SIAM J. MATRIX ANAL. APPL.
Vol. 31, No. 2, pp. 434–459

STABILIZING POLICY IMPROVEMENT FOR LARGE-SCALE INFINITE-HORIZON DYNAMIC PROGRAMMING*

MICHAEL J. O'SULLIVAN† AND MICHAEL A. SAUNDERS‡

Abstract. Today's focus on sustainability within industry presents a modeling challenge that may be dealt with using dynamic programming over an infinite time horizon. However, the curse of dimensionality often results in a large number of states in these models. These large-scale models require numerically stable solution methods. The best method for infinite-horizon dynamic programming depends on both the optimality concept considered and the nature of transitions in the system. Previous research uses policy improvement to find strong-present-value optimal policies within normalized systems. A critical step in policy improvement is the calculation of coefficients for the Laurent expansion of the present-value for a given policy. Policy improvement uses these coefficients to search for improvements of that policy. The system of linear equations that yields the coefficients will often be rank-deficient, so a specialized solution method for large singular systems is essential. We focus on implementing policy improvement for systems with substochastic classes (a subset of normalized systems). We present methods for calculating the present-value Laurent expansion coefficients of a policy with substochastic classes. Classifying the states allows for a decomposition of the linear system into a number of smaller linear systems. Each smaller linear system has full rank or is rank-deficient by one. We show how to make repeated use of a rank-revealing LU factorization to solve the smaller systems. In the rank-deficient case, excellent numerical properties are obtained with an extension of Veinott's method [*Ann. Math. Statist.*, 40 (1969), pp. 1635–1660] for substochastic systems.

Key words. dynamic programming, policy improvement, rank-revealing LU, numerical stability

AMS subject classifications. 49L20, 15A06, 65F50

DOI. 10.1137/060653305

1. Introduction.

A current focus of many industries is *sustainability*—ensuring that the required resources will never be exhausted. For example, both agriculture and aquaculture rely on renewable resources for continued profits. One method for addressing sustainability is by considering the effect of management policies over an infinite time horizon. Any policy that destroys the stock (even slowly) will be suboptimal compared to a policy that maintains (or renews) the resource over time.

Dynamic programming can be used to model systems over an infinite time horizon and can also incorporate uncertainty in the behavior of the system. However, these models often require a large state space to represent the system accurately. Thus, any solution method must be able to deal with the computational challenges presented by a large state space.

To select an optimal policy, one must differentiate between all the possible policies that exist for controlling such processes. Most previous research focuses on substochastic systems (where transitions between states are probabilistic), where the objective is maximum reward rate, present-value optimality, or strong-present-value

*Received by the editors March 1, 2006; accepted for publication (in revised form) by L. Vandenberghe December 10, 2008; published electronically April 22, 2009.
http://www.siam.org/journals/simax/31-2/65330.html

†Department of Engineering Science, University of Auckland, Auckland, 1142, New Zealand (michael.osullivan@auckland.ac.nz). This author's research was partially supported by National Science Foundation grant CCR-9988205 and Office of Naval Research grant N00014-96-1-0274.

‡Department of Management Science & Engineering, Stanford University, Stanford, CA 94305 (saunders@stanford.edu). This author's research was partially supported by National Science Foundation grants CCR-9988205 and CCR-0306662, and Office of Naval Research grants N00014-96-1-0274, N00014-02-1-0076, and N00014-08-1-0191.

optimality. Of these three concepts, only strong-present-value optimality considers short, intermediate, and long-term behavior. (Using the maximum reward rate as an objective ignores any transient behavior, and present-value optimality discounts away the importance of long-term behavior.)

Normalized systems don't require transitions to be substochastic, but rather, the transition matrix of every stationary policy has spectral radius not exceeding one. Rothblum [12] notes that Blackwell's existence theorem for substochastic systems extends to normalized systems and then generalizes the methods from Miller and Veinott [8] and Veinott [14, 15] to give a policy improvement method for normalized systems. This (more general) policy improvement requires the coefficients of an (augmented) Laurent expansion of the present-value for a policy. The coefficients are the unique solution of a set of linear equations (identical to those from Veinott [14] except in the number of arbitrary variables).

Veinott [14, p. 1651] shows how to solve these linear equations efficiently (for substochastic systems) by identifying recurrent classes, solving within each such (stochastic, irreducible) class by repeated application of Gaussian elimination, and using these solutions to solve the remainder of the system. Rothblum [12] notes that Veinott's method may be extended to normalized systems, but extra work is required to partition the system into communicating classes and identify which classes are recurrent. No previous research discusses the numerical properties of the linear equations or the numerical stability of solution methods (although Veinott [14] recognizes the linear dependence of the linear equations within recurrent classes).

This paper presents computationally efficient methods for solving the linear equations for any policy with substochastic classes. Rather than try to solve the equations as a single large system (with uncertain rank), the methods use the partitioning of the state space into communicating classes (similar to Veinott [14]). By restricting the linear system to each class and solving these smaller systems in a specified order, we reduce the full linear system to a sequence of smaller linear systems that have full rank or are rank-deficient by one. We give effective methods for solving the smaller linear systems, using repeated application of a rank-revealing LU factorization (RRLU). In the rank-deficient case, excellent numerical properties are obtained with an extension of Veinott's method [14] for substochastic systems.

The paper is organized as follows. Section 2 introduces the preliminary definitions and results necessary for policy improvement in systems with substochastic classes. Once the system has been partitioned into communicating classes, section 3 gives a method for finding the Laurent expansion coefficients within each (substochastic, irreducible) class. We also present Veinott's [14] method for stochastic, irreducible systems and extend it to systems with system degree $d > 1$. An example illustrating both methods is given in section 4. Section 5 discusses the stability and computational efficiency of the methods, and section 6 presents numerical comparisons. Finally, section 7 presents a comparison of our methods when used on a real-world stochastic dynamic programming application.

2. Preliminaries. This section contains a summary of definitions and results presented in [2, 8, 14, 15, 12].

Consider a *general system* observed in periods $1, 2, \ldots$. The system exists in a finite set $\mathcal{S}$ of S states. In each state $s \in \mathcal{S}$ the system takes one of a finite set $\mathcal{A}_s$ of actions. Taking action $a \in \mathcal{A}_s$ from $s \in \mathcal{S}$ earns *reward* $r(s, a)$ and causes a *transition* to state $t \in \mathcal{S}$, with rate $p(t|s, a)$.

Each (stationary) policy $\delta \in \Delta$ is a function that assigns a unique action $\delta_s \in \mathcal{A}_s$ to each $s \in \mathcal{S}$ and induces a single-period S-column *reward vector* $r_\delta \equiv (r(s, \delta_s))$ and an $S \times S$ *transition matrix* $P_\delta \equiv (p(t|s, \delta_s))$.

2.1. Systems with substochastic classes. Each transition matrix P_δ (corresponding to policy δ) defines a set of communicating classes as follows. A state s *communicates* with another state t if there exists some $N > 0$ such that $P^N_{\delta st} > 0$. A *communicating class* $\mathcal{C}$ is a maximal subset of $\mathcal{S}$ such that every pair of states $s, t \in \mathcal{C}$ communicate with each other. Each communicating class $\mathcal{C}$ may be

transient: $\lim_{N\to\infty} P^N_{\delta st} = 0$ $(C, 1)$ for all $s \in \mathcal{S}$, $t \in \mathcal{C}$, or

recurrent: $\lim_{N\to\infty} P^N_{\delta st} > 0$ $(C, 1)$ for some $s \in \mathcal{S}$, $t \in \mathcal{C}$,

i.e., the long-run probability of being in $\mathcal{C}$ is 0 or positive, where the limit is the *Cesàro limit of order* 1.

If every class $\mathcal{C}$ (under δ) has $0 \le p(t|s, a) \le 1$ and $\sum_{t\in\mathcal{C}} p(t|s, a) \le 1$, $s, t \in \mathcal{C}$, $a \in \mathcal{A}_s$, then δ has *substochastic classes*. The blocks of P_δ corresponding to these classes lie on the diagonal and have spectral radius not exceeding one, so P_δ has spectral radius not exceeding one. If every (stationary) policy has substochastic classes, the system has substochastic classes. Also, since the spectral radius of the transition matrix for every (stationary) policy does not exceed one, the system is normalized.

System degree. For each policy δ, let the *degree* of δ be the smallest nonnegative integer $d_\delta \equiv i$ such that Q^i_δ and Q^{i+1}_δ have the same null space (where $Q_\delta \equiv P_\delta - I$). Let the *system degree* $d \equiv \max_{\delta\in\Delta} d_\delta$.

2.2. Strong-present-value optimality. Suppose that rewards carried from one period to the next earn interest at the rate $100\rho\%$ $(\rho > 0)$, and let $\beta \equiv \frac{1}{1+\rho}$ be the *discount factor*. The *present value* V^ρ_δ of a policy δ is the (expected) present value of the rewards that δ earns in each period discounted to the beginning of period 0, i.e., $V^\rho_\delta \equiv \sum_{N=1}^\infty \beta^N P^{N-1}_\delta r_\delta$. A policy δ is *present-value optimal* if $V^\rho_\delta \ge V^\rho_\gamma$ for all $\gamma \in \Delta$. Finally, δ is *strong-present-value optimal* if it is present-value optimal for all sufficiently small ρ.

Blackwell [2] shows the existence of a stationary strong-present-value optimal policy for substochastic systems, and this theorem also holds for normalized systems. Hence, it suffices to restrict attention to stationary policies throughout this paper.

n-optimality. It is computationally challenging to discern directly whether or not a policy is strong-present-value optimal. However, building a sequence of n-optimal policies is more efficient and eventually attains strong-present-value optimality (when $n = S$). We define the concept of n-optimality in the remainder of this section.

A policy δ is *n-present-value optimal* if

$$\lim_{\rho\downarrow 0} \rho^{-n} \left(V^\rho_\delta - V^\rho_\gamma\right) \ge 0 \text{ for all } \gamma \in \Delta. \tag{2.1}$$

Evidently,

$$V^\rho_\delta = \beta r_\delta + \beta P_\delta V^\rho_\delta. \tag{2.2}$$

Rothblum [12] extends the Laurent expansion of Miller and Veinott [8] (for substochastic systems) to give, for small $\rho > 0$,

$$V^\rho_\delta = \sum_{n=-d}^{\infty} \rho^n v^n_\delta. \tag{2.3}$$

By substituting (2.3) into (2.2), multiplying by $1+\rho$, and equating coefficients of like powers of ρ, we see that $V^{n+d} \equiv \left(v^{-d}, \ldots, v^{n+d}\right) = \left(v_\delta^{-d}, \ldots, v_\delta^{n+d}\right)$ satisfies

$$r_\delta^j + Q_\delta v^j = v^{j-1}, \quad j = -d, \ldots, 0, \ldots, n+d, \tag{2.4}$$

where $Q_\delta = P_\delta - I$, $r_\delta^0 = r_\delta$, $r_\delta^j = 0, j \neq 0$, and $v_\delta^{-d-1} = 0$ [8, 12]. Conversely, if the matrix $V^{n+d} \equiv \left(V^n, v^{n+1}, \ldots, v^{n+d}\right)$ satisfies (2.4), then $V^n = V_\delta^n \equiv \left(v_\delta^{-d}, \ldots, v_\delta^n\right)$ [14, 12]. Thus, (2.4) uniquely determines the vector $V^n = V_\delta^n$, but not $v^{n+1}, \ldots, v^{n+d}$.

