SIAM J. MATRIX ANAL. APPL.
Vol. 32, No. 1, pp. 248–261

A PERRON ITERATION FOR THE SOLUTION OF A QUADRATIC VECTOR EQUATION ARISING IN MARKOVIAN BINARY TREES*

BEATRICE MEINI† AND FEDERICO POLONI‡

Abstract. We propose a novel numerical method for solving a quadratic vector equation arising in Markovian binary trees. The numerical method consists of a fixed-point iteration, expressed by means of the Perron vectors of a sequence of nonnegative matrices. A theoretical convergence analysis is performed. The proposed method outperforms the existing methods for close-to-critical problems.

Key words. Perron vector, Markov chain, Markovian binary tree, nonlinear matrix equation, fixed-point iteration, bilinear form

AMS subject classifications. 65H10, 15A24, 60J22, 15A63, 15B35

DOI. 10.1137/100796765

1. Introduction.

In this paper, we study the quadratic vector equation

$$x = a + B(x \otimes x), \tag{1.1}$$

where $a \in \mathbb{R}^n$, $B \in \mathbb{R}^{n\times n^2}$ have nonnegative entries, the symbol $\otimes$ denotes the Kronecker product, and the unknown x is an n-dimensional vector. The coefficients a and B are such that the vector $e = (1, 1, \dots, 1)^T$ is a solution of (1.1).

Equation (1.1) arises in the study of Markovian binary trees (MBT), which are a particular family of branching processes used to model the growth of populations consisting of several types of individuals who may produce offspring during their lifetime. One important issue related to MBTs is the computation of the extinction probability of the population, which is the minimal nonnegative solution $x^* \in \mathbb{R}^n$ of the quadratic vector equation (1.1).

Several numerical methods have been proposed for computing the vector x^*. In [2] the authors propose two linearly convergent algorithms, called *depth* and *order* algorithms. The *thicknesses* algorithm, still linearly convergent, is proposed in [5]. In [4] the authors apply the Newton method, which has quadratic convergence. A modification of Newton's method has been proposed in [6]. All these methods have a probabilistic interpretation, and each of them provides a sequence $\{x_k\}_k$ of nonnegative vectors, with $x_0 = (0, \dots, 0)^T$, which converges monotonically to the minimal nonnegative solution x^*. A common feature of all these methods is that their convergence speed slows down for a particular class of problems, called *close to critical*. Close-to-critical problems are the most challenging numerically, since the simplest simulation techniques require a larger number of iterates to capture the behavior of the modeled population. Moreover, the accuracy of the approximation deteriorates.

In this paper, we write (1.1) in the form $x = a + b(x, x)$ where $b(u, v) := B(u \otimes v)$ is the bilinear form defined by the matrix B. If we set $y = e - x$, the latter equation becomes

*Received by the editors May 28, 2010; accepted for publication (in revised form) by D. Boley January 17, 2011; published electronically March 17, 2011. This work was partly supported by the research project PRIN 2008 N. 20083KLJEZ.
http://www.siam.org/journals/simax/32-1/79676.html

†Dipartimento di Matematica, Università di Pisa, Largo B. Pontecorvo 5, 56127 Pisa, Italy (meini@dm.unipi.it).

‡Scuola Normale Superiore, Piazza dei Cavalieri 6, 56126 Pisa, Italy (f.poloni@sns.it).

$$y = b(y, e) + b(e, y) - b(y, y). \tag{1.2}$$

The sought solution y^* of (1.2) is $y^* = e - x^*$, where x^* is the minimal nonnegative solution of (1.1). In the probability interpretation of Markovian binary trees, since x^* is the vector of extinction probabilities, then $y^* = e - x^*$ is the survival probability.

Applying a functional iteration directly to (1.2), like Newton's method, gives nothing new, since (1.2) differs from (1.1) by a linear change of variable. However, the new equation (1.2) can be rewritten as

$$y = H_y y, \tag{1.3}$$

where $H_y := b(\cdot, e) + b(e - y, \cdot)$. The matrix H_y is nonnegative and irreducible if $y < e$. In particular, the solution y^* is such that $\rho(H_{y^*}) = 1$ and y^* is the Perron vector of the matrix H_{y^*}.

