THE GRADIENT-FREE BIAS-CORRECTED MONTE CARLO OPERATOR: SOLVING ECONOMIC MODELS WITH NEURAL NETWORKS WITHOUT BACKPROPAGATION

JULIEN PASCAL

Abstract. This paper presents a novel methodology for solving high-dimensional economic models with neural networks when the exact calculation of the gradient by backpropagation is impractical or inapplicable. This method relies on the gradient-free bias-corrected Monte Carlo (bc-MC) operator, which constitutes, under certain conditions stated in the paper, an asymptotically unbiased estimator of the gradient of the loss function characterizing the economic model under consideration. This new methodology, requiring only two evaluations of a residual function to approximate the gradient of the loss function, regardless of the model's dimensionality, is well-suited for high-dimensional models. I demonstrate that the gradient-free bias-corrected Monte Carlo operator has appealing properties as long as the economic model satisfies Lipschitz continuity, making it well-suited for situations involving non-differentiable loss functions. I demonstrate the broad applicability of the gradient-free bc-MC operator by solving large-scale overlapping generations (OLG) models with aggregate uncertainty, including scenarios involving borrowing constraints that introduce non-differentiability in households' optimization problems.

Keywords: Dynamic programming, Neural Networks, Machine Learning, Monte Carlo, Derivative-free Methods.
JEL: C45, C61, C63, C68, E32, E37.

February 2025. Julien Pascal: Banque centrale du Luxembourg, Département Économie et Recherche, 2 boulevard Royal, L-2983 Luxembourg (julien.pascal@bcl.lu). The results in this paper are preliminary materials circulated to stimulate discussion and critical comment. This paper should not be reported as representing the views of the BCL or the Eurosystem. The views expressed are those of the author and may not be shared by other research staff or policymakers in the BCL or the Eurosystem.

1. Introduction

Many important economic questions require the consideration of high-dimensional economic models. For instance, the Heterogeneous Agent New Keynesian (HANK) literature has emphasized that central banks ought to take into consideration the distribution of wealth among households when making monetary policy decisions (Kaplan, Moll, and Violante, 2018). This is a challenging task, as this distribution is, in principle, a time-varying and infinite-dimensional object. In a monetary union, the design of optimal fiscal and monetary policies requires monitoring economic activity across the N member countries (Farhi and Werning, 2017). In the euro area, this N-body problem involves $N = 20$ member states. Another example of high-dimensional economic models is overlapping generations (OLG) models, such as those used by Marchiori and Pierrard (2015) to analyze how life-cycle dynamics can shape Luxembourg's long-term economic behavior. In OLG models, tracking the wealth and employment status of each age group leads to a high-dimensional state vector. Often, these economic models are solved by linearization around a certain point, typically a steady state with no aggregate uncertainty. This approach is often appropriate, as linearization methods have strong theoretical foundations (see for instance Schmitt-Grohé and Uribe (2004) and Fernández-Villaverde, Rubio-Ramírez, and Schorfheide (2016)) and are relatively easy to implement.[1] However, when the non-stochastic steady state is ill-defined, or when linearization results in undesirable features for the problem under consideration (e.g., certainty equivalence), alternative global methods may be used.

This paper belongs to the literature that develops *global methods* to solve dynamic programming problems originating from economic models, in particular the body of work that uses machine learning techniques. It has recently been recognized that the tools used in the machine learning literature, especially neural networks in the form of multilayer perceptrons (MLPs), can be used to solve large scale economic models.[2] In particular, Azinovic, Gaegauf, and Scheidegger (2022) parametrize the policy functions and/or the value functions arising from economic optimization problems with an MLP, while using Monte Carlo integration to approximate expectations with respect to the state vector.

[1] Especially when using a linearization toolbox such as Dynare (Adjemian et al., 2011).

[2] An introduction to deep learning techniques in Economics with an application to DSGE models is available in Beck et al. (2024). See also Fernández-Villaverde, Nuño, and Perla (2024).

They demonstrate that this approach can successfully find numerical solutions to high-dimensional economic models. For instance, Folini et al. (2024) use this methodology to analyze climate change mitigation strategies within an economic model interacting with a climate block. However, in the method developed by Azinovic, Gaegauf, and Scheidegger (2022), uncertainty is modeled as low-dimensional discrete Markov chains, allowing the exact evaluation of expectations with respect to innovation vectors. Hence, this approach bypasses the problem that most economic models are generally expressed as stochastic functional equations involving two nested expectations: $\mathbb{E}_{\boldsymbol{s}}\left[\mathbb{E}_{\boldsymbol{\varepsilon}}\left(f(\boldsymbol{s},\boldsymbol{\varepsilon})\right)^2\right]=0$, where $\boldsymbol{s}$ denotes the state vector and $\boldsymbol{\varepsilon}$ denotes the innovation vector. In general, the expectation with respect to the innovation vector cannot be evaluated exactly, for instance because $\boldsymbol{\varepsilon}$ has continuous support. In that case, one solution is to use Monte Carlo integration for the expectation with respect to the innovation vector as well. However, an efficient Monte Carlo integration scheme would require approximating a single integral $\mathbb{E}_{\boldsymbol{s},\boldsymbol{\varepsilon}}\left[\left(f(\boldsymbol{s},\boldsymbol{\varepsilon})\right)^2\right]$, which unfortunately is not equal to $\mathbb{E}_{\boldsymbol{s}}\left[\mathbb{E}_{\boldsymbol{\varepsilon}}\left(f(\boldsymbol{s},\boldsymbol{\varepsilon})\right)^2\right]$, because of Jensen's inequality.

In independent work, Maliar, Maliar, and Winant (2021) develop an approach that can successfully deal with this issue. In a nutshell, their approach relies on working with two independent copies of each innovation vector, denoted by $\boldsymbol{\varepsilon^1}$ and $\boldsymbol{\varepsilon^2}$. Independence of these two copies allows the authors to express the two nested expectations with a single unnested expectation $\mathbb{E}_{\boldsymbol{s},\boldsymbol{\varepsilon^1},\boldsymbol{\varepsilon^2}}\left[f(\boldsymbol{s},\boldsymbol{\varepsilon^1})f(\boldsymbol{s},\boldsymbol{\varepsilon^2})\right]$, which can then be efficiently approximated using Monte Carlo integration. This approach has been used for instance to study discrete labor choices in a model à la Krusell-Smith (Maliar and Maliar, 2022). Pascal (2024) demonstrates that this approach can be generalized by using N independent copies of the innovation vector, where the optimal value of N depends on the model under scrutiny. The author also shows that proceeding in this way can also be interpreted as imposing a bias correction to a naive Monte Carlo integration applied to the nested formula $\mathbb{E}_{\boldsymbol{s}}\left[\mathbb{E}_{\boldsymbol{\varepsilon}}\left(f(\boldsymbol{s},\boldsymbol{\varepsilon})\right)^2\right]$, hence the name bias-corrected Monte Carlo (bc-MC) operator. The bc-MC operator is, under certain conditions on the primitives of the model, the best way to proceed, as it is the *minimum-variance unbiased estimator* (MVUE) of the nested expectation formula. Given this optimality result, this paper focuses on the bc-MC operator and its gradient-free extension.

The present paper extends the analysis of the bc-MC operator to the case in which gradients cannot be trivially calculated using backpropagation. Neural networks, in particular

MLPs, are generally trained by gradient descent or its variants like the Adam algorithm. Thus, knowledge of the gradient of the loss function characterizing an economic model is key. So far, the existing literature in Economics assumes that the gradient of the loss function can be easily calculated. This is certainly the case when economic models are implemented using modern software compatible with automatic differentiation, such as Pytorch (Paszke et al., 2017) or Tensorflow (Abadi et al., 2015), as the backpropagation algorithm can be used to get an exact value of the gradient very efficiently. Indeed, when using the backpropagation algorithm to calculate the gradient of a scalar-valued loss function, the runtime cost is at most four times the cost of evaluating the loss function itself. However, there are at least three reasons why one might prefer to train neural networks without backpropagation. First, if a legacy implementation of an economic model already exists, the time required to translate the existing codebase to software compatible with automatic differentiation might be prohibitive. Second, when using backpropagation, all intermediate computations must be stored in memory, which may not be possible in some extreme cases because of memory limitations. Third, extending backpropagation to non-smooth loss functions is non-trivial and significantly more computationally intensive than in the smooth case. This is why alternatives to backpropagation in this context have been developed.

The first alternative that may come to mind, which predates the invention of backpropagation, is the deterministic finite difference method. Because the gradient of a function is simply a vector of partial derivatives, one may be tempted to calculate each partial derivative using a forward difference scheme, requiring $n+1$ evaluations of the loss function if the neural network has n parameters. However, in the context of training neural networks, this approach is impractical, as neural networks often have thousands, if not millions or even billions of parameters. This is why in this paper I focus on using *randomized directional derivative estimators* to estimate the gradient.[3] As the name suggests, they involve using the directional derivative, which can be efficiently approximated, even when n is large. When the directional derivative is combined with the idea of random sampling from a symmetric distribution around the origin, one obtains an unbiased estimator of the gradient. These ideas can be applied to the loss functions characterizing the solutions of economic models, leading to the gradient-free bc-MC operator. While

[3]Alternatively, they are named "randomized finite difference methods" in Scheinberg (2022). Instead, Nesterov and Spokoiny (2017) use the term "random derivative-free methods".

this method is "gradient-free" because no exact value of the gradient is ever calculated during training, it remains gradient-based in spirit. When the exact value of the directional derivative is used, the expected update of the parameter vector is identical to that obtained using the gradient of the loss function.

In this paper, I make three main contributions to the existing literature. First, I present a new methodology, based on the gradient-free bc-MC operator, which can be used to find global numerical solutions of economic models when backpropagation is not available or not applicable. This methodology, relying on the *randomized directional derivative estimators*, is particularly relevant when the solution of an economic model is characterized by time-invariant functions, for instance, Bellman or policy functions, parameterized by an MLP. The gradient-free bc-MC operator requires only *two* evaluations of a residual function characterizing the solution of an economic model, independently of the number of parameters used in the underlying neural network. Additionally, intermediate calculations do not need to be stored in memory. Hence, this method remains applicable even when deep neural networks are used and when the residual function is slow to evaluate and memory-intensive. The gradient-free bc-MC operator is also applicable to loss functions with non-differentiabilities, which often occur because economic agents are facing inequality constraints, such as a borrowing constraint. Differentiability requirements are replaced by weaker Lipschitz continuity conditions. I give a set of conditions under which the gradient-free bc-MC operator inherits Lipschitz continuity from more primitive "residual functions", which are usually easier to analyze. With additional differentiability requirements, the gradient-free bc-MC operator is asymptotically equivalent to its gradient-based counterpart. Second, I demonstrate the broad applicability of the gradient-free bc-MC operator by solving two large-scale OLG models with aggregate uncertainty. In the first numerical experiment, households may freely borrow to smooth out consumption, but in the second one, households face an exogenous borrowing constraint, which introduces a non-differentiability in households' optimization problems. Third, inspired by earlier works using the Parameterized Expectation Algorithm (PEA), I present an efficient way of solving models with occasionally binding constraints (OBCs) using MLPs, by treating Lagrange multipliers as variables that can be adjusted ex post. I also make the novel observation that the bc-MC operator and the PEA are intimately

linked. In some sense, the PEA can be seen as a special case and as an asymptotically accurate approximation to the bc-MC operator.

This paper proceeds as follows. I first discuss how the gradient-free bc-MC operator relates to existing computational methods in Economics. I then formally define the gradient-free bc-MC operator before stating its key properties. In particular, I provide conditions under which the gradient-free bc-MC operator is smooth and unbiased. The next section illustrates the performance of the gradient-free bc-MC operator. The links between the bc-MC operator and PEA are discussed in that section within the context of OLG models with aggregate uncertainty and OBCs. The final section concludes.

2. Related literature

This paper relates to the literature that develops global methods to solve economic models with high-dimensional input or output spaces. When strong non-linearities are present and the use of linearization is inadequate, some authors have suggested the use of global solution algorithms for dynamic programming problems, such as value function iteration (VFI) or time iteration (TI), combined with grid constructions that can scale to high dimensional input spaces. In particular, Krueger and Kubler (2004) and Judd et al. (2014) argue in favor of using sparse grids, based on the seminal work of Smolyak (1963). Brumm and Scheidegger (2017) champion the use of adaptive sparse grids that selectively add grid points in areas where the value function or optimal decision rules exhibit high curvature or sharp changes.

This paper is more closely related to the body of work that combines machine learning tools to solve economic models. While some works rely on Gaussian processes (Scheidegger and Bilionis, 2019), there has been a growing interest around the use of neural networks, generally in the form of multilayer perceptrons (MLPs). For instance, Fernández-Villaverde, Hurtado, and Nuño (2023) use MLPs to approximate perceived laws of motion, rather than linear forecasting rules, as is more commonly the case when using the Krusell-Smith algorithm. In a discrete-time setting, Maliar, Maliar, and Winant (2021), Azinovic, Gaegauf, and Scheidegger (2022) and Pascal (2024) develop methods to approximate solutions of economic models where the policy or value functions are parameterized by neural networks. Duarte, Duarte, and Silva (2024) explore similar ideas in a continuous time setting. All these papers use the backpropagation algorithm

(Rumelhart, Hinton, and Williams, 1986) to update the MLP parameter vector. Hence, they implicitly assume an implementation compatible with automatic differentiation, as well as differentiability of the loss function characterizing an economic model. This paper contributes to this literature by developing a methodology that does not require these two conditions to be met. To do that, I use the *randomized directional derivative estimators*, first analyzed in the optimization mathematical literature (see, for instance, Nesterov and Spokoiny, 2017 and Berahas et al., 2022) and already used in the machine learning literature to build predictive models without relying on the backpropagation algorithm (Baydin et al., 2022, Silver et al., 2022). To the best of my knowledge, these estimators have not been used or studied in the context of economic modeling.

Interestingly, the neural network-based methods mentioned earlier resemble the parameterized expectations algorithm (PEA) proposed much earlier by Den Haan and Marcet (1990). In the PEA, expectations are parameterized by a polynomial function, and Monte Carlo is also used to sample from the state space. The parameter vector characterizing the polynomial function is fitted using least squares, while the updated parameter vector is used to generate a new sample of endogenous variables. This process is repeated until convergence of the parameter vector. In practical applications, the least squares stage might suffer from multicollinearity. Because neural networks are less likely to suffer from multicollinearity issues, Valaitis and Villa (2024) argue for using MLPs in the PEA context, rather than relying on hand-selected polynomials. The same idea was also explored earlier by Duffy and McNelis (2001), who also proposed to use neural networks when using the PEA. The authors also argued in favor of using genetic algorithms rather than gradient-based approaches for learning the MLP parameter vector. Because genetic algorithms do not rely on gradient information to search for optimal solutions, they can effectively handle functions that are non-differentiable. In this paper, I also propose a new computational method that can handle non-differentiable loss functions. However, instead of relying on minimization methods that do not use gradient information, which are often slow and computationally intensive, I retain a gradient-based learning method, where the gradient is taken for a smoothed version of the original loss function. Said differently, I handle the possibility of non-differentiable points at the level of the loss

function, rather than at the level of the minimization algorithm, as implemented by Duffy and McNelis (2001).

This work also relates to the PEA literature in the way it approaches solving economic models with occasionally binding constraints (OBCs). Economic models with OBCs generally result in systems of Karush–Kuhn–Tucker (KKT) conditions. For an economic model where households face a borrowing constraint, this typically results in Euler equations, augmented with Lagrange multipliers, with extra consistency conditions. One approach for numerically solving economic models with OBCs is to search for a function $f(.)$ that outputs consumption functions *and* Lagrange multipliers, such that they jointly solve the KKT conditions. This de facto doubles the size of the output vector that one must be predict, compared to similar models without OBCs. In this paper, I explore a less computationally intensive alternative, inspired by earlier works using the PEA. In particular, when dealing with models with borrowing or investment constraints, Marcet and Lorenzoni (2001) observe that Lagrange multipliers may be treated as residual variables that can be adjusted within the period. Christiano and Fisher (2000) found that this feature is the main reason why the PEA outperformed other global algorithms, when benchmarking different solution methods for solving models with OCBs.

In this paper, I show that the bc-MC operator can easily accommodate such a parameterized expectation approach, by decomposing the residual function into two parts, and similarly treating Lagrange multipliers as residual variables that are adjusted ex-post. In fact, I show that the bc-MC operator leads to a generalization of the PEA. Both methodologies can be seen as relying on realizations of a random vector next period, in order to approximate a conditional expectation. However, for each realization of a state vector, baseline PEA typically uses a single draw of an innovation vector, while the "bc-MC-PEA" operator uses N draws of independent innovation vectors. When further imposing linearity for the function predicting the value of the conditional expectation, and after using an approximation that becomes correct in a neighborhood of a solution, the gradient of the bc-MC operator leads to the usual PEA procedure of using the OLS solution of a linear regression of a target variable on a relevant state vector. Yet, with the "bc-MC-PEA" operator, the target variable is based on an average of N draws, rather than on a single draw. Said differently, baseline PEA can be interpreted as a special case

of a more general bc-MC operator. To the best of my knowledge, these facts are new in the computational economic literature.

3. The gradient-free bias-corrected Monte Carlo Operator

In this section, I first present the directional derivative estimator, which provides an unbiased estimator for the gradient of a function (under certain conditions). Hence, it can be used to minimize functions using (stochastic) gradient descent. For the directional derivative estimator to be well-behaved, the objective function must only be Lipschitz continuous, which is a weaker condition than everywhere-differentiability. I then focus on minimization problems arising from economic problems, which often result in nested expectations. When combining Monte Carlo integration and MLPs, such minimization problems can be solved using the bc-MC operator. I show that the directional derivative and the bc-MC operators can be combined to form a gradient-free bc-MC operator, which can be used to find approximate solutions of economic models, without backpropagation. I give conditions under which this estimator inherits Lipschitz continuity from more primitive "residual" functions, characterizing the solutions of an economic model (for instance, Euler equations and budget constraints). I also show that this gradient-free bc-MC operator is asymptotically equivalent to its gradient-based counterpart, when the "residual" functions are everywhere-differentiable. In that sense, the former can be seen as an extension of the latter to a less-smooth case.

3.1. **Randomized directional derivative estimator.**

Many economic models can be formulated as an optimization problem of the form

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^n} f(\boldsymbol{\theta}) \tag{1}$$

where $\boldsymbol{\theta}$ is a vector parametrizing policy or value functions that characterize the economic model under consideration. The function $f : \mathbb{R}^n \to \mathbb{R}$ is model-specific and might be for instance involve an Euler or a Bellman equation, as illustrated in the next section. A popular algorithm to find a numerical solution to the minimization problem (1) is the gradient descent algorithm, which uses the information contained in the gradient, to iteratively refine a current guess denoted by $\boldsymbol{\theta}_t$, using the updating rule

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma_t \nabla f(\boldsymbol{\theta}_t) \tag{2}$$

where γ_t is a parameter called the learning rate and $\nabla f(\boldsymbol{\theta}_i)$ denotes the gradient for the function f evaluated at the current guess $\boldsymbol{\theta}_t$. Because $\nabla f(\boldsymbol{\theta}_t)$ is the direction of steepest

ascent, the algorithm updates $\boldsymbol{\theta}_t$ by moving in the opposite direction, ensuring descent at each step. Thus, under sufficient smoothness conditions for f so that the gradient is well-defined, knowledge of $\nabla f(\boldsymbol{\theta}_t)$ is sufficient to numerically calculate the value of $\boldsymbol{\theta}^*$. Hence, having robust algorithms to calculate the gradient $\nabla f(\boldsymbol{\theta}_t)$ is key in order to solve numerically economic models.

The canonical numerical way to calculate the gradient of f at the point $\boldsymbol{\theta} \in \mathbb{R}^n$ is the *deterministic finite difference method*, which directly follows from the definition of the gradient. By definition, the gradient of f at the point $\boldsymbol{\theta} \in \mathbb{R}^n$ is the vector of partial derivatives, where the ith element is $\frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i}$. In turn, the partial derivative is defined as the limit $\frac{\partial f}{\partial \theta_i}(\boldsymbol{\theta}) = \lim_{h \to 0} \frac{f(\theta_1, \theta_2, ..., \theta_i + h, ..., \theta_n) - f(\boldsymbol{\theta})}{h}$. This definition motivates a popular scheme to approximate the value of $\frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i}$, replacing the limit operator with the use of a small value for h. More specifically, each component i of the gradient is approximated with $\frac{\partial f}{\partial \theta_i}(\boldsymbol{\theta}) \approx \frac{f(\theta_1, \theta_2, ..., \theta_i + h, ...\theta_n) - f(\boldsymbol{\theta})}{h}$, where h is a small strictly positive real number. This approach requires a total of $n+1$ evaluations of f. Clearly, when n is large and f is costly to evaluate, using the deterministic finite difference method to evaluate numerically the gradient becomes prohibitively slow and unpractical.

This is why alternatives methods to calculates the gradient are used when n is large. In particular, the backpropagation algorithm (Rumelhart, Hinton, and Williams, 1986) is the most popular algorithm used in machine learning to calculate the gradient of a given loss function. Backpropagation is an efficient algorithm that computes an approximation of the gradient with high numerical precision, subject to floating-point limitations.[4] Indeed, when using backpropagation to compute the gradient of a scalar-valued loss function, the runtime cost is at most four times that of evaluating the loss function. However, as previously noted, backpropagation comes with some drawbacks. Firstly, backpropagation requires implementation in a language and library that supports automatic differentiation. While this is usually not an issue for new research projects, it can be prohibitive when refactoring an existing implementation of f, as the time and effort needed for a complete overhaul of an existing code base may be substantial. Secondly, backpropagation requires storing in memory all intermediate computations, which might

[4]For computations in single-precision (*float32*), numerical accuracy is typically on the order of 10^{-7}, while in double-precision (*float64*), accuracy improves to approximately 10^{-16}.

not be feasible in some cases. Thirdly, the extension of backpropagation to non-smooth cases is non-trivial and is much more computationally intensive (Nesterov, 2005).

For these reasons, alternative ways to calculate the gradient of the loss function have been developed. They rely on using random draws of a vector $\mathbf{v}$ and numerical approximations of the directional derivative along that direction. That is why these methods might be called "randomized directional derivative estimators". The directional derivative of f at the point $\boldsymbol{\theta} \in \mathbb{R}^n$ in the direction of $\mathbf{v}$ is defined as the limit

$$D_{\mathbf{v}} f(\boldsymbol{\theta}) = \lim_{h \to 0} \frac{f(\boldsymbol{\theta} + h\mathbf{v}) - f(\boldsymbol{\theta})}{h} \tag{3}$$

provided this limit exists. By analogy with the deterministic finite difference scheme, one may approximate the directional derivative in the direction of $\mathbf{v}$ using a small positive real number denoted by h:

$$D_{F,\mathbf{v}} f(\boldsymbol{\theta}) = \frac{f(\boldsymbol{\theta} + h\mathbf{v}) - f(\boldsymbol{\theta})}{h} \tag{4}$$

Equation (4) requires only two evaluations of f, independently of the dimension of $\boldsymbol{\theta}$. Hence, one would hope to use them in order to approximate the gradient when n is large. It turns out that this is indeed possible. To see that, consider the case where f is differentiable at $\boldsymbol{\theta}$. In that case, the directional derivative can also be expressed as the dot product of the gradient of f at $\boldsymbol{\theta}$, denoted by $\nabla f(\boldsymbol{\theta})$, and the vector $\mathbf{v}$

$$D_{\mathbf{v}} f(\boldsymbol{\theta}) = \nabla f(\boldsymbol{\theta}) \cdot \mathbf{v} \tag{5}$$

Expression (5) makes it clear that the directional derivative can be interpreted as the projection of the gradient vector onto the direction spanned by $\mathbf{v}$. Hence, the directional derivative contains useful information about the gradient, which we would like to approximate in order to use the gradient descent algorithm. In fact, one may obtain unbiased estimators of the gradient using the directional derivative, as stated in proposition 1.

