procedure (10000 gradient descent steps with the ADAM optimization algorithm using a learning rate $\gamma = 1.10^{-3}$ to minimize the loss function (19)).

Key results regarding the time-accuracy trade-off for the bc-MC operator are presented in Figure 4. The left panel shows that the TI algorithm is much faster than the bc-MC operator for low-dimensional models. However, the time required to solve the model with TI using a dense grid grows exponentially with the dimension of the state vector denoted $d_s$. When $d_s$ is equal to 11, it takes more than 7 hours for the TI algorithm to converge. The bc-MC operator is slower for low dimensional models, but the computing time increases very slowly with $d_s$. A crossing occurs at $d_s = 8$, a value for which the bc-MC operator is faster than the TI algorithm. This can be easily explained since the number of grid points for the TI algorithm grows exponentially with $d_s$, while the number of draws for the bc-MC operator remains constant ($M \times N$). However, computing time for the bc-MC operator does increase gradually because the dimension of each draw also increases ($M$ draws of a vector of size $d_s$ and $N$ draws of a vector of size $d_\varepsilon$).

The middle panel of Figure 4 displays the Euler equation error $E_c$. For $d_s$ between 4 and 11, the TI algorithm is more accurate than the bc-MC operator. However, for both approaches the Euler equation error $E_c$ increases with $d_s$. This reflects the fact that as the dimension of the state variable increases the exact location of the kink point visible in Figure 2 becomes harder to locate.[19] We observe that the rate of increase in $E_c$ is higher for the TI algorithm than for the bc-MC operator, which indicates better scaling abilities for the latter. The right panel of Figure 4 shows the expected value of the Euler

[17] I use a Smolyak sparse-grid construction (see Judd et al., 2014). To generate the sparse Gaussian quadrature nodes and weights, I use the Python library Chaospy (see Feinberg and Langtangen, 2015).

[18] I compare the $10 \times 2^{2+l}$ grid points using the L-$\infty$ distance.

[19] The volume of the d-dimensional sphere with radius 1 goes to 0 as the dimension d increases to infinity.

equation error $E_c$ for a similar run time (100 seconds). A crossing occurs at $d_s = 8$, at which point the bc-MC operator performs better on a time-accuracy basis.

I also solve the model for $l$ up to 100, which raises the dimension of state vector $s = (w, r, \delta, p_1, ..., p_l)$ to $d_s = 103$. At this scale, the TI algorithm on a dense grid is untractable. To have a sense on how the TI algorithm would perform, I estimate the elapsed computing time and the Euler error as a function of $d_s$ using data for $d_s \in 4, 5, ..., 11$.[20] The left panel of Figure 5 indicates that it takes approximately 4 minutes to solve the model with $d_s = 103$ (right axis), while the TI algorithm on a dense grid would require an unacceptable amount of computing time (left axis). The right panel of Figure 5 shows that even with $d_s = 103$, the bc-MC operator remains reasonably accurate, with $E_c$ only slightly above 0.01.

The bc-MC operator remains accurate, although the architecture of the underlying ANN is unchanged as I scale up the dimensionality of the model (a single hidden layers of 32 nodes). One could have expected that increasing the complexity of the ANN architecture would become necessary by increasing the number of nodes or the number of hidden layers. To explain this success, note that using constant values for $\rho_{i,p} = \rho_{1,p} = 0.9$ and $\sigma_{\varepsilon_i^p} = \sigma_{\varepsilon_1^p} = 0.01$, the model can be solved with a 4-dimensional state vector $s^\star = (w, r, \delta, P)$ and a 3-dimensional shock vector: $\varepsilon = \left(\varepsilon^r, \varepsilon^\delta, \zeta\right)$