This interpretation facilitates the design of a new algorithm for computing y^*. To this purpose, define the map $\mathrm{PV}(M)$ as the Perron vector of a nonnegative irreducible matrix M so that we may rewrite (1.3) as

$$y = \mathrm{PV}(H_y). \tag{1.4}$$

The idea is to apply a fixed-point iteration to solve (1.4), thus generating a sequence $\{y_k\}_k$ of positive vectors such that $y_{k+1} = \mathrm{PV}(H_{y_k})$ and y_k converges to y^*. A suitable normalization of the Perron vector, consistent with the solution, is needed to obtain a well-posed iteration. In this way we obtain a new iterative scheme, which is completely different from classical functional iterations. Indeed, the proposed algorithm, unlike known methods, fully exploits the fact that the solution $x = e$ of (1.2) is known. Moreover, the fixed-point iteration at the basis of our algorithm relies on a new interpretation of the solution y^* in terms of the Perron vector. These differences with respect to classical methods lead to great improvements in the numerical solution of MBTs that are close to critical.

We perform a convergence analysis of the fixed-point iteration $y_{k+1} = \mathrm{PV}(H_{y_k})$ by giving an expression to the Jacobian of the map $y \to \mathrm{PV}(H_y)$. This expression allows us to derive a local convergence result. Moreover, most importantly, we prove that, although the convergence of the method is linear, the speed of convergence *increases* as the problem gets close to critical. In the limit case of a critical problem, the convergence becomes superlinear. This nice behavior is opposite to the one of Newton's method, whose speed of convergence is superlinear in the supercritical case and becomes linear in the critical case.

A wide numerical experimentation confirms our theoretical analysis. For far-from-critical problems the standard techniques are preferable, while for close-to-critical problems our method outperforms the existing ones.

The paper is organized as follows. In section 2, we introduce Markovian binary trees and recall some theoretical results and numerical methods. In section 3 we state our assumptions on the problem. In section 4 we rewrite the vector equation in terms of an equation for the vector $y = e - x$ and discuss the properties of the equation obtained in this way. The new algorithm, based on a Perron iteration, is presented in section 5. The theoretical convergence analysis is performed in section 6. Finally, in section 7 we present the results of the numerical experiments. Conclusions and open issues are addressed in section 8.

In the paper, we write $a \leq b$, for $a, b \in \mathbb{R}^n$, to denote the componentwise ordering; i.e., $a_i \leq b_i$ for all $i = 1, 2, \ldots, n$. Similarly, we write $a < b$ if $a_i < b_i$ for each i.

2. Markovian binary trees. A Markovian binary tree [2, 5] is a stochastic process that models a population of individuals that produces a random number of offspring and then dies. MBTs have applications in biology, epidemiology, and telecommunication systems.

They consist of a branching process [1] in which branches (individuals) can be in several different states, and all the branching events result in exactly two children. In more detail, the life of every individual is controlled by a so-called Markovian *phase process* on n states, which we label by $\{1, 2, \ldots, n\}$. At different times in its life, an individual may change state, die, or give birth to one child at a time. The life of the childs is controlled by an independent process, identical to the one governing the parent life; the child, at the birth time, is in the same state of the parent. The interest is in computing the extinction probability, i.e., the probability that all the individuals eventually die out, starting from an initial colony (typically, the starting colony is formed by a single individual).

In order to determine the extinction probability, it is not necessary to consider the time elapsed between births and deaths; therefore, it is not essential to use a complete description of the underlying phase process. The quantities determining the behavior of the tree are:

- the vector $a \in [0,1]^n$, containing the probabilities a_i that an individual in state i dies without producing offspring;
- the probabilities b_{ijk} that an individual in state i gives birth, as its first child, to an individual in state k, while simultaneously switching to state j. Such probabilities are often given through a matrix $B \in [0,1]^{n \times n^2}$ such that $b_{ijk} = B_{i,n(j-1)+k}$. In this paper, though, we adopt a different notation: we define the bilinear form $b : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ as $b : (u, v) \mapsto B(u \otimes v)$ so that

$$(b(u,v))_i = \sum_{j,k=1}^{n} b_{ijk} u_j v_k.$$

We call x_i^* the probability that a population starting from an individual in state i becomes extinct, and form the associated vector x^*. One can prove [2] that x^* is the minimal (in the componentwise ordering) solution to (1.1), which, with our notation, can be rewritten as

$$x = a + b(x, x). \tag{2.1}$$

The constraint $e = a + b(e, e)$ (i.e., $x = e$ is a solution to the equation) holds for probabilistic reasons: it corresponds to the fact that the probabilities of all possible outcomes for an individual in state i must sum to 1, for each $i = 1, 2, \ldots, n$.

