Staff Paper P06-3 January 2006

Staff Paper Series

Accuracy of Numerical Solution to Dynamic Programming Models

by
Heman D. Lohano and Robert P. King

DEPARTMENT OF APPLIED ECONOMICS
COLLEGE OF AGRICULTURAL, FOOD, AND ENVIRONMENTAL SCIENCES
UNIVERSITY OF MINNESOTA

Staff Paper P06-3 January 2006

Accuracy of Numerical Solution to Dynamic Programming Models

Heman D. Lohano and Robert P. King

ACCURACY OF NUMERICAL SOLUTION TO DYNAMIC PROGRAMMING MODELS

Heman D. Lohano and Robert P. King[1]

Abstract

Dynamic programming models with continuous state and control variables are solved approximately using numerical methods in most applications. We develop a method for measuring the accuracy of numerical solution of stochastic dynamic programming models. Using this method, we compare the accuracy of various interpolation schemes. As expected, the results show that the accuracy improves as number of nodes is increased. Comparison of Chebyshev and linear spline indicates that the linear spline may give higher maximum absolute error than Chebyshev, however, the overall performance of spline interpolation is better than Chebyshev interpolation for non-smooth functions. Two-stage grid search method of optimization is developed and examined with accuracy analysis. The results show that this method is more efficient and accurate. Accuracy is also examined by allocating a different number of nodes for each dimension. The results show that a change in node configuration may yield a more efficient and accurate solution.

1. INTRODUCTION

A decision maker must consider future decisions and uncertainty when making current decisions, so most empirical problems in economics are dynamic and stochastic in nature. Dynamic programming is an approach for numerically solving dynamic models. This approach offers considerable flexibility for incorporating real world situations such as nonlinearity and uncertainty in the dynamic economic models.

Economic models commonly have continuous variables such as prices, returns, wealth, assets, and consumption. Dynamic programming problems with continuous state and control variables are solved approximately using numerical methods in most applications, with the exception of some very simple deterministic problems that have closed-form solution. One approach for numerically solving these problems is the discretization method, in which continuous state and control variables are discretized in finite sets, and the value function is approximated with a step function (Judd, 1996, p. 562). This method is reliable if the discretization is made sufficiently fine. Despite advances in computer technology, however, this method becomes impractical for large-scale problems due to limitations of computer storage capacity and of execution time, referred to as the curse of dimensionality (Judd, 1998, pp. 430-33). An alternative, numerically efficient approach is the parametric approach, such as interpolation with polynomials or spline functions, for approximating the value function due to Bellman and Dreyfus and Bellman, Kalaba, and Kotkin.

Implementation of the parametric approach requires choosing the number of nodes, type of nodes and basis functions, optimization method, and integration method. A great challenge in implementation is to choose an appropriate scheme that gives maximum accuracy given the limitations of computer storage capacity and execution time. Johnson et al. and Santos (1999, 2000) have compared accuracy and reduction of computational efforts under various numerical methods for solving dynamic programming models. For multidimensional problems, there are several alternative ways of allocating number of nodes for each dimension. Numerically solving a large-scale stochastic dynamic programming problem involves technical

[1] Heman D. Lohano is an Assistant Professor of Agricultural Economics, Sindh Agriculture University, Tando Jam, Pakistan. Robert P. King is Professor and head of the Department of Applied Economics, University of Minnesota, St. Paul, Minnesota.

challenges in the implementation due to stochastic nature of variables.[2] In addition to comparing selected numerical methods, our purpose here is to propose new techniques for improving accuracy and reducing computational effort in solving large-scale stochastic dynamic programming model.

Measuring the accuracy of the solution of dynamic programming problem poses difficult challenges. One method for checking accuracy is to compare the solution with an actual closed-form solution. However, this method can only be applied when there is a closed-form solution. Another approach is to analyze the Euler equation residuals (Judd, 1992; den Haan and Marcet). However, this approach is of limited value since the problem may not be characterized by the required conditions such as the Euler equation. Another approach to checking accuracy is to compare the solution from one method with the solution from a method assured to be more reliable (Santos, 1999, 2000). However, this approach can be misleading since all numerical methods have error. Measuring accuracy is important not only for choosing an appropriate numerical method but also for knowing the validity and robustness of the solution of the problem. Thus, there is a need for a flexible, reliable method for measuring the accuracy of the solution of a stochastic dynamic programming problem.

This paper develops a method for measuring numerical solution accuracy of a stochastic dynamic programming model. Using this method, which is based on Monte Carlo simulation, this paper evaluates and compares accuracy of alternative numerical methods, and proposes techniques for improving accuracy in solving a large-scale dynamic programming model. This paper also addresses some of the technical challenges associated with representing stochastic variables in a large dynamic programming problem. The accuracy issues considered in this paper are addressed and tested using a dynamic programming problem of farmland investment and financial management. This problem has one control variable and four state variables of which two state variables are stochastic.

The remainder of this paper is organized as follows. Section 2 presents a general overview of the theory of dynamic programming and the parametric approach for solving continuous-state dynamic programming problems. Section 3 specifies the investment decision problem used for accuracy analysis, and describes its implementation for numerical solution. Section 4 develops the method for measuring the accuracy of numerical solution of a stochastic dynamic programming model. Section 5 presents accuracy results. Finally, Section 6 summarizes the results and draws conclusions.

2. DYNAMIC PROGRAMMING AND NUMERICAL METHODS

Dynamic programming is an approach that “takes a sequential or multistage decision process containing many interdependent variables and converts it into a series of single-stage problems, each containing only a few variables” (Nemhauser, p. 6). There is an extensive literature on dynamic programming and its application to economic problems. Stokey and Lucas with Prescott present a rigorous overview of dynamic programming and its properties with application to economic models. Rust surveys the literature on numerical methods and their properties for solving dynamic programming models in economics. Santos (1999) reviews some numerical techniques and their accuracy in solving economic models. Miranda and Fackler apply dynamic programming and numerical methods, implemented in MATLAB, to solve a wide range of dynamic decision problems. This section presents a brief overview of the dynamic programming approach and numerical methods for solving sequential decision problems with discrete time, finite horizon, stochastic states, and continuous state and control variables.

[2] Stochastic dynamic programming models are also referred to as Markov decision processes or stochastic control problems in the literature.

2.1 Sequential Decision Problem

Consider a general problem with J continuous state variables and K continuous control variables. Let S_j be a set of all levels of state variable j for $j = 1,2,\ldots,J$. The state space S is defined as $S = S_1 \times S_2 \times\ldots\times S_J$, which is the Cartesian product of all S_j's. An arbitrary element of S is denoted by s. Let X_k be the nonempty set of all levels of control variable k for $k = 1,2,\ldots,K$. The control space X is the Cartesian product of all X_k's. An arbitrary element of X is denoted by x. For each state variable j, a random shock ε_j is distributed with a density function $f_j(\varepsilon_j)$. Let $\mathcal{E}_j$ be a set of all levels of random shocks. $\mathcal{E}$ is the Cartesian product of all $\mathcal{E}_j$'s. An arbitrary element of $\mathcal{E}$ is denoted by ε. This specification also characterizes deterministic state variables. The random shocks and the density function can accordingly be defined to represent a deterministic state variable.[3] Let μ be a function from $S \times X$ into $\mathbf{R}$, and let G be a function[4] from $S \times X \times \mathcal{E}$ into S. We assume that the spaces S, X, $\mathcal{E}$, and the functions μ, G, f_j, are for all t, that is, they are invariant to time.[5]

Decisions are made at the beginning of each period, $t = 0,1,\ldots,T$, where T is a finite positive integer. If $s_t \in S$ is the state at the beginning of period t and $x_t \in X$ is chosen in this state, then $\mu(s_t,x_t)$ is the one-period reward, and the state in the next period is given by $s_{t+1} = G(s_t,x_t,\varepsilon_{t+1})$, where $\varepsilon_{t+1} \in \mathcal{E}$ is a random shock realized in the next period. Let $u(s_{T+1})$ be the terminal reward function for time $T+1$, and $\delta \in (0,1]$ be the one-period discount factor. The decision maker's objective is to maximize expected sum of rewards:

$$\max_{\{x_t\}_{t=0}^{T}} E\left[\sum_{t=0}^{T} \delta^t \mu(s_t,x_t) + \delta^{T+1} u(s_{T+1})\right]$$

$$\text{subject to:}$$

$$s_{t+1} = G(s_t,x_t,\varepsilon_{t+1}),\, x_t \in X,\ \text{for } t = 0,1,2,\ldots,T,$$

$$s_0 \in S \text{ given.} \tag{1}$$

[3]For example, when the random shock is additive, for a state variable i, we can define $\varepsilon_i = 0$ to be the only element in $\mathcal{E}_i$, and $f_i(\varepsilon_i) = 1$ to represent the deterministic variable.

[4]The control space, X, may be defined as $X = \{x : s \in S, \varepsilon \in \mathcal{E}, \text{and } G(s,x,\varepsilon) \in S\}$. However, especially with stochastic state variables, the condition of $G(s,x,\varepsilon) \in S$ with multiple state variables may yield stringent restrictions on the control space, or the condition may not satisfy for some state variables.

[5]While the dynamic programming approach does not require this assumption, it is common in most applications.

In the case of all deterministic state variables, the decision maker chooses a sequence of controls $\{x_t^*\}_{t=0}^{T}$ in period 0. In the case with at least one stochastic state variable, the decision maker chooses x_0^* and makes contingency plans of x_t^* for periods $t = 1,2,\ldots,T$. These contingency plans depend on s_t, which will be known after the realization of the shocks in period t, ε_t.

