Computational Economics (2020) 55:925–955
https://doi.org/10.1007/s10614-019-09921-y

Solving Stochastic Dynamic Programming Problems: A Mixed Complementarity Approach

Wonjun Chang[1] · **Michael C. Ferris**[4,5] · **Youngdae Kim**[2] · **Thomas F. Rutherford**[3,5]

Accepted: 24 September 2019 / Published online: 4 October 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract
We present a mixed complementarity problem (MCP) formulation of continuous state dynamic programming problems (DP-MCP). We write the solution to projection methods in value function iteration (VFI) as a joint set of optimality conditions that characterize maximization of the Bellman equation; and approximation of the value function. The MCP approach replaces the iterative component of projection based VFI with a one-shot solution to a square system of complementary conditions. We provide three numerical examples to illustrate our approach.

Keywords Dynamic Programming · Computable general equilibrium · Complementarity · Computational methods

1 Introduction

Dynamic programming (DP) is a standard optimization principle used to solve dynamic optimization problems. Due to the simple yet flexible feature of the Bellman equa-

This research was partially supported by the Electric Power Research Institute (EPRI). We would like to acknowledge the input of Richard Howitt for helpful comments and discussion. Rutherford thanks Wouter den Haan for his lectures and homework assignments in the Macroeconomics Summer School at the London School of Economics (2014).

✉ Wonjun Chang
wchang@crai.com

1 CRA International, Washington, D.C., USA

2 Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, USA

3 Department of Agricultural and Applied Economics, University of Wisconsin-Madison, Madison, WI, USA

4 Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI, USA

5 Optimization Group, Wisconsin Institutes for Discovery, Madison, WI, USA

tion, DP serves as a tractable solution method for dynamic optimization problems, especially when uncertainty is involved.

Over the past two decades, advances in numerical integration and approximation methods have made the application of DP to numerical economic modeling more accessible. Models are now less limited by algebraic tractability and can make more room for essential details of the economic system (Judd 1998; Rust 1996; Manuelli and Sargent 2009; Wright and Nocedal 1999; Miranda and Fackler 2004). Aided by the increasing use of DP in large scale economic models, numerical solution methods that can deal with a great number of continuous and discrete state variables have also been researched (Judd et al. 2014; Maliar and Maliar 2014, 2015; Powell 2011). As a result, there exists a multitude of DP based algorithms, each designed to achieve solution accuracy and computation efficiency for various types of numerical problems.

One field that benefits from these methodological advancements in particular, is climate economics. Stochastic recursive formulations of climate integrated assessment models (IAMs) are important tools in assessing policy implications of the uncertainty that surround climate change. However, given the complexity and size of even the simplest climate IAMs, the application of DP would be severely restricted without the use of advanced numerical methods.

A popular and effective solution method used for stochastic IAMs is value function iteration (VFI), in which projection methods are employed to approximate the value function (Cai et al. 2012; Lemoine and Traeger 2014, 2016; Rudik 2016; Traeger 2014a, b; Cai and Judd 2015). Prevalence of VFI as a solution method in climate, environmental and resource economics has called for surveys on frontier numerical methods that further strengthen VFI as an effective and viable solution method in these fields (Lemoine and Rudik 2017; Cai 2018; Cai and Judd 2014). In general, projection based VFI methods are widely considered workhorse solution methods for discrete-time DP problems.

This paper presents a mixed complementarity problem (MCP)[1] formulation of projection based VFI for discrete time, continuous state DP problems (DP-MCP). To illustrate our approach, we use the collocation method illustrated in Judd (1998) and Miranda and Fackler (2004). VFI collocation (henceforth simply referred to as VFI) is a standard fixed-point solution method in DP, used widely for its monotonic convergence properties and straightforward implementation. Despite its stability however, conventional VFI has several drawbacks that limit application to complex models.

The first is that convergence is slow, which is particularly the case in economic growth models with a discount factor close to unity. A second drawback is the curse of dimensionality. In VFI, the cost of computation increases exponentially in the number of state variables, rendering the solution method intractable for large scale applications.[2] We also point to the iterative component itself as a drawback. Iteration between function approximation and value maximization can make VFI implementation time consuming, which is made worse by aforementioned weaknesses.

[1] Generalization of nonlinear complementarity problems.

[2] We note that the level of the curse of dimensionality depends on the methods employed in VFI. As we demonstrate later in this paper, the use of non-product approximation methods such as complete Chebyshev polynomials, or the use of Smolyak sparse grids (Smolyak 1963), can mitigate the curse of dimensionality. See section on the curse of dimensionality in Cai (2018) for further discussion.

Reformulating conventional VFI as a complementarity problem removes the iterative aspect of the algorithm. The MCP of VFI is written as a square system of equilibrium constraints that encompass:

a. Bellman's optimality conditions with respect to a vector of control variables, a, *given a vector of coefficients in value function approximant*, α;
b. optimality conditions for function approximation with respect to the vector of coefficients in value function approximant , α, *given a vector of control variables,* a;

for which the solution is a pair (a, α) that characterizes an equilibrium between the two alternating objectives of VFI. And although the MCP formulation per se does not attend to the curse of dimensionality, the use of computational methods such as Smolyak sparse grids (Smolyak 1963; Judd et al. 2014; Maliar and Maliar 2014) and Nonlinear Certainty Equivalent Approximation (Cai et al. 2017) extends the application of our method to large scale DP problems.

The practicality of our solution method however comes at a cost. Formulating the complementary conditions of VFI involves the tedious task of hand-coding both primal and dual equations, which can double the scope for coding error. We hence introduce a framework for implementation–the use of the Extended Mathematical Programming (EMP) framework in GAMS (Kim and Ferris 2019; Ferris et al. 2009) mitigates such errors by automating the formulation of the MCP. The EMP framework also enables the modeler to flexibly adapt the solve procedure to the properties of the DP problem, further enhancing computational performance.

Our interest in the use of complementarity methods for solving dynamic programming problems was inspired by the work of Dubé et al. (2012) and Su and Judd (2012). Their work addresses structural estimation of discrete choice problems in a math programming with equilibrium constraints (MPEC) framework. Equilibrium conditions are used to obtain solutions to the Bellman equation in a one-shot fashion. Our objective is to demonstrate how their use of equilibrium conditions can be extended to discrete-time, continuous-choice problems, particularly to ones that feature occasionally binding constraints in the form of complementary conditions.

The paper lays out three sample applications: two standard nonlinear problems of stochastic economic growth; and a complementarity problem, with which we demonstrate the extension of DP-MCP to DP problems that are more naturally formulated as an MCP. To keep things simple, we deal with infinite-horizon problems and use Chebyshev collocation as the default projection method. For further simplicity, we minimize the sum of square residuals to compute the coefficient vector of the function approximant.

The paper is organized as follows. Section 2 provides an introduction to complementarity problems and presents the DP-MCP formulation of VFI. Section 3 provides an overview of the EMP framework and provides the EMP representation of DP-MCP. In Sects. 4 and 5, we provide simple numerical examples of stochastic optimal growth models, including a 4-sector model based on Global Trade Analysis Project (GTAP) data. In Sect. 6, we extend our method to a complementarity problem–a DP problem already formulated as an MCP. Section 7 concludes.

2 DP-MCP: Dynamic Programming as a Mixed Complementarity Problem

We write the Bellman equation in a deterministic infinite-horizon setting as follows:

$$V(x) = \max_{a \in A} \left[C(x,a) + \beta V(x')\right] \quad s.t.\ x' = h(x,a) \tag{1}$$

where x is the vector of state variables, A is the action space and C, the immediate contribution function. $\beta \in (0,1)$ is the discount factor. The sets of assumptions that guarantee existence of a unique solution, $V^*(x)$, are found in Judd (1998) and Stokey and Lucas Jr (1989). We know that, if the set of feasible next period states ($x' \in X$) is compact; and $C(\cdot)$ is real-valued, continuous and bounded—a set of assumptions commonly imposed for numerical feasibility—the solution to the above fixed point problem, $V^*(x)$, exists and is unique.

For m collocation points, VFI outputs a single solution set comprised of a set of coefficients, $\{\alpha_l\}_{l=1}^m$, and a set of optimal control values, $\{a_i\}_{i=1}^m$, that solve the Bellman equation.[3] Each iteration of VFI involves solving two optimization problems—approximation of the value function; and maximization of the Bellman equation. Our approach shows that the solution to the converged function iteration process can be expressed as the unique equilibrium that results from solving the two problems simultaneously.

Mathematically, we look for the solution pair, (a^*, α^*), that satisfies:

$$\begin{aligned}
&a^* \in \arg\max_{a \in A} \left[C(x,a) + \beta V(x';\alpha^*)\right] \quad \text{subject to } x' = h(x,a),\\
&\alpha^* \in \arg\min_{\alpha} \left[\sum_i \left(\left[C(x_i,a_i^*) + \beta V(h(x_i,a^*);\alpha)\right] - V(x_i;\alpha)\right)^2\right].\\
&\text{where } a^* = \{a_i^*\}_{i=1}^m,\ \ \alpha^* = \{\alpha_l^*\}_{l=1}^m
\end{aligned}$$

DP-MCP formulates this equilibrium as an MCP—a square system of equations and inequalities that characterizes the Karush–Kuhn–Tucker (KKT) optimality conditions of two nonlinear optimization problems. Formally, an MCP is defined by two components—a function $F : \mathbb{R}^n \to \mathbb{R}^n$ and a rectangular set $B = \{z \in \mathbb{R}^n \mid l_i \le z_i \le u_i,\ \text{for } i = 1,\ldots,n\}$; where $l_i, u_i \in \mathbb{R} \cup \{-\infty, \infty\}$, such that $z \in B$ is a solution to the MCP(B, F) if one of the following holds for all $i = 1,\ldots,n$:

$$\begin{aligned}
&0 \le F_i(z) \quad \text{and} \quad z_i = l_i,\\
&0 = F_i(z) \quad \text{and} \quad l_i \le z_i \le u_i,\\
&0 \ge F_i(z) \quad \text{and} \quad z_i = u_i.
\end{aligned} \tag{2}$$

In shorthand, we write $F(z) \perp l \le z \le u$ to imply (2), also known as complementary conditions. When z is unbounded, or bounded only on one-side, we can also explicitly include a constraint on F; e.g. $0 \le F(z) \perp z \ge l$.

[3] Refer to "Appendix" for an overview of the VFI algorithm.

