Machine learning for high-dimensional dynamic stochastic economies *

Simon Scheidegger
Department of Banking and Finance, University of Zurich
and Hoover Institution, Stanford University
simon.scheidegger@uzh.ch

Ilias Bilionis
Predictive Science Laboratory
School of Mechanical Engineering
Purdue University
ibilion@purdue.edu

June 23, 2017

Abstract

We present the first computational framework that can compute global solutions to very-high-dimensional dynamic stochastic economic models on arbitrary state space geometries. This framework can also resolve value and policy functions' local features and perform uncertainty quantification, in a self-consistent manner. We achieve this by combining Gaussian process machine learning with the active subspace method; we then embed this into a massively 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, based on this framework, that is capable of learning irregularly shaped ergodic sets as well as performing dynamic programming on them.

Keywords: Machine Learning, Gaussian Process, Active Subspaces, High-Performance Computing, Neoclassical Growth, Dynamic Programming.

JEL Classification: C63, C68, E130

*We thank Johannes Brumm, Daniel Harenberg, Ken Judd, Michel Juillard, Felix Kubler, Harry Paarsch, Gregor Reich, Philipp Renner, John Rust, Karl Schmedders 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, and PASC 16 for their extremely valuable comments. Simon Scheidegger gratefully acknowledges support from the Hoover Institution and PASC. Moreover, this work was supported by a grant from the Swiss National Supercomputing Centre (CSCS) under project ID s555. Ilias Bilionis acknowledges the startup support provided by the School of Mechanical Engineering at Purdue University.

1 Introduction

Many important economic phenomena can only be captured 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. Most recently, this has become apparent thanks to the financial crisis of 2008 with its significant price fluctuations and enormous spillover effects. However, solving for the global solution of a 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" [3, 4].

Additional complications stem from the fact that natural modeling domains may have very complex—that is, irregularly shaped—state spaces, and from the fact that 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 choice. Model-based economics has, for the most part, 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., [85]). 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., [19], with references therein). Since the 1990s, a steadily increasing number of economists have started to use global solution methods (see, e.g., [1, 17, 21, 46, 49, 50, 55, 83]). To address the challenges of large-scale dynamic models (see, e.g., [59] 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 the first computational framework that can compute global solutions to very-high-dimensional dynamic stochastic economic models on arbitrary 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 (UQ), in a self-consistent manner—that is, in a single model evaluation. We achieve this by combining Gaussian process regression (GPR) [65, 72] with the active subspace (AS) method [27, 30, 56], which we then embed into a massively parallelized discrete-time dynamic programming algorithm [3, 8, 28, 45, 73, 76].

GPR is a form of supervised machine learning [65, 72], and can be used to interpolate or approximate functions with prominent local features. We use it in this work to approximate both policy and value 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 Gaussian process (GP), resulting in a quantification of our posterior state of knowledge about the unknown function. If a point-wise surrogate[2] is required, one may use the mean of the posterior GP. Furthermore, predictive confidence intervals, corresponding 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 [65, 72]. In particular, Deisenroth and collaborators [28] use GPs in the engineering context 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.

[1] Following [21], 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. For a method that computes such a global solution, we use the term "global solution method". This use of the word "global" is not to be confused with its use in the term "global optimization method", which refers to a method that aims to find a global optimum.

[2] In the literature on GPs, the terms "interpolator" and "surrogate" are used interchangeably.

GPR combines the best of two worlds—namely, those of grid-free methods such as Monte Carlo (MC) [70] and of smooth function approximation theory. GPs learn a function based on the observations available at so-called design points. This is in stark contrast to ordinary, grid-based approximation schemes [70]. Second, since the construction of GP surrogates is achieved by sampling, they do not directly suffer from the curse of dimensionality, as do Cartesian grid-based algorithms. The latter usually start with a 1-dimensional discretization scheme that employs M grid points; a straightforward extension to D dimensions leads to M^D points, and thus, the total number of points grows exponentially in the dimension. This curse is a prohibitive obstacle if one wants to numerically deal with models of dimensionality D greater than three to four.[3] Standard GPs, on the other hand (as well as practically any generic regression method), are not able to deal with dimensions larger than $D \gg 20$. 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 [5], the number of observations required to learn the function grows enormously. We, therefore, alleviate the curse of dimensionality by coupling our framework to the recently developed AS method [27, 30, 56]. 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. Lastly, the GP surrogate can be made infinitely differentiable ($\mathbf{C}_\infty$). Note that differentiability can be controlled by the choice of the covariance function, and therefore does not suffer from the noise that is common to standard MC approximation methods [70]. Having a smooth function approximation is particularly helpful when solving models by dynamic programming, for which derivatives from the value and policy functions repeatedly need to be fed into an optimization routine.

Our way of approximating and interpolating functions by a probabilistic method is in stark contrast to what has been done in the previous literature to solve high-dimensional dynamic models globally. For example, Krueger and Kubler [48], as well as Judd et al., [46] 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 [48] 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 [17, 26, 40]. To this end, Brumm and Scheidegger [21] recently introduced a highly parallelized adaptive sparse grid framework that is based on piecewise linear basis functions [22, 89] and an adaptive grid refinement strategy (see, e.g., [57, 68]). 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 [64], 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[4], which causes enormous numerical inefficiencies in high-dimensional func-

[3] Note that Rust [75] 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.

[4] An exception is "adaptive simplicial" interpolation, as developed in [17]. This method can, however, deal with non-hypercubic geometries only in low-dimensional state spaces–that is, up to about four continuous dimensions.

tion spaces. For some overlapping generations models (see, e.g., [18]), as well as for some structural estimation models (see, e.g., [74, 81]) the natural modeling domain has a simplex geometry, whereas in some optimal growth models the domain of interest resembles a hypersphere [60, 61]. 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 [60, 61] where a complex construction merges simulation and projection methods. In contrast, GPR, being a grid-free method, can naturally operate on arbitrary domains 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, if coupled to infinite Gaussian model mixture [71], can deal with this. Firstly, the distribution of the ergodic set is learned by 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.

To demonstrate the broad applicability of this method to high-dimensional dynamic economic problems, we solve a stochastic optimal growth (SG) model by a highly parallelized value function iteration algorithm. In this model, the sector-specific capital stocks are subject to adjustment costs (see, e.g., [25, 36, 47], with references therein). Within each iteration step, we use the AS and GP jointly to approximate and interpolate the value function.[5] Using this framework, we solve up to 500-dimensional versions of the stochastic SG model, compared to 4 continuous dimensions as reported by [25].

One key comparative advantage of our algorithm lies in solving models that need to deal with parameter uncertainty. We show that being able to handle very-high-dimensional spaces allows us to treat parameters whose distributions are known (e.g., from data) as additional continuous dimensions.[6] By doing so, we provide—to the best of our knowledge—the first framework in economics that can perform uncertainty quantification in a self-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 DSGE models (see, e.g., [32], 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 outline the construction of surrogates by GP machine learning and AS, supported by a variety of illustrative test cases. In Section 4, we embed GPs and AS into a dynamic programming framework that is aimed at solving 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, and is optimally suited to performing computations on irregularly shaped ergodic sets. Section 6 concludes.

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

[6] Norets [66] uses artificial neural networks to approximate the dynamic program solution as a function of the parameters and state variables in finite-horizon, dynamic discrete choice models.

2 Dynamic stochastic economies

To demonstrate the capabilities of the 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

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

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., [3, 45, 54, 80]), 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 $x_{t+1} \sim F\left(\cdot | x_t, \xi_t\right)$, $x_0 \in \mathbb{R}$, $r(x_t, \dot{)}$ is the so-called *return function*, and $\xi_t \in \Gamma(x_t)$ with $\Gamma(x_t)$ denotes the set of feasible choices given x_t. Dynamic programming then seeks a time-invariant policy function p mapping the state x_t into the control ξ_t, such that for all $t \in \mathbb{N}$

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

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

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

where $x_{t+1} \sim F\left(\cdot | x, \xi\right)$. The solution is a fixed point of the Bellman operator T, defined by

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

Under certain conditions (see, e.g., [80]) 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 (VFI; see, e.g. [8] and Sec. 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 VFI to solve the economic model described in Sec. 2.2.[7]

[7] 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. Jointly, these conditions constitute a functional equation that the policy function p has to satisfy:

$$0 = \mathbb{E}\left\{E\Big(x_t, x_{t+1}, p\left(x_t\right), p\left(x_{t+1}\right)\Big) | x_t, p(x_t)\right\}, \tag{6}$$

where the expectation, represented by the operator $\mathbb{E}$, is taken with respect to the distribution $F\left(\cdot | x_t, p(x_t)\right)$ of next period's state x_{t+1}. The function $E : X^2 \times Y^2 \to \mathbb{R}^{2D}$ is then given by the period-to-period equilibrium conditions of the model. Formally, and analogously to the dynamic programming case, Eq. (6) can again be solved iteratively, by a so-called time-iteration procedure (see, e.g., [45], Sec. 17.8). This, however, is beyond the scope of this paper. Nevertheless, the framework we describe below could trivially be extended to time iteration.

2.2 A multi-dimensional stochastic optimal growth model

To test our algorithm, we choose an infinite-horizon, discrete-time multi-dimensional SG model very similar to the one described in [25], which has become one of the workhorses for studying methods for solving high-dimensional economic problems.[8] The SG 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 [25], we assume that there are D sectors, and let $\mathbf{k}_t = (k_{t,1}, ..., k_{t,D})$ denote the capital stocks of these sectors. $\mathbf{k}_t$ is a $D-$dimensional, continuous state vector at time t. Moreover, let $\mathbf{l}_t = (l_{t,1}, ..., l_{t,D})$ denote the elastic labor supply levels 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 $j = 1, ..., D$. Let $\mathbf{c}_t = (c_{t,1}, ..., c_{t,D})$ and $\mathbf{I}_t = (I_{t,1}, ..., I_{t,D})$ 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)):

$$
\begin{aligned}
V_0(\mathbf{k}_0) = \max_{\mathbf{k}_t, \mathbf{I}_t, \mathbf{c}_t, \mathbf{l}_t, \mathbf{\Gamma}_t} & \mathbb{E}\Big\{ \sum_{t=0}^{\infty} \beta^t \cdot u(\mathbf{c}_t, \mathbf{l}_t)) \Big\}, \\
& s.t. \\
k_{t+1,j} &= (1-\delta) \cdot k_{t,j} + I_{t,j} + \epsilon_{t,j}, \quad j = 1, ..., D \\
\Gamma_{t,j} &= \frac{\zeta}{2} k_{t,j} \left(\frac{I_{t,j}}{k_{t,j}} - \delta \right)^2, \quad j = 1, ..., D \\
\sum_{j=1}^{D} (c_{t,j} + I_{t,j} - \delta \cdot k_{t,j}) &= \sum_{j=1}^{D} \left(f(k_{t,j}, l_{t,j}) - \Gamma_{t,j} \right),
\end{aligned}
\tag{7}
$$

where $u(\cdot)$ is the utility function, $\mathbf{\Gamma}_t = (\Gamma_{t,1}, , ..., \Gamma_{t,D})$ 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 $\sim \mathcal{N}(0, \sigma^2)$.[9] The respective dynamic programming formulation of the problem then reads (see Eqs. (4) and (5))

$$
\begin{aligned}
V(\mathbf{k}) = \max_{\mathbf{I}, \mathbf{c}, \mathbf{l}} & \left(u(\mathbf{c}, \mathbf{l}) + \beta \mathbb{E}\Big\{ V_{next}(\mathbf{k}^+) \Big\} \right), \\
& s.t. \\
k_j^+ &= (1-\delta) \cdot k_j + I_j + \epsilon_j, \quad j = 1, ..., D \\
\Gamma_j &= \frac{\zeta}{2} k_j \left(\frac{I_j}{k_j} - \delta \right)^2, \quad j = 1, ..., D \\
\sum_{j=1}^{D} (c_j + I_j - \delta \cdot k_j) &= \sum_{j=1}^{D} \left(f(k_j, l_j) - \Gamma_j \right),
\end{aligned}
\tag{8}
$$

[8] Note that in the version of the model we are solving, we avoid discrete shocks to productivity for simplicity. Adding them, however, would not pose any problems, as it would just mean adding more discrete value functions of the same dimensionality. Having, for example, a Markov chain for productivity with N discrete states would increase the computational burden by $N\times$ and could easily be mitigated by using $N\times$ more CPUs (see Sec. 4.2 and [19]).

[9] Alternative choices of stochastic disturbances include, for example, autoregressive shocks to total factor productivity (see, e.g., [21, 36, 47]), or depreciation shocks (see, e.g., [49]). We deliberately follow the less common specification of [25], as it allows us to clearly demonstrate the capabilities of our framework (see, e.g, Sec. 5.4 below). Note, however, that other shock specifications do not pose any fundamental technical challenge for our framework.

Parameter	Value
β	0.96
δ	0.06
ζ	0.5
$\left[\underline{\mathbf{k}}, \overline{\mathbf{k}}\right]^D$	$[0.2, 3.0]^D$
ψ	0.36
A	$(1-\beta)/(\psi \cdot \beta)$
γ	2
η	1
σ	0.01
D	{1,...,500}

Table 1: Parameterization of the SG model.

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

2.2.1 Parameterization and implementation details

The detailed parametrization of the SG is chosen to be in line with related literature (see, e.g., [21, 25, 29], with references therein) and is reported in Tab.1. As production function, we have

$$f(k_i, l_i) = A \cdot k_i^{\psi} \cdot l_i^{1-\psi}, \tag{9}$$

whereas for the utility function we use

$$u(\mathbf{c}, \mathbf{l}) = \sum_{i=1}^{d} \left[\frac{(c_i/A)^{1-\gamma} - 1}{1-\gamma} - (1-\psi) \frac{l_i^{1+\eta} - 1}{1+\eta} \right]. \tag{10}$$

For the initial guess for the value function, we employ

$$V^{\infty}(\mathbf{k}) = u\left(f(k, \mathbf{e}, \mathbf{e}), \mathbf{e}\right)/(\mathbf{1} - \beta), \tag{11}$$

where $\mathbf{e}$ is the unit vector and is chosen because it is the steady-state labor supply for this model.