2.2 Bellman's Equation

In this section, problem (1) is converted into Bellman's Equation. As the optimal policy for problem (1) is sought for each period $t = 1,2,\ldots,T$, this problem can be viewed as solving the following sequence of problems for each $t = 0,1,\ldots,T$:

$$\max_{\{x_\tau\}_{\tau=t}^{T}} E\left[\sum_{\tau=t}^{T} \delta^{\tau-t}\mu(s_\tau,x_\tau) + \delta^{T+1-t}u(s_{T+1})\right]$$
$$\text{subject to:}$$
$$s_{\tau+1} = G(s_\tau,x_\tau,\varepsilon_{\tau+1}),\, x_\tau \in X, \text{ for } \tau = t,t+1,\ldots,T,$$
$$s_t \in S \text{ given.} \tag{2}$$

Note that s_t can take any value from S, so we seek the solution of the above problem for all $s_t \in S$. The formulation (2) is the basis for solving this problem by converting it into Bellman's Equation, and is explained by Bellman (p. 83) as the Principle of Optimality:

> An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Define the value function for each $t = 0,1,\ldots,T$, for all $s \in S$:

$$V_t(s) \equiv \max_{\{x_\tau\}_{\tau=t}^{T}} E\left[\sum_{\tau=t}^{T} \delta^{\tau-t}\mu(s_\tau,x_\tau) + \delta^{T+1-t}u(s_{T+1}) \;\middle|\; s_t = s\right] \tag{3}$$

Note that we have omitted subscript t for the state s in the value function because the value function is defined for all s, not just the realized state in period t. From the above definition, it follows that the value function satisfies Bellman's equation:

$$V_t(s) = \max_{x} \left\{\mu(s,x) + \delta E\left[V_{t+1}(s_{t+1}) \mid s_t = s,\, x_t = x\right]\right\} \tag{4}$$

with the terminal (boundary) condition:

$$V_{T+1}(s) = u(s) \tag{5}$$

For finite horizon problems, the value function $V_t(s)$ and optimal solution $x_t^*(s)$ can be found by backward induction using (4) and (5). Given $V_{T+1}(s)$ for all $s \in S$, as defined by (5), we find $x_T^*(s)$ for all $s \in S$ by using (4) for $t = T$ and get $V_T(s)$ for all $s \in S$. Using $V_T(s)$ for all $s \in S$, we find $x_{T-1}^*(s)$ for all $s \in S$ by using (4) for $t = T-1$ and get $V_{T-1}(s)$ for all $s \in S$. This way we find $x_t^*(s)$ and $V_t(s)$ for all $s \in S$ for each $t = 0,1,\ldots,T$. The result of the dynamic programming approach is that the optimal solution of (1) can be obtained by solving Bellman's equation (4).

2.3 Terminal Optimization

A special case of sequence problem (1) is terminal optimization, where the objective function is the utility of state in the end of the planning horizon. Let $\mu(.) = 0$ for $t = 0,1,2,\ldots,T$, and $\delta = 1$, then problem (1) is terminal optimization:

$$\max_{\{x_t\}_{t=0}^T} E[u(s_{T+1})]$$
$$\text{subject to:}$$
$$s_{t+1} = G(s_t,x_t,\varepsilon_{t+1}), x_t \in X, \text{for } t = 0,1,2,\ldots,T,$$
$$s_0 \in S \text{ given.} \tag{6}$$

The corresponding Bellman's equation can be written as:

$$V_t(s) = \max_x \left\{E\left[V_{t+1}(s_{t+1}) \mid s_t = s, x_t = x\right]\right\} \tag{7}$$

with the terminal (boundary) condition:

$$V_{T+1}(s) = u(s), \tag{8}$$

where the value function for each $t = 0,1,\ldots,T$, for all $s \in S$, is defined as:

$$V_t(s) \equiv \max_{\{x_\tau\}_{\tau=t}^T} E[u(s_{T+1}) \mid s_t = s] \tag{9}$$

2.4 Numerical Solution Methods

Continuous-state dynamic programming problems lack closed-form solutions in most applications, with the exception of some very simple deterministic dynamic programming models. Thus, model solutions must be approximated by numerical methods. A variety of computational methods are available for numerically solving a dynamic programming model with continuous state and control variables. The choice of method depends on the assumptions for a given model. One special case of the dynamic programming model is the linear-quadratic problem, in which the state equations are assumed to be linear and the reward function to be quadratic. Optimal value and policy functions can be derived analytically in this case. The method for solving such problems, the linear-quadratic approach, is of limited value, however, since the necessary assumptions do not hold for many problems. Methods that do not make these assumptions include: (1) the discretization method and (2) the parametric approach. Under the discretization method, continuous state and control variables are discretized in finite sets, and the value function is solved and stored for the selected elements in the state space. When solving the problem requires the value function at points other than selected state set, its approximation is used. As mentioned in the introduction, discretization method becomes impractical for large-scale problems (Judd, 1998, pp. 430-33).

To overcome these limitations, Bellman and Dreyfus (p. 17) suggest using affine (linear) functions or a high degree polynomial to approximate the value function. This approach was developed as interpolation by piecewise polynomial splines. Bellman and Dreyfus (p. 323) also introduced polynomial approximation for the whole domain of the state variables, formally presented in Bellman, Kalaba, and Kotkin. Interpolation by polynomials or piecewise polynomials is an efficient and common parametric approach used in dynamic programming models. This parametric approach yields a value function defined over all values in the range of the state space. “The basic idea is that it should be better to approximate the continuous-value function with continuous functions and put no restrictions on the states and controls other than those mandated by the problem” (Judd, 1998, p. 433). Like the discretization method, the parametric approach requires solving the value function for a prespecified finite state set. Instead of storing the value function for each element in the state set, this approach fits a functional form. Miranda and Fackler refer to the interpolation approach in dynamic programming as the collocation method, and this term is also used in this study.

2.4.1 Collocation Method

The collocation method approximates a functional equation in such a way that the approximated function fits exactly at the prespecified points of the domain (Judd, 1998, p. 384). The collocation method is flexible, accurate, and numerically efficient for most applications in economics and finance (Miranda and Fackler).

The collocation method approximates a function with a linear combination of N basis functions using N prescribed points of the domain, called the collocation nodes, where N is a finite positive integer. For approximating the value functions $\{V_1(s), V_2(s), \ldots, V_T(s)\}$, the domain is the state space. We define a series of N basis functions $\{\phi_i(s)\}_{i=1}^{N}$ for $s \in S$.

The value function $V_t(s)$ is approximated by:

$$V_t(s) \approx \sum_{i=1}^{N} c_{it}\phi_i(s)$$

where the basis coefficients $\{c_{it}\}_{i=1}^{N}$ are to be determined. Value functions are approximated following backward induction using Bellman’s equation. Now Bellman’s equation (4) can be written as:

$$\sum_{i=1}^{N} c_{it}\phi_i(s) = \max_x \left\{ u(s,x) + \delta E\left[\sum_{i=1}^{N} c_{i,t+1}\phi_i(s_{t+1}) \mid s_t = s,\ x_t = x \right] \right\}$$

(10)

where $s_{t+1} = G(s_t, x_t, \varepsilon_{t+1})$. Note that the value function V_{T+1} is given in (5) and is used for approximating V_T. For all other t, we use the approximated V_{t+1}, that is, $\sum_{i=1}^{N} c_{i,t+1}\phi_i(s)$. By the collocation method, (10) is solved for each of N nodes. Let $\{s_n\}_{n=1}^{N}$ be a series of N nodes selected from the state space such that $s_n \in S$.[6] Let v_{nt} be the maximum value in the above problem for each node. Then we have for each node $n = 1,2,\ldots,N$:

$$\sum_{i=1}^{N} c_{it}\phi_i(s_n) = v_{nt}$$

(11)

This gives a system of N equations with the N unknown coefficients $\{c_i^t\}_{i=1}^{N}$. The N equations in (11) may be expressed as:

$$\Phi * c_t = v_t$$

where Φ is an $N \times N$ matrix in which the *n*th row and *i*th column is $\phi_{it}(s_n)$, c_t is a column vector denoting the coefficients $\{c_{it}\}_{i=1}^{N}$, and v_t is a column vector of $\{v_{nt}\}_{n=1}^{N}$ for all nodes. Then we can find c_t by:

$$c_t = \Phi^{-1} V^t$$

Note that we have N nodes as well as basis functions for all state variables, $N = N_1 * N_2 * \ldots * N_J$, where N_j is number of nodes and basis functions for each state variable $j = 1,2,\ldots,J$. In this case, s and Φ are tensor products for all state variables. Miranda and Fackler show that matrix Φ^{-1} can be formed efficiently by inverting it for each state variable and then making the tensor product. This method saves storage and computational effort especially when the problem is large.

2.4.2 Implementation of the Collocation Method

Implementing the collocation method to solve a dynamic programming problem requires specifications of: (i) the type of collocation basis functions and nodes, and the number of nodes, (ii) an optimization method for finding the maximum value, and (iii) a method for finding expected value, which are described here. In Section 5 we compare the performance of alternative schemes for these specifications.

[6] Note that we are using subscript *n* for the nodal index for explaining the collocation method.

2.4.2.1 Basis Functions and Nodes

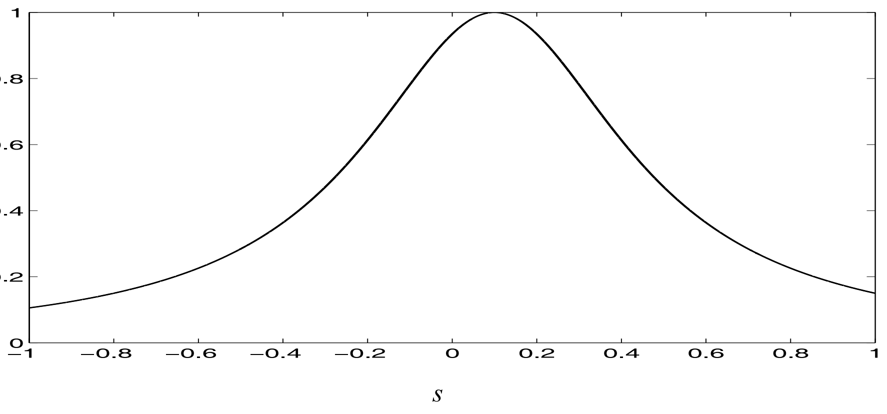
In implementing the collocation method, there are a number of choices available for collocation basis functions and nodes. Besides choosing the type of basis functions and nodes, the number of nodes must also be chosen. The Weierstrass Approximation Theorem asserts that every continuous function on a closed interval can be approximated uniformly to any prescribed accuracy by a polynomial (Schumaker, p. 91). However, Schumaker emphasizes that this theorem does not provide any guidance on the order of polynomials. Polynomial basis functions have long been used in approximating functions due to their properties of smoothness, differentiability, and efficiency in numerically implementation. Bellman and Dreyfus and Bellman, Kalaba, and Kotkin suggest using polynomial basis functions with orthogonality properties, such as Chebyshev and Legendre polynomials. Though polynomials have attractive properties, they may not perform well due to their oscillating behavior. In general the accuracy of polynomials increases as the order of polynomials is increased, but this does not hold for every function to be approximated. For example, when Runge's function, $\frac{1}{1+s^2}$, is approximated with Chebyshev polynomials and uniform nodes, the error grows as the order of polynomials is increased.[7] However, when Chebyshev nodes are used, instead of uniform nodes, the error decreases as the order of polynomials is increased.

