

SYSTEMS OPTIMIZATION LABORATORY  
DEPARTMENT OF MANAGEMENT SCIENCE AND ENGINEERING  
STANFORD UNIVERSITY  
STANFORD, CALIFORNIA 94305-4026

**Stabilizing Policy Improvement for Large-Scale  
Infinite-Horizon Dynamic Programming**

by

Michael J. O'Sullivan and Michael A. Saunders

TECHNICAL REPORT SOL 2006-1

March 2006

Submitted to *SIAM Journal on Matrix Analysis and Applications*, March 2006

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsors.

Reproduction in whole or in part is permitted for any purposes of the United States Government. This document has been approved for public release and sale; its distribution is unlimited.

# 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 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 [13] for substochastic systems.

**1. Introduction.** A current focus of many industries is *sustainability*—ensuring that the resources of the industry 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 sub-optimal when compared to a policy that maintains (or renews) the resource over time.

Dynamic programming (DP) 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 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.

---

\*Department of Engineering Science, University of Auckland, Auckland, New Zealand (michael.osullivan@auckland.ac.nz). 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 94305-4026 (saunders@stanford.edu). Partially supported by National Science Foundation grants CCR-9988205 and CCR-0306662, and Office of Naval Research grants N00014-96-1-0274 and N00014-02-1-0076.

Rothblum [11] notes that Blackwell’s existence theorem for substochastic systems extends to normalized systems and then generalizes the methods from Miller and Veinott [7] and Veinott [13, 14] 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 [13] except in the number of arbitrary variables).

Veinott [13, 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 [11] 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 [13] 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 [13]). 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 [13] 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, §3 gives a method for finding the Laurent expansion coefficients within each (substochastic, irreducible) class. We also present Veinott’s [13] method for stochastic, irreducible systems, and extend it to systems with system degree  $d > 1$ . An example illustrating both methods is given in §4. Section 5 discusses the stability and computational efficiency of the methods, and §6 presents numerical comparisons.

**2. Preliminaries.** Consider a *general system* observed in periods  $1, 2, \dots$ . 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  $P_\delta$  (corresponding to policy  $\delta$ ) defines a set of communicating classes as follows. A state  $s$  *communicates* with another state  $t$  if there exists some  $N > 0$  such that  $P_{\delta_{st}}^N > 0$ . 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 either *transient* (the long-run probability of being in  $\mathcal{C}$  is 0) or *recurrent* (the long-run probability of being in  $\mathcal{C}$  is positive).

If every class  $\mathcal{C}$  (under  $\delta$ ) has  $0 \leq p(t|s, a) \leq 1$  and  $\sum_{t \in \mathcal{C}} p(t|s, a) \leq 1$ ,  $s, t \in \mathcal{C}$ ,  $a \in \mathcal{A}_s$ , then  $\delta$  has *substochastic classes*. Also, since the blocks of  $P_\delta$  corresponding to these classes lie on the diagonal and have spectral radius not exceeding one,  $P_\delta$  has spectral radius not exceeding one. If every (stationary) policy has substochastic classes, then 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  $\delta$ , let the *degree* of  $\delta$  be the smallest nonnegative integer  $d_\delta \equiv i$  such that  $Q_\delta^i$  and  $Q_\delta^{i+1}$  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*  $V_\delta^\rho$  of a policy  $\delta$  is the (expected) present value of the rewards that  $\delta$  earns in each period discounted to the beginning of period 0, i.e.,  $V_\delta^\rho \equiv \sum_{N=1}^{\infty} \beta^N P_\delta^{N-1} r_\delta$ . A policy  $\delta$  is *present-value optimal* if  $V_\delta^\rho \geq V_\gamma^\rho$  for all  $\gamma \in \Delta$ . Finally,  $\delta$  is *strong-present-value optimal* if it is present-value optimal for all sufficiently small  $\rho$ .

Blackwell [2] shows the existence of a stationary strong-present-value optimal policy for substochastic systems, and this theorem also holds for normalized systems. 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  $n = S$ ).

