*Macroeconomic Dynamics*, 2016, Page 1 of 26. Printed in the United States of America.
doi:10.1017/S1365100515000528

# A NONLINEAR PROGRAMMING METHOD FOR DYNAMIC PROGRAMMING

**YONGYANG CAI**
*Hoover Institution*
*and*
*Becker Friedman Institute, University of Chicago*

**KENNETH L. JUDD**
*Hoover Institution*

**THOMAS S. LONTZEK**
*University of Zurich*

**VALENTINA MICHELANGELI**
*Bank of Italy*

**CHE-LIN SU**
*The University of Chicago Booth School of Business*

A nonlinear programming formulation is introduced to solve infinite-horizon dynamic programming problems. This extends the linear approach to dynamic programming by using ideas from approximation theory to approximate value functions. Our numerical results show that this nonlinear programming is efficient and accurate, and avoids inefficient discretization.

**Keywords:** Dynamic Programming, Optimal Control, Nonlinear Programming, Infinite-Horizon Decision Making Problem

## 1. INTRODUCTION

Dynamic programming (DP) is the essential tool for solving dynamic and stochastic control problems in economics, but the nonlinearities of dynamic economic problems make them numerically challenging. This has led to the development of a variety of methods for solving dynamic programming problems; see the surveys

Cai, Judd, Lontzek, and Michelangeli note with great sadness the passing of Che-Lin Su this past July. We thank him for his contributions. We are grateful to the editors and anonymous reviewers for their insightful comments and suggestions. We particularly thank Philipp Renner for his many helpful comments. Cai and Judd gratefully acknowledge NSF support (SES-0951576). The views expressed herein are those of the authors and do not necessarily reflect the views of the Bank of Italy. Address correspondence to: Yongyang Cai, Hoover Institution, Stanford, CA 94305, USA; e-mail: yycai@stanford.edu.

in Rust (1996; 2008) and Cai and Judd (2014). One approach to solving problems with continuous states has been to discretize the states and controls, and then apply methods for discrete problems. For example, Trick and Zin (1997) applied the linear programming (LP) approach to a discretized economic growth model. The LP approach allowed them to formulate the infinite-horizon DP problem as the solution to a finite-dimensional LP problem. The LP approach has been studied extensively in the operations research literature; see De Farias and Van Roy (2003) for more recent developments. However, it has limited value for problems with continuous actions and/or states, because sufficiently fine discretization will often create intractably large LP problems. For example, if each dimension of a $d$-dimensional state space is discretized by $m$ nodes, then the number of points in the discrete-state problem is $m^d$. The necessity to discretize the action space substantially increases the size of the LP problem, where the number of constraints equals the product of the number of discrete states and discrete actions. These issues are particularly important when we want to examine how utility changes in response to economic policy changes. In these cases, it is necessary to obtain accurate approximations to the decision rules as well as the value function, requirements that easily lead to intractably large LP problems.

This paper presents a nonlinear programming (NLP) method, called DPNLP, to solve infinite-horizon DP problems. DPNLP takes the underlying idea behind the LP approach and adapts it to problems with continuous states and actions. The DPNLP approach uses a variety of tools from numerical analysis, such as shape-preserving approximation methods, numerical quadrature, and solvers for general nonlinear constrained optimization. The use of efficient methods from approximation theory is particularly important because they allow us to mitigate, if not avoid, the curse of dimensionality for problems with smooth solutions.

Our numerical examples illustrates the efficiency of the DPNLP approach. For example, we solve optimal growth problems with two continuous state variables and six continuous control variables. DPNLP solves them in minutes with up to five-digit accuracy, a combination of speed and accuracy that is not possible with the LP approach.

The DPNLP algorithm is an efficient and easily implemented approach to solving infinite-horizon DP problems such as the optimal growth problems given in Judd (1998), Maliar and Maliar (2004), and Den Haan et al. (2011). In fact, the efficiency of the DPNLP algorithm allows it to be a critical component of empirical estimation methods. For example, Michelangeli (2009) used DPNLP inside an MPEC approach [see Su and Judd (2012)] to estimate a model of the demand for reverse mortgages.

The paper is organized as follows. Section 2 describes the kind of dynamic problems commonly used in economics and the subject of this paper. Section 3 defines the DPNLP algorithm for solving infinite-horizon DP problems. Section 4–6 apply DPNLP to optimal accumulation problems similar to many economics problems. Section 7 concludes.

## 2. DYNAMIC PROGRAMMING

An infinite-horizon stochastic optimal decision-making problem has the following general form:

$$V(x_0, \theta_0) = \max_{a_t \in \mathcal{D}(x_t, \theta_t)} \mathbf{E} \left\{ \sum_{t=0}^{\infty} \beta^t u(x_t, a_t) \right\}, \tag{1}$$

$$\text{s.t.} \quad x_{t+1} = g(x_t, \theta_t, a_t),$$

$$\theta_{t+1} = h(\theta_t, \epsilon_t),$$

where $x_t$ is an endogenous continuous-state vector process with initial state $x_0$, $\theta_t$ is an exogenous state vector process with initial state $\theta_0$, $\epsilon_t$ is a serially uncorrelated random vector process, $g$ is a continuous function representing the change in the state $x_t$ as a function of the state and the action $a_t$, $h$ represents the transition process for $\theta_t$, $\mathcal{D}(x_t, \theta_t)$ is a feasible set of $a_t$ dependent on $(x_t, \theta_t)$, $\beta$ is the discount factor with $0 < \beta < 1$, $u$ is a concave utility function, and $\mathbf{E}\{\cdot\}$ is the expectation operator. Although this description does not apply to many applications of dynamic programming, it does apply to most models in dynamic economics. Examples include economic growth, portfolio decisions, and investment decisions by firms.

The DP model for the general infinite-horizon problem is the following Bellman equation [Bellman (1957)]:

$$V(x, \theta) = \max_{a \in \mathcal{D}(x, \theta)} u(x, a) + \beta \mathbf{E} \left\{ V(x^+, \theta^+) \mid x, \theta, a \right\}, \tag{2}$$

$$\text{s.t.} \quad x^+ = g(x, \theta, a),$$

$$\theta^+ = h(\theta, \epsilon),$$

where $(x^+, \theta^+)$ is the next-stage state conditional on the current-stage state $(x, \theta)$ and the action $a$, $\epsilon$ is a random variable, and $V(x, \theta)$ is the value function. The case of no uncertainty is a special case of (1) with only one value of $\theta$, implying

$$V(x_0) = \max_{a_t \in \mathcal{D}(x_t)} \sum_{t=0}^{\infty} \beta^t u(x_t, a_t), \tag{3}$$

$$\text{s.t.} \quad x_{t+1} = g(x_t, a_t),$$

and the Bellman equation

$$V(x) = \Gamma(V)(x) := \max_{a \in \mathcal{D}(x)} \quad u(x, a) + \beta V(x^+), \tag{4}$$

$$\text{s.t.} \quad x^+ = g(x, a).$$

We call $\Gamma$ the Bellman operator.

The DP problem (4) is defined in terms of the optimal value function. When the state variable is continuous, the value function must be approximated in some finite-dimensional space; thus we have to use some approximation for the value functions, because computers cannot model the entire space of continuous functions.

An approximation scheme consists of two parts: basis functions and approximation nodes. Approximation nodes can be chosen as uniformly spaced nodes, Chebyshev nodes, or some other specified nodes. From the viewpoint of basis functions, approximation methods can be classified as either spectral methods or finite-element methods. A spectral method uses globally nonzero basis functions $\{\phi_j(x)\}$ and coefficients $\mathbf{b} = \{b_j\}$ such that $\hat{V}(x; \mathbf{b}) = \sum_{j=0}^{n} b_j \phi_j(x)$ is a degree-$n$ approximation. Examples of spectral methods include ordinary polynomial approximation, Chebyshev polynomial approximation, and shape-preserving Chebyshev polynomial approximation [Cai and Judd (2013)]. In contrast, a finite-element method uses local basis functions $\{\phi_j(x)\}$ that are nonzero over subdomains of the approximation domain. Examples of finite-element methods include piecewise linear interpolation, Schumaker interpolation, shape-preserving rational function spline Hermite interpolation [Cai and Judd (2012)], cubic splines, and B-splines. See Judd (1998), Cai (2010), and Cai and Judd (2010) for more details. Appendices A and B describe (multidimensional) Chebyshev polynomials.

## 3. NONLINEAR PROGRAMMING METHOD FOR SOLVING BELLMAN EQUATIONS

There are many approaches to solving Bellman equations, such as value function iteration and policy iteration methods or LP approaches. This section describes the general nonlinear programming method (DPNLP) for solving the Bellman equation (4) or (2) for concave dynamic programming problems. We describe the DPNLP in three stages. First, we present a formulation of (1), which is an infinite-dimensional optimization problem. Second, we describe the logic of our discretization of the infinite set of constraints in the infinite formulation. However, the direct implementation of this basic concept is difficult because of the undetermined fineness of the necessary discretization. Third, we describe the DPNLP algorithm, which will build up the value function approximation using successively finer discretization, but will terminate in finite time and produce an approximate solution to (1).