Proposition 1. *Consider the function $g : \mathbb{R}^n \to \mathbb{R}^n$ defined as*

$$g(\boldsymbol{\theta}) = D_{\mathbf{v}} f(\boldsymbol{\theta}) \mathbf{v} \tag{6}$$

where $\mathbf{v}$ is a normally distributed random vector in $\mathbb{R}^n$ with zero mean and a variance-covariance matrix given by the identity matrix $\mathbb{I}_n$. If f is differentiable at $\boldsymbol{\theta}$, then $g(\boldsymbol{\theta})$ is an unbiased estimator of the gradient $\nabla f(\boldsymbol{\theta})$.

Proof. See Appendix A □

To get some intuition on why $g(\boldsymbol{\theta})$ is an unbiased estimator for $\nabla f(\boldsymbol{\theta})$, first observe that the directional derivative of f gives the scalar projection of the gradient vector along the vector $\mathbf{v}$. Consider for instance the case where $\mathbf{v_k}$ points exactly in the direction of the gradient, which can be expressed as $\mathbf{v_k} = a\nabla f(\boldsymbol{\theta})$, with $a > 0$. Then $\nabla f(\boldsymbol{\theta}) \cdot \mathbf{v_k} = a\|\nabla f(\boldsymbol{\theta})\|^2$, so that $(\nabla f(\boldsymbol{\theta}) \cdot \mathbf{v_k})\mathbf{v_k}$ generates a large contribution in the direction of the true gradient in the average $\frac{1}{K}\sum_{k=1}^{K}(\nabla f(\boldsymbol{\theta})\cdot\mathbf{v_k})\mathbf{v_k} \approx \mathbb{E}\left[(\nabla f(\boldsymbol{\theta})\cdot\mathbf{v})\mathbf{v}\right]$. Now, consider the case where $\mathbf{v_k}$ points in the opposite direction of the gradient, which can be expressed as $\mathbf{v_k} = -a\nabla f(\boldsymbol{\theta})$, with $a > 0$. Then, the dot product between the gradient and the vector result in negative value: $\nabla f(\boldsymbol{\theta}) \cdot \mathbf{v_k} = -a\|\nabla f(\boldsymbol{\theta})\|^2$. However, multiplying this quantity with the vector $\mathbf{v_k}$ results in a vector that points in the direction of the gradient: $(\nabla f(\boldsymbol{\theta}) \cdot \mathbf{v_k})\mathbf{v_k} = a^2\|\nabla f(\boldsymbol{\theta})\|^2\nabla f(\boldsymbol{\theta})$. Hence, vectors pointing in the opposite direction of the gradient are reflected back and still provide a large contribution in the direction of the gradient in the average $\frac{1}{K}\sum_{k=1}^{K}(\nabla f(\boldsymbol{\theta}) \cdot \mathbf{v_k})\mathbf{v_k}$. Finally, if the vector $\mathbf{v_k}$ is orthogonal to the gradient, then its contribution in $\frac{1}{K}\sum_{k=1}^{K}(\nabla f(\boldsymbol{\theta}) \cdot \mathbf{v_k})\mathbf{v_k}$ is null, because $\nabla f(\boldsymbol{\theta}) \cdot \mathbf{v_k} = 0$. Because $\mathbf{v}$ is drawn from a multivariate normal distribution with zero mean and identity variance-covariance matrix, each direction is equally likely. Overall, this creates a situation in which the draws of $\mathbf{v_k}$ positively interfere so that, on average, $g(\boldsymbol{\theta})$ points exactly in the direction to the gradient of f evaluated at $\boldsymbol{\theta}$. Using an identity matrix for the variance-covariance matrix ensures that the expected value of $g(\boldsymbol{\theta})$ matches both the direction and magnitude of $\nabla f(\boldsymbol{\theta})$. This discussion is illustrated in Figure 1, using a simple two-dimensional function $f(x, y) = x^2 + y^2$, with $\frac{1}{K}\sum_{k=1}^{K}(\nabla f(\boldsymbol{\theta}) \cdot \mathbf{v_k})\mathbf{v_k}$ providing a good approximation of $\nabla f(\boldsymbol{\theta})$.

Proposition (1) combined with equation (4) lead to an estimator for the gradient of f evaluated at $\boldsymbol{\theta}$. The *randomized finite difference* scheme is defined as:

$$g_{F,\mathbf{v}}f(\boldsymbol{\theta}) = \frac{f(\boldsymbol{\theta} + h\mathbf{v}) - f(\boldsymbol{\theta})}{h}\mathbf{v} \tag{7}$$

where $\mathbf{v}$ is a normally distributed random vector in $\mathbb{R}^n$ with zero mean and a variance-covariance matrix given by the identity matrix $\mathbb{I}_n$.

The main advantage of estimator (7) is that it requires only two evaluations of f, independently of the dimension of $\boldsymbol{\theta}$. It is in sharp contrast with the $n+1$ function evaluations

required for the deterministic finite difference method. One drawback is that a single draw of $g_{F,\mathbf{v}}f(\boldsymbol{\theta})$ might give a poor approximation of the true gradient $\nabla f(\boldsymbol{\theta})$.

However, when minimizing a function by stochastic gradient descent, access to a noisy estimator of the gradient of the loss is generally enough to find numerically the minimum of a function. In fact, Nesterov and Spokoiny (2017) demonstrate that the estimator (7) can be used for minimization purposes, using an iterative scheme of the form (2), even for *non-smooth* and *non-convex* functions. Conditional on carefully choosing the learning rate γ_t and the value h, convergence of an iterative scheme of the form (2) is ensured for functions that are simply *Lipschitz continuous* on their domain[5]. Lipschitz continuity is a stronger condition than regular ϵ-δ continuity, but it is weaker condition than differentiability. For example, the functions $x \to |x|$ and $x \to \max\{0, x\}$ are Lipschitz continuous on $\mathbb{R}$, but they are not differentiable everywhere.

To get some intuition on why only Lipschitz continuity is required for the function being minimized, and not differentiability everywhere, it is instructive to see the connection between the randomized finite scheme and the *Gaussian smoothing operator*. The Gaussian smoothing operator applied to the function f is the new function $f_h(\boldsymbol{\theta}) = \mathbb{E}\left[f(\boldsymbol{\theta}+h\mathbf{v})\right]$, where the expectation is taken with respect to $\mathbf{v}$, denoting a zero mean normally distributed random vector with covariance matrix the identity matrix. One can show that $f_h(\boldsymbol{\theta})$ has better smoothness properties than the original function f, while preserving the potential convexity of f. In particular when $h > 0$, when the function f is Lipschitz continuous on its domain, $f_h(\boldsymbol{\theta})$, its Gaussian smoothed version f_h is a continuously differentiable function everywhere on its domain. The gradient of $f_h(\boldsymbol{\theta})$ is given by $\nabla_{\boldsymbol{\theta}} f_h(\boldsymbol{\theta}) = \frac{1}{h}\,\mathbb{E}\left[f(\boldsymbol{\theta} + h\mathbf{v})\mathbf{v}\right]$. Because $\mathbf{v}$ is a zero mean vector, $\mathbb{E}\left[f(\boldsymbol{\theta})\mathbf{v}\right] = 0$, and the gradient of $f_h(\boldsymbol{\theta})$ is also equal to $\mathbb{E}\left[\frac{f(\boldsymbol{\theta}+h\mathbf{v})-f(\boldsymbol{\theta})}{h}\mathbf{v}\right]$, which admits an unbiased estimator given by one realization of the randomized forward difference operator defined in equation (7). Said differently, the randomized forward difference scheme can be seen as originating

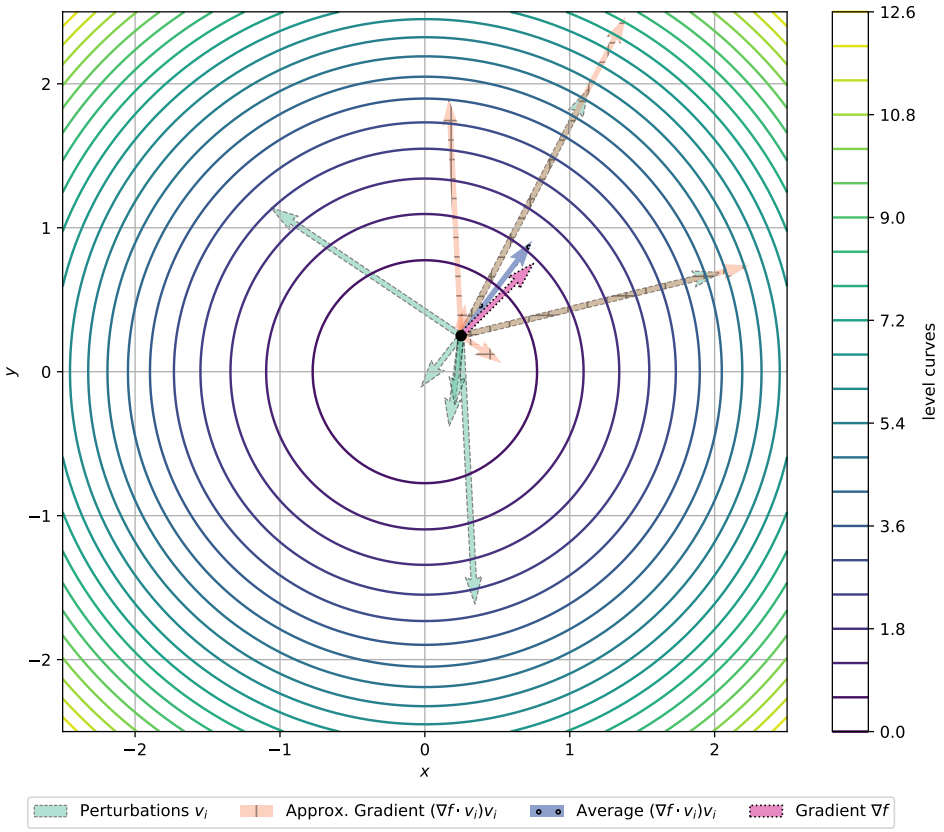
[5]A function $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous if there exists a constant $L \geq 0$ such that for all $x, y \in \mathbb{R}^n$, the following inequality holds: $|f(x) - f(y)| \leq L\|x - y\|$, where $\|.\|$ denotes the norm in $\mathbb{R}^n$, typically the Euclidean norm. The convergence result of Nesterov and Spokoiny (2017) requires knowing the constant L, in order to set the values for γ_t and the value h.

from the Gaussian smoothing operator, which *produces differentiable functions as long as the original function is only Lipschtiz continuous.*

I now illustrate the fact that the randomized directional derivative estimators can be used for minimization purposes using two test objective functions. First, I find the minimum of the function $f(x, y) = x^2 + y^2$ using the gradient descent algorithm with the analytical gradient $(2x, 2y)$, as well as with the randomized directional derivative estimator defined in equation (7). I use a learning rate equal to 0.05, set $h = 1.10^{-4}$. When using the randomized directional derivative estimator, I perform 100 replications of the gradient descent algorithm, using the same starting value. Results are presented in Figure 2. While gradient descent using the exact gradient clearly outperforms gradient descent based on a randomized finite difference scheme, the latter yields good approximations of the minimum of f. In a second experiment, I repeat the same procedure with the non-smooth function $f(x, y) = |x - 1| + |1 + y - 2x|$. This function, minimized at $(1, 1)$, is non-differentiable for $x = 1$ and $y = 2x - 1$, but it Lipschitz continuous on its domain.[6] Results, presented in Figure 3, illustrate that direct gradient descent struggles with non-smooth functions due to oscillations, whereas the randomized finite difference scheme better approximates the minimizer in such cases.

[6]The gradient of f is given by $\nabla f(x, y) = (-1, 1)$ when $x > 1$ and $1 + y - 2x > 0$, $(3, -1)$ when $x > 1$ and $1 + y - 2x < 0$, $(-3, 1)$ when $x < 1$ and $1 + y - 2x > 0$, $(1, -1)$ when $x < 1$ and $1 + y - 2x < 0$. For non-differentiable points, I use one-sided derivatives.

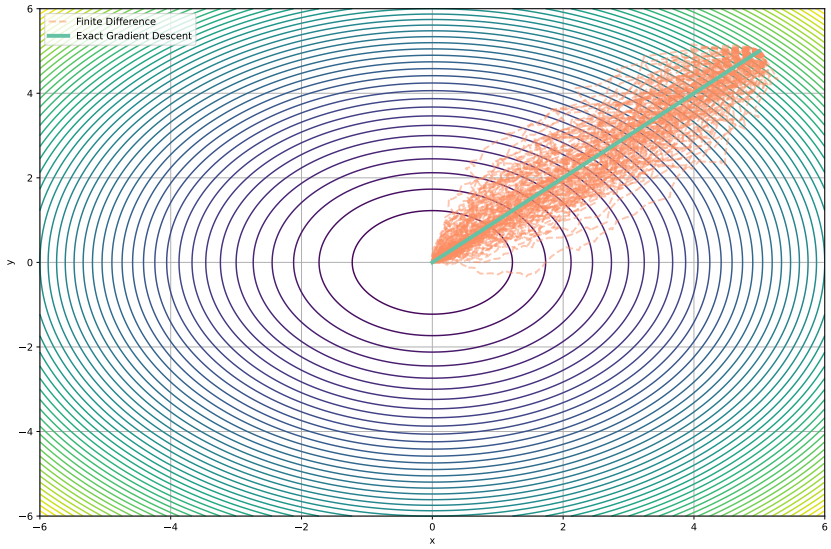
FIGURE 1. Approximating the gradient of $\nabla f(\boldsymbol{\theta})$ with $g(\boldsymbol{\theta})$ from proposition (1)



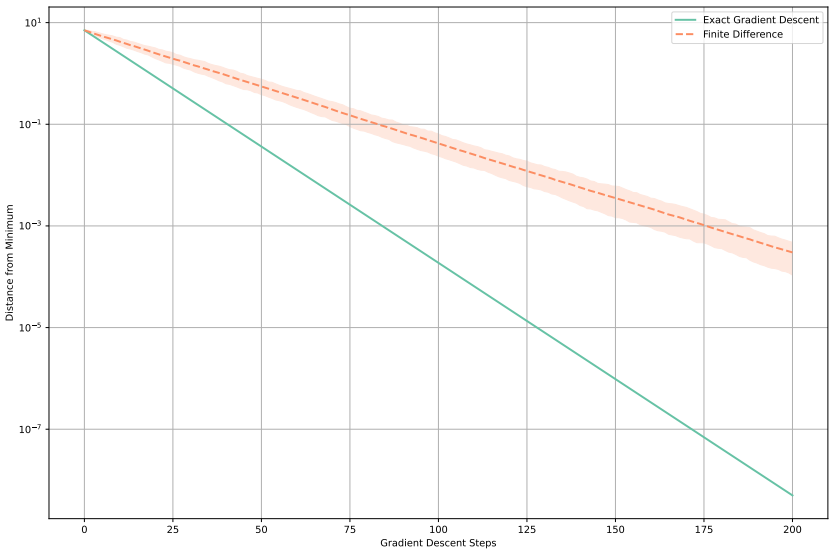
Notes. This figure illustrates the use $g(\boldsymbol{\theta}) = (\nabla f(\boldsymbol{\theta}) \cdot \mathbf{v})\mathbf{v}$ (orange stripped arrows) to approximate the gradient $\nabla f(\boldsymbol{\theta})$ (purple dotted arrow) for the function $f(\boldsymbol{\theta}) = f(x, y) = x^2 + y^2$, where $\mathbf{v}$ is a zero-mean normally distributed random vector with variance-covariance matrix the identity matrix. For this plot, I use seven independent draws of $\mathbf{v}$ (green dashed arrows). Calculations are realized for $\boldsymbol{\theta} = (\frac{1}{4}, \frac{1}{4})$. The average $\frac{1}{K}\sum_{k=1}^{K}(\nabla f(\boldsymbol{\theta}) \cdot \mathbf{v_k})\mathbf{v_k}$ is represented by the blue arrow filled with dark dots.

FIGURE 2. Minimization of $f(x, y) = x^2 + y^2$ by gradient descent

(A) Trajectories using the exact gradient and the randomized finite difference scheme



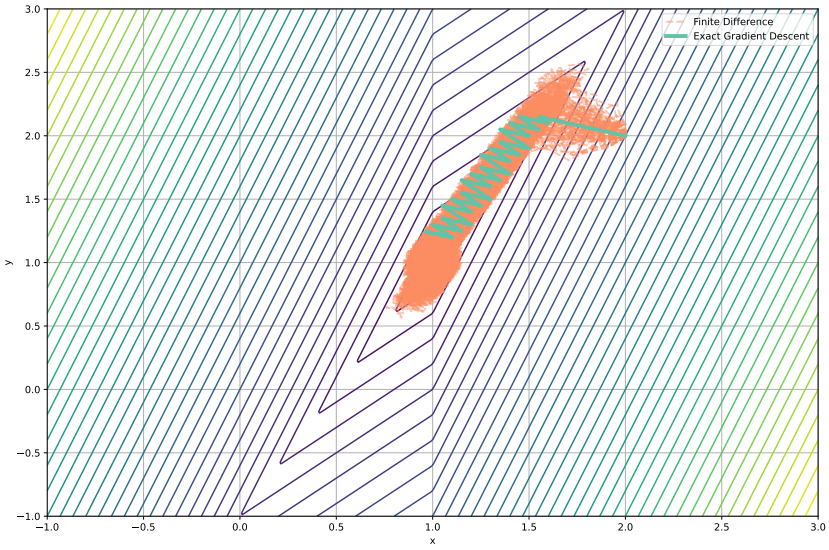
(B) Distance from minimum $(0, 0)$



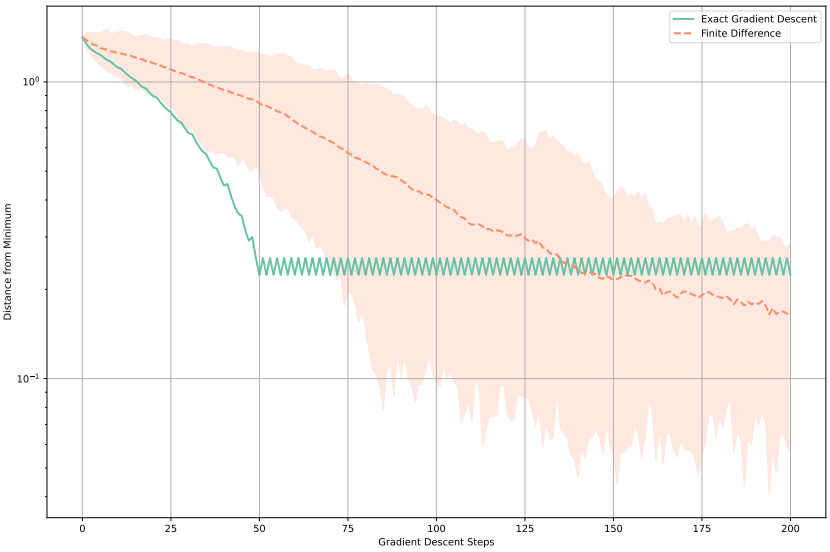
Notes. This figure illustrates the minimization of the function $f(x, y) = x^2 + y^2$ using gradient descent. The gradient is calculated exactly with the formula $(2x, 2y)$, or approximated using the randomized finite difference scheme defined in equation (7). The top panel shows the level curves of f, as well as gradient descent trajectories using different schemes to calculate the gradient. For the randomized finite difference scheme, 100 independent replications are used. The bottom panel displays the Euclidean distance from the minimum of f. For the randomized finite difference scheme, shaded areas represent the P10-P90 inter-percentile range, while lines represent the average value across the 100 replications.

FIGURE 3. Minimization of $f(x, y) = |x - 1| + |1 + y - 2x|$ by gradient descent

(A) Trajectories using the exact gradient and the randomized finite difference scheme



(B) Distance from minimum $(1, 1)$



Notes. This figure illustrates the minimization of the function $f(x, y) = |x - 1| + |1 + y - 2x|$ using gradient descent. The gradient is calculated exactly (see footnote [6]) or approximated using the randomized finite difference scheme defined in equation (7). The top panel shows the level curves of f, as well as gradient descent trajectories using different schemes to calculate the gradient. For the randomized finite difference scheme, 100 independent replications are used. The bottom panel displays the Euclidean distance from the minimum of f. For the randomized finite difference scheme, shaded areas represent the P10-P90 inter-percentile range, while lines represent the average value across the 100 replications.

3.2. **Randomized directional derivative estimator applied to the bc-MC operator.** Now, I focus on the analysis of minimization problems arising from economic modeling. In particular, economic models can often be written as a set of K stochastic equations of the form:

$$\mathbb{E}_{\boldsymbol{s}}\left[\mathbb{E}_{\boldsymbol{\varepsilon}}\left(f_k(\boldsymbol{s},\boldsymbol{\varepsilon}|\boldsymbol{\theta^*})\right)^2\right]=0 \tag{8}$$

where $\boldsymbol{s}$ denotes a state vector, $\boldsymbol{\varepsilon}$ denotes a zero-mean innovation vector, and $\boldsymbol{\theta^*}$ is vector parameterizing the policy or value functions characterizing the economic model under consideration. The K stochastic equations may be grouped together to form a unique loss function to be minimized:

$$\boldsymbol{\theta^*}=\arg\min_{\boldsymbol{\theta}\in\mathbb{R}^n}\mathcal{L}(\boldsymbol{\theta})=\arg\min_{\boldsymbol{\theta}\in\mathbb{R}^n}\sum_{k=1}^{K}\mu_k\,\mathbb{E}_{\boldsymbol{s}}\left[\mathbb{E}_{\boldsymbol{\varepsilon}}\left(f_k(\boldsymbol{s},\boldsymbol{\varepsilon}|\boldsymbol{\theta})\right)^2\right] \tag{9}$$

where μ_k are K subjective weights. Because closed-form solutions for the expectation operators in equations (8) and (9) are in general unavailable, numerical approximation must be used. Using Monte Carlo integration, which consists of using sample means to approximate population means, is a popular approach because it is easy to implement and works well even when $\boldsymbol{s}$ and $\boldsymbol{\varepsilon}$ are high-dimensional vectors. An intuitive implementation of Monte Carlo integration would suggest the following estimator for the loss function defined in equation (9):

$$\mathcal{L}^{B}_{M,N}(\boldsymbol{\theta})=\sum_{k=1}^{K}\mu_k\frac{1}{M}\sum_{m=1}^{M}\left[\left(\frac{1}{N}\sum_{n=1}^{N}f_k(\boldsymbol{s_m},\boldsymbol{\varepsilon_n}|\boldsymbol{\theta})\right)^2\right] \tag{10}$$

As shown in Pascal (2024), the naive use of two nested sample means to approximate (8) results in a biased estimator. This bias arises because, due to Jensen's inequality, the expectation of the squared sample mean exceeds the square of the population mean, unless the variance of the underlying distribution is exactly zero. The bias disappears when taking N to infinity, which is not feasible in practical applications. Instead, the author suggests to use the following unbiased estimator for the loss function, which removes the small sample bias in (10):

$$\mathcal{L}_{M,N}(\boldsymbol{\theta})=\sum_{k=1}^{K}\mu_k\frac{2}{MN(N-1)}\sum_{m=1}^{M}\sum_{1\le i<j}^{N}f_k(\boldsymbol{s_m},\boldsymbol{\varepsilon^i_m}|\boldsymbol{\theta})f_k(\boldsymbol{s_m},\boldsymbol{\varepsilon^j_m}|\boldsymbol{\theta}) \tag{11}$$

where $\boldsymbol{\varepsilon^i}$ and $\boldsymbol{\varepsilon^j}$ are random variables drawn from the same distribution as $\boldsymbol{\varepsilon}$, independent from each other. Under some assumptions on the primitives of the model, this bias-corrected Monte Carlo (bc-MC) operator is the best unbiased estimator of the loss

function, in the sense that it is the minimum-variance unbiased estimator (MVUE). The hyperparameters M and N control the extent to which one is willing to explore the state space, versus exploring the innovation space. They can be set automatically by minimizing the variance of the loss function during training, while keeping a constant computational budget, which can be proxied by MN. Importantly for practical applications, equation (11) can be factorized as:

$$\mathcal{L}_{M,N}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \mu_k \frac{2}{MN(N-1)} \boldsymbol{f_{k,\theta}^T} \Lambda \boldsymbol{f_{k,\theta}} \tag{12}$$

where $\boldsymbol{f_{k,\theta}}$ is a column vector of size MN such that its transpose $\boldsymbol{f_{k,\theta}^T}$ is defined as $\big(f_k(\boldsymbol{s_1}, \boldsymbol{\varepsilon_1^1}|\boldsymbol{\theta}), f_k(\boldsymbol{s_1}, \boldsymbol{\varepsilon_1^2}|\boldsymbol{\theta}), \ldots, f_k(\boldsymbol{s_1}, \boldsymbol{\varepsilon_1^N}|\boldsymbol{\theta}), f_k(\boldsymbol{s_2}, \boldsymbol{\varepsilon_2^1}|\boldsymbol{\theta}), f_k(\boldsymbol{s_2}, \boldsymbol{\varepsilon_2^2}|\boldsymbol{\theta}), \ldots, f_k(\boldsymbol{s_M}, \boldsymbol{\varepsilon_M^N}|\boldsymbol{\theta})\big)$. The matrix Λ is an $MN \times MN$ matrix defined as the Kronecker product $\mathbb{I}_M \otimes \mathbb{U}_N$, with $\mathbb{I}_M$ being the $M \times M$ identity matrix and $\mathbb{U}_N$ being an $N \times N$ upper triangular matrix whose diagonal elements are all equal to zero, while the entries above the main diagonal are equal to one. If the subjective weights are constant, for instance, $\mu_k = \frac{1}{K}$, equation (12) can be further factorized as a single quadratic form:

$$\mathcal{L}_{M,N}(\boldsymbol{\theta}) = \frac{2}{KMN(N-1)} \boldsymbol{f_\theta^T} \Sigma \boldsymbol{f_\theta} \tag{13}$$

where $\boldsymbol{f_\theta}$ is a column vector of size KMN such that its transpose $\boldsymbol{f_{k,\theta}^T}$ is defined as $\big(\boldsymbol{f_{1,\theta}^T}, \boldsymbol{f_{2,\theta}^T}, ..., \boldsymbol{f_{K,\theta}^T}\big)$, and Σ is a $KMN \times KMN$ matrix defined as the Kronecker product $\mathbb{I}_K \otimes \Lambda$. While the matrices Λ or Σ may be large, they are sparse, leading to very efficient implementation when using sparse matrix multiplication routines.