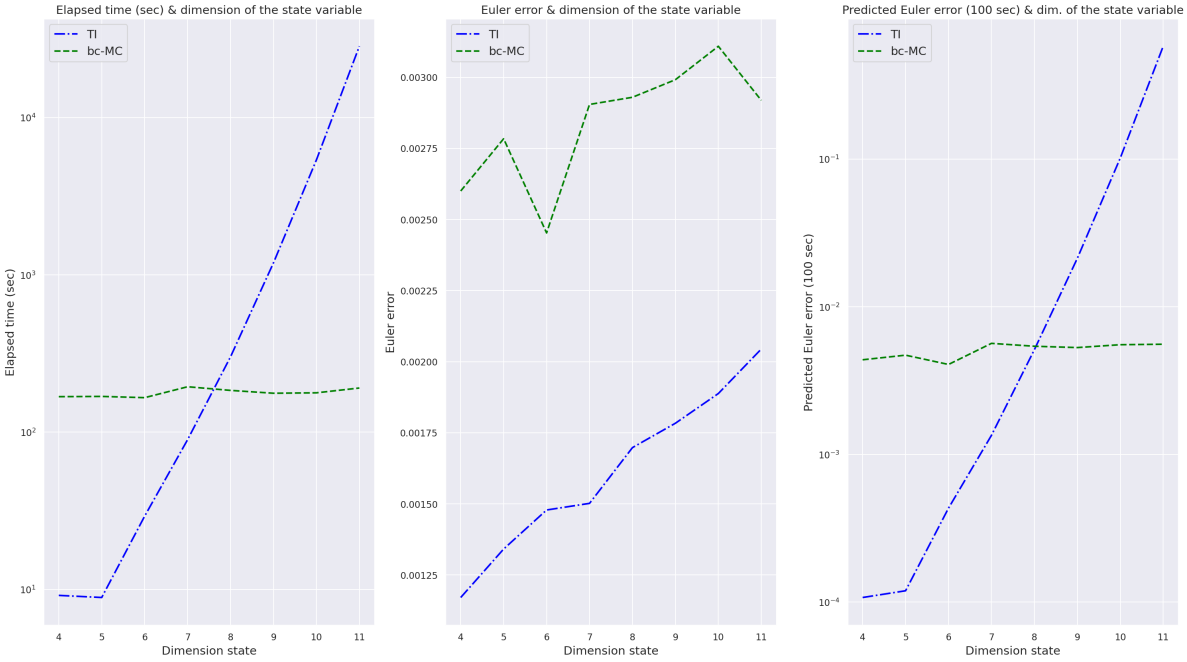
$$\underbrace{\sum_{i=1}^{l} p_{i,t+1}}_{P_{t+1}} = \rho_{1,p} \underbrace{\sum_{i=1}^{l} p_{i,t}}_{P_t} + \sigma_{\varepsilon_1^p} \underbrace{\sum_{i=1}^{l} \varepsilon_{i,t+1}^p}_{\zeta_{t+1}} \tag{22}$$

where $\zeta$ is a zero-mean Normal random variable of variance $l$. The good performance of the ANN is likely explained by its ability to automatically detect this dimension reduction. Indeed, it has been shown that ANNs outperform principal component analysis in dimension reduction tasks (Hinton and Salakhutdinov, 2006).[21]

[20] I use a log-linear model for the regression involving elapsed computing time, and a linear-linear model for the regression involving $E_c$.

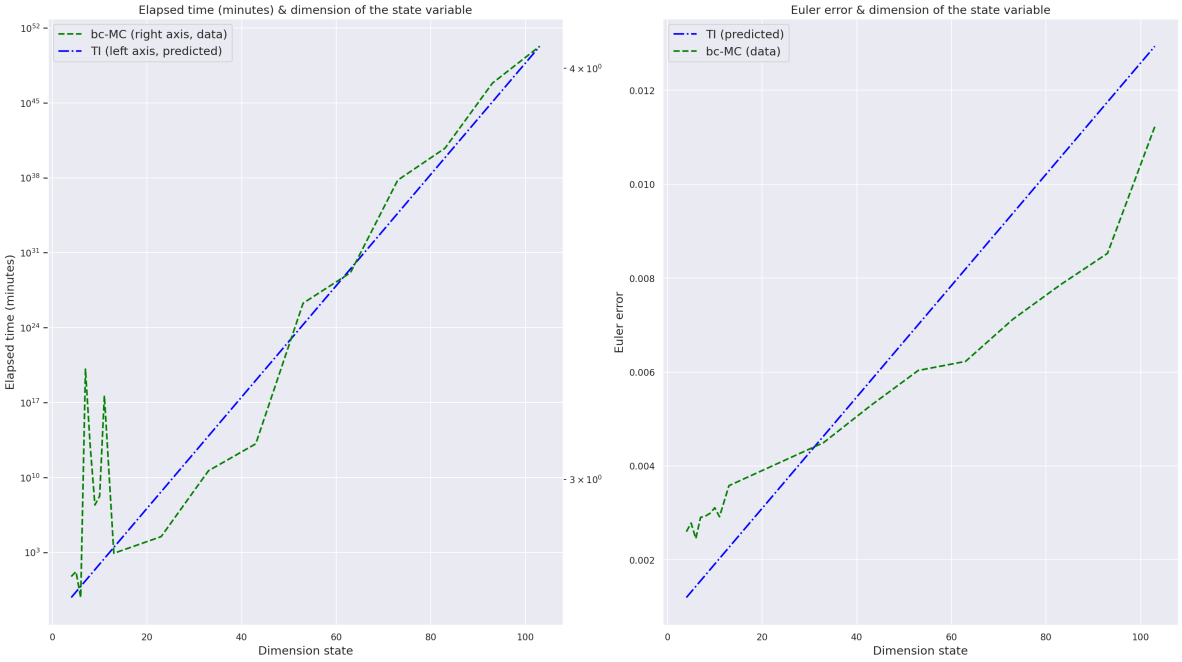
[21] See Maliar, Maliar, and Winant (2021) for a similar discussion on how ANNs are able to automatically reduce the dimension of the state variable in a model with heterogeneous agents.