### 3.1. Review of Linear Programming Approaches

When the DP problem (4) has a discrete state $x \in \{x_1, \ldots, x_m\}$ and a discrete control $a$ in a finite set of points $\mathbf{A}_i$ for each state point $x_i$, it can be solved by a LP model [see Trick and Zin (1997)]:

$$\begin{aligned} \min_{v} \quad & \sum_{i=1}^{m} v_i, \\ \text{s.t.} \quad & v_i \geq u(x_i, a) + \beta v_{i^+(a)}, \ \forall a \in \mathbf{A}_i, \ i = 1, \ldots, m, \end{aligned}$$

where $i^+(a)$ is the index of the next-period state $x_{i^+(a)}$ with current state $x_i$ and action $a$. Note that the constraints must hold for any possible action $a \in \mathbf{A}_i$, which implies that

$$v_i \geq \max_{a \in \mathbf{A}_i} u(x_i, a) + \beta v_{i^+(a)},$$

for $i = 1, \ldots, m$.

When the DP problem (4) has a continuous state $x$ and a discrete control $a$ in a finite set of points $\mathbf{A}_i$ for each state point $x_i$, Trick and Zin (1997) choose a set of state points $\{x_1, \ldots, x_m\}$, and then approximate the value function by a spline $\hat{V}(x_i; \mathbf{b})$, linear in the unknown coefficients $\mathbf{b}$. They solve the problem using the following LP model:

$$\begin{aligned} \min_{\mathbf{b}} \quad & \sum_{i=1}^{m} \hat{V}(x_i; \mathbf{b}), \\ \text{s.t.} \quad & \hat{V}(x_i; \mathbf{b}) \geq u(x_i, a) + \beta \hat{V}(g(x_i, a); \mathbf{b}), \quad \forall a \in \mathbf{A}_i,\ i = 1, \ldots, m. \end{aligned} \tag{5}$$

De Farias and Van Roy (2003) extend the preceding LP method by randomly choosing $x_i$, by changing the objective to a weighted sum [i.e., $\sum_{i=1}^{m} w_i \hat{V}(x_i; \mathbf{b})$], and by generalizing the form of $\hat{V}(x; \mathbf{b})$ to be a linear combination of preselected basis functions with the unknown coefficients $\mathbf{b}$.

## 3.2. Infinite-Horizon Dynamic Programming as an Infinite Programming Problem

In this paper, we generalize the concept of the LP method to problems with continuous states and continuous controls. We start with the following model:

$$\begin{aligned} \max_{a(x) \in \mathcal{D}(x), V(x)} \quad & \int_{\mathbf{X}} V(x) dx, \\ \text{s.t.} \quad & V(x) \leq u(x, a(x)) + \beta V(g(x, a(x))),\ \forall x \in \mathbf{X}, \end{aligned} \tag{6}$$

where $V(x)$ and $a(x)$ are two functional variables in the set of measurable functions on $\mathbf{X}$, and then we prove the following theorem.

THEOREM 1. *Suppose that*

(i) $\mathbf{X} \subset \mathbf{R}^d$ *is an open and bounded domain;*
(ii) $u(x, a)$ *and* $g(x, a)$ *are continuous in both* $x \in \mathbf{X}$ *and* $a \in \mathcal{D}(x)$*;*
(iii) *there exists an optimal solution of (6),* $\widehat{V}(x)$ *and* $\widehat{a}(x)$*, which are continuous and bounded on* $\mathbf{X}$*;*
(iv) *there exists an optimal measurable solution of (4),* $V^*(x)$ *and* $a^*(x)$*, where* $V^*(x)$ *is also continuous and bounded on* $\mathbf{X}$.

*Thus, $V^*(x) = \widehat{V}(x)$ for all $x \in \mathbf{X}$. Moreover, $\widehat{a}(x)$ is also an optimal policy function of (4).*

Proof. We first show by contradiction that the constraints of (6) bind at the optimal solution of (6), $\widehat{V}(x)$ and $\widehat{a}(x)$, for any $x \in \mathbf{X}$. Suppose that there exists $x^* \in \mathbf{X}$ such that

$$\widehat{V}(x^*) < u(x^*, \widehat{a}(x^*)) + \beta\widehat{V}(g(x^*, \widehat{a}(x^*))).$$

Let $V(x) \equiv u(x, \widehat{a}(x)) + \beta\widehat{V}(g(x, \widehat{a}(x)))$. We have $V(x^*) > \widehat{V}(x^*)$ and $V(x) \geq \widehat{V}(x)$ for all $x \in \mathbf{X}$, because $\widehat{V}$ and $\widehat{a}$ satisfy the constraints of (6). Thus,

$$\begin{aligned} V(x) &\equiv u(x, \widehat{a}(x)) + \beta\widehat{V}(g(x, \widehat{a}(x))) \\ &\leq u(x, \widehat{a}(x)) + \beta V(g(x, \widehat{a}(x))) \end{aligned}$$

for all $x \in \mathbf{X}$, which implies that $V(x)$ and $\widehat{a}(x)$ are feasible for (6). From continuity of $u$, $g$, $\widehat{V}$, and $\widehat{a}$, we know that $V$ is also continuous, and thus the set $\{x \in \mathbf{X} : V(x) > \widehat{V}(x)\}$ has a nonzero measurement, as $\mathbf{X}$ is an open set. B both $\mathbf{X}$ and $\widehat{V}$ are bounded, we have a finite $\int_{\mathbf{X}} \widehat{V}(x)dx$; then

$$\int_{\mathbf{X}} V(x)dx > \int_{\mathbf{X}} \widehat{V}(x)dx.$$

This contradicts the assumption that $\widehat{V}(x)$ and $\widehat{a}(x)$ are the optimal solution of (6). Therefore, we have the binding constraints at the optimal solution, i.e.,

$$\widehat{V}(x) = u(x, \widehat{a}(x)) + \beta\widehat{V}(g(x, \widehat{a}(x))), \tag{7}$$

for any $x \in \mathbf{X}$. This also implies that

$$\widehat{V}(x) = \sum_{t=0}^{\infty} \beta^t u(\widehat{x}_t, \widehat{a}(\widehat{x}_t)), \tag{8}$$

with $\widehat{x}_{t+1} = g(\widehat{x}_t, \widehat{a}(\widehat{x}_t))$ and $\widehat{x}_0 = x$.

Because $V^*(x)$ and $a^*(x)$ are optimal for (4), we have

$$V^*(x) = u(x, a^*(x)) + \beta V^*(g(x, a^*(x))) \tag{9}$$

for any $x \in \mathbf{X}$, which implies that

$$V^*(x) = \sum_{t=0}^{\infty} \beta^t u(x_t^*, a^*(x_t^*)), \tag{10}$$

with $x_{t+1}^* = g(x_t^*, a^*(x_t^*))$ and $x_0^* = x$. Because $V^*$ is the fixed point of the Bellman operator $\Gamma$, we know that $V^*$ and $a^*$ are also the solution of the problem (3), so from (8) and (10) we have $V^*(x) \geq \widehat{V}(x)$ for all $x \in \mathbf{X}$.

Now we show that $V^*(x) \leq \widehat{V}(x)$ for all $x \in \mathbf{X}$, by contradiction. Suppose that there exists $\widetilde{x} \in \mathbf{X}$ such that $V^*(\widetilde{x}) > \widehat{V}(\widetilde{x})$. Let

$$\Omega := \left\{x \in \mathbf{X} : \ V^*(x) > \widehat{V}(x)\right\}.$$

We know that $\Omega$ is nonempty and has a nonzero measurement from the openness of $\mathbf{X}$ and the assumption of continuity of $V^*$ and $\widehat{V}$. Let

$$\widetilde{a}(x) = \begin{cases} a^*(x), & \text{if } x \in \Omega, \\ \widehat{a}(x), & \text{otherwise,} \end{cases}$$

and

$$\widetilde{V}(x) = \begin{cases} V^*(x), & \text{if } x \in \Omega, \\ \widehat{V}(x), & \text{otherwise.} \end{cases}$$

From (7) and (9), we know that $\widetilde{V}$ and $\widetilde{a}$ are feasible in (6). Moreover,

$$\int_{\mathbf{X}} \widetilde{V}(x)dx > \int_{\mathbf{X}} \widehat{V}(x)dx,$$

as $\Omega$ has a nonzero measurement. This contradicts the assumption that $\widehat{V}(x)$ and $\widehat{a}(x)$ are the optimal solution of (6).

Therefore, we have shown that $V^*(x) = \widehat{V}(x)$ for all $x \in \mathbf{X}$. Thus, from (8) and (10), we know that $\widehat{a}(x)$ is also an optimal policy function for the problem (3), and thus $\widehat{a}(x)$ is also an optimal policy function for (4). ■

In the preceding theorem, we assume that $\mathbf{X} \subset \mathbf{R}^d$ is an open and bounded domain, but it is easy to see that the theorem can also hold on its corresponding closed domain $\mathbf{X} \cup \partial\mathbf{X}$, where $\partial\mathbf{X}$ is the boundary of $\mathbf{X}$. Moreover, under the assumptions of the theorem, if $a^*(x)$ is the unique optimal policy function of the problem (4), then we have $\widehat{a}(x) = a^*(x)$ for all $x \in \mathbf{X}$.

The problem defined in (6) is an infinite-dimensional problem with an infinite number of constraints. Any numerical method for solving (6) must use a parameterized approximation of $V(x)$. When we replace $V(x)$ in (6) with the finitely parameterized $\hat{V}(x; \mathbf{b})$, we arrive at

$$\max_{\mathbf{b}, a(x)} \int_{\mathbf{X}} \hat{V}(x; \mathbf{b})dx, \tag{11}$$
$$\text{s.t.} \quad \hat{V}(x; \mathbf{b}) \leq u(x, a(x)) + \beta\hat{V}(g(x, a(x)); \mathbf{b}), \ \forall x \in \mathbf{X}.$$

The only difference between (11) and (6) is the smaller choice set for value functions in (11). The solution to (11) will be our approximation to the solution of (6), and will be the solution to (6) if the set of functions defined by the functional form $\hat{V}(x; \mathbf{b})$ includes the solution of (6).