When using a software compatible with automatic differentiation, calculating the gradient of the loss using equations (12) or (13) can be efficiently done using backpropagation, because $\mathcal{L}_{M,N}(\boldsymbol{\theta})$ is a scalar value. This case is investigated in Maliar, Maliar, and Winant (2021) when $N = 2$, and in Pascal (2024) when N is learned during training and may vary over training instead of being fixed at $N = 2$. However, when automatic differentiation is not available or not desirable for the reasons listed previously, the randomized directional derivatives estimators from the previous section may be used.

I now discuss how the randomized directional derivative estimator from the previous section can be applied to the bc-MC loss function defined by equation (13). In order to invoke the results discussed in the previous section, this loss function must be Lipschitz continuous. The following proposition gives conditions under which the bc-MC loss

function inherits Lipschitz continuity from the more primitive functions f_k, which are easier to analyze and known once an economic model is specified.

Proposition 2. *Assume that for all values of the state and innovation vectors, the function $f : \mathbb{R}^n \to \mathbb{R}^{KMN}$ defined by $f(\boldsymbol{\theta}) = \boldsymbol{f_\theta}$ is Lipschitz continuous on its domain, where n denotes the dimension of the MLP's parameter vector, then*

(1) if f is bounded on $\mathbb{R}^n$, the loss function (13) is Lipschitz continuous on its domain;
(2) the restriction of the loss function (13) to the compact subset of $D \subset \mathbb{R}^n$ is Lipschitz continuous on D.

Proof. See Appendix B □

I now discuss the conditions listed in Proposition 2 and their applicability. First, Lipschitz continuity of the vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^{KMN}$ can be deduced from the Lipschitz continuity of the scalar-valued functions $f_k(\boldsymbol{s}, \boldsymbol{\varepsilon}|\boldsymbol{\theta})$. Indeed, if for all $k \in \{1, \ldots, K\}$ and for all $\boldsymbol{s}$ and $\boldsymbol{\varepsilon}$, the functions $f_k(\boldsymbol{s}, \boldsymbol{\varepsilon}|\boldsymbol{\theta}) : \mathbb{R}^n \to \mathbb{R}$ are Lipschitz continuous on $\mathbb{R}^n$, then the vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^{KMN}$, formed by stacking all $f_k(\boldsymbol{s}, \boldsymbol{\varepsilon}|\boldsymbol{\theta})$, is also Lipschitz continuous on $\mathbb{R}^n$ (as long as n is finite). Similarly, boundedness of the vector-valued function f, as mentioned in Proposition 2.1, is also achieved when, for all $k \in \{1, \ldots, K\}$ and for all $\boldsymbol{s}$ and $\boldsymbol{\varepsilon}$, the functions $f_k(\boldsymbol{s}, \boldsymbol{\varepsilon}|\boldsymbol{\theta}) : \mathbb{R}^n \to \mathbb{R}$ are bounded on $\mathbb{R}^n$.

Second, in economic applications, proving the boundedness of the function f, as required by 2.1, might be too restrictive or difficult. In such cases, Proposition 2.2 suggests working on a restricted compact subset $D \subset \mathbb{R}^n$, which contains the minimizer of (13). In practice, D might take the form of a Cartesian product of closed intervals $[a_1, b_1] \times ... \times [a_n, b_n]$. This can be enforced by using "gradient clipping", which consists of using a truncated or rescaled version of the gradient of the loss during gradient descent (see Zhang et al., 2020). Gradient clipping ensures that the norm of the gradient does not exceed a certain threshold, which de facto ensures that the parameter vector remains within a certain bounded set during the training process, as long as one uses a finite number gradient descent steps.

Let us now assume that the conditions of Proposition 2 are met. Thus, the bc-MC loss function (13) is Lipschitz continuous on $\mathbb{R}^n$ (or D). This in turn implies that its Gaussian smoothed version is continuously differentiable on its domain, with gradient

given by $\frac{1}{h}\mathbb{E}_{\mathbf{v}}\left[\mathcal{L}_{M,N}(\boldsymbol{\theta}+h\mathbf{v})\mathbf{v}\right]$. This leads to an estimator for the gradient of the loss function. Using the randomized finite difference scheme (7), one gets:

$$\begin{aligned}\nabla_\theta \mathcal{L}^F_{M,N}(\boldsymbol{\theta}) &= \frac{2}{KMN(N-1)}\frac{\mathbf{v}}{h}\Big(\boldsymbol{f}^T_{\boldsymbol{\theta}+h\mathbf{v}}\Sigma\boldsymbol{f}_{\boldsymbol{\theta}+h\mathbf{v}} - \boldsymbol{f}^T_{\boldsymbol{\theta}}\Sigma\boldsymbol{f}_{\boldsymbol{\theta}}\Big) \\ &= \frac{1}{KMN(N-1)}\frac{\mathbf{v}}{h}\Big(\big(\boldsymbol{f}_{\boldsymbol{\theta}+h\mathbf{v}} - \boldsymbol{f}_{\boldsymbol{\theta}}\big)^T\big(\Sigma+\Sigma^T\big)\big(\boldsymbol{f}_{\boldsymbol{\theta}+h\mathbf{v}} + \boldsymbol{f}_{\boldsymbol{\theta}}\big)\Big)\end{aligned} \tag{14}$$

where the first line simply follows from the definition of the randomized forward difference scheme, while the second line uses the fact that $\boldsymbol{x}^{\boldsymbol{T}}A\boldsymbol{x} = \boldsymbol{x}^{\boldsymbol{T}}\frac{A+A^T}{2}\boldsymbol{x}$, and that $\boldsymbol{x}^{\boldsymbol{T}}B\boldsymbol{x} - \boldsymbol{y}^{\boldsymbol{T}}B\boldsymbol{y} = (\boldsymbol{x}-\boldsymbol{y})^TB(\boldsymbol{x}+\boldsymbol{y})$ when B is a symmetric matrix. Equation (14) defines an estimator of the gradient of the bc-MC loss function $\mathcal{L}_{M,N}(\boldsymbol{\theta})$, where the bc-MC loss function is itself an estimator of the loss function $\mathcal{L}(\boldsymbol{\theta})$ appearing in equation (9). In what follows, I use the term "gradient-free bc-MC operator" when simultaneously using $\mathcal{L}_{M,N}(\boldsymbol{\theta})$ as an estimator for the loss function $\mathcal{L}(\boldsymbol{\theta})$, while using $\nabla_\theta\mathcal{L}^F_{M,N}(\boldsymbol{\theta})$ as an estimator for $\nabla_\theta\mathcal{L}(\boldsymbol{\theta})$.

Equation (14) can be seen as yielding a scaled version of the random direction vector $\mathbf{v}$, where the scaling coefficient depends on the number of stochastic functional equations characterizing the model (K), on hyperparameters (M, N, h), and on the value of a symmetric bilinear form $\boldsymbol{x}^{\boldsymbol{T}}B\boldsymbol{y}$ capturing the rate of change of the loss function in a neighborhood of the MLP's parameter $\boldsymbol{\theta}$. A striking feature of equation (14) is that, independently of the dimension of the MLP's parameter vector or of the underlying economic model, equation (14) requires only two different evaluations of the function $f:\mathbb{R}^n\to\mathbb{R}^{KMN}$ defined by $f(\boldsymbol{\theta}) = \boldsymbol{f}_{\boldsymbol{\theta}}$.

I now discuss unbiasedness. While the bc-MC operator (12) is an unbiased estimator of the loss function $\mathcal{L}(\boldsymbol{\theta})$, the estimator of the gradient $\nabla_\theta\mathcal{L}^F_{M,N}(\boldsymbol{\theta})$ is a biased estimator of the gradient of the loss function $\nabla_\theta\mathcal{L}(\boldsymbol{\theta})$. However, the bias vanishes as the scalar h tends to 0, as discussed in the next proposition.

Proposition 3. *Assume that for all values of the state and innovation vectors, the function $\psi:\mathbb{R}^n\to\mathbb{R}$ defined by $\psi(\boldsymbol{\theta}) = \mathcal{L}_{M,N}(\boldsymbol{\theta})$ is Lipschitz continuous and differentiable on its domain. Then, when the scalar h tends to 0, the estimator $\nabla_\theta\mathcal{L}^F_{M,N}(\boldsymbol{\theta})$ defined in equation (14) is an unbiased estimator of the gradient of the loss function, $\nabla_\theta\mathcal{L}(\boldsymbol{\theta})$.*

Proof. See Appendix C □

To simplify, Proposition 3 expresses conditions on the loss function itself. Following the approach of Proposition 2, it could also be expressed as conditions on the vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^{KMN}$ (or the scalar-values functions $f_k(\boldsymbol{s}, \boldsymbol{\varepsilon}|\boldsymbol{\theta})$, for $k \in \{1, \dots, K\}$). Note that if, for values of the state and innovation vectors, f is differentiable on its domain, then the function ψ is also differentiable on $\mathbb{R}^n$. This is because $\mathcal{L}_{M,N}(\boldsymbol{\theta})$ only involves products and linear combinations of elements of the vector $\boldsymbol{f_\theta}$ and differentiability is preserved under these operations. However, note that the product of two Lipschitz continuous functions is not necessarily Lipschitz continuous. Yet, Lipschitz continuity of the product holds if the two functions are bounded, in line with the requirements of Proposition 2.1.

Overall, Proposition 3 offers theoretical reassurance, because it shows that the bias can be controlled by the hyperparameter h. It also suggests to use a small value of h, in order to minimize this bias. However, in practice, an excessively small value for h might lead to accumulation of numerical errors, which may prevent convergence.

4. Numerical illustration

In this section, I illustrate the broad applicability of the gradient-free bc-MC operator using three different economic models. Firstly, I use the standard neoclassical stochastic growth model to illustrate the theoretical results from Section 3. I then solve a smooth large-scale OLG model, before solving a large-scale OLG model with non-differentiabilities. I use this last numerical experiment as an opportunity to compare the bc-MC operator and the PEA.

4.1. **The neoclassical stochastic growth model.**

Let us first consider a stylized example to illustrate the theoretical results from Section 3. Specifically, let us consider a special case of the neoclassical stochastic growth model developed by Brock and Mirman (1972), with full depreciation of capital and a logarithmic utility function, which admits a closed-form solution. A representative household maximizes her inter-temporal discounted utility

$$\max_{\{c_t\}_{t=0}^{\infty}} \mathbb{E}\Big[\sum_{t=0}^{\infty} \beta^t \log(c_t)\Big] \tag{15}$$

by choosing period t consumption c_t subject to the constraint $0 \leq c_t \leq y_t$, where y_t denotes the household's income in period t, with initial income y_0 given. Income in the

next period is stochastic and is defined as follows: $y_{t+1} = k_{t+1}^{\alpha}\eta_{t+1}$, with η denoting i.i.d. productivity shocks, and $k_{t+1} = y_t - c_t$ denoting the capital stock available at the beginning of period $t+1$. The productivity shocks are assumed to be lognormally distributed $\eta_t \equiv \exp(\nu_t)$, where ν is a zero-mean normally distributed random variable with variance σ_ν^2. For the problem to be well-posed, the discount factor β is positive but strictly less than one, while the parameter α, governing the marginal product of capital, is assumed to be strictly between 0 and 1. Under these conditions, the economic model is fully characterized by a single Euler equation that must hold every period:

$$\mathbb{E}_\nu \left[\beta \frac{c_t}{c_{t+1}} \alpha k_{t+1}^{\alpha-1} \exp(\nu_{t+1})\right] = 1 \tag{16}$$

Since the logarithmic utility function satisfies the Inada condition at zero ($\lim_{c\to 0} \frac{d}{dc}\log(c) = +\infty$), the optimal consumption function $c^*(y_t) : \mathbb{R}_+ \to \mathbb{R}_+$ is interior. As long as the function c^* is continuous on its domain, the transversality condition that accompanies the Euler condition is automatically met (Mitra and Roy, 2017). Here, we know that $c^*(y_t) = \phi^* y_t$, which is indeed a continuous function on its domain, with $\phi^* = (1 - \alpha\beta)$.

I now discuss how one may transform this economic problem into an optimization problem solvable with the gradient-free bc-MC operator. Let us use an MLP to approximate the consumption share: $c(y_t|\boldsymbol{\theta}) = \mathcal{MLP}(y_t|\boldsymbol{\theta})y_t$. In its general formulation, an MLP is a composition of N affine transformations, represented by matrices W_n, and $N-1$ activation functions σ_n:

$$\mathcal{MLP}(\boldsymbol{x}|\boldsymbol{\theta}) = \big(W_N \circ \sigma_{N-1} \circ W_{N-2} \circ \sigma_{N-3} \circ ... \circ W_1\big)(\boldsymbol{x}). \tag{17}$$

Hence, the parameter vector $\boldsymbol{\theta}$ can be defined as the values of the coefficients of the N matrices W_n. To ensure that the budget constraint is strictly respected, we can apply a max-min operator to the output of the final layer, to define a new function $\phi(y_t|\boldsymbol{\theta}) = \max\Big\{\tau, \min\big\{\mathcal{MLP}(y_t|\boldsymbol{\theta}), 1-\tau\big\}\Big\}$, such that $c(y_t|\boldsymbol{\theta}) = \phi(y_t|\boldsymbol{\theta})y_t$, with $\tau > 0$ a small positive real value.

Because the focus of this subsection is to illustrate the propositions from Section 3, I will make simplifying assumptions. Assume the MLP has a single node with a slope parameter $\phi_1 = 0$ and an intercept denoted by ϕ_0. While this is an extreme case of the general MLP architecture, it can be trained to find the true solution ($\phi_0^* = 1 - \alpha\beta$). With these assumptions and after simplifications, the Euler equation (16) can be written

as $\mathbb{E}_y\Big[\mathbb{E}_\nu\big[f(y,\nu|\boldsymbol{\theta})\big]^2\Big]$, with $f(y,\nu|\boldsymbol{\theta}) = \frac{\alpha\beta}{1-\max\{\tau,\min\{\phi_0,1-\tau\}\}} - 1$. The function f is Lipschitz continuous on $\mathbb{R}$, with a Lipschitz constant given by $\frac{\alpha\beta}{(1-\tau)^2}$. It is also bounded, with a maximum value given by $\frac{\alpha\beta}{\tau} - 1$ and a minimum value given by $\frac{\alpha\beta}{1-\tau} - 1$. Hence, the conditions for Proposition 2.1 apply, and the bc-MC loss function (13) is Lipschitz continuous. Because the function f does not depend on the realizations of the state and innovation variables y and ν, the bc-MC loss function can be expressed as the square of the function f: $\mathcal{L}_{M,N}(\phi_0) = \left(\frac{\alpha\beta}{1-\max\{\tau,\min\{\phi_0,1-\tau\}\}} - 1\right)^2$, which conveniently does not depend on the hyperparameters M and N.

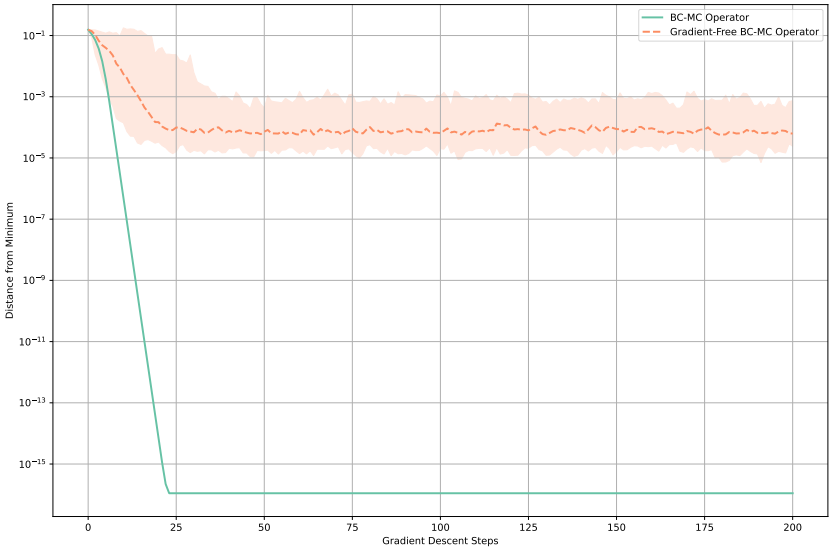
In order to illustrate Proposition 3, consider now the estimator of the gradient, given by $\nabla_\theta \mathcal{L}^F_{M,N}(\boldsymbol{\theta}) = \frac{\partial}{\partial\phi_0}\mathcal{L}^F_{M,N}(\phi_0) = \frac{v}{h}\Big(\big(\frac{\alpha\beta}{1-\max\{\tau,\min\{\phi_0+hv,1-\tau\}\}} - 1\big)^2 - \big(\frac{\alpha\beta}{1-\max\{\tau,\min\{\phi_0,1-\tau\}\}} - 1\big)^2\Big)$, where v is a standard normal variable. Assume that during training, ϕ_0 remains strictly within $(\tau, 1-\tau)$, so that $\mathcal{L}^F_{M,N}(\phi_0) = \frac{v}{h}\Big(\big(\frac{\alpha\beta}{1-\phi_0-hv} - 1\big)^2 - \big(\frac{\alpha\beta}{1-\phi_0} - 1\big)^2\Big)$, which is differentiable everywhere on this restricted domain. Straightforward calculations show that $\mathbb{E}_v\big[\lim_{h\to 0}\frac{\partial}{\partial\phi_0}\mathcal{L}^F_{M,N}(\phi_0)\big] = \mathbb{E}_v\big[\lim_{h\to 0} 2v^2\frac{\alpha\beta}{\big((1-\phi_0+hv)(1-\phi_0)\big)}\frac{\alpha\beta(1-\phi_0-hv/2)}{\big((1-\phi_0+hv)(1-\phi_0)\big)}\big]$, which is equal to $\frac{2\alpha\beta}{1-\phi_0}\big(\frac{\alpha\beta}{1-\phi_0} - 1\big) = \frac{\partial}{\partial\phi_0}\mathcal{L}(\phi_0)$, in line with Proposition 3.

I now numerically solve the model using the standard bc-MC operator and the gradient-free bc-MC operator. In the first case, I use backpropagation to calculate the gradient of the loss $\mathcal{L}_{M,N}(\phi_0)$. In the second case, I use the estimator $\frac{\partial}{\partial\phi_0}\mathcal{L}^F_{M,N}(\phi_0)$, as defined above. In terms of economic parameter values, I set the production parameter $\alpha = 0.36$ and the discount factor $\beta = 0.96$, which is a standard parametrization for RBC models at the yearly frequency. I use a Gaussian smoothing parameter $h = 1.10^{-4}$. I find the coefficient ϕ_0 by gradient descent, using two different starting points, $\phi_0 = 0.5$ and $\phi_0 = 0.0$. Results are presented in Figure 4.

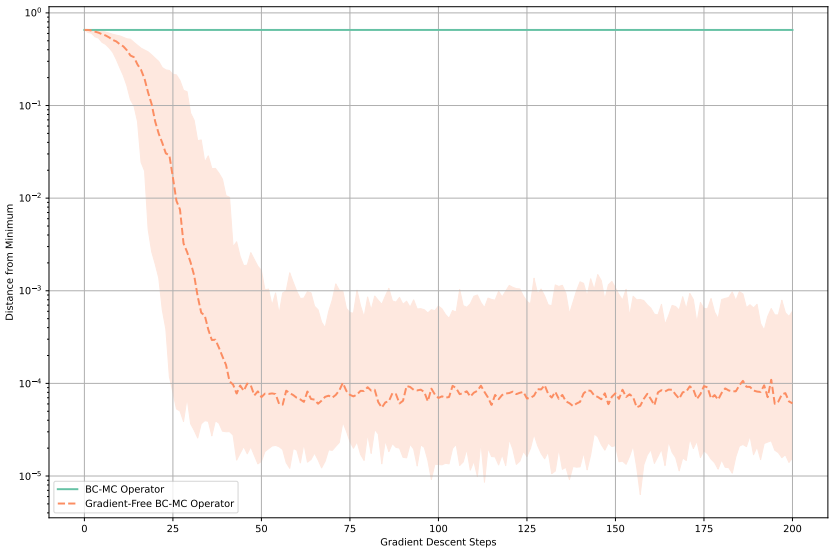
For $\phi_0 = 0.5$, as long as ϕ_0 remains within $(\tau, 1-\tau)$ during training, the gradient of the bc-MC loss function is continuously differentiable, and using $\frac{\partial}{\partial\phi_0}\mathcal{L}_{M,N}(\phi_0)$ rather than $\frac{\partial}{\partial\phi_0}\mathcal{L}^F_{M,N}(\phi_0)$ results in a more accurate solution, as indicated by the Euclidean distance from the true solution $\phi_0^* = 1 - \alpha\beta$ (see the top panel of Figure 4). For $\phi_0 = 0.0$, the gradient of the bc-MC loss function is equal to 0, which results in non-convergence for the gradient descent algorithm, as the algorithm converges to a flat region. The gradient-free bc-MC operator, which uses $\frac{\partial}{\partial\phi_0}\mathcal{L}^F_{M,N}(\phi_0)$, successfully escapes the flat region $(0, \tau)$ and ends up giving reasonably accurate results for ϕ_0 (see the bottom panel of Figure 4).

