Journal of Computational Science 33 (2019) 68–82

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Machine learning for high-dimensional dynamic stochastic economies

Simon Scheidegger[a,*], Ilias Bilionis[b]

[a] Department of Finance, Faculty of Business and Economics (HEC), University of Lausanne, Switzerland
[b] Predictive Science Laboratory, School of Mechanical Engineering, Purdue University, USA

ARTICLE INFO

Article history:
Received 4 June 2018
Received in revised form 17 October 2018
Accepted 18 March 2019
Available online 12 April 2019

Keywords:
Machine Learning
Gaussian processes
Active subspaces
Bayesian Gaussian mixture models
Parallel computing
Neoclassical growth
Dynamic programming

ABSTRACT

We present a novel computational framework that can compute global solutions to high-dimensional dynamic stochastic economic models on irregular state space geometries. This framework can also resolve value and policy functions' local features and perform uncertainty quantification, in a single model evaluation. We achieve this by combining Gaussian process machine learning with active subspaces; we then embed this into a parallelized discrete-time dynamic programming algorithm. To demonstrate the broad applicability of our method, we compute solutions to stochastic optimal growth models of up to 500 continuous dimensions. We also show that our framework can address parameter uncertainty and can provide predictive confidence intervals for policies that correspond to the epistemic uncertainty induced by limited data. Finally, we propose an algorithm that, based on combining this framework with Bayesian Gaussian mixture models, is capable of learning irregularly shaped ergodic sets as well as performing dynamic programming on them.

1. Introduction

Many important economic phenomena can only be simulated by models if one goes beyond considering the local dynamics around steady states, and at the same time takes into account the interdependence between different firms, sectors, or countries—that is, a large degree of heterogeneity. Most recently, this has become apparent thanks to the financial crisis of 2008 with its significant price fluctuations and enormous spillover effects. However, substantial heterogeneity and interconnectedness is not a feature unique to macroeconomic problems, but also occurs in many other economic research questions such as in industrial organization, labor economics, and option pricing (see, e.g. [1–3], and references therein). Solving for the global solution of an economic model[1] with substantial heterogeneity is very expensive: using conventional solution methods, the computation time and storage requirements increase exponentially with the amount of heterogeneity—that is, with the dimensionality of the problem. One commonly refers to this phenomenon as the "curse of dimensionality" [5]. Additional complications stem from the fact that natural modeling domains may have highly irregular shapes. Examples include non-hypercubic state spaces and cases in which model parameters are based on empirical data, the measurement uncertainty of which somehow should be propagated through the model. It is therefore often far beyond the scope of current methods to include the heterogeneity and uncertainty necessary for an ideal modeling environment.

Model-based economics has, in many cases, reacted to all these challenges in two ways: either by focusing on qualitative results obtained from simplified models with little heterogeneity (see, e.g., [6]) or by solving the equations that describe the dynamics around a so-called steady state locally (see, e.g., [7]). In contrast, to derive reliable quantitative results or even to test the robustness of these results, there is a consensus that one has to look at non-linearized, high-dimensional models with uncertainty (see, e.g., [2], with references therein). Since the 1990s, a steadily increasing number of economists have started to use global solution methods (see, e.g., [8–11,1,12–14,4,15]). To address the challenges of large-scale dynamic models (see, e.g., [16] and references therein for a thorough review), there have been significant advancements in the development of algorithms and numerical tools. However, at present no solution framework exists that can cope with all of the modeling challenges noted above at once.

In this paper, we present to the best of our knowledge the first framework in computational economics that can compute global solutions to high-dimensional dynamic stochastic economic mod-

* Corresponding author.
E-mail addresses: simon.scheidegger@unil.ch (S. Scheidegger), ibilion@purdue.edu (I. Bilionis).

[1] Following [4], we use the term "global solution" for a solution that is computed using equilibrium conditions at many points in the state space of a dynamic model—in contrast to a "local solution", which rests on a local approximation around a steady state of the model.

https://doi.org/10.1016/j.jocs.2019.03.004
1877-7503/

els on irregularly shaped state spaces and can (to some extent) resolve local features, such as steep gradients that may occur in value and policy functions, as well as perform uncertainty quantification, in a internally-consistent manner—that is, in a single model evaluation. Building on [17], we achieve this by combining Gaussian process regression (GPR) (see, e.g., [18–20]) with the exploitation of active subspaces (AS) [21–24], which we then embed into a parallelized discrete-time dynamic programming algorithm [25–27].

GPR is a form of supervised machine learning [18,19], and can be used to approximate functions with prominent local features. Gaussian processes (GPs) learn a function based on the observations available at so-called design points and without substantial geometric restrictions [28]. We use it in this work to approximate both value and policy functions. The underlying construction of GPR relies on a covariance function, a measure of similarity among input points, which is used to encode one's prior beliefs about an unknown function. Once some observations have been made, Bayes's rule is used to condition the prior GP, resulting in a quantification of our posterior state of knowledge about the unknown function. If a point-wise interpolator is required, one may use the mean of the posterior GP. Furthermore, predictive confidence intervals that depend on the covariance assumptions in the GP model and that correspond to the epistemic uncertainty induced by limited data, can be derived using the variance of the posterior GP. GPs have successfully been applied to a variety of applications in data science, engineering, and other fields to perform regression and classification tasks [18,19]. In particular, Deisenroth and collaborators [29] use GPs in the engineering context in combination with reinforcement learning to solve an optimal control problem of low dimensions. To the best of our knowledge, we are the first to apply GPs to dynamic economic models with substantial heterogeneity, and where—in contrast reinforcement learning problems—the state dynamics are assumed to be known a-priori.

Standard GPs however are not able to deal with dimensions larger than $D \gtrsim 10$ in our target applications. This is due to the fact that they rely on the Euclidean distance to define input-space correlations. Since the Euclidean distance becomes uninformative as the dimensionality of the input space increases [30], the number of observations required to learn the function grows exponentially [31]. Following [17], we alleviate the curse of dimensionality by coupling our framework to the recently developed method of exploiting AS [21–23]. An AS is a linear manifold of the input space that is characterized by maximal response variation. The basic idea is that one should first identify this low-dimensional manifold, then project the high-dimensional input onto it, and finally link the projection to the output. If the dimensionality of the AS is low enough, learning the link function is substantially easier than the original problem of learning a high-dimensional function.[2] In most economic models, there is reasonable hope to believe that such a low-dimensional manifold exists [1].

Our way of approximating and interpolating value and policy functions in the dynamic programming context by GPs is in stark contrast to what has been done in the previous literature to solve high-dimensional dynamic economic models globally. For example, Krueger and Kubler [33], as well as Judd et al. [13] solve dynamic economies using Smolyak sparse grids with global polynomials as basis functions, a construction that can alleviate the curse of dimensionality to some extent. Smolyak's method is capable of dealing with up to approximately 20 continuous dimensions, a state space that does not suffice to address many problems in their full complexity. For instance, Krueger and Kubler [33] consider the welfare implications of social security reform in an overlapping generations model where one period corresponds to six years. With this measure, they reduced the number of adult cohorts and thus the dimensionality of the problem by a factor of six. The second drawback of Smolyak's method is that it cannot capture the local behavior of functions including steep gradients—a feature that is present in many economic models [14,34,35]. To this end, Brumm and Scheidegger [4] recently introduced a parallelized adaptive sparse grid framework that is based on piecewise linear basis functions [36,37] and an adaptive grid refinement strategy (see, e.g., [38,39]). They were able to successfully solve smooth dynamic stochastic models of up to 100 continuous dimensions as well as models with kinks in the policy functions of up to 20 dimensions. However, beyond that model size, the data structures representing adaptive sparse grids become very intricate [40], making it almost infeasible to address models where additional heterogeneity would be required.

Both Smolyak's method and adaptive sparse grids, as well as all other grid-based approximation methods, suffer from a significant common drawback: they rely on a (sparse) tensor product construction—that is, the geometry on which one has to operate is hypercubic. However, there are many economic applications where one cannot trivially map the problem onto this domain, which causes enormous numerical inefficiencies in high-dimensional function spaces. For some overlapping generations models (see, e.g., [41]), as well as for some structural estimation models (see, e.g., [42,43]) the natural modeling domain has a simplex geometry, whereas in some optimal growth models the domain of interest resembles a hypersphere [44,45]. To partially overcome the numerical challenges and inefficiencies imposed by such geometries (if approximated by tensor product-based interpolation schemes) there are some workarounds, such as ergodic-set methods [44,45] where a complex construction merges simulation and projection methods. In contrast, GPR, being a grid-free method, can naturally operate on irregularly shaped domains [28] without any loss of efficiency, as the selection of design points is not confined to any geometry, and thus allows for function approximation and the interpolation on arbitrary state spaces.

GPR is far more powerful than just being a tool for approximating and interpolating functions on irregularly shaped state spaces. Two questions of great interest are, how can (i) arbitrarily shaped ergodic sets be detected, and (ii) dynamic programming be applied to such sets. GPs coupled to Bayesian Gaussian mixture models [46]—a method from unsupervised machine learning—can answer both questions. Firstly, the distribution of the ergodic set can be learned by a simulation. Secondly, this distribution is then used to generate design points from inside the approximate ergodic set, which in turn allows iterative methods like dynamic programming to work efficiently.

To demonstrate the broad applicability of the computational method presented in this paper to deal with high-dimensional dynamic economic problems, we solve a stochastic optimal growth model by a parallelized value function iteration algorithm. In this model, the sector-specific capital stocks are subject to adjustment costs (see, e.g., [47–49], with references therein). Within each iteration step, we use the AS and GP jointly to approximate and interpolate the value function.[3] Using this framework, we solve up to 500-dimensional versions of the stochastic optimal growth model, compared to 4 continuous dimensions as reported by [49].

One key comparative advantage of our algorithm lies in solving models that need to deal with parameter uncertainty. We show

[2] Note that Rust [32] was able to prove that it is possible to formally break the curse of dimensionality for a random multigrid algorithm for discrete choice dynamic programming problems, a setting that substantially differs from the one we are targeting here.

[3] Henceforth, we denote the combination of active subspaces and Gaussian processes as ASGP.

that being able to handle high-dimensional spaces allows us to treat parameters whose distributions are known (e.g., from data) as additional continuous dimensions. By doing so, we provide—to the best of our knowledge—the first framework in economics that can perform uncertainty quantification in a internally-consistent manner: we can directly solve for all possible steady state policies as functions of the economic states and parameters in a single computation, and in turn provide a robust quantitative assessment of how the uncertainty about input parameters influences the model's outcome. The ability to do so has important implications for practitioners interested in calibrating for example financial or macroeconomic models such as dynamic stochastic general equilibrium models (see, e.g., [50], and references therein for a thorough review). Since our method is capable of computing and quantifying all desirable uncertainty in a single model, including confidence intervals, it is now much easier to use data to confront even very complex problems.

To further emphasize the broad applicability of our framework in dynamic economics, we also propose an algorithm that shows how to learn ergodic sets, as well as how to perform dynamic programming on such potentially complicated geometries.

The remainder of this article is organized as follows: In Section 2, we specify the dynamic stochastic economic models we are aiming to solve. In Section 3, we summarize the construction of interpolators by combining GP machine learning with AS. In Section 4, we embed GPs and AS into a dynamic programming framework that is aimed at solving dynamic economic models. Subsequently, we discuss the performance of this algorithm when applied to a stochastic optimal growth model, in Section 5. Section 5 also shows that our framework naturally allows for uncertainty quantification, global sensitivity analysis, and is optimally suited to performing computations on irregularly shaped ergodic sets. The conclusions follow in Section 6.

2. Dynamic stochastic economies

To demonstrate the capabilities of the computational method we introduce in this paper, we now characterize the models we aim to solve formally. To this end, we first outline the general structure that is common to many (infinite-horizon, discrete-time) dynamic economic models. Subsequently, we describe one specific example—namely, a dynamic stochastic optimal growth model.