Given a single optimization problem:

$$\min_{x \geq 0} f(x) \quad \text{subject to } g(x) \geq 0, \tag{3}$$

its KKT conditions are:

$$\begin{aligned} 0 \leq \nabla f(x) - \nabla g(x)\lambda \perp x \geq 0; \\ 0 \leq g(x) \perp \lambda \geq 0. \end{aligned} \tag{4}$$

This is equivalent to the MCP(B, F) with $B = \{(x, \lambda) \mid x \geq 0, \lambda \geq 0\}$ and $F(x, \lambda) = (\nabla f(x) - \nabla g(x)\lambda, g(x))^T$. When there are N optimization problems that make up the equilibrium, we can formulate the MCP by concatenating each problem's KKT conditions. Solving this MCP is therefore equivalent to finding a point satisfying the N KKT conditions simultaneously.

The use of complementarity is not foreign to economic modeling applications. The field of general equilibrium modeling has long used complementarity problems to express the relationships between price and excess demand, profit and activity levels (Mathiesen 1985; Rasmussen and Rutherford 2004; Lau et al. 2002; Rutherford 1995). In general, complementarity relationships are used to characterize the optimality conditions of constrained nonlinear optimization problems.

We now cast VFI in complementarity form. The system of inequalities and equations that make up the MCP is as follows:

$$\begin{aligned} & -\left(\frac{\partial C(x_i, a_i)}{\partial a_i} + \beta \frac{\partial V(x_i{}' \,; \alpha)}{\partial a_i}\right) + \frac{\partial h}{\partial a_i} p_i \perp a_l \leq a_i \leq a_u, \quad \forall i \in \{1, \ldots, m\} \\ & U_i = C(x_i, a_i) + \beta V(x'_i \,; \alpha) \perp U_i \text{ free}, \quad \forall i \in \{1, \ldots, m\} \\ & \frac{\partial \sum_i \left(U_i - V(x_i \,; \alpha)\right)^2}{\partial \alpha_l} = 0 \perp \alpha_l \text{ free}, \quad \forall l \in \{1, \ldots, m\} \\ & x_i{}' = h(x_i, a_i) \perp p_i \text{ free}, \quad \forall i \in \{1, \ldots, m\} \end{aligned} \tag{5}$$

where we define $\alpha = [\alpha_1, \ldots, \alpha_m]$ to be a vector of m coefficients used in the value function approximation.

Once setup as an MCP, a well-established complementarity solver such as PATH can be applied to VFI to obtain a numerical solution to the Bellman equation (Ferris and Munson 2000; Dirkse and Ferris 1995). To guarantee that DP-MCP obtains a global solution, we note that the immediate contribution function, $C(\cdot)$, must be strictly concave, while also satisfying the set of assumptions that guarantee existence of a unique value function.

3 The Extended Mathematical Programming (EMP) Framework

The practicality of our approach however comes with a caveat: the complementarity formulation requires hand-coding of both the primal and dual equations that character-

ize the equilibrium. This doubles the scope for coding error when using an algebraic modeling language such as GAMS. We could avoid such errors by using the Extended Mathematical Programming (EMP) framework for equilibrium problems in GAMS .

The EMP framework is a useful resource in solving non-standard models that require reformulation into accessible models of established math programming classes. In DP-MCP, the EMP framework is used to formulate the corresponding MCP given a high-level characterization of the equilibrium problem as an input. After writing each optimization problem in primal form, the modeler need only specify the names of variables and equations involved in each problem. This information is to be specified by the modeler in the EMP input file (labeled `empfile`), which the framework automatically reads to generate the MCP formulation of VFI.

To demonstrate, we use the deterministic infinite horizon DP previously stated in (7) [MCP formulation stated in (5)]. The corresponding EMP input is written as follows:

```
max U(i) s.t. c(i), xprime(i), a(i), h(i), def_U(i),
              def_C(i), def_xprime(i), def_h(i)

min SUMSQR_RSDL s.t. alpha(l), def_SUMSQR_RSDL
```

The first and second lines state the Bellman maximization problem, and the third line states the function approximation problem. To specify a problem, the modeler need only state the objective term that is maximized or minimized (`max U(i)` in the maximization problem), the list of primal variables (`c(i),...,h(i)`), and lastly the primal equations (`def_U(i),...,def_h(i)`).

Another important feature of EMP is that post-formulation, the framework can easily conform the DP implementation to the properties of the resulting MCP (Kim and Ferris 2018). For instance, if the MCP formulation lacks global convergence properties, a poor initial point can degrade the solve performance significantly—DP-MCP is unable to exploit the locally superlinear (quadratic) convergence rate originally intended via the complementarity formulation. In this case, implementing conventional VFI initially may be advantageous, as it can guarantee global convergence regardless of the starting point. In the EMP framework, the modeler can combine both approaches by first running conventional VFI for small number of iterations and switching to the MCP formulation, once a good starting point for the MCP is obtained. Running `n` iterations of conventional VFI before solving DP-MCP for instance, can be implemented with one line of code added to the solver option file:

```
$echo major_iteration_limit n >> solver.opt
```

This "hybrid approach", can help achieve global convergence in DP-MCP, while preserving its computational performance.

Further note that the Bellman maximization step can be approached in different ways. This step can be solved separately for each grid point; it can also be solved as an aggregate by maximizing the sum of utility over all grid points. When this step is computationally costly, dividing up the independent subproblems of the DP problem and solving the set of problems in parallel can further enhance computational performance (Cai et al. 2015). The EMP framework can easily adapt the DP implementation

to include this "parallel dynamic programming", which is implemented in the same way as the hybrid approach:

```
$echo parallel_jacobi yes  >> solver.opt
```

As such, the compactness and adaptability of the EMP framework in implementing DP-MCP is showcased in the GAMS code "Appendix" of this paper.

4 Stochastic Neoclassical Growth Model

As a numerical exercise, we solve a variant of the well-known stochastic neoclassical growth model. In this model, a social planner picks a consumption trajectory $\{c_t\}_{t=0}^{\infty}$ to solve:

$$\max_{c_t, k_{t+1}} \mathbb{E}\left\{\sum_{t=0}^{\infty}(1-\beta)\beta^t \; u(c_t)\right\}$$

given utility function and resource constraint:

$$u(c_t) = \log c_t, \quad k_{t+1} = z_t k_t^{\phi} - c_t + (1-\delta)k_t,$$
$$k_{t+1} - (1-\delta)k_t \geq 0, \quad c_t \geq 0.$$

We use model parameters from Aruoba et al. (2006), where β (= 0.9896) is the discount factor, ϕ (= 0.4), the capital value share and δ (= 0.0196), the capital depreciation rate. Labor is fixed and the stochastic productivity shock z_t evolves according to a 5-point Markov chain:

$$z_t \in Z = \{4.9327, 4.9664, 5, 5.0336, 5.0673\}$$

The transition matrix $\mathbb{Q}$ is obtained using Tauchen (1986)'s method of discretizing AR(1) processes:

$$\mathbb{Q} = \begin{pmatrix} 0.9727 & 0.0273 & 0 & 0 & 0 \\ 0.0041 & 0.9806 & 0.0153 & 0 & 0 \\ 0 & 0.0082 & 0.9837 & 0.0082 & 0 \\ 0 & 0 & 0.0153 & 0.9806 & 0.0041 \\ 0 & 0 & 0 & 0.0273 & 0.9727 \end{pmatrix}$$

We solve the model with both traditional VFI and DP-MCP to compare solution outputs. In both approaches, we use complete Chebyshev polynomials to estimate the value function. This particular family of orthogonal polynomials is used due to favorable convergence properties and accuracy in function interpolation. Further discussion on the application of orthogonal polynomials and Chebyshev interpolation can be found in Judd (1998).

DP-MCP Formulation

We use a fourth degree Chebyshev polynomial interpolation with five collocation points for each state variable—capital and productivity levels. The Bellman equation for the single sector stochastic growth model is written as follows:

$$\begin{aligned} V(k,z) = \max_{c} \ & (1-\beta)u(c) + \beta\mathbb{E}\left\{V(k',z')|z\right\} \\ s.t. \ & u(c) = \log c, \\ & k' = zk^{\phi} - c + (1-\delta)k, \\ & k' - (1-\delta)k \geq 0, \\ & c \geq 0, \end{aligned}$$

where k' and z' are respectively the capital and productivity levels in the next period. With collocation indices (i, j) for each state variable, we discretize the expected value operator using the probabilities in transition matrix, $q_{j,j'} \in Q$; and approximate the value function with a complete Chebyshev polynomial; i.e.

$$V(k', z' \ ; \alpha) = \sum_{0 \leq s+t \leq 4} \alpha_{s,t} T_s^c(k') T_t^c(z');$$

where $T_k^c : [-1, 1] \to [-1, 1]$ is the kth order Chebyshev basis function. α represents the vector of coefficients in the approximant. We can thus rewrite the above problem. For all (i, j);

$$\begin{aligned} V(k_i, z_j) = \max_{c_{i,j}} \ & (1-\beta)u(c_{i,j}) + \beta \sum_{j'} q_{j,j'} \sum_{0 \leq s+t \leq 4} \alpha_{s,t} T_s^c(k'_{i,j}) T_t^c(z'_j) \\ s.t. \ & u(c_{i,j}) = \log c_{i,j}, \\ & k'_{i,j} = z_j k_i^{\phi} - c_{i,j} + (1-\delta)k_i, \\ & k'_{i,j} - (1-\delta)k_i \geq 0, \\ & c_{i,j} \geq 0, \end{aligned}$$

The corresponding DP-MCP is written as:

$$(1-\beta)\frac{\partial u(c_{i,j})}{\partial c_{i,j}} \leq p_{i,j} \perp c_{i,j} \geq 0, \quad \forall (i,j)$$

$$\beta \frac{\partial \sum_{j'} q_{j,j'} \sum_{0 \leq s+t \leq 4} \alpha_{s,t} T_s^c(k_{i,j}{}') T_t^c(z_j{}')}{\partial k_{i,j}{}'} \leq p_{i,j} + P_{i,j} \perp k_{i,j}{}' \geq 0, \quad \forall (i,j)$$

$$U_{i,j} = (1-\beta)u(c_{i,j}) + \beta \sum_{j'} q_{j,j'} \sum_{0 \leq s+t \leq 4} \alpha_{s,t} T_s^c(k_{i,j}{}') T_t^c(z_j{}') \perp U_{i,j} \text{ free}, \quad \forall (i,j)$$

$$\frac{\partial \sum_{i,j} \left(U_{i,j} - \sum_{0 \leq s+t \leq 4} \alpha_{s,t} T_s^c(k_i) T_t^c(z_j)\right)^2}{\partial \alpha_{s,t}} = 0 \perp \alpha_{s,t} \text{ free}, \quad \forall (s,t)$$