FIGURE 4. bc-MC operator and Time Iteration: time-accuracy trade-off




*Notes.* The left panel shows the computing time (in seconds) required to solve the model with a borrowing constraint ($b = 0$) as a function of the size of the state vector using the bc-MC operator (green dotted line) or the Time Iteration (TI) algorithm (blue solid line). The middle panel displays the Euler equation error $E_c$ calculated using (20), with 100 Monte Carlo draws to approximate expectations for each of 5000 random draws of the state vector. The right panel shows the predicted Euler equation error $E_c$ reached after 100 seconds of computing time (calculated as $\frac{E_c \times \text{time}}{100}$). I train an ANN for 10000 gradient descent steps by minimizing loss function (19) with the ADAM optimization algorithm using a learning rate $\gamma = 1.10^{-3}$. For the TI algorithm, I use a dense grid with equally spaced points. I use 10 points for $w$, and 3 for the other $2 + l$ state variables. To approximate expectations, I use a sparse Gaussian quadrature of order 1.

FIGURE 5. bc-MC operator and Time Iteration with large scale models




*Notes.* The left panel plots the computing time (in minutes) required to solve the model with a borrowing constraint ($b = 0$) against the dimension of the state vector, using the bc-MC operator (green dotted line) or the Time Iteration (TI) algorithm (blue solid line). The right panel displays the Euler equation error $E_c$ calculated using (20), with 100 Monte Carlo draws to approximate expectations for each of 5000 random draws of the state vector. Because it would be untractable otherwise, for the TI algorithm I use the predicted values from a linear regression using the data points appearing in Figure 4 (a log-linear regression for elapsed computing time, and a linear-linear regression for $E_c$).

## 5. Conclusion

In this paper, I develop a general methodology to find global solutions to economic models using Artificial Neural Networks (ANNs). This method does not involve linearization around a steady-state and can scale to high-dimensional models, such as overlapping generations models (Krueger and Kubler, 2006), models with partially insurable risk for consumers (Krusell and Smith, 1998), New Keynesian models with heterogeneous households (Kaplan, Moll, and Violante, 2018), models with non-convex capital adjustment costs for firms (Khan and Thomas, 2008), or multi-country models (Backus, Kehoe, and Kydland, 1992).

I obtain tractability by using Monte Carlo integration techniques to approximate expectations, in combination with ANNs to approximate decision functions. I transform two nested expectations into a single expectation using what I call a bias-corrected Monte Carlo operator (bc-MC), which generalizes the all-in-one operator of Maliar, Maliar, and Winant (2021). I study how to optimally set the hyperparameters defining the bc-MC operator, considering that ANNs are trained by stochastic gradient descent or its variants. I illustrate the theoretical properties of the bc-MC operator numerically by solving well-known economic models. I also show that the bc-MC operator can scale to high-dimensional models, which is not the case for other popular global algorithms relying on dense grids to explore the the state space, such as the Time Iteration algorithm.

In this paper, I rely on Monte Carlo integration to approximate expectations. However, one possible alternative would be to use sparse quadrature nodes. Using fixed grid points as integration nodes reduces the variance of the loss function, which leads to faster training with the stochastic gradient descent algorithm. I also note that there might be promising ways to combine methods using adaptive sparse grids (see Brumm and Scheidegger, 2017) and machine learning techniques. Using an adaptive sparse grid to guide state-space sampling near kinks in the decision functions might further reduce the variance of the loss function, in the spirit of the importance sampling algorithm (Katharopoulos and Fleuret, 2018). I leave these avenues for further research.

## References


Aiyagari, S Rao (1994). "Uninsured idiosyncratic risk and aggregate saving". In: *The Quarterly Journal of Economics* 109.3, pp. 659–684.

Azinovic, Marlon, Luca Gaegauf, and Simon Scheidegger (2022). "Deep equilibrium nets". In: *International Economic Review*.

Backus, David K, Patrick J Kehoe, and Finn E Kydland (1992). "International real business cycles". In: *Journal of political Economy* 100.4, pp. 745–775.

Barillas, Francisco and Jesús Fernández-Villaverde (2007). "A generalization of the endogenous grid method". In: *Journal of Economic Dynamics and Control* 31.8, pp. 2698–2712.

Barron, Andrew R (1993). "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3, pp. 930–945.

Bellman, Richard (1961). "Adaptive control processes: a guided tour". In: *Princeton, New Jersey, USA*.

Blackwell, David (1965). "Discounted dynamic programming". In: *The Annals of Mathematical Statistics* 36.1, pp. 226–235.

Brock, William A and Leonard J Mirman (1972). "Optimal economic growth and uncertainty: the discounted case". In: *Journal of Economic Theory* 4.3, pp. 479–513.

Brumm, Johannes and Simon Scheidegger (2017). "Using adaptive sparse grids to solve high-dimensional dynamic models". In: *Econometrica* 85.5, pp. 1575–1612.

Chen, Bintong, Xiaojun Chen, and Christian Kanzow (2000). "A penalized Fischer-Burmeister NCP-function". In: *Mathematical Programming* 88.1, pp. 211–216.

Choromanska, Anna et al. (2015). "The loss surfaces of multilayer networks". In: *Artificial intelligence and statistics*. PMLR, pp. 192–204.

Coeurdacier, Nicolas, Helene Rey, and Pablo Winant (2011). "The risky steady state". In: *American Economic Review* 101.3, pp. 398–401.