2.1. Abstract economic models

An infinite-horizon stochastic optimal decision-making problem can be expressed by the following general description: let $x_t \in X \subset \mathbb{R}^D$ denote the state of the economy at time $t \in \mathbb{N}$. Controls are represented by a *policy function* $p : X \to \Xi$, where Ξ is the space of possible controls. The discrete-time transition of the economy from one period to the next is represented by the distribution of x_{t+1}, which depends on the current state and policies

$$x_{t+1} \sim F(\cdot \,|x_t, p(x_t)). \tag{1}$$

While F is given, the policy function p needs to be determined from equilibrium or optimality conditions. A common way of addressing such problems is to use dynamic programming (see, e.g., [5,51,25,52]), where the original problem is to find an infinite sequence of *controls* $\{\xi_t\}_{t=0}^{\infty}$ to maximize the *value function*

$$V(x_0) = \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t r(x_t, \xi_t), \tag{2}$$

where $\beta \in (0, 1)$ is the discount factor, $x_{t+1} \sim F\left(\cdot \,|x_t, \xi_t\right)$, $x_0 \in X$, $r(\cdot, \cdot)$ is the so-called *return function*, and $\xi_t \in \Lambda(x_t) \subset \Xi$, with $\Lambda(x_t)$ denotes the set of feasible choices given x_t (see, e.g., chapter 3 of [52]). Moreover, the expectation value in Eq. (2) is computed with respect to the distribution of x_0. Dynamic programming then seeks a time-invariant policy function p mapping the state x_t into the control ξ_t, such that for all $t \in \mathbb{N}$

$$\xi_t = p(x_t) \in \Lambda(x_t) \tag{3}$$

and $\{\xi_t\}_{t=0}^{\infty}$ solves the original problem (2). The *principle of optimality* states that we can find such a solution by solving the *Bellman equation*

$$V(x) = \max_{\xi}\{r(x, \xi) + \beta \mathbb{E} V[x_{t+1}]\}, \tag{4}$$

where $x_{t+1} \sim F\left(\cdot \,|x, \xi\right)$. Thus, we have exchanged the original problem of finding an infinite sequence of controls that maximizes Eq. (2) for the problem of finding the optimal value function V and a function p that solves the continuum of maximization problems (see Eq. (4)), one for each individual value of x. The task therefore has become to jointly solve for $V(x)$ and $p(x)$. The solution is a fixed point of the Bellman operator $T : C^0(X) \to C^0(X)$, defined by

$$(TV)(x) = \max_{\xi}\{r(x, \xi) + \beta \mathbb{E} V[x_{t+1}]\}. \tag{5}$$

whose maximizer is a policy p. Under certain conditions (see, e.g., chapter 9 of [51]) the Bellman operator is a contraction mapping. In this case, iteratively applying T provides a sequence of value functions that converges to a unique fixed point. This procedure is called Value function iteration (value function iteration; see, e.g. [26], and Section 4.1) and is motivated by this theoretical justification and numerically implements the iterative application of the Bellman operator to successive approximations of the value function. We use value function iteration to solve the economic model described in Section 2.2. Note that alternatively to dynamic programming, the policy function can also be determined from equilibrium conditions that may include, for instance, agents' optimality conditions, budget constraints, and market clearing (see, e.g., [25], Section 17.8). This approach, however, is beyond the scope of this paper. Nevertheless, the framework we describe below could trivially be extended to handle such alternative model formulations.

2.2. A multi-dimensional stochastic optimal growth model

To test our algorithm, we choose an infinite-horizon, discrete-time multi-dimensional stochastic growth model very similar to the one described in [49], which has become one of the workhorses for studying methods for solving high-dimensional economic problems. The stochastic optimal growth model has few parameters and is relatively easy to explain, whereas the dimensionality of the problem can be scaled up in a straightforward but meaningful way, as it depends linearly on the number of sectors considered.

Following Cai and Judd [49], we assume that there are D sectors, and let $\mathbf{k}_t = \left(k_{t,1}, \ldots, k_{t,D}\right)$ denote the capital stocks of these sectors. $\mathbf{k}_t$ is a D-dimensional, continuous state vector at time t. Moreover, let $\mathbf{l}_t = \left(l_{t,1}, \ldots, l_{t,D}\right)$ denote the elastic labor supply levels by a representative household to each of the sectors, a D-dimensional continuous control variable at time t. We assume that the net production function of sector i at time t is $f(k_{t,j}, l_{t,j})$, for $j = 1, \ldots, D$. Let $\mathbf{c}_t = \left(c_{t,1}, \ldots, c_{t,D}\right)$ and $\mathbf{I}_t = \left(I_{t,1}, \ldots, I_{t,D}\right)$ denote consumption and investment of the sectors at time t. The goal now is to find optimal consumption and labor supply decisions such that expected total utility over an infinite time horizon is maximized (cf. Eq. (2)):

$$
\begin{aligned}
&V_0(\mathbf{k}_0) = \max_{\mathbf{k}_t,\mathbf{I}_t,\mathbf{c}_t,\mathbf{l}_t,\Gamma_t} \mathbb{E}\left\{\sum_{t=0}^{\infty}\beta^t \cdot u(\mathbf{c}_t,\mathbf{l}_t))\right\},\\
&s.t.\\
&k_{t+1,j} = (1-\delta)\cdot k_{t,j} + I_{t,j} + \epsilon_{t,j}, \quad j = 1,\ldots,D\\
&\Gamma_{t,j} = \frac{\zeta}{2} k_{t,j}\left(\frac{I_{t,j}}{k_{t,j}} - \delta\right)^2, \quad j = 1,\ldots,D\\
&\sum_{j=1}^{D}\left\{c_{t,j} + I_{t,j} - \delta\cdot k_{t,j}\right\} = \sum_{j=1}^{D}\left\{f(k_{t,j},l_{t,j}) - \Gamma_{t,j}\right\},
\end{aligned}
\tag{6}
$$

where $u(\cdot)$ is the utility function, $\Gamma_{t,j}$ is a convex adjustment cost of sector j, δ is the rate of capital depreciation, β is the discount factor, and $\epsilon_{t,j}$ are all i.i.d. standard normal shocks $\sim\mathcal{N}(0,\sigma^2)$.[4] The respective dynamic programming formulation of the problem then reads (see Eqs. (4) and (5))

$$
\begin{aligned}
&V(\mathbf{k}) = \max_{\mathbf{I},\mathbf{c},\mathbf{l}}\left(u(\mathbf{c},\mathbf{l}) + \beta\mathbb{E}\left\{V_{next}(\mathrm{k}^+)\right\}\right),\\
&s.t.\\
&k_j^+ = (1-\delta)\cdot k_j + I_j + \epsilon_j, \quad j = 1,\ldots,D\\
&\Gamma_j = \frac{\zeta}{2} k_j\left(\frac{I_j}{k_j} - \delta\right)^2, \quad j = 1,\ldots,D\\
&\sum_{j=1}^{D}\left\{c_j + I_j - \delta\cdot k_j\right\} = \sum_{j=1}^{D}\left\{f(k_j,l_j) - \Gamma_j\right\},
\end{aligned}
\tag{7}
$$

where $\mathbf{k} = (k_1,\ldots,k_D)$ represents the state vector, $\mathbf{l} = (l_1,\ldots,l_D)$, $\mathbf{c} = (c_1,\ldots,c_D)$, and $\mathbf{I} = (I_1,\ldots,I_D)$ are $3D$ control variables. $\mathbf{k}^+ = \left(k_1^+,\ldots,k_D^+\right)$ is the vector of next period's variables. Today's and tomorrow's states are restricted to the finite range $[\underline{\mathbf{k}},\bar{\mathbf{k}}]^D$, where the lower edge of the computational domain is given by $\underline{\mathbf{k}} = (\underline{k_1},\ldots,\underline{k_D})$, and the upper bound is given by $\bar{\mathbf{k}} = (\bar{k_1},\ldots,\bar{k_D})$.[5] Moreover, $\mathbf{c} > 0$ and $\mathbf{l} > 0$ holds component-wise.

The detailed parametrization of the stochastic optimal growth model as well as the bounds for computational domain are chosen to be in line with related literature (see [49], with references therein) and is reported in Table 1.

As production function $f_{prod}(\cdot,\cdot)$, and utility function $u(\cdot,\cdot)$, we have

$$
\begin{aligned}
&f_{prod}(k_i,l_i) = A\cdot k_i^{\psi}\cdot l_i^{1-\psi}, \quad \text{and} \quad u(\mathbf{c},\mathbf{l})\\
&\quad = \sum_{i=1}^{D}\left[\frac{(c_i/A)^{1-\gamma}-1}{1-\gamma} - (1-\psi)\frac{l_i^{1+\eta}-1}{1+\eta}\right].
\end{aligned}
\tag{8}
$$

Table 1
Parameterization of the stochastic optimal growth model.

Parameter	Value
β	0.96
δ	0.06
ζ	0.5
$\left[\underline{\mathbf{k}},\bar{\mathbf{k}}\right]^D$	$[0.2, 3.0]^D$
ψ	0.36
A	$(1-\beta)/(\psi\cdot\beta)$
γ	2.0
η	1.0
σ	0.01
D	$\{1,\ldots,500\}$

For the initial guess for the value function, we employ

$$
V^{\infty}(\mathbf{k}) = u\left(f_{prod}(k,\mathbf{e}),\mathbf{e}\right)/(1-\beta)\right), \tag{9}
$$

where $\mathbf{e}$ is the unit vector and is chosen because it is the steady-state labor supply for this model. To evaluate the expectations operator (see Eq. (7)), we follow [4] and use a monomial rule (see [25], Section 7.5).

3. Gaussian processes and active subspaces

No matter how we aim to solve dynamic stochastic economies—that is, by iterating on a Bellman equation (see Eq. (5)) to update the value function, or by iterating on a system of nonlinear equations that represent the equilibrium conditions to update policy functions that represent the economic choices (see, e.g., [25])—one major challenge remains the same: to approximate and interpolate in every iteration step a high-dimensional economic function on potentially complex, irregularly shaped state spaces. We achieve this in the following by the combined usage of GPR [18] and AS [21].[6] In this section, we therefore proceed in two main steps. In Section 3.1, we summarize how functions can be approximated by GP machine learning. In Section 3.2, we show how GPs, in conjunction with the recently developed method of exploiting AS (see, e.g., [21,17], with references therein) can alleviate the curse of dimensionality to some extent.

3.1. Gaussian process regression

Given Eq. (7), we use GPs to approximate the unknown value and policy functions. GPR is a form of supervised machine learning [18,19,54], whose functionality we shall now detail. Our description follows [17]. Let $f:\mathbb{R}^D\to\mathbb{R}$ be a multivariate function of dimensionality D. $f(\cdot)$ accepts an *input*, $\mathbf{x}\in\mathbb{R}^D$, and responds with an *output*, $f(\mathbf{x})$. We can measure $f(\mathbf{x})$ by querying an *information source*, which can either be a computer code—as in our case—or come from empirical data. Moreover, we allow for noisy information sources; that is, we assume that instead of measuring $f(\mathbf{x})$ directly, we may measure a noisy version of it $y = f(\mathbf{x}) + \epsilon$, where ϵ is a random variable.

In the numerical experiments below (see Section 5), information about $f(\cdot)$ comes at the cost of solving many individual, high-dimensional optimization problems. In such settings, we are necessarily restricted to a limited set of observations, as individual function evaluations are computationally expensive. Specifically,

[4] Alternative choices of stochastic disturbances include, for example, autoregressive shocks to total factor productivity (see, e.g., [47,48,4]), or depreciation shocks (see, e.g., [12]). We deliberately follow the less common specification of [49], as it allows us to clearly demonstrate the capabilities of our framework (see, e.g, Section 5.4). Note, however, that other shock specifications do not pose any fundamental technical challenge for our framework.

[5] In practice, it can be challenging to find appropriate bounds for the computational domain. In the absence of any prior knowledge about the bounds for the state space, we generally proceed in three distinct steps for its determination. First, we perform an ex-ante simulation by using the law of motion (cf. the first constraint in Eq. (7)) to obtain some first estimate for the upper and lower edges of the computational domain. Second, we solve the full model (see Algorithm 1) by imposing the ranges that were found in (i). Once the model has converged, we then use the computed policies to simulate the state again (cf. Eq. (7)) to validate ex-post whether the bounds found in (i) were appropriately chosen. If not—that is, if the simulated paths step outside the imposed computational domain, one has to re-adjust it by looking at the realized ranges in the simulation. Steps two and three have to be repeated until the computational domain matches the simulations.

[6] There exist alternative ways to find low-dimensional projections such as "active learning" (see [53]).

assume that we have queried the information source at N input points $\mathbf{X}$, and that we have measured the responses $\mathbf{t}$—that is,

$$\mathbf{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\}, \quad \text{and} \quad \mathbf{t} = \left\{t^{(1)}, \ldots, t^{(N)}\right\}. \tag{10}$$

Note that in the literature, $\mathbf{X}$ is often referred to as *training inputs*, whereas $\mathbf{t}$ is called *training targets* (observations). The core idea is to replace the expensive response surface, $f(\cdot)$, with a cheap-to-evaluate *surrogate* learned from the input-output examples, $\mathbf{X}$ and $\mathbf{t}$, respectively. To this end, we will use GPR.

GPR is a Bayesian regression method that works as follows. Before seeing any data, we model our state of knowledge about $f(\cdot)$ by assigning to it a GP prior. We say that $f(\cdot)$ is a GP with *mean function* $m(\cdot;\boldsymbol{\theta})$ and *covariance function* $k(\cdot,\cdot;\boldsymbol{\theta})$, and write

$$f(\cdot)|\boldsymbol{\theta} \sim GP(f(\cdot)|m(\cdot;\boldsymbol{\theta}), k(\cdot,\cdot;\boldsymbol{\theta})). \tag{11}$$

The parameters of the mean and the covariance function, $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^{d_\theta}$, are known as the *hyper-parameters* of the model. Our prior beliefs about the response are encoded in our choice of the mean and covariance functions. The mean function is required to model any general trends of the response surface, and it can have any functional form. If one does not have any knowledge about the trends in the response, then a reasonable choice is a zero mean function. We follow this approach in our numerical examples in Section 5. The covariance function, also known as the *covariance kernel*, is the most important part of GPR. Intuitively, it defines a nearness or similarity measure on the input space. That is, given two input points, their covariance models how close we expect the corresponding outputs to be. A valid choice for a covariance function must be positive semi-definite and symmetric. One of the most commonly used covariance function is the *square exponential* (see, e.g., [18])

$$k(\mathbf{x}, \mathbf{x}';\boldsymbol{\theta}) = S^2 \exp\left\{-\frac{1}{2}\sum_{i=1}^{D}\frac{\left(x_i - x_i'\right)^2}{\ell_i^2}\right\}, \tag{12}$$

where $\boldsymbol{\theta} = \{S, \ell_1, \ldots, \ell_D\}$, with $S > 0$ being the signal strength and $\ell_i > 0$ the lengthscale of the ith input. We will apply the square exponential kernel throughout paper, as it models our belief that the response is differentiable to any order. For more details on covariance functions with alternative properties, we refer to Ch. 4 of Rasmussen [18].