One important detail of the implementation of this model is the integration procedure used to evaluate the expectations operator (see, e.g., Eq. (8)). As we want to focus on our core contribution—that is, solving very heterogeneous dynamic problems with ASGP, we chose an integration rule that is simple and fast, yet not very accurate. In particular, we use a monomial rule that uses two evaluation points per shock—that is to say, $2D$ points in total (see, e.g., [45], Sec. 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 Eq. (6)), one major challenge remains the same: to approximate and interpolate in every iteration step a (potentially) very-high-dimensional economic function on complex—that is, irregularly shaped state spaces. We achieve this in the following by the combined usage of GPR [72] and AS [27].

In this section, we therefore proceed in two main steps. In Sec. 3.1, we summarize how functions can be approximated by GP machine learning, even if the state space of the function is non-hypercubic, such as the simplex that we often face in economic applications. In Sec. 3.2, we show how GPs, in conjunction with the recently developed AS method (see, e.g., [27], with references therein) can alleviate the curse of dimensionality. Additionally, we provide illustrative analytic examples to demonstrate the joint workings of GPs and AS.

3.1 Gaussian process regression

GPR is a form of supervised machine learning [15, 65, 72], whose functionality we shall now detail. Our description follows [84]. 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 real data, acquired, for example, in an empirical experiment or from financial markets. 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 empirical setups, measurement noise may arise from our inability to control all the influential factors or from irreducible (aleatory) uncertainties. In computer simulations, measurement uncertainty may stem from quasi-random stochasticity, or chaotic behavior.

In the numerical experiments below (see Sec. 5), information about $f(\cdot)$ comes at the cost of solving many individual, high-dimensional optimization problems (see Sec. 2). In such settings, we are necessarily restricted to a limited set of observations, as individual function evaluations are computationally expensive. Specifically, assume that we have queried the information source at N input points,

$$\mathbf{X} = \left\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\right\}, \tag{12}$$

and that we have measured the following responses:

$$\mathbf{t} = \left\{t^{(1)}, \dots, t^{(N)}\right\}. \tag{13}$$

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 Gaussian process regression (GPR).

GPR is a Bayesian regression method that works as follows. We start by defining a probability measure on the function space, where $f(\cdot)$ lives corresponding to our prior beliefs (see Sec. 3.1.1). This prior measure is associated with the observable input-output pairs via a likelihood function (see Sec. 3.1.2). 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. This posterior measure is also a GP (see Sec. 3.1.3 and Sec. 3.1.4). 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, 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.

3.1.1 Prior state of knowledge

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

$$f(\cdot)|\boldsymbol{\theta} \sim \mathrm{GP}(f(\cdot)|m(\cdot;\boldsymbol{\theta}), k(\cdot,\cdot;\boldsymbol{\theta})). \tag{14}$$

The parameters of the mean and the covariance function, $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbb{R}^{d_\theta}$, 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 (see Secs. 3.1.5, 3.2.5, and 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. The most commonly used covariance function is the *square exponential* (SE)

$$k_{\mathrm{SE}}(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = s^2 \exp\left\{ -\frac{1}{2} \sum_{i=1}^{D} \frac{(x_i - x_i')^2}{\ell_i^2} \right\}, \tag{15}$$

where $\boldsymbol{\theta} = \{s, \ell_1, \ldots, \ell_D\}$, with $s > 0$ being the signal strength and $\ell_i > 0$ the lengthscale of the i-th input. We will apply it throughout this paper. The SE covariance function models the a priori belief that the response is infinitely differentiable ($\mathbf{C}_\infty$). For more details on covariance functions see, for example, Ch. 4 of Rasmussen [72].

Given an arbitrary set of inputs $\mathbf{X}$, Eq. (14) induces a Gaussian prior on the corresponding response outputs:

$$\mathbf{f} = \left\{ f\left(\mathbf{x}^{(1)}\right), \ldots, f\left(\mathbf{x}^{(N)}\right) \right\}. \tag{16}$$

Specifically, $\mathbf{f}$ is a priori distributed according to

$$\mathbf{f} | \mathbf{X}, \boldsymbol{\theta} \sim \mathcal{N}\left(\mathbf{f} | \mathbf{m}, \mathbf{K}\right), \tag{17}$$

where $\mathcal{N}\left(\cdot | \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$ is the PDF of a multivariate Gaussian random variable with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, $\mathbf{m} := \mathbf{m}(\mathbf{X}; \boldsymbol{\theta}) \in \mathbb{R}^N$ is the mean function evaluated at all points in $\mathbf{X}$,

$$\mathbf{m} = \mathbf{m}(\mathbf{X}; \boldsymbol{\theta}) = \begin{pmatrix} m\left(\mathbf{x}^{(1)}; \boldsymbol{\theta}\right) \\ \vdots \\ m\left(\mathbf{x}^{(N)}; \boldsymbol{\theta}\right) \end{pmatrix}, \tag{18}$$

and $\mathbf{K} := \mathbf{K}(\mathbf{X}, \mathbf{X}; \boldsymbol{\theta}) \in \mathbb{R}^{N \times N}$ is the *covariance matrix*, a special case of the more general *cross-covariance matrix* $\mathbf{K}(\mathbf{X}, \hat{\mathbf{X}}; \boldsymbol{\theta}) \in \mathbb{R}^{N \times \hat{N}}$,

$$\mathbf{K}(\mathbf{X}, \hat{\mathbf{X}}; \boldsymbol{\theta}) = \begin{pmatrix} k\left(\mathbf{x}^{(1)}, \hat{\mathbf{x}}^{(1)}; \boldsymbol{\theta}\right) & \ldots & k\left(\mathbf{x}^{(1)}, \hat{\mathbf{x}}^{(\hat{N})}; \boldsymbol{\theta}\right) \\ \vdots & \ddots & \vdots \\ k\left(\mathbf{x}^{(N)}, \hat{\mathbf{x}}^{(1)}; \boldsymbol{\theta}\right) & \ldots & k\left(\mathbf{x}^{(N)}, \hat{\mathbf{x}}^{(\hat{N})}; \boldsymbol{\theta}\right) \end{pmatrix}, \tag{19}$$

defined between $\mathbf{X}$ and an arbitrary set of $\hat{N}$ inputs $\hat{\mathbf{X}} = \left\{ \hat{\mathbf{x}}^{(1)}, \ldots, \hat{\mathbf{x}}^{(\hat{N})} \right\}$.

3.1.2 Measurement process

The Bayesian formalism demands that we explicitly model the measurement process that gives rise to the observations $\mathbf{t}$ of Eq. (13). The simplest way to do so is to assume that measurements are independent of each other, and that they are normally distributed about $f(\cdot)$ with variance s_n^2; that is,

$$t^{(i)} | f\left(\mathbf{x}^{(i)}\right), s_n \sim \mathcal{N}\left(t^{(i)} \middle| f\left(\mathbf{x}^{(i)}\right), s_n^2\right). \tag{20}$$

Note that $s_n > 0$ is another hyper-parameter that must be determined from the data. The assumptions in Eq. (20) can be relaxed to allow for heteroscedastic (input dependent) noise

[34, 69], but this is beyond the scope of this work. Using the independence of the observations, we get

$$\mathbf{t}|\mathbf{f}, s_n \sim \mathcal{N}\left(\mathbf{t}\middle|\mathbf{f}, s_n^2\mathbf{I}_N\right). \tag{21}$$

Via the sum rule of probability theory and standard properties of Gaussian integrals, we can derive the *likelihood* of the observations given the inputs

$$\mathbf{t}|\mathbf{X}, \boldsymbol{\theta}, s_n \sim \mathcal{N}\left(\mathbf{t}\middle|\mathbf{m}, \mathbf{K} + s_n^2\mathbf{I}_N\right). \tag{22}$$

3.1.3 Posterior state of knowledge

Bayes' rule combines the prior GP (see Eq. (14)) with the likelihood, (see Eq. (22)), and yields the *posterior* GP

$$f(\cdot)|\mathbf{X}, \mathbf{t}, \boldsymbol{\theta}, s_n \sim \mathrm{GP}\left(f(\cdot)\middle|\tilde{m}(\cdot), \tilde{k}(\cdot,\cdot)\right), \tag{23}$$

where the *posterior* mean and covariance functions are

$$\tilde{m}(\mathbf{x}) := \tilde{m}(\mathbf{x};\boldsymbol{\theta}) = m(\mathbf{x};\boldsymbol{\theta}) + \mathbf{K}(\mathbf{x},\mathbf{X};\boldsymbol{\theta})\left(\mathbf{K} + s_n^2\mathbf{I}_N\right)^{-1}(\mathbf{t} - \mathbf{m}), \tag{24}$$

and

$$\begin{aligned} \tilde{k}(\mathbf{x},\mathbf{x}') &:= \tilde{k}(\mathbf{x},\mathbf{x}';\boldsymbol{\theta}, s_n) \\ &= k(\mathbf{x},\mathbf{x}';\boldsymbol{\theta}) - \mathbf{K}(\mathbf{x},\mathbf{X};\boldsymbol{\theta})\left(\mathbf{K} + s_n^2\mathbf{I}_N\right)^{-1}\mathbf{K}(\mathbf{Q},\mathbf{x};\boldsymbol{\theta}), \end{aligned} \tag{25}$$

respectively.

Eqs. (24) and (25) completely quantify our state of knowledge about the response surface after observing the data. However, in practice, it is more convenient to work with the *predictive probability density* at a single input $\mathbf{x}$ conditional on the hyper-parameters $\boldsymbol{\theta}$ and s_n—namely

$$f(\mathbf{x})|\mathbf{X}, \mathbf{t}, \boldsymbol{\theta}, s_n \sim \mathcal{N}\left(f(\mathbf{x})|\tilde{m}(\mathbf{x}), \tilde{\sigma}(\mathbf{x})\right), \tag{26}$$

where $\tilde{m}(\mathbf{x}) = \tilde{m}(\mathbf{x};\boldsymbol{\theta})$ is the *predictive mean* given in Eq. (24), and

$$\tilde{\sigma}^2(\mathbf{x}) := \tilde{k}(\mathbf{x},\mathbf{x};\boldsymbol{\theta}, s_n) \tag{27}$$

is the *predictive variance*.

The predictive mean can be used as a point-wise surrogate of the response surface—that is, the interpolation value[10], while the predictive variance can be used to derive point-wise predictive error bars. Note that the predictive mean $\tilde{m}(\mathbf{x})$ of Eq. (24) can also be written as

$$\tilde{m}(\mathbf{x}) = m(\mathbf{x}) + \sum_{i=1}^{N} a_i k(\mathbf{x}_i, \mathbf{x}), \tag{28}$$

where $\boldsymbol{a} = (a_1, \ldots, a_N) = \left(\mathbf{K} + s_n^2\mathbf{I}_N\right)^{-1}(\mathbf{t} - \mathbf{m})$. We can think of the GP posterior mean as an approximation of $f(\cdot)$ using N symmetric basis functions centered at each observed input. Thus, by choosing a covariance function $k(\mathbf{x},\mathbf{x}')$ that vanishes when $\mathbf{x}$ and $\mathbf{x}'$ are separated by a lot, for example the SE covariance function (see Eq. (15)), we see that an observed input-output will only affect the approximation locally. This observation also establishes a connection between GP regression and reproducing-kernel Hilbert spaces, which is discussed in [72].

[10] Note that the convergence rate of approximating a function with the predictive mean of a GP depends on the choice of the kernel and the smoothness of the function that needs to be learned. We refer the reader to [72], Chapters 7.2 and 7.3 for more details.

3.1.4 Fitting the hyper-parameters

The hyper-parameters of the covariance function, $\boldsymbol{\theta}$, are typically estimated by maximizing the likelihood (see Ch. 5 of [72]). For reasons of numerical stability, we prefer to work with the logarithm of Eq. (22),

$$\mathcal{L}(\boldsymbol{\theta}, s_n; \mathbf{X}, \mathbf{t}) := \log p(\mathbf{t}|\mathbf{Q}, \boldsymbol{\theta}, s_n), \tag{29}$$

and determine the hyper-parameters by solving the following optimization problem:

$$\boldsymbol{\theta}^*, s_n^* = \arg\max_{\boldsymbol{\theta}, s_n} \mathcal{L}(\boldsymbol{\theta}, s_n; \mathbf{X}, \mathbf{t}), \tag{30}$$

subject to any constraints (see Ch. 5 of [72]), e.g., the covariance lengthscales, signal strength, and variance must be positive. We solve the optimization problem of Eq. (30) via the BFGS optimization algorithm [23] and increase the chances of finding the global maximum by restarting the algorithm multiple times from random initial points. To deal with the positivity constraints we simply optimize over the logarithm of all positive quantities.