Das, Biswanath (1975). "Estimation Of $\mu 2$ in Normal Population". In: *Calcutta Statistical Association Bulletin* 24.1-4, pp. 135–140.

Devereux, Michael B and Alan Sutherland (2011). "Country portfolios in open economy macro-models". In: *Journal of the european economic Association* 9.2, pp. 337–369.

Feinberg, Jonathan and Hans Petter Langtangen (2015). "Chaospy: An open source tool for designing methods of uncertainty quantification". In: *Journal of Computational Science* 11, pp. 46–57.

Fernández-Villaverde, Jesús, Samuel Hurtado, and Galo Nuno (2019). *Financial frictions and the wealth distribution*. Tech. rep. National Bureau of Economic Research.

Fernández-Villaverde, Jesús, Juan Francisco Rubio-Ramírez, and Frank Schorfheide (2016). "Solution and estimation methods for DSGE models". In: *Handbook of macroeconomics*. Vol. 2. Elsevier, pp. 527–724.

Goda, Takashi (2017). "Computing the variance of a conditional expectation via nonnested Monte Carlo". In: *Operations Research Letters* 45.1, pp. 63–67. ISSN: 0167-6377. DOI: https://doi.org/10.1016/j.orl.2016.12.002. URL: https://www.sciencedirect.com/science/article/pii/S0167637716302450.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT press.

Hinton, Geoffrey E and Ruslan R Salakhutdinov (2006). "Reducing the dimensionality of data with neural networks". In: *science* 313.5786, pp. 504–507.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5, pp. 359–366.

Judd, Kenneth L and Sy-Ming Guu (1997). "Asymptotic methods for aggregate growth models". In: *Journal of Economic Dynamics and Control* 21.6, pp. 1025–1042.

Judd, Kenneth L et al. (2014). "Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain". In: *Journal of Economic Dynamics and Control* 44, pp. 92–123.

Kaplan, Greg, Benjamin Moll, and Giovanni L Violante (2018). "Monetary policy according to HANK". In: *American Economic Review* 108.3, pp. 697–743.

Katharopoulos, Angelos and François Fleuret (2018). "Not all samples are created equal: Deep learning with importance sampling". In: *International conference on machine learning*. PMLR, pp. 2525–2534.

Khan, Aubhik and Julia K Thomas (2008). "Idiosyncratic shocks and the role of nonconvexities in plant and aggregate investment dynamics". In: *Econometrica* 76.2, pp. 395–436.

Krueger, Dirk and Felix Kubler (2004). "Computing equilibrium in OLG models with stochastic production". In: *Journal of Economic Dynamics and Control* 28.7, pp. 1411–1436.

– (2006). "Pareto-Improving Social Security Reform when Financial Markets Are Incomplete!?" In: *The American Economic Review* 96.3, pp. 737–755. ISSN: 00028282. URL: http://www.jstor.org/stable/30034069 (visited on 01/30/2023).

Krusell, Per and Anthony A Smith Jr (1998). "Income and wealth heterogeneity in the macroeconomy". In: *Journal of political Economy* 106.5, pp. 867–896.

Ljungqvist, Lars and Thomas J Sargent (2018). *Recursive macroeconomic theory*. MIT press.

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

Rainforth, Tom et al. (2018). "On nesting monte carlo estimators". In: *International Conference on Machine Learning*. PMLR, pp. 4267–4276.

Raissi, Maziar and George Em Karniadakis (2018). "Hidden physics models: Machine learning of nonlinear partial differential equations". In: *Journal of Computational Physics* 357, pp. 125–141.

Raissi, Maziar, Paris Perdikaris, and George E Karniadakis (2019). "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational physics* 378, pp. 686–707.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Scheidegger, Simon and Ilias Bilionis (2019). "Machine learning for high-dimensional dynamic stochastic economies". In: *Journal of Computational Science* 33, pp. 68–82.

Schmidhuber, Jürgen (2015). "Deep learning in neural networks: An overview". In: *Neural networks* 61, pp. 85–117.

Schmitt-Grohé, Stephanie and Martın Uribe (2003). "Closing small open economy models". In: *Journal of international Economics* 61.1, pp. 163–185.

Silver, David et al. (2016). "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587, pp. 484–489.

Smolyak, Sergei Abramovich (1963). "Quadrature and interpolation formulas for tensor products of certain classes of functions". In: *Doklady Akademii Nauk*. Vol. 148. 5. Russian Academy of Sciences, pp. 1042–1045.

Stokey, Nancy L (1989). *Recursive methods in economic dynamics*. Harvard University Press.