Writing $B \succeq C$ for two matrices of like dimension means that each row of $B - C$ is lexicographically nonnegative. From (2.1) and (2.3), a policy δ is n-present-value optimal if and only if δ is *n-optimal*, i.e., $V_\delta^n \succeq V_\gamma^n$ for all $\gamma \in \Delta$.

Denote the set of n-optimal policies by Δ_n and notice that the n-optimal sets are nested: $\Delta \supseteq \Delta_{-d} \supseteq \Delta_{-d+1} \supseteq \Delta_{-d+2} \supseteq \cdots$. Extending [8], [14] shows that there exists an $m \in [-1, S]$ such that $\Delta \supseteq \Delta_{-1} \supseteq \cdots \supseteq \Delta_m = \Delta_{m+1} = \cdots$ (for substochastic systems). Moreover, an S-optimal policy is strong-present-value optimal.

Rothblum [12] shows how to extend the policy improvements from Miller and Veinott [8] and Veinott [14] (for substochastic systems) to normalized systems, i.e., systems where the spectral radius of the transition matrix for every (stationary) policy does not exceed one. As systems with substochastic classes are normalized, Rothblum's policy improvement can be applied to these systems.

Hereafter, we consider only stationary policies in systems that have substochastic classes. We present a numerically stable method for finding V_δ^n for a given $\delta \in \Delta$ and $-d \leq n$. More generally, the method finds a solution of

$$c^j + Q_\delta v^j = v^{j-1}, \quad j = m+1, \ldots, n+d \tag{2.5}$$

given $\delta \in \Delta$, v^m and c^j, $j = m+1, \ldots, n+d$. Again, only v^j, $j = m+1, \ldots, n$ are uniquely determined.

3. Finding the Laurent coefficients. The communicating class decomposition of a policy (see section 2.1) induces a *dependence* partial ordering amongst the classes. A class $\mathcal{C}$ *depends* on another class $\mathcal{D}$ if there is some $s \in \mathcal{C}$, $t \in \mathcal{D}$, with $P_{\delta st} > 0$. (If additionally, $P_{\delta ts} > 0$, then $\mathcal{C}$ and $\mathcal{D}$ would be the same communicating class.) If a class $\mathcal{C}$ doesn't depend on any other class, then $\mathcal{C}$ is *independent*.

Bather [1] and Veinott [14] both use the dependence partial ordering to solve (2.5) for substochastic systems. In substochastic systems, all recurrent classes are independent, so one may solve (2.5) for these classes separately. Once the values of the independent (recurrent) classes are known, i.e., v_s^j for s in a recurrent class, these values may be incorporated into the linear equations (2.5) for classes that depend on the independent classes, and these linear equations may then be solved separately. By repeating this process, (2.5) may be solved for the entire system by solving (2.5) within each class (using any necessary values from other classes).

For systems with substochastic classes, it is not necessarily true that recurrent classes are independent and vice versa. However, one may still use the dependence partial ordering to solve (2.5) by solving (2.5) within each class as just described. Also, even though it may not be clear if a class is transient or recurrent, each class is substochastic (by definition) and irreducible (because every state within a class communicates with all other states in the class).

The remainder of this section presents a method for solving the linear system (2.5) within a single substochastic, irreducible class. Throughout, the notation for system parameters denotes those same parameters restricted to the class. Thus, $\mathcal{S}$

refers to the states within the class, P_δ refers to the transition matrix restricted to the states in the class, and so on.

Each class may be transient or recurrent. In a transient class, $\lim_{N\to\infty} P_\delta^N = 0$ $(C, 1)$, so $Q_\delta^{-1} \equiv (P_\delta - I)^{-1} = -\left(I + P_\delta + P_\delta^2 + \cdots\right)$ is well-defined (Q_δ is nonsingular). If the class is recurrent, then it must be *stochastic*, i.e., $\sum_{t\in\mathcal{S}} p(t|s, \delta_s) = 1$ for every $s \in \mathcal{S}$. Then the rows of Q_δ sum to zero, so Q_δ is singular. Since the class is irreducible, eliminating any (single) state s from the class causes the remainder to become transient. This is equivalent to removing the row corresponding to the state-action pair (s, δ_s) and the column corresponding to s from P_δ. Removing this row and column from Q_δ causes it to become nonsingular. Hence, Q_δ has rank $S - 1$ (it is *rank-deficient by one*).

3.1. Rank-revealing LU factors. Given a (substochastic, irreducible) class, one may deduce if it is transient or recurrent by means of an *RRLU factorization* of Q_δ. This takes the form

$$T_1 Q_\delta T_2^T = LU = \begin{pmatrix} \hat{L} & 0 \\ l^T & 1 \end{pmatrix} \begin{pmatrix} \hat{U} & u \\ 0 & \varepsilon \end{pmatrix}, \tag{3.1}$$

where T_1 and T_2 are permutations that must be chosen to limit the size of the off-diagonal elements of L and U. If $|\varepsilon|$ is suitably large, then Q_δ is taken to have full rank, but if $|\varepsilon| = O(\epsilon)$, where ϵ is the machine precision, Q_δ is regarded as singular (in this case, rank-deficient by one). The remainder of this subsection discusses possible factorizations that provide T_1, T_2, L, and U.

Our discussion is centered on LUSOL, a package for computing sparse LU factors of a square or rectangular sparse matrix [3, 9, 10, 6]. LUSOL produces an L with unit diagonals and a U that tends to reflect the condition of the original matrix Q_δ. As in several other such packages, T_1 and T_2 are chosen to maximize sparsity as much as possible, subject to a stability test at each step of the factorization.

The stability test is a function of two parameters $L_{\max}$ and $U_{\max}$ (both 1 or more). At the kth step, the next column of L and row of U must satisfy

$$\begin{aligned} |L_{ik}| &\le L_{\max}, & i > k, \\ |U_{kj}| &\le U_{\max}|U_{kk}|, & j > k. \end{aligned}$$

Adequate stability is usually achieved with *threshold partial pivoting* (TPP), in which $L_{\max} = 10$ or less, and $U_{\max} = \infty$ (so that only the subdiagonals of L are controlled). To improve the rank-revealing properties, both $L_{\max}$ and $U_{\max}$ must be finite and closer to 1. Values such as 4, 2, and 1.1 are increasingly likely to reveal rank correctly, while retaining some freedom to keep L and U sparse.

LUSOL has two RRLU options. Threshold rook pivoting (TRP) uses $L_{\max} = U_{\max} \le 4$ (say) and provides a good compromise between stability and efficiency. Threshold complete pivoting (TCP) additionally requires *all* elements in the remaining unfactored matrix to be bounded relative to $|U_{kk}|$ at each stage. This compromises the sparsity of L and U. To obtain reliable RRLU properties in our experiments, we have used TRP with $L_{\max} = U_{\max} = 2.0$. There is no need to use TCP.

Define the permuted Q_δ and the combined permutations from (3.1) as follows:

$$Q \equiv T_1 Q_\delta T_2^T \equiv \begin{pmatrix} \hat{Q} & \hat{q} \\ q^T & \varphi \end{pmatrix} = LU, \qquad T \equiv T_1 T_2^T \equiv \begin{pmatrix} \hat{T} & \hat{t} \\ t^T & \theta \end{pmatrix}, \tag{3.2}$$

where $\hat{Q} = \hat{L}\hat{U}$ is square and nonsingular and q, $\hat{q}$, t, $\hat{t}$, l, u are $(S-1)$-vectors. Regardless of the nature of the class (based on the size of ε), the LU factors may be used to solve (2.5) because the system is consistent even if Q_δ is singular.

Note: The methods presented in the next two sections are essentially those developed in the first author's thesis [11], where it was inadvertently assumed that $T_1 = T_2$ (and thus $T = I$). Here we treat T as a general matrix.

3.2. Transient classes. The linear system (2.5) is block triangular:

$$\begin{bmatrix} Q_\delta & & & \\ -I & Q_\delta & & \\ & \ddots & \ddots & \\ & & -I & Q_\delta \end{bmatrix} \begin{bmatrix} v^{m+1} \\ v^{m+2} \\ \vdots \\ v^{n+d} \end{bmatrix} = \begin{bmatrix} v^m - c^{m+1} \\ -c^{m+2} \\ \vdots \\ -c^{n+d} \end{bmatrix}. \tag{3.3}$$

If the RRLU factorization shows that Q_δ has full rank, the whole system (3.3) is nonsingular and may be solved by block forward substitution:

$$Q_\delta v^j = v^{j-1} - c^j, \quad j = m+1, \ldots, n+d. \tag{3.4}$$

3.3. Recurrent classes. Applying the permutations to (3.3) gives

$$\begin{bmatrix} Q & & & \\ -T & Q & & \\ & \ddots & \ddots & \\ & & -T & Q \end{bmatrix} \begin{bmatrix} w^{m+1} \\ w^{m+2} \\ \vdots \\ w^{n+d} \end{bmatrix} = \begin{bmatrix} b^{m+1} \\ b^{m+2} \\ \vdots \\ b^{n+d} \end{bmatrix}, \tag{3.5}$$

where $v^j = T_2^T w^j$ $(j = m+1, \ldots, n+d)$ and also $b^{m+1} = T_1(v^m - c^{m+1})$ and $b^j \equiv -T_1 c^j$ $(j = m+2, \ldots, n+d)$. If the rank-revealing LU (RRLU) factorization shows that Q_δ is rank-deficient by one, system (3.5) has additional structure:

$$\begin{bmatrix} \hat{Q} & \hat{q} & & & & & & & \\ q^T & \varphi & & & & & & & \oplus \\ -\hat{T} & -\hat{t} & \hat{Q} & \hat{q} & & & & & \\ -t^T & -\theta & q^T & \varphi & & & & & \\ & & \ddots & & \ddots & & & & \\ & & & & -\hat{T} & -\hat{t} & \hat{Q} & \hat{q} & \\ & & & & -t^T & -\theta & q^T & \varphi & \end{bmatrix} \begin{bmatrix} \hat{w}^{m+1} \\ w_S^{m+1} \\ \hat{w}^{m+2} \\ w_S^{m+2} \\ \vdots \\ \hat{w}^{n+d} \\ w_S^{n+d} \end{bmatrix} = \begin{bmatrix} \hat{b}^{m+1} \\ b_S^{m+1} \\ \hat{b}^{m+2} \\ b_S^{m+2} \\ \vdots \\ \hat{b}^{n+d} \\ b_S^{n+d} \end{bmatrix}, \tag{3.6}$$

where $\hat{w}^j$ and $\hat{b}^j$ represent the first $S-1$ elements of w^j and b^j, respectively, and $\oplus$ marks one row and column that reveal a rank-deficiency of one in the full system. The marked row is redundant as it comes from the first block of (3.5), which is singular but consistent. Also, since w^{n+d} is not determined uniquely, we may assign $w_S^{n+d} \equiv 0$ and make the marked column redundant. Removing the marked row and column

gives the following system, which has full rank and therefore a unique solution:

$$\begin{bmatrix} \hat{Q} & \hat{q} & & & & & & \\ -\hat{T} & -\hat{t} & \hat{Q} & \hat{q} & & & & \\ -t^T & -\theta & q^T & \varphi & & & & \\ & & \ddots & & \ddots & & & \\ & & & & -\hat{T} & -\hat{t} & \hat{Q} \\ & & & & -t^T & -\theta & q^T \end{bmatrix} \begin{bmatrix} \hat{w}^{m+1} \\ w_S^{m+1} \\ \hat{w}^{m+2} \\ w_S^{m+2} \\ \vdots \\ \hat{w}^{n+d} \end{bmatrix} = \begin{bmatrix} \hat{b}^{m+1} \\ \hat{b}^{m+2} \\ b_S^{m+2} \\ \vdots \\ \hat{b}^{n+d} \\ b_S^{n+d} \end{bmatrix}. \tag{3.7}$$

We now describe two methods for solving system (3.7).