3.1.5 Analytic example—Gaussian processes on simplices

It is a well-known fact from the literature that approximation and interpolation methods that are based on a (sparse) tensor product construction such as (adaptive) sparse grids [20–22, 57, 68] or Smolyak's method [46, 62] have a hard time dealing with high-dimensional functions whose domain cannot trivially be mapped onto a hypercube. To this end, we demonstrate now the ability of GPs to efficiently represent functions on geometries other than hypercubes.[11] For the testing, we proceed as follows: We choose a function $f : \Delta^2 \to \mathbb{R}$ and observe it at a finite number of points from the 2-simplex.[12] Next, we fit the GP hyper-parameters by maximizing the likelihood as outlined in Sec. 3.1.4. We then use the posterior mean $\tilde{m}(\mathbf{x})$ (see Eq. (24)) as a surrogate of $f(\mathbf{x})$. Subsequently, we randomly generate $N = 1,000$ test points uniformly drawn from the 2-simplex Δ^2, denoted by $\mathbf{X} = \{\mathbf{x}^{(i)} : i = 1, \ldots, N\}$, and finally compute the average error, which we define as

$$e = \sqrt{\sum_{i=1}^{N} \left(f\left(\mathbf{x}^{(i)}\right) - \tilde{m}\left(\mathbf{x}^{(i)}\right)\right)^2 / f\left(\mathbf{x}^{(i)}\right)^2}. \tag{32}$$

As an analytic example, we have chosen the 2-dimensional test function

$$f(x_1, x_2) = |0.25 - x_1^2 - x_2^2|. \tag{33}$$

The upper left panel of Fig. 1 shows the evaluation of the analytical function $f(\cdot)$ at random test points on the 2-simplex Δ^2. The upper right panel of Fig. 1 illustrates how the convergence of the interpolator—that is, the predictive mean, constructed by GPs, improves with an increase in the number of sampling points. The convergence results obtained by GPs are contrasted with sparse grid as well as adaptive sparse grid solutions (see, e.g., [21, 22, 68]). The latter is an approximation scheme known to be able to deal with high-dimensional functions very efficiently on cubic $[0, 1]^D$ domains, as it concentrates the computational effort where it is needed most. To obtain a fair comparison, the (adaptive) sparse grids were constructed under the assumption that either i) the continuation value of the function outside the the simplex

[11] For a discussion in the economic context, we refer the reader to Sec. 5.4.

[12] A simplex is a generalization of a tetrahedron to arbitrary dimensions. The standard n-simplex is the subset of $\mathbb{R}^{n+1}$ and is usually defined as

$$\Delta^n = \{(t_0, ..., t_n) \in \mathbb{R}^{n+1} | \sum_{i=0}^{n} t_i = 1 \text{ and } t_i \geq 0 \ \forall i\}. \tag{31}$$

In the context of economic applications, simplices often occur when one has to deal with Negishi's weights (see, e.g., [18]). On such a geometry, global approximations of value and policy functions are notoriously difficult to construct for dimensions larger than 2.

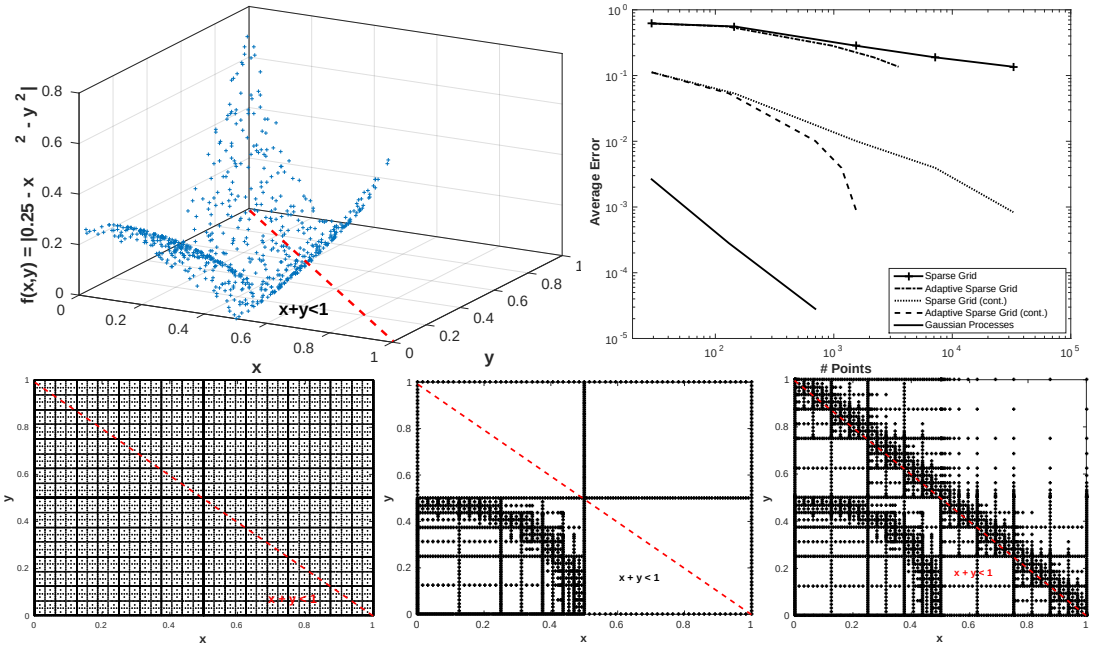

Figure 1: The upper left panel shows the analytical test function given by Eq. (33), evaluated at random test points on the 2-simplex Δ^2. The upper right panel displays a comparison of the interpolation error for GPs, sparse grids, and adaptive sparse grids of varying resolution and constructed under the assumption that a continuation value exists, (denoted by "cont"), or that there is no continuation value. The lower left panel displays a sparse grid consisting of $32,769$ points. The lower middle panel shows an adaptive sparse grid ("cont") that consists of $1,563$ points, whereas the lower right panel shows an adaptive sparse grid, constructed with $3,524$ points and under the assumption that the function outside Δ^2 is not known.

is known—that is to say, a functional expression that allows for a smooth continuation of the function to be approximated outside the domain Δ^2 exists[13], or ii) it is not known. In the latter case, we set the function value for the (adaptive) sparse grid outside the simplex to 0. Illustrative (adaptive) sparse grids are shown in the lower three panels of Fig. 1. From these three pictures, it becomes evident why GPs are superior in approximating functions that live on non-cubic geometries: GPs enable us to construct a surrogate of a function only on the domain of interest, whereas methods that are based on a tensor product construction such as (adaptive) sparse grids require an entire cubic domain. This, in turn, results in a tremendous waste of resources, as the interpolator also needs to be constructed outside the simplex, even though it is never required. The latter effect becomes particularly important in high-dimensional settings, as for example the D-dimensional volume of a unit simplex,

$$\mathrm{Vol}_\Delta = \frac{1}{D!}, \tag{34}$$

is drastically smaller than the volume of a unit cube, $\mathrm{Vol}_{unit} = 1^D$, on which the (adaptive) sparse grids are defined [22]. Similar behavior also occurs for example when one has to deal with hyperspheres, another geometry that regularly shows up in economic applications (see, e.g., [60]). There also, the volumes of those objects become negligibly small in high dimensions compared to the volume of the hypercube of equal diameter.

[13] In real economic applications where for example, the underlying dynamic optimization problem is infeasible in some regions of the computational domain, this assumption is clearly violated.

3.2 Active subspaces

Standard GPs (as well as practically any generic regression method) are not able to deal with very high input dimensions ($D \gg 20$). 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 [5], the number of observations required to learn the function grows enormously.

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:

$$f(\mathbf{x}) \approx h\left(\mathbf{W}^T\mathbf{x}\right), \tag{35}$$

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. (35) 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 (see Sec. 3.2.5 for examples). 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,

$$V_d\left(\mathbb{R}^D\right) := \left\{\mathbf{A} \in \mathbb{R}^{D\times d} : \mathbf{A}^T\mathbf{A} = \mathbf{I}_d\right\}, \tag{36}$$

with $\mathbf{I}_d$ the $d \times d$ unit matrix. $V_d\left(\mathbb{R}^D\right)$ is also known as the *Stiefel manifold*. Note that the representation of Eq. (35) is arbitrary up to rotations and relabeling of the active subspace coordinate system. Intuitively, we expect that there is a d-dimensional subspace of $\mathbb{R}^D$ over which $f(\cdot)$ exhibits most of its variation. If d is much smaller than D, then the problem of learning the surrogate is significantly simplified.

3.2.1 Gradient-based approach to active subspace regression

The "classical" approach to discovering the active subspace requires using gradient information [27, 30, 56], that is, in addition to $\mathbf{X}$ and $\mathbf{t}$ of Eq. (12) and Eq. (13), we need the respective gradients

$$\mathbf{G} = \left\{\mathbf{g}^{(1)}, \ldots, \mathbf{g}^{(N)}\right\}, \tag{37}$$

where

$$\mathbf{g}^{(i)} = \nabla f\left(\mathbf{x}^{(i)}\right) \in \mathbb{R}^D \tag{38}$$

and $\nabla f(\cdot)$ is the gradient of $f(\cdot)$,

$$\nabla f(\cdot) = \left(\frac{\partial f(\cdot)}{\partial x_1}, \ldots, \frac{\partial f(\cdot)}{\partial x_D}\right). \tag{39}$$

In our economic examples (see Secs. 2.2 and 5), these are the gradients of the optimal value of a constrained optimization problem. In Appendix 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 (see Sec. 3.2.2). 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 (see Sec. 3.2.3).

3.2.2 Finding the active subspace using gradient information

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

$$\mathbf{C} := \int (\nabla f(\mathbf{x}))(\nabla f(\mathbf{x}))^T \rho(\mathbf{x}) d\mathbf{x}. \tag{40}$$

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

$$\mathbf{C} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \tag{41}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_1, \cdots, \lambda_D)$ is a diagonal matrix containing the eigenvalues of $\mathbf{C}$ in decreasing order, $\lambda_1 \geq \cdots \geq \lambda_D \geq 0$, 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,

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_2 \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_2 \end{bmatrix},$$

(here $\mathbf{\Lambda}_1 = \text{diag}(\lambda_1, \ldots, \lambda_d), \mathbf{V}_1 = [\mathbf{v}_{11} \ldots \mathbf{v}_{1d}]$, and $\mathbf{\Lambda}_2, \mathbf{V}_2$ are defined analogously), and setting the projection matrix to

$$\mathbf{W} = \mathbf{V}_1. \tag{42}$$

Intuitively, $\mathbf{V}$ rotates the input space so that the directions associated with the largest eigenvalues correspond to directions of maximal function variability. See [27] for the theoretical justification.

It is impossible to evaluate Eq. (40) exactly. Instead, the usual practice is to approximate the integral via MC, that is, assuming that the observed inputs are drawn from $\rho(\mathbf{x})$, one approximates $\mathbf{C}$ using the observed gradients (see Eq. (37)) by

$$\mathbf{C}_N = \frac{1}{N}\sum_{i=1}^{N} \mathbf{g}^{(i)} \left(\mathbf{g}^{(i)}\right)^T. \tag{43}$$

In practice, the eigenvalues and eigenvectors of $\mathbf{C}_N$ are found using the singular value decomposition (SVD) [35] of $\mathbf{C}_N$. The dimensionality d is determined by looking for sharp drops in the spectrum of $\mathbf{C}_N$ [27]. Such drops guarantee that the response surface has an AS and that it can be approximated well. Alternatively, one may use the Bayesian information criterion, which is an approximation to model evidence. For the latter, see [84].

Relationship to principal component analysis. Principal component analysis (PCA) [86] 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 [27] that the mean square error is bounded by a term proportional to the sum of the neglected eigenvalues of $\mathbf{C}$ (cf. Eq. (41)). 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 $f(\mathbf{x})$.

3.2.3 Finding the map between the active subspace and the response

Using the projection matrix based on gradient information (see Eq. (42)), we obtain the projected observed inputs $\mathbf{Z} \in \mathbb{R}^{N\times d}$

$$\mathbf{Z} = \left\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(N)}\right\}, \tag{44}$$

where

$$\mathbf{z}^{(i)} = \mathbf{W}^T \mathbf{x}^{(i)}. \tag{45}$$

The link function $h(\cdot)$ that connects the active subspace to the output (see Eq. (35)) is identified using GP regression (see Sec. 3.1), with response $f(\cdot) \equiv h(\cdot)$, input points $\mathbf{x} \equiv \mathbf{z}$, observed inputs $\mathbf{X} \equiv \mathbf{Z}$, and observed outputs $\mathbf{t}$.

Note that GPR can resolve local features such as steep gradients (see Eq. (28)), but not discontinuities or extra sharp changes—that is, it cannot deal with non-stationary functions, that is to say, functions whose lengthscale changes dramatically (orders of magnitude) from one part of the computational domain to another. Moreover, ASGP can deal with local features only if they appear in the AS, but not with arbitrary local features that appear in the original input space.

3.2.4 Computational cost

Let N be the number of observations. The computational cost of standard GP regression is dominated by the need to construct the Cholesky decomposition of the $N \times N$ covariance matrix at each step of the likelihood optimization—that is, it is $O(N^3)$.[14] The computational cost of AS arises from a single SVD of an $N \times N$ matrix, plus the cost of a standard GP regression—that is, it is also $O(N^3)$. In both cases, the right number of observations N depends on the function smoothness and the input dimensionality: D for the GP regression and $d \ll D$ for the ASGP regression. The complete details of this relationship are not entirely understood theoretically and are well beyond the scope of the present work. However, we observe in our numerical experiments that the number of samples required by GP regression, say N_{GPR}, is much larger than the number of samples required by AS-GPR, say $N_{\text{AS-GPR}}$; that is, we expect that $N_{\text{GPR}} \gg N_{\text{AS-GPR}}$ when $D \gg d$.