FIGURE 4. Trajectories using gradient-based and gradient-free bc-MC operators

(A) Distance from minimum ϕ_0^*, starting at $\phi_0 = 0.5$



(B) Distance from minimum ϕ_0^*, starting at $\phi_0 = 0$



Notes. This figure illustrates the use of the bc-MC operator and the gradient-free bc-MC operator to learn the policy function that satisfies the Euler equation (16). In the first case, the gradient of the bc-MC loss function, denoted by $\nabla_\theta \mathcal{L}_{M,N}(\boldsymbol{\theta})$, is used. Here, $\nabla_\theta \mathcal{L}_{M,N}(\boldsymbol{\theta})$ is a deterministic value, as it does not depend on the state or innovation vectors. In the second case, the gradient of the loss function is approximated using $\nabla_\theta \mathcal{L}^F_{M,N}(\boldsymbol{\theta})$ from equation (14). For the gradient-free bc-MC operator, 100 independent replications are used, to take into account the randomness of the random direction vector $\mathbf{v}$. Lines represent the Euclidean distance from the true value $\phi_0^* = 1 - \alpha\beta$. For the gradient-free bc-MC operator, shaded areas represent the P10-P90 inter-percentile range, while the orange dotted lines represent the median value across the 100 replications.

4.2. **A smooth OLG model.** I now solve a large-scale overlapping generation (OLG) model. This OLG model, which also appears in Krueger and Kubler (2004) and Azinovic, Gaegauf, and Scheidegger (2022), based on the work of Huffman (1987), admits a closed-form solution in some specific instances, which is convenient for verifying the accuracy of the gradient-free bc-MC operator. This setting is also interesting because it does not feature "approximate aggregation", which is a key requirement for alternative global methods, in particular the method of Krusell and Smith (1998).[7]

The model is populated by A households, each representing a different age group, and living for A periods. There is no uncertainty about lifetime. Households only work in the first period of their lives, receiving a competitive wage and saving in risky capital. Households cannot die with debt and start life with no assets. At time $t \in \{t', ..., t' + A - 1\}$, households born at time t' can be identified by their age $s = t - t' + 1 \in \{1, ..., A\}$. At each time period t, the representative household of age s chooses consumption c_t^s and savings a_t^s in order to maximize their remaining time-separable discounted expected lifetime utility given by

$$\mathbb{E}_t \sum_{h=0}^{A-s} \beta^h u(c_{t+h}^{s+h}) \tag{18}$$

subject to the budget constraint

$$c_t^s + a_t^s = r_t k_t^s + w_t l^s$$

where $\beta < 1$ denotes the discount factor and $u(c) = \frac{c^{1-\gamma}-1}{1-\gamma}$ is a CRRA utility function, with $\gamma > 0$ the coefficient of relative risk aversion. The variable $k_t^s = a_{t-1}^{s-1}$ denotes the available capital in the beginning of the period of age group s at time t, inherited from last period's savings, while l^s denotes the exogenously supplied labor endowment of age group s. By assumption, households only work in the first period of their life: $l^s = 1$ for $s = 0$ and $l^s = 0$ otherwise. Hence, aggregate labor supply $L_t = \sum_{h=1}^{A} l^s$ is constant and equal to one.

The production side of the economy is modelled as a single representative firm operating a Cobb-Douglas production function. The quantity of consumption good produced at

[7] "Approximate aggregation" holds when agents may use a very limited number of moments in order to accurately predict next period's prices. Here, this property does not hold in general, as shown in Krueger and Kubler (2004).

time t is given by

$$f(K_t, L_t, \eta_t, \delta_t) = \eta_t K_t^{\alpha} L_t^{1-\alpha} + K_t(1-\delta_t) \tag{19}$$

where η_t denotes the stochastic TFP and δ_t denotes the stochastic depreciation rate. The vector of exogenous stochastic variables $\boldsymbol{z_t} = (\eta_t, \delta_t)$ is assumed to satisfy the Markov property. The parameter α denotes the capital share in production, while K_t denotes aggregate capital with $K_t = \sum_{h=1}^{A} k_t^h$. The firm's optimization problem implies that the return on capital and the wage are given by

$$\begin{aligned} r_t &= \alpha \eta_t K_t^{\alpha-1} L_t^{1-\alpha} + (1-\delta_t) \\ w_t &= (1-\alpha)\eta_t K_t^{\alpha} L_t^{-\alpha} \end{aligned} \tag{20}$$

In this economy, given initial conditions $\boldsymbol{z_0}, \{k_0^s\}_{s=1}^{A}$, a competitive equilibrium is defined as a collection of choices for households $\{(c_t^s, a_t^s)_{s=1}^{A}\}_{t=0}^{\infty}$ and for the representative firm $(K_t, L_t)_{t=0}^{\infty}$ as well as prices $(r_t, w_t)_{t=0}^{\infty}$, such that

(1) given prices r_t and w_t, households optimize (18)

(2) given prices, the firm maximizes profits (19)

(3) markets clear

Because labor supply is exogenous, the labor market automatically clears. Market clearing for the capital market requires that $K_t = \sum_{h=1}^{A} a_{t-1}^{h-1}$. If the capital market clears, Walras' law implies that the good market clears.

In this economy, an equilibrium is characterized by $A-1$ Euler equations. More specifically, for $h \in \{1, ..., A-1\}$:

$$\mathbb{E}_{\eta,\delta}\left[\beta\left(\frac{c_{t+1}^{h+1}}{c_t^h}\right)^{-\gamma} r_{t+1} - 1\right] = 0 \tag{21}$$

While Krueger and Kubler (2004) use the vector of capital holdings $\boldsymbol{k_t} = (k_t^1, k_t^2, ..., k_t^A)$ and the vector of exogenous stochastic variables $\boldsymbol{z_t} = (\eta_t, \delta_t)$ as the state vector, one may also define the state vector as $(\boldsymbol{y_t}, \boldsymbol{z_t})$, where $\boldsymbol{y_t} = (y_t^1, y_t^2, ..., y_t^A)$ denotes the vector of total available wealth at the beginning of period t, with $y_t^h = r_t k_t^h + w_t l^h$. Indeed, when households choose how much to consume using a policy function of the form $c_t^h(\boldsymbol{y_t}, \boldsymbol{z_t})$, one obtains savings decision using the budget constraint $a_t^h(\boldsymbol{y_t}, \boldsymbol{z_t}) = y_t^h - c_t^h(\boldsymbol{y_t}, \boldsymbol{z_t})$. With known savings decisions, one obtains the aggregate capital level next period K_{t+1}, which gives the interest rate and the wage level using equation (20). With known r_{t+1} and w_{t+1}, the vector of available wealth at the beginning of period $t+1$ denoted by $\boldsymbol{y_{t+1}}$ is fully determined, because labor supply is exogenous. Without losing much generality,

we may assume that the exogenous stochastic variables are of the form $\eta_t = \bar{\eta} + \varepsilon_\eta$ and $\delta_t = \bar{\delta} + \varepsilon_\delta$, with ε_η and ε_δ two i.i.d. zero-mean normal variables with finite variances σ_η^2 and σ_δ^2. With this assumption, the relevant state vector is reduced to $\boldsymbol{y_t}$, as $\boldsymbol{z_t}$ has no persistence.

To transform this economic problem into an optimization problem solvable with the gradient-free bc-MC operator, I use a single MLP, parameterized by $\boldsymbol{\theta}$, to approximate the consumption share of all age groups at once: $\boldsymbol{c}(\boldsymbol{y_t}|\boldsymbol{\theta}) = \begin{pmatrix} \mathcal{MLP}(\boldsymbol{y_t}|\boldsymbol{\theta}) \\ 1 \end{pmatrix} \odot \boldsymbol{y_t}$. Here, $\boldsymbol{c_t}$ denotes the vector of consumption choices $(c_t^1(\boldsymbol{y_t}|\boldsymbol{\theta}), c_t^2(\boldsymbol{y_t}|\boldsymbol{\theta}), ..., c_t^A(\boldsymbol{y_t}|\boldsymbol{\theta}))$. The MLP is a function of the form (17), which here is a mapping from $\mathbb{R}^A$ to $\mathbb{R}^{A-1}$. The symbol $\odot$ is here used to denote the element-wise multiplication between two vectors of same size. Here, I have imposed that the last generation consumes all their available wealth.

In line with the general procedure presented in Section 3.2, solving the OLG model here consists in finding the MLP's parameter vector $\boldsymbol{\theta}$ such the $A-1$ stochastic equations $\mathbb{E}_{\boldsymbol{s}}\left[\mathbb{E}_{\boldsymbol{\varepsilon}}\left(f_k(\boldsymbol{s},\boldsymbol{\varepsilon}|\boldsymbol{\theta})\right)^2\right] = 0$ hold, where the state vector is $\boldsymbol{s} = \boldsymbol{y}$, the innovation vector $\boldsymbol{\varepsilon} = (\varepsilon_\eta, \varepsilon_\delta)$, and the residual functions are given by

$$f_k(\boldsymbol{s},\boldsymbol{\varepsilon}|\boldsymbol{\theta}) = \beta\big(\frac{c_{t+1}^{k+1}(\boldsymbol{y_{t+1}}|\boldsymbol{\theta})}{c_t^k(\boldsymbol{y_t}|\boldsymbol{\theta})}\big)^{-\gamma}\big(\alpha(\bar{\eta}+\varepsilon_\eta)\big(\sum_{h=1}^{A} y_t^h - c_t^k(\boldsymbol{y_t}|\boldsymbol{\theta})\big)^{\alpha-1} + (1-\bar{\delta}-\varepsilon_\delta)\big) - 1 \quad (22)$$

These $A-1$ stochastic equations can be expressed as single bc-MC loss function, taking the form of a quadratic form, as in equation (13).

In terms of parametrization, I closely follow Krueger and Kubler (2004) and Azinovic, Gaegauf, and Scheidegger (2022). The model is at the yearly frequency. I set the capital share parameter to $\alpha = 0.3$ and use a discount factor $\beta = 0.7$. For the productivity parameter, I use the normalization $\bar{\eta} = 1$ and set the standard deviation parameter to $\sigma_\eta = 0.05$. I set the mean yearly depreciation rate to $\delta = 0.1$ and use a standard deviation of $\sigma_\delta = 0.01$. I use a logarithmic utility function. In this case, Krueger and Kubler (2004) show that the optimal consumption functions are given by $c_t^h(\boldsymbol{y_t}, \boldsymbol{z_t}) = \big(1-\beta\frac{1-\beta^{A-h}}{1-\beta^{A-h+1}}\big)y_t^h$, or equivalently that the optimal savings functions are given by $a_t^h(\boldsymbol{y_t}, \boldsymbol{z_t}) = \beta\frac{1-\beta^{A-h}}{1-\beta^{A-h+1}}y_t^h$.

I use 20 different age groups, which correspond to the baseline case analyzed by the authors.

Regarding the architecture of the MLP, I use a MLP with 2 hidden layers with 25 nodes each. This results in $n = 1669$ parameters to be learnt during training, which is much more than necessary, since the analytical solution could be represented by a single layer MLP outputting 19 constant values given by $\left(1 - \beta \frac{1-\beta^{A-h}}{1-\beta^{A-h+1}}\right)$. However, I do not want to encode knowledge of the analytical solution within the MLP architecture, this is why I use this general architecture. To ensure that consumption remains positive, I apply a sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$ to the output of the final layer, which constrains the output to be between 0 and 1. To avoid numerical instability created by almost zero consumption or almost zero savings, I also apply a max-min operator, as in Section 4.1. After these transformations, the consumption function is approximated with $\boldsymbol{c}(\boldsymbol{y_t}|\boldsymbol{\theta}) = \begin{pmatrix} \max\{\tau, \min\{\sigma\left(\mathcal{MLP}(\boldsymbol{y_t}|\boldsymbol{\theta})\right), 1-\tau\}\} \\ 1 \end{pmatrix} \odot \boldsymbol{y_t}$, where τ is a small but strictly positive real number.

With this choice of MLP architecture and economic parameter values, I show in sections D - F of the Appendix that the conditions for Proposition 2 are met, provided some mild restrictions are imposed during training. This ensures that $\mathbb{E}_{\mathbf{v}}\left[\mathcal{L}_{M,N}(\boldsymbol{\theta} + h\mathbf{v})\right]$ is continuously differentiable, with gradient that can be estimated with equation (14). If one uses smoothed versions of the maximum and minimum operator appearing in $\boldsymbol{c}(\boldsymbol{y_t}|\boldsymbol{\theta})$, then Proposition 3 also applies, as discussed in Section G of the Appendix.

I now discuss some important implementation details regarding the gradient-free bc-MC operator. Firstly, one must sample from the ergodic distribution for the state vector $\mathbf{y}$, which itself depends on the unknown parameter vector $\boldsymbol{\theta}^*$ that minimizes the $A-1$ equations (21). Here, I follow the approach of Azinovic, Gaegauf, and Scheidegger (2022), which consists of simulating the model with the current parameter vector $\boldsymbol{\theta}_t$ and using the draws Ω_e to approximate the ergodic distribution for the state vector $\boldsymbol{s}$, by randomly sampling from Ω_e. The draw Ω_e are kept unchanged for a predetermined number of gradient descent steps. Conditional on Ω_e and on the current parameter vector $\boldsymbol{\theta}_t$, one can calculate an estimator for the gradient of the loss using equation (14) and update the value of the parameter vector using an update rule of the form (2). Secondly, the gradient-free bc-MC operator depends on two hyperparameter M and N, where M denotes the

number of independent draws for the state vector $\boldsymbol{s}$, while N denotes the number of independent draws for the innovation vector $\boldsymbol{\varepsilon}$ for each realization of $\boldsymbol{s}$. As in Pascal (2024), I choose the combination (M, N) so that the variance of the bc-MC loss function (13) is minimized, while keeping a computational budget constant. Because the bc-MC loss function is unbiased, this is equivalent to minimizing its mean squared error. For my computational budget, I use a value proportional to MN, which measures the number of times the function (22) is evaluated, each time the bc-MC loss function is calculated. In terms of other hyperparameters, I set the step size parameter $h = 1.10^{-5}$, I use the Adam minimization algorithm with a learning rate $\gamma = 1.10^{-3}$. I also use gradient clipping, as suggested by Proposition 2. A detailed description of this procedure is available in Section H of the Appendix.

Before discussing the results, I note that there exists a tight connection between usual accuracy metrics and the practical output of the bc-MC methodology. Typically, accuracy metrics are based on the unit-less Euler equation error (EEE) for age group h, denoted by $\mathcal{E}_h(\boldsymbol{s}|\boldsymbol{\theta})$. The square root of the mean squared Euler equation error (RMSE), denoted by $\mathbb{E}_{\boldsymbol{s}}\left[\mathcal{E}_h(\boldsymbol{s}|\boldsymbol{\theta})^2\right]^{0.5}$, measures the extent to which numerical approximations ($\boldsymbol{\theta} \neq \boldsymbol{\theta}^*$) leads to pricing errors (see for instance Lettau and Ludvigson, 2009). When the RMSE is equal to 1.10^{-3}, agents of age group h misallocate \$1 for each \$1000 they spend. An interesting feature of the bc-MC computational method, is that the square root of the loss function, which is calculated during training, can be used to calculate an estimator of the RMSE. To see that, consider the EEE for age group h:

$$\begin{aligned}\mathcal{E}_h(\boldsymbol{s}|\boldsymbol{\theta}) &= \frac{1}{c_t^h}(u')^{-1}\Big(\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\beta u'(c_{t+1}^{h+1})r_{t+1}\right]\Big) - 1 \\ &= \Big(\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\beta\big(\frac{c_{t+1}^{h+1}}{c_t^h}\big)^{-\gamma} r_{t+1}\right]\Big)^{\frac{-1}{\gamma}} - 1\end{aligned} \tag{23}$$

Using the approximation $x^\alpha \approx 1+\alpha(x-1)$ for $x \approx 1$, one gets the following approximation for the square of $\mathcal{E}_h(s|\boldsymbol{\theta})$:

$$\mathcal{E}_h(\boldsymbol{s}|\boldsymbol{\theta})^2 \approx \frac{1}{\gamma^2}\Big(\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\beta\big(\frac{c_{t+1}^{h+1}}{c_t^h}\big)^{-\gamma} r_{t+1}\right] - 1\Big)^2 \tag{24}$$

Hence, one obtains

$$\mathbb{E}_{\boldsymbol{s}}\left[\mathcal{E}_h(\boldsymbol{s}|\boldsymbol{\theta})^2\right]^{0.5} \approx \frac{1}{\gamma}\,\mathbb{E}_{\boldsymbol{s}}\left[\Big(\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\beta\big(\frac{c_{t+1}^{h+1}}{c_t^h}\big)^{-\gamma} r_{t+1} - 1\right]\Big)^2\right]^{0.5} \tag{25}$$

where the right-hand side of equation (25) can be written as $\frac{1}{\gamma}\mathbb{E}_{\boldsymbol{s}}\left[\mathbb{E}_{\boldsymbol{\varepsilon}}\left(f_h(\boldsymbol{s},\boldsymbol{\varepsilon}|\boldsymbol{\theta})\right)^2\right]^{0.5}$. By an application of Jensen's inequality, the average RMSE across age group is (approximately) bounded above by the square root of the bc-MC loss function, estimated using equation (13), scaled by the factor $\frac{1}{\gamma}$:

$$\begin{aligned}\frac{1}{A-1}\sum_{h=1}^{A-1}\mathbb{E}_{\boldsymbol{s}}\left[\mathcal{E}_h(\boldsymbol{s}|\boldsymbol{\theta})^2\right]^{0.5} &\approx \frac{1}{\gamma}\frac{1}{A-1}\sum_{h=1}^{A-1}\mathbb{E}_{\boldsymbol{s}}\left[\mathbb{E}_{\boldsymbol{\varepsilon}}\left(f_h(\boldsymbol{s},\boldsymbol{\varepsilon}|\boldsymbol{\theta})\right)^2\right]^{0.5} \\ &\leq \frac{1}{\gamma}\Big(\frac{1}{A-1}\sum_{h=1}^{A-1}\mathbb{E}_{\boldsymbol{s}}\left[\mathbb{E}_{\boldsymbol{\varepsilon}}\left(f_h(\boldsymbol{s},\boldsymbol{\varepsilon}|\boldsymbol{\theta})\right)^2\right]\Big)^{0.5} \\ &\approx \frac{1}{\gamma}\mathcal{L}_{M,N}(\boldsymbol{\theta})^{0.5}\end{aligned} \tag{26}$$

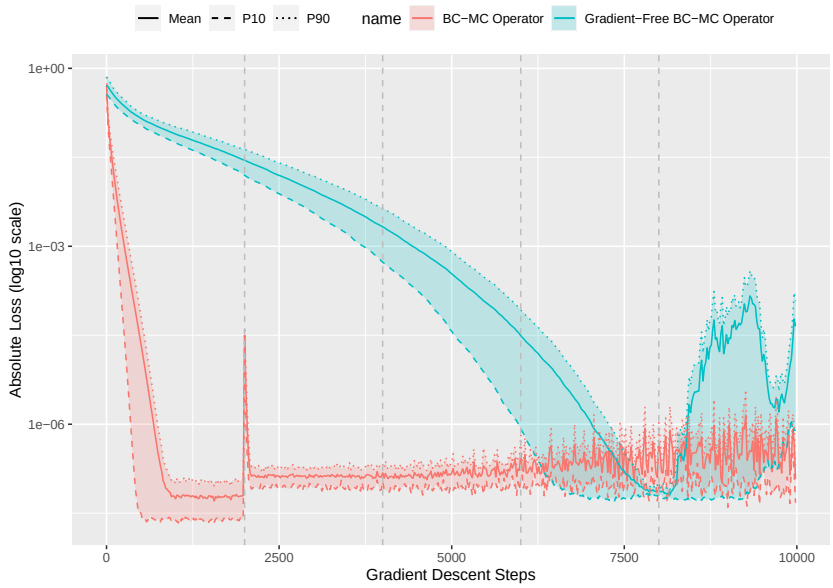
I now compare the gradient-free bc-MC operator, which uses equation (14) to approximate the gradient of the bc-MC loss function, to the bc-MC operator, which uses backpropagation to compute an exact value of the gradient via automatic differentiation, subject to floating-point precision limitations. For each method, I train 5 different neural networks, with different initial NN weights. I train the networks for a total of 10000 gradient descent steps, updating the guess for the ergodic distribution every 2000 steps. Figure 5 presents the results on time and accuracy for the two methods.

The top panel of Figure 5 indicates that the backpropagation-based bc-MC operator (in red) clearly outperforms in terms of accuracy the gradient-free bc-MC operator (in blue) in the first 6000 gradient descent steps. However, after this threshold, the gradient-free bc-MC operator leads to equally accurate solutions. The bottom panel shows that the gradient-free bc-MC operator is faster than its backpropagation-based counterpart. Overall, the backpropagation-based bc-MC operator achieves an average Euler equation error (EEE) of 1.62×10^{-4} for a total computing time of about 101 seconds, while its gradient-free counterpart yields an average EEE of 1.78×10^{-4} in about 93 seconds, as indicated in Table 1. For this particular example, taking into account both time

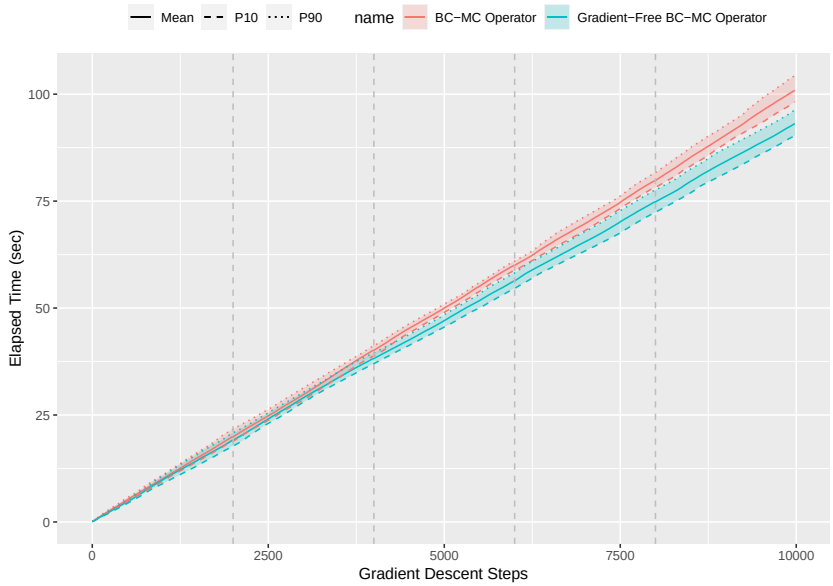
and accuracy, the gradient-free bc-MC operator outperforms its backpropagation-based counterpart.

FIGURE 5. Comparison of gradient-based and gradient-free bc-MC operators for the smooth OLG model

(A) Value loss function during training



(B) Required computing time



Notes. This figure compares the bc-MC operator and the gradient-free bc-MC operator in learning the policy function that satisfies the Euler equation (21). The first approach uses the gradient of the bc-MC loss function, denoted as $\nabla_\theta \mathcal{L}_{M,N}(\boldsymbol{\theta})$, while the second approximates the gradient using $\nabla_\theta \mathcal{L}^F_{M,N}(\boldsymbol{\theta})$ from equation (14). Solid lines represent average values from 5 independent replications, each with different initial NN weights. Shaded areas indicate the P10-P90 inter-percentile range. Panel A shows the loss function value relative to the number of gradient descent steps, and Panel B shows the computing time relative to the number of gradient descent steps. Vertical dotted lines mark the start of a new "episode", corresponding to a new estimation of the ergodic distribution.