The DPNLP method focuses on solving the problem (11). A standard approach for solving it is to discretize the constraints; see Polak (1997). In the remainder of this paper we will describe an efficient strategy that allows us to solve concave dynamic programming problems.

## 3.3. The Basic Concepts behind DPNLP

The problem in (11) uses a finite-dimensional representation of $V(x)$. We next need to replace the infinite set of constraints in (11) with a finite subset. The key discretization is to choose a finite number of points, $x_i, i = 1, \dots m$, in the domain $\mathbf{X}$. We call them the approximation nodes. In (11), for each $x \in \mathbf{X}$, all action choices are examined, creating a new constraint for each $a \in \mathcal{D}(x)$. In DPNLP, we will choose the best feasible action at each approximation node.

This leads to the following finite-dimensional nonlinear optimization problem:

$$\begin{aligned}
\max_{a_i \in \mathcal{D}(x_i), x_i^+, v_i, \mathbf{b}} \quad & \sum_{i=1}^{m} v_i, \qquad\qquad (\mathbf{12}) \\
\text{s.t.} \quad & v_i \leq u(x_i, a_i) + \beta \hat{V}(x_i^+; \mathbf{b}), \quad i = 1, \dots, m, \\
& x_i^+ = g(x_i, a_i), \quad i = 1, \dots, m, \\
& v_i = \hat{V}(x_i; \mathbf{b}), \quad i = 1, \dots, m,
\end{aligned}$$

where $x_i, i = 1, \dots m$, are the approximation nodes in the domain $\mathbf{X}$. For each approximation node $x_i$, we need to specify a continuous action choice, $a_i$, a successor state, $x_i^+$, and a value, $v_i$. The coefficients of the value function approximation, $\mathbf{b}$, are also chosen in the optimization problem.

The idea in (12) is to solve the DP problem at each approximation node and construct a value function that is a valid global representation of the resulting data. Unfortunately, the formulation in (12) may fail, as the interpolation at the approximation nodes might not produce acceptable value function approximations $\hat{V}$. For example, if the dynamic programming problem is concave, the solution for the coefficients $\mathbf{b}$ in (12) may imply a $\hat{V}(x; \mathbf{b})$ that is not concave in $x \in \mathbf{X}$. Shape preservation is a critical feature of stable DP solution methods; see Cai and Judd (2010, 2012, 2013) for a discussion of the problems when shape is not preserved and examples of shape preservation in dynamic programming. In this paper, we are concerned with solving problems with a concave value function. Therefore, we next add the requirement that the $\mathbf{b}$ coefficients imply a concave $\hat{V}(x; \mathbf{b})$. This produces an infinite number of constraints, but again we replace it with a finite subset of constraints. Specifically, we choose a set of shape nodes $\{y_{i'} : i' = 1, \dots, m'\} \subset \mathbf{X}$ and impose concavity and monotonicity at those points. This leads to the basic DPNLP problem:

$$\max_{a_i \in \mathcal{D}(x_i), x_i^+, v_i, \mathbf{b}} \quad \sum_{i=1}^{m} v_i, \tag{13}$$

$$\begin{aligned} \text{s.t.} \quad & v_i \leq u(x_i, a_i) + \beta \hat{V}(x_i^+; \mathbf{b}), \quad i = 1, \ldots, m, \qquad (\mathbf{14}) \\ & x_i^+ = g(x_i, a_i), \quad i = 1, \ldots, m, \\ & v_i = \hat{V}(x_i; \mathbf{b}), \quad i = 1, \ldots, m, \\ & \hat{V}'(y_{i'}; \mathbf{b}) \geq 0, \quad i' = 1, \ldots, m', \\ & \hat{V}''(y_{i'}; \mathbf{b}) \leq 0, \quad i' = 1, \ldots, m'. \end{aligned}$$

All of this assumes that there are polynomials with the desired shape properties for $\hat{V}$ and sets of approximation and shape nodes that will impose the interpolation and shape requirements. Approximation theory provides us with this justification. If we use a polynomial approximation, then the Stone–Weierstrass theorem tells us that there is a finite-degree polynomial that is close to $V(x)$. Moreover, Bernstein polynomials can be constructed to create shape-preserving approximations for a sufficiently dense collection of approximation nodes [see Bojanic and Cheng (1989); Goodman (1989); Carnicer and Pena (1993); Farouki (2012); etc.). Therefore, there do exist settings for $m$ and $m'$ and values for the coefficients $\mathbf{b}$ such that (13) will succeed at finding a good approximation to the true value function. We use only the existence results, not the inefficient Bernstein polynomial construction.

There are many considerations in choosing the approximation and shape nodes so that (13) will produce a good approximation to the true value function. One conservative approach would be to choose $m$ and $m'$ sufficiently large and $\mathbf{b}$ sufficiently flexible so that shape is surely satisfied, $\hat{V}$ interpolates the $v_i$ data at each $x_i$, and the optimization inequality, $v_i \leq u(x_i, a_i) + \beta \hat{V}(x_i^+; \mathbf{b})$, holds at each $x_i$. To ensure shape, $m'$ will generally need to exceed $m$. If no shape constraint is binding (as will be the case in our applications to strictly concave increasing value functions) the optimization inequality (14) will be binding from the proof of Theorem 1. This implies that, for any $a_i$, a solver will jointly adjust $v_i$ and $\mathbf{b}$ to maintain the interpolation constraint (which is possible if no shape constraint is binding) and increase $v_i$ until $v_i = u(x_i, a_i) + \beta \hat{V}(x_i^+; \mathbf{b})$. The solver will also change $a_i$ and $x_i^+$ to allow for further increase in $v_i$ if possible. Therefore, the $a_i$ choice will maximize $u(x_i, a_i) + \beta \hat{V}(x_i^+; \mathbf{b})$.

Although this conservative approach is a reliable strategy, it is not necessary. The objective of DPNLP is to solve infinite-horizon problems quickly and accurately. (13) may produce an excellent answer even if we delete the shape constraints. In fact, we aim to increase the flexibility of $\hat{V}$ until no shape constraint is binding, raising the question of why we include them. Without the shape constraints, there may be $\hat{V}$ functions that solve (13), even yielding greater values for the objective, but that are not close to the true value function. This problem is discussed in Cai and Judd (2013). The role of the shape constraints is to define an optimization problem with convex constraints that rule out values of **b** implying $\hat{V}$ functions that cannot be close to the true value function. Therefore, in practice, we could take an aggressive approach that initially imposed few shape constraints, but gradually increased the flexibility of the $\hat{V}$ specification and the number of shape constraints until the result passed an error bound test. Subsection 3.4 will discuss that error bound test and Subsection 3.5 will present the full DPNLP algorithm.

We immediately see a major advantage of DPNLP over the LP approach [Trick and Zin (1997); De Farias and Van Roy (2003)]. The LP approach is limited by a curse of dimensionality arising from the need to keep an inequality for every possible action choice for each state point. In DPNLP, the number of action choices grows only at the rate of the number of $x_i$ points, which does not grow exponentially if the choice for $\hat{V}(x;\mathbf{b})$ is an efficient method of approximating $V(x)$. Furthermore, any method that approximates a continuous-state problem with a discrete-state problem [as in Trick and Zin (1997); De Farias and Van Roy (2003)] will have to use a large number of discrete states to achieve high accuracy. The DPNLP approach allows us to use efficient approximation methods for continuous-state problems.

Solving the stochastic Bellman equation (2) where $\theta \in \Theta = \{\vartheta_j : j = 1,\ldots,J\}$ is a straightforward extension. In this case, the basic DPNLP model becomes

$$\begin{aligned}
\min_{a_{i,j}\in\mathcal{D}(x_i,\vartheta_j),x_i^+,v_i,\mathbf{b}} \quad & \sum_{j=1}^{J}\sum_{i=1}^{m} v_{i,j}, \qquad (15)\\
\text{s.t.} \quad & v_{i,j} \le u(x_i,a_{i,j}) + \beta\sum_{j'=1}^{J} P_{j,j'}\hat{V}(x_{i,j}^+,\vartheta_{j'};\mathbf{b}),\\
& x_{i,j}^+ = g(x_i,\vartheta_j,a_{i,j}),\\
& v_{i,j} = \hat{V}(x_i,\vartheta_j;\mathbf{b}),\\
& \hat{V}'(y_{i'},\vartheta_j;\mathbf{b}) \ge 0, \quad i'=1,\ldots,m',\\
& \hat{V}''(y_{i'},\vartheta_j;\mathbf{b}) \le 0, \quad i'=1,\ldots,m',\\
& i=1,\ldots,m, \;\; j=1,\ldots,J,
\end{aligned}$$

where $P_{j,j'}$ is the conditional probability of $\theta^+ = \vartheta_{j'}$ given $\theta = \vartheta_j$, i.e., $P_{j,j'} = \Pr\left(\theta^+ = \vartheta_{j'} \mid \theta = \vartheta_j\right)$, for any $j, j' = 1,\ldots,J$. For a case with continuous $\theta$, the expectation over $\theta^+$ is replaced by a numerical quadrature formula; then we have a model of the form (15).