A policy  $\delta$  is *n-present-value optimal* if

$$\lim_{\rho \downarrow 0} \rho^{-n} (V_\delta^\rho - V_\gamma^\rho) \geq 0 \text{ for all } \gamma \in \Delta. \quad (2.1)$$

Evidently

$$V_\delta^\rho = \beta r_\delta + \beta P_\delta V_\delta^\rho. \quad (2.2)$$

Rothblum [11] extends the Laurent expansion of Miller and Veinott [7] (for substochastic systems) to give

$$V_\delta^\rho = \sum_{n=-d}^{\infty} \rho^n v_\delta^n \quad (2.3)$$

in small  $\rho > 0$ . By substituting (2.3) into (2.2), multiplying by  $1+\rho$  and equating coefficients of like powers of  $\rho$ , we see that  $V^{n+d} \equiv (v^{-d}, \dots, v^{n+d}) = (v_\delta^{-d}, \dots, v_\delta^{n+d})$  satisfies

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

where  $Q_\delta = P_\delta - I$ ,  $r_\delta^0 = r_\delta$ ,  $r_\delta^j = 0, j \neq 0$ , and  $v_\delta^{-d-1} = 0$  [7, 11]. Conversely, if the matrix  $V^{n+d} \equiv (V^n, v^{n+1}, \dots, v^{n+d})$  satisfies (2.4) then  $V^n = V_\delta^n \equiv (v_\delta^{-d}, \dots, v_\delta^n)$  [13, 11]. Thus, (2.4) uniquely determines the vector  $V^n = V_\delta^n$ , but not  $v^{n+1}, \dots, v^{n+d}$ .

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  $\delta$  is *n-present-value optimal* if and only if  $\delta$  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  $\Delta_n$  and notice that the  $n$ -optimal sets are nested, i.e.,  $\Delta \supseteq \Delta_{-d} \supseteq \Delta_{-d+1} \supseteq \Delta_{-d+2} \supseteq \dots$ . Extending [7], [13] shows that there exists an  $m \in [-1, S]$  such that  $\Delta \supseteq \Delta_{-1} \supseteq \dots \supseteq \Delta_m = \Delta_{m+1} = \dots$  (for substochastic systems). Moreover, an  $S$ -optimal policy is strong-present-value optimal.

Rothblum [11] shows how to extend the policy improvements from Miller and Veinott [7] and Veinott [13] (for substochastic systems) to normalized systems (thus systems with substochastic classes).

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

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

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

**3. Finding the Laurent coefficients.** The communicating class decomposition of a policy (see §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 [13] 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,  $P_\delta$  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,  $\lim_{N \rightarrow \infty} P_\delta^N = 0$ , so  $Q_\delta^{-1} \equiv (P_\delta - I)^{-1} = -(I + P_\delta + P_\delta^2 + \dots)$  is well-defined ( $Q_\delta$  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  $Q_\delta$  sum to zero, so  $Q_\delta$  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  $(s, \delta_s)$  and the column corresponding to  $s$  from  $P_\delta$ . Removing this row and column from  $Q_\delta$  causes it to become nonsingular. Hence,  $Q_\delta$  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 a *rank-revealing LU (RRLU) factorization* of  $Q_\delta$ . This takes the form

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

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  $Q_\delta$  is taken to have full rank, but if  $|\varepsilon| = O(\epsilon)$  where  $\epsilon$  is the machine precision,  $Q_\delta$  is regarded as singular (in this case, rank-deficient by one).

Our discussion is centered on LUSOL, a package for computing sparse LU factors of a square or rectangular sparse matrix [4, 9, 10]. LUSOL produces an  $L$  with unit diagonals and a  $U$  that tends to reflect the condition of the original matrix  $Q_\delta$ . 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  $k$ th step, the next column of  $L$  and row of  $U$  must satisfy

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

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  $L_{\max} = U_{\max} \leq 4$  (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. To obtain reliable RRLU properties in our experiments, we have used TCP with  $L_{\max} = U_{\max} = 2.0$ .