This prior measure is associated with the observable input-output pairs via a likelihood function (see [18]). Then, we use Bayes's rule to condition this measure on the input–output examples, $\mathbf{X}$, and $\mathbf{t}$. Thus, we obtain a *posterior* probability measure on the space of functions. Its hyper-parameters $\boldsymbol{\theta}$ are typically estimated by maximizing the log-likelihood (see chapter 5 of [18]). This posterior measure is also a GP, where the *posterior* mean $\tilde{m}(\mathbf{x})$ and covariance functions $\tilde{k}(\mathbf{x}, \mathbf{x}')$ are

$$\tilde{m}(\mathbf{x}) := \tilde{m}(\mathbf{x};\boldsymbol{\theta}) = m(\mathbf{x};\boldsymbol{\theta}) + \mathbf{K}(\mathbf{x}, \mathbf{X};\boldsymbol{\theta})\left(\mathbf{K} + S^2\mathbf{I}_N\right)^{-1}(\mathbf{t} - \mathbf{m}), \tag{13}$$

and

$$\begin{aligned}\tilde{k}(\mathbf{x}, \mathbf{x}') &:= \tilde{k}(\mathbf{x}, \mathbf{x}';\boldsymbol{\theta}, S)\\ &= k(\mathbf{x}, \mathbf{x}';\boldsymbol{\theta}) - \mathbf{K}(\mathbf{x}, \mathbf{X};\boldsymbol{\theta})\left(\mathbf{K} + S^2\mathbf{I}_N\right)^{-1}\mathbf{K}(\mathbf{X}, \mathbf{x};\boldsymbol{\theta}),\end{aligned} \tag{14}$$

respectively. $\mathbf{m} := \mathbf{m}(\mathbf{X}, \boldsymbol{\theta}) \in \mathbb{R}^N$ is the mean function evaluated at all points in $\mathbf{X}$. $\mathbf{K} := \mathbf{K}(\mathbf{X}, \mathbf{X};\boldsymbol{\theta}) \in \mathbb{R}^{N\times N}$ is the *covariance matrix*, a special case of the more general *cross-covariance matrix* $\mathbf{K}(\mathbf{X}, \hat{\mathbf{X}};\boldsymbol{\theta}) \in \mathbb{R}^{N\times\hat{N}}$, with $\mathbf{K}_{ij}(\cdot,\cdot,\theta) = k\left(\mathbf{x}^{(i)}, \hat{\mathbf{x}}^{(j)};\boldsymbol{\theta}\right)$, and that is defined between $\mathbf{X}$ and an arbitrary set of $\hat{N}$ inputs $\hat{\mathbf{X}} = \left\{\hat{\mathbf{x}}^{(1)}, \ldots, \hat{\mathbf{x}}^{(\hat{N})}\right\}$.

If a point-wise surrogate is required—that is, if the value or policy functions need to be evaluated at specific points in the state space as in the process of performing dynamic programming—one may use the mean of the posterior GP. Moreover, predictive error bars, corresponding to the epistemic uncertainty induced by limited data can be derived using the variance of the posterior GP.

3.2. Active subspaces

Standard GPs (as well as practically any generic regression method) are not able to deal efficiently with very high input dimensions—that is, $D \gtrsim 10$ in our target applications. This is because they rely on the Euclidean distance to define input-space correlations. Since the Euclidean distance becomes uninformative as the dimensionality of the input space increases [30], the number of observations required to learn the function grows exponentially.

One way of dealing with this problem is to discover and exploit structures that reduce the dimensionality of the input space. Specifically, we assume that the response surface can be well approximated with the following form:

$$f(\mathbf{x}) \approx h\left(\mathbf{W}^T\mathbf{x}\right), \tag{15}$$

where the matrix $\mathbf{W} \in \mathbb{R}^{D\times d}$ projects the high-dimensional input space, $\mathbb{R}^D$, to the low-dimensional *active subspace*, $\mathbb{R}^d$, $d \ll D$, and $h : \mathbb{R}^d \to \mathbb{R}$ is a d-dimensional function known as the *link* function. Note that the representation of Eq. (15) is not unique. All matrices $\mathbf{W}$ whose columns span the same subspace of $\mathbb{R}^D$ yield identical approximations. Thus, without loss of generality, we restrict our attention to matrices with orthogonal columns. An added benefit of enforcing this orthogonality is that the columns of $\mathbf{W}$ correspond to directions of the input space on which the response is most sensitive. Mathematically, we write $\mathbf{W} \in V_d\left(\mathbb{R}^D\right)$, where $V_d\left(\mathbb{R}^D\right)$ is the set of $D \times d$ matrices with orthogonal columns,

$$V_d\left(\mathbb{R}^D\right) := \left\{\mathbf{A} \in \mathbb{R}^{D\times d} : \mathbf{A}^T\mathbf{A} = \mathbf{I}_d\right\}, \tag{16}$$

with $\mathbf{I}_d$ the $d \times d$ unit matrix. $V_d\left(\mathbb{R}^D\right)$ is also known as the *Stiefel manifold* (see, e.g., [55]). Note that the representation of Eq. (15) is arbitrary up to rotations and relabeling of the active subspace coordinate system.

The "classical" approach to discovering the active subspace requires using gradient information [21–23]—that is,

$$\mathbf{G} = \left\{\mathbf{g}^{(1)}, \ldots, \mathbf{g}^{(N)}\right\}, \quad \text{where} \quad \mathbf{g}^{(i)} = \nabla f\left(\mathbf{x}^{(i)}\right) \in \mathbb{R}^D \tag{17}$$

and where $\nabla f(\cdot)$ is the gradient of $f(\cdot)$. In our economic example, these are the gradients of the optimal value of a constrained optimization problem. In A, we show how they can be computed efficiently. The classical approach operates in two steps. First, it identifies the projection matrix $\mathbf{W} \in V_d\left(\mathbb{R}^D\right)$ using gradient information. Second, it projects all inputs to the active subspace and then applies GP regression to learn the map between the projected inputs and the output.

In particular, let $\rho(\mathbf{x})$ be a PDF on the input space such as the PDF of a uniform random variable, and define the matrix

$$\mathbf{C} := \int (\nabla f(\mathbf{x}))(\nabla f(\mathbf{x}))^T \rho(\mathbf{x}) d\mathbf{x}. \tag{18}$$

Since $\mathbf{C}$ is symmetric positive definite, it admits the form

$$\mathbf{C} = \mathbf{V}\Lambda\mathbf{V}^T, \tag{19}$$

where $\Lambda = \text{diag}(\lambda_1, \cdots, \lambda_D)$ is a diagonal matrix containing the eigenvalues of $\mathbf{C}$ in decreasing order, $\lambda_1 \geq \ldots \geq \lambda_D \geq 0$, and $\mathbf{V} \in \mathbb{R}^{D\times D}$ is an orthonormal matrix whose columns correspond to the eigenvectors of $\mathbf{C}$. The classical AS approach suggests separating the d largest eigenvalues from the rest—that is,

$$\Lambda = \begin{bmatrix}\Lambda_1 & 0\\ 0 & \Lambda_2\end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix}\mathbf{V}_1 & \mathbf{V}_2\end{bmatrix}. \tag{20}$$

Here, $\Lambda_1 = \text{diag}(\lambda_1, \ldots, \lambda_d)$, $\mathbf{V}_1 = [\,\mathbf{v}_{11} \quad \ldots \quad \mathbf{v}_{1d}\,]$, and Λ_2, $\mathbf{V}_2$ are defined analogously. Furthermore, one sets the projection matrix to $\mathbf{W} = \mathbf{V}_1$. Intuitively, $\mathbf{V}$ rotates the input space so that the directions associated with the largest eigenvalues correspond to directions of maximal function variability. See [21] for the theoretical justification. It is impossible to evaluate Eq. (18) exactly. Instead, the usual practice is to approximate the integral via Monte Carlo sampling, that is, assuming that the observed inputs are drawn from $\rho(\mathbf{x})$, one approximates $\mathbf{C}$ using the observed gradients by

$$\mathbf{C}_N = \frac{1}{N}\sum_{i=1}^{N} \mathbf{g}^{(i)} \left(\mathbf{g}^{(i)}\right)^T. \tag{21}$$

In practice, the eigenvalues and eigenvectors of $\mathbf{C}_N$ are found using the singular value decomposition [56] of $\mathbf{C}_N$. The dimension d is determined by looking for sharp drops in the spectrum of $\mathbf{C}_N$ [21]. Such drops guarantee that the response surface has an AS and that it can be approximated well.[7] Alternatively, one may use the Bayesian information criterion, which is an approximation to model evidence. For the latter, see [17].

Using the projection matrix based on gradient information, we obtain the projected observed inputs $\mathbf{Z} \in \mathbb{R}^{N\times d}$

$$\mathbf{Z} = \left\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(N)}\right\}, \quad \text{where} \quad \mathbf{z}^{(i)} = \mathbf{W}^T\mathbf{x}^{(i)}. \tag{22}$$

The link function $h(\cdot)$ that connects the active subspace to the output (see Eq. (15)) is identified using GPR, with response $f(\cdot) \equiv h(\cdot)$, input points $\mathbf{x} \equiv \mathbf{z}$, observed inputs $\mathbf{X} \equiv \mathbf{Z}$, and observed outputs $\mathbf{t}$.

4. Active subspace and Gaussian process dynamic programming

We now describe how to solve the stochastic optimal growth model presented in Section 2.2 using ASGP. For this purpose, we build a parallel value function iteration algorithm that applies ASGP (or GP) in each iteration step to learn the value function and, if needed, the policy functions. To this end, we first outline the algorithmic structure of our value function iteration framework. Second, we briefly discuss its parallelization. Third, we define the error measures we apply to check for convergence later in Section 5.

4.1. Value function iteration

The ASGP value function iteration algorithm that we propose for computing the optimal decision rules of dynamic stochastic economic models—and the stochastic optimal growth model in particular—proceeds as follows: We first make a starting guess for a value function V^∞ (see Eq. (9)). At iteration step s, we generate a relatively small set of n^s training inputs[8] in the state space, namely, $\mathbf{k}^s_{1:n}$ on which we evaluate the Bellman operator (see Eq. (5))

$$\mathbf{t}^s_{1:n^s} = \{\mathbf{t}^s_1, \ldots, \mathbf{t}^s_{n^s}\}, \quad \text{with} \quad \mathbf{t}^s_i = TV^{s-1}(k^s_i). \tag{23}$$

Depending on the dimension of the training data, we can now apply either GP or ASGP regression to learn the surrogate of the value function. The predictive mean of the latter is then used to interpolate and update V_{next} (see Eq. (7)). The iterative procedure is then continued until convergence is reached (see Section 4.3). We summarize the detailed steps of the value function iteration in Algorithm 1. Note that at convergence, one can, if desired, also learn the equilibrium policy functions ξ^*, using an individual GP or ASGP per policy.

A particular nice feature when working with GP and ASGP regression is that it allows the practitioner to closely steer the content of the training set. Given the data $\{\mathbf{X}, \mathbf{t}\}$ (see Algorithm 1) consisting of training inputs $\mathbf{k}_i$ and corresponding observations $\mathbf{t}_i$, one can exclude some of those pairs from the training set. This has practical implications: if an individual optimization problem at some particular state $\mathbf{k}_i$ does not converge, one does not need to deal with tuning the optimizer until it converges at this location of the state space. Instead, this training input and target can be discarded—that is to say, it is not added to the training set. This is in stark contrast to grid-based methods such as "Smolyak" [13,33] or "adaptive sparse grids" [4,15], where the construction of the interpolator breaks down if not every optimization problem required by the algorithm can be solved. Furthermore, having the ability to add any data to the training set makes it possible to operate, for example, on (irregularly shaped) ergodic sets. As this is of great interest to practitioners, we will discuss this item in greater detail below in Section 5.4.

Algorithm 1. Value function iteration.