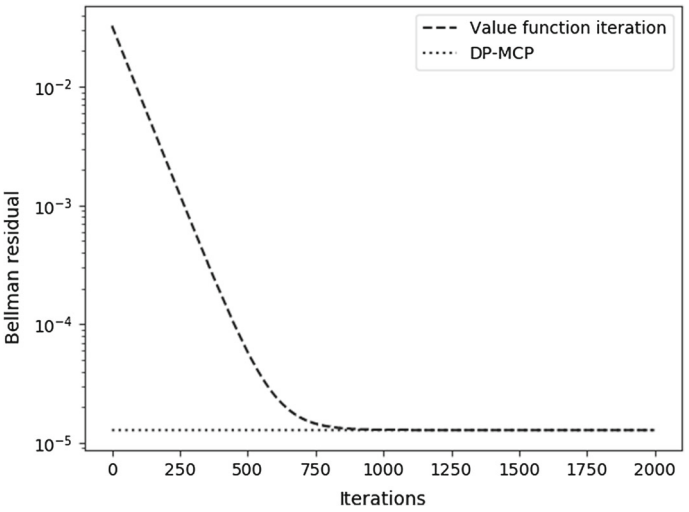

Fig. 1 Comparing Bellman equation residuals for VFI and DP-MCP logarithmic scale is used to plot residual values

$$
\begin{aligned}
&k_{i,j}{}' = z_j k_i^{\phi} - c_{i,j} + (1-\delta)k_i \perp p_{i,j} \text{ free}, \quad \forall (i,j) \\
&(1-\delta)k_i - k_{i,j}{}' \leq 0 \perp P_{i,j} \leq 0, \quad \forall (i,j).
\end{aligned} \tag{6}
$$

We perform accuracy checks to *a.* demonstrate that the one-shot approach replicates results from a fully converged VFI procedure; and *b.* to evaluate the quality of the value function approximation.

We first note that, although the standard Chebyshev collocation method is used to demonstrate implementation of DP-MCP, the choice of projection method and the details that ensure solution accuracy are not tied to our approach. Our method reformulates projection based VFI as an MCP, taking the VFI setup (state variable bounds, polynomial choice, etc.) as given. It is therefore intuitive to measure how close the MCP approximant is to the approximant output by a fully converged VFI process.

For this reason we report Bellman equation residuals as a measure of algorithm performance. The residual is a measure of the distance between the value function approximant and the functional fixed point of the Bellman equation at the collocation nodes. The residual that results from DP-MCP is compared to that from sufficiently converged VFI. See Fig. 1 for results. For VFI, we take the maximum absolute Bellman residual over the collocation nodes and plot the values across iteration count. For DP-MCP, we compute the same metric once, post-implementation of DP-MCP.

Figure 1 shows that the value function approximant from VFI gradually approaches that of DP-MCP. With a tolerance level set to 1E-6 using the infinity norm over absolute deviations of value function coefficients, the VFI process concludes in 796 iterations. However, to replicate the Bellman equation residual value output by DP-MCP (1.28E-5), VFI would have to run at least 1095 iterations, at which point the Bellman residuals for the two approaches become equal.

Table 1 $\log_{10}$ of normalized errors relative to true consumption value

Error metric	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$
$\log_{10} \|E\|_1$	-3.682	-4.499	-5.245	-5.666	-5.968	-6.034
$\log_{10} \|E\|_\infty$	-3.461	-4.31	-5.011	-5.299	-5.587	-5.653

n denotes number of collocationnodes

To evaluate the accuracy of the value function approximation, we implement an instance of the problem in which capital fully depreciates every period ($\delta = 1$). A closed form expression of decision rules exist in this setting, making it possible to assess the true error in the approximated solution over the state domain. We report actual errors for the consumption decision rule ($(1 - \alpha\beta)zk^\phi$), from 1000 random draws of state variable levels from their respective domains. Simulated consumption levels (c_t) are checked against the true consumption value ($\bar{c}_t$) and then normalized by the true consumption value. The error term (E) is thus expressed as:

$$E = \left| \frac{c_t - \bar{c}_t}{\bar{c}_t} \right|.$$

We construct two norms of E—the average ($\|E\|_1$) and the maximum ($\|E\|_\infty$). Table 1 displays the base 10 logarithms of these norms, by number of collocation nodes used. In the case with five collocation points, the error on average is approximately one dollar per every 1000 dollars. Approximation improves significantly as we increase the number of collocation points—at ten collocation points, average error is reduced to a dollar per million.

The tedious task of hand-coding the complementary conditions can be avoided through a few lines of code in the EMP framework. The following EMP input generates the MCP for the stochastic neoclassical growth model:

```
min SUMSQR_RSDL s.t. alpha(s,t), def_SUMSQR_RSDL
max U(i,j) s.t. c(i,j), k(i,j),
        def_U(i,j), def_KPrime(i,j), nonneg_Inv(i,j)
```

Note that this syntax automatically generates the first order conditions coupled with the corresponding dual variables, as specified in (6).

5 N-Sector Stochastic Growth Model

In this section we incorporate multiple sectors to the stochastic growth model, with perfectly correlated productivity shocks across sectors. A social planner chooses the consumption ($\{c_t^s\}_{t=0}^\infty$), investment ($\{I_t^s\}_{t=0}^\infty$), and labor supply trajectory ($\{L_t^s\}_{t=0}^\infty$), given the inverse of the elasticity of intertemporal substitution parameter ($\gamma = 0.5$),

to solve the following optimization problem:

$$\max_{\{c_t^s, I_t^s, Y_t^s\}} \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t \frac{u(c_t^1, \ldots, c_t^n)^{1-\gamma}}{1-\gamma}\right]$$

such that the following constraint equations are satisfied:

- Cobb–Douglas utility function with sectoral reference consumption levels, $\bar{c}^s$, and expenditure share, η_s:

$$u(c_t^1, \ldots, c_t^n) = \prod_{s=1}^{n} \left(\frac{c_t^s}{\bar{c}^s}\right)^{\eta_s}, \quad \sum_{s=1}^{n} \eta_s = 1;$$

- the law of motion for capital accumulation with capital depreciation rate, δ (= 0.07):

$$K_{t+1}^s = (1-\delta)K_t^s + I_t^s \quad \forall s;$$

- market clearing conditions for labor with labor supply fixed to the sum of all sectoral reference labor supply values:

$$\bar{L} = \sum_{s=1}^{n} L_t^s \quad \forall t;$$

- Cobb–Douglas production function with factor share of capital, α_s, productivity shock, z_t, and sectoral reference levels for output ($\bar{Y}^s$), capital ($\bar{K}^s$), and labor supply ($\bar{L}^s$). ψ_s denotes the magnitude of impact of the productivity shock to each sector:

$$Y_t^s = (1 + \psi_s(e^{z_t} - 1))\bar{Y}^s \left(\frac{K_t^s}{\bar{K}^s}\right)^{\alpha_s} \left(\frac{L_t^s}{\bar{L}^s}\right)^{1-\alpha_s} \quad \forall s;$$

- stochastic productivity shock, z_t is a random variable that follows the process:

$$z_t = \rho z_{t-1} + \epsilon_t, \ \epsilon \sim N(0, \sigma^2), \text{ with } \rho = 0.95 \text{ and } \sigma = 0.007$$

- and lastly, the equation describing the market for current output, where $\Lambda(s, s')$ is the unit demand for intermediate good s in sector ss.

$$Y_t^s = c_t^s + I_t^s + \sum_{s'=1}^{n} \Lambda(s, s') Y_t^{s'} \quad \forall s.$$

Table 2 Sector specific reference values

Parameter	Agriculture	Manufacturing	Services	Energy
η_s	0.005	0.247	0.723	0.025
α_s	0.518	0.438	0.203	0.730
ψ_s	0.278	0.222	0.148	0.353
$\bar{c}^s$	52.58	2822.32	8243.39	290.01
$\bar{K}^s$	50.17	1747.22	1488.44	210.57
$\bar{L}^s$	46.77	2246.11	5833.64	77.91
$\bar{Y}^s$	273.61	9074.22	13, 302.21	1153.50

The Bellman equation for the n-sector stochastic growth model is written as follows:

$$
\begin{aligned}
V_t(x) = \max_{\{c^s\}_{s=1}^n, \{I^s\}_{s=1}^n, \{L^s\}_{s=1}^n} & \frac{u(c^1, \ldots, c^n)^{1-\gamma}}{1-\gamma} + \beta \mathbb{E}\left\{ V_{t+1}(K^{1'}, K^{2'}, \ldots, K^{n'}, z') | z \right\} \\
s.t. \quad & u(c^1, \ldots, c^n) = \prod_{s=1}^n \left(\frac{c^s}{\bar{c}^s} \right)^{\eta_s}, \quad \sum_{s=1}^n \eta_s = 1; \\
& K^{s'} = (1-\delta) K^s + I^s \quad \forall s; \\
& \bar{L} = \sum_{s=1}^n L^s \\
& Y^s = (1 + \psi_s (e^z - 1)) \cdot \bar{Y}^s \left(\frac{K^s}{\bar{K}^s} \right)^{\alpha_s} \left(\frac{L^s}{\bar{L}^s} \right)^{1-\alpha_s} \quad \forall s; \\
& z' = \rho z + \epsilon, \ \ \epsilon \sim N(0, \sigma^2) \\
& Y^s = c^s + I^s + \sum_{s'=1}^n \Lambda(s, s') Y^{s'} \quad \forall s.
\end{aligned}
$$

Again we use a fourth degree complete Chebyshev polynomial interpolation with five interpolation points for each state variable, $k_i \in [0, 5000]$ and $z_j \in [0.4, 1.6]$ for $i = \{1, \ldots, 5\}$ and $j = \{1, \ldots, 5\}$.

4-Sector Stochastic Growth Model

As a numerical example, we solve a four sector stochastic optimal growth model. The description of sectors and model parameters, derived from the GTAP8 database (Aguiar et al. 2012), is summarized in Tables 2 and 3.

