Journal of Economic Dynamics & Control ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Journal of Economic Dynamics & Control

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

Valuing programs with deterministic and stochastic cycles

Harry J. Paarsch[a], John Rust[b,*]

[a] Department of Economics, The University of Melbourne, Parkville, Victoria 3010, Australia
[b] Department of Economics, University of Maryland, College Park, MD 20742, USA

A R T I C L E I N F O

Article history:
Received 6 August 2008
Accepted 22 August 2008

JEL classification:
C13
C14
C15

Keywords:
Dynamic programming
Policy iteration
Deterministic cycles
Stochastic cycles
Circulant matrix
Cyclic inversion algorithm

A B S T R A C T

In many dynamic programming problems, a mix of state variables exists – some exhibiting stochastic cycles and others having deterministic cycles. We derive a formula for the value function in infinite-horizon, stationary, Markovian decision problems by exploiting a special partitioned-circulant structure of the transition matrix Π. Our strategy for computing the left-inverse of the matrix $[\mathrm{I}-\beta\Pi]$, which is central to implementing Howard's policy iteration algorithm, yields significant improvements in computation time and major reductions in memory required. When the deterministic cycle is of order n, our *cyclic inversion algorithm* yields an $\mathrm{O}(n^2)$ speed-up relative to the usual policy iteration algorithm.

© 2008 Published by Elsevier B.V.

1. Introduction

Many dynamic programming problems encountered in practice involve a mix of state variables, some exhibiting stochastic cycles (such as unemployment rates) and others having deterministic cycles. Examples of the latter include the day of the week as well as the month and the season of the year. It is common practice in economics to remove trend and seasonal components from stochastic processes in an attempt to make them stationary and, thus, to simplify the analysis. In many problems, however, these transformations can cause important distortions: the real features of the data can be altered as can the behavior implied by empirical models estimated from those data. Most real-world problems involve complicated interactions between variables that evolve according to deterministic cycles and those that evolve according to stochastic cycles. In many nonlinear models, no simple method exists to isolate the deterministically evolving components from the stochastically evolving ones, especially when agents are responding endogenously to both kinds of components.

While it is possible (and generally preferable) to analyse dynamic models without inducing distortions using unjustified transformations of the underlying stochastic processes, doing things 'correctly' typically implies a substantial cost: model complexity. Extra state variables are required in order to capture nonstationarities and deterministic cycles, but the curse of dimensionality makes it costly in terms of the extra time and memory required to solve the models.

We derive a formula for the value function in infinite-horizon, stationary, Markovian decision problems that contain deterministic cycles – i.e., an integer-valued state variable c_t that evolves according to simple modulo arithmetic, $c_{t+1} = c_t + 1 \bmod n$. We exploit the special partitioned-circulant structure of the overall transition probability matrix Π for

* Corresponding author.
E-mail addresses: hjp@paarsch.ecom.unimelb.edu.au (H.J. Paarsch), jrust@gemini.econ.umd.edu (J. Rust).

0165-1889/$ - see front matter © 2008 Published by Elsevier B.V.
doi:10.1016/j.jedc.2008.08.007

Markovian decision problems containing both deterministic and stochastic cycles to derive a formula for the left-inverse of the matrix $[\mathrm{I}-\beta\Pi]$ which is central to solving infinite-horizon dynamic programming problems using the policy iteration algorithm of Howard (1960). This formula constitutes the policy valuation step of the policy iteration algorithm, and yields significant improvements in the computation time as well as major reductions in the memory required when solving infinite-horizon dynamic programming problems with deterministic cycles. In particular, if the problem contains a deterministic cycle of order n, then we demonstrate that our *cyclic inversion algorithm* for solving the linear system $V=u+\beta\Pi V$ for the value function V results in an $\mathrm{O}(n^2)$ speed-up relative to the usual policy iteration algorithm that solves the linear system function V without taking into account the special structure of Π.

Thus, the cyclic inversion algorithm enables us to capture nonstationarities arising from deterministic cycles at relatively low marginal cost. Instead of increasing the cost of solving a dynamic program that explicitly accounts for a deterministic cycle in the underlying problem by a factor of $\mathrm{O}(n^3)$ (the rate that is relevant if the standard policy iteration is used), the cost increases only linearly – by a factor $\mathrm{O}(n)$ – when the cyclic inversion algorithm is used to value candidate policies using the policy valuation step of the policy iteration algorithm. Another important feature of the cyclic inversion algorithm is that it requires less memory than the usual application of the policy iteration algorithm.

In Section 2, we define the class of dynamic programming problems we study, while in Section 3, we show how the presence of a cyclical state variable results in a dynamic programming problem whose transition probability matrix Π has a special form – a partitioned circulant matrix. We derive an expression for the left-inverse of $[\mathrm{I}-\beta\Pi]$ as well as for the solution V to the linear system $V=u+\beta\Pi V$ which is the key equation underlying the policy iteration algorithm. In Section 4, we discuss the computational cost of the policy iteration algorithm when a cyclical state variable is present and then show that our algorithm, which exploits the special structure of Π, obtains an $\mathrm{O}(n^2)$ speed-up relative to standard policy iteration algorithms which treat Π as an unstructured, dense matrix. In Section 5, we illustrate the utility of our algorithm by applying it to solve an optimal timber-harvesting problem which involves harvest costs that vary significantly across different months of the year.