The choice of basis functions and nodes depends on the characteristics of the function to be approximated. Approximation theory suggests that use of Chebyshev polynomial basis functions coupled with Chebyshev nodes may be a superior choice for smooth functions. However, if the approximated function is not smooth, use of spline basis functions coupled with uniform nodes may be a better choice (Miranda and Fackler). Approximation with spline basis functions is made piecewise and has narrow supports. Thus, it avoids the oscillating behavior that is common with polynomial interpolation (Santos, 1999, p. 338).

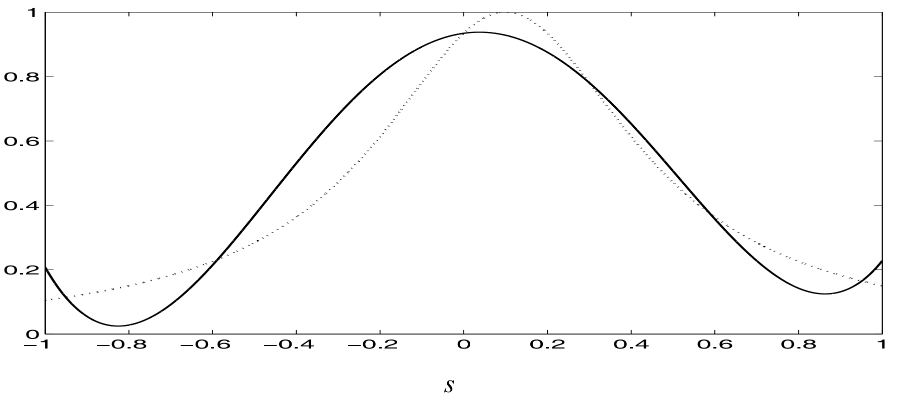
We explain here interpolation with Chebyshev basis functions and nodes and with spline basis functions and uniform nodes. These basis functions and nodes are also explained by Miranda and Fackler, by Gerald and Wheatly (Chapters 3 and 10), and by Santos (1999, p. 333). We illustrate these basis functions and nodes by using a type of Runge's function, $F(s) = \frac{1}{1+7(s-0.1)^2}$, where $s \in [-1, 1]$. Figure 1(a) plots the original function. In Figure 1(b), this function is approximated by Chebyshev polynomial basis functions coupled with Chebyshev nodes. In Figure 1(c), this function is approximated by linear spline basis functions coupled with uniform nodes. In both cases, the number of nodes is 5. In Figures 2(a) and (b), we use 9 nodes for approximation. For 9 nodes, Figure 2(c) shows locations of nodes for Chebyshev and uniform (i.e. equally spaced) nodes. Chebyshev nodes are more concentrated at the corners. Chebyshev nodes are appropriately located to avoid oscillation since Chebyshev interpolation is for the whole domain of the interval. However, in the spline interpolation, the function is approximated by dividing the domain in smaller intervals.

For both cases, the figures show that the accuracy of approximation improves as the number of nodes is increased from 5 to 9. When we compare Figures 2(a) and 2(b), we observe larger maximum absolute error in the spline interpolation than in the Chebyshev interpolation. This can be seen on these graphs at values of s around 0.1. However, the spline interpolation performs better than Chebyshev interpolation at other values in the domain. In Section 5, we examine accuracy of numerical solution of dynamic programming model. We compare the accuracy with different number of nodes, and between spline interpolation and Chebyshev interpolation.

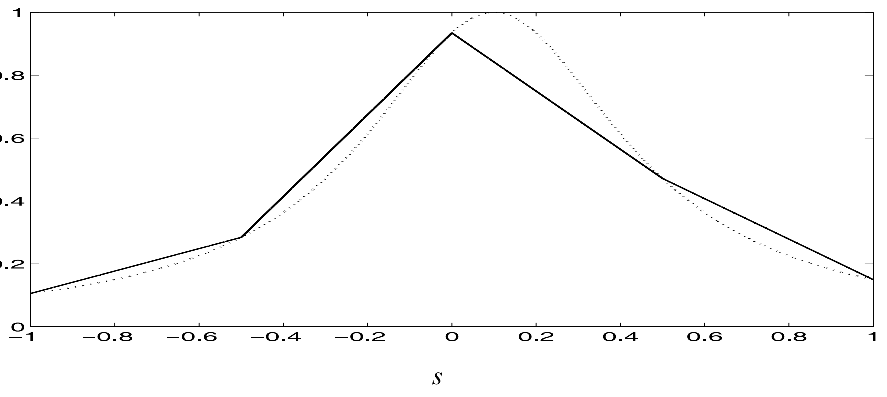
[7]The reader is referred to Schumaker (p. 101), Santos (1999, p. 335), and Miranda and Fackler for examples and their discussion.



(a) Original function: $F(s)=\dfrac{1}{1+7(s-0.1)^2}$

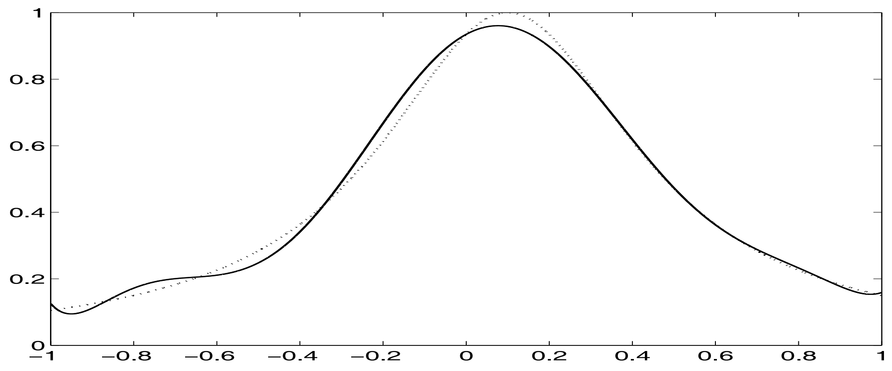


(b) Fitted with Chebyshev basis and nodes using 5 nodes

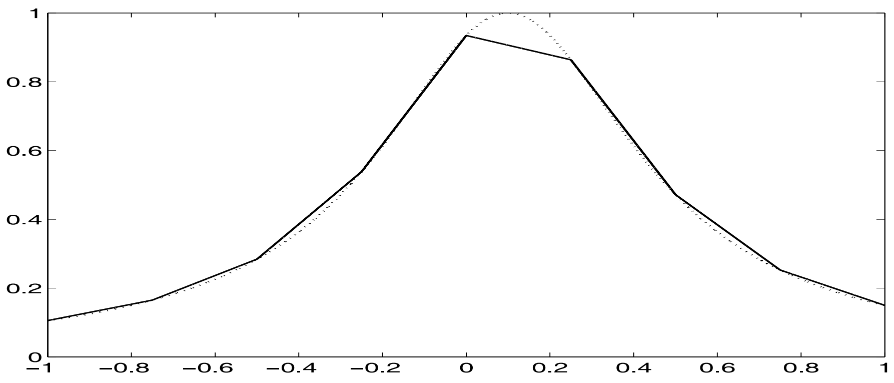


(c) Fitted with Linear spline basis and uniform nodes using 5 nodes

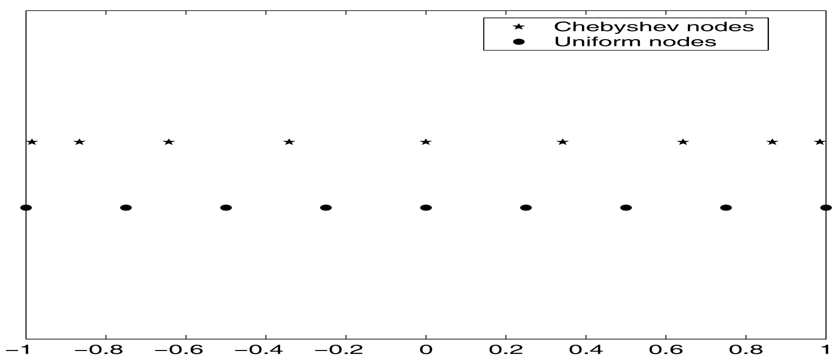
Figure 1: Approximation with 5 nodes



(a) Fitted with Chebyshev basis and nodes using 9 nodes



(b) Fitted with Linear spline basis and uniform nodes using 9 nodes



(c) Nodes

Figure 2: Approximation with 9 nodes

2.4.2.2 Optimization

When using collocation methods to approximate the value function, the optimal value function must be determined for each node using Bellman's equation. Miranda and Fackler and Judd (1998, p. 93) describe a variety of methods for numerical optimization. The golden-section search and Nelder-Mead methods are derivative-free methods for finding a local optimum. These methods work for any continuous, bounded function defined on a finite interval. The Newton-Raphson method is faster for finding a local optimum. However, this method requires a twice differentiable function. Furthermore, this method only identifies a critical point in the function and requires second derivative information to determine whether the critical point is a local maximum or a local minimum. Therefore, this method requires more search if we find, for example, a local minimum when we are searching a local maximum. Another problem with Newton-Raphson method is that it may not converge (Judd, 1998, p. 96). It is important to note that the above methods provide a local optimum, instead of the global optimum required for most problems.