3.2.5 Analytic examples—ASGP

In this section, we illustrate the ASGP method based on three instructive, synthetic examples. As a first test case, we choose a function $f : \Omega \to \mathbb{R}$ with $\Omega = [-1, 1]^2$, observe it at a finite number of points, discover the active subspace, and then fit the GP hyper-parameters by maximizing the likelihood as outlined in Sec. 3.1.4. Subsequently, we use the active subspace posterior mean, $\tilde{m}(\mathbf{x})$ (see Eq. (35)), as a surrogate of $f(\mathbf{x})$. Then, we randomly generate $N = 1,000$ test points uniformly drawn from Ω, denoted by $\mathbf{X} = \{\mathbf{x}^{(i)} : i = 1, \ldots, N\}$, and finally compute the average error, which is again defined as stated in Eq. (32). Following [27], we choose a 2-dimensional function, namely,

$$f(x, y) = \exp(0.3x + 0.7y). \tag{46}$$

The analytical function given by Eq. (46) is depicted in the left panel of Fig. 2. The arrows in the right panel of Fig. 2 indicate that $[0.3, 0.7]$ is the direction in which this function varies the most, whereas in its orthogonal direction $[-0.7, 0.3]$, $f(\cdot)$ it is constant. The projection matrix $\mathbf{W}$ of the underlying 1-dimensional AS is shown in the left panel of Fig. 3. The right panel of Fig. 3 illustrates how the convergence of the interpolator constructed by ASGPs compares to the performance of pure GPs with an increase in the number of sampling points. We see that the ASGP method yields very competitive results with a considerably reduced computational burden. As a second example, we construct an ASGP interpolator of $f : [-1, 1]^{10} \to \mathbb{R}$

$$\begin{aligned} f(x_1, ..., x_{10}) = \exp(&0.01x_1 + 0.7x_2 + 0.02x_3 + 0.03x_4 + 0.04x_5 \\ &+ 0.05x_6 + 0.06x_7 + 0.08x_8 + 0.09x_9 + 0.1x_{10}). \end{aligned} \tag{47}$$

Note that we deliberately put substantially more weight on the second dimension x_2. The left panel of Fig. 4 shows the sorted eigenvalues of the matrix $\mathbf{C}_N$. The gap after the first

[14] We expect that this will cause problems when N is in the order of $10,000$. In such cases, one has to resort to fast GP approximations, a natural extension of our current methodology, albeit beyond the scope of the present work.

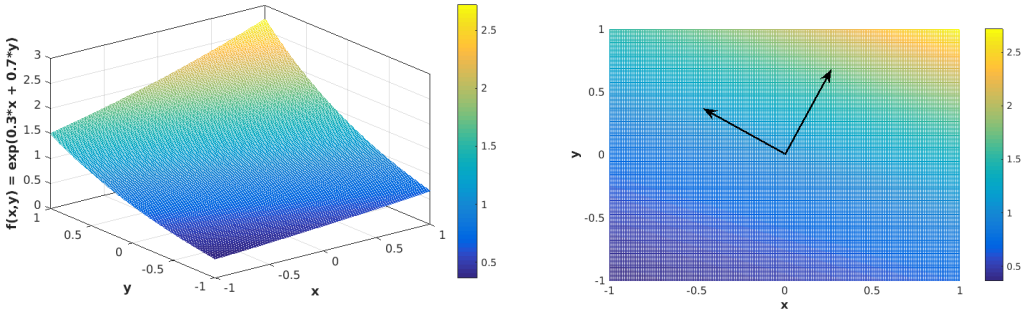

Figure 2: The left panel shows $f(x, y) = \exp(0.3x + 0.7y)$. The right panel displays arrows indicating that the test function given by (46) varies the most in the direction [0.3, 0.7] and is constant in the orthogonal direction.

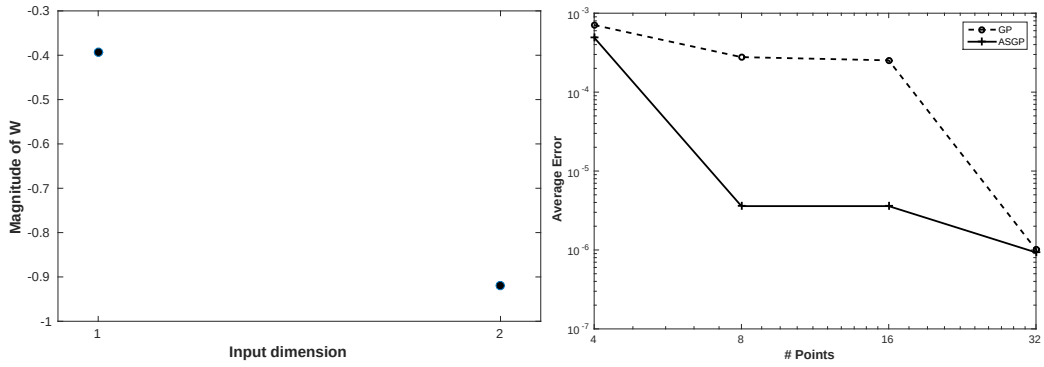

Figure 3: The left panel shows the projection matrix $\mathbf{W}$ of the 1-dimensional AS. The right panel displays a comparison of the interpolation error for 2-dimensional GPs and an 1-dimensional AS of varying resolution, respectively.

eigenvalue reveals that $f(\cdot)$ has a 1-dimensional AS. In the right panel of Fig. 4, we display again the projection matrix $\mathbf{W}$ of the 1-dimensional AS. It can be seen that, as one would expect from Eq. (47), the most dominant dimension is the second one. Third, we construct an ASGP surrogate of $f : [-1, 1]^{10} \to \mathbb{R}$

$$f(x_1, ..., x_{10}) = x_2 \cdot x_3 \cdot \exp(0.01x_1 + 0.7x_2 + 0.02x_3 + 0.03x_4 + 0.04x_5 + 0.05x_6 + 0.06x_7 + 0.08x_8 + 0.09x_9 + 0.1x_{10}). \quad (48)$$

The left panel of Fig. 5 displays the sorted eigenvalues of the matrix $\mathbf{C}_N$. One can clearly see that the gap in the spectrum only occurs after the third eigenvalue, indicating that the AS is a 3-dimensional space. The right panel of Fig. 5 illustrates how the convergence (see Eq. (32)) of ASGP surrogates of dimension $d = \{1, 2, 3\}$ performs with an increasing number of training inputs. We can clearly see that only the ASGP surrogate of dimension 3 quickly converges to a satisfactory level of accuracy.

Having now demonstrated with instructive examples that AS can correctly discover all the dimensions of interest and, coupled to GPs, can efficiently learn functions, we proceed next to economic applications.

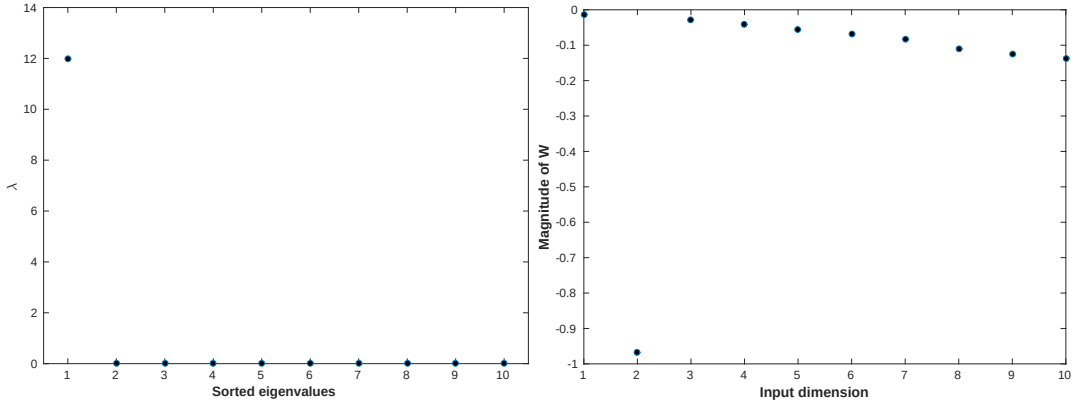

Figure 4: The left panel shows the sorted eigenvalues of $\mathbf{C}_N$ (see Eqn. 41), whereas the right panel displays the components of the projection matrix $\mathbf{W}$ of a 1-dimensional AS.

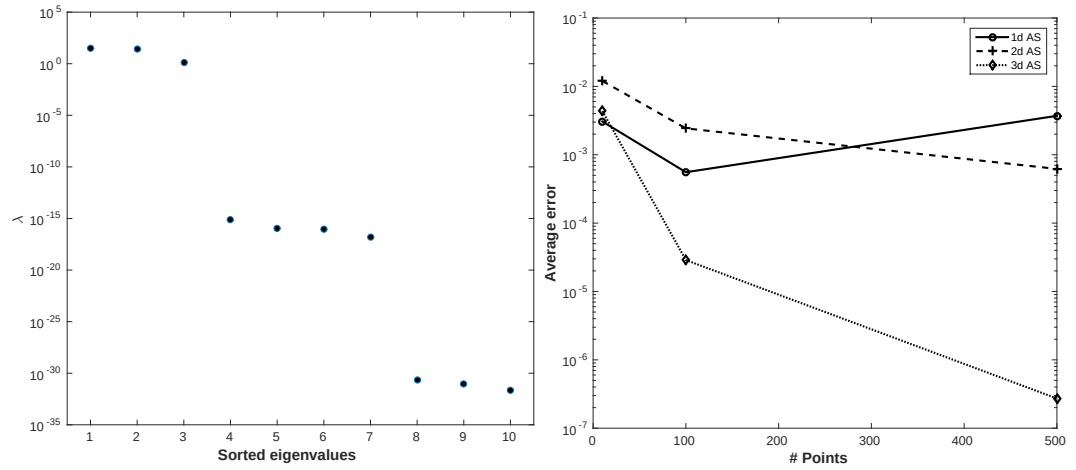

Figure 5: The left panel shows sorted eigenvalues of $\mathbf{C}_N$ (see Eqn. 41), whereas in the right panel we compare the interpolation error for $1, 2$, and 3-dimensional ASGPs as a function of increased sample size.

4 Active subspace and Gaussian process dynamic programming

We now describe how to solve the SG model presented in Sec. 2.2 using GP or ASGP machine learning. For this purpose, we build a parallel VFI 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 VFI framework. Second, we briefly discuss its parallelization. Third, we define the error measures we apply to check for convergence.

4.1 Value function iteration

Given the abstract VFI algorithm stated in Sec. 2.1 (see Eqs. (5) and (8)), we now aim to solve the SG model outlined in Sec. 2.2. The ASGP VFI algorithm that we propose for computing the optimal decision rules of the SG model proceeds as follows: We first make a starting guess for a value function V^{∞} (cf, Sec. 2.2). At iteration step s, we generate a relatively small set

of n^s training inputs[15] in the state space, namely, $\mathbf{k}^s_{1:n}$[16], on which we evaluate the Bellman operator $TV^s(\mathbf{k}^s_i)$ (see Eq. (5))

$$\mathbf{t}^s_{1:n^s} = \{\mathbf{t}^s_1, \ldots, \mathbf{t}^s_{n^s}\}, \tag{49}$$

with

$$\mathbf{t}^s_i = TV^{s-1}(k^s_i). \tag{50}$$

We can now–depending on the dimensionality of the training data–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. (8)). The iterative procedure stated in Eq. (50) is continued until convergence is reached (see Sec. 4.3). Alg. 1 summarizes the detailed steps of the ASGP VFI in a more formal way. Note that at convergence, one can, if desired, also learn the equilibrium policy functions u^*[17], 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 Alg. 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 both 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" [48] or "adaptive sparse grids" [21], 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 Sec. 5.4.

Finally, please note that ASGP is not restricted to dynamic programming and value function iteration, but could also be used in time iteration (see, e.g., [45])—that is, when one solves for recursive equilibria by iterating on policies (see Eq. (6)). The one notable difference between a time iteration algorithm [45] and Alg. 1 is that with the latter, multiple (policy) functions would need to be learned in every iteration step.

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 VFI algorithm described in Sec. 4.1 can exploit the availability of high-performance computing: i) the evaluation of the Bellman operator (see Eq. (5) and Alg. 1) and ii) the training of the GP that represents the function (see Sec. 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., [77]). 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 VFI 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. (49)), 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 Eq. (8). 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

[15] In our practical applications, $(5 \text{ to } 10) \cdot D$ sample points turned out to yield satisfactory results. This observation is in line with Constantine [27], who suggests $(2 \text{ to } 10) \cdot D$ sample points for active subspace methods to work well, based on more theoretical foundations.

[16] Note that, henceforth, we use the the variables $\mathbf{x_i}$ and $\mathbf{k_i}$ for the training inputs interchangeably.

[17] Our framework is capable of approximating each policy independently—that is, if necessary, for some policies we may use pure GPs, whereas for others we can apply ASGP.

Data: Initial guess V_{next} for the next period's value function. Approximation accuracy $\bar{\eta}$.

Result: The (approximate) equilibrium $3D$ policy functions $\xi^* = \{\xi_1^*, ..., \xi_{3D}^*\}$ and the corresponding value function V^*.

Set iteration step $s = 1$.

while $\eta > \bar{\eta}$ **do**

Generate n training inputs $\mathbf{X} = \{\mathbf{k_i^s} : \mathbf{1} \leq \mathbf{i} \leq \mathbf{n}\} \in [\underline{\mathbf{k}}, \overline{\mathbf{k}}]^{\mathbf{D}}$.

for $\mathbf{k}_i^s \in \mathbf{X}$ **do**

Compute the maximization problem

$$V(\mathbf{k}_i^s) = \max_{\mathbf{I},\mathbf{c},\mathbf{l}} \left(u(\mathbf{c}, \mathbf{l}) + \beta \mathbb{E}\Big\{ V_{next}(\mathbf{k}_i^+) \Big\} \right), \text{ s.t.}$$

$$k_j^+ = (1 - \delta) \cdot k_j + I_j + \epsilon_j, \quad j = 1, ..., D$$

$$\Gamma_j = \frac{\zeta}{2} k_j \left(\frac{I_j}{k_j} - \delta \right)^2, \quad j = 1, ..., D$$

$$\sum_{j=1}^{D} (c_j + I_j - \delta \cdot k_j) = \sum_{j=1}^{D} (f(k_j, l_j) - \Gamma_j)$$

given the next period's value function V_{next}.