Define the permuted  $Q_\delta$  and the combined permutations from (3.1) as follows:

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

where  $\hat{Q} = \hat{L}\hat{U}$  is an  $(S-1) \times (S-1)$  nonsingular matrix, 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  $\varepsilon$ ), the LU factors may be used to solve (2.5) because the system is consistent even if  $Q_\delta$  is singular.

Note: The methods presented in the next two sections are essentially those developed in the first author's thesis [8], 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:

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

If the rank-revealing LU (RRLU) factorization shows that  $Q_\delta$  has full rank, the whole system (3.3) is nonsingular and may be solved by block forward substitution:

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

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

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

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

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

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:

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

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:

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

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

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

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

$$\left[ \begin{array}{cc} A & B \\ C & D \end{array} \right] = \left[ \begin{array}{c} A \\ C \end{array} \right] \left[ \begin{array}{cc} I & Y \\ & Z \end{array} \right]$$

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):

$$\left[ \begin{array}{c} A \\ C \end{array} \right] \left[ \begin{array}{c} \hat{x} \\ x_S \end{array} \right] = \left[ \begin{array}{c} \hat{b} \\ b_S \end{array} \right] \quad \text{and} \quad \left[ \begin{array}{cc} I & Y \\ & Z \end{array} \right] \left[ \begin{array}{c} \hat{w} \\ w_S \end{array} \right] = \left[ \begin{array}{c} \hat{x} \\ x_S \end{array} \right]. \quad (3.9)$$

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

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

$$\left[ \begin{array}{cccc|ccc} \hat{Q} & & & & \hat{q} & & \\ -\hat{T} & \hat{Q} & & & -\hat{t} & \hat{q} & \\ & \ddots & \ddots & & & \ddots & \ddots \\ & & -\hat{T} & \hat{Q} & & & \\ & & & -\hat{T} & \hat{Q} & & \end{array} \right] Y = \left[ \begin{array}{ccc} \hat{q} & & \\ -\hat{t} & \hat{q} & \\ & -\hat{t} & \ddots \\ & & \ddots & \hat{q} \\ & & & -\hat{t} \end{array} \right]. \quad (3.10)$$

The first column of  $Y$  solves the block-diagonal system

$$\left[ \begin{array}{cccc|ccc} \hat{Q} & & & & y^{m+1} \\ -\hat{T} & \hat{Q} & & & y^{m+2} \\ & \ddots & \ddots & & \vdots \\ & & -\hat{T} & \hat{Q} & y^{n+d-1} \\ & & & -\hat{T} & \hat{Q} & y^{n+d} \end{array} \right] \left[ \begin{array}{c} y^{m+1} \\ y^{m+2} \\ \vdots \\ y^{n+d-1} \\ y^{n+d} \end{array} \right] = \left[ \begin{array}{c} \hat{q} \\ -\hat{t} \\ \vdots \\ 0 \\ 0 \end{array} \right]. \quad (3.11)$$

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:

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

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

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

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

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

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

$$Z = - \begin{bmatrix} \theta + \alpha_{m+1} & & & & \\ \alpha_{m+2} & \theta + \alpha_{m+1} & & & \\ \vdots & \vdots & \ddots & & \\ \alpha_{n+d-1} & \alpha_{n+d-2} & \dots & \theta + \alpha_{m+1} & \end{bmatrix}. \quad (3.12)$$

*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 [12]. Otherwise, ordinary forward substitution suffices to find  $w_S$ , and then  $\hat{w} = \hat{x} - Yw_S$ . Now  $w^{m+1}, w^{m+2}, \dots, w^{n+d}$  (and hence  $v^{m+1}, v^{m+2}, \dots, v^{n+d}$ ) may be determined from  $\hat{w}$  and  $w_S$ .