3.4. Block LU method (BLU) for recurrent classes. Rearranging (3.7) gives the following nonsingular system:

$$\left[\begin{array}{ccccc|cccc} \hat{Q} & & & & & \hat{q} & & & \\ -\hat{T} & \hat{Q} & & & & -\hat{t} & \hat{q} & & \\ & \ddots & \ddots & & & & \ddots & \ddots & \\ & & -\hat{T} & \hat{Q} & & & & -\hat{t} & \hat{q} \\ & & & -\hat{T} & \hat{Q} & & & & -\hat{t} \\ \hline -t^T & q^T & & & & -\theta & \varphi & & \\ & \ddots & \ddots & & & & \ddots & \ddots & \\ & & -t^T & q^T & & & & -\theta & \varphi \\ & & & -t^T & q^T & & & & -\theta \end{array}\right] \left[\begin{array}{c} \hat{w}^{m+1} \\ \hat{w}^{m+2} \\ \vdots \\ \hat{w}^{n+d-1} \\ \hat{w}^{n+d} \\ \hline w_S^{m+1} \\ w_S^{m+2} \\ \vdots \\ w_S^{n+d-1} \end{array}\right] = \left[\begin{array}{c} \hat{b}^{m+1} \\ \hat{b}^{m+2} \\ \vdots \\ \hat{b}^{n+d-1} \\ \hat{b}^{n+d} \\ \hline b_S^{m+2} \\ \vdots \\ b_S^{n+d-1} \\ b_S^{n+d} \end{array}\right]. \tag{3.7'}$$

To derive a solution method, we label the components of (3.7') as

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \left[\begin{array}{c} \hat{w} \\ \hline w_S \end{array}\right] = \left[\begin{array}{c} \hat{b} \\ \hline b_S \end{array}\right]. \tag{3.8}$$

Since $\hat{Q}$ is nonsingular (and $\hat{Q} = \hat{L}\hat{U}$ has already been found), it is easy to solve a system $Ax = b$ sequentially. Hence a block LU factorization

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} A & \\ C & I \end{bmatrix} \begin{bmatrix} I & Y \\ & Z \end{bmatrix}$$

solves the full system efficiently. First, solve $AY = B$ and calculate the *Schur complement* $Z = D - CY$, then use block forward and backward substitution to solve (3.7') and hence (3.6):

$$\begin{bmatrix} A & \\ C & I \end{bmatrix} \begin{bmatrix} \hat{x} \\ x_S \end{bmatrix} = \begin{bmatrix} \hat{b} \\ b_S \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} I & Y \\ & Z \end{bmatrix} \begin{bmatrix} \hat{w} \\ w_S \end{bmatrix} = \begin{bmatrix} \hat{x} \\ x_S \end{bmatrix}. \tag{3.9}$$

The remainder of this section breaks down each step of the process.

Solving $AY = B$. Expanding A and B into block-matrix form gives

$$\begin{bmatrix} \hat{Q} & & & & \\ -\hat{T} & \hat{Q} & & & \\ & \ddots & \ddots & & \\ & & -\hat{T} & \hat{Q} & \\ & & & -\hat{T} & \hat{Q} \end{bmatrix} Y = \begin{bmatrix} \hat{q} & & & \\ -\hat{t} & \hat{q} & & \\ & \ddots & \ddots & \\ & & -\hat{t} & \hat{q} \\ & & & -\hat{t} \end{bmatrix}. \tag{3.10}$$

The first column of Y solves the block-diagonal system

$$\begin{bmatrix} \hat{Q} & & & & \\ -\hat{T} & \hat{Q} & & & \\ & \ddots & \ddots & & \\ & & -\hat{T} & \hat{Q} & \\ & & & -\hat{T} & \hat{Q} \end{bmatrix} \begin{bmatrix} y^{m+1} \\ y^{m+2} \\ \vdots \\ y^{n+d-1} \\ y^{n+d} \end{bmatrix} = \begin{bmatrix} \hat{q} \\ -\hat{t} \\ \vdots \\ 0 \\ 0 \end{bmatrix}. \tag{3.11}$$

The first two vectors in the solution satisfy $\hat{Q}y^{m+1} = \hat{q}$ and $\hat{Q}y^{m+2} = \hat{T}y^{m+1} - \hat{t}$. But $Qe = 0$ gives $\hat{Q}e = -\hat{q}$, so that $y^{m+1} = -e$, and then $\hat{Q}y^{m+2} = -\hat{T}e - \hat{t} = -[\hat{T} \;\; \hat{t}]e = -e$. Thus, we may use the factors $\hat{Q} = \hat{L}\hat{U}$ to solve (3.11) as follows:

$$\begin{aligned} y^{m+1} &= -e, \\ \hat{Q}y^{m+2} &= -e, \\ \hat{Q}y^{j} &= \hat{T}y^{j-1}, \quad j = m+3, \ldots, n+d. \end{aligned}$$

The solution of (3.10) is then given by the block-Toeplitz matrix

$$Y = \begin{bmatrix} y^{m+1} & & & \\ y^{m+2} & y^{m+1} & & \\ \vdots & y^{m+2} & \ddots & \\ y^{n+d-1} & \vdots & \ddots & y^{m+1} \\ y^{n+d} & y^{n+d-1} & \cdots & y^{m+2} \end{bmatrix}.$$

Forming $Z = D - CY$. With $\alpha_{m+j} \equiv -t^T y^{m+j} + q^T y^{m+j+1}$, the structure of C and Y gives

$$Z = \begin{bmatrix} -\theta & \varphi & & \\ & -\theta & \ddots & \\ & & \ddots & \varphi \\ & & & -\theta \end{bmatrix} - \begin{bmatrix} \alpha_{m+1} & q^T y^{m+1} & & \\ \alpha_{m+2} & \alpha_{m+1} & \ddots & \\ \vdots & \vdots & \ddots & q^T y^{m+1} \\ \alpha_{n+d-1} & \alpha_{n+d-2} & \cdots & \alpha_{m+1} \end{bmatrix}.$$

However, $\hat{Q}y^{m+1} = \hat{q}$ implies that $q^T y^{m+1} = \varphi$ (because Q_δ is rank-deficient by one). Hence Z is both triangular and Toeplitz:

$$Z = -\begin{bmatrix} \theta + \alpha_{m+1} & & & \\ \alpha_{m+2} & \theta + \alpha_{m+1} & & \\ \vdots & \vdots & \ddots & \\ \alpha_{n+d-1} & \alpha_{n+d-2} & \cdots & \theta + \alpha_{m+1} \end{bmatrix}. \tag{3.12}$$

Block forward substitution. The solution of $A\hat{x} = \hat{b}$ in (3.9) is found sequentially like the first column of Y in (3.11). Then $x_S = b_S - C\hat{x}$ may be calculated directly.

Block backward substitution. If the dimension of Z is large, the solution of $Zw_S = x_S$ may be found using special methods for (lower) triangular Toeplitz systems [13]. Otherwise, ordinary forward substitution suffices to find w_S, and then $\hat{w} = \hat{x} - Yw_S$. Now $w^{m+1}, w^{m+2}, \ldots, w^{n+d}$ (and hence $v^{m+1}, v^{m+2}, \ldots, v^{n+d}$) may be determined from $\hat{w}$ and w_S.

3.5. Veinott's method for recurrent classes. Veinott [14, p. 1651] gives a method for solving the singular linear system from a recurrent class in a substochastic system. In his method, $d = 1$ and the singular system is

$$\begin{bmatrix} Q & \\ -I & Q \end{bmatrix} \begin{bmatrix} w^{m+1} \\ w^{m+2} \end{bmatrix} = \begin{bmatrix} b^{m+1} \\ b^{m+2} \end{bmatrix}. \tag{3.13}$$

The notation in this section and the next is the same as in section 3.3, where $\hat{a}$ represents the first $S-1$ elements of a vector a and $\hat{A}$ represents the leading $(S-1)\times(S-1)$ submatrix of a matrix A. Veinott also defines Q' as the first $S-1$ columns of Q and (most importantly) defines

$$w^{m+1} = \begin{bmatrix} \tilde{w}^{m+1} \\ 0 \end{bmatrix} + w_S^{m+1} e, \qquad w^{m+2} = \begin{bmatrix} \tilde{w}^{m+2} \\ 0 \end{bmatrix} + w_S^{m+2} e.$$

Substituting into (3.13) and setting $w_S^{m+2} = 0$ gives

$$\begin{bmatrix} Q' & & \\ -I & -e & Q' \end{bmatrix} \begin{bmatrix} \tilde{w}^{m+1} \\ w_S^{m+1} \\ \tilde{w}^{m+2} \end{bmatrix} = \begin{bmatrix} b^{m+1} \\ b^{m+2} \end{bmatrix}.$$

Veinott uses Gaussian elimination to solve $Q'\tilde{w}^{m+1} = b^{m+1}$ and observes that the elimination reduces the last row of Q' to zero so that $\tilde{w}^{m+1}$ is uniquely determined. He then applies the same elimination steps to

$$\begin{bmatrix} -e & Q' \end{bmatrix} \begin{bmatrix} w_S^{m+1} \\ \tilde{w}^{m+2} \end{bmatrix} = b^{m+2}$$

and notes that w_S^{m+1} is uniquely determined by the last row.