TABLE 1. Comparison of gradient-based and gradient-free bc-MC operators for the smooth OLG model

Method	Value Loss	Mean EEE	P50 EEE	Max EEE	Comp. time
bc-MC	4.75×10^{-8}	1.62×10^{-4}	1.31×10^{-4}	1.33×10^{-3}	100.90 sec
gradient-free bc-MC	4.96×10^{-8}	1.78×10^{-4}	1.38×10^{-4}	1.49×10^{-3}	93.10 sec

Notes. This table compares the bc-MC operator and the gradient-free bc-MC operator in learning the policy function that satisfies the Euler equation (21). Descriptive statistics for each method are based on 5 independent replications, each with different initial NN weights. Each network is trained over 10,000 gradient descent steps, spanning a total of 5 "episodes". An "episode" corresponds to a new estimation of the ergodic distribution, where the current NN is used to simulate the model. The Euler equation error (EEE) is calculated using equation (23), with NN weights selected based on the smallest loss during training, once the episode number exceeds a pre-determined threshold. This threshold, set at 3 in this study, ensures that the state vector distribution has converged to its ergodic distribution.

4.3. **An OLG model with non-differentiabilities.** I now solve an OLG model with non-differentiabilities created by a borrowing constraint. The model is similar to the one presented in the previous section, except in two key aspects. First, I adapt the process of exogenous stochastic variables to be more in line with the RBC literature. In particular, the logarithm exogenous productivity η_t parameter is now assumed to follow an AR(1) process: $\log(\eta_{t+1}) = \rho \log(\eta_t) + \varepsilon_{\eta,t+1}$, where ε_η is an i.i.d. zero-mean normal variable with variance given by σ_η^2. I also now assume that the depreciation rate is deterministic, which can be interpreted as the limit case of the previous section with σ_δ tending to zero. As a result, the relevant state vector at time t is now $\boldsymbol{s} = (\boldsymbol{y_t}, \eta_t)$, where $\boldsymbol{y_t}$ still denotes the vector of total available wealth at the beginning of period t, while the relevant exogenous innovation vector is now the one-dimensional vector $\boldsymbol{\varepsilon} = (\varepsilon_{\eta,t+1})$.

The second key difference is the presence of a borrowing constraint for households: $a_t^h \geq 0, \forall h \in \{1, ..., A\}$. The optimization problem faced by households is now characterized by

the following system of Karush–Kuhn–Tucker (KKT) conditions, for $h \in \{1, 2, ..., A-1\}$:

$$\begin{aligned}
(c_t^h)^{-\gamma} &= \mathbb{E}_\varepsilon \left[\beta (c_{t+1}^{h+1})^{-\gamma} r_{t+1}\right] + \lambda_t^h \\
\lambda_t^h &\geq 0 \\
a_t^h &\geq 0 \\
\lambda_t^h a_t^h &= 0 \\
c_t^h + a_t^h &= r_t k_t^h + w_t l^h \equiv y_t^h
\end{aligned} \tag{27}$$

where λ_t^h are Lagrange multipliers associated with the inequality constraint on asset holdings.

To solve this model using neural networks, one approach could consist of using an MLP that outputs both c_t^h and λ_t^h and to take into the KKT constraints by adding some extra penalty function to the loss function, as for instance in Maliar and Maliar (2022) or Pascal (2024). With this approach, taking into account the different age groups, the MLP would be a mapping from from $\mathbb{R}^A$ to $\mathbb{R}^{2(A-1)}$.

Instead, I choose a more economical approach, based on insights from the literature on the Parameterized Expectation Algorithm (PEA) applied to models with inequality constraints, as described by Marcet and Lorenzoni (2001). The key idea from the PEA literature is that if one knows the conditional expectation $\mathbb{E}_\eta \left[\beta (c_{t+1}^{h+1})^{-\gamma} r_{t+1}\right]$ and if the Lagrange multiplier λ_t^h is equal to zero, then one can get current consumption c_t^h using the inverse of the marginal utility of consumption $(u')^{-1}(c_t) = c_t^{-\frac{1}{\gamma}}$. In practice, one may start by assuming that $\lambda_t^h = 0$ and calculate the consumption that this would imply $\tilde{c}_t^h$, which leads to an implied savings decision $\tilde{a}_t^h$ using the budget constraint. If $\tilde{a}_t^h > 0$, the budget constraint is not binding ($\lambda_t^h = 0$) and one may set $c_t^h = \tilde{c}_t^h$, $a_t^h = \tilde{a}_t^h$. If $\tilde{a}_t^h$ is negative, it means that the budget constraint is binding ($\lambda_t^h > 0$) and one gets a consumption implied by the budget constraint: $c_t^h = r_t k_t^s + w_t l^s$, $a_t^h = 0$.

In the PEA literature, this approach is generally implemented by using *linear model* to approximate the conditional expectation $\boldsymbol{s_t^T \theta^{(n)}} \approx \mathbb{E}_\eta \left[\beta (c_{t+1}^{h+1})^{-\gamma} r_{t+1}\right]$, where $\boldsymbol{s_t}$ denotes the value of the state vector at time t. By using the procedure discussed above, the current guess of the conditional expectation implies a series of actual realizations for the "target variable" inside the conditional expectation operator: $\{\phi_{t+1}\}_{t=1}^T = \{\beta (c_{t+1}^{h+1})^{-\gamma} r_{t+1}\}_{t=1}^T$. One may then update the guess for θ using the OLS solution of a linear regression of

$\boldsymbol{\phi_{t+1}}$ on S, to get an updated value for $\boldsymbol{\theta^{(n+1)}} = c(S'S)^{-1}S'\boldsymbol{\phi_{t+1}} + (1-c)\boldsymbol{\theta^{(n)}}$, where c is a coefficient between 1 and 0, while S is the data matrix build from T observations of the state vector $\boldsymbol{s_t}$.

I now explain how one may use a similar strategy with neural networks and the bc-MC operator $\mathcal{L}_{M,N}(\boldsymbol{\theta})$, which provides an unbiased estimator of $\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{s}}\left[\mathbb{E}_{\boldsymbol{\varepsilon}}\left(f(\boldsymbol{s},\boldsymbol{\varepsilon}|\boldsymbol{\theta})\right)^2\right]$. If one assumes that the bc-MC residual function f can be decomposed as $f(\boldsymbol{s},\boldsymbol{\varepsilon}|\boldsymbol{\theta}) = g(\boldsymbol{s},\boldsymbol{\varepsilon}|\boldsymbol{\theta}) - h(\boldsymbol{s}|\boldsymbol{\theta})$, the loss function becomes $\mathbb{E}_{\boldsymbol{s}}\left[\left[\mathbb{E}_{\boldsymbol{\varepsilon}}\left(g(\boldsymbol{s},\boldsymbol{\varepsilon}|\boldsymbol{\theta})\right) - h(\boldsymbol{s}|\boldsymbol{\theta})\right]^2\right]$. Clearly, this loss function is minimized when the function $h(\boldsymbol{s}|\boldsymbol{\theta})$ is equal to the conditional expectation $\mathbb{E}_{\boldsymbol{\varepsilon}}\left(g(\boldsymbol{s},\boldsymbol{\varepsilon}|\boldsymbol{\theta})\right)$. This has the flavor of the PEA, except that the function $h(.)$ is not restricted to being a linear function of the state vector $\boldsymbol{s}$. In particular, one may use a non-linear MLP to generates guesses for the conditional expectations $\mathbb{E}_{\eta}\left[\beta(c_{t+1}^{h+1})^{-\gamma}r_{t+1}\right]$, for $h \in \{1,2,...,A-1\}$. Then, one may proceed as previously described. That is, by first assuming that $\lambda_t^h = 0$, we get $\tilde{c}_t^h$ using $(u')^{-1}(c_t) = c_t^{-1/\gamma}$. If $\tilde{a}_t^h > 0$, and one may set $c_t^h = \tilde{c}_t^h$, $a_t^h = \tilde{a}_t^h$, and λ_t^h is indeed equal to 0. If $\tilde{a}_t^h$ is negative, $c_t^h = r_t k_t^s + w_t l^s$, $a_t^h = 0$, and $\lambda_t^h > 0$. These implied policy functions, combined with the transition process for η, lead to actual realizations for the function $g(\boldsymbol{s},\boldsymbol{\varepsilon}|\boldsymbol{\theta}) = \beta(c_{t+1}^{h+1})^{-\gamma}r_{t+1}$.

For this neural network based approach to work well in practice, I apply some transformations to the raw output of an MLP. The guess for the conditional expectation is given by $h(\boldsymbol{s}|\boldsymbol{\theta}) = \left[\begin{pmatrix} s\big(\mathcal{MLP}(\boldsymbol{y_t}|\boldsymbol{\theta})\big) \\ 1 \end{pmatrix} \odot \boldsymbol{y_t}\right]^{-\gamma}$, where $s(x) = \log(1+\exp(x)) > 0$ is the softplus function, which provides a smooth approximation to $\max(0,x)$. This particular functional form leads to an implied consumption choice of $\tilde{c}_t^h = s\big(\mathcal{MLP}_h(\boldsymbol{y_t}|\boldsymbol{\theta})\big)y_t^h$ for $h \neq A-1$, while the last generation consumes all their disposable income. The softplus function ensures that consumption remains strictly positive, while also allowing for the possibility that $\tilde{c}_t^h$ exceeds y_t^h, hence leading to $\tilde{a}_t^h$ being negative and for corner solutions to occur ($c_t^h = r_t k_t^s + w_t l^s$, $a_t^h = 0$, and $\lambda_t^h > 0$).

In this setting, Proposition 2 applies, but Proposition 3 does not. This follows from the presence of a borrowing constraint, expressed as $a_t^h = \max(\tilde{a}_t^h, 0)$. The function $x \to \max(x,0)$ has a kink at $x = 0$, making it non differentiable at this point. This non-smoothness propagates to the bc-MC loss function, implying that it does not have a derivative with respect to $\boldsymbol{\theta}$ on the set of points for which $\tilde{a}_t^h(\boldsymbol{\theta})$ is zero. However, the function $x \to \max(x,0)$ is Lipschitz continuous, with Lipschitz constant equal to

1. Provided that the MLP's parameter vector $\boldsymbol{\theta}$ lies within a compact set in $\mathbb{R}^n$, the bc-MC loss function is also Lipschitz continuous, because it involves a finite composition of Lipschitz mapping on a bounded set.[8]

As in the previous section, I compare the gradient-free bc-MC operator, which uses equation (14) to approximate the gradient of the bc-MC loss function, to the bc-MC operator, which uses backpropagation to compute a highly accurate gradient estimate (subject to floating-point precision limitations). For the gradient-based bc-MC operator implementation, I ignore the potential non-differentiable issues discussed above. In terms of economic parameters, I still set the capital share parameter to $\alpha = 0.3$ and use a discount factor $\beta = 0.7$. One notable difference from the previous section is that households now work for approximately two thirds of their life ($l^s = 1, \forall s \in \{1, ..., 14\}$), before retiring ($l^s = 0, \forall s \in \{15, ..., 20\}$). This creates an endogenous incentive for young households to borrow against future labor income. For the productivity parameter, I use a persistence parameter of 0.95 and a standard deviation parameter of $\sigma_\eta = 0.01$, which are common parameter values for models at the quarterly frequency. I set the quarterly depreciation rate to $\delta = 0.1$, which is higher than the usual value of 0.025, typically used for quarterly data. This ensures that the borrowing constraint binds for some households.

Regarding the architecture of the MLP, I still use a MLP with 2 hidden layers with 25 nodes each. For the parameter vector update rule, I use the Adam minimization algorithm with a learning rate $\gamma = 1.10^{-5}$. When using the gradient-free version of the bc-MC operator, I set the step size parameter $h = 1.10^{-5}$. For each method, I train 5 different neural networks. I train the networks for a total of 10000 gradient descent steps, updating the guess for the ergodic distribution every 2000 steps. Before the actual training phase, I pre-train the neural networks so that they reproduce the policy functions of the model linearized at first order, ignoring non-differentiabilities issues caused by the borrowing constraint. To obtain this approximate linearized model, I use Dynare and add equations $\min(\lambda_t^h, a^h) = 0$ to Dynare's .mod file, to approximately take into account

[8] One could derive a formal proof of this fact, along the lines of the arguments presented in Appendices D - F.

borrowing constraints. This provides a reasonable starting value for the parameter vector $\boldsymbol{\theta}$.

Regarding the bc-MC hyperparameters M and N, as in the previous section, I make sure that the value MN is constant during training, which ensures that the total number of functions evaluations for $f(\boldsymbol{s}, \boldsymbol{\varepsilon}|\boldsymbol{\theta})$ remains constant when evaluating the bc-MC loss. Here, I use $MN = 400$. However, numerical experiments indicate that the required computing time for the bc-MC loss function increases with N, even when MN constant. This is because N leads to a less sparse matrix Σ in equation (13). To ensure that computing time differences between gradient-based and gradient-free bc-MC operators are not driven by this mechanism, I use the same path for N in both instances.[9] In practice, I start with a value $N = 2$ in the early stages of learning, while increasing it up to $N = 20$ as learning progresses. This path resembles the optimal path for N, obtained by minimizing the variance of the loss function when using the gradient-based version of the bc-MC operator.

I now compare time and accuracy results obtained using the backpropagation-based bc-MC operator and the gradient-free bc-MC operator. To measure accuracy, I still the Euler equation error (EEE) based on equation, but modified to take into account that when borrowing constraint binds, the household consumes all her available cash on hand:

$$\mathcal{E}_h(\boldsymbol{s}|\boldsymbol{\theta}) = \frac{1}{c_t^h}(u')^{-1}\Big[\max\Big(\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\beta u'(c_{t+1}^{h+1})r_{t+1}\right], u'(y_t^h)\Big)\Big] - 1 \tag{28}$$

Results regarding time and accuracy for the two methods are presented in Figure 5 and Table 2. Figure 5 shows that while training based on the backpropagation-based bc-MC operator leads to initially lower values for the loss function, the gradient-free bc-MC operator ends up yielding similar results. Overall, Table 2 shows that while the backpropagation-based bc-MC operator is slightingly more accurate, it is slower than its gradient-free counterpart.

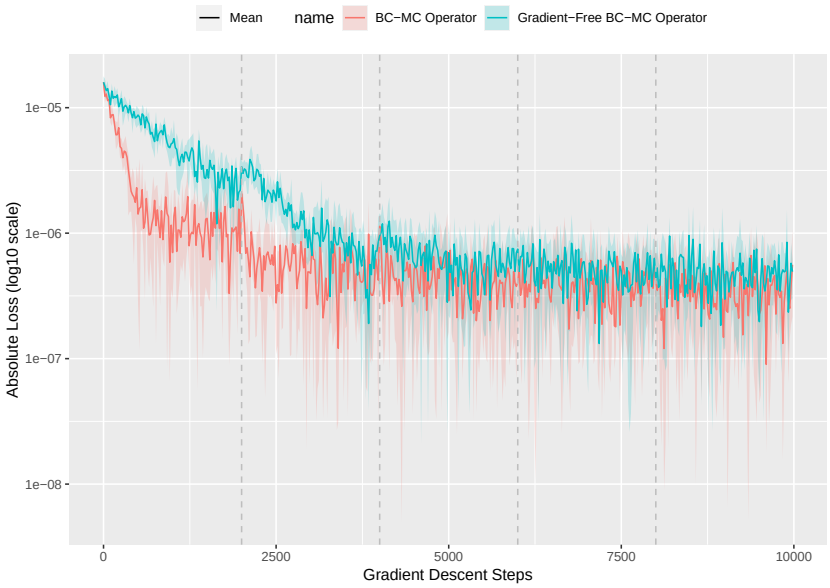
Table 3 indicates that the backpropagation-based bc-MC operator and the gradient-free counterpart yield similar descriptive statistics for key aggregate variables. In both cases, they (marginally) lead to an increase in aggregate capital accumulation, because of precautionary savings motives, compared to the first order linear approximation. As

[9]Results from Section 4.2 are not affected by this mechanism, since $N = N^* = 2$ during training for both methodologies.

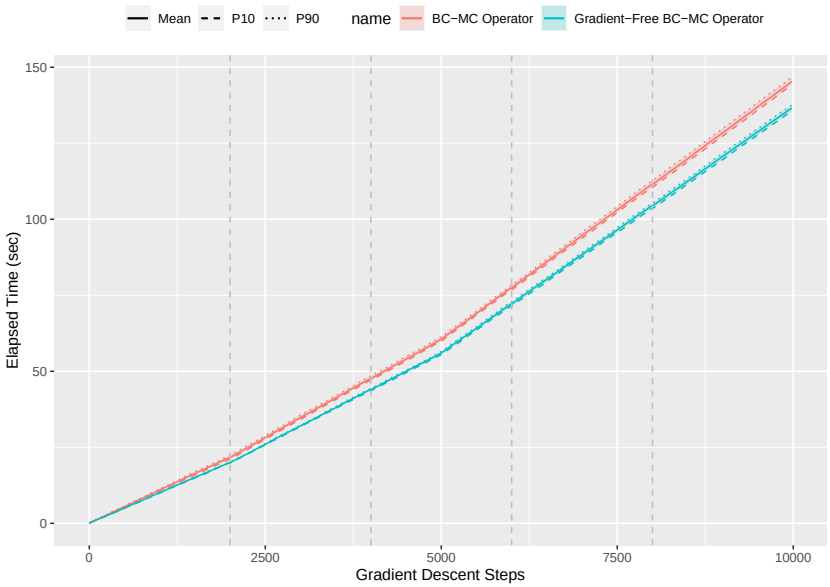
a result, aggregate production is marginally higher and the interest rate lower. These global solution methods lead to positive skewness in simulated series, which is absent from the first order solution, as linearized models are symmetric around the non-stochastic steady-state.

FIGURE 6. Comparison of gradient-based and gradient-free bc-MC operators for the OLG model with borrowing constraint

(A) Value loss function during training



(B) Required computing time



Notes. This figure compares the bc-MC operator and the gradient-free bc-MC operator in learning the policy function that solves the non-smooth OLG model. The first approach uses the gradient of the bc-MC loss function, denoted as $\nabla_\theta \mathcal{L}M, N(\boldsymbol{\theta})$, while the second approximates the gradient using $\nabla_\theta \mathcal{L}^F_{M,N}(\boldsymbol{\theta})$ from equation (14). Solid lines represent average values from 5 independent replications. Shaded areas indicate the P10-P90 inter-percentile range. Panel A shows the loss function value relative to the number of gradient descent steps, and Panel B shows the computing time relative to the number of gradient descent steps. Vertical dotted lines mark the start of a new "episode", corresponding to a new estimation of the ergodic distribution.

TABLE 2. Comparison of gradient-based and gradient-free bc-MC operators for the OLG model with borrowing constraint

Method	Value Loss	Mean EEE	P50 EEE	Max EEE	Comp. time
bc-MC	2.06×10^{-9}	3.15×10^{-4}	1.22×10^{-4}	4.83×10^{-3}	145.27 sec
gradient-free bc-MC	3.59×10^{-9}	3.78×10^{-4}	1.74×10^{-4}	5.13×10^{-3}	136.56 sec

Notes. This table compares the bc-MC operator and the gradient-free bc-MC operator in learning the policy function that solves the non-smooth OLG model. Descriptive statistics for each method are based on 5 independent replications, each with different initial NN weights. Each network is trained over 10,000 gradient descent steps, spanning a total of 5 "episodes". An "episode" corresponds to a new estimation of the ergodic distribution, where the current NN is used to simulate the model. The Euler equation error (EEE) is calculated using equation (28), with NN weights selected based on the smallest loss during training, once the episode number exceeds a pre-determined threshold. This threshold, set at 3 in this study, ensures that the state vector distribution has converged to its ergodic distribution.

TABLE 3. Descriptive statistics

Variable	Statistic	Linearized	bc-MC	gf-bc-MC
Aggregate capital K	Mean	6.47	6.48	6.48
	Std. Dev.	0.30	0.29	0.29
	Skewness	0.02	0.15	0.16
Aggregate output Y	Mean	11.11	11.13	11.12
	Std. Dev.	0.50	0.50	0.50
	Skewness	0.01	0.16	0.16
Interest rate r	Mean	1.42	1.41	1.41
	Std. Dev.	0.01	0.01	0.01
	Skewness	0.00	0.04	0.02
Wage w	Mean	0.56	0.56	0.56
	Std. Dev.	0.02	0.02	0.02
	Skewness	0.01	0.16	0.16

Notes. This table compares descriptive statistics for the life-cycle dynamics OLG model with borrowing constraint, obtained using three different solution methods. The first column is for the model linearized at first order, using Dynare, ignoring non-differentiabilities issues caused by the borrowing constraint. To obtain this approximate linearized model, I use Dynare and add equations $\min(\lambda_t^h, a^h) = 0$ to Dynare's .mod file. The two other columns are for the bc-MC operator and the gradient-free bc-MC operator ("gf-bc-MC"). Calculations are based on 100,000 periods..

4.4. **Connection with the Parameterized Expectation Algorithm.** The previous section suggests that the bc-MC operator and the Parameterized Expectation Algorithm (PEA) are closely related. In this section, I clarify these links, and show that the former can be interpreted as a generalization of the latter. In particular, I show that when a parameterized expectations approach is used with the bc-MC operator, the gradient of the bc-MC loss function and the gradient of PEA's loss function are equal in a neighborhood of a solution for the parameter $\boldsymbol{\theta}$.

For simplicity, consider an economic model characterized by a single functional equation and let us set the bc-MC hyperparameter $N = 2$, which implies using two independent innovation vectors ($\boldsymbol{\varepsilon_m^1}$ and $\boldsymbol{\varepsilon_m^2}$) for each realization of the state vector $\boldsymbol{s_m}$. In that case,

the bc-MC loss function is given by:

$$\mathcal{L}_{M,2}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{m=1}^{M} f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^1}|\boldsymbol{\theta}) f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^2}|\boldsymbol{\theta}) \tag{29}$$

As in the previous section, let us assume that the function f can be decomposed as $f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta}) = g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta}) - h(\boldsymbol{s_m}|\boldsymbol{\theta})$, which implies that when the loss function (29) is minimized at $\boldsymbol{\theta^*}$, $h(\boldsymbol{s_m}|\boldsymbol{\theta^*})$ is an unbiased predictor of the conditional mean $\mathbb{E}_{\boldsymbol{\varepsilon}}\left(g(\boldsymbol{s_m}, \boldsymbol{\varepsilon}|\boldsymbol{\theta^*})\right)$. If the function $h(.)$ is assumed to be a linear function of the state vector, minimizing equation (29) amounts to finding a linear function $\boldsymbol{s_m^T\theta}$ that best predicts (in the mean squared sense) the value of $g(\boldsymbol{s_m}, \boldsymbol{\varepsilon}|\boldsymbol{\theta})$. This closely resembles the description of the PEA, as previously mentioned.