Data: Initial guess V_{next} for the next period's value function and approximation accuracy $\bar{\eta}$.
Result: The (approximate) equilibrium $3D$ policy functions $\xi^* = \{\xi^*_1, \ldots, \xi^*_{3D}\}$ and the corresponding value function V^*.
Set iteration step $s = 1$.
while $\eta > \bar{\eta}$ **do**
  Generate n training inputs $\mathbf{X} = \{\mathbf{k}^s_i : 1 \le i \le n\} \in [\underline{k}, \bar{k}]^D$.
  for $\mathbf{k}^s_i \in \mathbf{X}$ **do**
Compute Eq. (7) given the next period's value function V_{next}.
Set the training targets for the value function: $t_i = V(\mathbf{k}^s_i)$.
If required, set the training targets to learn the policy function $\xi_{j_i}(\mathbf{k}^s_i) \in \text{argmax}_{p_j} V(\mathbf{k}^s_i)$.
  end for
  Set $\mathbf{t} = \{t_i : 1 \le i \le n\}$.
  Given $\{\mathbf{X}, \mathbf{t}\}$, learn a surrogate $V_{surrogate}$ of V with ASGP (or GP).
  Set $\xi_j = \{\xi_{j_i} : 1 \le i \le n\}$.
  Given $\{\mathbf{X}, \boldsymbol{\xi}_j\}$, learn a surrogate of the policy ξ_j with ASGP (or GP).
  Calculate (an approximation for) the error, e.g., $\eta = \| V_{surrogate} - V_{next} \|_\infty$.
  Set $V_{next} = V_{surrogate}$.
  Set $s = s + 1$.
end while
$V^* = V_{surrogate}$.
$\xi^* = \{\xi_1, \cdots, \xi_{3D}\}$.

4.2. Parallelization

In order to solve "large" problems in a reasonably short time, we make use of parallel computation. There are two locations where the value function iteration algorithm described in Section 4.1 can exploit the availability of parallel computing: (i) the evaluation of

[7] Note that principal component analysis (PCA) [57] is probably the most common methodology for reducing the dimensionality of a high-dimensional data set in economics and finance. Here, we briefly comment on the similarities and differences between PCA and AS. Similarly to AS, PCA identifies a projection of the input space. The goal of this projection, however, is not the same as in AS. PCA picks the projection that minimizes the mean square reconstruction error of the input $\mathbf{x}$—that is, it minimizes $\mathbb{E}[\|\mathbf{x} - \mathbf{W}(\mathbf{W}^T\mathbf{x})\|^2]$, where the expectation is with respect to the input-generating distribution. One can show that the mean square reconstruction error is equal to the sum of the eigenvalues of the ignored principal directions. AS has an objective that is very different to that of PCA: AS focuses on finding a $\mathbf{W}$ that allows us to approximate $f(\mathbf{x})$ with a function of the form $h(\mathbf{W}\mathbf{x})$ as well as possible. Even though AS has not (yet) been formulated to directly minimize the mean square error $\mathbb{E}[(f(\mathbf{x}) - h(\mathbf{W}\mathbf{x}))^2]$, it is shown by Constantine [21] that the mean square error is bounded by a term proportional to the sum of the neglected eigenvalues of $\mathbf{C}_N$. In other words, PCA focuses on finding the best linear projection that allows the reconstruction of the input, whereas AS focuses on the search for the best linear projection that enables the reconstruction of the response surface $f(\mathbf{x})$.

[8] In our practical applications, $(4–10)\cdot D$ sample points usually turned out to yield satisfactory results. This observation is in line with Constantine [21], who suggests $(2–10)\cdot D$ sample points for the active subspace method to work well, based on more theoretical foundations.

the Bellman operator (see Eq. (5) and Algorithm 1), and (ii) the training of the GP that represents the function (see Section 3.1). We now outline the basic steps in our parallelization of those sections of the algorithm using the Message Passing Interface (MPI; see, e.g., [58]). For simplicity, assume that we have n_{cpu} computational cores available, which we refer to as "workers" or "processes" (that correspond to individual MPI processes) from this point on. At each iteration step s of the value function iteration algorithm, we broadcast the current value function to all processes such that every process can evaluate the Bellman operator independently. The communication cost required to perform this operation is negligibly small. Then, the collection of the n^s training inputs $\mathbf{t}^s_{1:n^s}$ (see Eq. (23)), becomes embarrassingly parallelizeable. In consequence, each worker simply evaluates the Bellman operator at a fraction of the test points—that is, every worker is assigned with a fractional workload equal to solving n^s/n_{cpu} times. This is where most of the computational time is spent. Subsequently, all the workers gather the distributed data $\mathbf{t}^s_{1:n^s}$. This operation also has a negligible communication cost. The next step is to learn the next step value function using GP regression, which requires maximizing the likelihood of the GP parameters. To avoid being trapped at local likelihood maxima, we implemented a parallelized multi-start optimization algorithm—that is to say, each worker samples random GP parameters and uses them as starting points of a BFGS algorithm [59] for likelihood maximization. The best overall parameters are associated with the next step value function.

4.3. Convergence measure

In the economic applications we consider in this work, the Bellman operator is a contraction mapping [51,25,26]. It is for this reason that we can assess the convergence of the value function iteration by computing error measures after every iteration step s. The average and maximum error is obtained by uniformly generating $N = 10,000$ random test points from the domain $[\underline{\mathbf{k}}, \bar{\mathbf{k}}]^D$ (see Table 1), and then evaluating the expressions below on the constructed surrogates:

$$e^s = \frac{1}{N}\sum_{i=1}^{N} |V^s(\mathrm{x}^i) - V^{s-1}(\mathrm{x}^i)|,$$

$$\text{and} \quad a^s = \max_{i=1,N} |V^s(\mathrm{x}^i) - V^{s-1}(\mathrm{x}^i)|. \tag{24}$$

To assess whether the value function iteration converges, we want either of the two quantities from Eq. (24) to decrease. As we solve an infinite-horizon problem, we stop the iteration at step $s = \omega$ once

$$e^{s=\omega}/\delta < \epsilon \quad \text{or} \quad a^{s=\omega}/\delta < \epsilon_a \tag{25}$$

is reached, and where

$$\delta = \left(\max_{i=1,\dots,N} V^{s=\omega}(\mathrm{x}^i) - \min_{i=1,\dots,N} V^{s=\omega}(\mathrm{x}^i)\right) \tag{26}$$

holds. In practical applications, we typically stop the value function iteration when the normalized average error reaches about 10^{-4}. Note that we normalize our error measures e^s and a^s by δ in order to a enable a comparison of error levels across economic models of different dimensions, since the level of the individual value functions can vary.

5. Numerical experiments

In order to shed light on the broad applicability and versatility of the method introduced in this work, we apply the ASGP dynamic programming framework (see Section 4) now to economic examples—namely, to stochastic optimal growth models (see Section 2.2). We proceed in four steps. First, we consider in Section 5.1 a 1-dimensional version of the stochastic optimal growth model and discuss its convergence with the number of iteration steps and training inputs. Next, we show in Section 5.2 that AS in conjunction with GPs enable us to compute global solutions to dynamic stochastic models of unprecedented dimensionality. Section 5.3 introduces a novel new way of performing uncertainty quantification and global sensitivity analysis for dynamic economic models. In Section 5.4, we propose a simple yet powerful algorithm to efficiently solve high-dimensional dynamic stochastic economic problems on irregularly shaped state spaces.

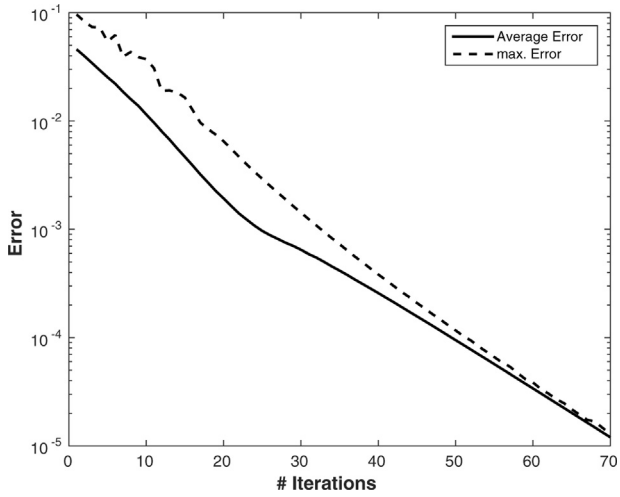

Fig. 1. Decreasing maximum and average error for a 1-dimensional stochastic optimal growth model that was solved by training a GP surrogate with $n^s = 10$ per iteration step.

5.1. Solving a 1-dimensional model with Gaussian processes

To gain a systematic understanding of how the convergence and accuracy of the stochastic optimal growth model behave with regard to the number of training data, we consider for simplicity a 1-dimensional case. In a model with one sector, the ASGP method collapses to the GP case. It is important to study its behavior before we turn our attention to higher-dimensional problems. Fig. 1 depicts the decreasing maximum and average error when performing value function iteration with GPs and $n^s = 10$ training inputs.

In Fig. 2, we display the predictive mean μ of a value function when the value function iteration has converged. They were learned from 4 and 10 training targets, respectively. In addition, we show the 95 percent confidence intervals for each of the value functions. From Fig. 2, it becomes evident that GPs provide a very efficient way of pinning down these nonlinear, smooth functions. With only ten training inputs for example, the uncertainty of the value function becomes negligibly small throughout the entire computational domain, as shown in the right panel of Fig. 2.

Fig. 3 plots the corresponding policy functions with the expected shapes: with an increasing capital stock $\mathbf{k}$, consumption goes up, whereas labor supply and investment go down, due to decreasing returns to capital.

5.2. Solving high-dimensional stochastic optimal growth models

We turn our attention now to high-dimensional stochastic optimal growth models. In order to solve for their global solutions, however, we first need to assess what the dimensionality of the required AS and the related quality of the solution will be. To do so, we compute the global solutions to a 10-dimensional stochastic optimal growth model, once computed with GPs alone, once with ASGP. The left panel of Fig. 4 shows the sorted eigenvalues of the $\mathbf{C}_N$

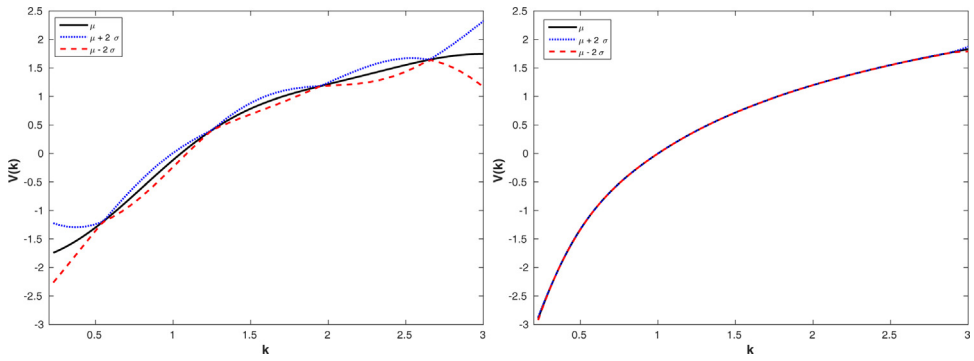

Fig. 2. The above panels display the predictive mean value function of a 1-dimensional growth model at convergence as a function of capital stock k. The 95 percent confidence intervals (corresponding to the point-wise predictive mean μ plus and minus two standard deviations for each input value of k) are also shown. The left panel was constructed based on a value function iteration that used only 4 training inputs per step in the state space, the right used 10 sample points.

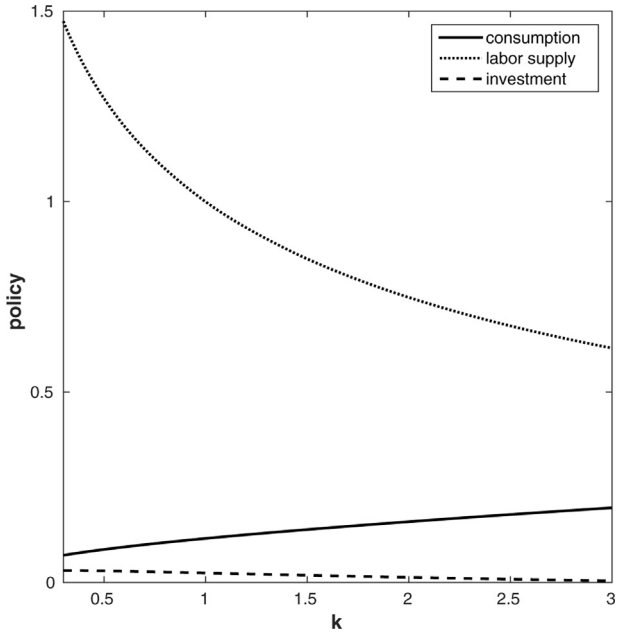

Fig. 3. Policy functions associated with the value function displayed in Fig. 2.

matrix (see Eq. (21)) for the stochastic optimal growth model. The presence of a sharp drop in the spectrum of the sorted eigenvalues, right after the first eigenvalue indicates that we are dealing with a 1-dimensional AS. The right panel of Fig. 4 compares the average errors of two 10-dimensional models, once computed with GP and once with ASGP. We can see that the performance of ASGP is not significantly deteriorated compared to the one of GP.[9]

In the hope that the AS of stochastic optimal growth model is in general low-dimensional, we next increase the dimensionality of the problem up to $D = 500$, while leaving the AS to be 1-dimensional. In Fig. 5, we show that across all multi-dimensional models as before, the average error converges to less than 10^{-4}. For the largest model we solve, every individual observation used to train the ASGP consists of solving an optimization problem with 500 continuous states and 1, 500 controls, which is far beyond what has been done so far in the literature in the context of computing global solutions to economic problems. Cai and Judd [49], for example, who also computed global solutions to this type of stochastic growth model by value function iteration with Chebyshev polynomial approximation were able to solve it for up to 4 continuous (plus 4 discrete) variables—that is, for a state space that is almost two orders of magnitude smaller than ours.