To obtain a numerical solution, we first apply Gauss–Hermite quadrature and discretize the expected value operator. This step amounts to discretizing the AR(1) process using Gauss–Hermite quadrature. For random variable $y \sim N(\mu, \sigma)$, a

Table 3 Intermediate demand of goods (s) in sectors (ss)

$\Lambda(s, s')$	Agriculture	Manufacturing	Services	Energy
AGR	35.03	167.59	18.06	0.010
MFR	71.88	2683.27	1352.60	125.590
SER	67.54	1397.65	3167.37	76.860
ENR	8.68	378.30	154.61	321.900

Gauss–Hermite approximation of the expected value $\mathbb{E}[h(y)]$ of some function $h(\cdot)$ is evaluated as follows:

$$\mathbb{E}[h(y)] = \sum_{i=1}^{m} \frac{1}{\sqrt{\pi}} \omega_i^{GH} h\left(\mu + \sigma\sqrt{2}\zeta_i\right) = \sum_{i=1}^{m} \overline{\omega}_i h(\mu + \overline{\zeta}_i),$$

where ζ_i and ω_i represent the original Gauss–Hermite nodes and weights. Given grid points $i \in \{1, \ldots, m\}$, and disturbance term $\epsilon \sim N(0, \sigma)$, where σ is the standard deviation of the stochastic disturbance, $\overline{\zeta}_i$ then represents the normalized disturbance associated with Gauss–Hermite term i. Further discussion on Gaussian quadratures can be found in Miranda and Fackler (2004) and Judd (1998). We apply the quadrature method to our application. The next period stochastic shock, z_{t+1} is a function of normalized weights, $\overline{\omega}_i$; i.e.

$$z_{t+1,i} = \rho z_t + \overline{\zeta}_i$$

For the numerical examples that employ Gauss–Hermite quadratures presented in this paper, we construct a 5-point Gauss–Hermite grid with the nodes and weights specified in Table 5 of the "Appendix".

Second, we use Smolyak's sparse grid (Smolyak 1963) to deal with the curse of dimensionality. Smolyak proposes a sparse grid method that allows for the efficient integration and interpolation of functions on multidimensional hypercubes (Judd et al. 2014). We summarize the main idea of this particular non-product approach, explained in Maliar and Maliar (2014).

The idea of this method is simple. Some elements produced by the tensor product rule matter more in the approximation procedure than others. Smolyak's method ranks the elements of the tensor product by importance. A user-defined approximation level parameter, determines the number of elements included in the sparse grid. The higher the approximation level, the more elements of the tensor product is included. The computational complexity of the problem increases polynomially in the dimension of the state space, providing a way to avoid the curse of dimensionality in large-scale applications. Illustration of the method in a two-dimensional state space is included in the "Appendix".

Our implementation of the model involved five unidimensional grid points for each of the five state variables, with an approximation level of $\mu = 2$. The Smolyak grid

Table 4 Mean monthly inflow to reservoir (million cubic meters)

Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Inflow	1.2	1.0	1.1	1.2	40.2	99.5	146.3	138.2	70.7	11.7	2.3	1.5

under this specification consists of 61 grid points. Note that the EMP framework only requires localized changes to the DP program—namely, setting up the grid and defining the polynomial form of the function approximant. Finding the equilibrium is again automated via a few lines of EMP syntax.

Third, to provide the PATH solver with a good starting point, we use the nonlinear certainty equivalent approximation (NLCEQ) method discussed in Cai et al. (2017). Choosing a good starting point can be challenging, especially when the dimensionality of the problem is high. As the name suggests, the method involves certainty equivalent approximation—approximating the solution to the stochastic infinite horizon DP problem by solving a finite-horizon optimization problem with perfect foresight. By setting initial variable values to grid values at each collocation node, solving the certainty equivalent problem can provide a good estimate of both the value and corresponding optimal decisions at each node. We implement NLCEQ plus 5 iterations of conventional VFI before passing off the solution to DP-MCP.

6 Stochastic Hydropower Planning

Some economic applications naturally take the form of a complementarity problem as opposed to a nonlinear fixed point or a root finding problem (Miranda and Fackler 2004). These application types include general equilibrium modeling, Nash games, and some applications that involve numerous occasionally binding constraints, all of which require the proper treatment of corner solutions when solving for the optimal policy. We thus conclude our demonstration by applying our solution method to complementarity problems.

The hydropower planning problem is an annual model of a single aggregated reservoir with a monthly release schedule for hydropower generation. The present model is similar in dynamics to the model of water management on the North Platte River in Nebraska presented in Howitt et al. (2002b). We begin with a brief overview of the model.

Two state variables characterize water management at time t, namely L_t, the stock of water in the reservoir and D_t, the inflow of water to the reservoir as a function of precipitation levels. Inflow D_t consists of stochastic deviations from average monthly inflow levels (displayed in Table 4.) characterizing the state variable as following a first order autoregressive process. z_t and r_t denote the water retained and released every month to generate electricity; if the capacity of the reservoir is exceeded, the excess water runs down the spillway and bypasses the power plant. In other words, spills (denoted s_t) balance the system when the reservoir overflows, but have no economic

value in the model. The maximum capacity of the reservoir dam is 250 million cubic meters (MCM).

Lastly, monthly electricity generation must meet a fixed monthly demand of 140 megawatt hours (MWh). In case electricity generated by hydropower does not meet demand, non-hydro electricity generation is employed, incurring marginal cost equal to the market price of electricity.

The Model

Water at the start of the month is either retained or released to generate electricity:

$$L_t \geq z_t + r_t$$

Total generation of electricity through hydro (r_t) and non-hydro (x_t) sources must equal the demand for electricity. Demand (denoted g_t) is fixed in each month:

$$r_t + x_t = g_t$$

The projected level of water at the start of the subsequent month ($\widetilde{L}_{t+1}$) depends on how much water is currently stored (z_t), how much inflow is projected to occur ($\widetilde{D}_{t+1}$) and how much water will be spilled ($\widetilde{s}_{t+1}$). Projected variable levels are represented using a tilde:

$$\widetilde{L}_{t+1} = z_t + \widetilde{D}_{t+1} - \widetilde{s}_{t+1}$$

The price of water in the subsequent month ($\widetilde{p}_{t+1}$) is imputed on the basis of the imputed water price (value of water as a function of the state):

$$\widetilde{p}_{t+1} = V\left(\widetilde{L}_{t+1}, \widetilde{D}_{t+1}\ ; \alpha\right)$$

Lastly, inflow D_t follows an $AR1$ process such that the projected inflow $\widetilde{D}_{t+1}$ is a function of the mean monthly inflow $\overline{D}_t$, the coefficient for rainfall persistence, ρ, the realized inflow levels, D_t, and a normally distributed disturbance term $\epsilon \sim N(0, \sigma)$ that represents stochastic departures from the model.

$$\widetilde{D}_{t+1} = \overline{D}_{t+1}\left[1 + \rho\left(\frac{D_t}{\overline{D}_t} - 1\right) + \epsilon_{t+1}\right]$$

To discretize the $AR1$ process, we again use a 5-point Gauss–Hermite quadrature method. Given normalized Gauss–Hermite disturbance $\overline{\zeta}_i$, associated with normalized weight, $\overline{\omega}_i$, we can rewrite the rainfall projection equation as follows:

$$\widetilde{D}_{t+1,i} = \overline{D}_{t+1}\left(1 + \rho\left(\frac{D_t}{\overline{D}_t} - 1\right) + \overline{\zeta}_i\right) \quad \forall i \in \{1, \ldots, 5\}.$$

The objective is to minimize non-hydro electricity generation and maximize the value of water retained, while meeting the demand for electricity. Every month, the

decision maker's problem is characterized by the following objective function:

$$\max_{r_t} \; -\overline{cx}\left(\frac{x_t(r_t)}{\overline{x}}\right)^{\eta} + \beta\mathbb{E}\left\{\widetilde{p}_{t+1}\widetilde{L}_{t+1}(r_t)|D_t\right\},$$

where $\overline{c}$ is the reference cost of non-hydro generation, $\overline{x}$ is the reference supply of non-hydro generation and η, the elasticity of non-hydro supply. As we estimate the projected shadow price of water, $\widetilde{p}_{t+1}$ using projection, we can rewrite the objective function as follows:

$$\underbrace{-\overline{cx}\left(\frac{x_t}{\overline{x}}\right)^{\eta}}_{contribution} + \beta\underbrace{\mathbb{E}\left\{\left(V\left(\widetilde{L}_{t+1}, \widetilde{D}_{t+1}\;;\alpha\right)\right)\widetilde{L}_{t+1}|D_t\right\}}_{carry-over}.$$

Under this objective, we continue to write the complementary conditions that characterize equilibrium.

The condition on non-hydro electricity generation states that non-hydro electricity generation only takes place when the marginal cost of non-hydro generation equals the market price of electricity (p_t^e):

$$\overline{c}\eta\left(\frac{x_t}{\overline{x}}\right)^{\eta-1} \geq p_t^e \;\perp\; x_t \geq 0.$$

Similarly, hydro electricity generation only takes place when the shadow price of water is equal to that of electricity:

$$p_t \geq p_t^e \;\perp\; r_t \geq 0.$$

Water is only retained to equate the shadow price of water to the projected value of water in the future:

$$p_t \geq \beta\sum_i \overline{\omega_i}\left[\frac{\partial V\left(\widetilde{L}_{t+1,i}, \widetilde{D}_{t+1,i}\;;\alpha\right)\widetilde{L}_{t+1,i}}{\partial z_t}\right] \;\perp\; z_t \geq 0.$$

And lastly, spilling water amounts to free disposal. This assures that the shadow value of water in subsequent months is nonnegative:

$$\frac{\partial V\left(\widetilde{L}_{t+1,i}, \widetilde{D}_{t+1,i}\;;\alpha\right)\widetilde{L}_{t+1,i}}{\partial\widetilde{s}_{t+1,i}} \geq 0 \;\perp\; s_t \geq 0.$$

MCP Formulation

We present the MCP formulation of the hydropower planning problem. For conciseness of notation, we omit the time subscripts:

$$L^j \geq z_{j,k} + r_{j,k} \;\perp\; p \geq 0, \quad \forall(j,k)$$

$$
\begin{aligned}
r_{j,k} + x_{j,k} = g_{j,k} \quad &\perp \quad p^e \geq 0, \quad \forall (j,k) \\
\widetilde{L}_{j,k,i} = z_t + \widetilde{D}_{j,k,i} - \widetilde{s}_{j,k,i} \quad &\perp \quad \widetilde{L}_{j,k,i} \text{ free}, \quad \forall (j,k,i) \\
\widetilde{p}_{j,k,i} = V\big(\widetilde{L}_{j,k,i}, \widetilde{D}_{j,k,i} \ ; \alpha\big) \quad &\perp \quad \widetilde{p}_{j,k,i} \text{ is free}, \quad \forall (j,k,i) \\
\overline{c}\eta \frac{x_{j,k}{}^{(\eta-1)}}{\overline{x}} \geq p^e_{j,k} \quad &\perp \quad x_{j,k} \geq 0, \quad \forall (j,k) \\
p_{j,k} \geq p^e_{j,k} \quad &\perp \quad r_{j,k} \geq 0, \quad \forall (j,k) \\
p_{j,k} \geq \beta \sum_i \overline{\omega_i} \left[\frac{\partial V\big(\widetilde{L}_{j,k,i}, \widetilde{D}_{j,k,i} \ ; \alpha\big) \widetilde{L}_{j,k,i}}{\partial z_{j,k}} \right] \quad &\perp \quad z_{j,k} \geq 0, \quad \forall (j,k) \\
\frac{\partial V\big(\widetilde{L}_{j,k,i}, \widetilde{D}_{j,k,i} \ ; \alpha\big) \widetilde{L}_{j,k,i}}{\partial \widetilde{s}_{j,k,i}} \geq 0 \quad &\perp \quad \widetilde{s}_{j,k,i} \geq 0, \quad \forall (j,k,i)
\end{aligned}
$$

The application of DP-MCP to a complementarity problem is no different from that of a maximization problem. In place of the first order conditions of the maximization problem, we now have a set of complementarity conditions that characterize the natural equilibrium of the economic system.