Set the training targets for the value function: $t_i = V(\mathbf{k_i^s})$.

If required, set the training targets to learn the policy function $\xi_{j_i}(\mathbf{k_i^s}) \in \arg\max_{p_j} V(\mathbf{k_i^s})$.

end

Set $\mathbf{t} = \{t_i : 1 \leq i \leq n\}$.

Given $\{\mathbf{X}, \mathbf{t}\}$, learn a surrogate $V_{surrogate}$ of V with ASGP (or GP).

Set $\xi_\mathbf{j} = \{\xi_{j_i} : 1 \leq i \leq n\}$.

Given $\{\mathbf{X}, \xi_\mathbf{j}\}$, 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

$V^* = V_{surrogate}$.

$\xi^* = \{\xi_1, \cdots, \xi_{3D}\}$.

Algorithm 1: Overview of the critical steps of the VFI algorithm.

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 [23] 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, (see Secs. 2.2 and 5), the Bellman operator is a contraction mapping [8, 45, 80]. It is for this reason that we can assess the convergence of the VFI by computing error measures after every iteration step s. The average error is obtained by uniformly generating $N = 10,000$ random test points from the domain

$[\underline{\mathbf{k}}, \overline{\mathbf{k}}]^D$ (see Sec. 2.2), and then evaluating the expressions below on the constructed surrogates:

$$e^s = \frac{1}{N} \sum_{i=1}^{N} |V^s(\mathbf{x^i}) - V^{s-1}(\mathbf{x^i})|, \tag{51}$$

whereas the maximum error is given by

$$a^s = \max_{i=1,N} |V^s(\mathbf{x^i}) - V^{s-1}(\mathbf{x^i})|. \tag{52}$$

To assess whether the VFI converges, we want either of the two quantities from Eq. (51) and Eq. (52) to decrease. As we solve an infinite-horizon problem, we stop the iteration at step $s = S$ once

$$e^{s=S}/\delta < \epsilon \tag{53}$$

or

$$a^{s=S}/\delta < \epsilon_a \tag{54}$$

is reached, and where

$$\delta = \left(\max_{i=1,\ldots,N} V^{s=S}(\mathbf{x^i}) - \min_{i=1,\ldots,N} V^{s=S}(\mathbf{x^i}) \right) \tag{55}$$

holds.[18] 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 Sec. 4) now to economic examples—namely, to SG models (see Sec. 2.2). We proceed in four steps. First, we consider in Sec. 5.1 a 1-dimensional version of the SG model and discuss its convergence with the number of iteration steps and training inputs. Moreover, we argue that the quality of the solution can be inferred not only by checking for the contraction (see Eq. (53) and Eq. (54)), but also by looking at the information on predictive error bars that is naturally encoded in GPs. Next, we show in Sec. 5.2 that AS in conjunction with GPs enable us to compute global solutions to dynamic stochastic models of unprecedented dimensionality. Sec. 5.3 introduces a novel new way of performing uncertainty quantification in economic models. In Sec. 5.4, we propose simple yet powerful algorithm to efficiently solve high-dimensional dynamic stochastic economic problems on irregularly shaped state spaces.

5.1 Solving a 1-dimensional model with Gaussian processes

To gain a systematic understanding of how the convergence and accuracy of the SG 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. 6 depicts the decreasing maximum and average error (see Eq. (53) and Eq. (54)) when performing VFI with GPs and $n^s = 10$ training inputs. In Fig. 7, we display the predictive mean μ of a value function when the VFI has converged. They were learned from $4, 6, 8$, and 10 training targets, respectively. In addition, we show the 95 percent confidence intervals for each of the value functions. From Fig. 7, it becomes evident that GPs provide a very efficient way of pinning down these nonlinear, smooth functions. With only eight training inputs for example, the uncertainty of the value function becomes negligibly small throughout the entire computational domain, as shown in the lower left panel of Fig. 7.

[18] In practical applications, we typically stop the VFI when the normalized average error (see Eq. (51)) reaches about 10^{-4}.

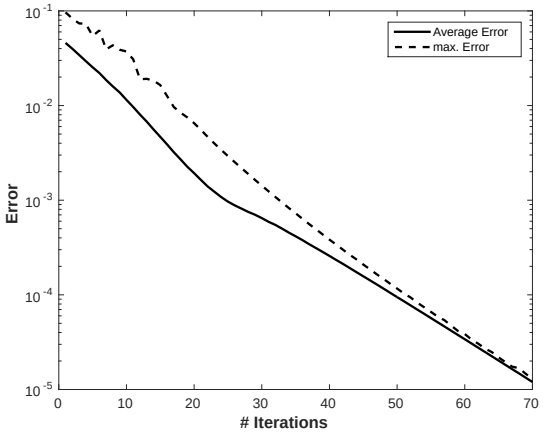

Figure 6: Decreasing maximum and average error for a 1-dimensional SG model that was solved by training a GP surrogate with $n^s = 10$ per iteration step.

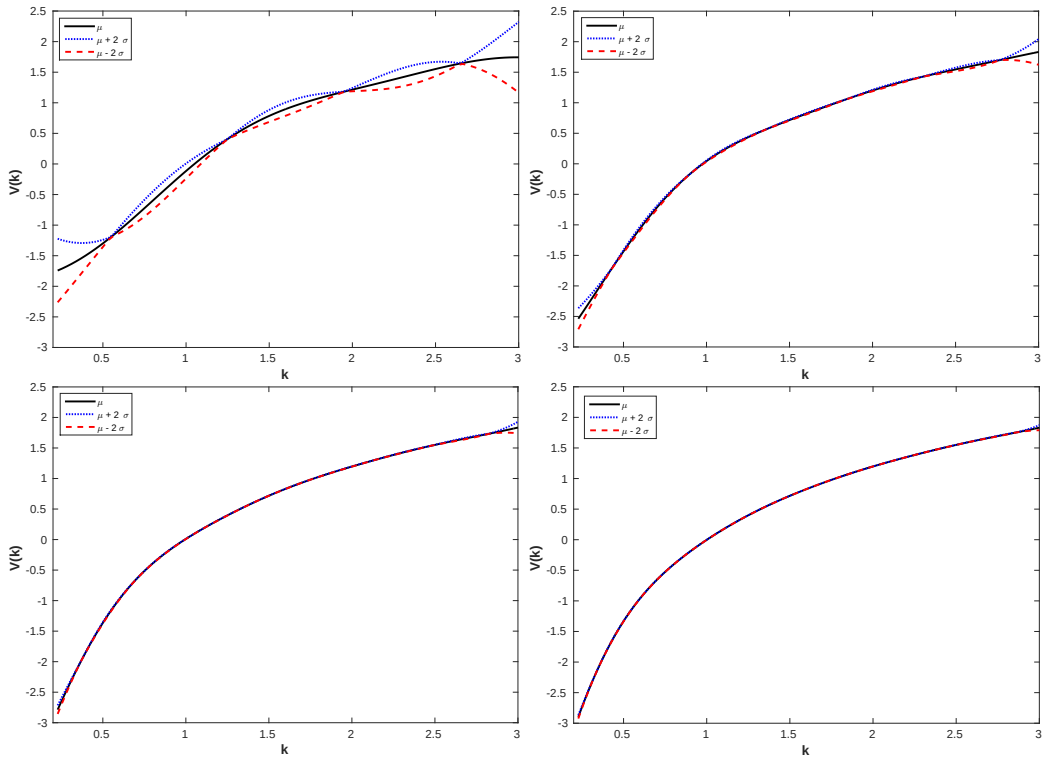

Figure 7: The above four panels display the predictive mean value function of a 1-dimensional growth model at convergence as a function of capital stock k. The 95 percent confidence intervals (corresponding to the point-wise predictive mean μ plus and minus two standard deviations for each input value of k) are also shown. The upper left panel was constructed based on a VFI that used only 4 training inputs per step in the state space, the upper right used 6 sample points. The lower left panel was a result of 8 sample points per iteration step, and the lower right is based on 10 sample points that pin down the predictive mean.

5.2 Solving very-high-dimensional SG models

We turn our attention now to very-high-dimensional SG models. In order to solve for their global solutions, however, we first need to assess what the dimensionality of the required AS

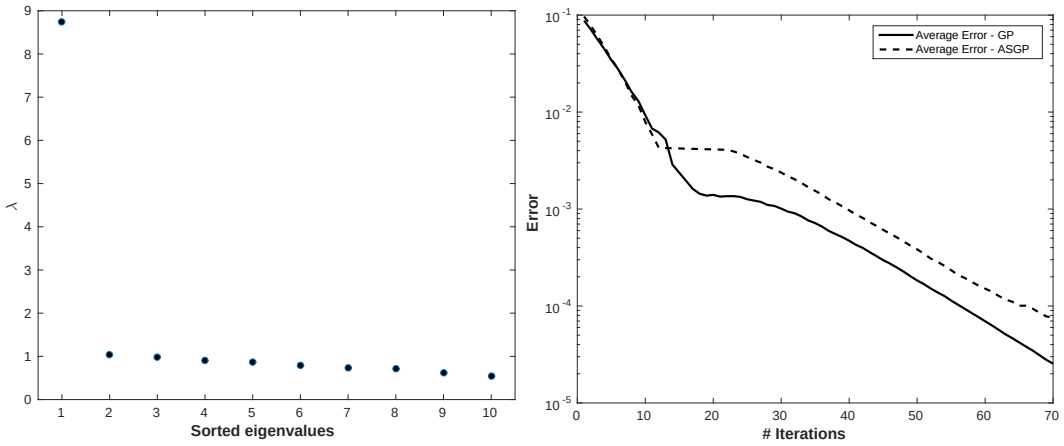

Figure 8: Left panel—sorted eigenvalues of $\mathbf{C}_N$ (see Eq. (41)) of the 10-dimensional SG model. Right panel—the decreasing average error for the 10-dimensional SG model, computed either by GPs or ASGP with an AS of dimension 1, respectively.

and the related quality of the solution will be. To do so, we compute the global solutions to a 10-dimensional SG, once computed with GPs alone, once with ASGP. The left panel of Fig. 8 shows the sorted eigenvalues of the $\mathbf{C}_N$ matrix (see Eq. (41)) for the SG model, indicating that we are dealing with a 1-dimensional AS. The right panel of Fig. 8 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.

In the hope that the AS of SG models is in general low-dimensional, we next increase the dimensionality of the problem up to $D = 500$, while fixing the AS to be 1-dimensional. In Fig. 9, we show that across all multi-dimensional models, the average error (see Eq. (32)) converges to less than 10^{-4}, which is worth mentioning. 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 [25], for example, who also computed global solutions to this type of stochastic growth model by VFI 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 two orders of magnitude smaller than ours.

Note that adaptive sparse grid-based solution algorithms (see, e.g., [20, 21]) as well as algorithms that are based on Smolyak's method [46, 48, 62] would not be able solve models of this size. Their underlying data structures become so intricate and slow to operate on [64] that they in practice become un-operational for problems of this size. Brumm and Scheidegger [21] 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.

5.3 Uncertainty quantification

Uncertainty quantification (UQ) is a field of applied mathematics concerned with the with the quantification of uncertainties and their propagation through computational models [78]. Model uncertainties either occur due to irreducible and uncontrollable random processes, or they quantify the analyst's lack of knowledge. UQ treats all uncertainties on an equal footing using probability theory. In the quantitative economics community, UQ 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 [39]. Answering these questions can, in turn, inform the researcher for example on which parts of the model she needs to focus on when

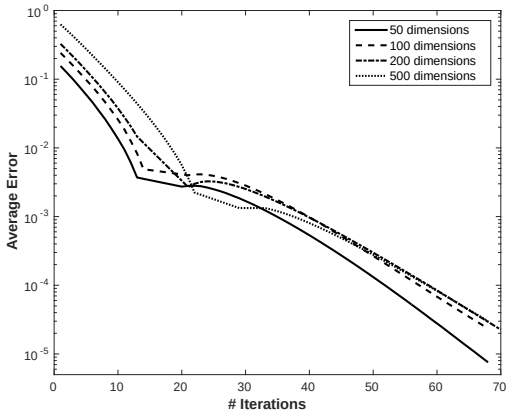

Figure 9: The decreasing average error from SG models ranging from from 50 to 500 dimensions.

calibrating it.

While UQ has had strong advocates in the economic modeling profession for quite some time (see, e.g., [38, 53]), it is still being ignored in the broader community. Cai, Judd, and Lontzek [24] 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. [39] 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 literature, it is however often the case that policy implications are derived based on a single set of model parameters. Hence, we elaborate below first on UQ in general before we show that our framework provides a novel and elegant way to perform UQ in economic models in a self-consistent fashion.