The CPU time increases substantially with the growing dimensionality of the model, as the right panel of Fig. 5 shows. This effect, however, is to be expected and driven by three main factors. First, an increasing amount of observations n^s have to be computed for larger models in every step of the value function iteration procedure (see footnote 9 and Algorithm 1). Second, the number of constraints in the individual optimization problems[10] scales up linearly with dimension, and thus exacerbates the computational burden (cf. Eq. (7)). Third, once the required training data has been generated, the cost for computing an ASGP surrogate also grows with increasing dimensionality.[11] While computing the solution to a 10-dimensional model consumes about 0.01 node hours, roughly ∼3700 node hours are needed for a 500-dimensional stochastic growth model.[12] However, this is still not a roadblock to solving such complex models. CPU time increases less than exponentially in the dimension of the problem. Therefore, we can still compute solutions of high-dimensional models in a reasonable time by employing a large number of nodes. The 500-dimensional model for instances can be tackled with 100 compute nodes, leading to an overall tolerable runtime of less than two days of human time.

Note that adaptive sparse grid-based solution algorithms (see, e.g., [4,61]) as well as algorithms that are based on Smolyak's method [13,33,62] would not be able solve models of this size. Their underlying data structures become so intricate and slow to operate on [40] that they in practice become un-operational for problems of this size. Brumm and Scheidegger [4] were able to compute global solutions for international real business cycle models up to 100 dimensions with adaptive sparse grids and a massively parallelized code, whereas Kruger and Kubler [62] deal with up to 20 continuous states when employing Smolyak's method.

[9] Note that re-running this model with an enforced 2- or 3-dimensional AS does not yield improved results over a 1-dimensional AS.

[10] We solve the individual optimization problems with `Ipopt` [60] (http://www.coin-or.org/Ipopt/).

[11] The computational complexity of constructing an ASGP surrogate is $O(N^3)$ and is driven by two factors. First, standard GP regression is dominated by the need to compute the Cholesky decomposition of the $N \times N$ covariance matrix at each step of the likelihood optimization—that is, it is $O(N^3)$. Second, the computational cost of for finding the AS arises from a singular value decomposition of an $N \times N$ matrix, which is also $O(N^3)$.

[12] Our numerical experiments were carried out on RICE (https://www.rcac.purdue.edu/knowledge/rice/overview). Its compute nodes combine two 10-core Intel Xeon-E5 processors and 64 GB of memory.

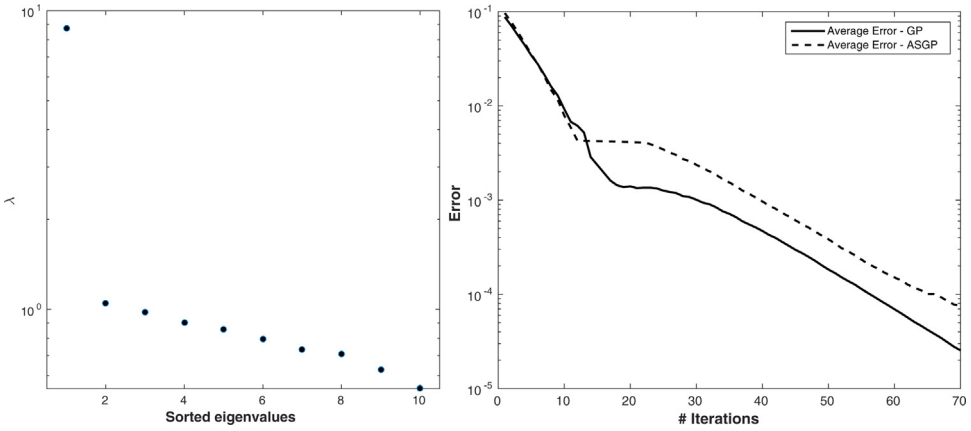

Fig. 4. Left panel—sorted eigenvalues of $\mathbf{C}_N$ of the 10-dimensional stochastic optimal growth model. Right panel—the decreasing average error for the 10-dimensional stochastic optimal growth model, computed either by GPs or ASGP with an AS of dimension 1, respectively.

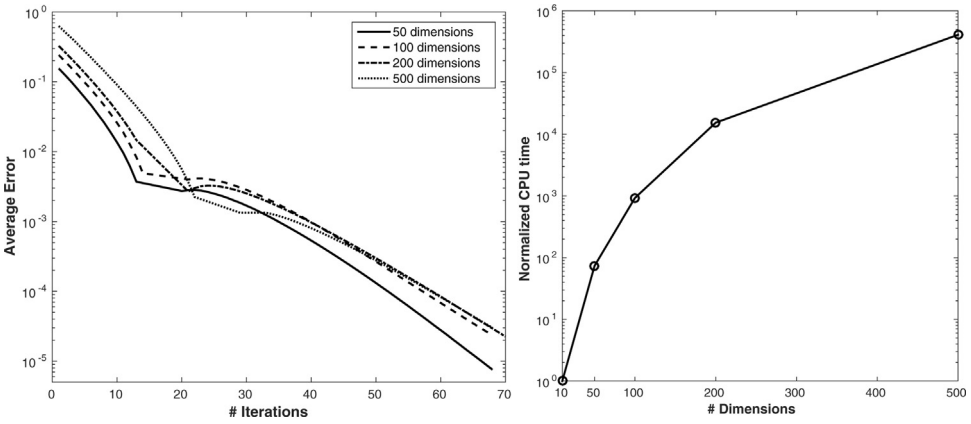

Fig. 5. Left panel—decreasing average error from stochastic optimal growth model ranging from 50 to 500 dimensions. Right panel—CPU time (normalized to the case $d = 10$, which takes about 0.01 h on a single compute node) as a function of the dimensionality of the problems. The y-axis is log-scale. The concave shape of the graph implies that growth in runtime is less than exponential.

5.3. Uncertainty quantification and global sensitivity analysis

Uncertainty quantification is a field of applied mathematics concerned with the quantification of uncertainties and their propagation through computational models [63]. In the quantitative economics community, uncertainty quantification should be of paramount interest, as it can help to address questions such as which parameters are driving the conclusions derived from an economic model [64]. Answering these questions can, in turn, inform the researcher for example on which parts of the model she needs to focus on when calibrating it. However, while the quantification of model uncertainty has had strong advocates in the economic modeling profession for quite some time (see, e.g., [65,66]), uncertainty quantification in general is being largely neglected in the broader community. Cai et al. [67] recently carried out a massive parameter sweep—that is, computed dozens of highly sophisticated models with varying parameters to address the social cost of carbon emission. Harenberg et al. [64] introduce a method to perform global sensitivity analysis based on Sobol indices to assess the robust impact of a parameter on the model's conclusions. In particular, they show that local sensitivity measures typically employed in economics often yield misleading results. Elsewhere in the economics literature, it is however often the case that policy implications are derived based on a single set of model parameters. To this end, we show that our computational framework provides a novel and elegant way to perform uncertainty quantification and global sensitivity analysis for dynamic economic models in a single model evaluation.

There are various sources of uncertainty that can enter economic models, including (i) parameter uncertainty, (ii) interpolation uncertainty, (iii) structural uncertainty, (iv) algorithmic uncertainty, (v) experimental uncertainty, and (vi) parametric variability. We refer the reader to, for example, [68] for more details. Below, we shall concern ourselves only with parameter uncertainty and interpolation uncertainty, as they are most relevant in our context. Interpolation uncertainty stems from the lack of available data collected in the process of solving for the global solution of an economic model by running a computational code. For other input settings that do not have simulation data, one must interpolate or extrapolate to predict the corresponding responses. However in either case, we need to be able to assess how reliable the computed solutions are. In Section 5.1, we have already shown that our algorithm addresses interpolation uncertainty. Using GPs (even in combination with AS)—being probability distributions over functions—can not only return predictive means that are used to interpolate functions, they also carry information about the confidence intervals. Increasing the number of observations in turn reduces the range of the point-wise predictive confidence intervals (cf. Fig. 2). Parameter uncertainty stems from the fact that many parameters that are inputted into the economic models are not known exactly, only their (empirical) distributions are known. Moreover, the mapping from parameters to results is potentially

Table 2
Parameter ranges for the stochastic optimal growth model. We assume that all parameters $\boldsymbol{\chi}_i$ are uniformly distributed in $[\underline{\chi}_i, \bar{\chi}_i]$. Furthermore, we display the original parametrization of the stochastic optimal growth problem as *Baseline value* for completeness (cf. Table 1).

Parameter	Baseline value	Lower bound for $\underline{\chi}_i$	Upper bound for $\bar{\chi}_i$
γ	2.0	0.5	5.0
δ	0.06	0.02	0.06
ψ	0.36	0.2	0.5
ζ	0.5	0.0	1.0
η	1.0	0.0	2.0

highly nonlinear within the parameter ranges. Thus, to provide a robust quantitative assessment of how the uncertainty about input parameters influences the model's outcome, a global approach to performing, for example, sensitivity analysis is necessary [64]. This approach, however, comes at a substantial cost of repeatedly solving the same model over the entire parameter range [67,64].

We, therefore, propose an alternative and substantially more efficient way of dealing with uncertainty quantification in economic models. Instead of frequently recomputing the same model for varying parameters to derive the statistics of interest, we leverage the fact that our framework is capable of dealing with high-dimensional state spaces. In particular, we add the parameters that carry uncertainty as additional, continuous pseudo-states to the model.[13] Enlarging the state space permits us to compute global solutions to all possible realizations that the model could take, in a single run. To do so, we have to define $\boldsymbol{\chi} = (\chi_1, \ldots, \chi_N)$ as the random vector of N input parameters with a joint density f_χ defined over a probabilistic space (see, e.g., [70–77]). Next, we need to specify the distribution for each parameter $\boldsymbol{\chi}_i$. In this paper, we assume that the input parameters are statistically independent. This allows us to factorize the PDF as products of N marginal distributions $f_\chi = \Pi_{i=1}^N f_{\chi_i}$. Furthermore, and in line with [72], we assume that all parameters $\boldsymbol{\chi}_i$ are uniformly distributed with support given by the lower and upper bounds reported in Table 2.

Next, we formally define the mapping

$$\chi \in \mathcal{D}_\chi \subset \mathbb{R}^N \rightarrow y = \mathcal{M}(\chi) \in \mathbb{R}^Q, \tag{27}$$

where y is a so-called "quantity of interest" (QoI)—a random, endogenous outcome of the computational model [64]. As a first example, let us consider the stochastic optimal growth model (see Section 2.2) and assume for simplicity that the entire parameter uncertainty is encoded in one single parameter, namely the risk aversion $\chi_1 = \gamma$. We now extend the D-dimensional state space $\mathbf{k}$ of the model by one continuous variable:

$$\tilde{S} = (\mathbf{k}, \chi_1 = \gamma). \tag{28}$$

Next, we have to slightly adjust the value function iteration algorithm described in Section 4.1. First, instead of sampling n^s training inputs in the D-dimensional state space $[\underline{\mathbf{k}}, \bar{\mathbf{k}}]^D$, we now have to generate training data from the $(D+1)$-dimensional, extended state space $[\underline{\mathbf{k}}, \bar{\mathbf{k}}]^D \times [\underline{\gamma}, \bar{\gamma}]$,where $[\underline{\gamma}, \bar{\gamma}]$ represent the lower and upper bound for the risk aversion, respectively. Second, we evaluate the Bellman operator in iteration step s at the $(D+1)$-dimensional input point—that is, $TV^s(\tilde{S})$. Finally, we learn a $(D+1)$-dimensional surrogate of the value function and—if required—the corresponding policy functions. To exemplify how our framework applies to uncertainty quantification, we consider first an stochastic optimal growth model with one sector and $\gamma \in [0.5, 5]$. The top left panel of Fig. 6 depicts the decreasing maximum and average error of this 2-dimensional model.

In Fig. 6, we also show the predictive mean μ of the value function $V(\mathbf{k}, \gamma)$ when the iteration has converged. The top right panel displays $V(\mathbf{k}, \gamma)$ as well as the 95 percent confidence intervals in the case where 10 observations from the extended state space $\tilde{S}$ were provided to the value function iteration algorithm. The lower left panel reveals the same information but for the case in which the surrogate was trained by using 20 observations. We see that the interpolation uncertainty in the solution is drastically reduced by adding more observation points, as is indicated by the decrease in the point-wise confidence intervals. In the lower right panel of Fig. 6, we show the predictive mean μ of the optimal consumption policy $c(\mathbf{k}, \gamma)$ as well as the 95 percent confidence intervals at convergence in the case where 20 observations were generated from the extended state space $\tilde{S}$. We turn our attention now to higher-dimensional cases to assess whether our method of extending the state space also translates to more complex situations. In particular, we compare two 10-dimensional models with continuous states $\tilde{S} = (k_1, \ldots, k_9, \gamma)$, once computed with GP and once with ASGP. The left panel of Fig. 7 shows the sorted eigenvalues of the matrix $\mathbf{C}_N$, indicating that we are dealing with a 1-dimensional AS. The right panel of Fig. 7 compares the average errors of the two 10-dimensional models. We see that the application of ASGP is fully justified as the model converges nicely.