We solve the hydropower planning problem using a weighted residual method with 10 grid points for both precipitation (D_k) and reservoir water levels (L_k). The value function for the shadow price of water is estimated using a 4th order *complete* Chebyshev polynomial. DP-MCP couples the MCP conditions with the optimality conditions for least-squares value function fitting:

$$
\frac{\partial \sum_{j,k} \big(p_{jk} - V(L_j, D_k \ ; \alpha)\big)^2}{\partial \alpha_l} = 0 \perp \alpha_l \ \text{ free}, \quad \forall l \in \{1, \ldots, 10\}
$$

We choose 10 Chebyshev interpolation nodes for the reservoir water level based on the minimum and maximum water levels permitted for the operation of the dam. Similarly, the nodes for water inflow, D^k, are determined by the average monthly inflow $\overline{D}$ and the normalized standard deviation of inflow σ such that: $D^k \in [\overline{D} - 3\sigma, \overline{D} + 3\sigma], \ k \in \{1, \ldots, 10\}$.

To illustrate the MCP framework's optimal policy at both the intensive and extensive margins, we display the optimal release schedule and approximated shadow prices of water in Figs. 2, 3, 4 and 5. In each of the plots, we hold fixed the reservoir water level to be high (grid point corresponding to highest water level) or low, and vary the precipitation levels. Grid points for precipitation range from 1 to 10, with grid point 1 representing the highest precipitation level. The resulting shadow price of water is high when both the stock of water and inflow are low, and is equal to zero during the rainy months especially when the reservoir is sufficiently filled. The optimal release schedule displays the opposite dynamics as anticipated.

Under the value function obtained, we run a 60 month simulation of stochastic inflows for which we solve for the optimal water release (hydropower generation) trajectory. The optimal release schedule is displayed in Fig. 6.