An MBT is called subcritical, supercritical, or critical if the spectral radius $\rho(R)$ of the matrix $R = B(e \otimes I + I \otimes e)$ is strictly less than 1, strictly greater than 1, or equal to 1, respectively. In the subcritical and critical cases, the minimal nonnegative solution is the vector of all ones, while in the supercritical case $x^* \leq e$, $x^* \neq e$ [5, 1, 3]. Thus, only the supercritical case is of interest for the computation of x^*. We call *close to critical* the problems for which the spectral radius of R is close to 1, and thus $x^* \approx e$.

The *depth* algorithm [2] can be seen as the functional iteration $x_{k+1} = a + b(x_k, x_k)$; the *order* algorithm consists of the iteration

$$x_{k+1} = a + b(x_{k+1}, x_k), \tag{2.2}$$

which we may solve by rewriting it as $x_{k+1} = (I - b(\cdot, x_k))^{-1}a$. Notice that $b(\cdot, x_k) : \mathbb{R}^n \to \mathbb{R}^n$ is a linear map that is represented by an $n \times n$ matrix. One may also consider the alternate form

$$x_{k+1} = a + b(x_k, x_{k+1}), \tag{2.3}$$

which is obtained from (2.2) by mirroring the tree in order to switch the left and right branches. Of course, the formulations of the two iterations are equivalent. In fact, by defining $\widetilde{b}(u, v) = b(v, u)$, we obtain a new MBT, defined by a and $\widetilde{b}$, for which (2.2) is formally equivalent to (2.3) for the original problem and vice versa.

This generalizes to the following observation: while the solution of (2.1) depends only on the quadratic form $x \mapsto b(x, x)$, *order* and possibly other algorithms depend on the (nonnecessarily symmetric) associated bilinear form $(u, v) \mapsto b(u, v)$. We may therefore alter the underlying bilinear form in any way that leaves unaltered the quadratic form, thus getting different algorithms, possibly with different numerical behaviors. We analyze these additional possibilities in more detail in the paper [3].

The *thicknesses* iteration [5] consists of alternating steps of (2.2) and (2.3); it has similar properties to the two versions of the *order* algorithm, but is more resilient, as it deals satisfactorily also with cases in which one of the two has extremely slow convergence speed.

The Newton algorithm applied to (2.1) [4] yields the iteration

$$x_{k+1} = (I - b(x_k, \cdot) - b(\cdot, x_k))^{-1}(a - b(x_k, x_k)),$$

which converges quadratically instead of linearly like the previous ones. All the algorithms described so far converge monotonically to the minimal solution x^*, in the sense that $0 = x_0 \le x_1 \le \cdots \le x_k \le \cdots \to x^*$. On close-to-critical problems, the convergence speed of all of them degrades; in fact, in the limit case $\rho(R) = 1$, $x^* = e$, the Newton method converges linearly, while the other functional iterations converge sublinearly.

3. Assumptions on the problem.

Let $a \in \mathbb{R}^n$, $B \in \mathbb{R}^{n \times n^2}$ have nonnegative entries, and consider the quadratic vector equation (1.1) where it is assumed that the vector $e = (1, 1, \ldots, 1)^T$ is a solution. Let $x^* \in \mathbb{R}^n$ be the minimal nonnegative solution of (1.1); i.e., $x^* \le x$ for any other nonnegative solution. A unique solution x^* exists, according to the results of [1, section V.3].

We assume that $\rho(R) > 1$, where $R = B(e \otimes I + I \otimes e)$. Under this assumption (supercritical case) $x^* \le e$, $x^* \ne e$ (see [5] and [1]). It is worth pointing out that, if $\rho(R) = 1$, then $x^* = e$; therefore as $\rho(R)$ is greater than 1 and gets closer to 1, then x^* approaches to the vector of all ones.

We assume that, for the minimal solution x^* of (2.1), it holds $x^* > 0$, and the Jacobian of the map $x \to x - a - b(x, x)$ at x^*, i.e., $I - b(x^*, \cdot) - b(\cdot, x^*)$, is a nonsingular irreducible M-matrix. Since irreducibility is only determined by the nonnegativity pattern of $b(\cdot, \cdot)$, the irreducibility condition is equivalent to requiring that $b(e, \cdot) + b(\cdot, e)$ is irreducible. Notice that the latter is just another notation to represent the matrix R and that this matrix can be computed explicitly and checked for irreducibility at the beginning of the algorithm.