## 3.4. Error Estimation

In any dynamic programming solution algorithm for infinite-horizon problems, it is important to estimate the error of a value function approximation $\hat{V}(x; \mathbf{b})$. According to the Bellman equation (4), we know that the true value function is the fixed point of the Bellman operator $\Gamma$. The contraction mapping property of the Bellman operator [Blackwell (1965)] and the triangle inequality imply the following upper bound on a value function approximation:[1]

$$\left\| \hat{V} - V \right\|_\infty \leq \frac{\left\| \hat{V} - \Gamma\left(\hat{V}\right) \right\|_\infty}{1-\beta}. \tag{16}$$

We will use this property to estimate the approximation error of a possible solution, $\hat{V}(x; \mathbf{b})$.

The idea is to evaluate the error at a random sample of points in the state space, and take the maximum to approximate the upper bound in (16). More specifically, we draw a random set of points in the domain $\mathbf{X}$, denoted by $z_1, \ldots, z_N$, and solve the optimization problem

$$\begin{aligned} \widetilde{v}_j = \Gamma\left(\hat{V}\right)(z_j) = \max_{a_j \in \mathcal{D}(z_j), z_j^+} \quad & u(z_j, a_j) + \beta \hat{V}(z_j^+; \mathbf{b}), \\ \text{s.t.} \quad & z_j^+ = g(z_j, a_j), \end{aligned} \tag{17}$$

for each point $z_j$, $j = 1, \ldots, N$. Letting $\widetilde{v} = [\widetilde{v}_1, \ldots, \widetilde{v}_N]$, $z = [z_1, \ldots, z_N]$, and $\hat{V}(z; \mathbf{b}) = \left[\hat{V}(z_1; \mathbf{b}), \ldots, \hat{V}(z_N; \mathbf{b})\right]$, we could use

$$\frac{\left\| \widetilde{v} - \hat{V}(z; \mathbf{b}) \right\|_\infty}{1-\beta} \tag{18}$$

to approximate the error upper bound on the right-hand side of (16).

The inequality expressed in (16) is always valid, but the actual magnitude of this upper bound is sensitive to the choice of units. To make a reliable judgment concerning the quality of a value function approximation, we need to rescale the error estimation (18) in some way that will indicate whether the error is important for the economic analysis. The first key step is to get rid of units. The unit of $\hat{V}$ is "utils," the magnitude of which has no economic value. In fact, any affine transformation of the utility function will lead to a different value function. $\hat{V}'(x; \mathbf{b})$ has the unit "utils per $x$," so $\overline{x} \cdot \hat{V}'(\overline{x}; \mathbf{b})$ has the same unit as $\hat{V}$, where $\overline{x}$ is a reference state point in the state space $\mathbf{X}$. We therefore get rid of the utility units by division with respect to $\overline{x} \cdot \hat{V}'(\overline{x}; \mathbf{b})$, and this gives us a unit-free expression describing the error with the scaled error norm

$$\left\| \frac{\widetilde{v} - \hat{V}(z; \mathbf{b})}{\overline{x} \cdot \hat{V}'(\overline{x}; \mathbf{b})(1-\beta)} \right\|_\infty, \tag{19}$$

where $\overline{x}$ should be representative of the states that are relevant to the problem, such as the steady state. Expression (19) is a unit-free measure of the percentage change in $x$ that is (approximately) $x$-equivalent to the value function error. For example, if the state variable is wealth, then our unit-free error norm expresses the difference between $\widetilde{v}$ and $\hat{V}$ at $z$ in terms of an equivalent relative change in wealth. The error measure is, of course, not an exact measure of a wealth equivalent change, but it is fine for our purposes, because we are concerned with the order of magnitude of the errors, not with their precise value.

For a stochastic problem (2), we use the normed error measure

$$\left\| \frac{\widetilde{v} - \hat{V}(z, \theta; \mathbf{b})}{\overline{x} \cdot V'(\overline{x}, \overline{\theta}; \mathbf{b})(1-\beta)} \right\|_{\infty}, \qquad \textbf{(20)}$$

where the $i$th elements of $\widetilde{v}$ and $\hat{V}(z, \theta; \mathbf{b})$ correspond to the $i$th sample point $(z_i, \theta_i)$, $\overline{\theta}$ is the median of $\theta$, and $\overline{x}$ is a reference point, such as the steady state of the related deterministic problem. In a multidimensional problem, $\overline{x} \cdot \hat{V}'$ is the inner product of $\overline{x}$ and the gradient vector of $\hat{V}$ at $\overline{x}$.

## 3.5. The DPNLP Algorithm

We now describe the steps in the DPNLP algorithm. As noted earlier, the Stone–Weierstrass theorem and Bernstein approximation theory tell us that (13) will work for a sufficiently flexible approximation $\hat{V}$ and sufficiently large sets of approximation and shape nodes. We do not know how much flexibility is needed. There may be approximation theorems that will give us specifications that will work, but they will tend to be far in excess of what will work. We therefore take a guess-and-adjust approach to DPNLP; that is, we take small grids and low-order approximations, solve (13), and check the shape of the result. If the shape is bad, we then try again with a more flexible strategy.

If the approximation method is not sufficiently flexible (for example, if the polynomial approximation is of too low a degree), the solution to (13) will imply a solution where the inequality in (21) is not binding, as it would be if we had the true solution for $V(x)$. It is also the case that some of the shape constraints, $\hat{V}'(y_{i'}; \mathbf{b}) \geq 0$ or $\hat{V}''(y_{i'}; \mathbf{b}) \leq 0$, are binding at some shape nodes, which also should not be the case for the true $V(x)$. That is, the true solution of the basic DPNLP model (13) should let the inequality constraints,

$$v_i \leq u(x_i, a_i) + \beta \hat{V}(x_i^+; \mathbf{b}), \qquad \textbf{(21)}$$

be binding for all $i = 1, \ldots, m$, and $\hat{V}(y_{i'}; \mathbf{b})$ should be strictly increasing and concave at all the shape nodes.

In this paper, we use the Chebyshev polynomial approximation in $\hat{V}$. For a smooth function, we know that the Chebyshev polynomial approximation usually has smaller coefficients for higher-degree terms. This tells us that a small-degree Chebyshev polynomial approximation in $\hat{V}$ is a good initial guess for a higher-degree Chebyshev polynomial approximation. Therefore, we use the following DPNLP algorithm with stepwise construction of value functions to solve the infinite-horizon deterministic optimal decision-making problems by stepwise construction of value functions.

*Algorithm* 1. DPNLP algorithm for infinite-horizon deterministic optimal decision-making problems

*Initialization.* Choose a lexicographically ordered sequence of integer vectors $(n_J, m_J, m'_J)$. For each $J = 1, 2, 3, \ldots$, choose $m_J$ approximation nodes $\{x_i^J : 1 \le i \le m_J\}$ and $m'_J$ shape nodes $\{y_i^J : 1 \le i \le m'_J\}$ in $\mathbf{X}$. Set $J = 1$.

*Step* 1. Solve the basic DPNLP model (13) with degree-$n_J$ Chebyshev polynomial approximation $\hat{V}(x; \mathbf{b}^J)$ over the approximation nodes $\{x_i^J : 1 \le i \le m_J\}$ and the shape nodes $\{y_i^J : 1 \le i \le m'_J\}$ (by using the previous step's solutions as an initial guess when $J > 1$).

*Step* 2. Check shape constraints; if there is a violation, increase $J$, and go to Step 1. Else go to Step 3.

*Step* 3. Draw $N$ random points $z_j \in \mathbf{X}$, compute $\widetilde{v}_j = \Gamma\left(\hat{V}\right)(z_j)$ in (17) for $j = 1, \ldots, N$, and then estimate the error norm (19). If the error is too large, increase $J$ and go to step 1; otherwise exit.

The choices of $(n_J, m_J, m'_J)$ and corresponding approximation nodes and shape nodes depend on experience. Because a quadratic Chebyshev polynomial approximation to $V(x)$ is a good shape-preserving approximation with increasing and concave properties, it is usually good to choose $n_1 = 2$ in the initialization step of Algorithm 1. In our examples, we just choose fixed $m_J = m$ and $m'_J = m'$ for simplicity and let $n_J = J + 1$ for $J = 1, 2, \ldots, m - 1$, and the approximation nodes are the expanded Chebyshev nodes in $\mathbf{X} = [x^{\min}, x^{\max}]$.

## 4. APPLICATIONS TO DETERMINISTIC OPTIMAL GROWTH PROBLEMS

An infinite-horizon economic problem is the discrete-time optimal growth model with one good and one capital stock, which is a deterministic model.[2] The aim is to find the value function defined by

$$V(k_0) = \max_{c,l} \sum_{t=0}^{\infty} \beta^t u(c_t, l_t), \qquad \textbf{(22)}$$
$$\text{s.t.} \quad k_{t+1} = F(k_t, l_t) - c_t,$$

where $k_t$ is the capital stock at time $t$ with $k_0$ in [0.3, 2], $c_t$ is consumption, $l_t$ is the labor supply, $\beta$ is the discount factor, $F(k, l)$ is the aggregate production function, and $u(c_t, l_t)$ is the utility function. In the examples, the aggregate production function is $F(k, l) = k + Ak^{\psi} l^{1-\psi}$ with $\psi = 0.25$ and $A = (1 - \beta)/(\psi\beta)$. The utility function is

$$u(c,l) = \frac{(c/A)^{1-\gamma}-1}{1-\gamma} - (1-\psi)\frac{l^{1+\eta}-1}{1+\eta}. \tag{23}$$

The functional forms for utility and production imply that the steady state of the infinite-horizon deterministic optimal growth problems is $k_{\text{ss}} = 1$, and the optimal consumption and the optimal labor supply at $k_{\text{ss}}$ are, respectively, $c_{\text{ss}} = A$ and $l_{\text{ss}} = 1$. The code for DPNLP is written in GAMS [McCarl et al. (2015)], and the optimization solver is CONOPT (in the GAMS environment).