2. Definition of dynamic programming problem

Consider an infinite-horizon, stationary, Markovian dynamic programming problem with state variables (x_t, c_t), decision variable d_t, and pay-off function $u(x_t, c_t, d_t)$. We assume that x_t evolves according to a transition density $g(x_{t+1}|x_t, c_t, d_t)$ that depends on c_t and d_t, but that c_t is a deterministically and cyclically evolving state variable taking the integer values $\{0, 1, \ldots, n-1\}$ according to standard modulo arithmetic

$$c_{t+1} = c_t + 1 \bmod n.$$

The deterministically cycling state variable c_t is also known as a *directed circuit*; see Kalpazidou (2006). As noted above, examples of cyclical state variables include the day of the week as well as the month and the season of a year. Cyclical variables can, therefore, be introduced into dynamic programming problems to capture seasonal or 'calendar effects' in an environment that is otherwise time stationary. Let $D(x,c)$ denote the set of feasible choices for the control variable d when the state is (x,c) and let $\beta\in(0,1)$ denote the discount factor. For the purposes of this paper, suppose that x_t, which contained in the set X, can assume only a finite number of possible values $m\equiv|X|$, the cardinality of the set X. Thus, without loss of generality, we can identify the number of possible values of x_t with the set of integers $\{0,1,\ldots,m-1\}$. Similarly, we assume that the number of feasible decisions is finite for each (x,c) and we let $|D(x,c)|$ denote the number of choices in state (x,c). Thus, there is no loss of generality if we identify this choice set by a subset of integers – e.g., $D(x,c)=\{0,1,\ldots,|D(x,c)|-1\}$.

The Bellman equation for this dynamic programming problem is

$$V(x,c) = \max_{d\in D(x,c)}\left[u(x,c,d) + \beta\sum_{x'} V(x', c+1 \bmod n) g(x'|x,c,d)\right], \tag{1}$$

where $V(x,c)$ represents the present discounted value of pay-offs under an optimal policy.

Under the well-known policy iteration algorithm, the key step in solving this program involves computing the value function corresponding to any given feasible trial policy δ. A policy is feasible if $\delta(x,c)\in D(x,c)$ for all possible values of (x,c). Letting $V_\delta(x,c)$ denote the value function associated with the policy δ for state (x,c), we then have

$$V_\delta(x,c) = u[x,c,\delta(x,c)] + \beta\sum_{x'} V_\delta(x', c+1 \bmod n) g[x'|x,c,\delta(x,c)].$$

If we array V_δ as an $m*n\times 1$ vector and, similarly, let u_δ be a conformable $m*n\times 1$ vector, then we can write Eq. (1) as the following matrix equation:

$$V_\delta = u_\delta + \beta\Pi_\delta V_\delta, \tag{2}$$

where Π_δ is an $m*n\times m*n$ transition matrix which will be described further below. As is well known, the solution to the linear system (2) is given by

$$V_\delta = [\mathrm{I}-\beta\Pi_\delta]^{-1}u_\delta,$$

where the left-inverse of $[\mathrm{I}-\beta\Pi_\delta]$ is guaranteed to exist because Π_δ is a transition matrix (and, therefore, has a matrix norm satisfying $\|\Pi_\delta\| \leqslant 1$) and because $\beta \in (0,1)$. In this case, the left-inverse has the following infinite-series (or Neumann) expansion:

$$[\mathrm{I}-\beta\Pi_\delta]^{-1} = \sum_{t=0}^{\infty} \beta^t [\Pi_\delta]^t.$$

As is also well known, the numerical solution to linear systems such as Eq. (2), using standard LU decompositions which do not exploit any special structure of the matrix, requires $\mathrm{O}([m*n]^3)$ floating point operations. In the next section, we illustrate that the presence of the cyclical state variable induces a special structure that can be exploited to reduce the solution to Eq. (2) to the solution to n linear systems of order m. Thus, after exploiting this special structure, the total work involved in solving for V_δ is $\mathrm{O}(nm^3)$ – a speed-up of $\mathrm{O}(n^2)$ relative to standard policy iteration which ignores the special structure of Π_δ.

3. Structure of Π matrix

In this section, we characterize the special structure of the matrix Π_δ which is induced by the cyclical state variable c_t. Suppose the state variables are arranged so that the x variables are ordered from 0 to $m-1$ in an inner do-loop and the cyclical state variable is ordered from 0 to $n-1$ in the outer do-loop. Thus, we define

$$V_\delta = \begin{bmatrix} V_\delta(\cdot,0) \\ V_\delta(\cdot,1) \\ \vdots \\ V_\delta(\cdot,n-1) \end{bmatrix},$$

where $V_\delta(\cdot,j)$ is the $m \times 1$ vector given by

$$V_\delta(\cdot,j) = \begin{bmatrix} V_\delta(0,j) \\ V_\delta(1,j) \\ \vdots \\ V_\delta(m-1,j) \end{bmatrix}.$$

Assume that u_δ is arrayed in a conformable fashion. Under this ordering of the state variables, we can represent the $mn \times mn$ transition matrix Π_δ as follows:

$$\Pi_\delta = \begin{bmatrix} \mathbf{0} & P_0 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & P_1 & \cdots & \mathbf{0} \\ \mathbf{0} & \cdots & \cdots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \cdots & \mathbf{0} & P_{n-2} \\ P_{n-1} & \cdots & \cdots & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{3}$$

where $\mathbf{0}$ is an $m \times m$ matrix of zeros, and P_j is an $m \times m$ matrix given by

$$P_j = \begin{bmatrix} g[0|0,j,\delta(0,j)] & g[1|0,j,\delta(0,j)] & \cdots & g[m-1|0,j,\delta(0,j)] \\ g[0|1,j,\delta(1,j)] & g[1|1,j,\delta(1,j)] & \cdots & g[m-1|1,j,\delta(1,j)] \\ \vdots & \vdots & \ddots & \vdots \\ g[0|m-2,j,\delta(m-2,j)] & g[1|m-2,j,\delta(m-2,j)] & \cdots & g[m-1|m-2,j,\delta(m-2,j)] \\ g[0|m-1,j,\delta(m-1,j)] & g[1|m-1,j,\delta(m-1,j)] & \cdots & g[m-1|m-1,j,\delta(m-1,j)] \end{bmatrix}.$$