## Appendix A. Case with several equations

In general, an economic model is characterized by a system of $J$ stochastic functional equations:

$$\mathbb{E}_\varepsilon\Big(f_j(s,\epsilon)\Big)=0 \quad \text{for } s\in S \text{ and } j\in 1,...,J \tag{23}$$

where s is vector representing values known today and $\epsilon$ a vector of i.i.d. shock occurring after decisions are made. These $J$ equations can be Euler or Bellman equations. They can also be budget constraints, or other types of constraints introduced in a smooth and differentiable way. A parametric approximation of the solution $f(s,\epsilon|\theta)=(f_1(s,\epsilon|\theta),...,f_J(s,\epsilon|\theta))$ can be obtained by minimizing a loss function that takes into account a weighted sum of the $j$ equations:

$$\mathcal{L}(\theta)=\sum_{j=1}^{J}\vartheta_j\,\mathbb{E}_s\left[\mathbb{E}_\varepsilon\left(f_j(s,\epsilon|\theta)\right)^2\right] \tag{24}$$

where $\vartheta_j$ is a subjective weight associated to each equation characterizing the model. To approximate the expectations, one may use Monte Carlo integration separately for the two integrals:

$$\mathcal{L}^M_{M,N}(\theta)=\sum_{j=1}^{J}\vartheta_j\Big(\frac{1}{M}\sum_{m=1}^{M}\Big\{\Big[\frac{1}{N}\sum_{n=1}^{N}f_j(s_m,\epsilon_n|\theta)\Big]^2\Big\}\Big) \tag{25}$$

The bias-corrected Monte Carlo operator writes:

$$\mathcal{L}^U_{M,N}(\theta)=\sum_{j=1}^{J}\vartheta_j\Big(\frac{1}{M}\sum_{m=1}^{M}\Big\{\Big[\frac{1}{N}\sum_{n=1}^{N}f_j(s_m,\epsilon_n|\theta)\Big]^2-\frac{S^2_{j,m,n}}{N}\Big\}\Big) \tag{26}$$

## Appendix B. Proof of proposition 2

Let us consider a random variable $x$. By definition, the sample variance $S_n^2$ is given by $S_n^2\equiv\frac{1}{N-1}\sum_{i=1}^{N}(x_i-\bar{x})$, with $\bar{x}$ the sample mean. Expanding the square and rearranging yields the following alternative expression for the sample mean:

$$S_n^2=\frac{1}{N-1}\sum_{i=1}^{N}x_i^2-\frac{N}{N-1}\bar{x}^2 \tag{27}$$

Using equation (27), one gets a value for $\bar{x}^2-\frac{S_n^2}{N}$ involving the product $x_ix_j$:

$$\begin{aligned}
\bar{x}^2 - \frac{S_n^2}{N} &= \frac{N}{N-1}\bar{x}^2 - \frac{N}{N(N-1)}\bar{x}^2 \\
&= \frac{1}{N(N-1)}\Big[\Big(\sum_{i=1}^{N} x_i\Big)^2 - \sum_{i=1}^{N} x_i^2\Big] \\
&= \frac{1}{N(N-1)}\Big[\sum_{i=1}^{N} x_i^2 + 2\sum_{1\le i<j}^{N} x_i x_j - \sum_{i=1}^{N} x_i^2\Big] \\
&= \frac{2}{N(N-1)}\sum_{1\le i<j}^{N} x_i x_j \\
&= \frac{1}{\binom{N}{2}}\sum_{1\le i<j}^{N} x_i x_j
\end{aligned} \tag{28}$$

where $\binom{N}{2}$ is equal to the number of unique pairs that can be formed from $N$ elements. Using equation (28) within (6), and using the notations $x_i \equiv f(s_m, \epsilon_m^i)$ and $x_j \equiv f(s_m, \epsilon_m^j)$ yields:

$$\mathcal{L}_{M,N}^{U}(\theta) = \frac{2}{MN(N-1)}\sum_{m=1}^{M}\sum_{1\le i<j}^{N} f(s_m, \epsilon_m^i|\theta) f(s_m, \epsilon_m^j|\theta) \tag{29}$$

where $\epsilon^i$ and $\epsilon^j$ are i.i.d. shocks with the same distribution as $\epsilon$.

When $N = 2$, equation (29) simplifies to:

$$\mathcal{L}_{M,2}^{U}(\theta) = \frac{1}{M}\sum_{m=1}^{M} f(s_m, \epsilon_m^1|\theta) f(s_m, \epsilon_m^2|\theta) \tag{30}$$

## Appendix C. Proof of proposition 3

Let $e_{M,N}(f|\theta)$ denote the integration error, defined as:

$$\begin{aligned}
e_{M,N}(f|\theta) &\equiv \mathcal{L}(\theta) - \mathcal{L}_{M,N}^{U}(\theta) \\
&= \mathbb{E}_s\Big[\mathbb{E}_\varepsilon\Big(f(s,\epsilon|\theta)\Big)^2\Big] - \frac{1}{M}\sum_{m=1}^{M}\frac{2}{N(N-1)}\sum_{1\le i<j}^{N} f(s_m, \epsilon_m^i|\theta) f(s_m, \epsilon_m^j|\theta)
\end{aligned} \tag{31}$$