3.6. Extended Veinott's method (EVM). We now extend Veinott's method in two ways: (1) to solve for systems with $d > 1$, and (2) to incorporate the improved stability of an RRLU.

To solve (2.5), we first compute the RRLU factors of Q_δ to obtain L, U, and T as in (3.1)–(3.2). Next, we remove the row and column marked $\oplus$ in (3.6) to obtain system (3.7) as before. (This is equivalent to Veinott setting $w_S^{m+2} = 0$ and

observing that the last row of Q' vanishes, but we have chosen the row and column more carefully.) Next, we substitute

$$\begin{bmatrix} \hat{w}^j \\ w_S^j \end{bmatrix} = \begin{bmatrix} I & e \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{w}^j \\ w_S^j \end{bmatrix} \tag{3.14}$$

(this is equivalent to Veinott's variable change) to get

$$\begin{bmatrix} \hat{Q} & 0 & & & & & \\ -\hat{T} & -e & \hat{Q} & 0 & & & \\ -t^T & -1 & q^T & 0 & & & \\ & & \ddots & & \ddots & & \\ & & & & -\hat{T} & -e & \hat{Q} \\ & & & & -t^T & -1 & q^T \end{bmatrix} \begin{bmatrix} \tilde{w}^{m+1} \\ w_S^{m+1} \\ \tilde{w}^{m+2} \\ w_S^{m+2} \\ \vdots \\ \tilde{w}^{n+d} \end{bmatrix} = \begin{bmatrix} \hat{b}^{m+1} \\ \hat{b}^{m+2} \\ b_S^{m+2} \\ \vdots \\ \hat{b}^{n+d} \\ b_S^{n+d} \end{bmatrix}. \tag{3.15}$$

Defining

$$T' \equiv \begin{bmatrix} \hat{T} \\ t^T \end{bmatrix}, \quad E \equiv \begin{bmatrix} 0 & T' \end{bmatrix}, \quad F \equiv \begin{bmatrix} -e & \hat{Q} \\ -1 & q^T \end{bmatrix} = \begin{bmatrix} -e & Q' \end{bmatrix}, \quad x^j \equiv \begin{bmatrix} w_S^{j-1} \\ \tilde{w}^j \end{bmatrix}$$

gives the block-triangular system

$$\begin{bmatrix} \hat{Q} & & & & \\ -T' & F & & & \\ & -E & F & & \\ & & \ddots & \ddots & \\ & & & -E & F \end{bmatrix} \begin{bmatrix} \tilde{w}^{m+1} \\ x^{m+2} \\ x^{m+3} \\ \vdots \\ x^{n+d} \end{bmatrix} = \begin{bmatrix} \hat{b}^{m+1} \\ b^{m+2} \\ b^{m+3} \\ \vdots \\ b^{n+d} \end{bmatrix}. \tag{3.15'}$$

We solve (3.15') using block forward substitution with an LU factorization of F, which may be obtained from a sparse Bartels–Golub update of the factors of Q (since F is a permutation of Q with a column replaced by e). Once $\tilde{w}^{m+1}$ and x^j $(j = m+2, \ldots, n+d)$ have been computed, $\tilde{w}^j$ $(j = m+2, \ldots, n+d)$ and w_S^j $(j = m+1, \ldots, n+d-1)$ may be recovered, and w^j $(j = m+1, \ldots, n+d)$ is calculated with a single addition. Now, $v^j = T_2^T w^j$ $(j = m+1, \ldots, n+d)$.

4. Example. Consider the system defined in Table 4.1 and represented graphically in Figure 4.1.

Also consider a policy δ in this system, with reward vector r_δ and transition matrix P_δ as follows:

$$\delta = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \qquad r_\delta = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \qquad P_\delta = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}.$$

The policy is depicted in Figure 4.2, where we can see that the communicating classes under δ are $\{1, 2\}$ and $\{3, 4\}$. Both classes are substochastic. Also, it is clear from

TABLE 4.1
System description.

$s \in \mathcal{S}$	$a \in \mathcal{A}_s$	$r(s,a)$	$p(t\|s,a), t \in \mathcal{S}$ 1	2	3	4
1	1	1	$\frac{1}{2}$	$\frac{1}{2}$	1	0
	2	1	0	1	0	0
2	1	1	1	0	0	0
3	1	0	0	0	0	$\frac{1}{2}$
4	1	1	0	0	$\frac{1}{2}$	0
	2	0	0	0	0	1

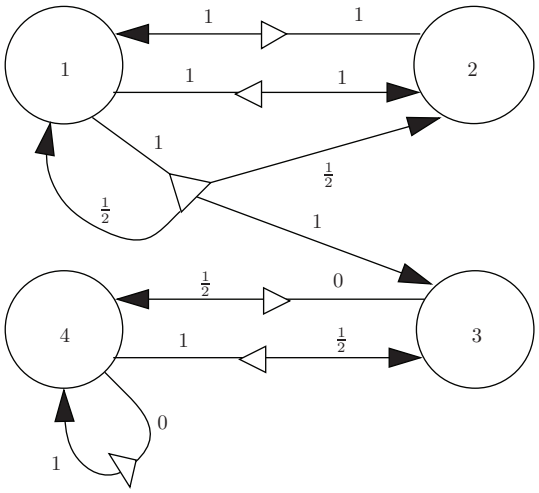

FIG. 4.1. *Graphical representation of system.*

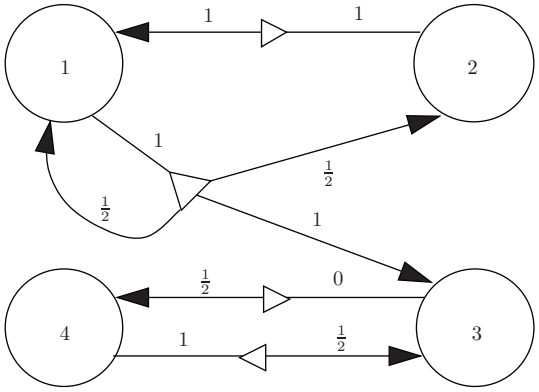

FIG. 4.2. *Graphical representation of δ.*

Figure 4.1 that any policy in this system has substochastic classes, i.e., the system has substochastic classes.

The degree of δ is 1, but the system degree is 2 (because a policy using action 2 instead of action 1 in state 4 has degree 2). Using d_δ instead of d in (2.4) may

decrease the computation required to find the Laurent coefficients for δ, but the amount of computation required to find d_δ is significant (and only slightly less than that required to find d). Therefore, it is computationally more efficient to calculate the system degree d at the start of policy improvement and use d in (2.4) throughout.

Finding v_δ^{-2} requires the solution of (2.4) for this system:

$$\begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & 1 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & -1 \end{bmatrix} v_\delta^{-2} \;=\; 0, \qquad \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & 1 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & -1 \end{bmatrix} v^{-1} \;=\; v_\delta^{-2},$$

and

$$\begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} + \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & 1 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & -1 \end{bmatrix} v^0 \;=\; v^{-1}.$$

Solving first for the independent class $\{3,4\}$, we have

$$Q_\delta = \begin{bmatrix} -1 & \frac{1}{2} \\ \frac{1}{2} & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} -1 & \frac{1}{2} \\ 0 & -\frac{3}{4} \end{bmatrix} \equiv LU$$

within the class. The LU factors are nonsingular, indicating a transient class. Therefore, finding v_δ^{-2} within $\{3,4\}$ requires the solution of (3.3):

$$\begin{bmatrix} -1 & \frac{1}{2} & & & & \\ \frac{1}{2} & -1 & & & & \\ & & -1 & \frac{1}{2} & & \\ & -I & \frac{1}{2} & -1 & & \\ & & & & -1 & \frac{1}{2} \\ & & & & \frac{1}{2} & -1 \end{bmatrix} \begin{pmatrix} v_{\delta 3}^{-2} \\ v_{\delta 4}^{-2} \\ v_3^{-1} \\ v_4^{-1} \\ v_3^0 \\ v_4^0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -r_{\delta 3} = 0 \\ -r_{\delta 4} = -1 \end{pmatrix},$$

and solving sequentially using the LU factors gives

$$\begin{pmatrix} v_{\delta 3}^{-2} \\ v_{\delta 4}^{-2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} v_3^{-1} \\ v_4^{-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} v_3^0 \\ v_4^0 \end{pmatrix} = \begin{pmatrix} \frac{2}{3} \\ \frac{4}{3} \end{pmatrix}.$$

The next class $\{1,2\}$ depends on $\{3,4\}$, whose values are known. Hence, v_δ^{-2} may be calculated within $\{1,2\}$. Restricting Q_δ to $\{1,2\}$ gives

$$Q_\delta = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \\ 1 & -1 \end{bmatrix}.$$

The rows of Q_δ are interchanged (for a marginal improvement in stability), giving the LU factorization

$$T_1 Q_\delta T_2 = LU, \quad T_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad T_2 = I, \quad L = \begin{bmatrix} 1 & \\ -\frac{1}{2} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & -1 \\ & 0 \end{bmatrix}.$$

The singularity of U indicates a recurrent class. Calculating v_δ^{-2} for $\{1,2\}$ therefore requires the solution of (3.6):

$$\begin{bmatrix} 1 & -1 & & & & \\ -\frac{1}{2} & \frac{1}{2} & & & & \\ & -1 & 1 & -1 & & \\ -1 & & -\frac{1}{2} & \frac{1}{2} & & \\ & & & -1 & 1 & -1 \\ & & -1 & & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{pmatrix} w_{\delta 1}^{-2} \\ w_{\delta 2}^{-2} \\ w_1^{-1} \\ w_2^{-1} \\ w_1^0 \\ w_2^0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -r_{\delta 2} = -1 \\ -v_3^0 - r_{\delta 1} = -\frac{5}{3} \end{pmatrix}$$

for which $\hat{Q} = 1$, $q^T = -\frac{1}{2}$, $\hat{q} = -1$, and $\varphi = \frac{1}{2}$. Eliminating the redundant row and arbitrary variable w_2^0 gives a slightly smaller nonsingular system corresponding to (3.7).