Lemma 1. *If Π_δ is a matrix given in Eq. (3), then for any $\beta \in (0,1)$ a left-inverse, Q, of the matrix $[\mathrm{I}-\beta\Pi_\delta]$ exists and can be partitioned as*

$$Q = \begin{bmatrix} Q_{0,0} & Q_{0,1} & \cdots & Q_{0,n-1} \\ Q_{1,0} & Q_{1,1} & \cdots & Q_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{n-2,0} & Q_{n-2,1} & \cdots & Q_{n-2,n-1} \\ Q_{n-1,0} & Q_{n-1,1} & \cdots & Q_{n-1,n-1} \end{bmatrix},$$

where $Q_{j,k}$ is an $m \times m$ matrix given by

$$Q_{j,k} = \begin{cases} A_j & \text{if } j = k, \\ \beta^{k-j} A_j \left[\prod_{i=j}^{k-1} P_i \right] & \text{if } k > j, \\ \beta^{k+n-j} A_j \left[\prod_{i=j}^{n-1} P_i \right] \left[\prod_{i=0}^{k-1} P_i \right] & \text{if } k < j \end{cases}$$

and A_j is an $m \times m$ matrix given by

$$A_j = \left[\mathrm{I} - \beta^n \left(\prod_{i=j}^{n-1} P_i \prod_{i=0}^{j-1} P_i \right) \right]^{-1}. \tag{4}$$

Lemma 2. *If Π_δ is a matrix given in Eq.* (3), *then for any $\beta \in (0,1)$ the unique solution V_δ to the linear system* (2) *is given by*

$$V_\delta(\cdot,j) = A_j u_j + A_j \left[\sum_{k=j+1}^{n-1} \beta^{k-j} \left(\prod_{i=j}^{k-1} P_i \right) u_k \right] + A_j \left[\sum_{k=0}^{j-1} \beta^{k+n-j} \left(\prod_{i=j}^{n-1} P_i \right) \left(\prod_{i=0}^{k-1} P_i \right) u_k \right], \tag{5}$$

where $u_k = u_\delta(\cdot,k)$.

Comment 1. Formula (5) has an intuitive interpretation: first, consider a dynamic programming problem without the cyclical state variable c_t, so the value function is given by

$$V_\delta = [\mathrm{I} - \beta \Pi_\delta]^{-1} u_\delta = \sum_{t=0}^{\infty} \beta^t [\Pi_\delta]^t u_\delta \tag{6}$$

which expresses V_δ directly as an infinite discounted sum of expected future pay-offs because $[\Pi_\delta]^t u_\delta$ is the conditional expectation of pay-offs t periods ahead. To wit, the kth element of this vector is $\mathscr{E}\{u[\tilde{x}_t, \delta(\tilde{x}_t)] | x_0 = k\}$ – the conditional expectation of the pay-off at time t given that the state at time $t = 0$ is $x_0 = k$, where the ~'s signify random quantities.

Comment 2. The formula for A_j in Eq. (4) and the solution for V_δ in Eq. (5) are also valid when the dynamic programming problem contains continuous state variables, or discrete-valued state variables that can assume an infinite number of possible values. In this case, the P_j's are not matrices, but rather *Markov operators* (i.e., linear operators that represent the conditional expectation operation) whose domain is an infinite-dimensional linear space. We have emphasized the discrete/finite case because, in any actual implementation, a discretization of continuous state variables would be employed resulting in P_j's which are finite-dimensional matrices that approximate the true infinite-dimensional Markov operators.

In the presence of a cyclical state variable c_t, in addition to the state variable x_t, the value function depends on the n possible values for c as well as the m possible values for x. We have written $V_\delta(\cdot,j)$ to denote the $m \times 1$ vector providing the value as a function of the m possible values of x when the current stage of the cycle is $c = j$. The formula for $V_\delta(\cdot,j)$ given in Eq. (5) shows that this discounted pay-off can be decomposed into n terms, corresponding to the infinite expected discounted sum of the stream of pay-offs in each of the n possible values of the cyclical variable c. If the current state of the cycle is $c = j$, then the leading term in $V_\delta(\cdot,j)$ is the infinite discounted expected stream of u_j pay-offs. These pay-offs are received every n periods from the current period, and the expected present value of the stream of pay-offs for the 'current cycle value' $c = j$ is just $A_j u_j$, by direct analogy to Eq. (6), except that the transition probability is not a single-period transition probability, but rather an *n-stage transition probability*. For example, in state $c = j$, this n-stage transition probability is

$$\Pi_j = \left[\prod_{i=j}^{n-1} P_i \right] \left[\prod_{i=0}^{j-1} P_i \right], \tag{7}$$

where the first product in this equation is the transition probability matrix for the transition between $c = j$ to $c = n-1$ (i.e., the 'first part of the cycle'), and the second product is the transition from $c = 0$ to $c = j-1$, representing the crossing of the maximum value that the cyclical state variable can assume $c = n-1$, and resetting it to $c = 0$ in accordance with the modulo-arithmetic law of motion for the cyclical state variable. For example, if n were 12 and the values of the cyclical state variable are then interpreted as months of the year (with $c = 1$ being treated as January), then when $c = 3$ (March), the first term of the product on the right-hand side of the equal sign in Eq. (7) is the transition probability for March through December, while the second term is the transition probability for the months January and February. Thus, $\Pi_j u_j$ would be an $m \times 1$ vector that gives the expected pay-off received when $c = j$ one year from now, as a function of the current value of x today.