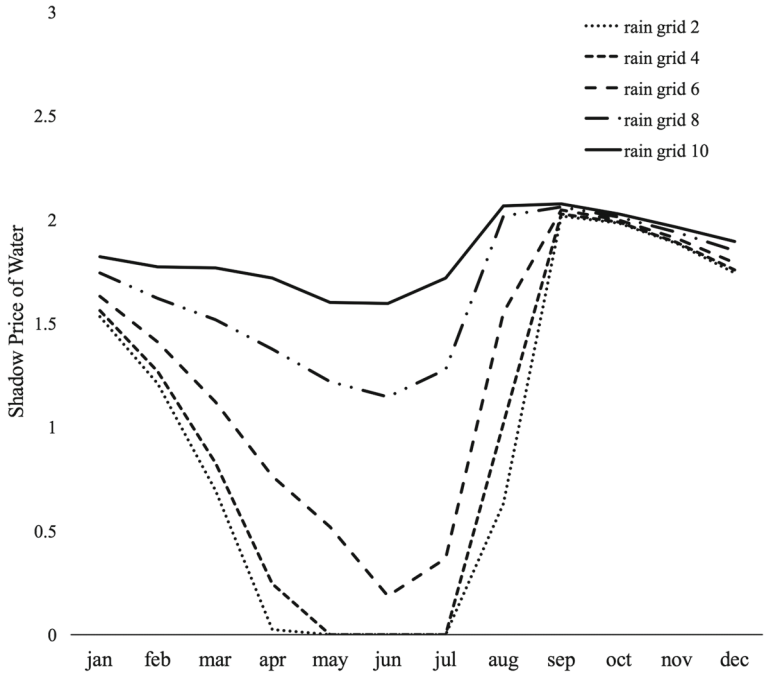

Fig. 2 Shadow price of water approximation for high precipitation scenario

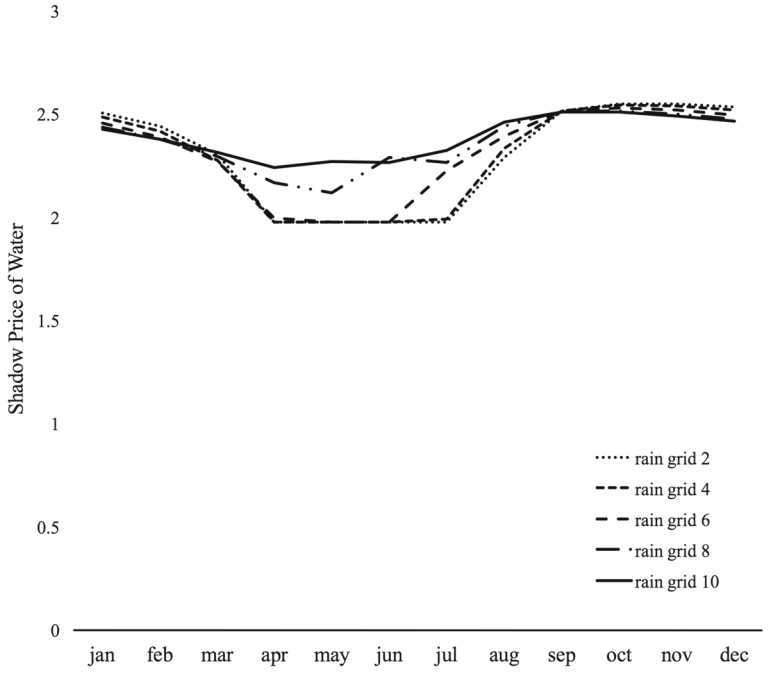

Fig. 3 Shadow price of water approximation for low precipitation scenario

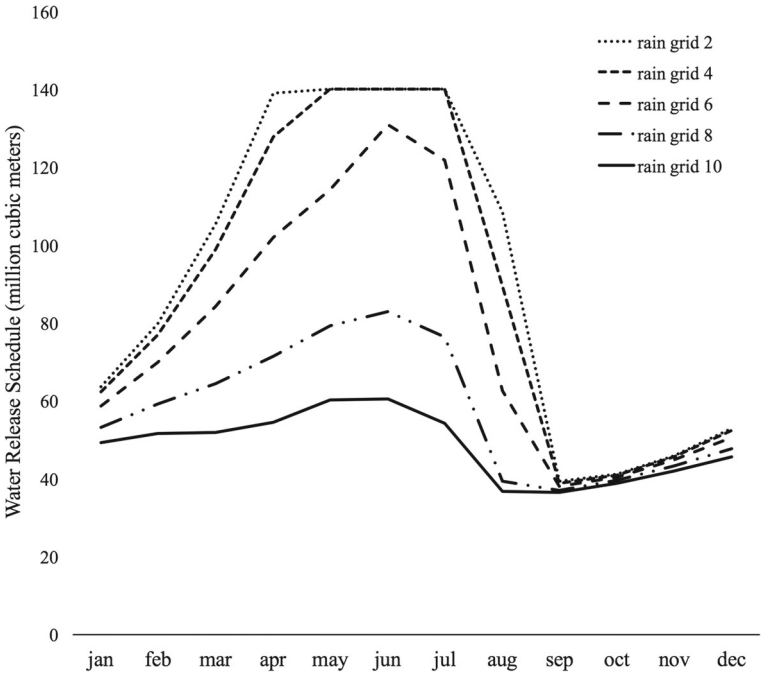

Fig. 4 Release schedule for high precipitation scenario

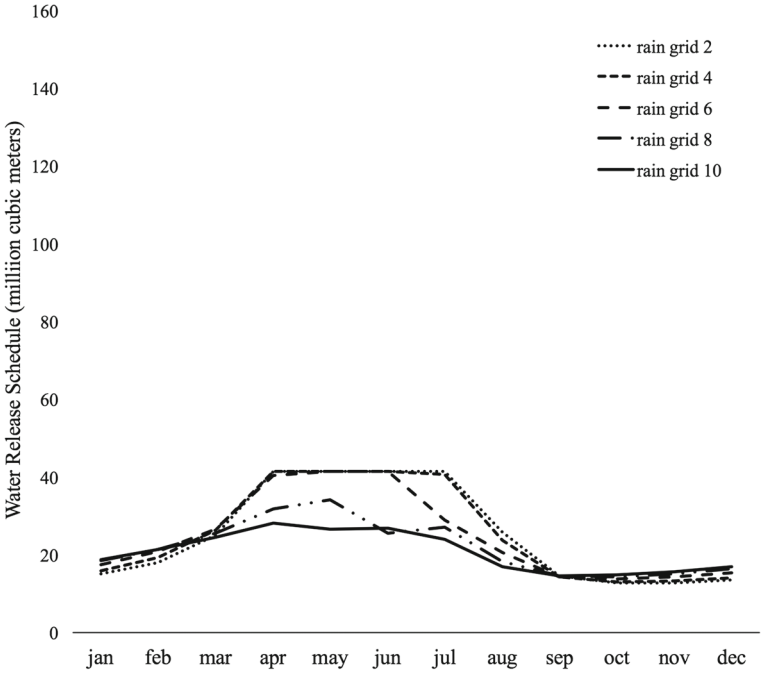

Fig. 5 Release schedule for low precipitation scenario

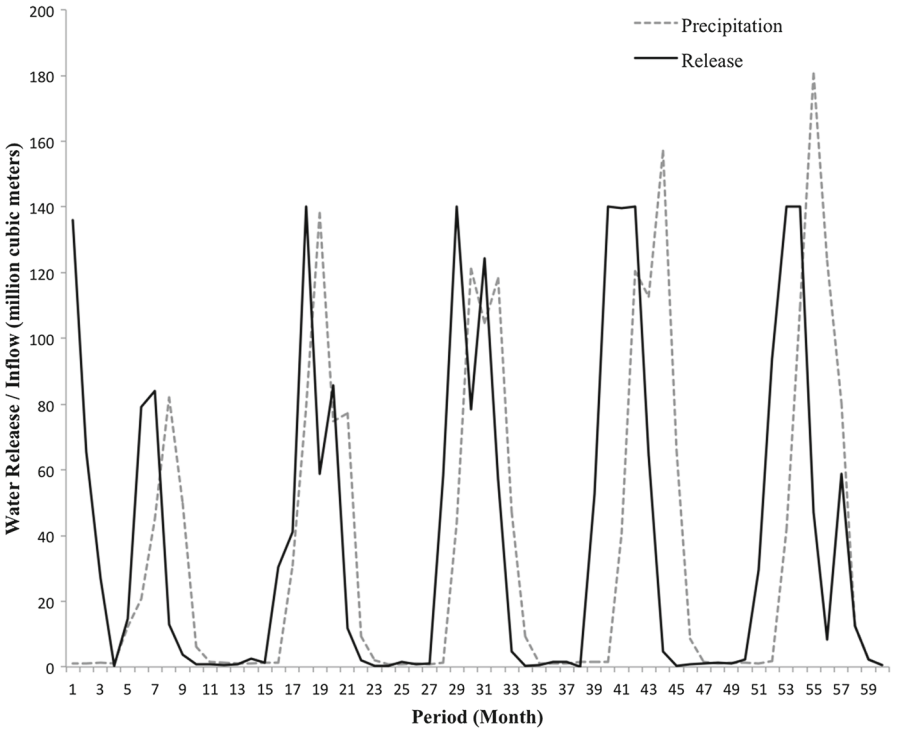

Fig. 6 Simulation result for release schedule

7 Conclusion

We introduce a complementarity (MCP) formulation for discrete time, continuous-state dynamic programming problems. We convert projection-based value function iteration (VFI)—a solution method traditionally based on fixed-point iteration—into a system of equilibrium conditions. These equilibrium conditions consist of equations and inequalities that characterize an equilibrium between two optimization problems of VFI: value function approximation and Bellman equation maximization. The MCP approach (DP-MCP) hence replaces the iterative aspect of VFI with a one-shot solution to a square system of complementary conditions.

DP-MCP however requires both the primal and dual equations to characterize the equilibrium, which can increase the scope for coding error. The use of the Extended Mathematical Programming (EMP) framework in GAMS, can mitigate such errors by automating the formulation and implementation of the MCP. The EMP framework also enables the modeler to readily adapt the solve process to the properties of the DP problem. If the performance of the resulting DP-MCP is relies heavily on an adequate starting point, a hybrid approach—running conventional VFI to provide a good starting point for the complementarity solver—can be deployed with a line of code, to enhance the computational performance of DP-MCP.

Appendix

A. Value Function Iteration Using Collocation

Value function iteration is already a workhorse of numerous studies and textbooks aimed to make DP more accessible to numerical economic models (Howitt et al. 2002a; Aruoba and Fernández-Villaverde 2014; Aruoba et al. 2006; Manuelli and Sargent 2009; Sargent and Stachurski 2015). In this section of the "Appendix", we describe the conventional collocation based VFI algorithm, in which the least-squares norm is used to obtain coefficients for the value function approximant.

The Bellman equation in a deterministic infinite-horizon setting takes the following form:

$$V(x) = \max_{a \in A} \left[C(x,a) + \beta V(x')\right] \quad s.t.\ x' = h(x,a) \tag{7}$$

where x is the vector of state variables, A is the action space and C, the immediate contribution function. $\beta \in (0,1)$ is the discount factor. As a standard numerical algorithm for finding V^*, VFI is motivated by the contraction properties of the Bellman equation. VFI updates the value function via the Bellman operator, given the current estimate of the value function ($V^n(x)$); i.e.

$$V^{n+1}(x) = \max_{a \in A} \left[C(x,a) + \beta V^n(x')\right] \quad s.t.\ x' = h(x,a)$$

By the contraction mapping theorem, the solution to the iterative scheme converges to the true value function for any initial guess $V^0(\cdot)$ (Judd 1998).

Algorithm. Value Function Iteration Using Collocation for Infinite Horizon Problems

1. Set m collocation points in state space and a functional form for $V(x\ ;\alpha)$;
   for $\forall i \leq m$, choose approximation nodes $x_i \in X$;
   fix a tolerance parameter ϵ;
   denote $V^n(x)$ to be value estimate output for iteration count n.
2. Initialize estimate of value function $V^0(x)$.
3. For $n \geq 1$: obtain parameters α^{n-1} $s.t.$ $V(x_i\ ;\alpha^{n-1}) = V^{n-1}(x_i)$
   solve $\min_{\alpha^{n-1}} \left[\sum_i \left(V^{n-1}(x_i) - V(x_i\ ;\alpha^{n-1})\right)^2\right]$
4. For $\forall i$, compute:
   $V^n(x_i) = \max_{a_i \in A} \left[C(x_i,a_i) + \beta V(x_i'\ ;\alpha^{n-1})\right]$ s.t. $x_i' = h(x_i,a_i)$.
5. If $\|V^n - V^{n-1}\| < \epsilon(1-\beta)/2\beta$, **stop**;
   else set $n = n+1$ and go to **step** 3.

The algorithm seeks an approximation to the value function such that the sum of the maximized contribution and the discounted carry-over value, based on the approximant, maximizes the total value function. The value iteration procedure solves for two objectives. The function approximation objective in step 3 computes coefficients of the value function approximant, by minimizing the sum of square deviations between

the maximized Bellman value at each collocation point, and the function approximant. The next objective, (step 4) is to maximize the Bellman value via the control variable, given the coefficients of the approximated value function.

Conventional VFI is stable; yet without the use of techniques that help convergence (e.g. implementing Howard's improvement, exploiting concavity or monotonicity of value function), the procedure is typically known to be slow. The slowness is particularly salient in economic growth models with a discount factor close to unity. Again, without the use of frontier numerical methods, the use of VFI is limited by the curse of dimensionality. Finding the equilibrium quickly becomes a daunting computational task as we increase the dimensions of the state space.

B. Gauss–Hermite Approximation Data

See Table 5

Table 5 Gauss–Hermite approximation data

i	ζ_i	ω_i
1	2.0202	0.02
2	0.9586	0.3936
3	0	0.9453
4	− 0.9586	0.3936
5	− 2.0202	0.02

C. Illustration of Smolyak Sparse Grid Method

We illustrate the method in a two dimensional state space. Consider the extrema of the Chebyshev polynomial function as grid points $\left\{-1, \frac{-1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 1\right\}$. It is not essential to use these points for the Smolyak method. Other unidimensional grid points can be used instead, per the collocation points used in this example. Using the common tensor product would produce 25 grid points $\left\{(-1,-1), (-1, \frac{-1}{\sqrt{2}}), \ldots\right\}$.

The Smolyak grid samples grid points in the following way. We first construct a sequence of nested sets, S_i, that exhaust the unidimensional grid previously defined; i.e.

$$
\begin{aligned}
i = 1 &\rightarrow S_1 = \{0\} \\
i = 2 &\rightarrow S_2 = \{-1, 0, 1\} \\
i = 3 &\rightarrow S_3 = \left\{-1, \frac{-1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 1\right\}
\end{aligned}
$$

From all possible two-dimensional tensor products using elements of sets, S_i, the Smolyak method chooses grid points such that:

$$i_1 + i_2 \leq d + \mu$$

where i_1 and i_2 are the set numbers corresponding to each dimension; d is the number of dimensions of the state space; and μ is the approximation level parameter. With $\mu = 2$, the Smolyak sparse grid consists of 13 points.

The Smolyak polynomials that accompany the sparse grid is constructed in a similar way. Instead of unidimensional grid points, the sequence of sets now correspond to unidimensional Chebyshev basis functions. i.e.

$$\begin{aligned} i = 1 &\rightarrow S_1 = \left\{1\right\} \\ i = 2 &\rightarrow S_2 = \left\{1, x, 2x^2 - 1\right\} \\ i = 3 &\rightarrow S_3 = \left\{1, x, 2x^2 - 1, 4x^3 - 3x, 8x^4 - 8x^2 + 1\right\}. \end{aligned}$$

The GAMS code for the 4-sector model automates the construction of the Smolyak grid and polynomial, given the dimension and approximation level of the problem. Smolyak's method was introduced to dynamic economic modeling in Krueger and Kubler (2004), and is currently used as a popular non-product approach to avoid the curse of dimensionality in numerical DP modeling (Fernández-Villaverde et al. 2015; Lemoine and Rudik 2017).