**3.5. Veinott's method for recurrent classes.** Veinott [13, 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

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





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

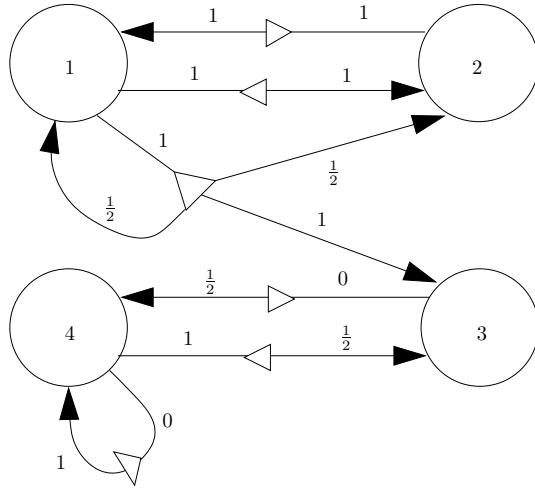


FIG. 4.1. Graphical representation of system

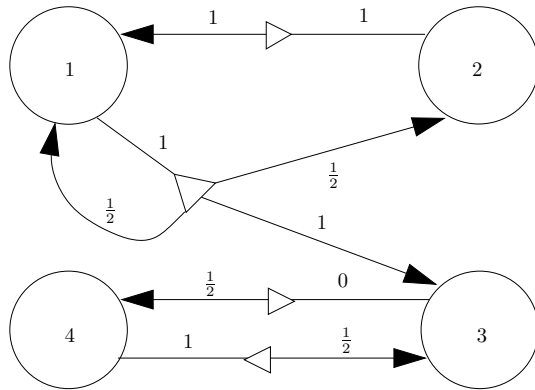


FIG. 4.2. Graphical representation of  $\delta$



**4.1. Block LU method (BLU).** Rearranging as in (3.7') gives

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

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

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

Block forward substitution solves

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

Then,

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

Finally, block backward substitution first solves

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

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.



**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  $Q_\delta$  repeatedly as in (3.4). If  $Q_\delta$  is ill-conditioned, any errors in solving with  $Q_\delta$  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 [13]. By finding  $m$ -optimal policies for  $m = -d, \dots, n$  sequentially, Veinott maintains  $V_\delta^{m-1}$  throughout and simply searches for  $m, \dots, (m + d)$ -improvements (again in order). Finding an  $(m + d)$ -improvement requires  $V_\delta^{m+d}$ , 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  $Q_\delta$  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  $Q_\delta$ .

The RRLU factorization of  $Q_\delta$  is essential for numerical computation. If a class is recurrent but the LU fails to identify the singularity, the factors of  $Q_\delta$  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 Block LU 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  $\alpha_{m+2}, \dots, \alpha_{n+d-1}$ . 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 Block LU method (BLU) and the extended Veinott method (EVM) involve similar amounts of computation, but may differ in their numerical accuracy.

To compare the methods, we performed some experiments using MATLAB 7.0.4 [6] with machine precision  $\epsilon \approx 2 \times 10^{-16}$ . We generated 100 sparse linear systems (order 100, density approximately 20%), 50 corresponding to transient classes and 50 corresponding to recurrent classes. Hence, half the systems are of the form (3.3) and the other half contain one singularity as in (3.6). We also implemented each method with three different LU factorizations:

1. MATLAB's sparse LU factorization: `[L,U,P,Q] = lu(A,thresh)`.
2. LUSOL with threshold partial pivoting (TPP).
3. LUSOL with threshold complete pivoting (TCP).

MATLAB's LU also uses TPP. The threshold parameters were set to keep  $|L_{ij}| \leq 2.0$ ,

a fairly strict bound that favors stability over sparsity but allows a little freedom. 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 solved for the Laurent coefficients in the test classes with both BLU and EVM using each of the factorizations. For each class we performed 100 experiments. We randomly permuted the states in the class (this corresponds to a symmetric permutation of the rows and columns of the linear system, but does not change the Laurent coefficient values) and calculated  $v^j$ ,  $j = -1, 0, \dots, 6$  from the resulting linear system. We then calculated the eight residual norms for (2.5):  $\rho_j \equiv \|c^j + Q_\delta v^j - v^{j-1}\|_\infty$ ,  $j = -1, 0, \dots, 6$ , which should be 0 for all  $j$ . We observed the largest residual norms for each method and factorization over the 100 experiments, obtaining the following results:

- For transient classes, the residual norms for the calculated  $v^j$  increase with  $j$ , but reach only  $O(10^{-13})$  (when  $j = 6$ ) for all combinations of method and factorization, so are numerically insignificant.
- For recurrent classes, the residual norms resulting from BLU increase with  $j$  from  $O(10^{-16})$  to  $O(1)$  (varying slightly with the factorization used), implying significant numerical error with all factorizations.
- In contrast, the EVM residual norms stabilize at  $O(10^{-14})$  as  $j$  increases for all factorizations, so are numerically insignificant. Figure 6.1 shows the behaviour of the residual norms for the recurrent classes.

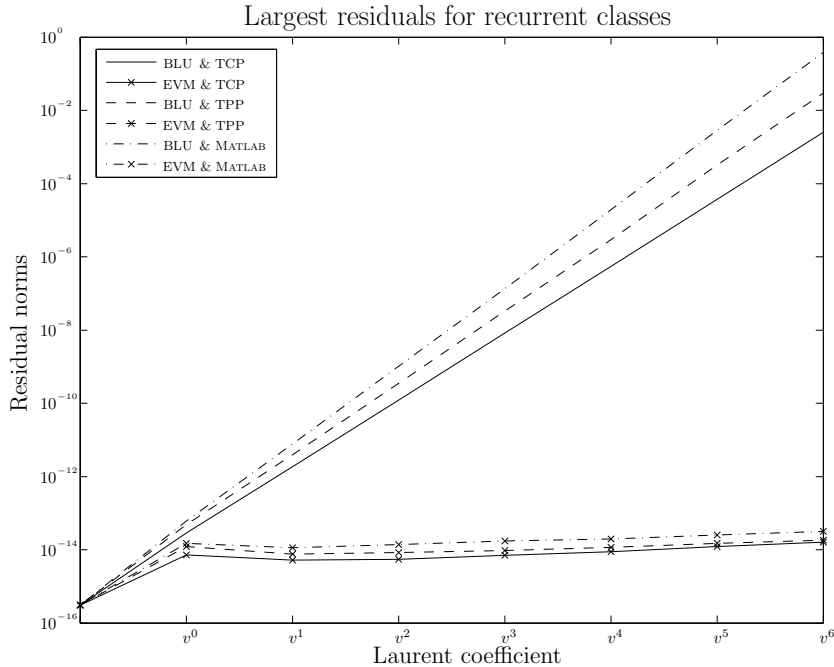


FIG. 6.1. Residual norms for recurrent classes

To identify differences in the methods and factorizations more accurately, we analyzed pairwise differences of the residual norms:  $\rho_j^* - \rho_j^{**}$  for each  $j$ , where the superscripts refer to two different methods or factorizations. Statistical analysis of these pairwise differences in residuals (for each of the 100 systems) 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 first evidence of difference between BLU and EVM appears in the calculation of  $v^0$ . There is evidence that  $\rho_j^{\text{BLU}} - \rho_j^{\text{EVM}} \gg 0$  for most  $j$ . The difference is initially insignificant, but grows with  $j$ : see Figure 6.2. Note that the difference grows slowest for TCP, followed by TPP and then MATLAB.