The other terms in Eq. (5) are the discounted pay-offs corresponding to the other $n-1$ values of the cyclical state variable other than $c = j$. The formula reflect computing the expected discounted value of the pay-offs for other values of the cycle, say $c = k$ back to the 'reference value' $c = j$, and then taking the expected discounted sum of this stream of pay-offs into the infinite future. We do this by multiplying by A_j. In this way, formula (5) represents the contribution to the expected discounted value $V_\delta(\cdot,j)$ of the infinite stream of pay-offs from the other $n-1$ values of the cycle except $c = j$, but to do this the formula shifts the other pay-offs appropriately, so they can be treated as if they were being received in future values when the cycle equals its reference value $c = j$ instead of the actual values of the cycle $c = k$ in which these other pay-offs are received. Thus, in terms of the example, where the cycle represents months of the year, Eq. (5) decomposes $V_\delta(\cdot, j)$ into the sum of 12 terms where each terms is the infinite discounted stream of pay-offs received in each separate month of the year.

4. Speed-up from exploiting special structure of Π

As is well known, using standard algorithms to solve a system of m linear equations in m unknowns requires $O(m^3)$ operations – multiplications and additions. For example, Gauss–Jordan elimination requires $(m^3/2) + m^2 - (5m/2) + 2$ multiplications and $(m^3/2) - (3m/2) + 1$ additions – approximately m^3 floating point operations in total. Solving the linear system using the LU decomposition requires approximately $\frac{2}{3}m^3$ floating point operations.

If the model has a cyclical state variable, which can assume n possible values, and if the remaining state variables can assume m possible values, then the policy valuation step of the policy iteration algorithm requires solving a system of $m * n$ equations in as many unknowns – the value function V_δ. Thus, any of the standard algorithms for solving this system will require $O([m * n]^3)$ operations.

Now consider solving for V_δ using formula (5) in Lemma 2. This requires solving n linear systems with m equations and m unknowns for each of the 'segments' of V_δ, $V_\delta(\cdot, j)$, $j = 1, \ldots, n$. Furthermore, one needs to construct the matrices A_j defining these n linear systems. A total of $2n - 3$ multiplications of the P_j transition matrices are required to construct the A_j matrices. As is well known, multiplying two $m \times m$ matrices requires $2m^3 - m^2$ multiplications and additions. Thus, approximately $4nm^3$ floating point operations are required to create the A_j matrices. There are also $n - 1$ additional matrix–vector multiplications required to compute the individual terms in the brackets in Eq. (5), but each of these matrix–vector multiplications requires only $2m^2 - m$ floating point operations. We conclude this discussion with

Lemma 3. *The total number of operations required to compute V_δ using formula* (5) *is* $O(nm^3)$, *whereas the number of operations required to compute V_δ using standard linear equation solvers that do not exploit the special structure created by the presence of the cycle state variable is* $O([mn]^3)$. *Thus, our cyclic inversion algorithm for solving for V_δ in formula* (5) *results in a speed-up of* $O(n^2)$ *over standard policy iteration algorithms that do not exploit this special structure.*

5. Application

Paarsch and Rust (2008) studied the problem of optimal timber harvesting by a large player in the timber market, in their case the British Columbia Ministry of Forests and Range (MoFR). They formulated and solved the optimal harvesting strategy – the policy that maximizes the discounted expected profits from timber harvesting over an infinite-horizon. Their model accounted for the potential impact that large harvests could have on the price of lumber as well as on the cost of harvesting timber and, *à propos* the topic of this paper, their model accounted for significant monthly variations in the cost of harvesting timber.

Surprisingly (to some), the best time to harvest timber is in winter when the ground is frozen. When the ground is firm, heavy machinery and large trucks can haul away felled trees easily. The most costly months in which to harvest are in spring. In spring, the melting of snow is called 'break-up' by loggers. In muddy conditions, the costs of harvesting timber increase significantly. Sometimes, it is impossible to harvest safely. In the summer, extreme heat as well as dry conditions can make it costly to harvest because of the increased risk of forest fires. Consequently, a good portion of timber harvesting is undertaken during the fall and winter months, so it is natural to include a state variable indexing the current month of the year to reflect these natural variations in harvest costs as well as the resulting variation in the actual volumes harvested which are observed in the data.

Let q denote the current volume of timber stewarded by the MoFR, and let p denote the current price of lumber. These are naturally treated as continuous state variables, but they will be made discrete to facilitate numerical solution of the harvesting problem below. Let n denote the current month. The Bellman equation for the optimal harvesting problem, formulated at a monthly time interval, is

$$V(p, q, n) = \max_{0 \leqslant h \leqslant f(q,n)} \left\{ \pi(p, h, n) + \beta \int_{p'} V[p', f(q, n) - h, n + 1 \bmod 12] g(p'|p, h, n)\, \mathrm{d}p' \right\},$$

where $\beta \in (0, 1)$ is the MoFRs discount factor, π is the expected profit from harvesting, f is the law of motion for timber growth, and $g(p'|p, h, n)$ is a transition probability density function specifying the stochastic evolution of lumber prices.

The equation

$$q_{t+1} = f(q_t, n_t) - h_t$$

is the 'controlled' law of motion for timber volume (measured in millions of cubic metres of timber), taking into account harvesting h_t. The current month n_t affects timber growth; e.g., growth is slower in the winter than in the spring. Lumber prices are assumed to evolve according to a Markov process with transition density $g(p_{t+1}|p_t, h_t, n_t)$ which depends not only on the previous lumber price p_t, but also on the volume harvested h_t and, potentially, on the current month of the year n_t.

The dependence of future prices on the current harvest decision h_t is the key avenue through which we account for the impact on prices of harvesting unusually large volumes of timber in short periods of time. The model can account for seasonal patterns in lumber prices and in timber growth through the state variable n_t.