Moreover, we may assume that $e^T b(e - x^*, e - x^*) > 0$; otherwise $b(e - x^*, e - x^*) = 0$ (since $B \ge 0$), and the problem is trivial since it becomes a linear problem.

The two more restrictive assumptions that we have made are that $x^* > 0$ and R is irreducible. These two assumptions are discussed more carefully in [8] and [3], respectively, where it is shown how to reduce problems that do not fulfill them to ones that do. Namely, in [8] it is shown how to compute the zero-nonzero pattern of x^* without solving the equation; after that, we may restrict the problem by dropping the zero entries of x^* via a projection. On the other hand, in [3] it is shown that a problem satisfying the first assumption but with reducible R can be split into two smaller-size problems, and this approach can be applied recursively until the resulting associated matrices are irreducible.

4. The optimistic equation.

A property of (2.1) which has not been exploited so far in the existing literature is that $x = e$ is a solution. If we set $x = e - y$, by using the bilinearity of the operator $b(\cdot,\cdot)$ and the property that $e = a + b(e,e)$, (2.1) can be rewritten as

$$y = b(y,e) + b(e,y) - b(y,y). \tag{4.1}$$

The trivial solution is $y = 0$, which corresponds to $x = e$. We are interested in the nontrivial solution $0 \leq y^* < e$, which gives the sought solution $x^* = e - y^*$.

In the probabilistic interpretation of Markovian binary trees, x^* is the vector of extinction probabilities; thus $y^* = e - x^*$ is the vector of survival probabilities; i.e., y_i^* is the probability that a colony starting from a single individual in state i does not become extinct in a finite time. For this reason, we refer to (4.1) as the *optimistic equation.*

Notice that (4.1) admits the following probabilistic interpretation. The term $b(y,e)$ represents the probability that the original individual $\mathcal{M}$ (for "mother") spawns an offspring $\mathcal{F}$ (for "first-born"), and after that the colony generated by the further offspring of $\mathcal{M}$, excluding $\mathcal{F}$, survives. The term $b(e,y)$ represents the probability that $\mathcal{M}$ spawns $\mathcal{F}$, and the colony generated by $\mathcal{F}$ survives. The term $b(y,y)$ represents the probability that $\mathcal{M}$ spawns $\mathcal{F}$, and after that both their colonies survive. Thus (4.1) follows by the well-known *inclusion-exclusion* principle

$$\mathbb{P}[\mathcal{M} \text{ or } \mathcal{F} \text{ survives}] = \mathbb{P}[\mathcal{M} \text{ survives}] + \mathbb{P}[\mathcal{F} \text{ survives}] - \mathbb{P}[\text{both } \mathcal{M} \text{ and } \mathcal{F} \text{ survive}],$$

where $\mathbb{P}[X]$ denotes the probability of the event X.

Equation (4.1) can be rewritten as

$$y = H_y y, \tag{4.2}$$

where

$$H_y = b(\cdot,e) + b(e,\cdot) - b(y,\cdot). \tag{4.3}$$

Notice that H_y is the sum of a fixed matrix and a matrix that depends linearly on y. Therefore, the quadratic operator on the right-hand side of (4.1) is "factored" as the product of a matrix which depends on y, and y.

An important property is that H_y is a nonnegative irreducible matrix whenever $y < e$. Therefore, by the Perron–Frobenius theorem [9], if $y < e$, then H_y has a positive eigenvalue $\lambda_y = \rho(H_y)$, and to λ_y corresponds a positive eigenvector w_y, unique up to a multiplicative constant so that $H_y w_y = \lambda_y w_y$. The eigenvalue λ_y is the so-called *Perron value*, and the positive eigenvector w_y is the so-called *Perron vector.* Therefore, the sought solution y^* can be interpreted as a vector $y^* < e$ such that $\rho(H_{y^*}) = 1$ and y^* is a Perron vector of H_{y^*}. Notice that, with our assumptions in place, the fact that $y^* > 0$, i.e., $x^* < e$, follows from the Perron–Frobenius theorem applied to (4.2).

It is worth pointing out that this interpretation of y^* in terms of the Perron vector allows us to keep away from the trivial solution $y = 0$ of (4.2), since the Perron vector has strictly positive elements.

The formulation of the quadratic vector equation in terms of the Perron vector allows us to design a new algorithm for its solution.

5. The Perron iteration.

If we set up a fixed-point iteration or a Newton method for y based on (4.1), we get the traditional fixed-point iterations and Newton methods for MBTs [4], since what we have done is simply a linear change of variables. Instead, we exploit the fact that y^* is a Perron vector of the nonnegative irreducible matrix H_{y^*} (compare (4.2)).