First, observe that $\mathcal{L}_{M,N}^{U}(\theta)$ is an unbiased estimator of $\mathcal{L}(\theta)$. By linearity of the expectation operator and because $\epsilon_m^i$ and $\epsilon_m^j$ are independent and have the same distribution, one gets:

$$\begin{aligned}
\mathbb{E}_\varepsilon\Big[\frac{2}{N(N-1)}\sum_{1\leq i<j}^{N} f(s_m,\epsilon_m^i|\theta)f(s_m,\epsilon_m^j|\theta)\Big] &= \frac{2}{N(N-1)}\sum_{1\leq i<j}^{N}\mathbb{E}_\varepsilon\Big[f(s_m,\epsilon_m^i|\theta)f(s_m,\epsilon_m^j|\theta)\Big] \\
&= \frac{2}{N(N-1)}\sum_{1\leq i<j}^{N}\mathbb{E}_{\varepsilon^i}\Big[f(s_m,\epsilon_m^i|\theta)\Big]\mathbb{E}_{\varepsilon^j}\Big[f(s_m,\epsilon_m^j|\theta)\Big] \\
&= \frac{2}{N(N-1)}\sum_{1\leq i<j}^{N}\mathbb{E}_\varepsilon\Big(f(s_m,\epsilon|\theta)\Big)^2 \\
&= \frac{2}{N(N-1)}\frac{N(N-1)}{2}\mathbb{E}_\varepsilon\Big(f(s_m,\epsilon|\theta)\Big)^2 \\
&= \mathbb{E}_\varepsilon\Big(f(s_m,\epsilon|\theta)\Big)^2
\end{aligned}$$

Again, by linearity of the expectation operator:

$$\mathbb{E}_s\Big[\frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_\varepsilon\Big(f(s_m,\epsilon|\theta)\Big)^2\Big] = \mathbb{E}_s\Big[\mathbb{E}_\varepsilon\Big(f(s,\epsilon|\theta)\Big)^2\Big]$$

To conclude the proof, I use the fact that the mean squared error of an estimator is equal to the sum of the square if its bias plus its variance. Because $\mathcal{L}^U_{M,N}(\theta)$ is an unbiased estimator of $\mathcal{L}(\theta)$, its mean squared error is equal to its the variance:

$$\mathbb{E}\Big[e_{M,N}(f|\theta)^2\Big] = \mathrm{Var}\Big[\mathcal{L}^U_{M,N}(\theta)\Big]$$

## Appendix D. Proof of proposition 4

D.1. **Proof of proposition 4.1.** I first prove the first point of proposition 4. I start with the expression of the loss function from proposition 2:

$$\mathcal{L}^U_{M,N}(\theta) = \frac{2}{MN(N-1)}\sum_{m=1}^{M}\sum_{1\leq i<j}^{N} f(s_m,\epsilon_m^i|\theta)f(s_m,\epsilon_m^j|\theta)$$

For the outer sum, the $m$ draws of the state variable $s$ are independent from each other. Hence, the variance of the sum if equal to the sum of variances:

$$\begin{aligned}
\mathrm{Var}\Big(\mathcal{L}^U_{M,N}(\theta)\Big) &= \Big(\frac{2}{MN(N-1)}\Big)^2\sum_{m=1}^{M}\mathrm{Var}\Big(\sum_{1\leq i<j}^{N} f(s_m,\epsilon_m^i|\theta)f(s_m,\epsilon_m^j|\theta)\Big) \\
&= \Big(\frac{2}{MN(N-1)}\Big)^2 M\,\mathrm{Var}\Big(\sum_{1\leq i<j}^{N} f(s_m,\epsilon_m^i|\theta)f(s_m,\epsilon_m^j|\theta)\Big)
\end{aligned}$$

For the inner sum, the random variables $(s_m, \epsilon_m^i)$ and $(s_m, \epsilon_m^j)$ share the same value of the state $s_m$, so they are in general correlated. The variance of the sum $N$ of correlated random variables is equal to the sum of variances, plus a term involving the sum of covariances. The inner sum has $\frac{N(N-1)}{2}$ elements. The number of unique pairs that can assembled from $\frac{N(N-1)}{2}$ elements is equal to:

$$\binom{\frac{N(N-1)}{2}}{2} = \frac{1}{2}\Big(\frac{N(N-1)}{2} - 1\Big)\Big(\frac{N(N-1)}{2}\Big)$$

There a two types of covariance terms that can appear when calculating the variance of $\sum_{1\leq i<j}^{N} f(s_m, \epsilon_m^i|\theta) f(s_m, \epsilon_m^j|\theta)$. The first type contains the same shock appearing twice, for instance:

$$\mathrm{Cov}(f(s_m, \epsilon_m^1|\theta) f(s_m, \epsilon_m^2|\theta), f(s_m, \epsilon_m^1|\theta) f(s_m, \epsilon_m^3|\theta))$$

There are $N\binom{N-1}{2} = \frac{N(N-1)}{2}(N-2)$ of such elements. Indeed, there are $N$ possible ways of selecting an index $i$ among $N$ elements. Then, there are $\binom{N-1}{2}$ possible ways of selecting two other indices among the remaining $N-1$ available indices. These covariance terms have equal value, because $\epsilon_m^i$ and $\epsilon_m^j$ have the same distribution and are independent from each other (which allows us to select three indices $i, j, k = 1, 2, 3$).