## 4.1. True Solution

To estimate the accuracy of the solution given by DPNLP, we compute the "true" optimal solution on a large set of test points for initial capital $k_0 \in [0.3, 2]$ and then compare those results with the computed optimal solution from DPNLP. To get the "true" optimal solution, we discretize the range of capital, [0.3, 2], with one million equally spaced capital nodes, and also discretize the range of labor supply, [0.4, 2.5], with another one million equally spaced labor supply nodes. For any discretized capital node $k$ and labor supply node $l$, we choose consumption $c = F(k,l) - k^+$ such that $k^+$ is also one of the discretized capital nodes. Using the discretized capital nodes as discrete states, we apply the alternating sweep Gauss–Seidel algorithm [Judd (1998)] to compute the optimal value function until it converges under the stopping criterion $10^{-7}$.

## 4.2. DPNLP Solution

We use the DPNLP algorithm (Algorithm 1) to solve the deterministic optimal growth problem. The basic DPNLP model is

$$\max_{c,l,k^+,v,\mathbf{b}} \sum_{i=1}^{m} v_i, \tag{24}$$

$$\begin{aligned}
\text{s.t.}\quad & v_i \le u(c_i,l_i) + \beta \hat{V}(k_i^+;\mathbf{b}), \quad i = 1,\ldots,m,\\
& k_i^+ \le F(k_{i,},l_i) - c_i, \quad i = 1,\ldots,m,\\
& v_i = \hat{V}(k_i;\mathbf{b}), \quad i = 1,\ldots,m,\\
& \hat{V}'(y_{i'};\mathbf{b}) \ge 0, \quad i' = 1,\ldots,m',\\
& \hat{V}''(y_{i'};\mathbf{b}) \le 0, \quad i' = 1,\ldots,m'.
\end{aligned}$$

For our examples in this section, we always choose $m = 19$ expanded Chebyshev nodes, $k_i$, in [0.3, 2] as the approximation nodes; the approximation method, $\hat{V}$, is the expanded Chebyshev polynomial up to the maximal degree 18; and we choose $m' = 100$ expanded Chebyshev nodes, $y_{i'}$, in [0.3, 2] as the shape nodes.

**TABLE 1.** Errors of DPNLP for deterministic optimal growth problems

| $\beta$ | $\gamma$ | $\eta$ | Error to truth $c^*_{\text{DPNLP}}$ | Error to truth $l^*_{\text{DPNLP}}$ | Error norm (19) for $\hat{V}$ | Time (seconds) |
|---|---|---|---|---|---|---|
| 0.9 | 0.5 | 0.2 | 1.5(−6) | 1.8(−6) | 5.7(−8) | 3.6 |
| | | 1 | 3.1(−6) | 1.5(−6) | 5.7(−8) | 3.5 |
| | | 5 | 3.0(−6) | 1.1(−6) | 5.6(−8) | 3.6 |
| | 2 | 0.2 | 1.1(−6) | 3.6(−6) | 1.6(−7) | 4.0 |
| | | 1 | 1.4(−6) | 2.3(−6) | 1.9(−7) | 4.0 |
| | | 5 | 2.2(−6) | 1.2(−6) | 2.4(−7) | 3.5 |
| | 8 | 0.2 | 9.7(−6) | 3.7(−6) | 2.7(−7) | 4.6 |
| | | 1 | 1.0(−6) | 2.6(−6) | 4.9(−7) | 3.8 |
| | | 5 | 1.5(−6) | 3.5(−6) | 2.2(−6) | 3.8 |
| 0.95 | 0.5 | 0.2 | 3.1(−6) | 3.7(−6) | 7.5(−8) | 3.9 |
| | | 1 | 4.7(−6) | 1.9(−6) | 7.2(−8) | 3.7 |
| | | 5 | 4.8(−6) | 1.2(−6) | 7.0(−8) | 3.5 |
| | 2 | 0.2 | 1.6(−6) | 5.8(−6) | 2.0(−7) | 4.1 |
| | | 1 | 2.2(−6) | 3.4(−6) | 2.3(−7) | 4.0 |
| | | 5 | 3.5(−6) | 1.9(−6) | 2.8(−7) | 3.8 |
| | 8 | 0.2 | 1.2(−6) | 6.7(−6) | 3.3(−7) | 4.5 |
| | | 1 | 1.2(−6) | 5.2(−6) | 5.8(−7) | 4.3 |
| | | 5 | 2.8(−6) | 4.8(−6) | 2.5(−6) | 4.1 |
| 0.99 | 0.5 | 0.2 | 1.2(−5) | 1.3(−5) | 1.1(−7) | 4.5 |
| | | 1 | 3.0(−5) | 1.1(−5) | 9.7(−8) | 4.0 |
| | | 5 | 4.2(−5) | 4.3(−6) | 9.1(−8) | 3.7 |
| | 2 | 0.2 | 6.1(−6) | 2.4(−5) | 2.7(−7) | 5.4 |
| | | 1 | 1.0(−5) | 1.6(−5) | 2.7(−7) | 5.2 |
| | | 5 | 1.8(−5) | 7.7(−6) | 3.2(−7) | 5.5 |
| | 8 | 0.2 | 2.0(−6) | 3.2(−5) | 4.2(−7) | 7.0 |
| | | 1 | 3.9(−6) | 2.2(−5) | 6.6(−7) | 6.0 |
| | | 5 | 1.1(−5) | 1.6(−5) | 2.8(−6) | 6.6 |

*Note*: $a(k)$ means $a \times 10^k$.

In fact, in some cases among our examples, we could use fewer numbers to save computational time, but with almost the same accuracy.

## 4.3. Error Analysis of DPNLP Solution

We next use some basic examples of the deterministic optimal growth problem to test DPNLP. We try $\beta = 0.9,\ 0.95,\ 0.99$, $\gamma = 0.5, 2, 8$, and $\eta = 0.2, 1, 5$; all these examples give us good solutions.

Table 1 lists relative errors of optimal solutions computed by DPNLP for these cases in comparison with the "true" solution given by the high-precision discretization method. The errors for optimal consumptions are computed by

$$\max_{k\in[0.3,2]} \frac{|c^*_{\mathrm{DPNLP}}(k) - c^*(k)|}{|c^*(k)|},$$

where $c^*_{\mathrm{DPNLP}}(k)$ is the optimal consumption computed by DPNLP, and $c^*(k)$ is the "true" optimal consumption, for $k \in [0.3, 2]$. The errors for optimal labor supply, $l^*_{\mathrm{DPNLP}}$, have a similar computation formula. The last column of Table 1 lists the running time of the DPNLP algorithm for various cases in the GAMS environment, on a single core of a Mac laptop with a 2.5 GHz processor.

Table 1 shows that DPNLP solves the examples with accuracy up to five digits or higher for optimal control policy functions in all cases. Moreover, the DPNLP algorithm is fast and takes only a few seconds for each case. For example, row 1 in Table 1 assumes $\beta = 0.9$, $\gamma = 0.5$, and $\eta = 0.2$. For this case, the error relative to the "true" solution in consumption is $1.5 \times 10^{-6}$, the error relative to the "true" solution in labor supply is $1.8 \times 10^{-6}$, and the running time is only 3.6 seconds. We also use the measure (19) to estimate the upper bounds of errors of the value function approximation $\hat{V}(x; \mathbf{b})$ using $N = 1,000$ sample points and find that the errors are from $O\left(10^{-6}\right)$ to $O\left(10^{-8}\right)$ for all cases; e.g., the estimated error upper bound for the value function approximation in row 1 of Table 1 is $5.7 \times 10^{-8}$ [whereas $\hat{V}'(\overline{x}; \mathbf{b}) = 2.5$ with $\overline{x} = k_{\mathrm{ss}} = 1$ in (19)].

## 5. APPLICATIONS TO STOCHASTIC OPTIMAL GROWTH PROBLEMS

When the capital stock is dependent on a random economic shock $\theta_t$, the optimal growth problem (22) becomes a stochastic dynamic optimization problem. Assume that the random economic shock $\theta_t$ is a stochastic process following $\theta_{t+1} = h(\theta_t, \varepsilon_t)$, where $\epsilon_t$ is a serially uncorrelated random process. Let $f(k, l, \theta)$ denote the net production function, and $F(k, l, \theta) = k + f(k, l, \theta)$. Then the infinite-horizon discrete-time stochastic optimization problem becomes

$$\begin{aligned} V(k_0, \theta_0) = \max_{k,c,l} \ &\mathbf{E}\left\{\sum_{t=0}^{\infty} \beta^t u(c_t, l_t)\right\}, \\ \text{s.t.} \quad k_{t+1} &= F(k_t, l_t, \theta_t) - c_t, \\ \theta_{t+1} &= h(\theta_t, \varepsilon_t), \end{aligned} \tag{25}$$

where $k_0 \in [0.3, 2]$ and $\theta_0$ are given. The parameter $\theta$ has many economic interpretations. In the life-cycle interpretation, $\theta$ is a state variable that may affect asset income, labor income, or both. In the monopolist interpretation, $\theta$ may reflect shocks to costs, demand, or both.