Appendix: GAMS Code

A. Stochastic Neoclassical Growth Model Data File: data.gms

```
$title DP-MCP: Stochastic Neoclassical Growth Model Using Chebyshev
      Collocation: Data File

Sets   s      State variables
           /cap,phi/,
       ik     Nodes for capital at which value function is evaluated
              /1*5/,
       ip     Nodes for productivity
              /1*5/,
       ic     Dimension of Chebyshev polynomial
              /1*5/,
       iter   Dynamic programming iterations
              /1*10/;

alias (ik,jk)
alias (ip,jp)
alias (ic,jc)
alias (s,ss)

Parameters
       beta   Utility discount factor /0.9869/,
       delta  Depreciation rate       /0.0196/,
       alpha  Capital value share     /0.4/,
```

```
      pi     PI                        /3.141593/;


*     Parameters to define CS Polynomial terms
*     Defined for both capital (K) and productivity (p)
Parameters
      arg_k, arg_p       Argument of cosine weighting function,
      x_k, x_p           Node value for the state variable on the
          unit interval,
      lo_k, lo_p         Lowerbound on stock variable,
      up_k, up_p         Upperbound on stock variable,
      csbar(ic)          Chebyshev polynomial terms,
      cap(ik)            Stock level value at node for grid point
          calculation,
      phi(ip)            Stock level value at node for grid point
          calculation,
      sscap              Steady-state capital;

Parameters
      sigma_s      Unconditional std deviation of phi,
      sigma        Standard deviation of AR1 disturbance term
          /0.007/,
      rho          Persistence term in AR1 process
          /0.95/,
      p_mean       Mean value of productivity phi
          /5/,
      p_std        Standard deviation of phi;

**    Set lower and upper bound on state variables
*     Productivity
sigma_s = sigma/(1-rho**2)**0.5;
lo_p = p_mean - 3 * sigma_s;
up_p = p_mean + 3 * sigma_s;

*     Capital
sscap = (beta * alpha * p_mean)**(1/(1-alpha));
lo_k = 0.7 * sscap;
up_k = 1.3 * sscap;

*     Define basis for chebyshev polynomial expansion
*     Capital
arg_k(ik) = ((2*ord(ik)-1)*pi)/(2*card(ik));
x_k(ik) = cos(arg_k(ik));
cap(ik) = (lo_k+up_k+(up_k-lo_k)*x_k(ik))/2;

*     Productivity
arg_p(ip) = ((2*ord(ip)-1)*pi)/(2*card(ip));
x_p(ip) = cos(arg_p(ip));
phi(ip) = (lo_p+up_p+(up_p-lo_p)*x_p(ip))/2;

*-----------------------------------------------------------------------
*     Define Chebyshev Basis Polynomial Functions
*-----------------------------------------------------------------------

$eval ptcard ( round(card(ic)/2) )

set   pt           Chebyshev polynomial terms /1 * %ptcard%/,
      csp(ic,pt)   Index for Chebyshev polynomial coefficients and
          exponents;

Parameters
      ce(ic,pt)    Exponents of state variables for Chebyshev
          Polynomial terms;

*     Assign polynomial coefficients to express each chebyshev
    polynomial basis
*     e.g. cc("1","1") = coefficient for 1st polynomial term of 1st
     order Chebyshev basis
```

```
table  cc(ic,pt)  Coefficients of state variables for Chebyshev
    Polynomial terms

            1      2      3
      1     1
      2     1
      3     2      1
      4     4      3
      5     8      8      1;

csp(ic,pt) = yes$cc(ic,pt);

*     Exponents are assigned
ce(csp(ic,pt)) = ord(ic)-1 - (ord(pt)-1)*2;

*     Signs of coefficients alternate
cc(csp(ic,pt)) = cc(ic,pt) * power(-1,(ord(pt)-1));

*---------------------------------------------------------------------
*     Apply above parameters to compute chebyshev polynomial bases
*---------------------------------------------------------------------

Parameters
      kbar(ik,ic)       Chebyshev polynomial bases for function
          approximation,
      phibar(ip,ic)     Chebyshev Polynomial bases for function
          approximation;

*---------------------------------------------------------------------

kbar(ik,ic) = sum(pt$csp(ic,pt), cc(ic,pt) * power(x_k(ik),ce(ic,pt
    )));
phibar(ip,ic) = sum(pt$csp(ic,pt), cc(ic,pt) * power(x_p(ip),ce(ic,
    pt)));

*---------------------------------------------------------------------
*     Function Approximation using Complete Chebyshev Polynomial
*---------------------------------------------------------------------

$eval cardcp ( (card(ic) * (card(ic)+1))/2 )

set   cp    Terms in the complete polynomial /1 * %cardcp%/;
alias(cp,cpp)

Parameters        cpe(s,cp)          Polynomial term number in the
    complete polynomial;

loop(ic,
      loop(cp$(ord(cp) ge (sum(jc$(ord(jc) le ord(ic)),jc.val) + 1
          - ord(ic)) and
              ord(cp) le (sum(jc$(ord(jc) le ord(ic)),jc.val))),
            cpe("cap",cp) = card(ic) + 1 - ic.val;
            cpe("phi",cp) = ord(cp)-(sum(jc$(ord(jc) le ord(ic)),jc
                .val)-ord(ic));
      );
);

*---------------------------------------------------------------------
*     Value function based on Chebyshev polynomial terms
*---------------------------------------------------------------------
*     Least-squares estimation
$macro PVL(kbar,phibar,ik,ip)  (sum(cp, A(cp) * \
                          sum(ic,kbar(ik,ic)$(ic.val eq cpe("cap",cp
                              ))) * \
                          sum(ic,phibar(ip,ic)$(ic.val eq cpe("phi",
                              cp)))))
```

```
*     Value function computation
$macro PV(KCS,phitcs,ik,ip,jp) (sum(cp, A(cp) * \
                          sum(ic,KCS(ik,ip,ic)$(ic.val eq cpe("cap",
                              cp))) * \
                          sum(ic,phitcs(jp,ic)$(ic.val eq cpe("phi",
                              cp)))))

*     Normalized value of K used in Chebyshev Polynomials
$macro KN(ik,ip) ((K(ik,ip)-(lo_k+up_k)/2)/((up_k-lo_k)/2))


*-----------------------------------------------------------------------
*     Transition Matrix
*-----------------------------------------------------------------------
Table tmatrix(ip,jp)
      1           2           3           4           5
1     0.9727      0.0273      0           0           0
2     0.0041      0.9806      0.0153      0           0
3     0           0.0082      0.9837      0.0082      0
4     0           0           0.0153      0.9806      0.0041
5     0           0           0           0.0273      0.9727;

Parameters
      phit        Grid point values of projected phi (productivity)
          ,
      phitn       Normalized grid point values of projected phi,
      phitcs      CS Polynomial terms used for value approximation;

phit(ip) = phi(ip);
phitn(ip) = ((phit(ip)-(lo_p+up_p)/2)/((up_p-lo_p)/2));

*-----------------------------------------------------------------------
*     Apply Chebyshev polynomial algorithm on productivity
*-----------------------------------------------------------------------
phitcs(ip,ic) = sum(pt$csp(ic,pt), cc(ic,pt) * power(phitn(ip),ce(
    ic,pt)));
```

B. Stochastic Neoclassical Growth Model: Hand-Coded MCP Formulation

```
$title DP-MCP: Stochastic Neoclassical Growth Model Using Chebyshev
     Collocation: MCP Version

$include data.gms

Variables
      OBJBELL           Bellman objective
      OBJLSQR           Least-squares objective
      C(ik,ip)          Consumption,
      K(ik,ip)          Subsequent period capital stock,
      U(ik,ip)          Nodal approximations of utility,
      A(cp)             Terms in the value function approximation,
      KCS(ik,ip,ic)     Chebyshev polynomial terms (ic) for capital
          ,
      P(ik,ip)          Shadow price of capital,
      P_I(ik,ip)        Shadow price of investment;

Equations
      utility           Present value benefit function,
      market            Market for current output,
      invest            Constraint for investment,
      objdef            Least squares objective,
      k_csdef           Chebyshev polynomial terms for capital,
      foca              First order condition for coefficient A,
      copt              First order condition for consumption,
```

```
      kopt              First order condition for capital,
      udef              Defines nodal utility;

*---------------------------------------------------------------------
*     NLP Equations
*---------------------------------------------------------------------
utility..          OBJBELL =e= sum((ik,ip), (1-beta) * log(C(ik,ip))
     +
                           beta * sum(jp, tmatrix(ip,jp) * PV(KCS,
                               phitcs,ik,ip,jp)));

market(ik,ip)..    C(ik,ip) + K(ik,ip) =e= phi(ip) * cap(ik)**alpha;

invest(ik,ip)..    cap(ik) * (1 - delta) - K(ik,ip) =l= 0;

objdef..           OBJLSQR =e= sum((ik,ip), sqr(PVL(kbar,phibar,ik,
    ip) - U(ik,ip)));

k_csdef(ik,ip,ic)..
             KCS(ik,ip,ic) =e= sum(pt$csp(ic,pt), cc(ic,pt) * power(
                 KN(ik,ip),ce(ic,pt)));
*---------------------------------------------------------------------
*     MCP Equations
*---------------------------------------------------------------------
foca(cpp).. sum((ik,ip), 2 *
                    (PVL(kbar,phibar,ik,ip) - U(ik,ip)) *
                           sum(ic,kbar(ik,ic)$(ic.val eq cpe("cap",
                               cpp))) *
                           sum(ic,phibar(ip,ic)$(ic.val eq cpe("phi",
                               cpp)))
             ) =e= 0;

copt(ik,ip)..      P(ik,ip) =g= (1-beta)/C(ik,ip);

kopt(ik,ip)..      P(ik,ip) + P_I(ik,ip) =g= beta/((up_k - lo_k)/2)
    *
             sum(jp, tmatrix(ip,jp) *
               sum(cp, A(cp) * sum(ic, phitcs(jp,ic)$(ic.val eq cpe
                    ("phi",cp))) *
                 sum(ic,
                   sum(pt,
                          cc(ic,pt)$(csp(ic,pt) and ce(ic,pt) ge 1) *
                            ce(ic,pt)$(csp(ic,pt) and ce(ic,pt) ge 1)
                                 *
                            power(KN(ik,ip),ce(ic,pt)$(ce(ic,pt) ge
                                1)-1)
                   )$(ic.val eq cpe("cap",cp))
                 )
               )
             );

udef(ik,ip)..      U(ik,ip) =e= (1-beta) * log(C(ik,ip)) +
                           beta * sum(jp, tmatrix(ip,jp) * PV(KCS,
                               phitcs,ik,ip,jp));

model bellman /utility,market,invest,k_csdef/;
model lsqr /objdef/;
model oneshot_mcp /foca.A, copt.C, kopt.K, market.P, invest.P_I,
    udef.U, k_csdef.KCS/;
*---------------------------------------------------------------------
*     Initialize and Solve
*---------------------------------------------------------------------
C.LO(ik,ip) = 1e-6;
K.LO(ik,ip) = 0;
A.L(cp) = 0;
C.L(ik,ip) = phi(ip) * cap(ik)**alpha;
*---------------------------------------------------------------------
*     Conventional Value Iteration
```

```
*---------------------------------------------------------------------
*     Parameters for value function iteration
Parameters
      dev          /1/,
      itlog        Iteration log,
      residuals    Sum of squared Bellman residuals;

*     Initial value:
U.FX(ik,ip) = (1-beta) * log(C.L(ik,ip));
file ktitle; ktitle.lw=0;

bellman.solvelink = 2;
loop(iter$round(dev,6),

      itlog(iter,cp) = A.L(cp);

      A.LO(cp) = -INF;  A.UP(cp) = +INF;

      solve lsqr using nlp minimzing OBJLSQR;

      dev = smax(cp, abs(A.L(cp)-itlog(iter,cp)));

      itlog(iter,"dev") = dev;

      A.FX(cp) = A.L(cp);

      solve bellman using nlp maximizing OBJBELL;
      abort$(bellman.solvestat<>1 and bellman.modelstat>2) "Bellman
           does not solve.";
$ondotl
      U.FX(ik,ip) = (1-beta) * log(C(ik,ip)) +
                    beta * sum(jp, tmatrix(ip,jp) * PV(KCS,phitcs,
                        ik,ip,jp));
      residuals(iter) = smax((ik,ip), abs(U(ik,ip) - PVL(kbar,
          phibar,ik,ip)));
$offdotl
      put ktitle;
      put_utility 'title' /'Iter: ',iter.tl,' Deviation = ',dev;

);
*---------------------------------------------------------------------
*     Pass off NLP solution to MCP solver as starting point
*---------------------------------------------------------------------
A.LO(cp) = -INF;
A.UP(cp) = +INF;
U.UP(ik,ip) = +inf;
U.LO(ik,ip) = -inf;
P.L(ik,ip) = market.m(ik,ip);
P_I.L(ik,ip) = -invest.m(ik,ip);
P_I.UP(ik,ip) = 0;
oneshot_mcp.iterlim = 1000;
solve oneshot_mcp using mcp;
```