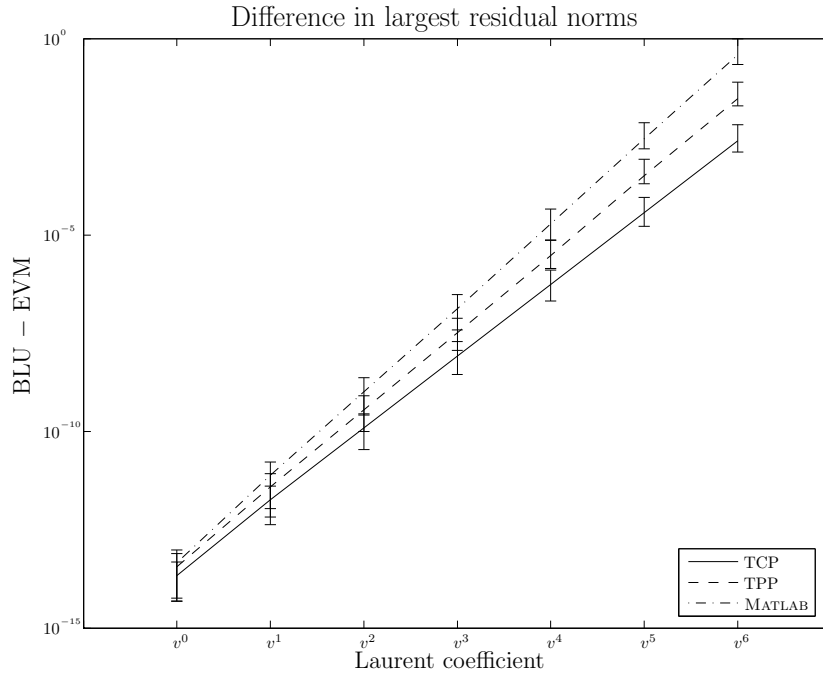


FIG. 6.2. Confidence intervals for pairwise comparison of BLU and EVM (recurrent classes)

**Comparison of factorizations.**

- For transient classes, there is statistical evidence of different factorizations giving different residual norms, for both BLU and EVM. However, the difference is  $O(10^{-13})$ , so is numerically insignificant.
- For recurrent classes with BLU, there is evidence of difference among the three factorizations: see Figure 6.3.  $\rho_j^{\text{TPP}} - \rho_j^{\text{TCP}}$ ,  $\rho_j^{\text{MATLAB}} - \rho_j^{\text{TCP}}$ , and  $\rho_j^{\text{MATLAB}} - \rho_j^{\text{TPP}}$  exceed  $\sqrt{\epsilon}$  for  $j \geq 3$ , so that TCP outperforms TPP, which outperforms MATLAB.
- For recurrent classes with EVM,  $\rho_j^{\text{TPP}} - \rho_j^{\text{TCP}}$ ,  $\rho_j^{\text{MATLAB}} - \rho_j^{\text{TCP}}$ , and  $\rho_j^{\text{MATLAB}} - \rho_j^{\text{TPP}}$  are  $O(10^{-14})$  for all  $j$ , and thus numerically insignificant.

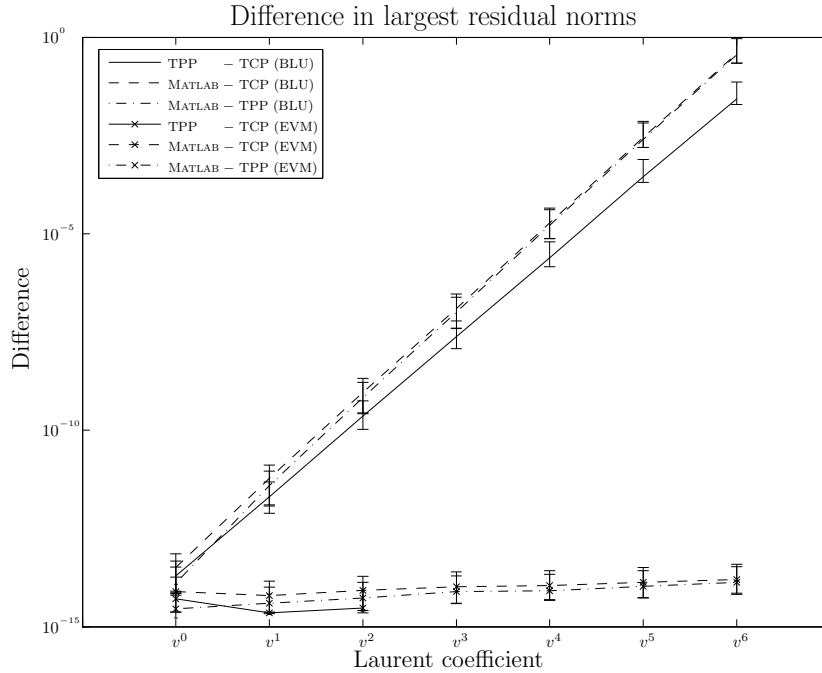


FIG. 6.3. Confidence intervals for pairwise comparison of factorizations (recurrent classes)