The grid search method specifies a grid of points over an interval and finds the optimum from those points (Judd, 1998, p. 100). Though this method is slow, it is likely to give a near global optimum, since the grid is specified for the entire range of the interval. Use of the grid search method with the collocation method is referred to as hybrid method by Fackler and Miranda, because this method discretizes the control space while using continuous method for approximating the value function. The hybrid method is suitable for implementation in a matrix processing environment such as MATLAB or GAUSS (Fackler and Miranda). Discretization of the control space makes it possible to perform the optimization step for all elements of state set in a matrix. In the present study, we use this method for maximization in the dynamic programming model. We also introduce a two-stage grid search method, and compare accuracy of numerical solution of dynamic programming model between one-stage and two-stage grid search method in Section 5.

2.4.2.3 Integration

In Bellman's equation (10), the value function depends on the next period state, s_{t+1} . For each stochastic variable, the next period state is a function of a random shock, which generally has a continuous density function. In models with at least one stochastic state variable, the expected value function must be evaluated as part of dynamic programming procedure. Numerical integration is a practical approach for evaluating the expected value function.

Miranda and Fackler and Judd (1998, p. 251) describe a variety of methods available for numerical integration. The Gaussian quadrature method discretizes the continuous random variable to approximate the integral of a continuous density function. This method efficiently chooses these discrete points and their weights for approximating the integral (Judd, 1998, p. 257). A version of this method for normal random variables is called Gauss-Hermite quadrature, which is explained in Miranda and Fackler and Judd (1998, p. 261).

3. TEST PROBLEM AND IMPLEMENTATION

In order to improve accuracy and to minimize computational efforts for numerical solution, a dynamic programming problem should be specified so that it has the smallest possible number of state and control variables. Burt and Taylor provide a statistical procedure to reduce a state variable for use in a dynamic programming problem. Santos (1999, p. 350) uses solution of one-period optimization problem to reduce a control variable of dynamic programming problem. Also, normalization is a common procedure in economics to reduce the number of variables in the model. Using these techniques, this section specifies an investment decision problem and describes its implementation for numerical solution. This problem is used for measuring the accuracy and testing the alternative numerical methods.

3.1 Test Problem

3.1.1 Problem Specification

The investment decision problem has one control variable and four state variables, two of which are stochastic. In this problem, a farm manager's objective is to maximize the expected utility of net terminal wealth through implementation of an optimal policy for investment in risky farmland and a riskless nonfarm asset or debt financing on farmland in the presence of transaction costs, credit constraints, stochastic farmland prices and farm returns. This problem is specified as follows:

$$\max_{\{x_t\}_{t=0}^{T}} E_0[u(W_{T+1})]$$

$$\begin{aligned}
&\text{subject to :}\\
&\ln R_{t+1} = \beta_0 + \beta_1 \ln R_t + \varepsilon_{1,t+1},\\
&\ln P_{t+1} = a_0 + a_1 \ln P_t + a_2 \ln R_t + \varepsilon_{2,t+1},\\
&L_{t+1} = L_t + x_t,\\
&W_{t+1} = (1+r)[\{W_t - (P_t + \kappa - tc_s) * L_t\} - (P_t + \kappa + tc) * x_t - c * L_{t+1}]\\
&\qquad + R_{t+1} * L_{t+1} + (P_{t+1} + \kappa - tc_s) * L_{t+1},\\
&x_t \in X_t,\\
&\text{for } t = 0,1,2,\ldots,T, \text{ and } (R_0, P_0, L_0, W_0) \text{ are given,}
\end{aligned} \tag{12}$$

where

$$r = \begin{cases} r_b \text{ if}[\{W_t - (P_t + \kappa - tc_s) * L_t\} - (P_t + \kappa + tc) * x_t - c * L_{t+1}] > 0 \\ r_l \text{ otherwise,} \end{cases}$$

$$tc = \begin{cases} tc_p \ \text{ if } x_t > 0 \\ -tc_s \text{ otherwise,} \end{cases}$$

$0 < \rho < 1$, and E_0 is expectation operator over the random shocks $\varepsilon_{1,t+1}, \varepsilon_{2,t+1}$.

The above problem is an extension of models developed by Larson, Stauber, and Burt, and by Schnitkey, Taylor, and Barry. The utility function in problem (12) is specified as $u(W) = W$, for all W, which represents risk neutral preferences.[8] We assume a 20 -year planning horizon, where decisions are made at the beginning of each year $t = 0,1,\ldots,19$. The four state variables include: gross return per acre, R_t, farmland price per acre P_t, farmland acreage, L_t, and net wealth, W_t. The control variable is number of

[8]This utility function's strategically equivalent form is $u(W) \equiv \frac{W^{1-\theta}-1}{1-\theta}$ when $\theta = 0$.

acres of farmland to purchase or sell, x_t. Given in constraints 1 and 2 in problem (12), the gross return and farmland price per acre are stochastic state variables and are unaffected by the control variable x_t because this is a firm level decision model where returns and prices cannot be affected by an individual firm's action. At the beginning of each year t, the farm manager can purchase or sell farmland. The state equation of farmland is given in constraint 3 in problem (12).

The state equation of net wealth, W_t, is given in constraint 4 in problem (12). Net wealth is defined as the sum of liquid assets and the net sale-value of farm assets, which include farmland and the associated farm machinery and equipment. The price of machinery and equipment required per acre for farming land is denoted by κ. There is a transaction cost per acre of tc_p on purchasing land, and no transaction cost on purchasing machinery and equipment. There is a transaction cost on selling land and machinery and equipment, and the sum of these transaction costs per acre is denoted by tc_s. The purchase or sale decision for farmland is made in the beginning of each year t, so total land available for farming is L_{t+1}. The costs of production per acre are denoted by c, and the total costs of production are equal to $c * L_{t+1}$. From the definition of net wealth, the term $\{W_t - (P_t + \kappa - tc_s) * L_t\}$ in constraint 4 is liquid assets in the beginning of year t. Investment expenses, $(P_t + \kappa + tc) * x_t$, and production costs, $c * L_{t+1}$, are financed from liquid assets and determine net liquid assets. The interest rate is denoted by r. If net liquid assets are negative, then $r = r_b$ (borrowing rate). If net liquid assets are positive, then these assets are invested in a riskless investment, and $r = r_l$ (lending rate). Gross revenue is equal to $R_{t+1} * L_{t+1}$, and is added into liquid assets in the beginning of the next year. From the definition, net wealth in the beginning of the next year is the sum of liquid assets in the next year and the net sale-value of farm assets in the next year, which is last term in constraint 4.

There are four constraints on farmland purchase or sale decisions represented in constraint 5 in problem (12). These include the feasibility constraint, bankruptcy condition, the choice of exiting farming, and credit constraint. The feasibility constraint is $\underline{L} \leq L_t \leq \overline{L}$ or $L_t = 0$, where $\underline{L}$ denotes minimum acres required for farming and $\overline{L}$ denotes maximum feasible acres the manager can own in the location.[9] Substituting $L_{t+1} = L_t + x_t$ in the feasibility constraint for year $t + 1$ yields the following constraint on x_t:

$$\underline{L} - L_t \leq x_t \leq \overline{L} - L_t \text{ or } x_t = -L_t. \tag{18}$$

Under the bankruptcy condition, the farm is liquidated if net wealth is negative at any time:

$$x_t = -L_t \text{ if } W_t < 0 \tag{19}$$

[9]The bounds on state variables are also required for numerically solving a dynamic programming problem.

Note that when $W_t \geq 0$, the manager can potentially choose to sell all land. We assume that once all land is sold, however, the business cannot re-enter farming:

$$\text{if } x_t = -L_t, \text{ then } x_{t+1} = x_{t+2} = \ldots = x_T = 0 \tag{20}$$

Another constraint is on loans provided by the lender. This constraint allows purchase of land and the associated machinery and equipment as long as the debt to asset ratio is less than or equal to ρ, where $0 < \rho < 1$. Farmland price is stochastic, so it is possible that the debt to asset ratio can be above ρ. We assume that the lender does not require to sell land in order to maintain this ratio. Therefore, the credit constraint on x_t is:

$$x_t \leq \max\left(0, \frac{W_t - (1-\rho) * (P_t + \kappa - tc_s) * L_t}{(P_t + \kappa + tc_p) - \rho * (P_t + \kappa - tc_s)}\right) \tag{21}$$

For notational convenience we denote X_t as the set of all levels of control variables that satisfy (18)-(20), which are represented in constraint 5 in problem (12). Though the control variable x_t is number of acres of farmland purchased or sold, it implicitly also determines the amount of either debt financing or investment in the riskless asset, as explained above.

3.1.2 Parameter Estimation

Numerically solving the above investment problem by dynamic programming requires estimates of parameters of the stochastic state equations and other parameters of the model. Constraints 1 and 2 in problem (12) are estimated using time series data for Southwestern Minnesota. Gross return per acre is calculated for the 50-50 corn-soybean mix using data for farms that belong to the Southwestern Minnesota Farm Business Management Association (SWMFBMA), as published in their Annual Reports for the years 1967-99 (Olson et al.). Time series data for the price of farmland are obtained for Southwestern Minnesota for years 1966-1992 from the Minnesota Rural Real Estate Market (Schwab and Raup; Brekke, Tao, and Raup), and for years 1990-1999 from Minnesota Land Economics (Taff). After adjusting data for inflation using the Consumer Price Index, parameters for these equations are estimated by Ordinary Least Square (OLS) method following the Box and Jenkins approach for forecasting. For this approach, the Ljung-Box test is used for testing tat error term is a white noise process. In addition, we tested for the normality of error term using the Bera-Jarque test.

Estimates for the gross return equation (constraint 1 in problem (12)) are $\beta_0 = 1.052028$, $\beta_1 = 0.82197$, and error term $\varepsilon_1 \sim N(0, 0.033155)$, denoting a normal distribution with its mean and variance. Gross return equation is a first order autoregressive model, which has one lag and requires one state variable for t. However, estimates of the farmland price equation turned out to have two lags of farmland price and one lag of gross returns. Including two lags of farmland price requires an additional state variable in the model. Using the Burt and Taylor approach, we reduce a state variable in the farmland price equation and represent the process with one lag of farmland price and one lag of gross return as given in (constraint 2 in problem (12)). Following this procedure, we have estimates of parameters as $a_0 = 0.048655$, $a_1 = 0.884465$, $a_2 = 0.134044$, and error term $\varepsilon_2 \sim N(0, 0.014619)$ for constraint 2 in problem (12).