After having verified, by enlarging the state space, that our framework is capable of performing basic uncertainty quantification tasks, we exemplify next how it can be used to propagate the uncertainty of multiple parameters at once through an economic model. In particular, we want to study univariate effects on the model's output [78,64]. To do so, let us consider an stochastic optimal growth model with one sector and assume that the parametric uncertainty is encoded in the following five parameters (see Table 2 for their ranges),

$\chi = \{\gamma, \delta, \psi, \zeta, \eta\}$, resulting in a 6-dimensional, extended state space $\tilde{S} = (\mathbf{k}, \chi)$. The training data to learn the surrogates for the value function and the policy functions are generated from $[\underline{\mathbf{k}}, \bar{\mathbf{k}}] \times [\underline{\gamma}, \bar{\gamma}] \times [\underline{\delta}, \bar{\delta}] \times [\underline{\psi}, \bar{\psi}] \times [\underline{\zeta}, \bar{\zeta}] \times [\underline{\eta}, \bar{\eta}]$. The QoI we are concerned with in this example is the average production (see Eq. (8)):

$$\mathcal{M}(\chi) = \mathbb{E}[f_{prod}]. \tag{29}$$

In uncertainty quantification, univariate effects are defined as the conditional expectation of the QoI as a function of a single parameter, where the expectations are taken over all other parameters:

$$\mathcal{M}_i(\chi_i) = \mathbb{E}[\mathcal{M}(\Theta|\Theta_i = \chi_i)]. \tag{30}$$

These effects are commonly interpreted as a robust relationship between an input parameter and the QoI (see, e.g., [78,64], and references therein).[14] Fig. 8 displays the univariate effects associated with the five input parameters for average production.

For γ and ψ, the expected average production decreases over the parameter range whereas for δ and ζ, it remains roughly constant and finally for η, it increases. Moreover, most of the parameters influence $\mathcal{M}_i(\chi_i)$ almost linearly. This observation is somehow surprising, as all the parameters (except ζ) enter in a nonlinear fashion into the stochastic optimal growth model. Fig. 8 also reveals that in the example presented here, the QoI is most sensitive to ψ. This finding implies that the researcher would need to pay most attention to this parameter when, for example, calibrating such a model. The above examples confirm that our novel computational framework is ideally suited to efficiently addressing—for the first

[13] Norets [69] estimates finite-horizon, dynamic discrete choice models by using artificial neural networks to approximate the dynamic program solution as a function of the parameters and state variables prior to estimation, a setting that substantially differs from the one we are targeting here.

[14] Univariate effects are of great interest to quantitative economists and policy makers, as they provide information about the direction in which parameters affect the QoI and therefore on the model outcome.

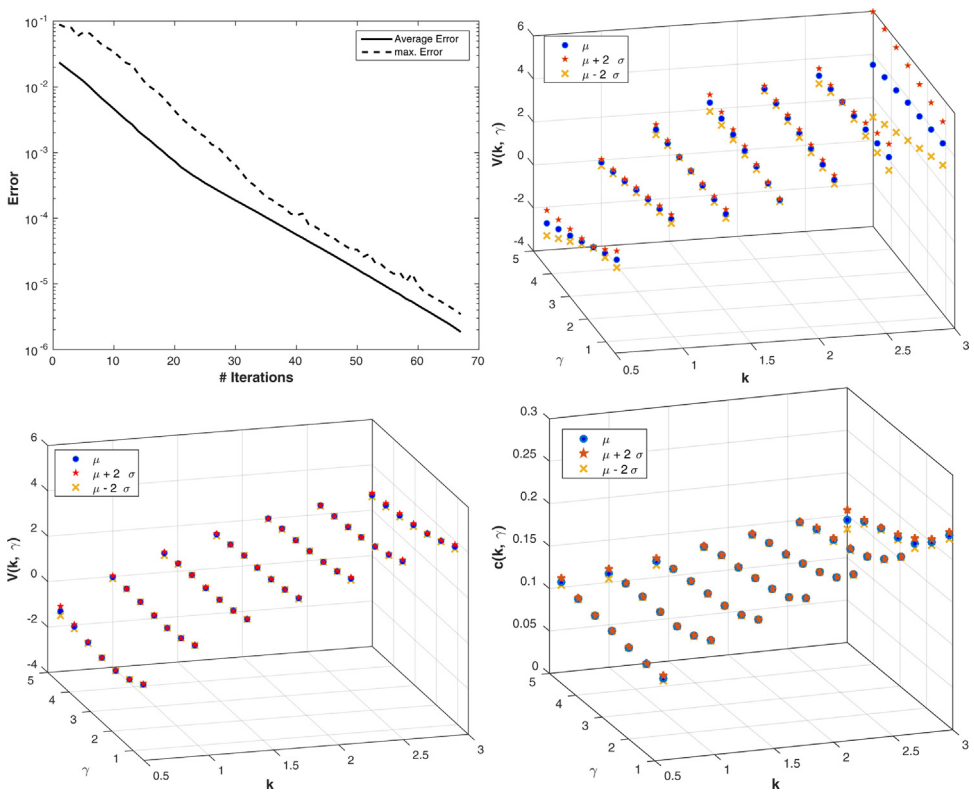

Fig. 6. Top left panel—decreasing maximum and average error for a 2-dimensional stochastic optimal growth model (the continuous states being capital stock and risk aversion) that was solved by training a GP surrogate with 20 training targets per iteration step. Top right panel—the predictive mean μ of $V(\mathbf{k}, \gamma)$ at convergence, evaluated on a uniformly spaced Cartesian grid consisting of 7×7 points. The surrogate was constructed by using 10 observations. Lower left panel—$V(\mathbf{k}, \gamma)$ surrogate at convergence, generated by 20 observations. Lower right panel—consumption $c(\mathbf{k}, \gamma)$ at convergence, generated by 20 training observations. We also show the 95 percent confidence intervals for the value and consumption functions (corresponding to the point-wise predictive mean μ plus and minus two standard deviations) in both panels.

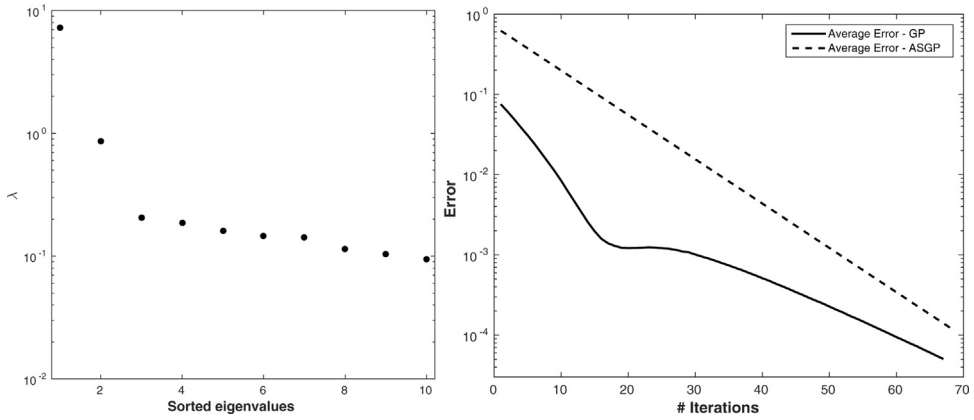

Fig. 7. Left panel—sorted eigenvalues of $\mathbf{C}_N$ (see Eq. (21)) for the 10-dimensional stochastic optimal growth problem with continuous states $\tilde{S} = (k_1, \ldots, k_{10}, \gamma)$. Right panel—the decreasing average error for the two 10-dimensional stochastic optimal growth model, computed either by GPs or by ASGP with an AS of dimension 1, respectively.)

time in economic applications—in a internally-consistent fashion both parameter and interpolation uncertainty in a single model evaluation.[15] Moreover, it avoids tedious calibration exercises. Being able to deal with large parameter uncertainty in the powerful way described above has obviously significant implications for practitioners who are interested in calibrating, for example, financial and macroeconomic models such as so-called dynamic stochastic general equilibrium models (see, e.g., [50]). Since our method is capable of quantifying in a single run all the desirable uncertainty and the model-implied heterogeneity, it is now much easier to use data to confront even a very complex model.

5.4. Comments on ergodic sets

In many economic applications, two interrelated questions of great practical interest are how one can efficiently determine ergodic sets and how one can operate on them—that is, perform dynamic programming (or time iteration) on irregularly

[15] Note that the ability to perform uncertainty quantification as efficiently as proposed here stands in stark contrast to [67,64], who have to repeatedly solve their models to compute, for example, global sensitivity measures. Moreover, both the methods cited are strongly restricted regarding model heterogeneity—that is, in the dimensionality they can handle.

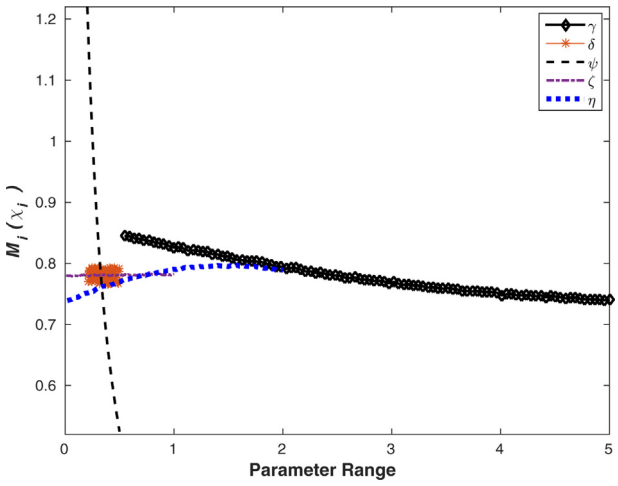

Fig. 8. Univariate effects for the average production as a function of χ_i.

shaped state spaces. Being able to do so has one distinct advantage: computing solutions solely on a domain of relevance allows one to carry out value function iteration on a potentially complex, high-dimensional geometries without suffering from massive inefficiencies. Especially in high-dimensional settings, this can potentially speed up the time-to-solution process by orders of magnitude, as the ergodic set might have a negligibly small volume compared to the computational domain that standard approximation methods require.

We now address the issue of performing value function iteration on irregularly shaped ergodic sets with our computational framework. There is a relatively broad literature on how to determine ergodic sets $\Omega_{ergodic}$ (see, e.g., [79,45,44] and references therein). Den Haan and Marcet [80], Judd et al. [44], and Maliar and Maliar [45] show how to approximate ergodic sets, whose geometries often resemble hyperspheres, by simulation. To perform value function iteration or time iteration on such state spaces, [44,45] merge these simulations with a projection approach. In particular, they construct a costly mapping from the ergodic set to a Smolyak grid, whose geometry is a D-dimensional hypercube of size $[-1,1]^D$. A hypercube, however, is obviously not an optimal computational domain for efficiently approximating hyperspherical state spaces. We therefore propose merging their simulation-based approach with GP or ASGP within value function iteration as follows: First, we solve the economic model at few points in a given time step s (see Section 2.2) on a "sufficiently" large domain (see Table 1) to learn the respective value function and policies. Second, following ideas of Judd et al. [44] and Maliar and Maliar [45], we use the computed surrogate of the policies to simulate the economy by using the law of motion for capital (see Eq. (7)),

$$k_j^+ = (1-\delta)\cdot k_j + I_j + \epsilon_j, \quad j = 1,\ldots,D, \tag{31}$$

in order to numerically generate an estimate for the ergodic set $\Omega_{ergodic}$. In contrast to [44,45], however, we now construct a probability density from the computed distribution of capital by a method from unsupervised machine learning: Bayesian Gaussian mixture models [46]. Suppose that we have n samples $\mathbf{X}_{ergodic} = \{\mathbf{k}_i^+ : 1 \le i \le n\}$ obtained, for example, using Eq. (31). We then can approximate $\rho_{estimated}$ as a mixture of Gaussians:

$$\rho_{estimated}(\mathbf{x}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m), \tag{32}$$

where the mean vectors $\boldsymbol{\mu}_m \in \mathbb{R}^D$, the covariance matrices $\boldsymbol{\Sigma}_m \in \mathbb{R}^{D\times D}$, the weights π_m with $\sum_{m=1}^{M}\pi_m = 1$, and the number of components M are fitted to $\mathbf{X}_{ergodic}$. Fig. 9 shows an example distribution of 1000 simulated capital stocks from a 2-sector stochastic optimal growth model that was overlapped with 1000 samples generated from the respective density $\rho_{estimated}$.