The first problem of UQ is to assign a joint probability distribution to the parameters of the model. The construction of this probability distribution is problem dependent. However, there are some generic principles that can serve as a starting point, including the maximum entropy principle [41–43], transformation group analysis [44], and reference priors [7]. If experimental data are available, one may take them into account using Bayes's rule [9, 11, 82]. The second task of UQ is to propagate the input uncertainty through the computational model and estimate the statistics of output quantities of interest (QoI). This problem is known as the uncertainty propagation (UP) problem. The simplest way to solve the UP problem is to use the MC method [2, 52, 63]—that is to say, by randomly sampling from the input distribution, evaluating the model at each sample point, and accumulating the statistics of interest. MC has the remarkable properties of having a convergence rate that is independent of the dimensionality of the random variables and of being easily parallelizable. Nevertheless, MC approaches typically require tens or hundreds of thousands of model evaluations. Unfortunately, in many problems involving time-consuming simulation models it is not feasible to achieve this number of simulations. In the words of the statistician A. O'Hagan, "Monte Carlo is fundamentally unsound" [67], in the sense that it fails to learn and exploit features hidden in the observed simulation data. The goal of UQ, therefore, is to construct methodologies that beat MC when certain conditions hold. All UQ methodologies rely on the idea of building a surrogate of the response surface of the model that is computationally inexpensive to evaluate. If this surrogate can be constructed at a cost smaller than that of MC, then the methodology is successful. UQ methods may be intrusive or non-intrusive. Intrusive UQ methodologies look into the mathematical details of the computational model and modify the equations so that uncertainty is propagated at once [33]. As such, intrusive UQ methodologies are rather involved, and they are not universally applicable. Non-intrusive UQ techniques attempt to build the response surface surrogate from a finite number of, potentially adaptively, selected input–output observations. The response

surfaces are usually represented using globally [87] or locally [58] supported polynomials, radial basis functions [14], or GP regression [10, 12, 13].

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, [79] for more details. Below, we shall concern ourselves only with parameter uncertainty and interpolation uncertainty, as they are most relevant to us.

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 Sec. 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 (see Fig. 7).

Parameter uncertainty stems from the fact that many parameters that are inputted into the economic models are not known exactly, only their (empirical) probability distributions are known. Moreover, the mapping from parameters to results is potentially 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 [39]. This approach, however, comes at a substantial cost of repeatedly solving the same model over the entire parameter range [24, 39].

We, therefore, propose an alternative and substantially more efficient way of dealing with UQ 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 very-high-dimensional state spaces. In particular, we add the parameters that carry uncertainty as additional, continuous states to the model.[19] 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 $\chi = (\chi_1, ..., \chi_N)$ as the random vector of N input parameters with a joint density f_χ defined over a probabilistic space (see, e.g., [7, 9, 11, 41–44, 82]). Next, we need to specify the distribution for each parameter χ_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 [43], we assume that all parameters χ_i are uniformly distributed with support given by the lower and upper bounds reported in Tab. 2.[20] Next, we formally define the mapping

$$\chi \in \mathcal{D}_\chi \subset \mathbb{R}^N \rightarrow y = \mathcal{M}(\chi) \in \mathbb{R}^Q, \tag{56}$$

where y is a so-called QoI—a random, endogenous outcome of the computational model [39].[21]

As a first example, let us consider the SG model (see Sec. 2.2) and assume for simplicity that the entire parameter uncertainty is encoded in one single parameter, namely the risk aversion $\chi_1 = \gamma$ (see Tab. 2 and Eq. (10)). We now extend the D-dimensional state space $\mathbf{k}$ of the model by one continuous variable:

$$\tilde{S} = (\mathbf{k}, \chi_1 = \gamma). \tag{57}$$

Next, we have to slightly adjust the VFI algorithm described in Sec. 4.1. First, instead of sampling n^s training inputs in the D-dimensional state space $[\underline{\mathbf{k}}, \overline{\mathbf{k}}]^D$, we now have to generate

[19] Norets [66] 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.

[20] Methodologically, there are no fundamental limitations on what distributions we can apply.

[21] In the examples below, we restrict ourselves to $Q = 1$.

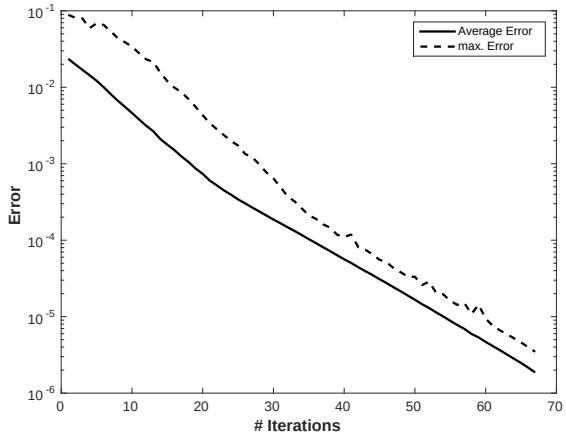

Figure 10: The decreasing maximum and average error for a 2-dimensional SG model (the continuous states being capital stock and risk aversion) that was solved by training a GP surrogate with 20 training targets per iteration step.

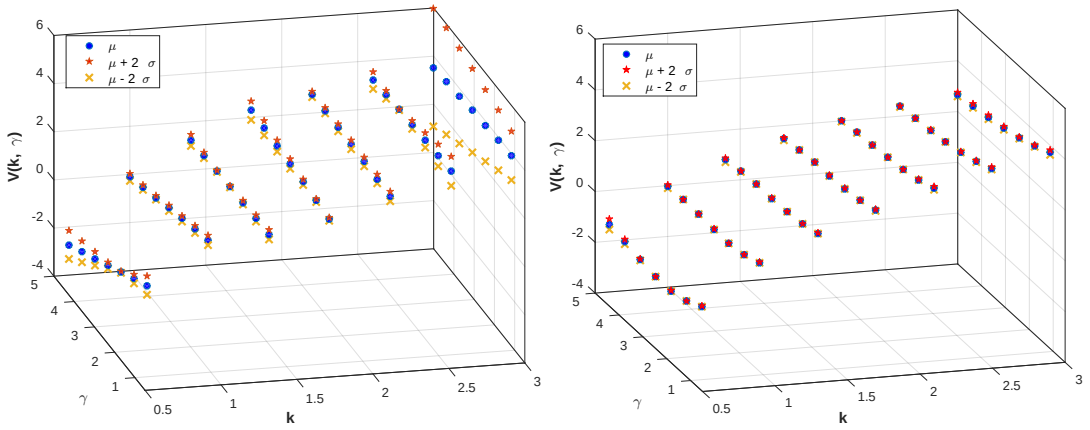

Figure 11: Left panel—We display the predictive mean μ of $V(\mathbf{k}, \gamma)$ at convergence, evaluated on a uniformly spaced Cartesian grid consisting of 7×7 points. The surrogate was constructed by using 10 observations. Right panel—$V(\mathbf{k}, \gamma)$ surrogate at convergence, generated by 20 observations. We also show the 95 percent confidence intervals (corresponding to the point-wise predictive mean μ plus and minus two standard deviations) in both panels.

training data from the $(D+1)$-dimensional, extended state space

$$[\underline{\mathbf{k}}, \overline{\mathbf{k}}]^D \times [\underline{\gamma}, \overline{\gamma}], \tag{58}$$

where $[\underline{\gamma}, \overline{\gamma}]$ 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, $TV^s(\tilde{S})$ (see Eq. (49) and Eq. (50)). 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 UQ, we consider first an SG model with one sector and $\gamma \in [0.5, 5]$. Fig. 10 depicts the decreasing maximum and average error (see Eq. (53) and Eq. (54)) of this 2-dimensional model. In Fig. 11, we show the predictive mean μ of the value function $V(\mathbf{k}, \gamma)$ when the iteration has converged. The left 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 VFI algorithm. The right 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

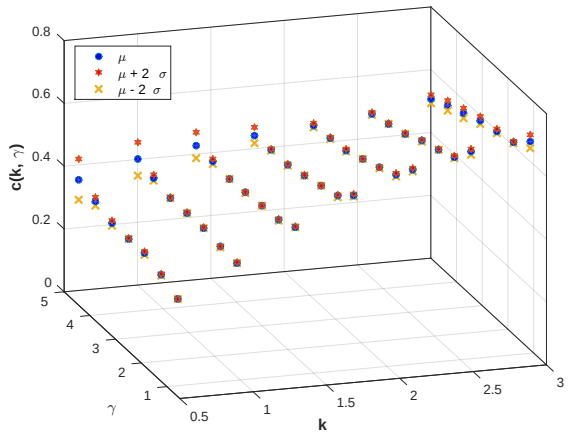

Figure 12: Predictive mean μ of consumption $c(\mathbf{k}, \gamma)$, evaluated on a uniformly spaced Cartesian grid that consists of 7×7 points. The surrogate was constructed by using 20 observations. We also show the 95 percent confidence intervals (corresponding to the point-wise predictive mean μ plus and minus two standard deviations).

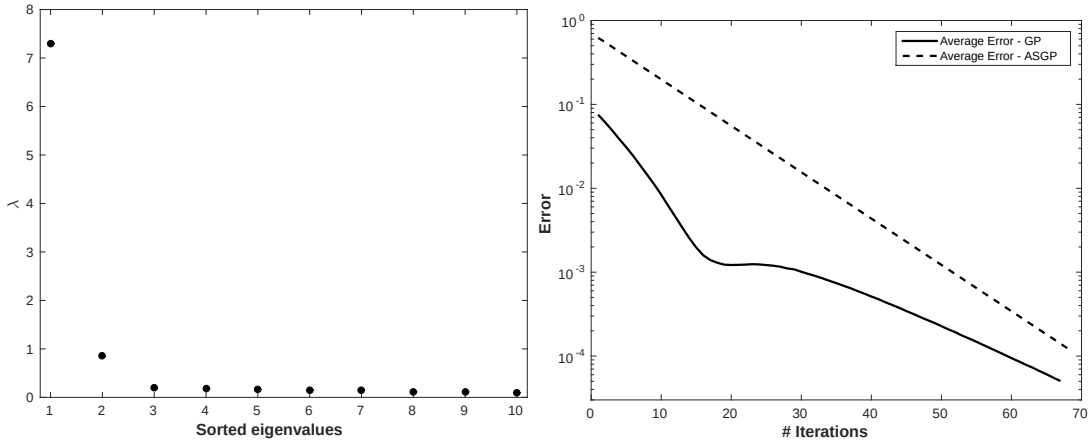

Figure 13: Left panel—sorted eigenvalues of $\mathbf{C}_N$ (see Eq. (41)) for the 11-dimensional SG problem with continuous states $\tilde{S} = (k_1, ..., k_{10}, \gamma)$. Right panel—the decreasing average error for the two 10-dimensional SG models, computed either by GPs or by ASGP with an AS of dimension 1, respectively.)

indicated by the decrease in the point-wise confidence intervals. In Fig. 12, we show the predictive mean μ of the optimal consumption policy $c(\mathbf{k}, \gamma)$ when the iteration has converged. The figure displays $c(\mathbf{k}, \gamma)$ as well as the 95 percent confidence intervals in the case where 20 training observations from the extended state space $\tilde{S}$ were provided to the VFI algorithm. 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 $\tilde{S} = (k_1, ..., k_9, \gamma)$, once computed with GP and once with ASGP. The left panel of Fig. 13 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. 13 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—hardly any deterioration in the quality of the solution can be found compared to the GP case.

After having verified, by enlarging the state space, that our framework is capable of performing basic UQ 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

Parameter	Baseline value	Lower bound for $\underline{\chi}_i$	Upper bound for $\overline{\chi}_i$
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

Table 2: Parameter ranges for the SG model. We assume that all parameters χ_i are uniformly distributed in $[\underline{\chi}_i, \overline{\chi}_i]$. Furthermore, we display the original parametrization of the SG model as *Baseline value* for completeness (see Tab. 1).

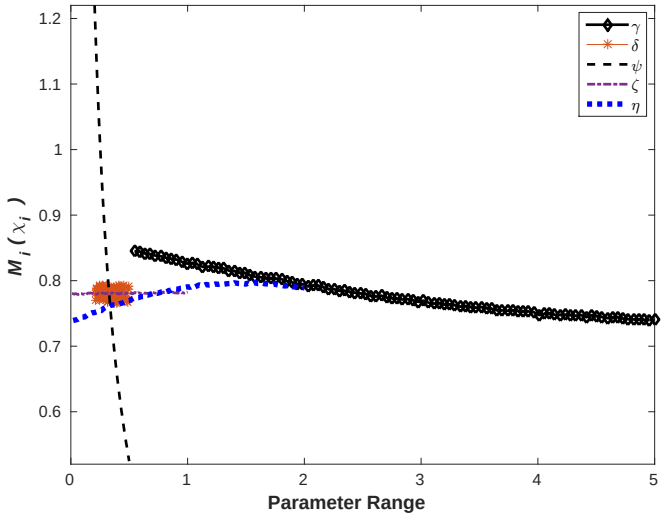

Figure 14: Univariate effects for the average production as a function of χ_i.

univariate effects on the model's output [39, 88]. To do so, let us consider an SG model with one sector (see Sec. 2.2) and assume that the parametric uncertainty is encoded in the following five parameters (see Tab. 2 for their ranges),

$$\chi = \{\gamma, \delta, \psi, \zeta, \eta\}, \tag{59}$$

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

$$[\underline{\mathbf{k}}, \overline{\mathbf{k}}] \times [\underline{\gamma}, \overline{\gamma}] \times [\underline{\delta}, \overline{\delta}] \times [\underline{\psi}, \overline{\psi}] \times [\underline{\zeta}, \overline{\zeta}] \times [\underline{\eta}, \overline{\eta}]. \tag{60}$$

The QoI we are concerned with in this example is the average production (see Eq. (9)):

$$\mathcal{M}(\chi) = \mathbb{E}[f]. \tag{61}$$

In UQ, univariate effects are defined as the conditional expectation of the QoI (see Eq. (56) and [88]) as a function of a single parameter, where the expectations are taken over all other parameters:

$$\mathcal{M}_i(\chi_i) = \mathbb{E}[\mathcal{M}(\mathbf{\Theta}|\Theta_i = \chi_i)]. \tag{62}$$

These effects are commonly interpreted as a robust relationship between an input parameter and the QoI (see, e.g., [39, 88], and references therein).[22] Fig. 14 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

[22] 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.

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 model (see Sec. 2.2). Fig. 14 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 time in economic applications—in a self-consistent fashion both parameter and interpolation uncertainty in a single model evaluation.[23] Moreover, it avoids tedious calibration exercises.[24] 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 DSGE models. 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 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 VFI on complex, high-dimensional geometries without suffering from massive inefficiencies. Especially in very-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 (see Sec. 3.1.5).