4.1. BLU. Rearranging as in (3.7') gives

$$\left[\begin{array}{ccc|cc} 1 & & & -1 & \\ & 1 & & -1 & -1 \\ & & 1 & & -1 \\ \hline -1 & -\frac{1}{2} & & & \frac{1}{2} \\ & -1 & -\frac{1}{2} & & \end{array}\right] \begin{pmatrix} w_{\delta 1}^{-2} \\ w_1^{-1} \\ w_1^0 \\ w_{\delta 2}^{-2} \\ w_2^{-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -r_{\delta 2} = -1 \\ 0 \\ -v_3^0 - r_{\delta 1} = -\frac{5}{3} \end{pmatrix}.$$

Now A, B, C, and D are known. Solving $AY = B$ requires the solution of

$$\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \begin{pmatrix} y_1^{-2} \\ y_1^{-1} \\ y_1^0 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ 0 \end{pmatrix}$$

by sequential use of the nonsingular LU factors $\hat{L} = 1$, $\hat{U} = 1$. Thus

$$y_1^{-2} = y_1^{-1} = -1, \quad y_1^0 = 0, \quad \text{and} \quad Y = \begin{bmatrix} -1 & \\ -1 & -1 \\ & -1 \end{bmatrix},$$

and then

$$Z = \begin{bmatrix} & \frac{1}{2} \\ & \end{bmatrix} - \begin{bmatrix} -1 & -\frac{1}{2} & \\ & -1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} -1 & \\ -1 & -1 \\ & -1 \end{bmatrix} = -\begin{bmatrix} \frac{3}{2} & \\ 1 & \frac{3}{2} \end{bmatrix}.$$

Block forward substitution solves

$$\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \begin{pmatrix} x_1^{-2} \\ x_1^{-1} \\ x_1^0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \qquad \Rightarrow \qquad x_1^{-2} = x_1^{-1} = 0, \quad x_1^0 = -1.$$

Then,

$$\begin{pmatrix} x_2^{-2} \\ x_2^{-1} \end{pmatrix} = \begin{pmatrix} 0 \\ -\frac{5}{3} \end{pmatrix} - \begin{bmatrix} -1 & -\frac{1}{2} & \\ & -1 & -\frac{1}{2} \end{bmatrix} \begin{pmatrix} x_1^{-2} = 0 \\ x_1^{-1} = 0 \\ x_1^0 = -1 \end{pmatrix} = \begin{pmatrix} 0 \\ -\frac{13}{6} \end{pmatrix}.$$

Finally, block backward substitution first solves

$$-\begin{bmatrix}\frac{3}{2} & \\ 1 & \frac{3}{2}\end{bmatrix}\begin{pmatrix}w_{\delta 2}^{-2}\\ w_2^{-1}\end{pmatrix}=\begin{pmatrix}x_2^{-2}=0\\ x_2^{-1}=-\frac{13}{6}\end{pmatrix}\quad\Rightarrow\quad\begin{pmatrix}w_{\delta 2}^{-2}\\ w_2^{-1}\end{pmatrix}=\begin{pmatrix}0\\ \frac{13}{9}\end{pmatrix}$$

and then calculates

$$\begin{pmatrix}w_{\delta 1}^{-2}\\ w_1^{-1}\\ w_1^{0}\end{pmatrix}=\begin{pmatrix}x_1^{-2}=0\\ x_1^{-1}=0\\ x_1^{0}=-1\end{pmatrix}-\begin{bmatrix}-1 & \\ -1 & -1\\ & -1\end{bmatrix}\begin{pmatrix}w_{\delta 2}^{-2}=0\\ w_2^{-1}=\frac{13}{9}\end{pmatrix}=\begin{pmatrix}0\\ \frac{13}{9}\\ \frac{4}{9}\end{pmatrix}.$$

Since $T_2 = I$, the v and w variables are the same.

4.2. EVM. Here we obtain $v_{\delta 1}^{-2}, v_1^{-1}, v_1^0$ within the class $\{1, 2\}$ following Veinott's approach. The system to be solved is (3.6):

$$\begin{bmatrix}1 & -1 & & & & \\ -\frac{1}{2} & \frac{1}{2} & & & & \\ & -1 & 1 & -1 & & \\ -1 & & -\frac{1}{2} & \frac{1}{2} & & \\ & & & -1 & 1 & -1\\ & & -1 & & -\frac{1}{2} & \frac{1}{2}\end{bmatrix}\begin{pmatrix}w_{\delta 1}^{-2}\\ w_{\delta 2}^{-2}\\ w_1^{-1}\\ w_2^{-1}\\ w_1^0\\ w_2^0\end{pmatrix}=\begin{pmatrix}0\\ 0\\ 0\\ 0\\ -r_{\delta 2}=-1\\ -v_3^0-r_{\delta 1}=-\frac{5}{3}\end{pmatrix}$$

for which the quantities in (3.15)–(3.15') are

$$\hat{Q}=1,\quad Q'=\begin{bmatrix}1\\ -\frac{1}{2}\end{bmatrix},\quad T'=\begin{bmatrix}0\\ 1\end{bmatrix},\quad E=\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix},\quad F=\begin{bmatrix}-1 & 1\\ -1 & -\frac{1}{2}\end{bmatrix}.$$

Setting $w_2^0 = 0$ gives the system

$$\begin{bmatrix}1 & & & & & \\ & -1 & 1 & & & \\ -1 & -1 & -\frac{1}{2} & & & \\ & & & -1 & 1 & \\ & & & -1 & -1 & -\frac{1}{2}\end{bmatrix}\begin{pmatrix}\tilde{w}_{\delta 1}^{-2}\\ w_{\delta 2}^{-2}\\ \tilde{w}_1^{-1}\\ w_2^{-1}\\ \tilde{w}_1^0\end{pmatrix}=\begin{pmatrix}0\\ 0\\ 0\\ -1\\ -\frac{5}{3}\end{pmatrix}.$$

Recall $\hat{L} = 1$, $\hat{U} = 1$, and the LU factors of F are

$$\tilde{L}=\begin{bmatrix}1 & \\ 1 & 1\end{bmatrix},\quad \tilde{U}=\begin{bmatrix}-1 & 1\\ & -\frac{3}{2}\end{bmatrix}.$$

Solving $\hat{L}\hat{U}\tilde{w}_1^{-2} = 0$ gives $\tilde{w}_1^{-2} = 0$, and solving

$$\begin{bmatrix}-1 & 1 & & \\ -1 & -\frac{1}{2} & & \\ & & -1 & 1\\ & -1 & -1 & -\frac{1}{2}\end{bmatrix}\begin{pmatrix}w_{\delta 2}^{-2}\\ \tilde{w}_1^{-1}\\ w_2^{-1}\\ \tilde{w}_1^0\end{pmatrix}=\begin{pmatrix}0\\ 0+\tilde{w}_1^{-2}=0\\ -1\\ -\frac{5}{3}\end{pmatrix}$$

sequentially using $\tilde{L}$ and $\tilde{U}$ gives $w_2^{-2} = \tilde{w}_1^{-1} = 0$, $w_2^{-1} = \frac{13}{9}$, and $\tilde{w}_1^0 = \frac{4}{9}$. Performing the appropriate additions gives $w_{\delta 1}^{-2} = w_{\delta 2}^{-2} = 0$, $w_1^{-1} = w_2^{-1} = \frac{13}{9}$, $w_1^0 = \frac{4}{9}$, and $w_2^0 = 0$. Again, $T_2 = I$ means the v and w variables are the same.

Therefore, irrespective of which method is used for the recurrent class $\{1, 2\}$,

$$v_\delta^{-2} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad v^{-1} = \begin{pmatrix} \frac{13}{9} \\ \frac{13}{9} \\ 0 \\ 0 \end{pmatrix}, \quad v^0 = \begin{pmatrix} \frac{4}{9} \\ 0 \\ \frac{2}{3} \\ \frac{4}{3} \end{pmatrix}$$

solves (2.4) for the example system, and the Laurent coefficients v_δ^{-2} of the policy δ have been found.

5. Numerical stability. For transient classes, the natural approach is to solve the block-triangular system (3.3) by block forward substitution, using the nonsingular Q_δ repeatedly as in (3.4). If Q_δ is ill-conditioned, any errors in solving with Q_δ will grow exponentially. One may reduce the effect by using a small *interval*, namely, $n + d - m$. This may be achieved by implementing policy improvement as suggested by Veinott [14]. By finding m-optimal policies for $m = -d, \ldots, n$ sequentially, Veinott maintains V_δ^{m-1} throughout and simply searches for $m, \ldots, (m + d)$-improvements (again in order). Finding an $(m + d)$-improvement requires V_δ^{m+d}, so the interval for (2.5) is at most $m + 2d - (m - 1) = 2d + 1$. Therefore, the smaller the degree of the system, the more reliable calculations become.

Note: If Q_δ is ill-conditioned, but not singular, (3.3) is intrinsically ill-conditioned, and the computed v^j will have error regardless of the numerical method used. Keeping d small is advisable until the policy improvement leads to a better conditioned Q_δ.

The RRLU factorization of Q_δ is essential for numerical computation. If a class is recurrent but the LU fails to identify the singularity, the factors of Q_δ will be extremely ill-conditioned, and computational errors will become prominent in the block forward substitution for solving (3.3).

If singularity is identified, the BLU method works with the factorization (3.7')–(3.8), which is stable as long as A is not almost singular and the elements of either B or C (or both) are not much larger than the biggest element of A. Denote these requirements by Property P1. It is not clear when Property P1 will hold, but it can be tested a priori.

The Schur complement Z (3.12) is lower triangular with constant diagonal elements $\theta + \alpha_{m+1}$. The condition of Z will be reasonable if that diagonal value is not significantly smaller in magnitude than the off-diagonal elements $\alpha_{m+2}, \ldots, \alpha_{n+d-1}$. Denote this state by Property P2.

If P1 and P2 both hold, we have a stable method solving a well-behaved problem. If P1 holds, the condition of Z reflects the condition of the original problem. In practice with block factorizations of this kind, a single iteration of iterative refinement [5] is likely to give acceptable accuracy in most cases (without the use of higher precision). If the refinement procedure declares failure, interval reduction would be necessary.

For the extended Veinott method, there is more assurance of stability because the triangular transformations (3.14) are well-conditioned and the block-triangular system (3.15) $\equiv$ (3.15′) accurately reflects the condition of system (3.7). This becomes evident in the following numerical results.

6. Numerical experiments. The BLU method and the EVM involve similar amounts of computation, but may differ in their numerical accuracy.

To obtain a valid statistical comparison of the methods, we performed some experiments using MATLAB 7.5.0.342 (R2007b) [7] with machine precision $\epsilon \approx 2 \times 10^{-16}$. We generated 100 random sparse linear systems (3.3) with Q_δ of order 100 and density $\approx 20\%$, c_0 nonzero, and the remaining $c_j = 0$. Fifty of the Q_δ were irreducible transient classes and 50 irreducible recurrent classes. Thus, half the systems (3.3) had Q_δ nonsingular, and the other half contained one singularity as in (3.6).

We implemented each method, BLU and EVM, with three different factorizations:

TPP LUSOL with threshold partial pivoting;

TRP LUSOL with threshold rook pivoting;

MLU MATLAB's sparse LU factorization: `[L,U,P,Q] = lu(A,thresh)`.

A further method **FULL** used MATLAB's \ (backslash) operator to solve the entire system (3.3) without any extra guidance. This is equivalent to MLU on the full system. MLU uses a TPP strategy. The threshold parameters for LUSOL and MLU were set to keep $|L_{ij}| \leq 2.0$, a fairly strict bound that favors stability over sparsity.[1] To estimate rank, LUSOL counts the number of diagonals that are small in absolute terms or relative to their own column:

$$|U_{jj}| \leq \epsilon^{2/3} \max(1, \|U_j\|_\infty)$$

and regards them as singularities. We applied the same test to MATLAB's LU factors.