Estimates of other parameters are also represented in constant dollars. The interest rate on borrowing is assumed to be $r_b = 0.06$, which is based on reviewing last five-year interest rate on long-term loans in farm credit system after adjusting it for inflation (USDA). On lending (riskless investment), the interest rate is assumed to be $r_l = 0.03$. Based on farm data for Southwestern Minnesota (Olson et al.) and an interview with Dale Nordquist[10], we have following estimates: $\kappa = \$300$, $tc_p = 0.01 * P_t$, $tc_s = 0.06 * P_t + 0.07 * \kappa$, $\rho = 0.7$, and $c = \$247$.

3.1.3 Dynamic Programming Formulation

Bellman's equation for solving the above investment problem can be written as:

$$\begin{aligned} V_t(R,P,L,W) = \max_x \{E[V_{t+1}(R_{t+1},P_{t+1},L_{t+1},W_{t+1}) \mid R_t = R, P_t = P, \\ L_t = L, W_t = W, x_t = x]\} \end{aligned} \tag{22}$$

subject to constraints 1-5 in problem (12) with the boundary condition:

$$V_{T+1}(R,P,L,W) = u(W) \tag{23}$$

where $V_t(R,P,L,W)$ is the value function defined for each $t = 0,1,\ldots,T$, as:

$$V_t(R,P,L,W) \equiv \max_{\{x_\tau\}_{\tau=t}^{T}} E[u(W_{T+1}) \mid R_t = R, P_t = P, L_t = L, W_t = W] \tag{24}$$

3.2 Model Implementation

Implementation of the dynamic programming approach to solve the above problem requires specification state space. The value function for this problem is solved for the following ranges of the states: $220 \leq R_t \leq 620$, $950 \leq P_t \leq 2{,}215$, $400 \leq L_t \leq 2{,}000$, and $0 \leq W_t \leq 6{,}000{,}000$. The ranges for R_t and P_t are based on a 90 percent region for their distributions, estimated with their equations. The range for farmland is chosen for fully owned farms. The lower bound of net wealth is due to the bankruptcy condition ($W_t < 0$), thus $W_t \geq 0$ is the condition for staying in farming. The upper bound of net wealth is arbitrarily chosen to allow no debt at the upper bounds of farmland acreage and its price.

Given the current state from the above state space, the next period state may go out of bounds. In backward recursion, we have the value function for the above state space, however, it needs to be computed when the state in period $t + 1$ is not in the above the ranges.

[10] Department of Applied Economics, University of Minnesota, St. Paul, Minnesota.

The control variable, x_t, is restricted so that L_{t+1} satisfies the feasibility constraint: $400 \leq L_{t+1} \leq 2,000$ or $L_{t+1} = 0$. Given the range of farmland, the value function needs to be computed for $L_{t+1} = 0$, when all farmland is sold. As assumed in the problem, once all farmland is sold, the business cannot re-enter farming. When there is no farmland, there would be no debt financing, since it is based on farmland as collateral. Thus, W_{t+1} is all invested in liquid assets. First, W_{T+1} is computed by compounding W_{t+1} at the liquid asset return rate, and then the utility function is evaluated as $u(W_{T+1})$. Since the liquid asset is a riskless investment, $u(W_{T+1})$ is the value function for $L_{t+1} = 0$, as defined in (24).

W_{t+1} can go out of the bounds from the above range due to the stochastic state variables. Unlike the state variable L_{t+1}, it is generally not possible to use restrictions on the control variable to ensure that W_{t+1} will be in the allowable range.[11] When $W_{t+1} < 0$, the value function is computed for the bankruptcy condition. Under this condition, the farm is liquidated if net wealth is negative $(W_{t+1} < 0)$ at any time, which makes $L_{t+1} = 0$. Thus, the value function can be computed for $L_{t+1} = 0$ as described in the above paragraph. However, the compounding is done at the debt rate because $W_{t+1} < 0$. When W_{t+1} is greater than the upper bound, the extra amount above the upper bound earns the liquid asset return rate and is compounded for period $T+1$ to compute the value function. Note that compounding of net wealth for $W_t = 0$ yields $W_{T+1} = 0$. In order to maintain continuity of the value function, the value function for zero current net wealth, $W_t = 0$, is set as $u(W_{T+1}) = u(0)$.

For the return from crops, to assure that the state in period $t+1$ is within the bounds, we assume $R_{t+1} = \max(\underline{R}, \min(g(R_t, \varepsilon_{1,t+1}), \overline{R}))$, where $\underline{R}$ is the lower bound, $\overline{R}$ is the upper bound, and $g(R_t, \varepsilon_{1,t+1})$ denotes the right hand side in the state equation of gross return per acre. This assumption assigns to the bounds the probability mass for the states beyond the bounds. Similarly, we make this assumption for the farmland price state variable to assure that P_{t+1} is within its bounds.[12]

4. ACCURACY MEASUREMENT

In this section, we develop a method for measuring accuracy of collocation methods for numerically solving a dynamic programming model. This method is illustrated by using the investment problem described in the preceding section, but it can be used for any finite planning horizon continuous-state dynamic programming model. Though this method is developed for checking accuracy of the collocation

[11] For example, $W_t = 6000000$, $P_t = 1000,\quad L_t = 1000$, and $x_t > -L_t$. If in the next period, $P_{t+1} = 1500$ and $R_{t+1} = 250$ with some probability, the net wealth will be above 6000000, and cannot be put into the bounds with the control variable.

[12] Another way to solve this problem, as proposed by Miranda and Fackler, is to widen the ranges of the states for given minimum and maximum error term from numerical integration. However, applying this method for these equations gives very wide bounds, which are well outside the range of value found in the data.

method, the approach can also be applied for checking the accuracy of other numerical methods such as discretization.

Suppose we have chosen the type of collocation basis functions and nodes, and the number of nodes, an optimization method for finding the maximum value, and a method for finding expected value, as described in Section 3. In the investment problem, there are four state variables, and let $N = \Pi_{j=1}^{4} N_j$ be the total nodes where N_j are nodes for state $j = 1,2,3,4$. Following the dynamic programming procedure, we approximate the value function by $V_t(R,P,L,W) \approx \sum_{i=1}^{N} c_{it}\phi_i(R,P,L,W)$ for each $t = 0,1,\ldots,19$, as we have assumed a 20-year planning horizon. Having approximated the value function, the optimal policy x_t^* for any state, R,P,L,W, can be found by solving:

$$\max_x \{E[\sum_{i=1}^{N} c_{i,t+1}\phi_i(R_{t+1},P_{t+1},L_{t+1},W_{t+1}) \mid R_t = R, P_t = P, L_t = L, W_t = W]\} \tag{25}$$

subject to the constraints 1-5 in the investment problem (12). Suppose we have a initial state R_0,P_0,L_0,W_0. For this initial state at $t = 0$ we solve the above problem (25) using the approximated value function. Solving this problem gives the optimal policy x_0^* and the maximum value of the objective function at that policy, $\tilde{V}$, for this initial state. We refer to $\tilde{V}$ as the estimated value function, since it is based on estimation by the collocation method.

The dynamic programming approach ensures that the optimal policy and its objective function take future decisions into account. By definition of the value function of this problem, $\tilde{V}$ is the expected utility of net terminal wealth for this initial state. To test the accuracy of the solution, we simulate the objective function by the Monte Carlo method. For this initial state in $t = 0$, we apply the optimal policy x_0^* derived using (25). Given this optimal policy, the state in next period depends on the random shocks. In the model we have error terms in each of two stochastic state variable equations: gross return per acre, R_t, and farmland price per acre, P_t. These error terms have normal density functions and their parameters are estimated in Section 3.1.2. We draw a 20-year set of the error terms from these density functions. The optimal policy x_0^* and the error term in the next period determine the state in period t = 1. Then we apply the optimal policy for period t = 1, realize shocks in that period, and move on to period t = 2. Following this procedure to the end of the planning horizon gives a path of states from the initial state to the end of planning horizon state. Of particular interest is the state in period T+1, $R_{T+1},P_{T+1},L_{T+1},W_{T+1}$, which is the basis for computing $U(W_{T+1})$.

To compute $E[U(W_{T+1})]$, we draw 500 trails of error terms, and follow the above procedure for each of 500 trials. Since there are 20 years, these will be $500 * 20 = 10,000$ sets of error terms for each stochastic variable: R_t and P_t. From the above procedure, now we have 500 paths of states and 500 values of $U(W_{T+1})$. The average of these values is the expected utility of net terminal wealth, $E[U(W_{T+1})]$. We refer to this as the expected simulated value function, denoted by $E\left[\mathrm{V}_{\mathrm{sim}}\right]$.

For the accuracy analysis here, we choose 81 different initial state levels in period $t = 0$ from the state space. Given the ranges of the state variables in the problem, we choose $R_0 = [320, 420, 520]$, $P_0 = [1265, 1580, 1900]$, $L_0 = [800, 1200, 1600]$, $W_0 = [1500000, 3000000, 4500000]$, from which we make a grid of 81 states by their Cartesian product. For each of the 81 initial state levels, we follow the above procedure and compute the expected simulated value function, $E\left[\mathrm{V}_{\mathrm{sim}}\right]$, and the estimated value function, $\widetilde{V}$, which are used for accuracy analysis.

Two approaches are proposed here for accuracy analysis. In the first approach, we analyze $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ computed for the 81 initial state levels. The average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ is calculated as the mean for 81 initial states. We compare alternative collocation schemes by determining which method gives a higher $E\left[\mathrm{V}_{\mathrm{sim}}\right]$, since the objective function is to be maximized. This comparison provides the gain in the objective function from one scheme to another. This first approach only compares alternative collocation schemes of by measuring the gain in the objective function. It does not measure the error in the approximated value function.