We use the same utility function (23), but the production function is changed to

$$F(k, l, \theta) = k + \theta A k^{\psi} l^{1-\psi},$$

where $\theta$ is the stochastic state, $\psi = 0.25$, and $A = (1-\beta)/(\psi\beta)$. In the examples, $\theta_t$ is assumed to be a Markov chain with three possible values,

$$\vartheta_1 = 0.95, \ \vartheta_2 = 1.0, \ \vartheta_3 = 1.05,$$

and the probability transition matrix from $\theta_t$ to $\theta_{t+1}$ is

$$P = \begin{bmatrix} 0.75 & 0.25 & 0 \\ 0.25 & 0.5 & 0.25 \\ 0 & 0.25 & 0.75 \end{bmatrix}.$$

The code for DPNLP is written in GAMS [McCarl et al. (2015)], and the optimization solver is CONOPT (in the GAMS environment).

## 5.1. True Solution

For the deterministic optimal growth problem (22), we use the discretized method and the alternating sweep Gauss–Seidel algorithm to get the "true" solution. But the DP method with high-precision discretization will be too time-consuming for solving the stochastic optimal growth problem (25). However, Cai and Judd (2012) introduce a value function iteration method using a shape-preserving rational spline interpolation and show that it is very accurate for solving multiperiod portfolio optimization problems. For the deterministic optimal growth problem, because the value function is smooth, increasing, and concave over the continuous state, capital $k$, we can also apply this shape-preserving DP algorithm to solve the deterministic optimal growth problem and realize that it is also very accurate (by comparing its solution with those given by the alternating sweep Gauss–Seidel algorithm).

For the stochastic optimal growth problem, the value function for each discrete state is also smooth, increasing, and concave over the continuous state, capital $k$. Therefore, we can again choose the shape-preserving value function iteration method to solve the stochastic optimal growth problem and iterate until it converges under the stopping criterion $10^{-7}$. We use 1,000 equally spaced interpolation nodes on the range of the continuous state, [0.3, 2], for each discrete state $\theta$.

## 5.2. DPNLP Solution

We use the DPNLP algorithm (the stochastic version of Algorithm 1) to solve the stochastic optimal growth problem. The basic DPNLP model is

$$\max_{c,l,k^+,v,\mathbf{b}} \quad \sum_{j=1}^{J} \sum_{i=1}^{m} v_{i,j}, \tag{26}$$

$$\begin{aligned} \text{s.t.} \quad & v_{i,j} \leq u(c_{i,j}, l_{i,j}) + \beta \sum_{j'=1}^{J} P_{j,j'} \hat{V}(k_{i,j}^{+}, \vartheta_{j'}; \mathbf{b}), \\ & k_{i,j}^{+} \leq F(k_{i,}, l_{i,j}, \vartheta_j) - c_{i,j}, \\ & v_{i,j} = \hat{V}(k_i, \vartheta_j; \mathbf{b}), \\ & \hat{V}'(y_{i'}, \vartheta_j; \mathbf{b}) \geq 0, \\ & \hat{V}''(y_{i'}, \vartheta_j; \mathbf{b}) \leq 0, \\ & i = 1, \ldots, m, \;\; j = 1, \ldots, J, \;\; i' = 1, \ldots, m', \end{aligned}$$

where $J = 3, m = 19, m' = 100$, $k_i$ are expanded Chebyshev nodes in [0.3, 2], $\hat{V}$ is the expanded Chebyshev polynomial up to the maximal degree 18, and $y_{i'}$ are expanded Chebyshev nodes in [0.3, 2] as the shape nodes.

## 5.3. Error Analysis of the DPNLP Solution

We examine the errors for the stochastic model in a manner similar to that for the deterministic optimal growth problems: We apply high-precision value function iteration to get the "true" optimal solution for every test point of initial capital $k_0$ and every possible initial discrete state $\theta_0$ and then use them to check the accuracy of the computed optimal solution from the DPNLP model (26).

Table 2 lists relative errors of optimal solutions computed by DPNLP for the stochastic optimal growth problem with the following cases: $\beta = 0.9,\ 0.95,\ 0.99$, $\gamma = 0.5, 2, 8$, and $\eta = 0.2, 1, 5$. The errors for optimal consumptions at time 0 are computed by

$$\max_{k\in[0.3,2],\theta\in\{0.95,1.0,1.05\}} \frac{|c^*_{\text{DPNLP}}(k,\theta) - c^*(k,\theta)|}{|c^*(k,\theta)|},$$

where $c^*_{\text{DPNLP}}$ is the optimal consumption computed by DPNLP on the model (26), and $c^*$ is the "true" optimal consumption computed by the high-precision value function iteration method. A similar formula applies to compute errors for optimal labor supply. The last column of Table 2 lists the running time of the DPNLP algorithm for various cases in the GAMS environment, on a single core of a Mac laptop with a 2.5 GHz processor.

TABLE 2. Errors of DPNLP for stochastic optimal growth problems

| | | | Error to truth | | | |
|---|---|---|---|---|---|---|
| $\beta$ | $\gamma$ | $\eta$ | $c^*_{\text{DPNLP}}$ | $l^*_{\text{DPNLP}}$ | Error norm (19) for $\hat{V}$ | Time (seconds) |
| 0.9 | 0.5 | 0.2 | 1.9(−7) | 5.2(−7) | 5.8(−8) | 10 |
| | | 1 | 2.5(−7) | 4.5(−7) | 5.8(−8) | 9 |
| | | 5 | 2.5(−7) | 4.7(−7) | 5.8(−8) | 9 |
| | 2 | 0.2 | 1.4(−7) | 5.0(−7) | 1.7(−7) | 15 |
| | | 1 | 2.0(−7) | 5.9(−7) | 1.9(−7) | 12 |
| | | 5 | 3.0(−7) | 4.4(−7) | 2.4(−7) | 11 |
| | 8 | 0.2 | 1.1(−7) | 8.4(−7) | 2.8(−7) | 21 |
| | | 1 | 1.6(−7) | 8.8(−7) | 5.3(−7) | 18 |
| | | 5 | 8.5(−7) | 1.2(−6) | 2.5(−6) | 15 |
| 0.95 | 0.5 | 0.2 | 3.7(−7) | 4.8(−7) | 7.9(−8) | 15 |
| | | 1 | 3.9(−7) | 4.2(−7) | 7.5(−8) | 11 |
| | | 5 | 4.4(−7) | 4.4(−7) | 7.2(−8) | 9 |
| | 2 | 0.2 | 2.9(−7) | 6.6(−7) | 2.0(−7) | 22 |
| | | 1 | 3.2(−7) | 5.9(−7) | 2.3(−7) | 18 |
| | | 5 | 4.4(−7) | 4.4(−7) | 2.8(−7) | 13 |
| | 8 | 0.2 | 2.3(−7) | 9.6(−7) | 3.4(−7) | 25 |
| | | 1 | 3.0(−7) | 8.7(−7) | 6.0(−7) | 21 |
| | | 5 | 9.7(−7) | 1.3(−6) | 2.6(−6) | 22 |
| 0.99 | 0.5 | 0.2 | 4.1(−7) | 6.1(−7) | 1.2(−7) | 31 |
| | | 1 | 4.5(−7) | 4.6(−7) | 1.0(−7) | 22 |
| | | 5 | 4.1(−7) | 4.6(−7) | 9.6(−8) | 17 |
| | 2 | 0.2 | 3.0(−7) | 1.1(−6) | 2.9(−7) | 49 |
| | | 1 | 3.4(−7) | 7.4(−7) | 3.0(−7) | 40 |
| | | 5 | 5.9(−7) | 5.4(−7) | 3.4(−7) | 45 |
| | 8 | 0.2 | 1.5(−7) | 1.5(−6) | 4.5(−7) | 54 |
| | | 1 | 1.8(−7) | 1.3(−6) | 7.0(−7) | 57 |
| | | 5 | 2.2(−6) | 3.1(−6) | 2.9(−6) | 55 |

*Note*: $a(k)$ means $a \times 10^k$.

From Table 2, we can also see a pattern similar to that shown in Table 1. That is, DPNLP solves the examples with accuracy up to six or more digits for optimal control policy functions in all cases. Moreover, for these stochastic examples, DPNLP is also fast, and takes less than one minute to solve any one case. For example, row 1 in Table 2 assumes $\beta = 0.9$, $\gamma = 0.5$, and $\eta = 0.2$. For this case, the error in the "true" solution in consumption is $1.9 \times 10^{-7}$, the error in the "true" solution in labor supply is $5.2 \times 10^{-7}$, and the running time is only 10 seconds. We also use the stochastic variant of the measure (19) to estimate the upper bounds of errors of the value function approximation $\hat{V}(x; \mathbf{b})$ using $N = 1,000$ sample points and find that the errors are from $O\left(10^{-6}\right)$ to $O\left(10^{-8}\right)$ for all cases; e.g., the estimated error upper bound of the value function approximation in row 1 of Table 2 is $5.8 \times 10^{-8}$ [whereas $\hat{V}'(\overline{x}, \overline{\theta}; \mathbf{b}) = 2.5$ with $\overline{x} = 1$ and $\overline{\theta} = 1$ in (20)].

## 6. APPLICATIONS TO TWO-DIMENSIONAL OPTIMAL GROWTH PROBLEMS

The key DPNLP idea is clearly applicable to multidimensional problems. Of course, multidimensional problems are more demanding. Our next example illustrates DPNLP applied to two-dimensional extensions of our earlier models. The results indicate that DPNLP is a reasonable method for low-dimensional problems.