However, there exist two differences with the PEA. To see that, first consider the gradient of the loss function (29) with respect to the parameter vector $\boldsymbol{\theta}$, assuming that the function $h(.)$ is a linear function of the state vector:

$$\begin{aligned}\nabla\mathcal{L}_{M,2}(\boldsymbol{\theta}) = \frac{-2}{M}\sum_{m=1}^{M} \boldsymbol{s_m}\Big[\frac{g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^1}|\boldsymbol{\theta}) + g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^2}|\boldsymbol{\theta})}{2} - \boldsymbol{s_m^T\theta}\Big] \\ + \frac{1}{M}\sum_{m=1}^{M} \nabla g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^1}|\boldsymbol{\theta})\Big[g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^2}|\boldsymbol{\theta}) - \boldsymbol{s_m^T\theta}\Big] + \nabla g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^2}|\boldsymbol{\theta})\Big[g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^1}|\boldsymbol{\theta}) - \boldsymbol{s_m^T\theta}\Big]\end{aligned} \tag{30}$$

The first line of equation (30) corresponds to the gradient of the *OLS minimization problem* of regressing the average value $\bar{g}(\boldsymbol{s_m}|\boldsymbol{\theta}) \equiv \frac{g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^1}|\boldsymbol{\theta}) + g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^2}|\boldsymbol{\theta})}{2}$ on $\boldsymbol{s_m}$. Thus, if one were to ignore the second line of equation (30), the bc-MC operator, when combined with a linear model ($h(\boldsymbol{s_m^T}|\boldsymbol{\theta}) = \boldsymbol{s_m^T\theta}$), is equivalent to using the PEA where the depend variable in the OLS regression is an *average* of *two realizations* of a "target variable" one aims to predict. These two realizations are based on two independent innovation shocks for each value of the state vector $\boldsymbol{s_m}$. Instead, the PEA uses a *single realization* $g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m}|\boldsymbol{\theta})$ for each draw of the state vector $\boldsymbol{s_m}$, leading to the following expression for the gradient of the PEA's loss function:

$$\nabla\mathcal{L}_M^{\text{PEA}}(\boldsymbol{\theta}) = \frac{-2}{M}\sum_{m=1}^{M} \boldsymbol{s_m}\Big[g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m}|\boldsymbol{\theta}) - \boldsymbol{s_m^T\theta}\Big] \tag{31}$$

Since $\bar{g}(\boldsymbol{s_m}|\boldsymbol{\theta})$ and $g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m}|\boldsymbol{\theta})$ are estimates of the sample mean estimator, both of them provide unbiased estimates for the true (unobservable) target $\mathbb{E}_{\varepsilon}\left[g(\boldsymbol{s_m}, \boldsymbol{\varepsilon}|\boldsymbol{\theta})\right]$. Yet,

because the variance of the sample mean is inversely proportional to the number of observations, the variance of the former is half that of the latter. This matters, because variance-reduction schemes have been shown to improve convergence speed in gradient-descent-based minimization problems(Hofmann et al., 2015).

The second line of equation (30) constitutes the second difference between the (gradient-free) bc-MC operator and the PEA. The bc-MC operator accounts for the target being obtained via simulation and as a result also depends on the current value of the parameter vector $\boldsymbol{\theta}$. When using the PEA, this dependence is simply ignored.

However, the omission of this feedback effect of $\boldsymbol{\theta}$ on the value of the gradient of the loss can be interpreted as an asymptotically valid approximation. To see that, one may note that the second line of equation (30) is equal to zero when $\boldsymbol{\theta}$ is equal to its solution $\boldsymbol{\theta^*}$. Indeed, let us assume that the conditional expectation is indeed described by a linear function $h(\boldsymbol{s_m^T}|\boldsymbol{\theta^*}) = \boldsymbol{s_m^T}\boldsymbol{\theta^*}$. In that case, the gradient of $\nabla g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^1}|\boldsymbol{\theta})$ evaluated at $\boldsymbol{\theta^*}$ is equal to $\boldsymbol{s_m}$. Thus, the first term appearing in the second line is equal to the sample covariance between regressors ($\boldsymbol{s_m}$) and OLS residuals ($g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^2}|\boldsymbol{\theta}) - \boldsymbol{s_m^T}\boldsymbol{\theta}$), which is equal to zero, by construction of the OLS estimator. By the same token, the second term appearing in the second line of equation (30) is also equal to zero. This observation justifies the use of stochastic gradient descent, or its variants, with the *approximate gradient* only given by the *first line of equation* (30). If a solution $\boldsymbol{\theta^*}$ is found using the approximate gradient, it is also a solution of the *true gradient* $\nabla \mathcal{L}_{M,2}(\boldsymbol{\theta})$.

The results of this section, based on the case $N = 2$, can be extended to the general case ($N \geq 2$). Instead of using two draws to calculate $\bar{g}(\boldsymbol{s_m}|\boldsymbol{\theta})$, one uses N independent draws $\bar{g}(\boldsymbol{s_m}|\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta})$ in equation (30), as explained in Appendix J. Hence, if one uses a linear model for the function $h(.)$, one may use the following asymptotically accurate approximation to gradient of the bc-MC operator:

$$\nabla \mathcal{L}^{\text{PEA}}_{M,N}(\boldsymbol{\theta}) = \frac{-2}{M}\sum_{m=1}^{M} \boldsymbol{s_m}\Big[\underbrace{\frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta})}_{\equiv \bar{g}(\boldsymbol{s_m}|\boldsymbol{\theta})} - \boldsymbol{s_m^T}\boldsymbol{\theta}\Big] \tag{32}$$

This "bc-MC-PEA" operator constitutes an alternative to the gradient-free bc-MC estimator based on randomized directional derivative estimators. In particular, one may simply replace equation (14) with equation (32) in the general N^*-algorithm described in Appendix H. This procedure based on the bc-MC-PEA operator requires only MN

forward evaluations of the function $f(.)$, with no need for using the backpropagation algorithm. It can be interpreted as an extension to the PEA, because setting $N = 1$ results in the usual PEA, described by equation (31).

This methodology based on the bc-MC-PEA operator is less general than the one based on the gradient-free bc-MC operator, because it requires (i) feasibility of a parameterized expectation approach and (ii) *linearity* of the conditional expectation function $h(.)$. On the one hand, feasibility of a parameterized expectation approach might be restrictive for some applications. In the present context, this is the assumption that the function f can be decomposed as $f(\boldsymbol{s}, \boldsymbol{\varepsilon}|\boldsymbol{\theta}) = g(\boldsymbol{s}, \boldsymbol{\varepsilon}|\boldsymbol{\theta}) - h(\boldsymbol{s}|\boldsymbol{\theta})$. For example, such a decomposition does not exist with the recursive preferences of Epstein and Zin (1989). However, the general methodology described in Section 3 does not rely on this assumption and can accommodate Epstein-Zin preferences.[10]

On the other hand, linearity of $h(.)$ might not be too restrictive in practical applications, as one may use transformations of the state vector, denoted by $b(\boldsymbol{s_m})$, for which the approximation $h(b(\boldsymbol{s_m})^T|\boldsymbol{\theta}) \approx b(\boldsymbol{s_m})^T\boldsymbol{\theta}$ is a better fit. In the PEA literature, this is generally achieved by using power or logarithmic transformations of the state vector, and/or by creating interaction terms between elements of the state vector. However, one drawback of this approach is that it requires taking a stance on the appropriate form for the relevant features of the data. This is not the case when using MLPs, as they have the capacity to automatically detect the relevant features of the data. Additionally, one may also have to deal with (near) multicollinearity when using the transformed state vector $b(\boldsymbol{s_m})$. These drawbacks may be offset by a reduction in required computing time, as the gradient-free bc-MC operator requires $2MN$ evaluations of the function $f(.)$, twice as many as those required for the bc-MC-PEA operator described here. Because it is not the main focus of this paper, I leave a precise numerical investigation of the bc-MC-PEA operator for future research.

[10] Azinovic, Gaegauf, and Scheidegger, 2022 solve a Bewley (1977) model with aggregate uncertainty and Epstein-Zin preferences using neural networks. An MLP jointly approximate value and policy functions.

5. Conclusion

In this paper, I develop a new methodology for finding global solutions of high-dimensional economic models using neural networks without requiring the backpropagation algorithm. This methodology is based on the gradient-free bias-corrected Monte Carlo (bc-MC) operator, which extends the work of Pascal (2024) to the non-smooth case. The proposed methodology does not require exact computation of the gradient of the loss function characterizing the solution of an economic model. Instead, gradients of the loss function are approximated using directional derivatives along random directions drawn from a standard normal distribution. Directional derivatives can be easily calculated, even in high-dimensional settings, which is often the case when neural networks are involved. The resulting gradient-free bc-MC operator can be used to solve economic models using neural networks when automatic differentiation software is unavailable or not feasible due to non-differentiabilities. This method is also well-suited for situations where memory limitations prevent the use of backpropagation, which requires storing all intermediate operations.

In this paper, I examine key theoretical properties of the gradient-free bc-MC operator. In particular, I demonstrate that, under certain conditions stated in the paper, the gradient-free bc-MC operator is asymptotically an unbiased estimator of the gradient of the loss function. I also demonstrate the broad applicability of this methodology by solving large-scale overlapping generation (OLG) models with aggregate uncertainty, including scenarios involving borrowing constraints that introduce non-differentiabilities in households' optimization problems. When solving the OLG model with occasionally binding constraints (OBCs), I use an efficient scheme that treats Lagrange multipliers as residual variables that can be adjusted ex post, as suggested in earlier applications of the Parameterized Expectation Algorithm (PEA). In fact, I also observe a strong connection between the bc-MC operator methodology and the PEA. In a sense, the latter is a special case of the former.

To conclude, I note that the method presented in this paper is quite general, in the sense that it does not constrain the type of neural network architecture that can be used to approximate policy or value functions. While using multilayer perceptrons (MLPs) has been the norm in economic applications, alternative architectures may be used. It would be particularly interesting to combine the present methodology with Kolmogorov–Arnold

Networks (Liu et al., 2024), recently proposed as a promising alternative to MLPs in machine learning. Further research is needed to investigate the properties of the "bc-MC-PEA" methodology proposed in this paper.

References

Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Adjemian, Stéphane et al. (Apr. 2011). *Dynare: Reference Manual Version 4*. Dynare Working Papers 1. CEPREMAP. URL: https://ideas.repec.org/p/cpm/dynare/001.html.

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

Baydin, Atılım Güneş et al. (2022). "Gradients without backpropagation". In: *arXiv preprint arXiv:2202.08587*.

Beck, Pierre et al. (2024). "Deep learning solutions of DSGE models: a technical report". In: *BCL Working Papers n°184*.

Berahas, Albert S et al. (2022). "A theoretical and empirical comparison of gradient approximations in derivative-free optimization". In: *Foundations of Computational Mathematics* 22.2, pp. 507–560.

Bewley, Truman (1977). "The permanent income hypothesis: A theoretical formulation". In: *Journal of Economic Theory* 16.2, pp. 252–292. ISSN: 0022-0531. DOI: https://doi.org/10.1016/0022-0531(77)90009-6. URL: https://www.sciencedirect.com/science/article/pii/0022053177900096.

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.

Christiano, Lawrence J and Jonas DM Fisher (2000). "Algorithms for solving dynamic models with occasionally binding constraints". In: *Journal of Economic Dynamics and Control* 24.8, pp. 1179–1232.

Den Haan, Wouter J. and Albert Marcet (1990). "Solving the Stochastic Growth Model by Parameterizing Expectations". In: *Journal of Business and Economic Statistics* 8.1,

pp. 31–34. ISSN: 07350015. URL: http://www.jstor.org/stable/1391746 (visited on 06/13/2024).

Duarte, Victor, Diogo Duarte, and Dejanir H. Silva (2024). *Machine Learning for Continuous-Time Finance*. CESifo Working Paper Series 10909. CESifo. URL: https://ideas.repec.org/p/ces/ceswps/_10909.html.

Duffy, John and Paul D McNelis (2001). "Approximating and simulating the stochastic growth model: Parameterized expectations, neural networks, and the genetic algorithm". In: *Journal of Economic Dynamics and Control* 25.9, pp. 1273–1303. ISSN: 0165-1889. DOI: https://doi.org/10.1016/S0165-1889(99)00077-9. URL: https://www.sciencedirect.com/science/article/pii/S0165188999000779.

Epstein, Larry G. and Stanley E. Zin (1989). "Substitution, Risk Aversion, and the Temporal Behavior of Consumption and Asset Returns: A Theoretical Framework". In: *Econometrica* 57.4, pp. 937–969. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1913778 (visited on 01/27/2025).

Farhi, Emmanuel and Iván Werning (2017). "Fiscal Unions". In: *American Economic Review* 107.12, 3788–3834. DOI: 10.1257/aer.20130817. URL: https://www.aeaweb.org/articles?id=10.1257/aer.20130817.

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.

Fernández-Villaverde, Jesús, Samuel Hurtado, and Galo Nuño (2023). "Financial Frictions and the Wealth Distribution". In: *Econometrica* 91.3, pp. 869–901. DOI: https://doi.org/10.3982/ECTA18180. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.3982/ECTA18180. URL: https://onlinelibrary.wiley.com/doi/abs/10.3982/ECTA18180.

Fernández-Villaverde, Jesús, Galo Nuño, and Jesse Perla (2024). *Taming the Curse of Dimensionality: Quantitative Economics with Deep Learning*. Working Paper 33117. National Bureau of Economic Research. DOI: 10.3386/w33117. URL: http://www.nber.org/papers/w33117.

Folini, Doris et al. (Jan. 2024). "The Climate in Climate Economics". In: *The Review of Economic Studies*, rdae011. ISSN: 0034-6527. DOI: 10.1093/restud/rdae011. eprint: https://academic.oup.com/restud/advance-article-pdf/doi/10.1093/restud

/rdae011/56663801/rdae011.pdf. URL: https://doi.org/10.1093/restud/rdae011.

Hofmann, Thomas et al. (2015). "Variance Reduced Stochastic Gradient Descent with Neighbors". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/effc299a1addb07e7089f9b269c31f2f-Paper.pdf.

Huffman, Gregory W. (1987). "A Dynamic Equilibrium Model of Asset Prices and Transaction Volume". In: *Journal of Political Economy* 95.1, pp. 138–159. ISSN: 00223808, 1537534X. URL: http://www.jstor.org/stable/1831303 (visited on 06/24/2024).

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.

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.

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.

Lettau, Martin and Sydney C. Ludvigson (2009). "Euler equation errors". In: *Review of Economic Dynamics* 12.2, pp. 255–283. ISSN: 1094-2025. DOI: https://doi.org/10.1016/j.red.2008.11.004. URL: https://www.sciencedirect.com/science/article/pii/S1094202508000616.

Liu, Ziming et al. (2024). "Kan: Kolmogorov-arnold networks". In: *arXiv preprint arXiv:2404.19756*.

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

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

Marcet, Albert and Guido Lorenzoni (Oct. 2001). "The Parameterized Expectations Approach: Some Practical Issues". In: *Computational Methods for the Study of Dynamic Economies*. Oxford University Press. ISBN: 9780199248278. DOI: 10.1093/0199248273.003.0007. eprint: https://academic.oup.com/book/0/chapter/193307872

/chapter-ag-pdf/44625550/book_25747_section_193307872.ag.pdf. URL: https://doi.org/10.1093/0199248273.003.0007.

Marchiori, Luca and Olivier Pierrard (2015). *LOLA 3.0: Luxembourg OverLapping generation model for policy Analysis: Introduction of a financial sector in LOLA*. Banque centrale du Luxembourg.

Mitra, Tapan and Santanu Roy (2017). "Optimality of Ramsey–Euler policy in the stochastic growth model". In: *Journal of Economic Theory* 172, pp. 1–25. ISSN: 0022-0531. DOI: https://doi.org/10.1016/j.jet.2017.08.001. URL: https://www.sciencedirect.com/science/article/pii/S0022053117300790.

Nesterov, Yu (2005). "Lexicographic differentiation of nonsmooth functions". In: *Mathematical programming* 104, pp. 669–700.

Nesterov, Yurii and Vladimir Spokoiny (2017). "Random gradient-free minimization of convex functions". In: *Foundations of Computational Mathematics* 17.2, pp. 527–566.

Pascal, Julien (2024). "Artificial neural networks to solve dynamic programming problems: A bias-corrected Monte Carlo operator". In: *Journal of Economic Dynamics and Control* 162, p. 104853. ISSN: 0165-1889. DOI: https://doi.org/10.1016/j.jedc.2024.104853. URL: https://www.sciencedirect.com/science/article/pii/S0165188924000459.

Paszke, Adam et al. (2017). "Automatic differentiation in PyTorch". In: *NIPS-W*.

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. ISSN: 1877-7503. DOI: https://doi.org/10.1016/j.jocs.2019.03.004. URL: https://www.sciencedirect.com/science/article/pii/S1877750318306161.

Scheinberg, Katya (2022). "Finite difference gradient approximation: To randomize or not?" In: *INFORMS Journal on Computing* 34.5, pp. 2384–2388.

Schmitt-Grohé, Stephanie and Martın Uribe (2004). "Solving dynamic general equilibrium models using a second-order approximation to the policy function". In: *Journal of Economic Dynamics and Control* 28.4, pp. 755–775.

Silver, David et al. (2022). "Learning by Directional Gradient Descent". In: *International Conference on Learning Representations*. URL: https://api.semanticscholar.org/CorpusID:251649228.

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.

Valaitis, Vytautas and Alessandro T Villa (2024). "A machine learning projection method for macro-finance models". In: *Quantitative economics* 15.1, pp. 145–173.

Zhang, Jingzhao et al. (2020). "Why Gradient Clipping Accelerates Training: A Theoretical Justification for Adaptivity". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=BJgnXpVYwS.

Appendix

Appendix A. Proof proposition 1

Let us consider the vector-valued function $g : \mathbb{R}^n \to \mathbb{R}^n$ defined as:

$$g(\boldsymbol{\theta}) = D_{\mathbf{v}} f(\boldsymbol{\theta})\mathbf{v}$$

where $\mathbf{v}$ is a normally distributed random vector in $\mathbb{R}^n$ with zero mean and a variance-covariance matrix given by the identity matrix $\mathbb{I}_n$.

Let us fix the value of $\boldsymbol{\theta}$ in $\mathbb{R}^n$. Because f is differentiable at $\boldsymbol{\theta}$ by assumption, one can express the directional derivative as a dot product between the gradient and the vector $\mathbf{v}$, as in equation (5):

$$g(\boldsymbol{\theta}) = D_{\mathbf{v}} f(\boldsymbol{\theta})\mathbf{v} = (\nabla f(\boldsymbol{\theta}) \cdot \mathbf{v})\mathbf{v}$$

Developing the formula $\nabla f(\boldsymbol{\theta}) \cdot \mathbf{v}$ and taking the expectation with respect to the random vector $\mathbf{v}$ yields:

$$\mathbb{E}\left[g(\boldsymbol{\theta})\right] = \mathbb{E}\left[\left(\sum_{i=1}^{n} \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i} v_i\right)\mathbf{v}\right]$$

$\mathbb{E}\left[g(\boldsymbol{\theta})\right]$ is a vector of size $n \times 1$. The j-th element of this vector can be expressed as:

$$\begin{aligned}\mathbb{E}\left[g_j(\boldsymbol{\theta})\right] &= \mathbb{E}\left[\left(\sum_{i=1}^{n} \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i} v_i\right) v_j\right] \\ &= \sum_{i=1}^{n} \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i} \mathbb{E}\left[v_i v_j\right]\end{aligned}$$

where I use the linearity of the expectation operator and the fact that, for a fixed value of $\boldsymbol{\theta}$, the partial derivatives $\frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i}$ are constant scalar values. Using the formula for the covariance between two random variables, one obtains:

$$\mathbb{E}\left[v_i v_j\right] = \mathrm{Cov}(v_i, v_j) + \mathbb{E}\left[v_i\right]\mathbb{E}\left[v_j\right]$$

The term $\mathbb{E}\left[v_i\right]\mathbb{E}\left[v_j\right]$ is equal to 0, because v is a zero-mean random vector. Because the variance-covariance matrix is the identity matrix, $\mathrm{Cov}(v_i, v_j)$ is 0 for $i \neq j$ and equal to 1 when $i = j$. Hence, $\mathbb{E}\left[g_j(\boldsymbol{\theta})\right] = \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_j}$, for all $j \in 1, 2, ..., n$. Since this holds for every $j \in \{1, 2, \ldots, n\}$, we conclude that $\mathbb{E}[g(\boldsymbol{\theta})] = \nabla f(\boldsymbol{\theta})$.

Appendix B. Proof proposition 2

Let us assume that for all values of the state and innovation vectors, the function $f : \mathbb{R}^n \to \mathbb{R}^{KMN}$ defined by $f(\boldsymbol{\theta}) = \boldsymbol{f_\theta}$ is Lipschitz continuous on its domain, where n denotes the dimension of the MLP's parameter vector. The loss function can be written as a quadratic form:

$$\mathcal{L}_{M,N}(\boldsymbol{\theta}) = \frac{2}{KMN(N-1)} \boldsymbol{f_\theta^T} \Sigma \boldsymbol{f_\theta}$$

where $\boldsymbol{f_\theta}$ is a column vector of size KMN and Σ is a $KMN \times KMN$ matrix with entries equal to 0 or 1. More specifically, Σ defined as the Kronecker product $\mathbb{I}_K \otimes \Lambda$, while Λ is an $MN \times MN$ matrix defined as the Kronecker product $\mathbb{I}_M \otimes \mathbb{U}_N$, with $\mathbb{I}_M$ being the $M \times M$ identity matrix and $\mathbb{U}_N$ being an $N \times N$ upper triangular matrix whose diagonal elements are all equal to zero, while the entries above the main diagonal are equal to one.

Using the fact that $\boldsymbol{x^T} A \boldsymbol{x} = \boldsymbol{x^T} \frac{A+A^T}{2} \boldsymbol{x}$, the loss function can be written as:

$$\mathcal{L}_{M,N}(\boldsymbol{\theta}) = \frac{1}{KMN(N-1)} \boldsymbol{f_\theta^T} (\Sigma + \Sigma^T) \boldsymbol{f_\theta}$$

Now, consider two points $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$ in $\mathbb{R}^n$. Using the fact that $\boldsymbol{x^T} B \boldsymbol{x} - \boldsymbol{y^T} B \boldsymbol{y} = (\boldsymbol{x} - \boldsymbol{y})^T B (\boldsymbol{x} + \boldsymbol{y})$ when B is a symmetric matrix, the difference can be written as:

$$\mathcal{L}_{M,N}(\boldsymbol{\theta_1}) - \mathcal{L}_{M,N}(\boldsymbol{\theta_2}) = \frac{1}{KMN(N-1)} \big(\boldsymbol{f_{\theta_1}} - \boldsymbol{f_{\theta_2}}\big)^T (\Sigma + \Sigma^T) \big(\boldsymbol{f_{\theta_1}} + \boldsymbol{f_{\theta_2}}\big)$$

Now, using the fact that f is Lipschtiz continuous and using $\|.\|_2$ to denote the Euclidean norm for vectors and the spectral norm for matrices:

$$\begin{aligned} \|\mathcal{L}_{M,N}(\boldsymbol{\theta_1}) - \mathcal{L}_{M,N}(\boldsymbol{\theta_2})\|_2 &\leq \frac{1}{KMN(N-1)} \|\big(\boldsymbol{f_{\theta_1}} - \boldsymbol{f_{\theta_2}}\big)\|_2 \|(\Sigma + \Sigma^T)\|_2 \|\big(\boldsymbol{f_{\theta_1}} + \boldsymbol{f_{\theta_2}}\big)\|_2 \\ &\leq \frac{1}{KMN(N-1)} L \|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\|_2 \|(\Sigma + \Sigma^T)\|_2 \|\big(\boldsymbol{f_{\theta_1}} + \boldsymbol{f_{\theta_2}}\big)\|_2 \end{aligned}$$

with $L \geq 0$ a constant. By construction, $\Sigma = \mathbb{I}_K \otimes \mathbb{I}_M \otimes \mathbb{U}_N$. Hence, $\Sigma + \Sigma^T = \mathbb{I}_K \otimes \mathbb{I}_M \otimes \big(\mathbb{U}_N + \mathbb{U}_N^T\big)$. Using the fact that $\|A \otimes B\|_2 = \|A\|_2 \|B\|_2$, we get $\|\Sigma + \Sigma^T\|_2 = \|\mathbb{U}_N + \mathbb{U}_N^T)\|_2$. Note that the spectral norm $\|\mathbb{U}_N + \mathbb{U}_N^T\|_2$ is bounded above by $\sqrt{\|\mathbb{U}_N + \mathbb{U}_N^T\|_1 \|\mathbb{U}_N + \mathbb{U}_N^T\|_\infty}$, where $\|A\|_1$ is equal to the maximum absolute column sum of the matrix A and $\|A\|_\infty$ is equal to the maximum absolute row sum of the matrix A. Given the construction of $\mathbb{U}_N$, both values are equal to $N-1$. Hence, one gets:

$$\|\mathcal{L}_{M,N}(\boldsymbol{\theta_1}) - \mathcal{L}_{M,N}(\boldsymbol{\theta_2})\|_2 \leq \frac{1}{KMN} L \|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\|_2 \|\big(\boldsymbol{f_{\theta_1}} + \boldsymbol{f_{\theta_2}}\big)\|_2 \tag{33}$$

If the function f is bounded, there exist a real constant $B \geq 0$ such that $\|\boldsymbol{f_{\theta_1}}\|_2 \leq B$ and $\|\boldsymbol{f_{\theta_2}}\|_2 \leq B$, yielding:

$$\|\mathcal{L}_{M,N}(\boldsymbol{\theta_1}) - \mathcal{L}_{M,N}(\boldsymbol{\theta_2})\|_2 \leq \frac{1}{KMN} 2BL\|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\|_2$$

which proves that the loss function is indeed Lipschitz continuous, with Lipschitz constant equal to $\frac{2BL}{KMN}$.