The second approach of accuracy analysis measures the absolute error in approximating the value function, and is also useful for comparing the alternative collocation methods. For each initial state level, the absolute error in the value function is calculated by dividing the difference between the expected simulated value function and the estimated value function by the expected simulated value function, then taking the absolute value:

$$\text{Absolute Error} = \left| \frac{E\left[\mathrm{V}_{\mathrm{sim}}\right] - \widetilde{\mathrm{V}}}{E\left[\mathrm{V}_{\mathrm{sim}}\right]} \right|$$

The absolute error is calculated for each of 81 initial states. The maximum absolute error is calculated as the maximum of the absolute error from these 81 initial states. The average absolute error is calculated as the mean for 81 initial states. Computation of errors in this way provides information on the error in the value function associated with a collocation method.
For comparing alternative collocation schemes, we can also compare the error associated with each scheme. In our experiments, we will present results for both approaches. We expect that the smaller the absolute error in the value function, the higher the value of $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ will be in the objective function.

5. ACCURACY RESULTS

Using the accuracy measurement methods developed in Section 4, we examine here the accuracy of alternative schemes for implementing the collocation method for solving a dynamic programming model. These experiments are conducted using the investment problem introduced in Section 3.

First we compare the accuracy for Chebyshev interpolation and spline interpolation, and we explore the effect of increasing the number of nodes. Chebyshev nodes are not evenly spaced, and this scheme does not assign the nodes in the corners of the range for interpolation. The function approximated by interpolation may give poor approximation for extrapolation. To avoid extrapolation, we increase the range of each state variable for Chebyshev nodes, so that the first and the last nodes are the lower and upper bound of the state range respectively. Uniform nodes are evenly spaced and include the corners of the range for interpolation. Spline interpolation gives good approximation with an odd number of nodes (Schumaker). Thus, we choose odd numbers, 3,5,7,9,..., for the number of nodes. The grid search method is used for maximization in the dynamic programming model. We discretize the control space, X, in 81 levels to find the optimal policy.

Next we examine the efficiency of the grid search method. We introduce a two-stage grid search method and compare accuracy of numerical solution of dynamic programming model between one-stage and two-stage grid search method. Finally, we examine accuracy when the node configuration is allowed to differ among state variables rather than keeping an equal number of nodes in each dimension.

In all cases, we use the Gaussian quadrature method of numerical integration for taking the expected value. In the model, we have two stochastic state variables, R_t and P_t. For each state we use 5 nodes for the numerical integration. This gives 25 combinations with their probabilities.

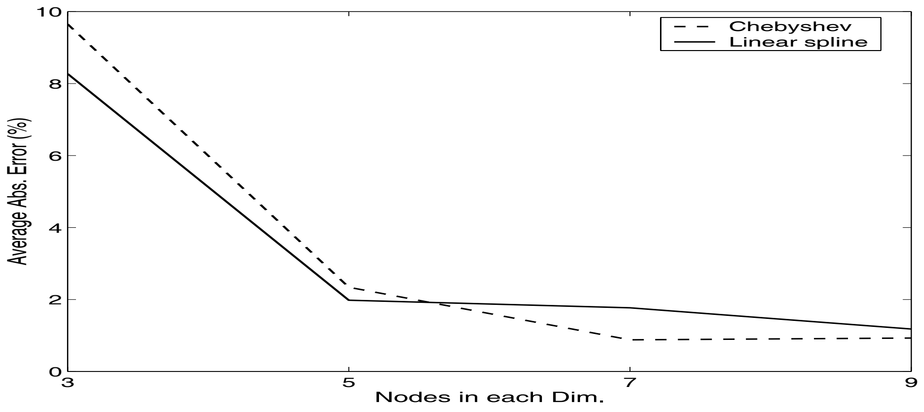
The dynamic programming model is programmed in MATLAB. For evaluating the basis functions and nodes for the collocation methods, and for numerical integration, we use MATLAB code developed by Miranda and Fackler. The model was run using MATLAB on a Dell PC with 512 MB RAM and an 800 MHz processor. The CPU time is measured in seconds.

5.1 Chebyshev and Linear Spline

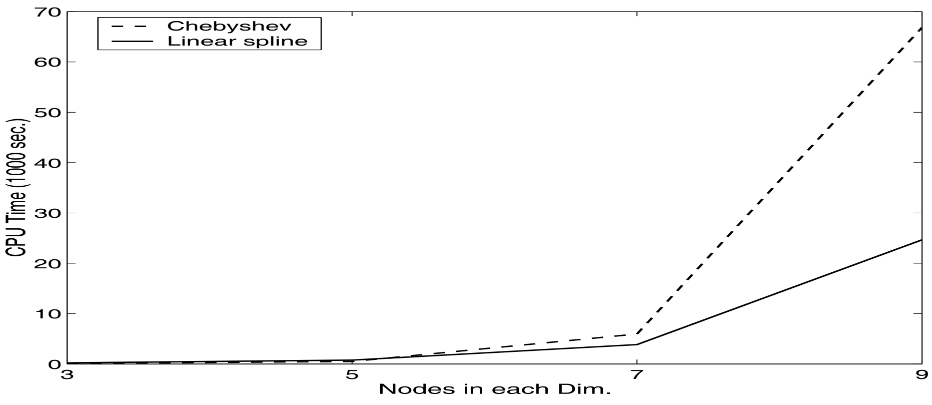
In this section we compare accuracy for (i) Chebyshev basis function and nodes and (ii) linear spline basis function and uniform nodes. Table 1 presents the CPU time and absolute error for the collocation with Chebyshev and linear spline interpolation methods. There are four state variables in the model, R_t, P_t, L_t, W_t, so the value function has four dimensions. The first column of Table 1 indicates the number of nodes in each dimension. The results show that as the number of nodes increases, average and maximum absolute errors decrease with linear spline collocation and Chebyshev collocation in almost all cases. The only exception is a slightly higher average absolute error with 9 nodes as compared 7 nodes under Chebyshev collocation. The marginal improvement in accuracy from increasing the number of nodes declines as the number of nodes increases, as is shown in Figure 3(a). However, CPU time increases exponentially with increases in the number of nodes, as shown in Figure 3(b). This is because the model is multidimensional. For example, with 3 nodes in each dimension, the total number of nodes is $3^4 = 81$, and with 9 nodes in each dimension, the total number of nodes is $9^4 = 6561$.

Table 1: CPU Time and Absolute Error in Value Function

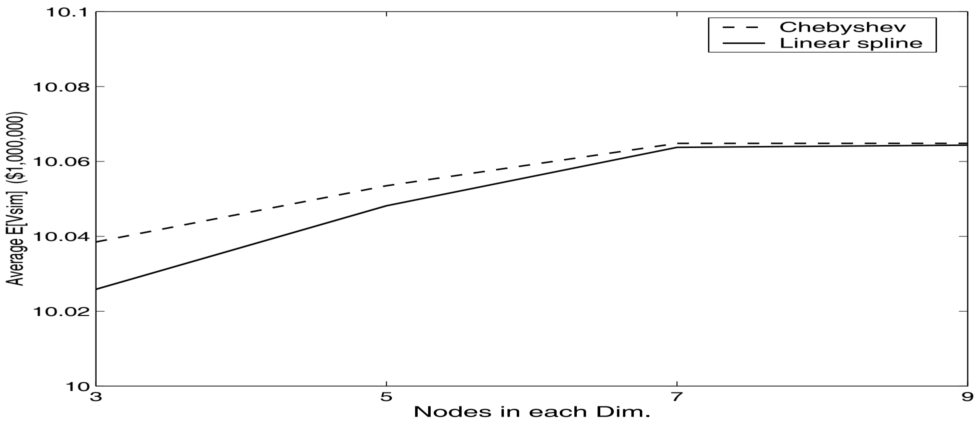
	Chebyshev			Linear Spline		
Nodes in each Dim.	CPU Time	Average Absolute Error	Maximum Absolute Error	CPU Time	Average Absolute Error	Maximum Absolute Error
	(Seconds)	(%)	(%)	(Seconds)	(%)	(%)
3	105	9.65	18.68	219	8.27	26.50
5	500	2.34	4.20	766	1.98	7.38
7	5,959	0.88	2.32	3,858	1.77	5.25
9	66,845	0.93	2.27	24,665	1.18	3.42



(a)



(b)



(c)

Figure 3: (a) Average absolute error, (b) Computation time, (c) Average simulated value function

Table 2 presents the average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ for different numbers of nodes under the Chebyshev and linear spline methods. As the number of nodes increases, the average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ also increases. Again, the marginal improvement from increasing the number of nodes declines as the number of nodes increases, as shown in Figure 3(c). The maximum gain in the objective function due to increasing nodes is less than 0.5 percent on average. For the Chebyshev interpolation, the average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ with 3 nodes is only 0.26150 percent less than the average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ with 9 nodes. We also examine the probability distribution of performance using 81 observations of $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ for 81 initial states. The frequency of lower $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ with 3 nodes as compared to $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ with 9 nodes is 92.59 percent for Chebyshev interpolation. For the linear spline approach, this frequency is 98.77. The frequency estimate indicates that the probability of poorer performance with 3 nodes than with 9 nodes is very high.

Table 2: Expected Simulated Value Function

Nodes in each Dim.	**Chebyshev**			**Linear Spline**		
	Average	Comparison with 9 Nodes		Average	Comparison with 9 Nodes	
	$E\left[\mathrm{V}_{\mathrm{sim}}\right]$ (\$$10^6$)	% Decrease in Average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$	% Frequency $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ decreased	$E\left[\mathrm{V}_{\mathrm{sim}}\right]$ (\$$10^6$)	% Decrease in Average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$	% Frequency $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ decreased
3	10.038	0.26150	92.59	10.026	0.3823	98.77
5	10.054	0.11240	90.12	10.048	0.1609	82.72
7	10.065	0.00004	38.27	10.064	0.0058	53.09
9	10.065	- - -	- - -	10.064	- - -	- - -

The results in Table 1 show that the linear spline collocation scheme has higher maximum absolute errors than Chebyshev collocation with each number of nodes. However, the results for average absolute error are mixed and values are quite similar for the two schemes. Table 3 provides a summary comparison of the Chebyshev and linear spline collocation schemes. As noted above, with increase in the number of nodes, the CPU time increases exponentially. However, linear spline collocation takes much less time than Chebyshev collocation with 7 and 9 nodes.[13] CPU time under Chebyshev collocation is 2.7 times the CPU time under linear spline collocation for 9 nodes. For this additional cost of CPU time, the gain in $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ under the Chebyshev method is only 0.005 percent. From Table 1, we observe that with 9

[13]Miranda and Fackler indicate that this is due to the use of sparse function in MATLAB.

nodes Chebyshev collocation has an average absolute error of 0.93 percent, while the average absolute error is 1.18 percent under linear spline collocation. Taking into account both CPU time and average absolute error, linear spline collocation may be a better choice than Chebyshev collocation for 9 nodes. The fourth and fifth columns of Table 3 show the probability that $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ will be higher under Chebyshev methods than under linear spline methods and vice versa. These results indicate no clear-cut superiority for either Chebyshev or linear spline interpolation.