We assume that there are two countries, and let $k_t = (k_{t,1}, k_{t,2})$, which is a two-dimensional continuous state vector at time $t$, denote their capital stocks. Let $l_t = \left(l_{t,1}, l_{t,2}\right)$, which is a two-dimensional continuous control vector variable at time $t$,denote the elastic labor supply levels of the countries. Assume that the net production of country $i$ at time $t$ is

$$f_i(k_{t,i}, l_{t,i}) = Ak_{t,i}^{\psi} l_{t,i}^{1-\psi},$$

with $A = (1-\beta)/(\psi\beta)$, for $i = 1, 2$. Let $c_t = \left(c_{t,1}, c_{t,2}\right)$ denote the consumption of the countries, which is another two-dimensional continuous control vector variable at time $t$. The utility function is

$$u(c, l) = \sum_{i=1}^{2} \left[ \frac{(c_i/A)^{1-\gamma} - 1}{1-\gamma} - (1-\psi) \frac{l_i^{1+\eta} - 1}{1+\eta} \right].$$

We want to find optimal consumption and labor supply decisions such that expected total utility over the infinite-horizon time is maximized. That is,

$$\begin{aligned} V(k_0) = \max_{k_t, I_t, c_t, l_t} & \sum_{t=0}^{\infty} \beta^t u(c_t, l_t), \\ \text{s.t.}\quad & k_{t+1,i} = (1-\delta)k_{t,i} + I_{t,i}, \\ & G_{t,i} = \frac{\zeta}{2} k_{t,i} \left( \frac{I_{t,i}}{k_{t,i}} - \delta \right)^2, \\ & \sum_{i=1}^{2} \left(c_{t,i} + I_{t,i} - \delta k_{t,i}\right) = \sum_{i=1}^{2} \left(f_i(k_{t,i}, l_{t,i}) - G_{t,i}\right), \end{aligned} \qquad (27)$$

where $\delta$ is the depreciation rate of capital, $I_{t,i}$ is the investment of country $i$, $G_{t,i}$ is the investment adjustment cost of country $i$, and $\zeta$ governs the intensity of the friction. Detailed discussions of multicountry growth models with infinite horizon can be seen in Den Haan et al. (2011) and Juillard and Villemot (2011). Multicountry growth models with finite horizon can be solved efficiently using dynamic programming with Hermite approximation; see Cai and Judd (2015). In our examples, we let $\psi = 0.36$, $\delta = 0.025$, and $\zeta = 0.5$.

The functional forms for utility and production imply that the steady state of the infinite-horizon deterministic optimal growth problems is $k_{ss,1} = k_{ss,2} = 1$, and the optimal consumption, labor supply, and investment in the steady state are, respectively, $c_{ss,1} = c_{ss,2} = A$, $l_{ss,1} = l_{ss,2} = 1$, and $I_{ss,1} = I_{ss,2} = \delta$. The code for DPNLP is written in GAMS [McCarl et al. (2015)], and the optimization solver is CONOPT (in the GAMS environment).

## 6.1. True Solution

The discretization method of solving the two-country optimal growth problem with two continuous state variables $(k_{t,1}, k_{t,2})$ and six continuous control variables $(c_{t,1}, c_{t,2}, l_{t,1}, l_{t,2}, I_{t,1}, I_{t,2})$ will be too time-consuming. To get the "true" solution, we use value function iteration with high-degree complete Chebyshev polynomials and iterate until it converges under the stopping criterion $10^{-7}$ (i.e., the difference between two consecutive value functions is less than $10^{-7}$). We use $51^2$ tensor Chebyshev nodes on the state space $[0.5, 1.5]^2$, and the degree of the complete Chebyshev polynomials is 30.

## 6.2. DPNLP Solution

Using the DPNLP algorithm for multidimensional problems requires a strategy for imposing concavity and monotonicity of multidimensional functions. In the one-dimensional case, we just need to impose inequality constraints on derivatives, constraints that are linear in the unknowns $\mathbf{b}$, as we use the polynomial approximation $\hat{V}$. For multidimensional functions, concavity is usually expressed in terms of the definiteness of the Hessian. Any inequality constraint regarding the Hessian at shape points will be highly nonlinear. The DPNLP method utilizes a more fundamental definition of concavity: a function is concave in $\mathbf{R}^d$ if it is concave in all directions $\mathbf{r} \in \mathbf{R}^d$. Therefore, the DPNLP algorithm imposes a collection of directional derivative conditions, which are linear in the $\mathbf{b}$ unknowns. In the two-dimensional examples of this paper, we only needed to impose concavity in the coordinate directions [i.e., $\mathbf{r} = (1, 0)^\top$ and $\mathbf{r} = (0, 1)^\top$] . More generally, one will need to be ready to use a greater variety of directions, such as $\mathbf{r} = (1, 1)^\top$ and $\mathbf{r} = (1, -1)^\top$. The key fact is that the Stone–Weierstrass and related theorems tell us that this will work for sufficiently numerous choices of directional shape constraints.

The basic DPNLP model is the multidimensional extension of the model (24) with a set of directions $\{\mathbf{r}_1, \ldots, \mathbf{r}_{m''}\}$ for the concavity constraints, i.e.,

$$\begin{aligned}
\max_{c,l,I,k^+,v,\mathbf{b}} \quad & \sum_{i=1}^{m} v_i, \qquad\qquad (\mathbf{28})\\
\text{s.t.} \quad & v_i \le u(c_i, l_i) + \beta \hat{V}(k_i^+; \mathbf{b}), \quad i = 1, \dots, m,\\
& k_{i,j}^+ \le (1-\delta)k_{i,j} + I_{i,j}, \quad i = 1, \dots, m, \quad j = 1, 2,\\
& G_{i,j} = \frac{\zeta}{2} k_{i,j} \left( \frac{I_{i,j}}{k_{i,j}} - \delta \right)^2, \quad i = 1, \dots, m, \quad j = 1, 2,\\
& \sum_{j=1}^{2} \left( c_{i,j} + I_{i,j} - \delta k_{i,j} \right) = \sum_{j=1}^{2} \left( f_j(k_{i,j}, l_{i,j}) - G_{i,j} \right), \quad i = 1, \dots, m,\\
& v_i = \hat{V}(k_i; \mathbf{b}), \quad i = 1, \dots, m,\\
& \hat{V}'(y_{i'}; \mathbf{b}) \ge 0, \quad i' = 1, \dots, m',\\
& \mathbf{r}_j^\top \hat{V}''(y_{i'}; \mathbf{b}) \mathbf{r}_j \le 0, \quad i' = 1, \dots, m', \; j = 1, \dots, m'',
\end{aligned}$$

where $k_i = (k_{i,1}, k_{i,2})$, $c_i = (c_{i,1}, c_{i,2})$, $l_i = (l_{i,1}, l_{i,2})$, $k_i^+ = (k_{i,1}^+, k_{i,2}^+)$, $y_{i'} = (y_{i',1}, y_{i',2})$, $\hat{V}'$ is the 2-dimensional gradient of $\hat{V}$, and $\hat{V}''$ is the Hessian matrix of $\hat{V}$.

For our examples in this section, we choose $m = 11^2$ tensor Chebyshev nodes in the state space $[0.5, 1.5]^2$ as the approximation nodes. The approximation method, $\hat{V}$, is the complete Chebyshev polynomial up to the maximal degree 10. We choose $m' = 100$ tensor Chebyshev nodes, $y_{i'}$, in the state space $[0.5, 1.5]^2$ as the shape nodes.

## 6.3. Error Analysis of the DPNLP Solution

We examine the errors for the multidimensional model in a manner similar to that for the unidimensional optimal growth problems: We apply high-precision value function iteration to get the "true" optimal solution for every test point of initial capitals, and then use them to check the accuracy of the computed optimal solution from the DPNLP model (28).

Table 3 lists relative errors of optimal solutions computed by DPNLP for the two-dimensional optimal growth problem with the following cases: $\beta = 0.95$, $\gamma = 0.5, 2, 8$, and $\eta = 0.2, 1, 5$. The last column of Table 3 lists the running time of the DPNLP algorithm for various cases in the GAMS environment, on a single core of a Mac laptop with a 2.5 GHz processor.

Table 3 shows that DPNLP solves the examples with accuracy up to four digits or higher for optimal control policy functions in all the cases except one case having three digits. Moreover, the DPNLP algorithm is not slow and takes only several minutes for each case. For example, row 1 in Table 3 assumes $\gamma = 0.5$ and $\eta = 0.2$. For this case, the error to the "true" solution in consumption is

**TABLE 3.** Errors of DPNLP for two-country optimal growth problems

| | | Error to truth | | | |
|---|---|---|---|---|---|
| $\gamma$ | $\eta$ | $c^*_{\text{DPNLP}}$ | $l^*_{\text{DPNLP}}$ | Error norm (19) for $\hat{V}$ | Time (minutes) |
| 0.5 | 0.2 | 6.4(−5) | 1.5(−4) | 4.3(−5) | 3.7 |
| | 1 | 9.0(−6) | 3.0(−5) | 8.1(−6) | 2.5 |
| | 5 | 8.0(−6) | 8.0(−7) | 2.5(−6) | 1.4 |
| 2 | 0.2 | 6.2(−5) | 2.3(−4) | 8.5(−5) | 3.6 |
| | 1 | 4.5(−5) | 6.5(−5) | 4.3(−5) | 3.0 |
| | 5 | 8.5(−5) | 3.2(−5) | 2.2(−5) | 2.2 |
| 8 | 0.2 | 8.4(−5) | 1.2(−3) | 2.3(−4) | 4.9 |
| | 1 | 2.1(−5) | 1.1(−4) | 8.1(−5) | 6.6 |
| | 5 | 1.3(−4) | 2.0(−4) | 6.3(−5) | 5.6 |

*Note*: $a(k)$ means $a \times 10^k$.