Now consider the compact subset $D \subset \mathbb{R}^n$. Because f is assumed to be Lipschitz continuous on $\mathbb{R}^n$, it is also continuous on $\mathbb{R}^n$, and by extension on $D \subset \mathbb{R}^n$. Because a continuous function on a compact set D is bounded and attains a maximum and minimum on D, there exists a finite constant $B_D \geq 0$ in $\mathbb{R}$ such that $\|\boldsymbol{f_{\theta_1}}\|_2 \leq B_D$ and $\|\boldsymbol{f_{\theta_2}}\|_2 \leq B_D$. Hence, we also get that:

$$\|\mathcal{L}_{M,N}(\boldsymbol{\theta_1}) - \mathcal{L}_{M,N}(\boldsymbol{\theta_2})\|_2 \leq \frac{1}{KMN} 2B_D L\|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\|_2$$

which proves that the loss function restricted to D is indeed Lipschitz continuous, with Lipschitz constant equal to $\frac{2B_D L}{KMN}$.

Appendix C. Proof proposition 3

Assume that for all values of the state and innovation vectors, the function $\psi : \mathbb{R}^n \to \mathbb{R}$ defined by $\psi(\boldsymbol{\theta}) = \mathcal{L}_{M,N}(\boldsymbol{\theta})$ is Lipschitz continuous and differentiable on its domain. Let $\mathbf{v}$ be a normally distributed random vector in $\mathbb{R}^n$ with zero mean and a variance-covariance matrix given by the identity matrix $\mathbb{I}_n$. I now show that when the scalar h tends to 0, the $\nabla_\theta \mathcal{L}^F_{M,N}(\boldsymbol{\theta})$ is an unbiased estimator of $\nabla_\theta \mathcal{L}(\boldsymbol{\theta})$.

First, note that the function $\nabla_\theta \mathcal{L}^F_{M,N}(\boldsymbol{\theta})$ can be compactly written as:

$$\begin{aligned}\nabla_\theta \mathcal{L}^F_{M,N}(\boldsymbol{\theta}) &= \mathbb{E}_{\mathbf{v}}\left[\frac{1}{h}\mathcal{L}_{M,N}(\boldsymbol{\theta} + h\mathbf{v})\mathbf{v}\right] \\ &= \mathbb{E}_{\mathbf{v}}\left[\frac{\mathcal{L}_{M,N}(\boldsymbol{\theta} + h\mathbf{v}) - \mathcal{L}_{M,N}(\boldsymbol{\theta})}{h}\mathbf{v}\right]\end{aligned}$$

because $\mathbb{E}_{\mathbf{v}}\left[\mathcal{L}_{M,N}(\boldsymbol{\theta})\mathbf{v}\right] = \mathbb{E}_{\mathbf{v}}\left[\mathcal{L}_{M,N}(\boldsymbol{\theta})\right]\mathbb{E}_{\mathbf{v}}\left[\mathbf{v}\right] = \mathcal{L}_{M,N}(\boldsymbol{\theta}) \times 0 = 0$.

Lipschitz continuity of the function $\mathcal{L}_{M,N}$ for all values of the state and innovation vectors gives us that the value $\frac{\mathcal{L}_{M,N}(\boldsymbol{\theta}+h\mathbf{v}) - \mathcal{L}_{M,N}(\boldsymbol{\theta})}{h}\mathbf{v}$ is bounded by some integrable function:

$$\begin{aligned}\|\frac{\mathcal{L}_{M,N}(\boldsymbol{\theta} + h\mathbf{v}) - \mathcal{L}_{M,N}(\boldsymbol{\theta})}{h}\mathbf{v}\|_2 &\leq \|\frac{\mathcal{L}_{M,N}(\boldsymbol{\theta} + h\mathbf{v}) - \mathcal{L}_{M,N}(\boldsymbol{\theta})}{h}\|_2\|\mathbf{v}\|_2 \\ &\leq L\frac{h\|\mathbf{v}\|_2\|\mathbf{v}\|_2}{h} = L\|\mathbf{v}\|_2^2\end{aligned}$$

where $L \geq 0$ is a finite real value. Because $\mathbf{v}$ is a normally distributed random vector in $\mathbb{R}^n$ with zero mean and a variance-covariance matrix given by the identity matrix, we have that $\mathbb{E}_{\mathbf{v}}\left[\|\mathbf{v}\|_2^2\right] = n < +\infty$, as n denotes the dimension of the MLP's parameter vector. Hence, we may use Lebesgue's dominated convergence theorem to get:

$$\begin{aligned}\lim_{h\to 0}\mathbb{E}_{\mathbf{v}}\left[\nabla_\theta \mathcal{L}^F_{M,N}(\boldsymbol{\theta})\right] &= \mathbb{E}_{\mathbf{v}}\left[\lim_{h\to 0}\frac{\mathcal{L}_{M,N}(\boldsymbol{\theta}+h\mathbf{v})-\mathcal{L}_{M,N}(\boldsymbol{\theta})}{h}\mathbf{v}\right] \\ &= \mathbb{E}_{\mathbf{v}}\left[\big(\nabla_\theta \mathcal{L}_{M,N}(\boldsymbol{\theta})\cdot\mathbf{v}\big)\mathbf{v}\right] \\ &= \nabla_\theta \mathcal{L}_{M,N}(\boldsymbol{\theta})\end{aligned}$$

where the last equality follows from Proposition 1. Hence, we have that

$$\begin{aligned}\mathbb{E}_{\boldsymbol{s},\boldsymbol{\varepsilon}}\left[\lim_{h\to 0}\mathbb{E}_{\mathbf{v}}\left[\nabla_\theta \mathcal{L}^F_{M,N}(\boldsymbol{\theta})\right]\right] &= \mathbb{E}_{\boldsymbol{s},\boldsymbol{\varepsilon}}\left[\nabla_\theta \mathcal{L}_{M,N}(\boldsymbol{\theta})\right] \\ &= \nabla_\theta\, \mathbb{E}_{\boldsymbol{s},\boldsymbol{\varepsilon}}\left[\mathcal{L}_{M,N}(\boldsymbol{\theta})\right] \\ &= \nabla_\theta \mathcal{L}(\boldsymbol{\theta})\end{aligned}$$

where the second line follows from another application of Lebesgue's dominated convergence theorem, while the third line follows from the fact that $\mathcal{L}_{M,N}(\boldsymbol{\theta})$ is an unbiased estimator of $\mathcal{L}(\boldsymbol{\theta})$, as shown in Pascal (2024). Here, one may use Lebesgue's dominated convergence theorem because $\mathcal{L}_{M,N}(\boldsymbol{\theta})$ is bounded by an integrable function. Indeed, because we assume that $\mathcal{L}_{M,N}(\boldsymbol{\theta})$ is both Lipschitz continuous and differentiable on its domain, the norm of its gradient is bounded by some finite positive value, denoted by K: $\|\nabla_\theta \mathcal{L}_{M,N}(\boldsymbol{\theta})\|_2 \leq K$. Clearly, $\mathbb{E}_{\boldsymbol{s},\boldsymbol{\varepsilon}}\left[K\right] = K < +\infty$.

Appendix D. Smooth OLG model: the two-agent case

Let us consider the two-agent version of the OLG model from Section 4.2. The general A-agent version is treated in Section E of the Appendix. In the two-agent version, the model is characterized by a single Euler equation

$$\mathbb{E}_{\varepsilon_\eta,\varepsilon_\delta}\left[\beta\big(\frac{c^{h+1}_{t+1}}{c^h_t}\big)^{-\gamma} r_{t+1} - 1\right] = 0$$

where $h = 1$. Let $\tilde{\mathcal{MLP}}(\boldsymbol{y_t}|\boldsymbol{\theta}) = \max\{\tau, \min\{\sigma\big(\mathcal{MLP}(\boldsymbol{y_t}|\boldsymbol{\theta})\big), 1-\tau\}\}$ denote the share of total wealth $\boldsymbol{y_t} = (y^1_t, y^2_t) = (w_t, r_t k^2_t)$ consumed in period t. The function $\sigma(.)$ is the sigmoid activation function, while $\tau > 0$ is a small positive real number, truncating corner solutions. Here, because there are only two age groups, for a fixed parameter

vector $\boldsymbol{\theta} \in \mathbb{R}^n$, the function $\boldsymbol{y} \to \tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})$ is a mapping from $\mathbb{R}^2$ to $\mathbb{R}$. For a fixed value of $\boldsymbol{y} \in \mathbb{R}^2$, the function $\boldsymbol{\theta} \to \tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})$ is a mapping from $\mathbb{R}^n$ to $\mathbb{R}$.

Consumption of the first age group in period t is given by $c_t^1 = \tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})y_t^1$. By assumption, the same generation next period consumes all their available wealth: $c_t^2 = (1 - \tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta}))r_{t+1}y_t^1$. Hence, the Euler equation can be rewritten as

$$\mathbb{E}_{\varepsilon_\eta,\varepsilon_\delta}\left[\beta\big(\frac{1-\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})}{\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})}\big)^{-\gamma}\big(r_{t+1}\big)^{1-\gamma} - 1\right] = 0$$

For the logarithmic utility case ($\gamma = 1$), this equation simplifies further to

$$\mathbb{E}_{\varepsilon_\eta,\varepsilon_\delta}\left[\beta\frac{\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})}{1-\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})} - 1\right] = 0$$

The bc-MC residual function here is $f(\boldsymbol{y_t}|\boldsymbol{\theta}) = \beta\frac{\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})}{1-\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})} - 1$, which in this special case does not depend on the realization of the innovation vector $(\varepsilon_\eta, \varepsilon_\delta)$.

Let us now analyze the function f and see under which conditions Proposition 2 applies. In particular, let us assume that the MLP under consideration is Lipschitz continuous for all values of $\boldsymbol{y_t}$, with Lipschitz constant denoted by $L \geq 0$. The conditions under which the function $\boldsymbol{\theta} \in \mathbb{R}^n \to \mathcal{MLCP}(\boldsymbol{y_t}|\boldsymbol{\theta}) \in \mathbb{R}$ is Lipschitz continuous for all values of $\boldsymbol{y_t}$ are expressed in Section F of the Appendix.

First, observe that the function $f(\boldsymbol{x}) = \max\{\tau, \min\{\boldsymbol{x}, 1-\tau\}\}$ is Lipschitz continuous with Lipschitz constant equal to 1, where the function $x \to \max\{\tau, \min\{x, 1-\tau\}\}$ is applied element-wise to the vector $\boldsymbol{x}$. Hence, for two parameter vectors $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$ in $\mathbb{R}^n$, we have

$$\begin{aligned}
\|\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta_1}) - \tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta_2})\|_2 &\leq \|\sigma\big(\mathcal{MLCP}(\boldsymbol{y_t}|\boldsymbol{\theta_1})\big) - \sigma\big(\mathcal{MLCP}(\boldsymbol{y_t}|\boldsymbol{\theta_2})\big)\|_2 \\
&\leq \frac{1}{4}\|\mathcal{MLCP}(\boldsymbol{y_t}|\boldsymbol{\theta_1}) - \mathcal{MLCP}(\boldsymbol{y_t}|\boldsymbol{\theta_2})\|_2 \\
&\leq \frac{1}{4}L\|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\|_2
\end{aligned}$$

where the second line uses the fact the sigmoid function is Lipschitz continuous with Lipschtiz constant equal to $\frac{1}{4}$, while the third line uses Lipschitz continuity of the MLP, for all values of $\boldsymbol{y_t}$.

Now, let us analyse the function $g(t) = \frac{t}{1-t}$ on the interval $[\tau, 1-\tau]$. The derivative of g is given by $g'(t) = \frac{1}{(1-t)^2}$, which attains a maximum of $\frac{1}{(1-\tau)^2}$ on $[\tau, 1-\tau]$. An application

of the mean-value inequality gives us that $|g(a) - g(b)| \leq \max_{t\in(\tau,1-\tau)} g'(t)|a-b| = \frac{1}{(1-\tau)^2}|a-b|$. Hence, for a fixed value of $\boldsymbol{y_t}$, we have

$$\begin{aligned}\|g\big(\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta_1})\big) - g\big(\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta_2})\big)\|_2 &\leq \frac{1}{(1-\tau)^2}\|\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta_1}) - \tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta_2})\|_2 \\ &\leq \frac{L}{4(1-\tau)^2}\|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\|_2\end{aligned}$$

The bc-MC residual function can be written as $f(\boldsymbol{y_t}|\boldsymbol{\theta}) = \beta g\big(\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})\big) - 1$. Hence, we have proven that when $\gamma = 1$ and when the conditions listed in Section F are met, the function f is Lipschitz continuous with Lipschitz constant given by $\frac{\beta L}{4(1-\tau)^2}$, for any value of $\boldsymbol{y_t}$.

Regarding boundedness of $f(\boldsymbol{y_t}|\boldsymbol{\theta})$, note that by construction, $\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})$ reaches a maximum value equal to $1-\tau > 0$ and a minimum value equal to $\tau > 0$. Hence, $f(\boldsymbol{y_t}|\boldsymbol{\theta})$ has for maximum $\frac{\beta}{\tau} - 1$ and for minimum $\frac{\beta\tau}{1-\tau} - 1$.

Appendix E. Smooth OLG model: the A-agent case

Let us now consider the OLG model from Section 4.2 with A households, representing different age groups. An equilibrium is characterized by $A-1$ Euler equations. For $h \in \{1, 2, ..., A-1\}$:

$$\mathbb{E}_{\varepsilon_\eta,\varepsilon_\delta}\left[\beta(\frac{c_{t+1}^{h+1}}{c_t^h})^{-\gamma} r_{t+1} - 1\right] = 0$$

As in the previous section, let $\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta}) = \max\{\tau, \min\{\sigma\big(\mathcal{MLCP}(\boldsymbol{y_t}|\boldsymbol{\theta})\big), 1-\tau\}\}$ denote the share of total wealth $\boldsymbol{y_t} = (y_t^1, y_t^2, ..., y_t^A) = (w_t, r_t k_t^2, ..., r_t k_t^A)$ consumed in period t. Here, because there are A age groups, for a fixed parameter vector $\boldsymbol{\theta} \in \mathbb{R}^n$, the function $\boldsymbol{y} \to \tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta}) = (\tilde{\mathcal{MLCP}}_1(\boldsymbol{y_t}|\boldsymbol{\theta}), ..., \tilde{\mathcal{MLCP}}_h(\boldsymbol{y_t}|\boldsymbol{\theta}), ..., \tilde{\mathcal{MLCP}}_{A-1}(\boldsymbol{y_t}|\boldsymbol{\theta}))$ is a mapping from $\mathbb{R}^A$ to $\mathbb{R}^{A-1}$. For a fixed value of $\boldsymbol{y} \in \mathbb{R}^A$, the function $\boldsymbol{\theta} \to \tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})$ is a mapping from $\mathbb{R}^n$ to $\mathbb{R}^{A-1}$. As before, let us assume that the MLP under consideration is Lipschitz continuous for all values of $\boldsymbol{y_t}$, with Lipschitz constant denoted by $L \geq 0$ (see Section F of the Appendix).

Consumption of age group h in period t is given by $c_t^h = \tilde{\mathcal{MLCP}}_h(\boldsymbol{y_t}|\boldsymbol{\theta}) y_t^h$. Consumption of the same generation next period is given by $c_{t+1}^{h+1} = \tilde{\mathcal{MLCP}}_{h+1}(\boldsymbol{y_{t+1}}|\boldsymbol{\theta}) y_{t+1}^{h+1} = \tilde{\mathcal{MLCP}}_{h+1}(\boldsymbol{y_{t+1}}|\boldsymbol{\theta})\big(1 - \tilde{\mathcal{MLCP}}_h(\boldsymbol{y_t}|\boldsymbol{\theta})\big) y_t^h r_{t+1}$, because what is not consumed in period t

is invested. Hence, the Euler equations can be written as

$$\mathbb{E}_{\varepsilon_\eta,\varepsilon_\delta}\big[\beta\Big(\frac{\tilde{\mathcal{ML}}\mathcal{P}_{h+1}(\boldsymbol{y_{t+1}}|\boldsymbol{\theta})\big(1-\tilde{\mathcal{ML}}\mathcal{P}_h(\boldsymbol{y_t}|\boldsymbol{\theta})\big)}{\tilde{\mathcal{ML}}\mathcal{P}_h(\boldsymbol{y_t}|\boldsymbol{\theta})}\Big)^{-\gamma} r_{t+1}^{1-\gamma}-1\big]=0$$

which simplifies to the following for $\gamma=1$:

$$\mathbb{E}_{\varepsilon_\eta,\varepsilon_\delta}\big[\underbrace{\beta\frac{\tilde{\mathcal{ML}}\mathcal{P}_h(\boldsymbol{y_t}|\boldsymbol{\theta})}{1-\tilde{\mathcal{ML}}\mathcal{P}_h(\boldsymbol{y_t}|\boldsymbol{\theta})}\frac{1}{\tilde{\mathcal{ML}}\mathcal{P}_{h+1}(\boldsymbol{y_{t+1}}|\boldsymbol{\theta})}}_{\equiv f_h(\boldsymbol{y_t},\boldsymbol{\varepsilon}|\boldsymbol{\theta})}-1\big]=0$$

For the penultimate age group ($h=A-1$), the previous equation simplifies further to

$$\mathbb{E}_{\varepsilon_\eta,\varepsilon_\delta}\big[\beta\frac{\tilde{\mathcal{ML}}\mathcal{P}_h(\boldsymbol{y_t}|\boldsymbol{\theta})}{1-\tilde{\mathcal{ML}}\mathcal{P}_h(\boldsymbol{y_t}|\boldsymbol{\theta})}-1\big]=0$$

because the households consume all their remaining savings in their final period. This equation is similar to the one appearing in Section D of the Appendix, where it was established that the function $\boldsymbol{\theta}\to\beta\frac{\tilde{\mathcal{ML}}\mathcal{P}_h(\boldsymbol{y_t}|\boldsymbol{\theta})}{1-\tilde{\mathcal{ML}}\mathcal{P}_h(\boldsymbol{y_t}|\boldsymbol{\theta})}-1$ is Lipschitz continuous with Lipschitz constant given by $\frac{\beta L}{4(1-\tau)^2}$, for any value of $\boldsymbol{y_t}$.

Let us now consider the function $f_h(\boldsymbol{y_t},\boldsymbol{\varepsilon}|\boldsymbol{\theta})$ for $h\neq A-1$. It is straightforward to see that $\tilde{\mathcal{ML}}\mathcal{P}_h$ reaches a minimum equal to $\tau>0$. Thus, $\big(\tilde{\mathcal{ML}}\mathcal{P}_h\big)^{-1}$ reaches a maximum equal to $\frac{1}{\tau}$. Hence, $\|f_h(\boldsymbol{y_t},\boldsymbol{\varepsilon}|\boldsymbol{\theta})\|_2\leq\|\beta\frac{\tilde{\mathcal{ML}}\mathcal{P}_h(\boldsymbol{y_t}|\boldsymbol{\theta})}{1-\tilde{\mathcal{ML}}\mathcal{P}_h(\boldsymbol{y_t}|\boldsymbol{\theta})}-1\|_2\|\frac{1}{\tau}\|_2$. It directly follows that, for $h\in\{1,...,A-2\}$, $f_h(\boldsymbol{y_t},\boldsymbol{\varepsilon}|\boldsymbol{\theta})$ is Lipschitz continuous, with Lipschitz constant $L_h=\frac{\beta L}{4\tau(1-\tau)^2}$, for all realizations of the state and innovation vectors.

Hence, the mapping from $\mathbb{R}^n$ to $\mathbb{R}^{A-1}$ given by $f(\boldsymbol{y_t},\boldsymbol{\varepsilon}|\boldsymbol{\theta})=(f_1(\boldsymbol{y_t},\boldsymbol{\varepsilon}|\boldsymbol{\theta}),...,f_{A-1}(\boldsymbol{y_t},\boldsymbol{\varepsilon}|\boldsymbol{\theta}))$ is itself Lipschitz continuous, with Lipschitz constant given by $\big(\frac{A-2}{\tau^2}+1\big)^{0.5}\frac{\beta L}{4(1-\tau)^2}$. Indeed, using the Lipschitz constant L_h previously established, one gets, for $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$ in $\mathbb{R}^n$

$$\begin{aligned}
\|f(\boldsymbol{y_t},\boldsymbol{\varepsilon}|\boldsymbol{\theta_1})-f(\boldsymbol{y_t},\boldsymbol{\varepsilon}|\boldsymbol{\theta_2})\|_2 &= \Big(\sum_{h=1}^{A-1}|f_h(\boldsymbol{y_t},\boldsymbol{\varepsilon}|\boldsymbol{\theta_1})-f_h(\boldsymbol{y_t},\boldsymbol{\varepsilon}|\boldsymbol{\theta_2})|^2\Big)0.5\\
&\leq \Big(\sum_{h=1}^{A-1}L_h^2\|\boldsymbol{\theta_1}-\boldsymbol{\theta_2}\|_2^2\Big)^{0.5}\\
&= \Big(\sum_{h=1}^{A-1}L_h^2\Big)^{0.5}\|\boldsymbol{\theta_1}-\boldsymbol{\theta_2}\|_2\\
&= \Big((A-2)\big(\frac{\beta L}{4\tau(1-\tau)^2}\big)^2+\big(\frac{\beta L}{4(1-\tau)^2}\big)^2\Big)^{0.5}\|\boldsymbol{\theta_1}-\boldsymbol{\theta_2}\|_2\\
&= \big(\frac{A-2}{\tau^2}+1\big)^{0.5}\frac{\beta L}{4(1-\tau)^2}\|\boldsymbol{\theta_1}-\boldsymbol{\theta_2}\|_2
\end{aligned}$$

Appendix F. Lipschitz continuity of neural networks

In this section, I discuss the conditions under which an MLP (Multilayer Perceptron) is Lipschitz continuous with respect to its parameter vector. Let us consider a two-layer MLP with a bounded and Lipschitz continuous activation function σ, with Lipschitz constant denoted by $L_\sigma \geq 0$ and such that $|\sigma(x)| \leq B_\sigma$ for all x, where $B_\sigma \geq 0$. This includes popular activation functions, such as the sigmoid ($L_\sigma = \frac{1}{4}$) or the hyperbolic tangent function ($L_\sigma = 1$). The argument made in this section could then be extended by induction to MLPs with additional layers. Let us further assume that the Euclidean norm of the input vector $\|\boldsymbol{y}\|_2$ is bounded by $B_y < +\infty$.