We now address the issue of performing VFI on irregularly shaped ergodic sets with our framework. There is a relatively broad literature on how to determine ergodic sets $\Omega_{ergodic}$ (see, e.g., [31, 60, 61] and references therein). Den Haan and Marcet [37], Judd et al. [61], and Maliar and Maliar [60] show how to approximate ergodic sets, whose geometries often resemble hyperspheres, by simulation. To perform VFI or time iteration on such state spaces, [60, 61] for example subsequently 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 VFI as follows: First, we solve the economic model at few points in a given time step s (see Sec. 2.2) on a "sufficiently" large domain (see Tab. 1) to learn the respective value function and policies. Second, following Judd et al. [61] and Maliar and Maliar [60], we use the computed surrogate of the policies to simulate the economy by using the law of motion for capital (see Eq. (8)),

$$k_j^+ = (1-\delta)\cdot k_j + I_j + \epsilon_j, \quad j = 1, ..., D, \tag{63}$$

in order to numerically generate an estimate for the ergodic set $\Omega_{ergodic}$.[25] In contrast to [60, 61], however, we now construct a probability density from the computed—that is, the observed—distribution of capital. Suppose that we have n samples $\mathbf{X}_{ergodic} = \{\mathbf{k}_i^+ : 1 \leq i \leq n\}$ obtained, for example, using Eq. (63). We then can approximate $\rho_{estimated}$ as a mixture of

[23] Note that the ability to perform UQ as efficiently as proposed here stands in stark contrast to [24, 39], 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.

[24] Sec. 5.2 indicates that we are capable of potentially accommodating dozens of parameters as additional states as long as the extended state space $\tilde{S}$ is not too big.

[25] In the very unlikely event that capital k_j^+ would leave the computational domain along the simulated path, we would truncate it to the lower or upper bound of $\mathbf{k}$, respectively (see Tab. 1).

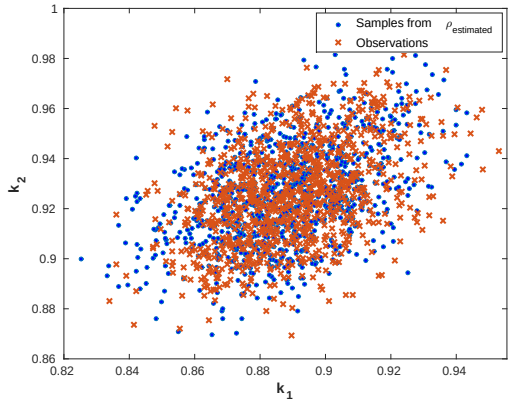

Figure 15: Scatterplot of the observed ergodic distribution of capital (see Eq. (63)) from a 2-dimensional SG model, contrasted with states that were randomly generated from $\rho_{estimated}$ (see (64)). A total of 1,000 samples are shown in either case.

Gaussians:

$$\rho_{estimated}(\mathbf{x}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m), \tag{64}$$

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 $\mathbf{X}_{ergodic}$. Note that there are many ways of fitting the relevant parameters. However currently, the most robust approach is based on the infinite Gaussian model mixture of [71], which uses variational inference [16]. Fig. 15 shows an example distribution of 1,000 simulated capital stocks from a 2-sector SG model (see Sec. 2.2) that was overlapped with 1,000 samples generated from the respective density $\rho_{estimated}$. From Fig. 15, 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 Sec. 3.1.5 that GP and ASGP work on arbitrary geometries as long as there exists a method that can provide training inputs and observations. We therefore propose to subsequently draw N samples from $\rho_{estimated}$ within a VFI step:

$$\mathbf{X}_{ergodic} = \left\{ \mathbf{x}_{ergodic}^{(1)}, \ldots, \mathbf{x}_{ergodic}^{(N)} \right\}, \tag{65}$$

where we observe training targets:

$$\mathbf{t}_{ergodic} = \left\{ t_{ergodic}^{(1)}, \ldots, t_{ergodic}^{(N)} \right\}. \tag{66}$$

Next, we train a surrogate by using the data $\{\mathbf{X}_{ergodic}, \mathbf{t}_{ergodic}\} \in \Omega_{ergodic}$. The details of the procedure are outlined in Alg. 2.[26] Once the corresponding surrogate is constructed, we can carry out one VFI step on the ergodic set with a very complex, high-dimensional state space (see Fig. 16) and without suffering from major computational inefficiencies. For illustrative purposes, we display in the left panel of Fig. 16 100 points in the state space that were drawn from the approximate ergodic set of a 2-dimensional SG 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 Tab. 1). The right panel shows the converged 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.

[26] Note that in practice, $\rho_{estimated}$ does not need to be updated in every individual iteration step of the VFI, as it may change relatively slowly within VFI. The rate at which one has to recompute the probability density may however strongly depend on the actual application. In our case, updating $\rho_{estimated}$ every 5 to 10 steps resulted in excellent performance.

Data: Initial guess V_{next} for the next period's value function. Approximation accuracy $\bar{\eta}$.
Result: The m (approximate) equilibrium policy functions ξ^* and the corresponding value function V^* on $\Omega_{ergodic}$.
Set iteration step $s = 1$.
while $\eta > \bar{\eta}$ **do**
  Determine $\Omega_{ergodic}$ by using (63) and (64).
  Generate n training inputs $\mathbf{X}_{ergodic} = \{\mathbf{k}_i^s : 1 \leq i \leq n\} \in \Omega_{ergodic}$.
  for $\mathbf{k}_i^s \in \mathbf{X}_{ergodic}$ **do**
    Evaluate the Bellman operator $TV^s(\mathbf{k}_i^s)$ (see Eq. (5))
    given next period's value function V_{next}.
    Set the training targets for the value function: $t_i = TV(\mathbf{k_i^s})$.
    If required, set the training targets to learn the j-th policy function:
    $\xi_{j_i}(\mathbf{k_i^s}) \in \arg\max_{p_j} TV(\mathbf{k_i^s})$.
  end
  Set $\mathbf{t}_{ergodic} = \{t_i : 1 \leq i \leq n\}$.
  Given $\{\mathbf{X}_{ergodic}, \mathbf{t}_{ergodic}\}$, learn a surrogate $V_{surrogate}$ of V.
  Set $\xi_{\mathbf{j}} = \{\xi_{j_i} : 1 \leq i \leq n\}$.
  Given $\{\mathbf{X}_{ergodic}, \xi_{\mathbf{j}}\}$, learn a surrogate of the policy ξ_j.
  Calculate (an approximation for) the error, e.g.: $\eta = \|V_{surrogate} - V_{next}\|_\infty$.
  Set $V_{next} = V_{surrogate}$.
  Set $s = s + 1$.
end
$V^* = V_{surrogate}$.
$u^* = \{\xi_1, \cdots, \xi_m\}$.

Algorithm 2: Overview of the critical steps of the VFI algorithm that operates on the ergodic set $\Omega_{ergodic}$.

With the examples presented in this section, it becomes evident that our framework—being highly versatile regarding how it incorporates information to construct a surrogate, is the tool of choice when one has to deal with irregularly shaped state spaces. To the best of our knowledge, there is to date no other method available that is capable of addressing dynamic stochastic economic problems on high-dimensional, non-trivial geometries as efficiently as that proposed in this work.

6 Conclusion

In this paper, we present the first computational framework that can compute global solutions to very-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 self-consistent manner. We achieve this by combining GP machine learning with the AS method, which we then embed into a massively 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 pointwise predictive confidence intervals. However, conventional GPs cannot deal with dimensions larger than $D \gg 20$. 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

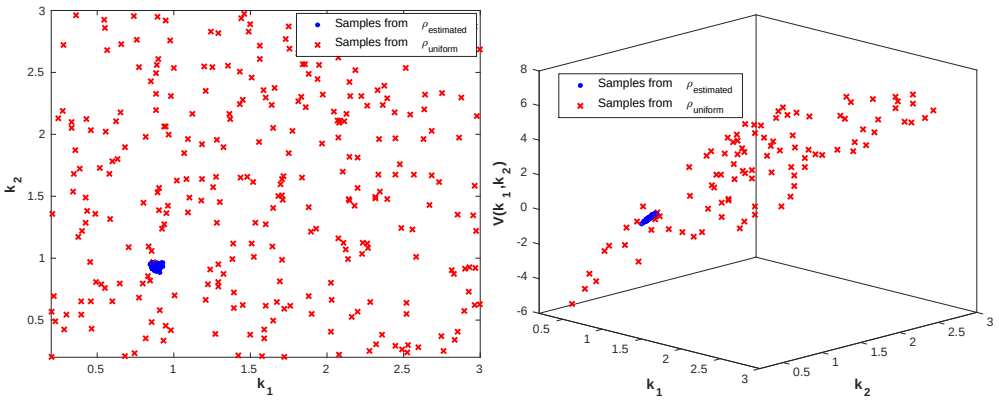

Figure 16: The left panel shows 100 sample points for capital, once generated uniformly from $\rho_{uniform}$—that is, a cubic domain (cf. Tab. 1), and once from $\rho_{estimated}$ (cf., (64)). The right panel displays the respective value function evaluated on the same sample points.

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 very-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 economics that can perform uncertainty quantification in a self-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 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, as well as how to perform dynamic programming on such potentially complex geometries.

This all suggests that our highly 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.

A Determination of gradients

The ASGP methodology requires the 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:*

$$\max_u h(x,u) \text{ subject to } g_i(x,u) \leq 0, \quad i = 1,\ldots,n_c, \tag{67}$$

where $n_c \in \mathbb{N}$, and assume the regularity assumptions of the Karush-Kuhn-Tucker (KKT) theorem [51] 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,

$$\mathcal{A}(x) = \{i \in \{1,\ldots,n_c\} : g_i(x,\mu(x)) = 0\}, \tag{68}$$

$m = |\mathcal{A}(x)|$ be the number of elements in set $\mathcal{A}(x)$, $i_1,\ldots,i_m$ an enumeration of the elements in set $\mathcal{A}(x)$, $g^a(x,u) = (g_{i_1}(x,u),\ldots,g_{i_m}(x,u))$ be a vector function corresponding to the active constraints, $\lambda^a(x) = (\lambda_{i_1}(x),\ldots,\lambda_{i_m}(x))$ 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:

$$\begin{pmatrix} \nabla_u^2\mathcal{L} & -(\nabla_u\nabla_\lambda\mathcal{L})^T \\ -\nabla_u\nabla_\lambda\mathcal{L} & 0 \end{pmatrix} \begin{pmatrix} \nabla_x\mu \\ \nabla_x\lambda^a \end{pmatrix} = \begin{pmatrix} -\nabla_x\nabla_u\mathcal{L} \\ -\nabla_x\nabla_\lambda\mathcal{L} \end{pmatrix}, \tag{69}$$

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

$$\mathcal{L}(x,u) = h(x,u) - \sum_{j=1}^{m} \lambda_{i_j}^a g_{i_j}(x,u). \tag{70}$$

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 $\lambda_i(x) \geq 0, i = 1,\ldots,n_c$ s.t.

$$\nabla_u h(x,\mu(x)) = \sum_{i=1}^{n_c} \lambda_i(x)\nabla_u g_i(x,\mu(x)), \tag{71}$$

$$g_i(x,\mu(x)) \leq 0, i = 1,\ldots,n_c, \tag{72}$$

and:

$$\lambda_i g(x,\mu(x)) = 0, i = 1,\ldots,n_c. \tag{73}$$

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. (73) that $\lambda_i(x) = 0$. Using, this information we may write:

$$\frac{\partial h(x,\mu(x))}{\partial u_j} = \sum_{i\in\mathcal{A}(x)} \lambda_i(x)\frac{\partial g_i(x,\mu(x))}{\partial u_j}, j = 1\ldots,d_u, \tag{74}$$

$$g_i(x,\mu(x)) = 0, \forall i \in \mathcal{A}(x), \tag{75}$$

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

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

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. (67) and let $h^*(x)$ be the optimal value, i.e.,*

$$h^*(x) = h(x,\mu(x)), \tag{76}$$

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

$$\nabla_x h^*(x) = \nabla_x h(x,\mu(x)) + \nabla_x \mu(x) \nabla_u h(x,\mu(x)). \tag{77}$$

References

[1] R. Aiyagari. "Uninsured Idiosyncratic Risk and Aggregate Saving". In: *The Quarterly Journal of Economics* 109(3) (1994), pp. 659–684.

[2] A. Barth, C. Schwab, and N. Zollinger. "Multi-level Monte Carlo Finite Element method for elliptic PDEs with stochastic coefficients". In: *Numerische Mathematik* 119.1 (2011), pp. 123–161.

[3] R. Bellman. *Adaptive Control Processes: A Guided Tour*. 'Rand Corporation. Research studies. Princeton University Press, 1961.

[4] R. Bellman. "Dynamic programming and lagrange multipliers". In: *Proceedings of the National Academy of Sciences of the United States of America* 42.10 (1956), pp. 767–769.

[5] Y. Bengio, O. Delalleau, and N. Le Roux. "The curse of highly variable functions for local kernel machines". In: *Neural Information Processing Systems*. 2005.

[6] J. Bengui, E. G. Mendoza, and V. Quadrini. "Capital mobility and international sharing of cyclical risk". In: *Journal of Monetary Economics* 60.1 (2013), pp. 42–62.

[7] J. O. Berger, J. M. Bernardo, and D. Sun. "The formal definition of reference priors". In: *The Annals of Statistics* (2009).

[8] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. 2nd. Athena Scientific, 2000.

[9] I. Bilionis, B. A. Drewniak, and E. M. Constantinescu. "Crop physiology calibration in the CLM". In: *Geoscientific Model Development* 8.4 (2015), pp. 1071–1083.