The second type of covariance term contains four different shocks, for instance:

$$\mathrm{Cov}(f(s_m, \epsilon_m^1|\theta) f(s_m, \epsilon_m^2|\theta), f(s_m, \epsilon_m^3|\theta) f(s_m, \epsilon_m^4|\theta))$$

The number of covariance terms of this second type is given by $\binom{\frac{N(N-1)}{2}}{2} - \frac{N(N-1)}{2}(N-2) = \frac{N(N-1)}{2}\Big(\frac{N(N-1)}{4} - N + 3/2\Big)$. Once again, because the shocks are independent from each other and have the same distribution, these covariance terms have the same value. Thus, we can select four indices $i, j, k, l = 1, 2, 3, 4$ without loss of generality.

Taking into account the number of variance and covariance terms, the variance of the inner sum can be written as:

$$\begin{aligned}
\mathrm{Var}\Big(\sum_{1\leq i<j}^{N} f(s_m, \epsilon_m^i|\theta) f(s_m, \epsilon_m^j|\theta)\Big) = \frac{N(N-1)}{2}\Big[ & \mathrm{Var}\Big(f(s_m, \epsilon_m^1|\theta) f(s_m, \epsilon_m^2|\theta)\Big) \\
& + 2(N-2)\,\mathrm{Cov}\Big(f(s_m, \epsilon_m^1|\theta) f(s_m, \epsilon_m^2|\theta), f(s_m, \epsilon_m^1|\theta) f(s_m, \epsilon_m^3|\theta)\Big) \\
& + 2\Big(\frac{N(N-1)}{4} - N + 3/2\Big)\times \\
& \quad \mathrm{Cov}\Big(f(s_m, \epsilon_m^1|\theta) f(s_m, \epsilon_m^2|\theta), f(s_m, \epsilon_m^3|\theta) f(s_m, \epsilon_m^4|\theta)\Big)\Big]
\end{aligned}$$

Using the previous expression, as well as the definition of $T \equiv \frac{MN}{2}$, one gets the expression reported in proposition 4:

$$\begin{aligned}\text{Var}\left(\mathcal{L}^U_{M,N}(\theta)\right) = \frac{1}{T(N-1)}\Big[&\text{Var}\left(f(s_m,\epsilon^1_m|\theta)f(s_m,\epsilon^2_m|\theta)\right)+\\ &+2(N-2)\,\text{Cov}\left(f(s_m,\epsilon^1_m|\theta)f(s_m,\epsilon^2_m|\theta), f(s_m,\epsilon^1_m|\theta)f(s_m,\epsilon^3_m|\theta)\right)\\ &+2\Big(\frac{N(N-1)}{4}-N+3/2\Big)\,\text{Cov}\left(f(s_m,\epsilon^1_m|\theta)f(s_m,\epsilon^2_m|\theta), f(s_m,\epsilon^3_m|\theta)f(s_m,\epsilon^4_m|\theta)\right)\Big]\end{aligned}$$

D.2. **Proof of proposition 4.2.** For the special case in which the state variable $s$ does not contribute to the variance of the loss function ($f(s_m,\varepsilon_m|\theta) = f(\varepsilon_m|\theta), \quad \forall s \in S$), I rewrite the covariance term as:

$$\begin{aligned}\text{Cov}(f(s_m,\varepsilon^1_m|\theta)f(s_m,\varepsilon^2_m|\theta), f(s_m,\varepsilon^1_m|\theta)f(s_m,\varepsilon^3_m|\theta)) &= \mathbb{E}\left(f(s_m,\varepsilon^1_m|\theta)^2 f(s_m,\varepsilon^2_m|\theta)f(s_m,\varepsilon^3_m|\theta)\right)\\ &\quad-\mathbb{E}\left(f(s_m,\varepsilon^1_m|\theta)f(s_m,\varepsilon^2_m|\theta)\right)\mathbb{E}\left(f(s_m,\varepsilon^1_m|\theta)f(s_m,\varepsilon^3_m|\theta)\right)\\ &= \mathbb{E}\left(f(s_m,\varepsilon^1_m|\theta)\right)^2\Big[\mathbb{E}\left(f(s_m,\varepsilon^1_m|\theta)^2\right)-\mathbb{E}\left(f(s_m,\varepsilon^1_m|\theta)\right)^2\Big]\\ &= \mathbb{E}\left(f(s_m,\varepsilon^1_m|\theta)\right)^2\text{Var}(f(s_m,\varepsilon^1_m|\theta))\end{aligned}$$

The first line uses the equality $\text{Cov}(X,Y) = \mathbb{E}[XY] - \mathbb{E}[X]\,\mathbb{E}[Y]$. To go from the first to the second line, I note that when $f(s_m,\varepsilon_m|\theta) = f(\varepsilon_m|\theta)$ for all $s \in S$, the only source of randomness in $f(s_m,\varepsilon^i_m|\theta)$ comes from $\varepsilon^i_m$. By assumption $\varepsilon^i_m$ and $\varepsilon^j_m$ are i.i.d. for $i \neq j$. I then use the fact that the expectation of the product of independent variables is equal to the product of expectations. These expectation terms have equal value, because the shocks all follow the same distribution, which allows us to select an index $i = 1$. The third line comes from recognizing the equality $\text{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$.