Without loss of generality, the MLP can be written as $f(A_1, A_0, \boldsymbol{y}) = \sigma(A_1\sigma(A_0\boldsymbol{y}))$.[11] Let us assume that $A_0 \in \mathbb{R}^{n_0 \times n_y}$ and $A_1 \in \mathbb{R}^{n_1 \times n_0}$. With these assumptions, and for a fixed value of $\boldsymbol{y}$, the MLP is a mapping from the space of two matrices $M = (A_1, A_0) \in \mathbb{R}^{n_1 \times n_0} \times \mathbb{R}^{n_0 \times n_y}$ to $\mathbb{R}^{n_1}$. Let us equip this space with the norm $\|M_1\|_F + \|M_0\|_F$, where $\|A\|_F = \left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2\right)^{0.5}$ is the Frobenius norm for a $m \times n$ matrix. For a vector (a $m \times 1$ matrix), the Frobenius norm is the usual Euclidean norm $\|\boldsymbol{v}\|_F = \left(\sum_{i=1}^m |\boldsymbol{v}_i|^2\right)^{0.5} = \|\boldsymbol{v}\|_2$. For a single matrix (or vector), we may also use the max norm $\|A\|_\infty = \max_{i,j} |a_{ij}|$. Note that for $m \times n$ matrix, $\|A\|_F \leq \sqrt{mn}\|A\|_\infty$. Note that since we are only considering finite-dimensional vector spaces, the proof does not depend on which norms are used, as all norms are equivalent in this context.

Let us now consider two MLP parameters $A^{(0)} = (A_1^{(0)}, A_0^{(0)})$ and $A^{(1)} = (A_1^{(1)}, A_0^{(1)})$ in $\mathbb{R}^{n_1 \times n_0} \times \mathbb{R}^{n_0 \times n_y}$ and find some upper bound of the form $L\|A^{(0)} - A^{(1)}\|_F$ for the difference $\|g(A^{(0)}) - g(A^{(1)})\|_F$, where $g(A^{(i)}) = f(A_1^{(0)}, A_0^{(0)}, \boldsymbol{y})$. For such an upper bound to exist, we further need to assume that the MLP parameters are restricted to a compact subset of $\mathbb{R}^{n_1 \times n_0} \times \mathbb{R}^{n_0 \times n_y}$. Here, it implies that all the entries of A_1 and A_0 are bounded (in absolute value) by a finite real number. This ensures that $\|A_1^{(0)}\|_F$ and $\|A_1^{(1)}\|_F$ are bounded by a finite real number $B_A \geq 0$, because for $m \times n$ matrix $\|A\|_F \leq \sqrt{mn}\|A\|_\infty$.

[11] In general, MLPs also include the addition of a bias vector at each layer: $f(A_1, A_0, \boldsymbol{y}) = \sigma(A_1\sigma(A_0\boldsymbol{y} + \boldsymbol{b_0}) + \boldsymbol{b_1})$. This can be written as $\sigma(\tilde{A}_1\sigma(\tilde{A}_0\tilde{\boldsymbol{y}}))$, where $\tilde{A}_i = \begin{pmatrix} A_i & \boldsymbol{b_i} \\ \boldsymbol{0} & 1 \end{pmatrix}$ denotes an augmented matrix, and $\tilde{\boldsymbol{y}} = \begin{pmatrix} \boldsymbol{y} \\ 1 \end{pmatrix}$ denotes an augmented vector, such that the affine transformation $A_i y + \boldsymbol{b_i}$ can be represented by a matrix multiplication $\tilde{A}_i\tilde{\boldsymbol{y}}$.

Combining these assumptions, we get:

$$
\begin{aligned}
\|g(A^{(0)}) - g(A^{(1)})\|_F &= \|\sigma(A_1^{(0)}\sigma(A_0^{(0)}\boldsymbol{y})) - \sigma(A_1^{(1)}\sigma(A_0^{(1)}\boldsymbol{y}))\|_F \\
&\leq L_\sigma \|A_1^{(0)}\sigma(A_0^{(0)}\boldsymbol{y}) - A_1^{(1)}\sigma(A_0^{(1)}\boldsymbol{y})\|_F \\
&= L_\sigma \|A_1^{(0)}\sigma(A_0^{(0)}\boldsymbol{y}) - A_1^{(0)}\sigma(A_0^{(1)}\boldsymbol{y}) + A_1^{(0)}\sigma(A_0^{(1)}\boldsymbol{y}) - A_1^{(1)}\sigma(A_0^{(1)}\boldsymbol{y})\|_F \\
&\leq L_\sigma \Big(\|A_1^{(0)}\|_F \|\sigma(A_0^{(0)}\boldsymbol{y}) - \sigma(A_0^{(1)}\boldsymbol{y})\|_F + \|\sigma(A_0^{(1)}\boldsymbol{y})\|_F \|A_1^{(0)}\boldsymbol{y} - A_1^{(1)}\boldsymbol{y}\|_F \Big) \\
&\leq L_\sigma \|\boldsymbol{y}\|_F \big(L_\sigma B_A \|A_0^{(0)} - A_0^{(1)}\|_F + B_\sigma \|A_1^{(0)} - A_1^{(1)}\|_F \big) \\
&\leq L_\sigma B_y (L_\sigma B_A + B_\sigma) \big(\|A_0^{(0)} - A_0^{(1)}\|_F + \|A_1^{(0)} - A_1^{(1)}\|_F \big) \\
&= \underbrace{L_\sigma B_y (L_\sigma B_A + B_\sigma)}_{=L \geq 0} \|A^{(0)} - A^{(1)}\|_F
\end{aligned}
$$

The second line uses Lipschitz continuity of the activation function. The third line adds and subtracts the same value, while the fourth lines uses the triangular inequality. The fifth line uses once again Lipschitz continuity of the activation function, and boundedness of $\|A_1^{(0)}\|_F$, $\|\boldsymbol{y}\|_F$, and $\|\sigma(A_0^{(1)}\boldsymbol{y})\|_F$. The sixth lines adds and factorize to the right hand side $B_\sigma \|A_0^{(0)} - A_0^{(1)}\|_F$ and $(L_\sigma B_A)\|A_1^{(0)} - A_1^{(1)}\|_F$, which are positive quantities.

To summarize, when (i) the activation function σ is Lipschitz continuous and bounded, (ii) the input vector $\|\boldsymbol{y}\|_2$ is bounded, (iii) the MLP parameter vector can only take finite values, then the MLP is Lipschitz continuous with respect to its parameter values.

Condition (i) is not too restrictive, as it is met with popular activation functions (e.g. sigmoid of hyperbolic tangent functions). One may even use a ReLU activation function, since $\|\sigma(A_0^{(1)}\boldsymbol{y})\|_F = \|\max\{0, A_0^{(1)}\boldsymbol{y}\}\|_F \leq \|A_0^{(1)}\boldsymbol{y}\|_F \leq \|A_0^{(1)}\|_F \|\boldsymbol{y}\|_F = B_A B_y < +\infty$. As already discussed in the main text, condition (iii) can be met by using gradient clipping combined with a finite number of gradient descent steps, which ensures that the norm of the gradient does not exceed a certain threshold. Similarly, condition (ii) might require truncating exogenous variables with support $\mathbb{R}$. Consider for instance the neoclassical growth model where period t household's income y_t is equal to $k_t^\alpha \eta_t$, with η denoting lognormal i.i.d. productivity shocks. Boundedness of y_t can be ensured by working with a truncated lognormal distribution, or by using a finite discretization of the continuous productivity process. It could also be truncated ex-post by discarding realizations of state vectors exceeding a norm threshold during training.

Appendix G. Smooth OLG model with softmax and softmin functions

This section is concerned with the applicability of Proposition 3 to the OLG model of Section 4.2. Sections D and E use the function $\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta}) = \max\{\tau, \min\{\sigma\big(\mathcal{MLCP}(\boldsymbol{y_t}|\boldsymbol{\theta})\big), 1-\tau\}\}$, which introduces non-differentiabilities because of the presence of max and min operators. Hence, Proposition 3 cannot be used directly.

Fortunately, the functions $\max(x, y)$ and $\min(x, y)$ can be smoothly approximated with softmin and softmax functions, defined as:

$$\text{softmax}(x, y) = \frac{1}{a} \ln\left(e^{ax} + e^{ay}\right) \approx \max(x, y)$$
$$\text{softmin}(x, y) = -\frac{1}{a} \ln\left(e^{-ax} + e^{-ay}\right) \approx \min(x, y)$$

where $a > 0$ is a parameter that controls the sharpness of these approximations. These approximations can be made as accurate as one wants, in the sense that at the limit for a tending to infinity, they yield the correct values:

$$\lim_{a\to\infty} \text{softmax}(x, y) = \max(x, y)$$
$$\lim_{a\to\infty} \text{softmin}(x, y) = \min(x, y)$$

Combining the softmin and softmax functions, one gets an *everywhere differentiable functions* $\check{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta}) = \text{softmax}\{\tau, \text{softmin}\{\sigma\big(\mathcal{MLCP}(\boldsymbol{y_t}|\boldsymbol{\theta})\big), 1-\tau\}\}$, that has for limit $\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})$ when a tends to infinity.

It remains to be shown that using $\check{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})$, rather than $\tilde{\mathcal{MLCP}}(\boldsymbol{y_t}|\boldsymbol{\theta})$, does not alter the Lipschitz continuity of the bc-MC residual function, as established in sections D - F. These sections use the fact that the function $f(\boldsymbol{x}) = \max\{\tau, \min\{\boldsymbol{x}, 1-\tau\}\}$ is Lipschitz continuous with Lipschitz constant equal to 1, where the function $x \to \max\{\tau, \min\{x, 1-\tau\}\}$ is applied element-wise to the vector $\boldsymbol{x}$. If we show that the function $\check{f}(\boldsymbol{x}) = \text{softmax}\{\tau, \text{softmin}\{\boldsymbol{x}, 1-\tau\}\}$ is also Lipschitz continuous with Lipschitz constant equal to 1, then the rest of the arguments developed in sections D - F also apply.

Let us first consider the function $x \in \mathbb{R} \to \text{softmax}\{\tau, x\}$, for $\tau > 0$ fixed. This function is differentiable everywhere on its domain and has for derivative $\sigma(a(x-\tau))$, where σ is the sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$. Because the sigmoid activation function satisfies $0 < \sigma(x) < 1$, $\forall x \in \mathbb{R}$, it follows that the function $x \in \mathbb{R} \to \text{softmax}\{\tau, x\}$ is Lispschitz continuous with Lispschitz constant $L_1 = 1$. By the same token, the function

$x \in \mathbb{R} \to \text{softmin}\{\tau, x\}$ is also differentiable everywhere on its domain and has for derivative $\sigma(a(1-\tau-x))$. Hence, it is also Lispschitz continuous with Lispschitz constant $L_2 = 1$. The function $\check{f}(x) = \text{softmax}\{\tau, \text{softmin}\{x, 1-\tau\}\}$ is the composition of two Lipschitz continuous functions, hence it is also Lipschitz continuous with Lipschitz constant $L_3 = L_1 L_2 = 1$. Thus, the function $\boldsymbol{x} \in \mathbb{R}^n \to \text{softmax}\{\tau, \text{softmin}\{\boldsymbol{x}, 1-\tau\}\}$ is also Lipschitz continuous with Lipschitz constant equal to 1. Hence, the function $\check{f}$ is indeed Lipschitz continuous with Lipschitz constant equal to 1 and Proposition 3 applies.

Appendix H. The (gradient-free) bc-MC operator with dynamic selection of N^* and unknown ergodic distribution

Algorithm 1 N^*-algorithm: (gradient-free) bc-MC operator with dynamic selection of N^* and unknown ergodic distribution

Initialization:

- Choose artificial neural network (ANN) architecture (including number of layers, nodes per layer, activation functions) and define loss function using bc-MC operator.
- Initialize ANN parameter vector $\boldsymbol{\theta}$.
- Set number of episodes E.
- Set number of gradient descent iterations G per episode.
- Determine initial computational budget $T \equiv \frac{MN}{2}$.
- Initialize hyperparameters (M, N).
- Choose frequency F for hyperparameters (M, N) update.
- Initialize learning rate γ and choose optimization algorithm (e.g., ADAM).

Episodes Loop:

for $e = 1$ to E **do**

Simulate time series using current ANN parameter vector $\boldsymbol{\theta}$. This gives us a current guess for the ergodic distribution of endogeneous variables Π

Training Loop, given Π**:**

for $i = 1$ to G **do**

Compute decision functions using current ANN parameter vector $\boldsymbol{\theta}$, current guess for ergodic distribution Π and current hyperparameters (M, N).

Compute loss using bc-MC operator with equation (13) and calculate its gradient using back-propagation

or calculate an estimator of the gradient of the loss using equation (14).

Perform gradient descent step to update ANN parameter vector.

if $i \mod F == 0$ **then** ▷ Update (M, N) every F iterations

Evaluate variance of loss function using current ANN parameter vector for different values of (M, N), keeping the budget T constant (see Section I of the Appendix).

Dynamically adjust N to N^* to minimize variance of loss.

Adjust M to M^* to keep the budget T constant ($M^* = \frac{2T}{N^*}$).

end if

end for

end for

Output:

- Final trained ANN model, $\boldsymbol{\theta^*}$

Appendix I. Variance of the bc-MC loss function

The variance of the bc-MC loss function can be tightly characterized under certain assumptions. For the moment, consider a model with a single stochastic equation, in which case the bc-MC loss functions is

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

Let $\boldsymbol{f_{m|\theta}} \equiv \begin{pmatrix} f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^1}|\boldsymbol{\theta}) \\ f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^2}|\boldsymbol{\theta}) \\ ... \\ f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^N}|\boldsymbol{\theta}) \end{pmatrix}$ denote the random vector of size N. Let us assume that $\big(\boldsymbol{f_{1|\theta}}, \boldsymbol{f_{2|\theta}}, ..., \boldsymbol{f_{M|\theta}}\big)$ is an i.i.d. sample drawn from a multivariate normal distribution $\mathcal{N}\Big(\boldsymbol{\mu_\theta}, \Sigma_\theta\Big)$. Further assume that for all $m \in \{1, ..., M\}$ and $(i, j) \in \{1, ..., N\}^2$, $\mathbb{E}(f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta})) = \mu_f$, $\text{Var}(f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta})) = \sigma_f^2$, and $\text{Cov}(f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta}), f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^j}|\boldsymbol{\theta})) = \rho_f$ for all $i \neq j$. In particular, $\boldsymbol{\mu_\theta}$ is a vector of size N where each element is equal to μ_f. Σ_θ is a $N \times N$ matrix where diagonal elements are equal to σ_f^2 and off-diagonal elements are equal to ρ_f. These assumptions can be seen as originating from a first order Taylor expansion of the function f around the mean value for $(\boldsymbol{s}, \boldsymbol{\varepsilon})$, when the state vector $\boldsymbol{s}$ and the zero-mean innovation vector $\boldsymbol{\varepsilon}$ are both independent and normally distributed.

Under these assumptions, one can show (see Pascal, 2024) that the variance of the loss is equal to:

$$\begin{aligned}\text{Var}(\mathcal{L}_{M,N}(\boldsymbol{\theta})) = \frac{2}{MN(N-1)} \Big[\big((N-2)^2 + N - 1\big)\rho_f^2 + \big(2(N-2)\rho_f + \sigma_f^2\big)\sigma_f^2 \\ + 2(N-1)\big(\sigma_f^2 + (N-1)\rho_f\big)\mu_f^2 \Big]\end{aligned}$$

Let us now consider the case where the model has K stochastic equations (with $\mu_k = \frac{1}{K}$):

$$\mathcal{L}_{M,N}(\boldsymbol{\theta}) = \frac{1}{K} \sum_{k=1}^{K} \frac{2}{MN(N-1)} \sum_{m=1}^{M} \sum_{1 \leq i < j}^{N} f_k(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta}) f_k(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^j}|\boldsymbol{\theta})$$

Let $\boldsymbol{f_{k,m|\theta}} \equiv \begin{pmatrix} f_k(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^1}|\boldsymbol{\theta}) \\ f_k(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^2}|\boldsymbol{\theta}) \\ ... \\ f_k(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^N}|\boldsymbol{\theta}) \end{pmatrix}$ denote the random vector of size N, assumed to be an i.i.d. draw from a multivariate normal distribution $\mathcal{N}\Big(\boldsymbol{\mu_{k,\theta}}, \Sigma_{k,\theta}\Big)$, with $\mu_{k,\theta} = \mu_{f_k} \mathbf{1}_N$

and $\Sigma_{k,\theta} = \sigma^2_{f_k}\mathbb{I}_N + (\mathbb{J}_N - \mathbb{I}_N)\rho_{f_k}$. Here, $\mathbf{1}_N$ denotes a $N \times 1$ vector of ones, $\mathbb{I}_N$ is the $N \times N$ identity matrix, and $\mathbb{J}_N$ denotes a $N \times N$ matrix of ones. If we assume that $\boldsymbol{f_{k_1,m|\theta}}$ and $\boldsymbol{f_{k_2,m|\theta}}$ are independent, for all $k_1 \neq k_2$ in $\{1, ..., K\}$, one gets

$$\mathrm{Var}(\mathcal{L}^U_{M,N}(\boldsymbol{\theta})) = \frac{2}{KMN(N-1)} \sum_{k=1}^{K} \Big[\big((N-2)^2 + N - 1\big)\rho^2_{f_k} + \big(2(N-2)\rho_{f_k} + \sigma^2_{f_k}\big)\sigma^2_{f_k} + 2(N-1)\big(\sigma^2_{f_k} + (N-1)\rho_{f_k}\big)\mu^2_{f_k} \Big]$$

where for all $k \in \{1, ..., K\}$, $m \in \{1, ..., M\}$ and $(i, j) \in \{1, ..., N\}^2$, $\mathbb{E}(f_k(\boldsymbol{s_m}, \boldsymbol{\varepsilon^i_m}|\boldsymbol{\theta})) = \mu_{f_k}$, $\mathrm{Var}(f_k(\boldsymbol{s_m}, \boldsymbol{\varepsilon^i_m}|\boldsymbol{\theta})) = \sigma^2_{f_k}$, and $\mathrm{Cov}(f_k(\boldsymbol{s_m}, \boldsymbol{\varepsilon^i_m}|\boldsymbol{\theta}), f_k(\boldsymbol{s_m}, \boldsymbol{\varepsilon^j_m}|\boldsymbol{\theta})) = \rho_{f_k}$ for all $i \neq j$. For a fixed number of equations (K) and for a fixed computational budget given by $T = \frac{MN}{2}$, finding the minimum of $\mathrm{Var}(\mathcal{L}^U_{M,N}(\boldsymbol{\theta}))$ is a one-dimensional minimization problem in N. The quantities μ_{f_k}, $\sigma^2_{f_k}$, and ρ_{f_k} can be approximated by random sampling and using sample statistics.

Appendix J. Connection with the Parameterized Expectation Algorithm

This section demonstrates that the results of Section 4.4 can be generalized to the case $N > 2$. In particular, I show that if the function f used to form the bc–MC loss function can be decomposed as $f(\boldsymbol{s}_m, \boldsymbol{\varepsilon}^i_m|\boldsymbol{\theta}) = g(\boldsymbol{s}_m, \boldsymbol{\varepsilon}^i_m|\boldsymbol{\theta}) - \boldsymbol{s}^T_m\boldsymbol{\theta}$, then the gradient of the bc–MC loss function reduces to an expression that is equivalent to the gradient from a linear regression of the sample averages $\frac{1}{N}\sum_{i=1}^N g(\boldsymbol{s}_m, \boldsymbol{\varepsilon}^i_m|\boldsymbol{\theta})$ on $\boldsymbol{s}_m$.

Consider an economic model characterized by a single equation. The extension to the multiple-equations case is trivial, since equation (11) is a weighted average of the single-equation case, and the gradient is a linear operator. The bc-MC loss function is given by

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

The gradient of $\mathcal{L}_{M,N}$ with respect to the parameter vector $\boldsymbol{\theta}$ is

$$\nabla\mathcal{L}_{M,N}(\boldsymbol{\theta}) = \frac{2}{MN(N-1)} \sum_{m=1}^{M} \sum_{1 \leq i < j}^{N} \nabla f(\boldsymbol{s_m}, \boldsymbol{\varepsilon^i_m}|\boldsymbol{\theta}) f(\boldsymbol{s_m}, \boldsymbol{\varepsilon^j_m}|\boldsymbol{\theta}) + \nabla f(\boldsymbol{s_m}, \boldsymbol{\varepsilon^j_m}|\boldsymbol{\theta}) f(\boldsymbol{s_m}, \boldsymbol{\varepsilon^i_m}|\boldsymbol{\theta})$$

Let us assume that the function f can be decomposed as $f(\boldsymbol{s_m}, \boldsymbol{\varepsilon^i_m}|\boldsymbol{\theta}) = g(\boldsymbol{s_m}, \boldsymbol{\varepsilon^i_m}|\boldsymbol{\theta}) - \boldsymbol{s^T_m}\boldsymbol{\theta}$. In that case, $\nabla f(\boldsymbol{s_m}, \boldsymbol{\varepsilon^i_m}|\boldsymbol{\theta}) = \nabla g(\boldsymbol{s_m}, \boldsymbol{\varepsilon^i_m}|\boldsymbol{\theta}) - \boldsymbol{s_m}$ and one can rewrite the gradient

of the loss as

$$\nabla\mathcal{L}_{M,N}(\boldsymbol{\theta}) = \frac{2}{MN(N-1)} \sum_{m=1}^{M} \sum_{1\le i<j}^{N} \big[\nabla g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta}) - \boldsymbol{s_m}\big] f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^j}|\boldsymbol{\theta}) + \big[\nabla g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^j}|\boldsymbol{\theta}) - \boldsymbol{s_m}\big] f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta})$$

The term inside the double sum can be written as

$$-\boldsymbol{s_m}\big[g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta}) + g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^j}|\boldsymbol{\theta}) - 2\boldsymbol{s_m^T}\boldsymbol{\theta}\big] + \nabla g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta}) f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^j}|\boldsymbol{\theta}) + \nabla g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^j}|\boldsymbol{\theta}) f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta})$$

Hence, the gradient of the loss can be expressed as

$$\begin{aligned}\nabla\mathcal{L}_{M,N}(\boldsymbol{\theta}) =& \frac{1}{M} \sum_{m=1}^{M} -\boldsymbol{s_m} \frac{2}{N(N-1)} \sum_{1\le i<j}^{N} \big[g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta}) + g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^j}|\boldsymbol{\theta}) - 2\boldsymbol{s_m^T}\boldsymbol{\theta}\big] \\ &+ \frac{1}{M} \sum_{m=1}^{M} \frac{2}{N(N-1)} \sum_{1\le i<j}^{N} \nabla g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta}) f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^j}|\boldsymbol{\theta}) + \nabla g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^j}|\boldsymbol{\theta}) f(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta})\end{aligned} \tag{34}$$

Using the fact that $\sum_{1\le i<j}^{N} \big[g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta}) + g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^j}|\boldsymbol{\theta})\big] = (N-1)\sum_{i=1}^{N} g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i})$, the first line of equation (34) can be simplified as

$$\nabla\mathcal{L}_{M,N}^{\text{PEA}}(\boldsymbol{\theta}) = \frac{1}{M} \sum_{m=1}^{M} -2\boldsymbol{s_m}\Big[\Big(\frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta})\Big) - \boldsymbol{s_m^T}\boldsymbol{\theta}\Big] \tag{35}$$

Equation (35) is the gradient for the linear regression of $\frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{s_m}, \boldsymbol{\varepsilon_m^i}|\boldsymbol{\theta})$ on $\boldsymbol{s_m}$.