From Fig. 9, one can clearly see that the ergodic set has a geometry similar to that of a hypersphere (or an ellipse) rather than to that of a hypercube. Recall from Section 3 that GPs work irregularly shaped geometries as long as there exists a method that can provide training inputs and observations. We therefore propose to subsequently draw N samples $\mathbf{X}_{ergodic}$ from $\rho_{estimated}$ within a value function iteration step and observe the respective training targets—that is:

$$\mathbf{X}_{ergodic} = \left\{\mathbf{x}_{ergodic}^{(1)}, \ldots, \mathbf{x}_{ergodic}^{(N)}\right\},$$
$$\text{and} \quad \mathbf{t}_{ergodic} = \left\{t_{ergodic}^{(1)}, \ldots, t_{ergodic}^{(N)}\right\}. \tag{33}$$

Next, we train a surrogate by using the data $\{\mathbf{X}_{ergodic}, \mathbf{t}_{ergodic}\} \in \Omega_{ergodic}$. Once the corresponding surrogate is constructed, we can carry out one value function iteration step on the ergodic set with a very complex, high-dimensional state space and without suffering from major computational inefficiencies. For illustrative purposes, we display in the right panel of Fig. 9 100 points in the state space that were drawn from the approximate ergodic set of a 2-dimensional stochastic optimal growth problem at convergence. We contrast them by showing the same number of sample points that were generated from a uniform distribution of the original parameter range (see Table 1). The right panel shows the con-

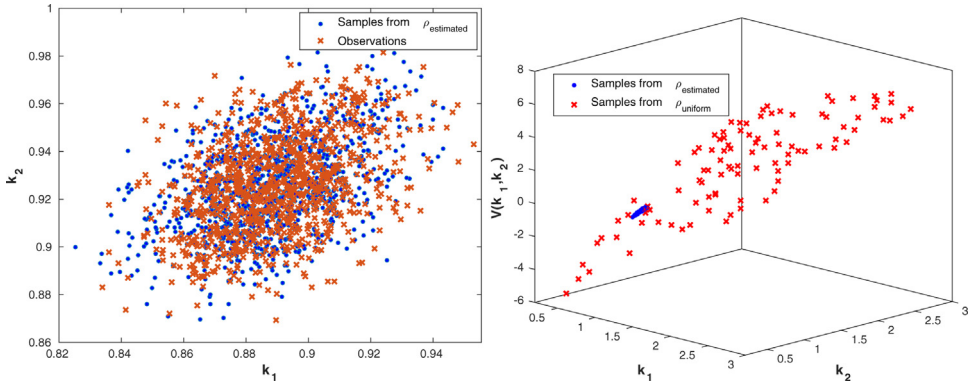

Fig. 9. Left panel—scatterplot of the observed ergodic distribution of capital (see Eq. (31)) from a 2-dimensional stochastic optimal growth model, contrasted with states that were randomly generated from $\rho_{estimated}$ (see Eq. (32)). A total of 1000 samples are shown in either case for illustrative purposes. Right panel—100 sample points for capital, once generated uniformly from $\rho_{uniform}$—that is, a cubic domain (cf. Table 1), and once from $\rho_{estimated}$ (cf., Eq. (32) displays the respective value function evaluated on the same sample points.

verged value function, evaluated again at 100 sample points from the original parameter range and from the ergodic distribution, respectively. From this 2-dimensional example, it becomes evident that a simulation-based approach, being embedded into our ASGP framework, is capable of concentrating computational resources where they are needed—that is, on a computational domain $\Omega_{ergodic}$ that is substantially smaller than the original, hypercubic one (cf. Table 1).

6. Conclusion

In this paper, we present to the best of our knowledge the first computational framework that can compute global solutions to high-dimensional dynamic stochastic economic models on irregularly shaped state spaces, can resolve prominent local features in value and policy functions such as steep gradients, and can perform uncertainty quantification, in a single model evaluation. We achieve this by combining GP machine learning with the AS method, which we then embed into a parallelized discrete-time dynamic programming algorithm.

GPR is a form of supervised learning that can be used to approximate and interpolate functions on irregularly shaped state spaces. Furthermore, it can be used to derive point-wise predictive confidence intervals. However, conventional GPs in our target applications cannot deal with dimensions larger than $D \gtrsim 10$.

To address this shortcoming, we couple our framework to AS. An AS is a linear manifold of the input space that is characterized by maximal response variation. The underlying idea of AS is to (i) identify this low-dimensional manifold, (ii) project the high-dimensional input onto it, and (iii) link the projection to the output. If the dimensionality of the active subspace is low enough, learning the link function is much easier than the original problem of learning a high-dimensional function.

We apply this algorithm first to stochastic optimal growth models with convex adjustment costs and do so for up to 500 continuous dimensions. Second, we show that being able to handle high-dimensional spaces allows us to treat parameters whose distributions are known (e.g., from data) as additional continuous dimensions. By doing so, we provide the first framework in computational economics that can perform uncertainty quantification in a internally-consistent manner: we can directly solve for all possible steady state policies as functions of economic states and parameters, and do so in a single computation, thus abandoning the "axiom of correct specification". Thus far, it has not been possible to deal with large parameter uncertainty in dynamic economic models in such an efficient fashion. This has important implications for practitioners interested in calibrating models. Since our method is capable of computing and quantifying all the desirable uncertainty in a single model, including confidence intervals to policies, it is now much easier to use data to confront even very complex problems. Third, to further emphasize the broad applicability of our framework in dynamic economics, we also propose a method that shows how to learn irregularly shaped ergodic sets in high-dimensional settings with Bayesian Gaussian mixture models, as well as how to perform dynamic programming on such potentially complex geometries.

This all suggests that our novel, parallelized, scalable, and flexible framework is very well suited for computing global solutions to large-scale economic models that can now include far more heterogeneity and uncertainty than was previously possible.

Acknowledgments

The authors would like to thank Johannes Brumm, Daniel Harenberg, Xavier Gabaix, Ken Judd, Michel Juillard, Felix Kubler, Harry Paarsch, Gregor Reich, Philipp Renner, John Rust, Olaf Schenk, Karl Schmedders, Tony Smith and seminar participants at the Banque de France, the University of Zurich, the IMF, Carnegie Mellon Tepper School of Business, Stanford University, the University of Lausanne, EPFL Lausanne, PASC 16, and the University of Copenhagen for their extremely valuable comments. Simon Scheidegger gratefully acknowledges support from the Hoover Institution at Stanford University, PASC, and the Cowles Foundation at Yale University. Moreover, this work was supported by a grant from the Swiss National Supercomputing Centre (CSCS) under project IDs s555, s790, and s885. Ilias Bilionis acknowledges the startup support provided by the School of Mechanical Engineering at Purdue University.

Appendix A. Determination of gradients

The ASGP methodology requires the ability to differentiate the Jacobi-Bellman operator with respect to the state of the system. This is not a trivial task, as it involves differentiating the optimal value of a constrained optimization problem. We start by proving a Lemma that shows how to differentiate the policy function.

Lemma 1. *Consider the optimization problem:*

$$\max_u h(x,u) \quad \text{subjectto} \quad g_i(x,u) \le 0, \quad i = 1,\dots,n_c, \tag{A.1}$$

where $n_c \in \mathbb{N}$, and assume the regularity assumptions of the Karush–Kuhn–Tucker (KKT) theorem [81] for each $x \in \mathcal{X} \subset \mathbb{R}^{d_x}$, with $\mathcal{X}$ open, and h, g_i are sufficiently smooth (twice differentiable suffices). Let $\mu : \mathcal{X}$ be the solution to the optimization problem, $\mathcal{A}(x)$ be the set of active constraints,

$$\mathcal{A}(x) = \left\{ i \in \{1,\dots,n_c\} : g_i(x,\mu(x)) = 0 \right\}, \tag{A.2}$$

$m = |\mathcal{A}(x)|$ be the number of elements in set $\mathcal{A}(x)$, $i_1,\dots,i_m$ an enumeration of the elements in set $\mathcal{A}(x)$, $g^a(x,u) = (g_{i_1}(x,u),\dots,g_{i_m}(x,u))$ be a vector function corresponding to the active constraints, $\lambda^a(x) = (\lambda_{i_1}(x),\dots,\lambda_{i_m}(x))$ be the Lagrange multipliers of the active constraints. Then, the gradients $\nabla_x \mu(x) \in \mathbb{R}^{d_u \times d_x}$ and $\nabla_x \lambda^a(x) \in \mathbb{R}^{|\mathcal{A}(x)| \times d_x}$ can be found by solving the following linear system of equations:

$$\begin{pmatrix} \nabla_u^2 \mathcal{L} & -(\nabla_u \nabla_\lambda \mathcal{L})^T \\ -\nabla_u \nabla_\lambda \mathcal{L} & 0 \end{pmatrix} \begin{pmatrix} \nabla_x \mu \\ \nabla_x \lambda^a \end{pmatrix} = \begin{pmatrix} -\nabla_x \nabla_u \mathcal{L} \\ -\nabla_x \nabla_\lambda \mathcal{L} \end{pmatrix}, \tag{A.3}$$

where all functions are evaluated at $u = \mu(x)$, and

$$\mathcal{L}(x,u) = h(x,u) - \sum_{j=1}^{m} \lambda_{i_j}^a g_{i_j}(x,u). \tag{A.4}$$

Proof. Under some regularity assumptions, the Karush–Kuhn–Tucker (KKT) conditions must be satisfied. That is, if the point $\mu(x)$ is a local optimum, then there exist constants $\lambda_i(x) \ge 0$, $i = 1,\dots,n_c$ s.t.

$$\nabla_u h(x,\mu(x)) = \sum_{i=1}^{n_c} \lambda_i(x) \nabla_u g_i(x,\mu(x)), \tag{A.5}$$

$$g_i(x,\mu(x)) \le 0, \; i = 1,\dots,n_c, \tag{A.6}$$

and:

$$\lambda_i g(x,\mu(x)) = 0, \; i = 1,\dots,n_c. \tag{A.7}$$

Almost surely, there is an open neighborhood of x in which $\mathcal{A}(x)$ remains constant. Then, for all $i \notin \mathcal{A}(x)$ we have from Eq. (A.7) that $\lambda_i(x) = 0$. Using, this information we may write:

$$\frac{\partial h(x, \mu(x))}{\partial u_j} = \sum_{i \in \mathcal{A}(x)} \lambda_i(x) \frac{\partial g_i(x, \mu(x))}{\partial u_j}, j = 1 \ldots, d_u, \tag{A.8}$$

$$g_i(x, \mu(x)) = 0, \forall i \in \mathcal{A}(x), \tag{A.9}$$

and we may differentiate both expressions with respect to x. Differentiating the first one with respect to x_k, we have:

$$\frac{\partial^2 h(x, \mu(x))}{\partial x_k \partial u_j} + \sum_{s=1}^{d_u} \frac{\partial^2 h(x, \mu(x))}{\partial u_s \partial u_j} \frac{\partial \mu_s(x)}{\partial x_k} = \sum_{i \in \mathcal{A}(x)} \left\{ \frac{\partial g_i(x, \mu(x))}{\partial u_j} \frac{\partial \lambda_i(x)}{\partial x_k} + \lambda_i(x) \frac{\partial^2 g_i(x, \mu(x))}{\partial x_k \partial u_j} + \lambda_i(x) \sum_{s=1}^{d_u} \frac{\partial^2 g_i(x, \mu(x))}{\partial u_s \partial u_j} \frac{\partial \mu_s(x)}{\partial x_k} \right\}.$$

Differentiating the second one with respect to x_k, we have:

$$\frac{\partial g_i(x, \mu(x))}{\partial x_k} + \sum_{s=1}^{d_u} \frac{\partial g_i(x, \mu(x))}{\partial u_s} \frac{\partial \mu_s(x)}{\partial x_k} = 0.$$

The result follows after re-arranging terms. □

Using the Lemma, we can easily prove the following corollary which can be directly associated to the derivative of the Jacobi-Bellman operator.