Table 3: Comparison of Chebyshev and Linear Spline

Nodes in Each Dim.	Ratio of CPU Time: Chebyshev/Spline	Ratio of Average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$: Chebyshev/Spline	Comparison of % Frequency $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ increased with	
			Chebyshev vs. Spline	Spline vs. Chebyshev
3	0.48	1.00126	62.96	37.04
5	0.65	1.00053	37.04	62.96
7	1.54	1.00011	43.21	56.79
9	2.71	1.00005	59.26	40.74

Finally, we examined the policy function for the investment problem approximated with Chebyshev interpolation and linear spline interpolation. The solution of the model with Chebyshev collocation indicates some unexpected behavior of the policy function. This indicates that the approximated function for this problem may not be smooth, and thus, spline interpolation may be a better choice than Chebyshev interpolation, as suggested by Miranda and Fackler. Further accuracy analysis is performed for the linear spline interpolation.

The above accuracy results show that as the linear spline interpolation may indicate higher maximum absolute error than Chebyshev interpolation. However, for non-smooth functions, spline interpolation performs better than Chebyshev interpolation to represent the solution to the problem.

5.2 Optimization Procedure

For the accuracy analysis presented above, we discretized the control space, X, in 81 levels to find the optimal policy. Here we introduce a two-stage grid search method. By this method, we first discretize the control space, X, in 41 levels to find the optimal policy. For the interval $[400, 2000]$ of farmland acreage state variable, L_t, this discretizes the farmland purchase/sale control variable, x_t, with a 40 -acre increment. Given the optimal policy from the first stage, we find a new control space for the second stage optimization. For the second stage the lower bound is the optimal policy minus 40 acres, and the upper bound is the optimal policy plus 40 acres. This new control space is determined such that it satisfies the constraints of the model. This control space is again discretized into 21 levels to find the optimal policy. Here we examine the accuracy of the two-stage grid search method as compared to one-stage grid search method with linear spline basis functions and uniform nodes.

Table 4 presents the CPU time, average absolute error, and the average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ for the two-stage optimization procedure with the linear spline interpolation. As before, the average and maximum absolute errors decrease and $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ increases as the number of nodes increases. Comparison between the one-stage and the two-stage optimization methods is presented in Table 5. The table shows that the CPU time under the two-stage method is only about 77 percent of the CPU time under the one-stage method. The average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ for the two-stage method is slightly higher than that for the one-stage method, as their ratio is greater than one, but the difference is very small. Furthermore, the frequency analysis indicates a higher probability of better performance with the two-stage method than that with the one-stage method in all cases. The average absolute error for both methods is identical, which can be noted by comparing Table 4 and Table 1 for the linear spline case. These results show that the two-stage method is more efficient, as it takes less CPU time and performs slightly better than one-stage method.

Table 4: Two-stage Optimization with Linear Spline

Nodes in each Dim.	**CPU Time**	**Average Absolute Error**	**Maximum Absolute Error**	**Average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$**
	(Seconds)	(%)	(%)	($\$10^6$)
3	165	8.27	26.5	10.026
5	584	1.98	7.38	10.048
7	3,006	1.77	5.24	10.064
9	18,946	1.18	3.42	10.064

Table 5: Comparison of Two-stage and One-stage Optimization with Linear Spline

Nodes in Each Dim.	**Ratio of CPU Time: 2-stage/1-stage**	**Ratio of Average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ 2-stage/1-stage**	**Comparison of % Frequency E [V $_{\mathrm{sim}}$] increased**	
			2-stage vs. 1-stage	1-stage vs. 2-stage
3	0.76	1.00001	70.07	29.93
5	0.76	1.00001	60.49	39.51
7	0.78	1.00001	71.60	28.40
9	0.77	1.00001	64.20	35.80

5.3 Node Configuration

The accuracy analysis presented here shows that accuracy increases with an increase in the number of nodes, but this comes at a cost of increased CPU time. Despite advances in computer technology, solving a large-scale multidimensional dynamic programming model can only be done by limiting the number of nodes. The investment problem used in this study has four state variables (R_t, P_t, L_t, W_t) requiring specification of the number of nodes for each state variable. We note that CPU time increases exponentially when we increase number of nodes in each dimension. Since this is a multidimensional approximation, it is possible that some state variables may need more nodes while other state variables may need fewer nodes. Here we perform an accuracy analysis for alternative node configurations using two-stage optimization with linear spline collocation.

All of our results show that the model performs poorly with 3 nodes. Thus, for node configurations we consider a minimum of 5 nodes for any dimension. For initial analysis, we first increase nodes from this minimum level in one dimension at a time rather than increasing nodes in all dimensions. Table 6 shows results with number of nodes for each state variable ordered by (R_t, P_t, L_t, W_t) in the first column.

Table 6: Increasing Nodes in One Dimension at a time

Nodes in each Dim.	**CPU Time**	**Average Abs. Error**	**Maximum Abs. Error**	**Average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$**
	(Seconds)	(%)	(%)	($\$10^6$)
9,5,5,5	1,123	2.32	8.32	10.052
5,9,5,5	1,119	2.37	8.28	10.053
5,5,9,5	1,089	1.98	7.33	10.049
5,5,5,9	1,115	0.60	1.64	10.062

Average and maximum absolute errors are lowest with node configuration 5,5,5,9, indicating that accuracy improves the most when we increase the number of nodes for the state variable net wealth, W_t. Also, the average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ is highest, that is $10.062 * 10^6$, in this case. Therefore, the marginal benefit of increasing nodes is highest for W_t.

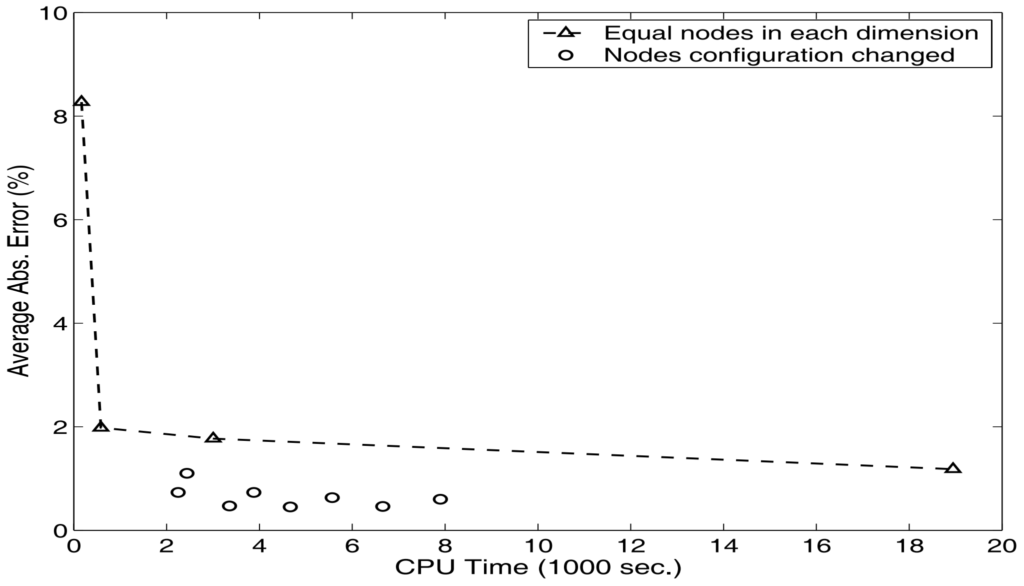
Given these results, we start with 9 nodes in W_t and 5 nodes in the other state variables (5,5,5,9), where the average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ is $10.062 * 10^6$. Now we experiment of increasing the number of nodes for R_t, P_t, L_t. When we increase the nodes of L_t from 5 to 9, (5,5,9,9), the average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ remains unchanged, i.e. $10.062 * 10^6$. However, when we increase the nodes of each R_t, (9,5,5,9), and P_t, (5,9,5,9), the average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ improves from $10.062 * 10^6$ to $10.065 * 10^6$. Also, the average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ is $10.065 * 10^6$ for (7,7,5,9). It is important to note that the total number of nodes is equal to the product of number of nodes for each state variable $(N = N_1 * N_2 * N_3 * N_4)$. The CPU time increases as N increases. For node configuration (7,7,5,9), $N = 2{,}205$. If we increase two nodes of L_t, (7,7,7,9), $N = 3{,}087$. If we increase two nodes of W_t, (7,7,5,11), $N = 2{,}695$, which is much less than $3{,}087$. Thus, increasing nodes of W_t would have a small marginal cost of CPU time and a greater marginal benefit of accuracy performance.

Table 7 presents accuracy results on selected nodes configurations. The table presents the CPU time, absolute error, and the average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ with two stage optimization and the linear spline method for different number of nodes for W_t. In this case also, as we increase the number of nodes, the average and maximum absolute errors decrease and average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ gets slightly higher. The results are also compared with increasing number of nodes in every dimension in Figure 4. Figure 4(a) plots the average absolute error as a function of time. The figure shows that changing the node configuration gives a smaller average absolute error for a given CPU time. Similarly, Figure 4(b) shows a higher average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ with changed node configuration for a given CPU time.