We assume that a timber-harvesting decision is made at the beginning of each month, while the actual harvesting does not occur until the end of the month, reflecting delays involved in assembling logging crews and carrying out the harvest. The profit function for harvesting is given by

$$\pi(p,h,n) = h\int p' g(p'|p,h,n)\,\mathrm{d}p' - c(h,n),$$

where $c(h,n)$ is the cost function for harvesting a total volume of timber h in month n. The cost function c is the primary avenue through which the current month affects harvesting decisions: empirically, the other possible avenues – i.e., the effect of the month on aggregate timber growth and on lumber prices – are negligible. In fact, we exclude them from the model solved below. The *only* avenue through which the current month affects harvesting decisions is through the harvest cost function for timber $c(h,n)$.

We solved the model for varying numbers of grid points for the continuous state variables $x=(p,q)$. For small numbers of grid points (e.g., 10 grid points for p and 10 for q, so $m=100$), the implied transition matrices (P_j) are of dimension 100×100, while the overall matrix $[\mathrm{I}-\beta\Pi]$ and the dimension of the linear system $V_\delta = u_\delta + \beta\Pi_\delta V_\delta$ that must be solved at each policy iteration step (where δ is a candidate optimal harvesting rule) is of dimension $mn = 1200 = 100\times 12$. Systems of this size can be solved quickly on laptop computers in a matter of seconds, without exploiting the special structure of Π. However, for finer discretizations, say when p and q are discretized into 30 possible values each, then the linear system that must be solved in each policy valuation step is of dimension $10\,800 = 900\times 12$. Sheer memory requirements to store the matrix Π rule out the direct of application of policy iteration on current laptops: storing a $10\,800\times 10\,800$ matrix requires over 933 megabytes of memory. By contrast, the cyclic inversion algorithm requires storing 12 900×900 matrices and a working space of at most 36 900×900 matrices, just under 30 megabytes of memory.

Lemma 3 predicts improvements of $O(n^2)$ or $O(144)$. When we implemented our cyclic inversion algorithm, we only obtained speed-ups of just over 14. We conjecture that the overhead involved in preparing the matrices relative to the time required by the conventional policy iteration algorithm (e.g., when p and q were discretized into a product of only 100 points) was the cause. We must emphasize, however, that our result is correct: it applies asymptotically when the number of possible values for the other state values m is sufficiently large. When m is sufficiently large, the overhead involved to create the n matrices and to carry-out the $2n$ multiplications of $m\times m$ matrices as well as the other intermediate calculations required to implement formula (5), becomes small relative to the time it takes to solve $V_\delta = u_\delta + \beta\Pi_\delta V_\delta$ directly as an unstructured, dense linear system: that requires $O([12m]^3)$ operations.

In our tests, we were unable to obtain speed-ups approaching $12^2 = 144$ because memory constraints dominated. The largest unstructured, dense system that we could solve on a laptop with two gigabytes of memory was $m=400$; i.e., p and q were both discretized to assume 20 values each. The resulting linear system was then of dimension $4800 = 12\times 400$. Of course, we might have accessed more memory than this by using virtual memory, but this almost always comes at an extremely high price in terms of elapsed time. In general, it is inadvisable to solve large systems in virtual memory: swapping to the hard-drive will take more time than it is worth. Suffice it to say that the cyclic inversion algorithm we have proposed makes it possible to solve the timber-harvesting problem for sufficiently fine discretizations that would have been otherwise impossible – unless, of course, we resorted to solving the problem on a supercomputer. But even on a supercomputer, we could undertake even finer discretizations that would rule out the methods currently used.

In Figs. 1 and 2, we depict the optimal harvesting decision rules

$$h_t = \delta(p_t, q_t, n_t)$$

from the policy iteration algorithm when 30 point grids are used for both p and q, resulting in a discretized value function and decision rule with $10\,800 = 12\times 900$ elements. In Fig. 1, we illustrate the harvesting policy for January, while in Fig. 2 we illustrate the harvesting policy in April. It is clear that the higher harvesting costs in April are reflected in the solution, and significantly less timber is harvested in April than in January.

In Figs. 3 and 4, we plot a two-dimensional slice of the optimal harvest function for four months (January, March, May, and July) when $q_t = 83.26$ and when $q_t = 12.34$. In both cases, the optimal volume harvested in January monotonically dominates harvests in July, which dominate harvests in May, which dominate March. This pattern is a direct implication of the ordering of harvesting costs because harvesting costs are the lowest in January followed by July, May, and then March. For this example, the marginal harvesting costs in March were assumed twice those in January.

Most importantly, however, it should be evident that the solution is complicated and the relative position of the harvest functions depends on the volume of timber q_t. In particular, there is no simple uniform shift or transformation of the January harvest policy that would result in valid harvest rules for the other months of the year, and for all volumes q_t. This implies that there is no simple way to ‘deseasonalize’ the timber- harvesting problem by simply shifting the decision rules in some simple manner, such as a simple parallel shift of the January harvest rule by a pre-determined amount.

In Fig. 5, we illustrate the difference between the value of timber (in billions of Canadian dollars) in January and that in April. That is, we display a plot of $[V(p,q,1) - V(p,q,4)]$, where $n=1$ corresponds to January and $n=4$ corresponds to April. It should be obvious here, too, that no simple uniform or parallel shift of the January value function (an approach that