We computed the Laurent coefficients in the test classes using each of the methods {BLU, EVM, FULL}, and for methods {BLU, EVM}, we used all three LU factorizations {TPP, TRP, MLU}. For each class $i = 1, 2, \ldots, 100$, we performed an experiment using method **M** and factorization **F** as follows:

> **Experiment.** We randomly permuted the states in class i 100 times (this corresponds to a symmetric permutation of the linear system but does not change the Laurent coefficient values). For each of the 100 permutations, we used method **M** and (if appropriate) factorization **F** to calculate the residual norms for (2.5): $\|c^j + Q_\delta v^j - v^{j-1}\|_\infty$, $j = -1, 0, \ldots, 6$, which should be zero for all eight j. For each v^j, we set the residual metric $\rho_{i,j}^{\mathbf{M},\mathbf{F}}$ to be the largest residual value for v^j over the 100 permutations. Thus, the residual metric is an estimate of the the poorest performance of method **M** with factorization **F** when finding coefficient v^j for class i.

During our experiments we encountered floating-point overflow when using FULL: An obvious failure by this method. We also consider a residual of $O(1)$ or higher to be a failure by a method/factorization because it means there are no correct digits in the computed v^j.

6.1. Comparison of residuals. From the residual metric values, we make the following observations.

[1] Actually, MATLAB's `[L,U,P,Q] = lu(A,thresh)` might not bound *all* elements of L. If A can be permuted so that the first group of rows is strictly upper triangular and the first group of remaining columns is strictly *lower* triangular, the latter columns may become part of L regardless of the size of their subdiagonals. This does not affect MLU's stability for solving square nonsingular systems such as we have here (because only permutations are involved), but it affects other applications that require L to be well-conditioned.

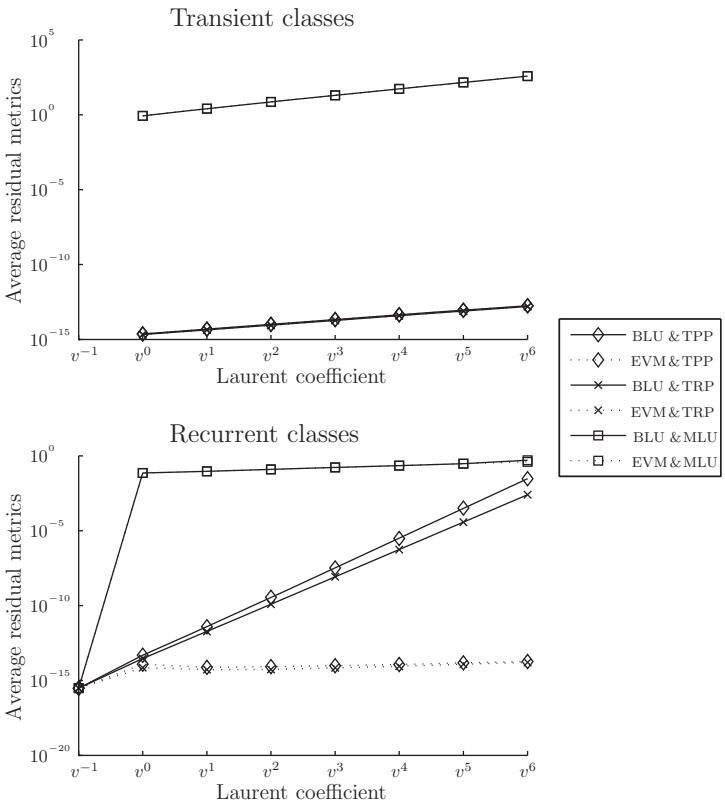

FIG. 6.1. *Largest residual norms.*

- For transient classes, the average residual metrics for FULL

$$\overline{\rho}_j^{\text{FULL}} \equiv \frac{1}{100}\sum_{i=1}^{100} \rho_{i,j}^{\text{FULL}}$$

were similar in magnitude to the average residual metrics for BLU and EVM with the LUSOL factorizations, namely, $\overline{\rho}_j^{\text{BLU, TPP}}$, $\overline{\rho}_j^{\text{BLU, TRP}}$, $\overline{\rho}_j^{\text{EVM, TPP}}$, $\overline{\rho}_j^{\text{EVM, TRP}}$, respectively (see Figure 6.1 top).
- For recurrent classes, FULL generated floating-point overflow on 39 of the 50 examples. This is to be expected because the MATLAB LU is not intended for singular systems. Even when the residual metrics could be calculated without overflow, their magnitude grew by a factor of $O(10^{16})$ for each j, from $O(10^{-15})$ when $j = -1$ to $O(10^{110})$ when $j = 6$. Thus (in general) FULL fails for recurrent classes except when $j = -1$.
- For transient classes, methods BLU and EVM perform similarly. Figure 6.1 (top) shows $\overline{\rho}_j^{\mathbf{M},\mathbf{F}}$ for $\mathbf{M} = \{\text{BLU, EVM}\}$, $\mathbf{F} = \{\text{TPP, TRP, MLU}\}$, and $j = -1, 0, \ldots, 6$. For $j = -1$, all residual metrics are numerically insignificant (actually less than machine precision and so not shown). For $j = 0$, the residual metrics are $O(10^{-15})$ with TPP or TRP, but $O(1)$ with MLU. For $j = 1$ to 6, the residual metrics increase at a moderate rate from their value for $j = 0$. Thus, for transient classes with methods BLU and EVM, the

residual metrics are numerically insignificant when TPP or TRP are used, but MLU fails for $j \geq 0$.

- For recurrent classes, methods BLU and EVM differ significantly. Figure 6.1 (bottom) shows insignificant average residual metrics for $j = -1$ for both methods. However,
  - For BLU with TPP and TRP, the residual metrics increase steadily as j increases, reaching $O(10^{-2})$ when $j = 6$ and thus becoming numerically significant. The effect is slightly more pronounced with TPP (more on this later).
  - For EVM with TPP and TRP, the residual metrics remain at $O(10^{-14})$. Any increase with j is very small, so a large number of coefficients could be calculated simultaneously before numerical error becomes a concern. Thus, EVM shows a definite advantage.
  - For MLU, the residual metrics for $j \geq 0$ are $O(10^{-1})$, so that MLU is bordering on failure for those j.

6.2. Pairwise comparisons. To identify differences in the methods and factorizations more accurately, we computed the average pairwise differences of the residual metrics for each j:

$$\overline{\psi}_j^{\mathbf{M},\mathbf{F},\mathbf{M}',\mathbf{F}'} \equiv \frac{1}{100}\sum_{i=1}^{100}\left(\rho_{i,j}^{\mathbf{M},\mathbf{F}} - \rho_{i,j}^{\mathbf{M}',\mathbf{F}'}\right).$$

Statistical analysis of these average differences led to the following observations.

Comparison of methods.

- For transient classes, there is no statistical evidence of a difference between BLU and EVM for any of the coefficients, regardless of the factorization used.
- For recurrent classes, the only statistical evidence of a difference between BLU and EVM appears for TPP and TRP. There is evidence that

$$\overline{\psi}_j^{\text{BLU, TPP, EVM, TPP}} \gg 0 \quad \text{and} \quad \overline{\psi}_j^{\text{BLU, TRP, EVM, TRP}} \gg 0, \ j = 0, 1, \ldots, 6.$$

  The difference is initially small but grows with j: see Figure 6.2. The trend indicates that numerical errors grow faster with BLU, and thus BLU will fail before EVM (i.e., at lower j). Note that the difference grows somewhat slower for TRP; thus BLU will stay "competitive" with EVM for longer with TRP. However, as the differences are positive and increasing, EVM is outperforming BLU in terms of average residual metrics, and the residual metrics deteriorate faster for BLU.

Comparison of factorizations.

- For transient classes, with both BLU and EVM there is statistical evidence of different factorizations giving different residual metrics: see Figure 6.3. For both methods, the difference TPP − TRP exists only for v^0 and v^1 and is $O(10^{-15})$ so is numerically insignificant. In contrast, and again for both BLU and EVM, the difference between MLU and the LUSOL factorizations TPP and TRP starts at $O(1)$ for v^j, $j = 0$ and increases steadily with j. This demonstrates that MLU will fail even when the LUSOL factorizations succeed and that the MLU residual metrics deteriorate faster than those for the LUSOL factorizations.

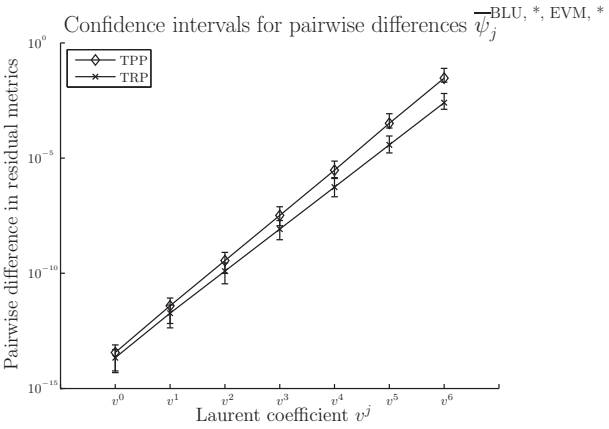

FIG. 6.2. *Comparing BLU to EVM using the difference in residuals; confidence intervals for recurrent classes.*

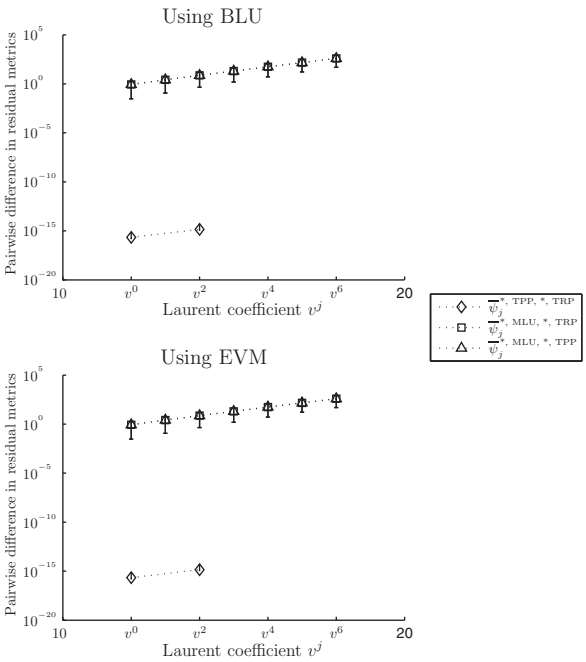

FIG. 6.3. *Comparing factorizations using the difference in residuals; confidence intervals for BLU and EVM on transient classes.*