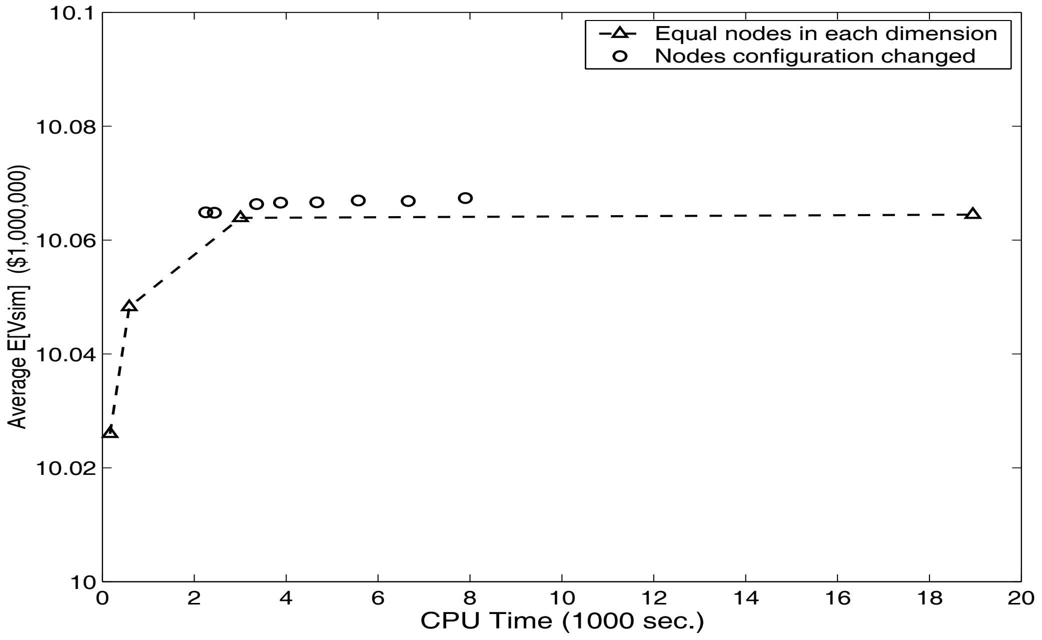
From Table 7, we have results with nodes $(5,9,5,17)$, where average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ is $10.067 * 10^6$, the maximum absolute error is 1.24 percent, and the average absolute error is 0.45 percent. Without changing the node configuration, we have results for $(9,9,9,9)$ in Table 4, where the average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$ is $10.064 * 10^6$, the maximum absolute error is 3.42 percent, and the average absolute error is 1.18 percent. Thus, changing the node configuration yields a more accurate approximation by each criterion. Furthermore, changing the node configuration improves computational efficiency, since the CPU time is $4{,}666$ seconds with nodes $(5,9,5,17)$, which is much less than with nodes $(9,9,9,9)$, where CPU time is $18{,}946$ seconds. These results show that a change in node configuration can improve the performance of approximation methods by reducing both the size of errors and CPU time. For this multidimensional function, some state variables may require only few nodes while others may require more nodes to yield a desired accuracy level.

Table 7: Nodes Configuration with Two-stage Linear Spline

Nodes in each Dim.	**CPU Time**	**Average Abs. Error**	**Maximum Abs. Error**	**Average $E\left[\mathrm{V}_{\mathrm{sim}}\right]$**
	(Seconds)	(%)	(%)	($\$10^6$)
7,7,5,9	2,435	1.10	3.28	10.065
7,7,5,13	3,877	0.73	2.28	10.067
7,7,5,17	5,568	0.63	1.84	10.067
7,7,5,21	7,899	0.60	1.70	10.067
5,9,5,9	2,251	0.73	2.45	10.065
5,9,5,13	3,355	0.47	1.45	10.066
5,9,5,17	4,666	0.45	1.24	10.067
5,9,5,21	6,655	0.46	1.48	10.067



(a)



(b)

Figure 4: Accuracy and CPU time with different nodes configuration

6. SUMMARY AND CONCLUSIONS

Dynamic programming problems with continuous state and control variables lack closed-form solution in most applications. Therefore, they must be solved approximately using numerical methods. Since the solution is approximated, it is useful to examine the accuracy of the numerical solution to the model. We develop a method for measuring the accuracy of the numerical solution for a stochastic dynamic programming model. Using this method, we compare the accuracy of various schemes of the collocation method using an investment decision problem.

The results show that the accuracy improves as number of nodes increases, but the marginal advantage of increasing the number of nodes declines as the number of nodes increases. Since this is a multidimensional problem, CPU time increases exponentially with increases in the number of nodes. Accuracy results indicate that the linear spline interpolation may have higher maximum absolute error than Chebyshev interpolation. However, the overall performance of spline interpolation is better than Chebyshev interpolation. Furthermore, linear spline collocation takes much less time than Chebyshev collocation with a large number of nodes.

A two-stage grid search method of optimization is developed and examined with accuracy analysis. The results show that this method is more efficient and accurate, as it saves CPU time and improves the accuracy as compared to the one-stage grid search method. We also examine the accuracy effects of allocating different number of nodes for each state variable. The results show that a change in node configuration may result in a more efficient and accurate solution, as it can reduce both the CPU time and the error in the solution to the problem.

REFERENCES

Bellman, Richard E. *Dynamic Programming.* Princeton, NJ: Princeton University Press, 1957.

Bellman, Richard E., and Stuart E. Dreyfus. *Applied Dynamic Programming.* Princeton, NJ: Princeton University Press, 1962.

Bellman, Richard E., R. Kalaba, and B. Kotkin. "Polynomial Approximation--A New Computational Technique in Dynamic Programming: Allocation Processes." *Mathematics of Computation* 17(1963):155-161.

Bera, A., and C. Jarque. "Efficient Tests for Normality, Heteroscedastcity, and Serial Independence of Regression Residuals: Monte Carlo Evidence." *Economics Letters* 7(1981):313-318.

Box, George. E. P., and Gwilym. M. Jenkins. *Time Series Analysis: Forecasting and Control.* Rev. Ed. San Francisco: Holden-Day, Inc., 1976.

Brekke, Jon, Hung-Lin Tao, and Philip M. Raup. "The Minnesota Rural Real Estate Market in 1992." University of Minnesota, Economic Report ER 93-5, July 1993.

Burt, Oscar R., and C. Robert Taylor. "Reduction of State Variable Dimension in Stochastic Dynamic Optimization Models which Use Time-Series Data." *Western Journal of Agricultural Economics*, 14-2(December 1989):213-222.

den Haan, W. J. and Marcet, A. "Accuracy in Simulations." *Review of Economic Studies,* 61:3-17.

Fackler, Paul L., and Mario J. Miranda. "Hybrid Methods for Continuous State Dynamic Programming" (Presented at Computing in Economics and Finance, Boston College, June 1999) [Online]. Available HTTP: http://www4.ncsu.edu/unity/users/p/pfackler/www/ [March 1, 2002].

Gerald, Curtis F., and Patrick O. Wheatly. *Applied Numerical Analysis.* 5th ed. Reading, MA: Addison-Wesley Publishing Company, 1994.

Johnson, Sharon A., Jery R. Stedinger, Christine A. Shoemaker, Ying Li, and Jose A. Tejada-Guibert. "Numerical Solution of Continuous-State Dynamic Programs using Linear and Spline Interpolation." *Operations Research* 41-3(May-June 1993):484-500.

Judd Kenneth L. "Projection Methods for Solving Aggregate Growth Models." *Journal of Economic Theory*, 58:410-452.

Judd Kenneth L. "Approximation, Perturbation, and Projection Methods in Economic Analysis." *Handbook of Computational Economics, Volume I.* H.M. Amman, D.A. Kendrick and J. Rust, eds. Amsterdam: Elsevier Science B.V., 1996, pp. 511-585.

Judd Kenneth L. *Numerical Methods in Economics.* Cambridge, MA: The MIT Press, 1998.

Larson, Donald K., Martin S. Stauber, and Oscar R. Burt. "Economic Analysis of Farm Firm Growth in Northcentral Montana." Montana Agricultural Experiment Station, Montana State University, Bozeman, Research Report 62, August 1974.

Ljung, G., and G. Box. "On a Measure of Lack of Fit in Time Series Models." *Biometrika* 66(1979):265-270.

Miranda, Mario J., and Paul L. Fackler. *Applied Computational Economics and Finance.* The MIT Press, 2002.

Nemhauser, George, L. *Introduction to Dynamic Programming*. New York, NY: John Wiley & Sons, 1966.

Olson, Kent D., and Others. "Annual Report of the Southwestern Minnesota Farm Business Management Association." Staff Paper, Department of Applied Economics, University Of Minnesota, various issues, 1967-99.

Rust, John. "Numerical Dynamic Programming in Economics." *Handbook of Computational Economics, Volume I.* H.M. Amman, D.A. Kendrick and J. Rust, eds. Amsterdam: Elsevier Science B.V., 1996, pp. 619-729.

Santos, Manuel S. "Numerical Solution of Dynamic Economic Models." *Handbook of Macroeconomics*. J. B. Taylor and M. Woodford, eds. Amsterdam: Elsevier Science B.V., 1999, pp. 305-380.

Santos, Manuel S. "Accuracy of Numerical Solutions Using the Euler Equation Residuals." *Econometrica* 68-6(November 2000):1377-1402

Schnitkey, Gary D., C. Robert Taylor, and Peter J. Barry. "Evaluating Farmland Investments Considering Dynamic Stochastic Returns and Farmland Prices." *Western Journal of Agricultural Economics* 14-1(July 1989):143-156.

Schumaker, Larry L. *Spline Functions: Basic Theory*. New York, NY: John Wiley & Sons, 1981.

Schwab, Andrew, and Philip M. Raup. “The Minnesota Rural Real Estate Market in 1988.” University of Minnesota, Economic Report ER 89-3, July 1989.

Stokey, Nancy L., and Robert E. Lucas, Jr., with Edward C. Prescott. *Recursive Methods in Economic Dynamics.* Cambridge, MA: Harvard University Press, 1989.

Taff, Steve J. “Minnesota Land Economics” [Online]. Available HTTP: http://apec.umn.edu/faculty/sjtaff/ [March 1, 2001].

U.S. Department of Agriculture, Economic Research Service. *Agricultural Income and Finance: Situation and Outlook*. Washington DC: February 2001.