[10] I. Bilionis and N. Zabaras. "Multi-output local Gaussian process regression: Applications to uncertainty quantification". In: *J. Comput. Phys.* 231.17 (2012), pp. 5718–5746.

[11] I. Bilionis and N. Zabaras. "Solution of inverse problems with limited forward solver evaluations: a Bayesian perspective". In: *Inverse Problems* 30.1 (2014).

[12] I. Bilionis, N. Zabaras, B. A. Konomi, and G. Lin. "Multi-output separable Gaussian process: Towards an efficient, fully Bayesian paradigm for uncertainty quantification". In: *Journal Of Computational Physics* 241 (2013), pp. 212–239.

[13] I. Bilionis and N. Zabaras. "Bayesian uncertainty propagation using Gaussian processes". English. In: *Handbook of Uncertainty Quantification*. Cham: Springer International Publishing, 2016, pp. 1–45.

[14] I. Bilionis and N. Zabaras. "Multidimensional Adaptive Relevance Vector Machines for Uncertainty Quantification". In: *SIAM Journal on Scientific Computing* 34.6 (2012), B881–B908.

[15] C. M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. New York: Springer, 2006, xx, 738 p.

[16] D. M. Blei and M. I. Jordan. "Variational inference for Dirichlet process mixtures". In: *Bayesian Analysis* 1 (2005), pp. 121–144.

[17] J. Brumm and M. Grill. "Computing equilibria in dynamic models with occasionally binding constraints". In: *Journal of Economic Dynamics and Control* 38 (2014), pp. 142–160.

[18] J. Brumm and F. Kubler. "Applying Negishi's method to stochastic models with overlapping generations". In: *Working Paper, University of Zurich* (2014).

[19] J. Brumm, F. Kubler, and S. Scheidegger. "Computing equilibria in dynamic stochastic macro-models with heterogeneous agents". In: *Advances in Economics and Econometrics, Eleventh World Congress, Forthcoming* (2016).

[20] J. Brumm, D. Mikushin, S. Scheidegger, and O. Schenk. "Scalable high-dimensional dynamic stochastic economic modeling". In: *Journal of Computational Science* 11 (2015), pp. 12–25.

[21] J. Brumm and S. Scheidegger. "Using Adaptive Sparse Grids to Solve High-Dimensional Dynamic Models". In: *Econometrica, Forthcoming* (2017).

[22] H.-J. Bungartz and M. Griebel. "Sparse grids". In: *Acta Numerica* 13 (2004), pp. 1–123.

[23] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. "A Limited Memory Algorithm for Bound Constrained Optimization". In: *SIAM J. Sci. Comput.* 16.5 (Sept. 1995), pp. 1190–1208.

[24] Y. Cai, K. L. Judd, and T. S. Lontzek. "The Social Cost of Carbon with Economic and Climate Risks". In: *ArXiv e-prints* (Apr. 2015). arXiv: 1504.06909.

[25] Y. Cai and K. L. Judd. "Advances in numerical dynamic programming and new applications". In: *Handbook of computational economics* 3 (2014).

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

[27] P. G. Constantine. *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2015.

[28] M. P. Deisenroth, C. E. Rasmussen, and J. Peters. "Gaussian process dynamic programming". In: *Neurocomputing* 72.7 (2009), pp. 1508–1524.

[29] W. J. Den Haan, K. L. Judd, and M. Juillard. "Computational suite of models with heterogeneous agents II: Multi-country real business cycle models". In: *Journal of Economic Dynamics and Control* 35.2 (Feb. 2011), pp. 175–177.

[30] E. Dow and Q. Wang. "Output based dimensionality reduction of geometric variability in compressor blades". In: *51st AIAA Aerospace Sciences Meeting* (2013).

[31] A. Fernandes and C. Phelan. "A Recursive Formulation for Repeated Agency with History Dependence". In: *Journal of Economic Theory* 91.2 (2000), pp. 223–247.

[32] J. Fernandez-Villaverde, J. F. Rubio-Ramirez, and F. Schorfheide. *Solution and Estimation Methods for DSGE Models*. CEPR Discussion Papers 11032. C.E.P.R. Discussion Papers, Dec. 2015.

[33] R. Ghanem and P. D. Spanos. *Stochastic finite elements: a spectral approach*. Minneola, N.Y.: Dover Publications, 2003.

[34] P. W. Goldberg, C. K. I. Williams, and C. M. Bishop. "Regression with input-dependent noise: A Gaussian process treatment". In: *Advances in Neural Information Processing Systems 10* 10 (1998), pp. 493–499.

[35] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.)* Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[36] W. J. D. Haan, K. L. Judd, and M. Juillard. "Computational suite of models with heterogeneous agents II: Multi-country real business cycle models". In: *Journal of Economic Dynamics and Control* 35.2 (2011), pp. 175–177.

[37] W. J. D. Haan and A. Marcet. "Accuracy in Simulations". In: *Review of Economic Studies* 61.1 (1994), pp. 3–17.

[38] L. P. Hansen and J. J. Heckman. "The Empirical Foundations of Calibration". In: *Journal of Economic Perspectives* 10.1 (Mar. 1996), pp. 87–104.

[39] D. Harenberg, S. Marelli, B. Sudret, and V. Winschel. "Uncertainty Quantification and Global Sensitivity Analysis for Economic Models". In: *Available at SSRN 2908760* (2017).

[40] T. Hintermaier and W. Koeniger. "The method of endogenous gridpoints with occasionally binding constraints among endogenous variables". In: *Journal of Economic Dynamics and Control* 34.10 (2010), pp. 2074–2088.

[41] E. T. Jaynes. "Information Theory and Statistical Mechanics". English. In: *Physical review* 106.4 (May 1957), pp. 620–630.

[42] E. T. Jaynes. "Information Theory and Statistical Mechanics. II". English. In: *Physical review* 108.2 (Oct. 1957), pp. 171–190.

[43] E. T. Jaynes. "On the rationale of maximum-entropy methods". In: *Proceedings of the IEEE* 70.9 (Sept. 1982), pp. 939–952.

[44] E. T. Jaynes. "Prior probabilities". In: *IEEE Transactions on Systems Science and Cybernetics* 4.3 (1968), pp. 227–241.

[45] K. L. Judd. *Numerical methods in economics*. The MIT press, 1998.

[46] K. L. Judd, L. Maliar, S. Maliar, and R. Valero. "Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain". In: *Journal of Economic Dynamics and Control* 44 (2014), pp. 92–123.

[47] M. Juillard and S. Villemot. "Multi-country real business cycle models: Accuracy tests and test bench". In: *Journal of Economic Dynamics and Control* 35.2 (2011), pp. 178–185.

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

[49] D. Krueger and F. Kubler. "Pareto-Improving Social Security Reform when Financial Markets are Incomplete!?" In: *American Economic Review* 96.3 (June 2006), pp. 737–755.

[50] P. Krusell and A. A. Smith Jr. "Income and wealth heterogeneity in the macroeconomy". In: *Journal of Political Economy* 106.5 (1998), pp. 867–896.

[51] H. W. Kuhn and A. W. Tucker. "Nonlinear Programming. " In: *2nd Berkeley Symposium on Mathematical Statistics and Probability*.

[52] F. Y. Kuo, C. Schwab, and I. H. Sloan. "Quasi-Monte Carlo Finite Element Methods for a Class of Elliptic Partial Differential Equations with Random Coefficients". In: *SIAM Journal on Numerical Analysis* 50.6 (2012), pp. 3351–3374.

[53] F. E. Kydland. "On the econometrics of world business cycles". In: *European Economic Review* 36.2 (1992), pp. 476–482.

[54] L. Ljungqvist and T. Sargent. *Recursive macroeconomic theory*. Mit Press, 2000.

[55] D. J. Lucas. "Asset pricing with undiversifiable income risk and short sales constraints: Deepening the equity premium puzzle". In: *Journal of Monetary Economics* 34.3 (1994), pp. 325–341.

[56] T. W. Lukaczyk, P. Constantine, F. Palacios, and J. J. Alonso. "Active subspaces for shape optimization". In: *10th AIAA Multidisciplinary Design Optimization Conference*. 2014, p. 1171.

[57] X. Ma and N. Zabaras. "An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations". In: *J. Comput. Phys.* 228.8 (2009), pp. 3084–3113.

[58] X. Ma and N. Zabaras. "An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations". In: *Journal of Computational Physics* 228.8 (2009), pp. 3084–3113.

[59] L. Maliar and S. Maliar. "Chapter 7 - Numerical Methods for Large-Scale Dynamic Economic Models". In: *Handbook of Computational Economics Vol. 3*. Ed. by K. Schmedders and K. L. Judd. Vol. 3. Handbook of Computational Economics. Elsevier, 2014, pp. 325–477.

[60] L. Maliar and S. Maliar. "Merging simulation and projection approaches to solve high-dimensional problems with an application to a new Keynesian model". In: *Quantitative Economics* 6.1 (2015), pp. 1–47.

[61] S. Maliar, L. Maliar, and K. L. Judd. *Solving the Multi-Country Real Business Cycle Model Using Ergodic Set Methods*. Working Paper 16304. National Bureau of Economic Research, Aug. 2010.

[62] B. A. Malin, D. Krüger, and F. Kübler. "Solving the Multi-Country Real Business Cycle Model using a Smolyak-Collocation Method". In: *Journal of Economic Dynamics and Control* 35.2 (Oct. 2010), pp. 229–239.

[63] W. J. Morokoff and R. E. Caflisch. "Quasi-Monte Carlo integration". In: *Journal of Computational Physics* 122.2 (1995), pp. 218–230.

[64] A. Murarasu, J. Weidendorfer, G. Buse, D. Butnaru, and D. Pflüeger. "Compact Data Structure and Parallel Alogrithms for the Sparse Grid Technique". In: *16th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. 2011.

[65] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[66] A. Norets. "Estimation of Dynamic Discrete Choice Models Using Artificial Neural Network Approximations". In: *Econometric Reviews* 31.1 (2012), pp. 84–106.

[67] A. O'Hagan. "Monte Carlo Is fundamentally unsound". In: *Statistician* 36.2-3 (1987). J4389 Times Cited:13 Cited References Count:3, pp. 247–249.

[68] D. Pflüger. "Spatially Adaptive Sparse Grids for High-Dimensional Problems". PhD thesis. München: Institut für Informatik, Technische Universität München, Aug. 2010.

[69] C. Plagemann, K. Kersting, and W. Burgard. "Nonstationary Gaussian Process Regression Using Point Estimates of Local Smoothness". In: *Machine Learning and Knowledge Discovery in Databases, Part Ii, Proceedings* 5212 (2008), pp. 204–219.

[70] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd ed. New York, NY, USA: Cambridge University Press, 2007.

[71] C. E. Rasmussen. "The Infinite Gaussian Mixture Model". In: *In Advances in Neural Information Processing Systems 12*. MIT Press, 2000, pp. 554–560.

[72] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[73] J. Rust. "Dynamic Programming, Numerical". In: *Wiley StatsRef: Statistics Reference Online*. John Wiley & Sons, Ltd, 2014.

[74] J. Rust. "Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher". In: *Econometrica* 55.5 (Sept. 1987), pp. 999–1033.

[75] J. Rust. "Using Randomization to Break the Curse of Dimensionality". In: *Econometrica* 65.3 (1997), pp. 487–516.

[76] J. P. Rust. "Numerical dynamic programming in economics". In: *Handbook of Computational Economics*. Ed. by H. M. Amman, D. A. Kendrick, and J. Rust. 1st ed. Vol. 1. Elsevier, 1996. Chap. 14, pp. 619–729.

[77] A. Skjellum, W. Gropp, and E. Lusk. *Using MPI*. MIT Press, 1999.

[78] C. Smith Ralph. *Uncertainty quantification: Theory, implementation, and applications*. SIAM, 2014.

[79] R. C. Smith. *Uncertainty Quantification: Theory, Implementation, and Applications*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2013.

[80] N. L. Stokey, R. E. Lucas, Jr., and E. C. Prescott. *Recursive Methods in Economic Dynamics*. Cambridge, MA: Harvard University Press, 1989.

[81] C.-L. Su and K. L. Judd. "Constrained Optimization Approaches to Estimation of Structural Models". In: *Econometrica* 80.5 (2012), pp. 2213–2230.

[82] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2004.

[83] C. I. Telmer. "Asset-pricing Puzzles and Incomplete Markets". In: *The Journal of Finance* 48.5 (1993), pp. 1803–1832.

[84] R. Tripathy, I. Bilionis, and M. Gonzalez. "Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation". In: *Journal of Computational Physics* 321 (2016), pp. 191–223.

[85] H. Uhlig. *A Toolkit for Analysing Nonlinear Dynamic Stochastic Models Easily*. 1998.

[86] S. Wold, K. Esbensen, and P. Geladi. "Principal component analysis". In: *Chemometrics and Intelligent Laboratory Systems* 2.1-3 (Aug. 1987), pp. 37–52.

[87] D. Xiu and G. E. Karniadakis. "The Wiener–Askey Polynomial Chaos for Stochastic Differential Equations". In: *SIAM Journal on Scientific Computing* 24.2 (2002), pp. 619–644.

[88] A. Younes, T. A. Mara, N. Fajraoui, F. Lehmann, B. Belfort, and H. Beydoun. "Use of global sensitivity analysis to help assess unsaturated soil hydraulic parameters". In: *Vadose Zone Journal* 12.1 (2013).

[89] C. Zenger. "Sparse Grids". In: *Parallel Algorithms for Partial Differential Equations*. Ed. by W. Hackbusch. Vol. 31. Notes on Numerical Fluid Mechanics. Vieweg, 1991, pp. 241–251.