From our experiments, it is unclear if TCP is required to identify the rank of a class correctly, or if TPP (or TRP) are sufficient. As the threshold value 2.0 is quite low (favoring stability over sparsity), all factorizations seem to have determined the rank of  $Q_\delta$  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 [3] suggests the use of TRP with  $1.1 \leq \text{threshold} \leq 2.0$ .

**7. Contributions.** Veinott [13] 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 results in the last row of the system vanishing, leaving a nonsingular upper triangular matrix.)

Here we present BLU, a new method for computing the Laurent coefficients of a system with substochastic classes (although the entire system may not be substochastic). This method follows Veinott and Bather by using the dependence partial ordering to decompose the problem into a sequence of computations on irreducible, substochastic systems. With the help of a RRLU 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 revisit Veinott’s method for calculating the Laurent coefficients for a substochastic system. We extend his idea of repeated use of Gaussian elimination to solve for systems with substochastic classes. We also im-

plement 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.

Note that a stable LU factorization could be applied to the entire system (3.3), but this direct approach would become increasingly inefficient with the system dimension. 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 the EVM approach should be used. By incorporating EVM into policy improvement, we should be able to deal successfully with the large models that arise in the sustainable management of resources.

**Acknowledgement.** 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.

## REFERENCES

- [1] J. A. Bather. Optimal decision procedures for finite Markov chains. Part III: General convex systems. *Adv. Appl. Prob.*, 5:541–558, 1973.
- [2] D. Blackwell. Discrete dynamic programming. *Ann. Math. Statist.*, 33(2):719–726, 1962.
- [3] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Rev.*, 47:99–131, 2005.
- [4] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Maintaining LU factors of a general sparse matrix. *Linear Algebra and its Applications*, 88/89:239–270, 1987.
- [5] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, 2nd edition, 2002.
- [6] Matlab. Matrix laboratory. <http://www.mathworks.com>.
- [7] B. L. Miller and A. F. Veinott, Jr. Discrete dynamic programming with a small interest rate. *Ann. Math. Statist.*, 40(2):366–370, 1969.
- [8] M. J. O'Sullivan. *New Methods for Dynamic Programming Over an Infinite Time Horizon*. PhD thesis, Dept of Management Science and Engineering, Stanford University, 2002.
- [9] M. J. O'Sullivan and M. A. Saunders. Sparse rank-revealing LU factorization via Threshold Complete Pivoting and Rook Pivoting. <http://www.stanford.edu/group/SOL/talks.html>, presented at Householder Symposium XV on Numerical Linear Algebra, Peebles, Scotland, June 17–21, 2002.
- [10] M. J. O'Sullivan and M. A. Saunders. LUSOL: A basis package for constrained optimization. <http://www.stanford.edu/group/SOL/talks.html>, presented at IFORS triennial conference on OR/MS, Honolulu, HI, July 11–15, 2005.
- [11] U. G. Rothblum. Normalized Markov decision chains I: Sensitive discount optimality. *Oper. Res.*, 23(4):785–795, 1975.
- [12] W. F. Trench. A note on solving nearly triangular Toeplitz systems. *Linear Algebra Appl.*, 93:57–65, 1987.
- [13] A. F. Veinott, Jr. Discrete dynamic programming with sensitive discount optimality criteria. *Ann. Math. Statist.*, 40(5):1635–1660, 1969.
- [14] A. F. Veinott, Jr. Markov decision chains. In B. C. Eaves and G. B. Dantzig, editors, *Studies in Optimization. MAA Studies in Mathematics*, volume 10, pages 124–159. Mathematical Association of America, 1974.