To this purpose we introduce the operator

$$u = \mathrm{PV}(X)$$

which returns the Perron vector u of the irreducible nonnegative matrix X.

Thus, we can devise a fixed-point iteration to compute the solution y^* by defining the sequence of vectors

$$y_{k+1} = \mathrm{PV}(H_{y_k}), \quad k = 0, 1, 2, \ldots, \tag{5.1}$$

starting from an initial approximation y_0. In order to define uniquely the sequence $\{y_k\}$, we need to impose a normalization for the Perron vector, which is uniquely defined up to a multiplicative constant. A possible choice for the normalization is imposing that the residual of (4.1) is orthogonal to a suitable vector $w \in \mathbb{R}^n$, i.e.,

$$w^T(y_{k+1} - b(y_{k+1}, e) - b(e, y_{k+1}) + b(y_{k+1}, y_{k+1})) = 0. \tag{5.2}$$

Clearly, this normalization is consistent with the solution of (4.1). We choose w as the left Perron vector of the matrix $R = b(\cdot, e) + b(e, \cdot)$; the rationale for this choice is discussed in section 6.

Given a Perron vector u of H_{y_k}, the equation to compute the normalization factor α such that $y_{k+1} = \alpha u$ satisfies (5.2) reduces to

$$\alpha w^T u = \alpha w^T b(u, e) + \alpha w^T b(e, u) - \alpha^2 w^T b(u, u),$$

whose only nonzero solution is

$$\alpha = -\frac{w^T(u - b(u, e) - b(e, u))}{w^T b(u, u)}.$$

Notice that the solution $\alpha = 0$ corresponds to the trivial solution $y = 0$ ($x = e$), which we want to avoid.

The $\mathrm{PV}(\cdot)$ operator is defined on the set of irreducible nonnegative matrices. If $y < e$, then the matrix H_y is nonnegative irreducible; therefore the sequence y_k generated by (5.1) is well-defined if $y_k < e$ for any k.

In section 6 we show that the iteration (5.1) is locally convergent. Therefore, if y_0 is quite close to y^*, one can expect that $y_k < e$ for any k. In the case where H_{y_k} is not a nonnegative irreducible matrix, we can define y_{k+1} as an eigenvector corresponding to the eigenvalue of H_{y_k} having maximal real part. We call *maximal eigenvector* this eigenvector. Clearly, if H_{y_k} is a nonnegative irreducible matrix, the maximal eigenvector is the Perron vector. We see in section 7 that this concern is not necessary in practice.

As a starting approximation y_0 we may choose the zero vector. For close-to-critical problems, where y^* is close to zero, this choice should guarantee the convergence, according to the results of section 6.

The resulting iterative process is summarized in Algorithm 1.

Algorithm 1. The Perron iteration

```
Set $k \leftarrow 0$
Set $y_0 \leftarrow 0$
Set $w \leftarrow$ the left Perron vector of $b(e,\cdot) + b(\cdot,e)$
**while** $\|H_{y_k} y_k - y_k\|_1 \geq \varepsilon$ **do**
  Set $u \leftarrow$ the maximal eigenvector of $H_{y_k}$
  Compute the normalization factor $\alpha = -\frac{w^T(u - b(u,e) - b(e,u))}{w^T b(u,u)}$
  Set $y_{k+1} \leftarrow \alpha u$
  Set $k \leftarrow k+1$
**end while**
**return** $x = e - y_k$
```

6. Convergence analysis of the Perron iteration.

In this section, we show that the Perron iteration (5.1) is locally convergent and its convergence is linear. Moreover, the convergence speed gets faster as the problem gets closer to critical.

6.1. Derivatives of eigenvectors.

It is well-known [10] that the eigenvalues and eigenvectors of a matrix are analytic functions of the matrix entries in a neighborhood of a simple eigenpair. The following formula for an analytical expression of their first derivatives is from Meyer and Stewart [7, Theorem 1].