- For recurrent classes, with both BLU and EVM there is evidence again of difference among the factorizations: see Figure 6.4.
  - When BLU is used, $\overline{\psi}_j^{\text{BLU, TPP, BLU, TRP}} > 0$ and exceeds $\sqrt{\epsilon}$ for $j \geq 3$, so that TRP outperforms TPP for $j \geq 3$, i.e., produces lower residuals. Also, the average difference is growing, so the TPP residual metrics are deteriorating at a greater rate. Further, the difference between the MATLAB and LUSOL factorizations is numerically significant for all coefficients except v^{-1}, with the LUSOL factorizations clearly outperforming those of MATLAB. However, note that the difference between

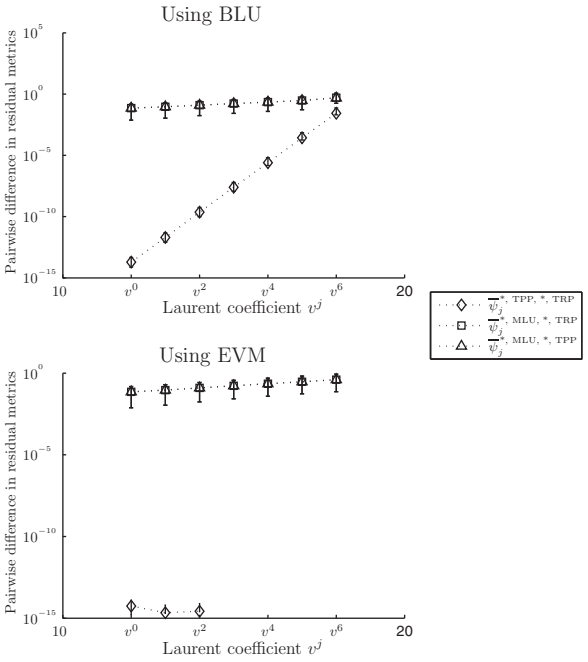

FIG. 6.4. *Comparing factorizations using the difference in residuals; confidence intervals for BLU and EVM on recurrent classes.*

TPP and TRP approaches the difference between the MLU and LUSOL factorizations. This indicates that the gap between the residual metrics with TPP and TRP (with BLU) is soon greater than the gap between the residual metrics with MLU and TRP. This is difficult to reconcile with the difference between MLU and TPP continuing to grow.

A look at the raw data shows some interesting cases that have caused this apparent anomaly. There is one matrix where TPP continues to perform well, $O(10^{-11})$ for $j = 6$, while both TRP and MLU deteriorate as usual. This causes the average difference of residual metrics between MLU and TPP to continue growing (as it "washes out" the difference of the other matrices). Conversely, there are several matrices where TRP gives residual metrics of $O(10^{-14})$, while TPP and MLU deteriorate as usual. This keeps the average difference between MLU and TRP growing and also causes the difference between TPP and TRP to grow (faster, as there are several matrices). More research is needed into the reason for the unusual performance of LUSOL on these matrices.

– When EVM is used, $\overline{\psi}_j^{\text{EVM, TPP, EVM, TRP}} > 0$ is statistically significant only for $j = 0, 1, 2$ and is not numerically significant. Also, the difference between the MATLAB and LUSOL factorizations is numerically significant and increasing (although less than that for BLU) for all coefficients except v^{-1}, so again the LUSOL factorizations clearly outperform the MATLAB factorizations.

6.3. Rank and computation time. From our experiments, it is unclear if TRP is required to identify the rank of a class correctly, or if TPP is sufficient. As the threshold parameter 2.0 is quite low (favoring stability over sparsity), all factorizations

seem to have determined the rank of Q_δ correctly. If we increase the threshold (thus sacrificing numerical stability to preserve sparsity), all threshold strategies become less able to determine the rank of a class. To preserve efficiency and reliability, experience in the context of sparse constrained optimization [4] suggests the use of TRP with $1.1 \leq \text{threshold} \leq 2.0$.

Finally, we compare the computation time for all approaches. We timed the computation of all coefficients for each of the 100 linear systems (ignoring permutation time, as permuting the system did not alter computation times). For the transient classes with both the BLU and EVM methods, the time for all factorizations was approximately 0.01 seconds. There was no statistical evidence of any difference among the factorizations with either of the methods. However, for the recurrent classes, there was statistical evidence of difference between BLU and EVM with each factorization. With BLU, the computation time was approximately 0.025 seconds; with EVM, it was approximately 0.01 seconds. FULL required approximately 1.21 seconds for the transient classes and 1.23 seconds for the recurrent classes.

7. Experiments with a real-world application. The numerical experiments from section 6 indicate that the EVM with either of the LUSOL factorizations (TPP or TRP) should ensure a numerically stable calculation of the Laurent coefficients for both transient and recurrent classes in general. However, the transition matrices for these classes were generated randomly; they did not come from a practical application that uses dynamic programming (DP). In this section we consider the performance of our methods on a real-world large-scale infinite-horizon DP problem.

Dr. R. Grinold (Barclays Global Investments, San Francisco) provided a large, sparse, substochastic DP problem from a portfolio management application. We used our policy improvement software to find a -1-optimal policy (the classical maximum-average-reward criterion). During policy improvement, many policies are considered, and each policy δ has a set of classes with corresponding submatrices of r_δ and Q_δ. We saved the submatrices of r_δ and Q_δ encountered during policy improvement and used this data as a testbed for further numerical experiments.

To check that our initial findings were valid, we again calculated the eight residual norms for (2.5) over the testbed: $\rho_j \equiv \|c^j + Q_\delta v^j - v^{j-1}\|_\infty$, $j = -1, 0, \ldots, 6$, which should be zero for all j.

Figure 7.1 shows the residuals for all the methods and factorizations. The pattern for all combinations is the same: $O(10^{-20})$ residuals for v^j, $j = -1$ and increasing as j increases to 6. The residuals also increase (although to a lesser extent) as the dimensions of Q_δ increase. The residuals for the LUSOL factorizations increase at the slowest rate, sometimes reaching $O(1)$ and hence failure when $j = 6$. The residuals for the MATLAB factorization MLU typically reach $O(10^{20})$ and in many cases, MLU fails for $j \geq 0$. One class showed extreme deterioration, reaching $O(10^{97})$ for $j = 6$.

For these real-world classes, FULL produces the best residuals and does not produce any exceptional errors (in contrast to the results from the previous numerical experiments). We also note that the residuals for large systems tend to be smaller with EVM than with BLU. These results confirm our previous conclusion that EVM with either TPP or TRP is the preferred approach.

Next, we compared the classes and coefficients where factorizations failed. In general, MLU fails much earlier than either of the LUSOL factorizations (indicated by the high residuals). The LUSOL factorizations usually fail at the same coefficient, although for some of the large systems, the LUSOL factorizations with BLU fail before the same factorization with EVM.

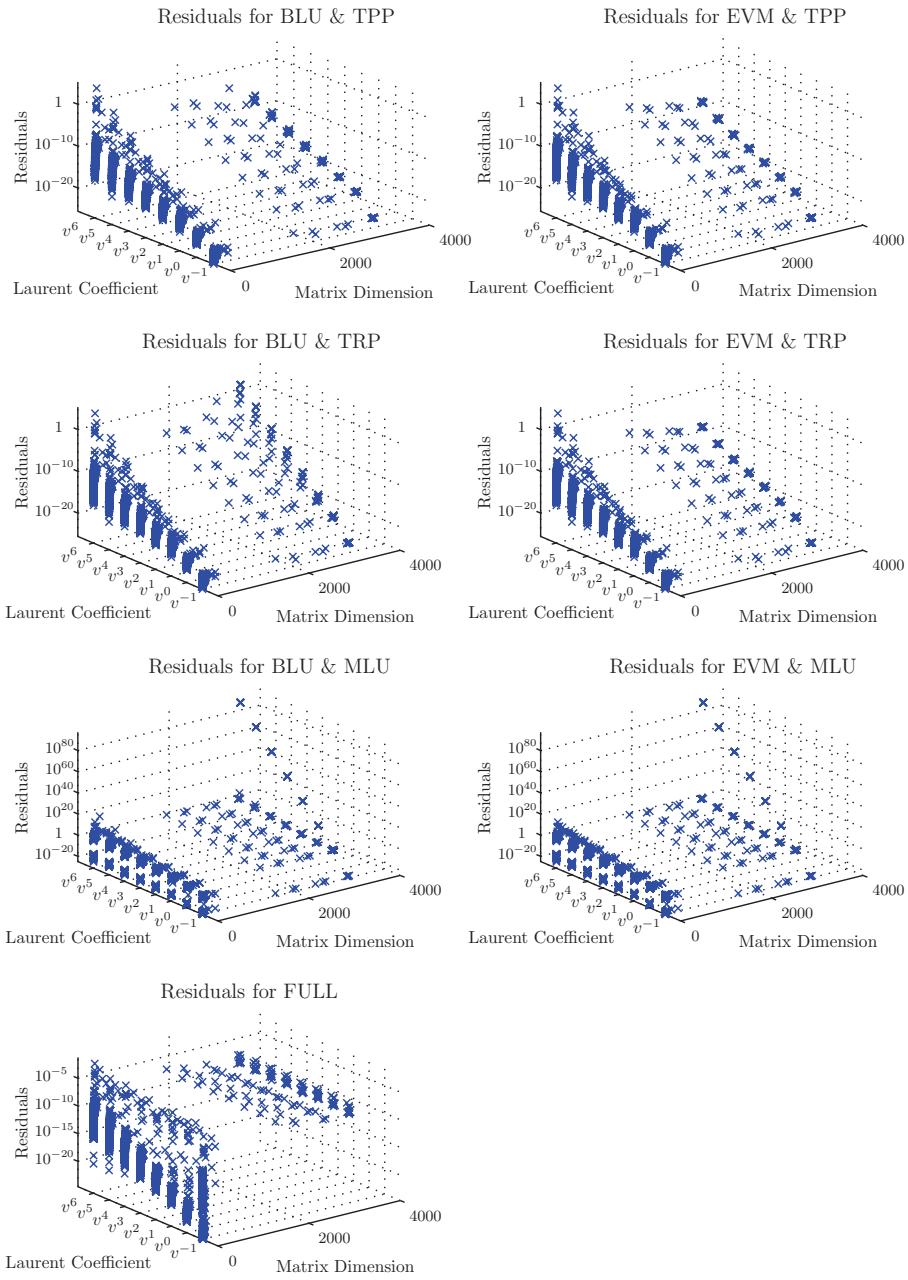

FIG. 7.1. *Residuals from real-world classes (both transient and recurrent).*

For both the BLU and EVM methods, we performed a pairwise comparison of the TPP, TRP, and MLU factorizations, but we removed the classes where factorizations failed, in order to judge the performance of successful factorizations. Figure 7.2 shows the differences on a log scale (one for positive differences and one for negative differences). The plots show that the LUSOL factorizations are generally superior to MLU, and the difference grows as j increases. There are some classes where MLU outperforms TPP and TRP, but the numerical advantage is small: $O(10^{-10})$. The