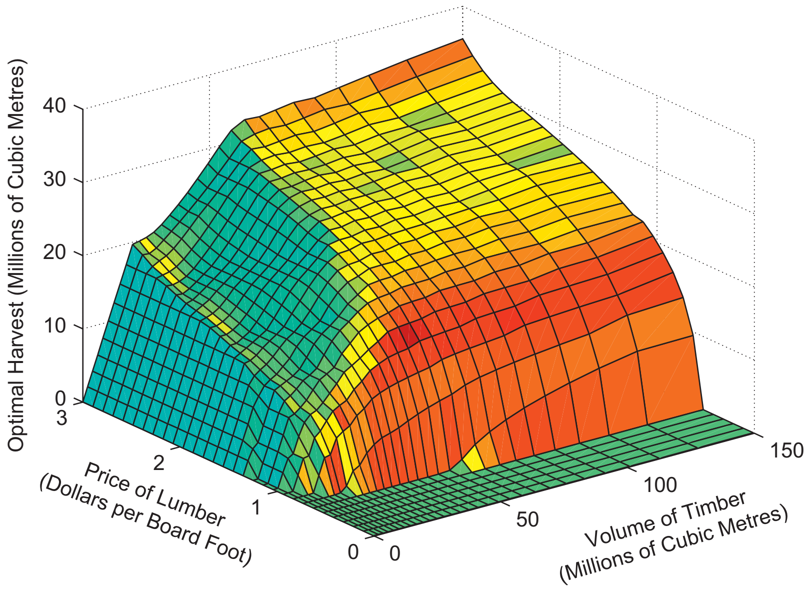

Fig. 1. Optimal harvest function in January.

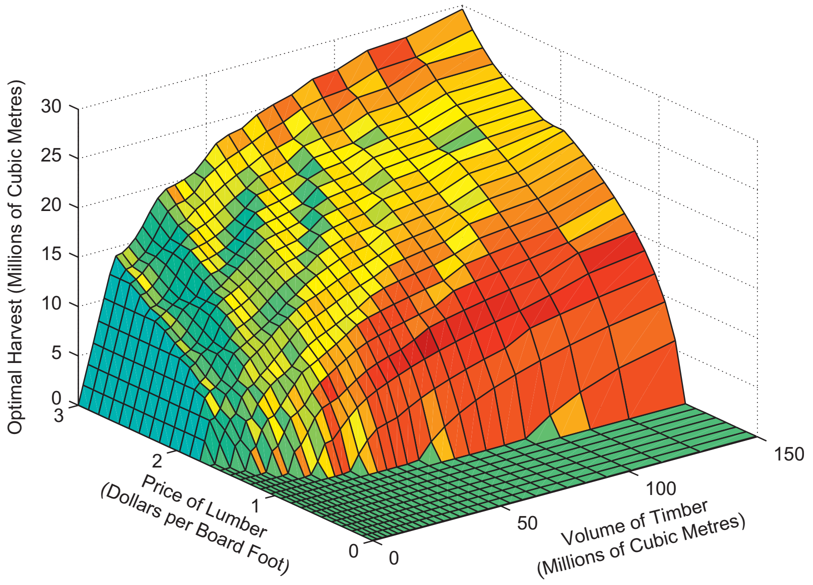

Fig. 2. Optimal harvest function in April.

would be implied by most naïve attempts to 'deseasonalize' data) can provide an accurate prediction of how timber values actually shift across months of the year. The actual amount of the shift ranges from 0 for sufficiently low values of p (since the optimal decision rule is not to harvest anything in either April or January when prices are sufficiently low) to a difference approaching \$1.5 billion dollars when prices are extremely high. Note, however, that the differences in values do not vary in any simple monotonic fashion: the gain from harvesting in January relative to April actually *falls* as a function of q_t when q_t exceeds seventy.

Fig. 6 depicts the deterministic cycle in the value of timber for three different values of lumber prices. No deterministic cycles exist in the value of timber for sufficiently low lumber prices, the lowest line in Fig. 5 which corresponds to a lumber price $p = 0.319$. This is because it is not optimal to harvest in any month when lumber prices are this low; see the flat sections of the optimal harvest functions in Figs. 1 and 2 which correspond to no harvesting at sufficiently low lumber prices. We do, however, see a cycle when lumber prices are high. For example, the top line in Fig. 6 corresponds to a lumber price of $p = 0.939$; an obvious cycle exists in this case. The value of harvesting reaches its lowest point in the spring due to the high costs of harvesting during this period of the year; soggy ground conditions make it difficult to access and to haul

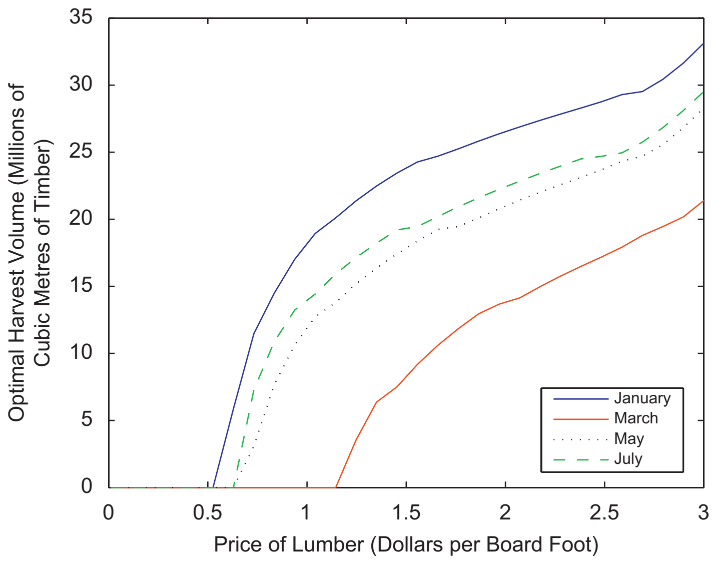

Fig. 3. Two-dimensional slices of optimal harvest function, $q_t = 83.26$.