Theorem 6.1. *Let $A = A(z)$, $\lambda = \lambda(z)$, $u = u(z)$ be a matrix, eigenvalue, and associated eigenvector depending on a parameter $z \in \mathbb{C}$. Let us suppose that $\lambda(z_0)$ is simple and $A'(z_0)$, $\lambda'(z_0)$, $u'(z_0)$ each exist. Let $w = w(z)$ be another vector such that $w'(z_0)$ exists, and let $\sigma(u,w)$ be a function whose value is a real scalar constant for all z. Let σ_1^H and σ_2^H be the partial gradients of $\sigma(\cdot,\cdot)$ seen as a function, respectively, of its first and second vector argument only.*

If $\sigma_1^H u \neq 0$ for $z = z_0$, then the derivative u' of u at $z = z_0$ is given by

$$u' = \frac{\sigma_1^H (A - \lambda I)^{\#} A' u - \sigma_2^H w'}{\sigma_1^H u} u - (A - \lambda I)^{\#} A' u.$$

Here $X^{\#}$ denotes the so-called *group inverse* of a singular matrix X, i.e., the inverse of X in the maximal multiplicative subgroup containing X. We refer the reader to the above-mentioned paper for more details on group inverses.

While the eigenvalue perturbation theory is formulated naturally through group inverses, the Moore–Penrose pseudoinverse $X^{\dagger}$ is more common in the context of matrix computations; for instance, a native Matlab command is provided for the latter but not for the former. In fact, very little is needed on group inverses here, and we may replace the formula with a similar one involving the Moore–Penrose pseudoinverse.

Theorem 6.2. *With the same hypotheses as Theorem 6.1, let $v(z)$ be the left eigenvector of $A(z)$ corresponding to the eigenvalue $\lambda(z)$. If $\sigma_1^H u \neq 0$ for $z = z_0$, then the derivative u' of u at $z = z_0$ is given by*

$$u' = \frac{\sigma_1^H (A - \lambda I)^{\dagger} (A' - \lambda' I) u - \sigma_2^H w'}{\sigma_1^H u} u - (A - \lambda I)^{\dagger} (A' - \lambda' I) u \tag{6.1}$$

with $\lambda' = \frac{v^H A' u}{v^H u}$.

Proof. The proof is a minor modification of the original proof [7] of Theorem 6.1. By differentiating the identity $Au = \lambda u$ we get $A'u + Au' = \lambda' u + \lambda u'$; i.e.,

$$(A - \lambda I)u' = -(A' - \lambda' I)u. \tag{6.2}$$

By left-multiplying everything by v^H, and noting that $v^H A = \lambda v^H$, we get the required expression for the eigenvalue derivative λ'. Moreover, since u is a simple eigenvector at $z = z_0$, the kernel of $(A - \lambda I)$ is span(u). Thus from (6.2) we can determine u' up to a scalar multiple of u:

$$u' = -(A - \lambda I)^{\dagger}(A' - \lambda' I)u + \delta u. \tag{6.3}$$

We shall now use the normalization condition $\sigma(u, w) = k$ to determine the value of δ. By differentiating it, we get

$$\sigma_1^H(u, w)u' + \sigma_2^H(u, w)w' = 0. \tag{6.4}$$

Plugging (6.3) into (6.4) yields a linear equation for δ. ☐

6.2. Jacobian of the Perron iteration. The Perron iteration is a fixed-point iteration for the function $F(y) := PV(H_y)$, where the function $u = PV(X)$ returns the Perron vector u of the nonnegative irreducible matrix X, normalized such that $w^T(u - H_u u) = 0$, where w is a fixed positive vector. We can use Theorem 6.2 to compute the Jacobian of this map F.

THEOREM 6.3. *Let y be such that H_y is nonnegative and irreducible. Let $u = F(y)$, and let v such that $v^T H_y = \lambda v^T$, where $\lambda = \rho(H_y)$. Then the Jacobian of the map F at y is*

$$JF_y = \left(I - \frac{u\sigma_1^T}{\sigma_1^T u}\right)(H_y - \lambda I)^{\dagger}\left(I - \frac{uv^T}{v^T u}\right)b(\cdot, u), \tag{6.5}$$

where

$$\sigma_1^T = w^T(I - b(e - u, \cdot) - b(\cdot, e - u)).$$

Proof. We shall compute first the directional derivative of F at y along the direction a. To this purpose, let us set $y(z) := y + az$, for any $z \in \mathbb{C}$, and $A(z) = H_y$. We have

$$A'(z) = \frac{d}{dz}H_y = -b(a, \cdot).$$

Moreover, set

$$\sigma(u, w) = w^T(u - b(e, u) - b(u, e) + b(u, u)),$$

where $w(z) = w$ for each z (so that $w' = 0$). The partial gradient of $\sigma(\cdot, \cdot)$ with respect to the first argument is $\sigma_1^T = w^T(I - b(e - u, \cdot) - b(\cdot, e - u))$. Plugging everything into