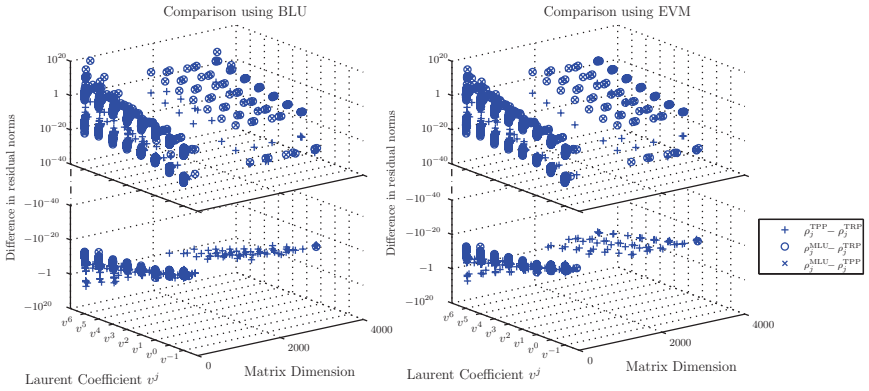

FIG. 7.2. *Comparing differences in residual norms between factorizations on a log scale.*

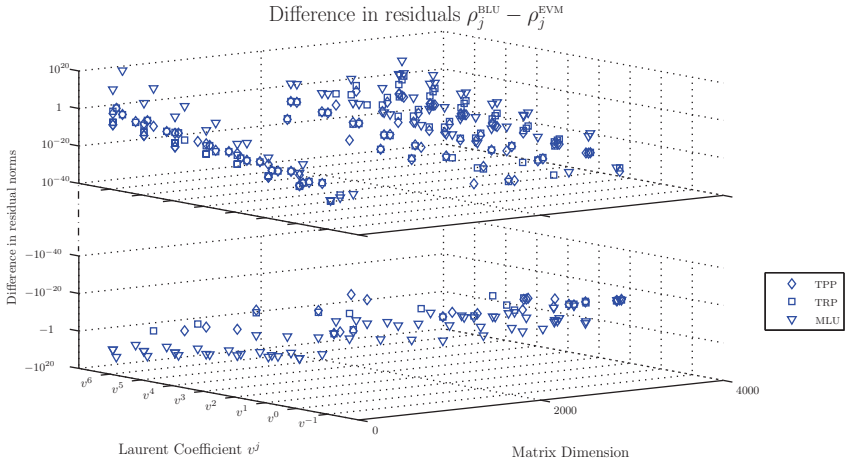

FIG. 7.3. *Comparing BLU to EVM using the difference in residuals on a log scale.*

plots also show that there are many classes where TPP outperforms TRP but in a majority of cases, the numerical advantage is small: $O(10^{-10})$. However, there are a limited number of classes where TPP outperforms TRP and the numerical advantage is increasing. These cases correspond to the raw data anomalies encountered in section 6. For these systems, the residuals produced by TPP are not only better than those from TRP but they deteriorate at a slower rate. More research is needed to determine why TPP outperforms TRP on these systems.

We also consider a pairwise comparison of the BLU and EVM methods using TPP, TRP, and MLU. Figure 7.3 shows the differences in residual norms on a log scale (one plot for positive values and one for negative values). The plots indicate that EVM generally produces smaller residuals that deteriorate more slowly. However, with MLU, there are several systems where the BLU residuals are better than those from EVM and deteriorate more slowly. This is also true for the LUSOL factorizations, although there are fewer cases where BLU performs better.

Finally, Figure 7.4 illustrates the computation time required for each of the solution approaches. It is clear that FULL takes much longer than the specialized

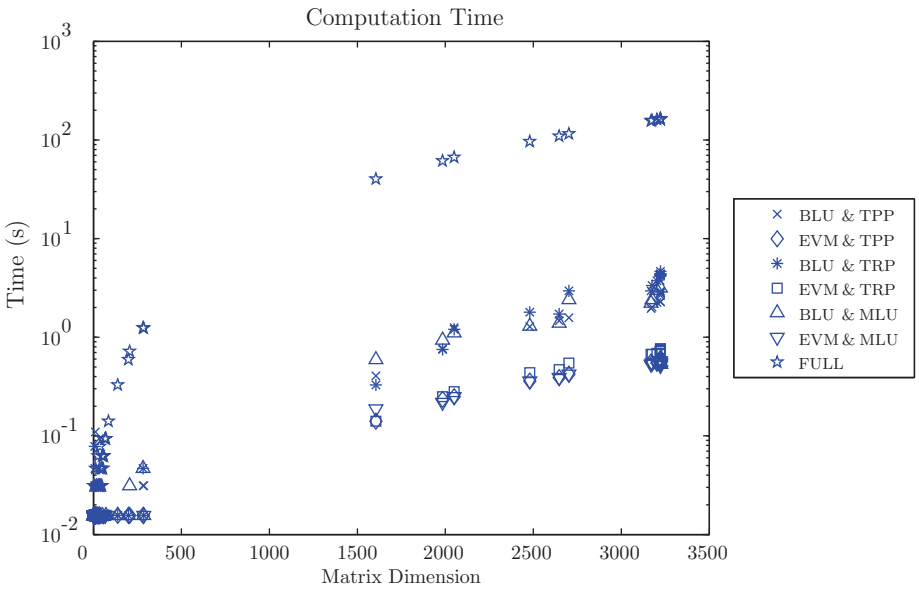

FIG. 7.4. *Computation time for real-world classes.*

methods. As the size of the class grows, EVM becomes the preferred method with any of the factorizations.

Summarizing the results from our experiments on real-world classes:

1. The results for FULL contradict the results from the other experiments. FULL never failed to calculate a coefficient and did not overflow. The Barclays problem gives systems with a diagonal band of 5 nonzeros, which could lead to exponential growth in the LU factors if the threshold parameter were not sufficiently low, yet the difficulties encountered with FULL in the random-data experiments did not surface. However, the computation time for FULL is prohibitive for larger systems.
2. In general, EVM with TRP is the preferred method. However, there are some anomalous cases where TPP outperforms TRP (with both BLU and EVM) and where BLU outperforms EVM (even when using TRP). These anomalies need to be examined in future research.

8. Contributions. Veinott [14] originally used the recurrent class decomposition with repeated Gaussian elimination to solve for the Laurent expansion coefficients of a substochastic system. For singular systems, he removes the last column from the system and notes that Gaussian elimination generates an empty last row, leaving a nonsingular upper triangular matrix.

Here we have presented BLU, a new method for computing the Laurent coefficients of a system with substochastic classes (although the entire system may not be substochastic). The method follows Veinott and Bather by using dependence partial ordering to decompose the problem into a sequence of computations on irreducible, substochastic systems. With the help of an RRLU factorization for each of these systems, BLU identifies transient and recurrent classes. Also, the coefficients for the transient classes are found sequentially with the LU factors. For recurrent classes, BLU uses the LU factors and a block LU decomposition to find the coefficients in a way that has proved stable for some systems but not all.

In search of greater reliability, we revisited Veinott's method for calculating the Laurent coefficients for a substochastic system. We extended his idea of repeated use of Gaussian elimination to solve systems with substochastic classes. We also implemented the Gaussian elimination using sparse LU factorization with various types of threshold pivoting to provide stability. For recurrent classes, we follow Veinott's method (which removes the last column of the singular matrix), but only after applying any permutations from the LU factorization; see (3.15). The resulting method (EVM) has proved to be extremely reliable.

Applying a stable LU factorization to the entire system (3.3), FULL, produces poor residuals for randomized systems, sometimes resulting in numerical overflow. In contrast, FULL produces good residuals on sample systems from a real-world example. However, the computation time for FULL is too great for it to be considered a viable option for large stochastic DP problems. Some sort of block factorization (with repeated use of factors of small matrices) is certain to be more effective.

The BLU method is one such approach, but for maximum reliability and the same efficiency, it is clear that EVM should be used.

LUSOL's TRP (rook pivoting) factorization increases the chance of detecting singular systems correctly if used with a threshold parameter sufficiently close to 1. We conclude that by incorporating EVM into policy improvement and employing sparse LU factors that favor stability over sparsity, we should be able to deal successfully with the large models that arise in the sustainable management of resources.

Acknowledgments. We thank Arthur F. Veinott, Jr. for sharing his knowledge of substochastic systems and for encouraging the development of reliable solution methods. We also thank Cameron G. Walker for his input into the statistical analysis of our numerical experiments. Two referees provided further valuable help with their insightful comments and questions.

REFERENCES

[1] J. A. Bather, *Optimal decision procedures for finite Markov chains. Part* III: *General convex systems*, Adv. Appl. Prob., 5 (1973), pp. 541–558.
[2] D. Blackwell, *Discrete dynamic programming*, Ann. Math. Statist., 33 (1962), pp. 719–726.
[3] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *Maintaining LU factors of a general sparse matrix*, Linear Algebra Appl., 88/89 (1987), pp. 239–270.
[4] P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM Rev., 47 (2005), pp. 99–131.
[5] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, PA, 2002.
[6] LUSOL, *Sparse LU factorization package*, http://www.stanford.edu/group/SOL/software.html.
[7] Matlab, *Matrix laboratory*, http://www.mathworks.com.
[8] B. L. Miller and A. F. Veinott, Jr., *Discrete dynamic programming with a small interest rate*, Ann. Math. Statist., 40 (1969), pp. 366–370.
[9] M. J. O'Sullivan and M. A. Saunders, *Sparse rank-revealing LU factorization via threshold complete pivoting and rook pivoting*, presented at Householder Symposium XV on Numerical Linear Algebra, Peebles, Scotland, 2002, http://www.stanford.edu/group/SOL/talks.html.
[10] M. J. O'Sullivan and M. A. Saunders, *LUSOL: A basis package for constrained optimization*, presented at IFORS triennial conference on OR/MS, Honolulu, HI, 2005. http://www.stanford.edu/group/SOL/talks.html.
[11] M. J. O'Sullivan, *New Methods for Dynamic Programming Over an Infinite Time Horizon*, Ph.D. thesis, Dept. of Management Science and Engineering, Stanford University, 2002.
[12] U. G. Rothblum, *Normalized Markov decision chains* I: *Sensitive discount optimality*, Oper. Res., 23 (1975), pp. 785–795.

[13] W. F. Trench, *A note on solving nearly triangular Toeplitz systems*, Linear Algebra Appl., 93 (1987), pp. 57–65.

[14] A. F. Veinott, Jr., *Discrete dynamic programming with sensitive discount optimality criteria*, Ann. Math. Statist., 40 (1969), pp. 1635–1660.

[15] A. F. Veinott, Jr., *Markov decision chains*, in Studies in Optimization, MAA Stud. Math., 10, B. C. Eaves and G. B. Dantzig, eds., Mathematical Association of America, Washington, DC, 1974, pp. 124–159.