$6.4 \times 10^{-5}$, the error to the "true" solution in labor supply is $1.5 \times 10^{-4}$, and the running time is 3.7 minutes. We also use the multidimensional variant of the measure (19) to estimate the upper bounds of the errors of the value function approximation $\hat{V}(x; \mathbf{b})$ using $N = 1,000$ sample points, and find that the errors are from $O\left(10^{-4}\right)$ to $O\left(10^{-6}\right)$ for all cases; e.g., the estimated error upper bound of the value function approximation in row 1 of Table 3 is $4.3 \times 10^{-5}$ [whereas $\hat{V}'(\overline{x}; \mathbf{b}) = (0.36, 0.36)$ with $\overline{x} = k_{\text{ss}} = (1, 1)$ in (19)].

## 7. CONCLUSION

This paper presents a nonlinear programming formulation of dynamic programming problems common in economic decision making. We have applied it to a variety of optimal accumulation problems, showing that our DPNLP algorithm performs very well, with high accuracy, reliability, and efficiency for those problems. A variety of example problems indicate that our algorithm could be applied to many problems.

## NOTES

1. From the contraction mapping property of $\Gamma$ and $\Gamma(V) = V$, as $V$ is the fixed point of $\Gamma$, we have

$$\left\|\Gamma\left(\hat{V}\right) - V\right\|_\infty = \left\|\Gamma\left(\hat{V}\right) - \Gamma\left(V\right)\right\|_\infty \leq \beta\left\|\hat{V} - V\right\|_\infty;$$

thus, from the triangle inequality, we get

$$\left\|\hat{V} - V\right\|_\infty \leq \left\|\hat{V} - \Gamma\left(\hat{V}\right)\right\|_\infty + \left\|\Gamma\left(\hat{V}\right) - V\right\|_\infty \leq \left\|\hat{V} - \Gamma\left(\hat{V}\right)\right\|_\infty + \beta\left\|\hat{V} - V\right\|_\infty.$$

This implies the inequality (16).

2. See Judd (1998) for a detailed description of this.

## REFERENCES

Bellman, R. (1957) *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Blackwell, D. (1965) Discounted dynamic programming. *Annals of Mathematical Statistics* 36(1), 226–235.

Bojanic, R. and F. Cheng (1989) Rate of convergence of Bernstein polynomials for functions with derivatives of bounded variation. *Journal of Mathematical Analysis and Applications* 141, 136–151.

Cai, Y. (2010) Dynamic Programming and Its Application in Economics and Finance. Ph.D. thesis, Stanford University.

Cai, Y. and K.L. Judd (2010) Stable and efficient computational methods for dynamic programming. *Journal of the European Economic Association* 8(2–3), 626–634.

Cai, Y. and K.L. Judd (2012) Dynamic programming with shape-preserving rational spline Hermite interpolation. *Economics Letters* 117(1), 161–164.

Cai, Y. and K.L. Judd (2013) Shape-preserving dynamic programming. *Mathematical Methods of Operations Research* 77(3), 407–421.

Cai, Y. and K.L. Judd (2014) Advances in numerical dynamic programming and new applications. In Karl Schmedders and Kenneth L. Judd (eds.), *Handbook of Computational Economics*, Vol. 3, Chap. 8. Elsevier.

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

Carnicer, J.M. and J.M. Pena (1993) Shape preserving representations and optimality of the Bernstein basis. *Advances in Computational Mathematics* 1 173{196.

De Farias, D.P. and B. Van Roy (2003) The linear programming approach to approximate dynamic programming. *Operations Research* 51(6), 850–865.

Den Haan, W.J., K.L. Judd, and M. Juillard (2011) Computational suite of models with heterogeneous agents: II. Multi-country real business cycle models. *Journal of Economic Dynamics and Control* 35, 175–177.

Farouki, R.T. (2012) The Bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design* 29, 379–419.

Goodman, T.N.T. (1989) Shape preserving representations. In T. Lyche and L.L. Schumaker (eds.), *Mathematical Methods in Computer Aided Geometric Design*, pp. 333–351. Boston: Academic Press.

Judd, K.L. (1998) *Numerical Methods in Economics*. Cambridge, MA: MIT Press.

Juillard, M. and S. Villemot (2011) Multi-country real business cycle models: Accuracy tests and test bench. *Journal of Economic Dynamics and Control* 35, 178–185.

Maliar, L. and S. Maliar (2004) Endogenous growth and endogenous business cycles. *Macroeconomic Dynamics* 8(5), 559–581.

McCarl, B., A. Meeraus, P. van der Eijk, M. Bussieck, S. Dirkse, P. Steacy, and F. Nelissen (2015) *McCarl GAMS User Guide*. GAMS Development Corporation. http://www.gams.com/help/index.jsp?topic=%2Fgams.doc%2Fuserguides%2Fmccarl%2Findex.html.

Michelangeli, V. (2009) Economics of the Life-Cycle: Reverse Mortgage, Mortgage and Marriage. Ph.D. thesis, Boston University.

Polak, E. (1997) *Optimization: Algorithms and Consistent Approximations*. New York: Springer.

Rust, J. (1996) Numerical dynamic programming in economics. In H. Amman, D. Kendrick, and J. Rust (eds.), *Handbook of Computational Economics*, Vol. 1, Chap. 14. Elsevier North Holland.

Rust, J. (2008) Dynamic programming. In S.N., Durlauf and L.E. Blume (eds.), *New Palgrave Dictionary of Economics*, 2nd ed. Palgrave Macmillan.

Su, C.H. and K.L. Judd (2012) Constrained optimization approaches to estimation of structural models. *Econometrica* 80(5), 2213–2230.

Trick, M.A. and S.E. Zin (1997) Spline approximations to value functions—Linear programming approach. *Macroeconomic Dynamics* 1, 255–277.

## APPENDIX A: CHEBYSHEV POLYNOMIAL APPROXIMATION

Chebyshev polynomials on $[-1, 1]$ are defined as $\mathcal{T}_j(z) = \cos(j \cos^{-1}(z))$. Economics problems typically live on an interval $[x^{\min}, x^{\max}]$; if we let

$$Z(x) = \frac{2x - x^{\min} - x^{\max}}{x^{\max} - x^{\min}},$$

then $\mathcal{T}_j(Z(x))$ are Chebyshev polynomials adapted to $[x^{\min}, x^{\max}]$ for $j = 0, 1, 2, \ldots$. A degree-$n$ Chebyshev polynomial approximation for $V(x)$ on $[x^{\min}, x^{\max}]$ is

$$\hat{V}(x; \mathbf{b}) = \sum_{j=0}^{n} b_j \mathcal{T}_j(Z(x)), \tag{A.1}$$

where $\mathbf{b} = \{b_j\}$ are the Chebyshev coefficients. It is often more stable to use the expanded Chebyshev polynomial interpolation [Cai (2010)], as the standard Chebyshev polynomial interpolation gives poor approximation in the neighborhood of endpoints.

In this section we describe the Chebyshev polynomial approximation because it is the approximation scheme used in our examples. Although other approximation schemes may also be adequate with good performances, the Chebyshev polynomial approximation presents advantages in terms of coding simplicity and reliability and easy extension to multidimensional approximation.

## APPENDIX B: MULTIDIMENSIONAL COMPLETE CHEBYSHEV APPROXIMATION

In a $d$-dimensional approximation problem, let the domain of the approximation function be

$$\left\{x = (x_1, \ldots, x_d) : x_i^{\min} \le x_i \le x_i^{\max},\ i = 1, \ldots d\right\},$$

for some real numbers $x_i^{\min}$ and $x_i^{\max}$ with $x_i^{\max} > x_i^{\min}$ for $i = 1, \ldots, d$. Let $x^{\min} = (x_1^{\min}, \ldots, x_d^{\min})$ and $x^{\max} = (x_1^{\max}, \ldots, x_d^{\max})$. Then we denote $[x^{\min}, x^{\max}]$ as the domain. Let $\alpha = (\alpha_1, \ldots, \alpha_d)$ be a vector of nonnegative integers. Let $\mathcal{T}_\alpha(z)$ denote the product $\mathcal{T}_{\alpha_1}(z_1) \cdots \mathcal{T}_{\alpha_d}(z_d)$ for $z = (z_1, \ldots, z_d) \in [-1, 1]^d$. Let

$$Z(x) = \left(\frac{2x_1 - x_1^{\min} - x_1^{\max}}{x_1^{\max} - x_1^{\min}}, \ldots, \frac{2x_d - x_d^{\min} - x_d^{\max}}{x_d^{\max} - x_d^{\min}}\right),$$

for any $x = (x_1, \ldots, x_d) \in [x^{\min}, x^{\max}]$.

Using these notations, the degree-$n$ complete Chebyshev approximation for $V(x)$ is

$$\hat{V}_n(x; \mathbf{b}) = \sum_{0 \le |\alpha| \le n} b_\alpha \mathcal{T}_\alpha(Z(x)), \tag{B.1}$$

where $|\alpha| = \sum_{i=1}^{d} \alpha_i$ for the nonnegative integer vector $\alpha = (\alpha_1, \ldots, \alpha_d)$. So the number of terms with $0 \leq |\alpha| = \sum_{i=1}^{d} \alpha_i \leq n$ is $\binom{n+d}{d}$ for the degree-$n$ complete Chebyshev approximation in $\mathbf{R}^d$.