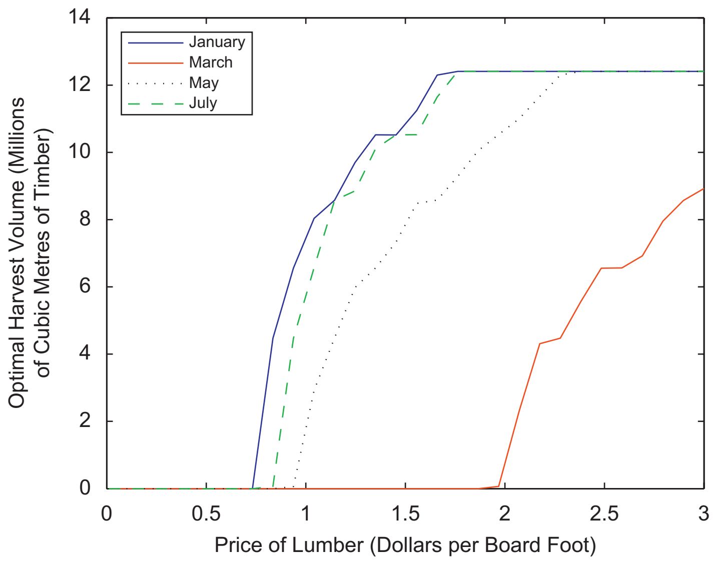

Fig. 4. Two-dimensional slices of optimal harvest function, $q_t = 12.34$.

logs from harvesting sites. The value is the highest in the fall when colder, dryer conditions make it much easier to fell trees and to haul logs. It is, however, evident that the deterministic cycles in timber values are relatively minor compared to the much large stochastic cycles induced by the stochastic cycles in lumber prices p_t.

Fig. 7 depicts a time-series simulation which illustrates how closely the value of timber is related to the price of lumber. The top panel illustrates a five-year realization of lumber prices that exhibits two large peaks in prices in the first and second years of the simulation. The second panel illustrates that a large volume of timber is cut shortly after lumber prices reach their peak value in the first year of the simulation. Then, at the time of the second slightly smaller peak in lumber prices in the second year of the simulation, another large quantity of timber is harvested, although not as large a volume as the first harvest around first year of the simulation. After the second year, lumber prices remain lower for the duration of the five-year simulation period and the model predicts that it is not optimal to harvest any more at these prices. Consequently, the volume of timber slowly grows, resulting in the slight positive slope of the volume of timber curve in the middle panel of Fig. 7 after the last harvest in the second year of the simulation.

The bottom panel of Fig. 7 depicts the value of timber during this same simulation period: it is evident that the value of timber has, generally, the same shape as the simulated price path for timber, including the peaks in the first and second years. Even though there are no more harvests of timber after the second year, the simulated value of timber moves up and

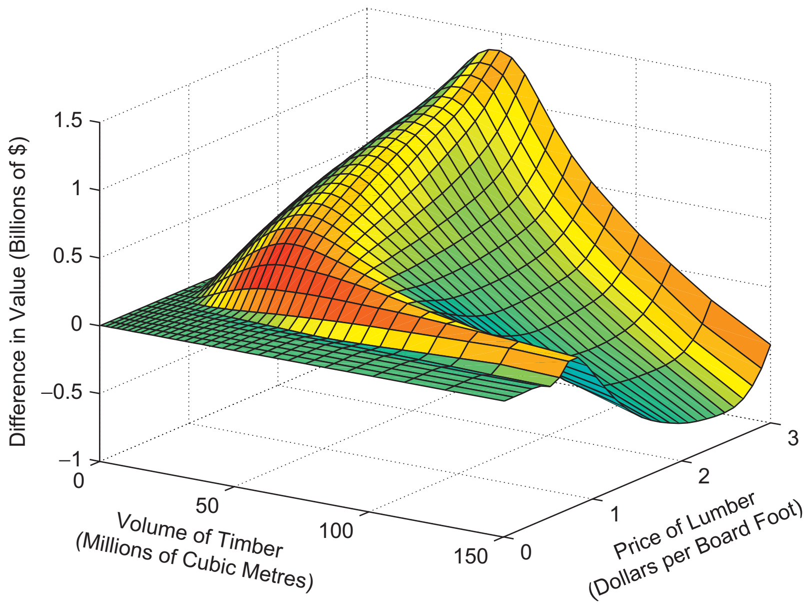

Fig. 5. $[V(p,q,1)-V(p,q,4)]$, difference in value functions, January over April.

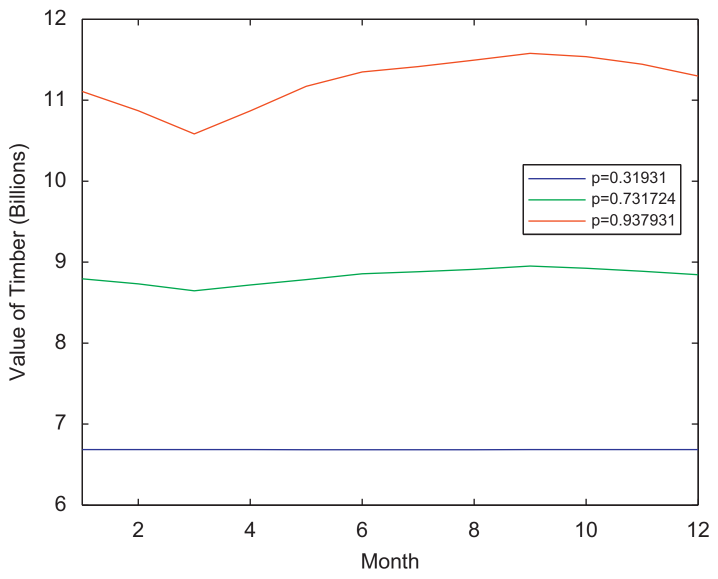

Fig. 6. Deterministic cycles in timber value, by month, $q_t = 76.0666$.

down with the price of lumber in a way that is roughly parallel to the lumber price, although the amplitude of fluctuations in timber values is somewhat muted relative to the fluctuations in lumber prices. This is an indication that the value function is not a simple product of the price of lumber and volume of timber; i.e., the value function does *not* have the form $V(p,q,n) = pq$. It is clear, however, that the stochastic cycles in the value of timber induced by the stochastic cycles in the price of lumber are far larger and more important than the much smaller deterministic cycles induced by variations in harvesting costs at different months of the year.