C. Stochastic Neoclassical Growth Model: Automated EMP Formulation

```
$title DP-MCP: Stochastic Neoclassical Growth Model Using Chebyshev
     Collocation: EMP Version

$include data.gms

Variables
      OBJBELL(ik,ip)     Bellman objective
      OBJLSQR            Least-squares objective
      C(ik,ip)           Consumption,
```

```
      K(ik,ip)          Subsequent period capital stock,
      A(cp)             Terms in the value function approximation,
      KCS(ik,ip,ic)     Chebyshev polynomial terms (ic) for capital
          ,
      P(ik,ip)          Shadow price of capital,
      P_I(ik,ip)        Shadow price of investment;

Equations
      utility           Present value benefit function,
      market            Market for current output,
      invest            Constraint for investment,
      objdef            Least squares objective,
      k_csdef           Chebyshev polynomial terms for capital,
      foca              First order condition for coefficient A,
      copt              First order condition for consumption,
      kopt              First order condition for capital,
      udef              Defines nodal utility;

*---------------------------------------------------------------------
*     NLP Equations
*---------------------------------------------------------------------
utility(ik,ip)..  OBJBELL(ik,ip) =e= (1-beta) * log(C(ik,ip)) +
                                  beta * sum(jp, tmatrix(ip,jp) * PV
                                      (KCS,phitcs,ik,ip,jp);

market(ik,ip)..   C(ik,ip) + K(ik,ip) =e= phi(ip) * cap(ik)**alpha;

invest(ik,ip)..   cap(ik) * (1 - delta) - K(ik,ip) =l= 0;

objdef..          OBJLSQR =e= sum((ik,ip), sqr(PVL(kbar,phibar,ik,
    ip) - OBJBELL(ik,ip)));

k_csdef(ik,ip,ic)..
            KCS(ik,ip,ic) =e= sum(pt$csp(ic,pt), cc(ic,pt) * power(
                KN(ik,ip),ce(ic,pt)));

model oneshot_emp /objdef,utility,market,invest,k_csdef/;
*---------------------------------------------------------------------
*       Generate EMP Info File
*---------------------------------------------------------------------
$onecho > empfile.txt
equilibrium
min OBJLSQR         s.t. A(cp), objdef
max OBJBELL(ik,ip) s.t. C(ik,ip), K(ik,ip), KCS(ik,ip,ic), utility(
    ik,ip),
                market(ik,ip), invest(ik,ip), k_csdef(ik,ip,ic)
$offecho
$libinclude empmodel empfile.txt
*---------------------------------------------------------------------
*     Initialize and Solve
*---------------------------------------------------------------------
C.LO(ik,ip) = 1e-6;
K.LO(ik,ip) = 0;
A.L(cp) = 0;
C.L(ik,ip) = phi(ip) * cap(ik)**alpha;

*     5 iterations of conventional VFI to establish starting point
    for DP-MCP
$onecho > selkie.opt
major_iteration_limit 5
$offecho
option emp = selkie;
oneshot_emp.optfile = 1;
solve oneshot_emp using emp;
*     Solve with DP-MCP
option emp = jams;
solve oneshot_emp using emp;
```

References

Aguiar, A., McDougall, R., & Narayanan, B. (2012). *Global trade, assistance, and production: The gtap 8 data base*. West Lafayette, IN: Center for Global Trade Analysis, Purdue University.

Aruoba, S. B., & Fernández-Villaverde, J. (2014). *A comparison of programming languages in economics*. National Bureau of Economic Research: Technical report.

Aruoba, S. B., Fernandez-Villaverde, J., & Rubio-Ramirez, J. F. (2006). Comparing solution methods for dynamic equilibrium economies. *Journal of Economic Dynamics and Control*, *30*(12), 2477–2508.

Cai, Y. (2018). Computational methods in environmental and resource economics. Available at SSRN.

Cai, Y., & Judd, K. L. (2014). Advances in numerical dynamic programming and new applications. In *Handbook of computational economics* (Vol. 3, pp. 479–516). Elsevier.

Cai, Y., & Judd, K. L. (2015). Dynamic programming with Hermite approximation. *Mathematical Methods of Operations Research*, *81*(3), 245–267.

Cai, Y., Judd, K. L., & Lontzek, T. S. (2012). Dsice: A dynamic stochastic integrated model of climate and economy.

Cai, Y., Judd, K. L., Thain, G., & Wright, S. J. (2015). Solving dynamic programming problems on a computational grid. *Computational Economics*, *45*(2), 261–284.

Cai, Y., Judd, K., & Steinbuks, J. (2017). A nonlinear certainty equivalent approximation method for dynamic stochastic problems. *Quantitative Economics*, *8*(1), 117–147.

Dirkse, S. P., & Ferris, M. C. (1995). The PATH solver: A nommonotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software*, *5*(2), 123–156.

Dubé, J.-P., Fox, J. T., & Su, C.-L. (2012). Improving the numerical performance of static and dynamic aggregate discrete choice random coefficients demand estimation. *Econometrica*, *80*(5), 2231–2267.

Fernández-Villaverde, J., Gordon, G., Guerrón-Quintana, P., & Rubio-Ramirez, J. F. (2015). Nonlinear adventures at the zero lower bound. *Journal of Economic Dynamics and Control*, *57*, 182–204.

Ferris, M. C., & Munson, T. S. (2000). Complementarity problems in gams and the PATH solver. *Journal of Economic Dynamics and Control*, *24*(2), 165–188.

Ferris, M. C., Dirkse, S. P., Jagla, J.-H., & Meeraus, A. (2009). An extended mathematical programming framework. *Computers & Chemical Engineering*, *33*(12), 1973–1982.

Howitt, R., Msangi, S., Reynaud, A., & Knapp, K. (2002a). *Using polynomial approximations to solve stochastic dynamic programming problems: Or a 'betty crocker' approach to sdp*. Davis, CA: University of California.

Howitt, R. E., Reynaud, A., Msangi, S., Knapp, K. C., et al. (2002b). Calibrated stochastic dynamic models for resource management. In *The 2nd world congress of environmental and resource economists* (Vol. 2427).

Judd, K. L. (1998). *Numerical methods in economics*. MIT press.

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

Kim, Y., & Ferris, M. C. (2018). Selkie: A model transformation and distributed solver for equilibrium problems. Technical report, University of Wisconsin-Madison.

Kim, Y., & Ferris, M. C. (2019). Solving equilibrium problems using extended mathematical programming. *Mathematical programming computation*. https://doi.org/10.1007/s12532-019-00156-4.

Krueger, D., & Kubler, F. (2004). Computing equilibrium in olg models with stochastic production. *Journal of Economic Dynamics and Control*, *28*(7), 1411–1436.

Lau, M. I., Pahlke, A., & Rutherford, T. F. (2002). Approximating infinite-horizon models in a complementarity format: A primer in dynamic general equilibrium analysis. *Journal of Economic Dynamics and Control*, *26*(4), 577–609.

Lemoine, D., & Rudik, I. (2017). Managing climate change under uncertainty: Recursive integrated assessment at an inflection point. *Annual Review of Resource Economics*, *9*, 117–142.

Lemoine, D., & Traeger, C. (2014). Watch your step: Optimal policy in a tipping climate. *American Economic Journal: Economic Policy*, *6*(1), 137–66.

Lemoine, D., & Traeger, C. P. (2016). Economics of tipping the climate dominoes. *Nature Climate Change*, *6*(5), 514.

Maliar, L., & Maliar, S. (2014). Numerical methods for large-scale dynamic economic models. In *Handbook of computational economics* (Vol. 3, pp. 325–477). Elsevier.

Maliar, L., & Maliar, S. (2015). Merging simulation and projection approaches to solve high-dimensional problems with an application to a new Keynesian model. *Quantitative Economics*, *6*(1), 1–47.
Manuelli, R. E., & Sargent, T. J. (2009). *Exercises in dynamic macroeconomic theory*. Cambridge, MA: Harvard University Press.
Mathiesen, L. (1985). Computation of economic equilibria by a sequence of linear complementarity problems. In *Economic equilibrium: Model formulation and solution* (pp. 144–162). Springer.
Miranda, M. J., Fackler, P. L. (2004). *Applied computational economics and finance*. MIT press.
Powell, W. B. (2011). *Approximate dynamic programming: Solving the curses of dimensionality* (Vol. 842). Hoboken, NJ: Wiley.
Rasmussen, T. N., & Rutherford, T. F. (2004). Modeling overlapping generations in a complementarity format. *Journal of Economic Dynamics and Control*, *28*(7), 1383–1409.
Rudik, I. (2016). Optimal climate policy when damages are unknown. Available at SSRN 2516632.
Rust, J. (1996). Numerical dynamic programming in economics. *Handbook of Computational Economics*, *1*, 619–729.
Rutherford, T. F. (1995). Extension of gams for complementarity problems arising in applied economic analysis. *Journal of Economic Dynamics and Control*, *19*(8), 1299–1324.
Sargent, T., & Stachurski, J. (2015). *Quantitative economics with python*. Technical report, Lecture Notes: Technical report.
Smolyak, S. (1963). Quadrature and interpolation formulas for tensor products of certain classes of functions. *Soviet Mathematics Doklady*, *4*, 240–243.
Stokey, N. L. (1989). Robert E with Edward C. Prescott Lucas Jr. Recursive methods in economic dynamics.
Su, C.-L., & Judd, K. L. (2012). Constrained optimization approaches to estimation of structural models. *Econometrica*, *80*(5), 2213–2230.
Tauchen, G. (1986). Finite state Markov-chain approximations to univariate and vector autoregressions. *Economics Letters*, *20*(2), 177–181.
Traeger, C. P. (2014a). A 4-stated dice: Quantitatively addressing uncertainty effects in climate change. *Environmental and Resource Economics*, *59*(1), 1–37.
Traeger, C. P. (2014b). Why uncertainty matters: Discounting under intertemporal risk aversion and ambiguity. *Economic Theory*, *56*(3), 627–664.
Wright, S., & Nocedal, J. (1999). Numerical optimization. *Springer Science*, *35*, 67–68.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.