(6.1), we get

$$\begin{aligned} u' =& \frac{\sigma_1^T (A-\lambda I)^\dagger (A'-\lambda' I)u}{\sigma_1^T u} u - (A-\lambda I)^\dagger (A'-\lambda' I)u \\ =& -\left(I - \frac{u\sigma_1^T}{\sigma_1^T u}\right)(A-\lambda I)^\dagger \left(A' - \frac{v^T A' u}{v^T u} I\right) u \\ =& -\left(I - \frac{u\sigma_1^T}{\sigma_1^T u}\right)(A-\lambda I)^\dagger \left(I - \frac{uv^T}{v^T u}\right) A' u \\ =& -\left(I - \frac{u\sigma_1^T}{\sigma_1^T u}\right)(A-\lambda I)^\dagger \left(I - \frac{uv^T}{v^T u}\right) (-b(a,u)) \\ =& \left(I - \frac{u\sigma_1^T}{\sigma_1^T u}\right)(A-\lambda I)^\dagger \left(I - \frac{uv^T}{v^T u}\right) b(\cdot,u)a. \end{aligned}$$

From this expression for the directional derivative, it is immediate to recognize that the Jacobian is (6.5). ▯

6.3. Local convergence of the iteration. The fixed-point iteration $y_{k+1} = F(y_k)$ is locally convergent in a neighborhood of y^* if and only if the spectral radius of JF_{y^*} is strictly smaller than 1. First notice that it makes sense to compute the Jacobian using (6.5) in a neighborhood of the solution y^*. In fact, $\sigma_1^T y^* = w^T(y^* - b(e-y^*, y^*) - b(y^*, e-y^*)) = w^T b(y^*, y^*)$, and the latter quantity is positive as $w > 0$ and $b(y^*, y^*) \gneq 0$, as stated in section 3. Moreover, since $\lambda = 1$ is a simple eigenvalue, the left and right eigenvectors $v = v^*$ and $u = y^*$ cannot be orthogonal.

By evaluating (6.5) at $y = y^*$, we get

$$JF_{y^*} = \left(I - \frac{y^* \sigma_1^{*T}}{w^T b(y^*, y^*)}\right) A^\dagger \left(I - \frac{y^* v^{*T}}{v^{*T} y^*}\right) b(\cdot, y^*), \tag{6.6}$$

where we have set $\sigma_1^{*T} = w^T(I - b(e-y^*, \cdot) - b(\cdot, e-y^*))$ and $A = (b(e-y^*, \cdot) + b(\cdot, e) - I)$.

Let us try to understand what happens to the spectral radius $\rho(JF_{y^*})$ when the problem is close to critical.

THEOREM 6.4. *Let $b_t(\cdot,\cdot)$, $t \in [0,1]$ be an analytic one-parameter family of Markovian binary trees, which is supercritical for $t \in [0,1)$ and critical for $t = 1$, and let us denote with an additional subscript t the quantities defined above for this family of problems. Let us suppose that $R_t := b_t(e,\cdot) + b_t(\cdot, e)$ is irreducible for every $t \in [0,1]$, and let $\rho(JF_{y_t^*,t})$ be the spectral radius of the Jacobian of the Perron iteration as defined in* (6.6). *Then*

$$\lim_{t\to 1^-} \rho(JF_{y_t^*,t}) = \left| 1 - \frac{\widehat{v}_1^T b_1(\widehat{y}_1, \widehat{y}_1)}{w^T b_1(\widehat{y}_1, \widehat{y}_1)} \frac{w^T \widehat{y}_1}{\widehat{v}_1^T \widehat{y}_1} \right|,$$

where $\widehat{y}_1$ and $\widehat{v}_1^T$ are left and right Perron vectors of R_1. As a special case, if the vector w is a scalar multiple of $\widehat{v}_1$, the limit is zero.