6. Conclusions

Our cyclic inversion algorithm makes it possible to account for important deterministic cycles which exist in many real-world problems. Once we account explicitly for these variables, we are likely to realize that there are generally complicated

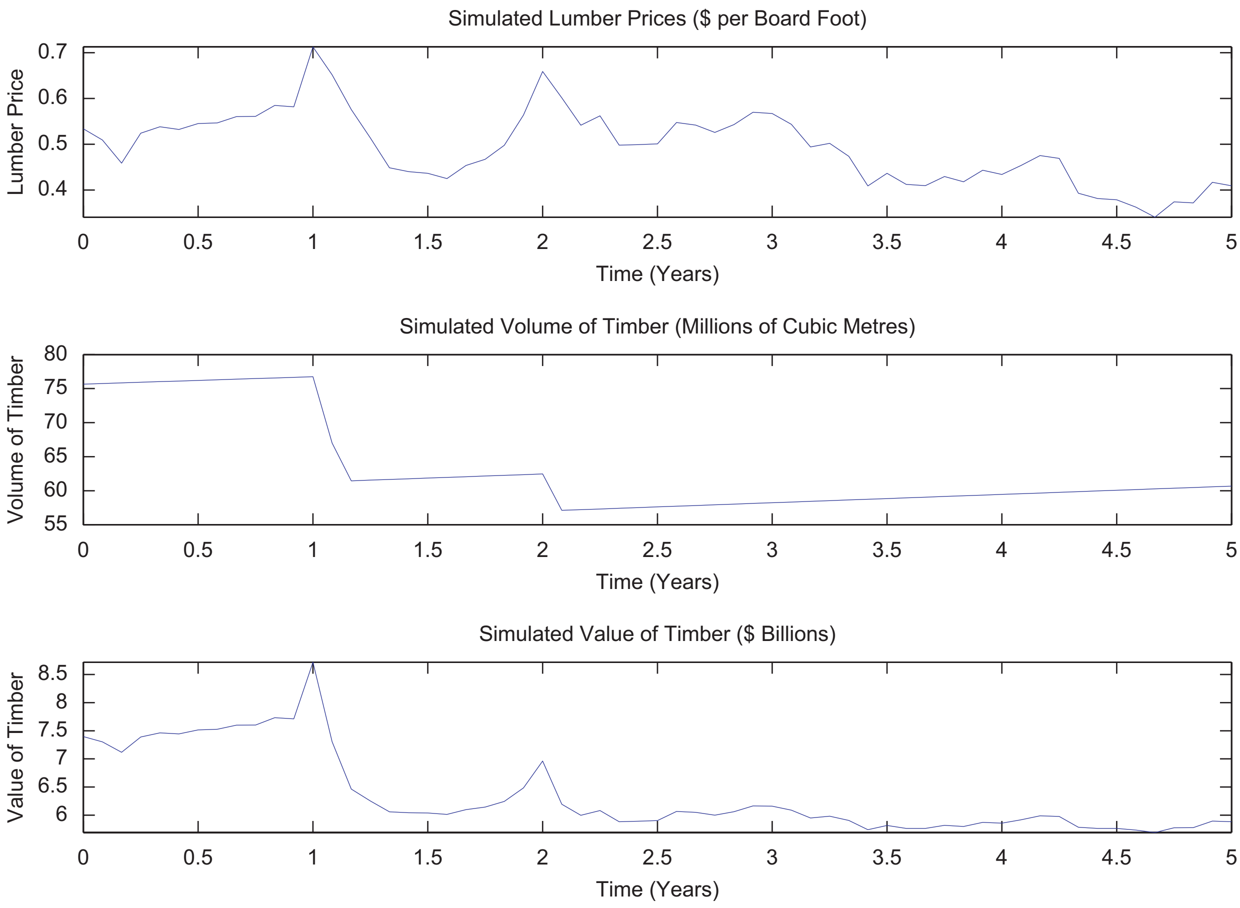

Fig. 7. Stochastic cycles in timber value.

interactions between the deterministically and stochastically evolving variables that can make it very misleading to apply the commonly used deseasonalizations and transformations often employed in macroeconomic analyses. Instead, we believe that it is preferable to model directly the most important deterministic, regularly varying features present in many economic problems. Even though the laws of motion for these variables are often trivial, we generally have no way of knowing *a priori* how the deterministic variation interacts with the stochastic variation in other variables to affect the optimal behavior of agents in these environments.

Acknowledgments

The research reported here builds on research by the authors that was supported by National Science Foundation Grant 0241509, 'Optimal Harvesting of Timber: Valuing Timberland with Stochastically Evolving Timber Volume and Prices Using Linked Biological/Geographical Data from British Columbia' during the period July 1, 2003 to June 30, 2007. The ideas contained in this paper represent the views of the authors and do not reflect the views of the National Science Foundation or the British Columbia Ministry of Forests and Range. We are grateful to Kurt Anstreicher, Samuel Burer, Srihari Govindan, Clinton J. Levitt, David Prentice, Alberto M. Segre, Yiğit Sağlam, Che-Lin Su, and John Tsitsiklis for helpful comments and useful suggestions on an earlier draft of the paper.

References

Howard, R., 1960. Dynamic Programming and Markov Processes. MIT Press, Cambridge, USA.
Kalpazidou, S., 2006. Cycle Representations of Markov Processes. Springer, New York.
Paarsch, H.J., Rust, J., 2008. Timber Cycles. Unpublished manuscript, Department of Economics, University of Iowa.