Corollary 1. *Consider the optimization problem given in Eq. (A.1) and let $h^*(x)$ be the optimal value, i.e.,*

(A.10)$h^*(x) = h(x, \mu(x))$,

$$h^*(x) = h(x, \mu(x)), \tag{A.10}$$

under the notation of Lemma 1. Then, the derivative of h^* with respect to x_k is:

$$\nabla_x h^*(x) = \nabla_x h(x, \mu(x)) + \nabla_x \mu(x) \nabla_u h(x, \mu(x)). \tag{A.11}$$

References

[1] P. Krusell, A.A. Smith Jr., Income and wealth heterogeneity in the macroeconomy, J. Polit. Econ. 106 (5) (1998) 867–896.
[2] J. Brumm, F. Kubler, S. Scheidegger, Computing Equilibria in Dynamic Stochastic Macro-Models with Heterogeneous Agents, Vol. 2 of Econometric Society Monographs, Cambridge University Press, 2017, pp. 185–230.
[3] Scheidegger, Simon, Treccani, Adrien, Pricing American Options under High-Dimensional Models with Recursive Adaptive Sparse Expectations, 2018, http://dx.doi.org/10.1093/jjfinec/nby024;
S. Scheidegger, A. Treccani, Pricing American options under high-dimensional models with recursive adaptive sparse expectations, J. Finan. Econometr., Available at SSRN 2867926 (forthcoming).
[4] J. Brumm, S. Scheidegger, Using adaptive sparse grids to solve high-dimensional dynamic models, Econometrica 85 (5) (2017) 1575–1612.
[5] R. Bellman, Adaptive Control Processes: A Guided Tour, 'Rand Corporation. Research Studies, Princeton University Press, 1961.
[6] J. Bengui, E.G. Mendoza, V. Quadrini, Capital mobility and international sharing of cyclical risk, J. Monetary Econ. 60 (1) (2013) 42–62.
[7] H. Uhlig, A Toolkit for Analysing Nonlinear Dynamic Stochastic Models Easily, 1998.
[8] C.I. Telmer, Asset-pricing puzzles and incomplete markets, J. Finan. 48 (5) (1993) 1803–1832.
[9] D.J. Lucas, Asset pricing with undiversifiable income risk and short sales constraints: deepening the equity premium puzzle, J. Monetary Econ. 34 (3) (1994) 325–341.
[10] M.P. Keane, K.I. Wolpin, The solution and estimation of discrete choice dynamic programming models by simulation and interpolation: Monte carlo evidence, Rev. Econ. Stat. 76 (4) (1994) 648–672.
[11] R. Aiyagari, Uninsured idiosyncratic risk and aggregate saving, Q. J. Econ. 109 (3) (1994) 659–684.
[12] D. Krueger, F. Kubler, Pareto-improving social security reform when financial markets are incomplete!? Am. Econ. Rev. 96 (3) (2006) 737–755.
[13] K.L. Judd, L. Maliar, S. Maliar, R. Valero, Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain, J. Econ. Dyn. Control 44 (2014) 92–123.
[14] J. Brumm, M. Grill, Computing equilibria in dynamic models with occasionally binding constraints, J. Econ. Dyn. Control 38 (2014) 142–160.
[15] S. Scheidegger, D. Mikushin, F. Kubler, O. Schenk, Rethinking large-scale economic modeling for efficiency: optimizations for GPU and Xeon phi clusters, 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS) (2018) 610–619, http://dx.doi.org/10.1109/IPDPS.2018.00070.
[16] L. Maliar, S. Maliar, Numerical methods for large-scale dynamic economic models, in: K. Schmedders, K.L. Judd (Eds.), Handbook of Computational Economics, Vol. 3 of Handbook of Computational Economics, Elsevier, 2014, pp. 325–477 (Chapter 7).
[17] R. Tripathy, I. Bilionis, M. Gonzalez, Gaussian processes with built-in dimensionality reduction: applications to high-dimensional uncertainty propagation, J. Comput. Phys. 321 (2016) 191–223.
[18] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning), The MIT Press, 2005.
[19] K.P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, 2012.
[20] E. Snelson, Z. Ghahramani, Sparse Gaussian processes using pseudo-inputs, in: Y. Weiss, B. Schölkopf, J.C. Platt (Eds.), Advances in Neural Information Processing Systems 18, MIT Press, 2006, pp. 1257–1264.
[21] P.G. Constantine, Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2015.
[22] T.W. Lukaczyk, P. Constantine, F. Palacios, J.J. Alonso, Active subspaces for shape optimization, 10th AIAA Multidisciplinary Design Optimization Conference (2014) 1171.
[23] E. Dow, Q. Wang, Output based dimensionality reduction of geometric variability in compressor blades, 51st AIAA Aerospace Sciences Meeting (2013).
[24] P.G. Constantine, E. Dow, Q.Q. Wang, Active subspace methods in theory and practice: applications to Kriging surfaces, SIAM J. Sci. Comput. 36 (6) (2014) A3030–A3031.
[25] K.L. Judd, Numerical Methods in Economics, The MIT Press, 1998.
[26] D.P. Bertsekas, Dynamic Programming and Optimal Control, 2nd ed., Athena Scientific, 2000.
[27] J. Rust, Dynamic Programming, Numerical, John Wiley & Sons, Ltd, 2014.
[28] H. Wendland, Scattered Data Approximation, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2004.
[29] M.P. Deisenroth, C.E. Rasmussen, J. Peters, Gaussian process dynamic programming, Neurocomputing 72 (7) (2009) 1508–1524.
[30] Y. Bengio, O. Delalleau, N. Le Roux, The curse of highly variable functions for local kernel machines, Neural Information Processing Systems (2005).
[31] J. Traub, A. Werschulz, Complexity and Information, Lezioni Lincee, Cambridge University Press, 1998.
[32] J. Rust, Using randomization to break the curse of dimensionality, Econometrica 65 (3) (1997) 487–516.
[33] D. Krueger, F. Kubler, Computing equilibrium in OLG models with stochastic production, J. Econ. Dyn. Control 28 (7) (2004) 1411–1436.
[34] L.J. Christiano, J.D. Fisher, Algorithms for solving dynamic models with occasionally binding constraints, J. Econ. Dyn. Control 24 (8) (2000) 1179–1232.
[35] T. Hintermaier, W. Koeniger, The method of endogenous gridpoints with occasionally binding constraints among endogenous variables, J. Econ. Dyn. Control 34 (10) (2010) 2074–2088.
[36] C. Zenger, Sparse grids, in: W. Hackbusch (Ed.), Parallel Algorithms for Partial Differential Equations, Vol. 31 of Notes on Numerical Fluid Mechanics, Vieweg, 1991, pp. 241–251.
[37] H.-J. Bungartz, M. Griebel, Sparse grids, Acta Numer. 13 (2004) 1–123.
[38] D. Pflüger, Spatially adaptive sparse grids for high-dimensional problems, Ph.D. thesis, München, 2010.
[39] X. Ma, N. Zabaras, An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations, J. Comput. Phys. 228 (8) (2009) 3084–3113.
[40] A. Murarasu, J. Weidendorfer, G. Buse, D. Butnaru, D. Pflüeger, Compact data structure and parallel alogrithms for the sparse grid technique, 16th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (2011).
[41] J. Brumm, F. Kubler, Applying Negishi's Method to Stochastic Models with Overlapping Generations, Working Paper, University of Zurich, 2013.
[42] J. Rust, Optimal replacement of GMC bus engines: an empirical model of Harold Zurcher, Econometrica 55 (5) (1987) 999–1033.
[43] C.-L. Su, K.L. Judd, Constrained optimization approaches to estimation of structural models, Econometrica 80 (5) (2012) 2213–2230.
[44] S. Maliar, L. Maliar, K.L. Judd, Solving the multi-country real business cycle model using ergodic set methods, Working Paper 16304, National Bureau of Economic Research, 2010.
[45] L. Maliar, S. Maliar, Merging simulation and projection approaches to solve high-dimensional problems with an application to a new Keynesian model, Quantit. Econ. 6 (1) (2015) 1–47.
[46] C.E. Rasmussen, The infinite Gaussian mixture model, Advances in Neural Information Processing Systems 12 (2000) 554–560.
[47] W.J.D. Haan, K.L. Judd, M. Juillard, Computational suite of models with heterogeneous agents. ii: Multi-country real business cycle models, J. Econ. Dyn. Control 35 (2) (2011) 175–177.
[48] M. Juillard, S. Villemot, Multi-country real business cycle models: accuracy tests and test bench, J. Econ. Dyn. Control 35 (2) (2011) 178–185.

[49] Y. Cai, K.L. Judd, Advances in numerical dynamic programming and new applications, Handb. Comput. Econ. 3 (2014).
[50] J. Fernandez-Villaverde, J.F. Rubio-Ramirez, F. Schorfheide, Solution and estimation methods for DSGE models, CEPR Discussion Papers 11032, C.E.P.R. Discussion Papers (2015).
[51] N.L. Stokey, R.E. Lucas Jr., E.C. Prescott, Recursive Methods in Economic Dynamics, Harvard University Press, Cambridge, MA, 1989.
[52] L. Ljungqvist, T. Sargent, Recursive Macroeconomic Theory, MIT Press, 2000.
[53] R. Garnett, M.A. Osborne, P. Hennig, Active learning of linear embeddings for Gaussian processes, in: Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, UAI'14, AUAI Press, Arlington, Virginia, United States, 2014, pp. 230–239, URL http://dl.acm.org/citation.cfm?id=3020751.3020776.
[54] C.M. Bishop, Pattern recognition and machine learning, in: Information Science and Statistics, Springer, New York, 2006.
[55] I. James, C.U. Press, N. Hitchin, The Topology of Stiefel Manifolds, Cambridge Books Online, Cambridge University Press, 1976, URL https://books.google.ch/books?id=9ss7AAAAIAAJ.
[56] G.H. Golub, C.F. Van Loan, Matrix Computations, 3rd ed., Johns Hopkins University Press, Baltimore, MD, USA, 1996.
[57] S. Wold, K. Esbensen, P. Geladi, Principal component analysis, Chemom. Intell. Lab. Syst. 2 (1–3) (1987) 37–52.
[58] A. Skjellum, W. Gropp, E. Lusk, Using MPI, MIT Press, 1999.
[59] R.H. Byrd, P. Lu, J. Nocedal, C. Zhu, A limited memory algorithm for bound constrained optimization, SIAM J. Sci. Comput. 16 (5) (1995) 1190–1208.
[60] A. Waechter, L.T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Math. Program. 106 (1) (2006) 25–57, http://dx.doi.org/10.1007/s10107-004-0559-y.
[61] J. Brumm, D. Mikushin, S. Scheidegger, O. Schenk, Scalable high-dimensional dynamic stochastic economic modeling, J. Comput. Sci. 11 (2015) 12–25.
[62] B.A. Malin, D. Krüger, F. Kübler, Solving the multi-country real business cycle model using a Smolyak-collocation method, J. Econ. Dyn. Control 35 (2) (2010) 229–239.
[63] C. Smith, Ralph, Uncertainty Quantification: Theory, Implementation, and Applications, SIAM, 2014.
[64] D. Harenberg, S. Marelli, B. Sudret, V. Winschel, Uncertainty quantification and global sensitivity analysis for economic models, 2017 (Available at SSRN 2908760).
[65] F.E. Kydland, On the econometrics of world business cycles, Eur. Econ. Rev. 36 (2) (1992) 476–482.
[66] L.P. Hansen, J.J. Heckman, The empirical foundations of calibration, J. Econ. Perspect. 10 (1) (1996) 87–104.
[67] Y. Cai, K.L. Judd, T.S. Lontzek, The social cost of carbon with economic and climate risks, in: ArXiv e-prints, 2015, arXiv:1504.06909.
[68] R.C. Smith, Uncertainty Quantification: Theory, Implementation, and Applications, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2013.
[69] A. Norets, Estimation of dynamic discrete choice models using artificial neural network approximations, Econometr. Rev. 31 (1) (2012) 84–106.
[70] E.T. Jaynes, Information theory and statistical mechanics. II, Phys. Rev. 108 (2) (1957) 171–190.
[71] E.T. Jaynes, Information theory and statistical mechanics, Phys. Rev. 106 (4) (1957) 620–630.
[72] E.T. Jaynes, On the rationale of maximum-entropy methods, Proc. IEEE 70 (9) (1982) 939–952.
[73] E.T. Jaynes, Prior probabilities, IEEE Trans. Syst. Sci. Cybern. 4 (3) (1968) 227–241.
[74] J.O. Berger, J.M. Bernardo, D. Sun, The formal definition of reference priors, Ann. Stat. (2009).
[75] A. Tarantola, Inverse Problem Theory and Methods for Model Parameter Estimation, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2004.
[76] I. Bilionis, N. Zabaras, Solution of inverse problems with limited forward solver evaluations: a Bayesian perspective, Inverse Probl. 30 (1) (2014).
[77] I. Bilionis, B.A. Drewniak, E.M. Constantinescu, Crop physiology calibration in the CLM, Geosci. Model Dev. 8 (4) (2015) 1071–1083.
[78] A. Younes, T.A. Mara, N. Fajraoui, F. Lehmann, B. Belfort, H. Beydoun, Use of global sensitivity analysis to help assess unsaturated soil hydraulic parameters, Vadose Zone J. 12 (1) (2013).
[79] A. Fernandes, C. Phelan, A recursive formulation for repeated agency with history dependence, J. Econ. Theory 91 (2) (2000) 223–247.
[80] W.J.D. Haan, A. Marcet, Accuracy in simulations, Rev. Econ. Stud. 61 (1) (1994) 3–17.
[81] H.W. Kuhn, A.W. Tucker, Nonlinear programming, 2nd Berkeley Symposium on Mathematical Statistics and Probability (1983).

Simon Scheidegger is currently an Assistant Professor at the Department of Finance, Faculty of Business and Economics (HEC), University of Lausanne, Switzerland.

He has been a visiting fellow at the Hoover Institution (Stanford University), and senior research associate at the University of Zurich, Department of Finance.

His research focuses on developing computational methods for high-dimensional dynamic stochastic economic modeling and applying them to macroeconomics, monetary policy, option pricing, and optimal tax policy.

He currently is a principal investigator of a variety of projects related to high-performance computing as well as co-principal investigator on a PASC project (platform for advanced scientific computing).

Scheidegger was a credit risk modeler at Credit Suisse from 2010–2012. He received a BSc (2005), a MSc in physics (2007), and a PhD (2010) in theoretical physics, summa cum laude, from the University of Basel.

Ilias Bilionis is an Assistant Professor at the School of Mechanical Engineering, Purdue University, USA.

As part of his professorship, Ilias Bilionis provides leadership in the predictive science area. His research is motivated by energy and material science applications. It focuses on the development of generic methodologies for design and optimization under uncertainty, reliability analysis, model calibration, and learning models out of data.

Before his appointment at Purdue he was a postdoctoral researcher at the Mathematics and Computer Science Division at Argonne National Laboratory.

He received his PhD in applied mathematics from Cornell University in 2013 and his diploma in applied mathematics from the National Technical University of Athens in 2008.