If two variables X and Y are independent, the variance of their product is given by:

$$\text{Var}(XY) = \mathbb{E}(X)^2\,\text{Var}(Y) + \mathbb{E}(Y)^2\,\text{Var}\,X + \text{Var}(X)\,\text{Var}(Y)$$

Using the formula of the variance of the product of independent random variables for $\text{Var}(f(s_m,\varepsilon^1_m|\theta)f(s_m,\varepsilon^2_m|\theta))$, one gets:

$$\text{Var}(f(s_m,\varepsilon^1_m|\theta)f(s_m,\varepsilon^2_m|\theta)) = \text{Var}(f(s_m,\varepsilon^1_m|\theta))^2 + 2\,\mathbb{E}\left(f(s_m,\varepsilon^1_m|\theta)\right)^2\text{Var}(f(s_m,\varepsilon^1_m|\theta))$$

Using once again the assumption that $\varepsilon^i_m$ and $\varepsilon^j_m$ are i.i.d. for $i \neq j$, it is easy to see that the covariance terms in which four different shocks appear are equal to zero:

$$\begin{aligned}\text{Cov}(f(s_m,\varepsilon_m^1|\theta)f(s_m,\varepsilon_m^2|\theta),f(s_m,\varepsilon_m^3|\theta)f(s_m,\varepsilon_m^4|\theta)) &= \mathbb{E}\left(f(s_m,\varepsilon_m^1|\theta)f(s_m,\varepsilon_m^2|\theta)f(s_m,\varepsilon_m^3 f(s_m,\varepsilon_m^4|\theta)\right)\\ &\quad - \mathbb{E}\left(f(s_m,\varepsilon_m^1|\theta)f(s_m,\varepsilon_m^2|\theta)\right)\mathbb{E}\left(f(s_m,\varepsilon_m^3|\theta)f(s_m,\varepsilon_m^4|\theta)\right)\\ &= \mathbb{E}\left(f(s_m,\varepsilon_m^1|\theta)\right)^4 - \mathbb{E}\left(f(s_m,\varepsilon_m^1|\theta)\right)^4\\ &= 0\end{aligned}$$

Substituting these formulas within the expression for the variance of the loss and simplifying, one gets the result reported in the second point of proposition 4:

$$\text{Var}(\mathcal{L}_{M,N}^U(\theta)) = \frac{1}{T(N-1)}\,\text{Var}\left(f(s_m,\varepsilon_m^1|\theta)\right)^2 + \frac{2}{T}E\big(f(s_m,\varepsilon_m^1|\theta)\big)^2\,\text{Var}(f(s_m,\varepsilon_m^1|\theta))$$

D.3. **Proof of proposition 4.3.**

For the special case in which the shock variable $\varepsilon$ does not contribute to the variance of the loss function ($f(s_m,\varepsilon_m|\theta) = f(s_m|\theta), \quad \forall\varepsilon_m \in \mathcal{E}$), the covariance terms are all equal:

$$\begin{aligned}&\text{Cov}\left(f(s_m,\epsilon_m^1|\theta)f(s_m,\epsilon_m^2|\theta),f(s_m,\epsilon_m^1|\theta)f(s_m,\epsilon_m^3|\theta)\right)\\ &= \text{Cov}\left(f(s_m,\epsilon_m^1|\theta)f(s_m,\epsilon_m^2|\theta),f(s_m,\epsilon_m^3|\theta)f(s_m,\epsilon_m^4|\theta)\right)\\ &= \text{Cov}\left(f(s_m,\epsilon_m^1|\theta)^2,f(s_m,\epsilon_m^1|\theta)^2\right)\\ &= \text{Var}\left(f(s_m,\epsilon_m^1|\theta)^2\right)\end{aligned}$$

We also have:

$$\text{Var}\left(f(s_m,\epsilon_m^1|\theta)f(s_m,\epsilon_m^2|\theta)\right) = \text{Var}\left(f(s_m,\epsilon_m^1|\theta)^2\right)$$

Simplifying and using the definition of $T = \frac{MN}{2}$, we get the third point of proposition 4:

$$\text{Var}(\mathcal{L}_{M,N}^U(\theta)) = \frac{1}{M}\Big[\,\text{Var}\left(f(s_m,\varepsilon_m^1|\theta)^2\right)\Big]$$

## Appendix E. Computational details

Numerical exercises are realized on the same computer (Intel(R) Core(TM) i7-8850H CPU @ 2.60GHz), which does not have a GPU. For the code, I use Python version 3.8.10. I code the bc-MC operator using Pytorch 1.13.1. For numerical integration in the Time Iteration algorithm, I use Chaospy.

BANQUE CENTRALE DU LUXEMBOURG

EUROSYSTÈME

2, boulevard Royal
L-2983 Luxembourg

Tél.: +352 4774-1
Fax: +352 4774 4910

www.bcl.lu • info@bcl.lu