Proof. Let us define $\widehat{y}_t$ as the Perron vector of $H_{y_t^*,t}$ normalized so that $\|\widehat{y}_t\|_1 = 1$, and similarly $\widehat{v}_t^T$ as the left Perron vector of the same matrix, normalized so that $\|\widehat{v}_t\|_1 = 1$. Since $H_{y_t^*,t}$ is irreducible, its left and right Perron vectors are analytic functions of t, and thus $\widehat{y}_t$ and $\widehat{v}_t$ converge to $\widehat{y}_1$ and $\widehat{v}_1$, respectively. We have $H_{y_1^*,1} = H_{0,1} = R_1$. Notice that $\sigma_{1,t}^{*,T} \to w^T(I - R_1)$ and that

$$A_t^\dagger = (b(e-y^*, \cdot) + b(\cdot, e) - I)^\dagger \to (R_1 - I)^\dagger.$$

Moreover, since $\widehat{y}_1$ and $\widehat{v}_1$ span the right and left kernel of $I - R_1$, we have

$$(I - R_1)(I - R_1)^\dagger = I - \frac{\widehat{y}_1 \widehat{v}_1^T}{\widehat{v}_1^T \widehat{y}_1}.$$

Additionally, we shall make use of the relation $\rho(AB) = \rho(BA)$, valid for any A and B such that AB and BA are square matrices, in the first and second-to-last step of the following computation.

Putting all together, we get

$$\begin{aligned}\rho(JF_{y_t^*,t}) =& \rho\left(\left(I - \frac{y_t^* \sigma_{1,t}^{*T}}{w^T b_t(y_t^*, y_t^*)}\right) A_t^\dagger \left(I - \frac{y_t^* v_t^{*T}}{v_t^{*T} y_t^*}\right) b_t(\cdot, y_t^*)\right)\\ =& \rho\left(\left(b_t(\cdot, y_t^*) - \frac{b_t(y_t^*, y_t^*)\sigma_{1,t}^{*T}}{w^T b_t(y_t^*, y_t^*)}\right) A_t^\dagger \left(I - \frac{y_t^* v_t^{*T}}{v_t^{*T} y_t^*}\right)\right)\\ =& \rho\left(\left(b_t(\cdot, y_t^*) - \frac{b_t(\widehat{y}_t, \widehat{y}_t)\sigma_{1,t}^{*T}}{w^T b_t(\widehat{y}_t, \widehat{y}_t)}\right) A_t^\dagger \left(I - \frac{\widehat{y}_t \widehat{v}_t^T}{\widehat{v}_t^T \widehat{y}_t}\right)\right).\end{aligned}$$

Therefore, we obtain

$$\begin{aligned}\lim_{t\to 1^-} \rho(JF_{y_t^*,t}) =& \rho\left(-\frac{b_1(\widehat{y}_1, \widehat{y}_1) w^T (I - R_1)}{w^T b_1(\widehat{y}_1, \widehat{y}_1)} (R_1 - I)^\dagger \left(I - \frac{\widehat{y}_1 \widehat{v}_1^T}{\widehat{v}_1^T \widehat{y}_1}\right)\right)\\ =& \rho\left(\frac{b_1(\widehat{y}_1, \widehat{y}_1) w^T}{w^T b_1(\widehat{y}_1, \widehat{y}_1)} \left(I - \frac{\widehat{y}_1 \widehat{v}_1^T}{\widehat{v}_1^T \widehat{y}_1}\right)^2\right)\\ =& \rho\left(\frac{b_1(\widehat{y}_1, \widehat{y}_1) w^T}{w^T b_1(\widehat{y}_1, \widehat{y}_1)} \left(I - \frac{\widehat{y}_1 \widehat{v}_1^T}{\widehat{v}_1^T \widehat{y}_1}\right)\right)\\ =& \rho\left(\frac{1}{w^T b_1(\widehat{y}_1, \widehat{y}_1)} w^T \left(I - \frac{\widehat{y}_1 \widehat{v}_1^T}{\widehat{v}_1^T \widehat{y}_1}\right) b_1(\widehat{y}_1, \widehat{y}_1)\right)\\ =& \left|1 - \frac{\widehat{v}_1^T b_1(\widehat{y}_1, \widehat{y}_1)}{w^T b_1(\widehat{y}_1, \widehat{y}_1)} \frac{w^T \widehat{y}_1}{\widehat{v}_1^T \widehat{y}_1}\right|. \qquad \square\end{aligned}$$

For the normalization condition, the above result suggests taking w as the left Perron vector of $b(e, \cdot) + b(\cdot, e)$. Indeed, this choice guarantees the local convergence of the Perron iteration for close-to-critical problems. Moreover, even though the convergence is linear, the speed of convergence increases as the MBT gets closer to critical; in particular, the convergence is superlinear in the critical case.

7. Numerical experiments.

We compared the Perron iteration (PI) with the Newton method (NM) [4] and with the *thicknesses* algorithm (TH) [5]. As stated before, TH and PI are linearly convergent algorithms, while NM is a quadratically convergent one. All the experiments were performed using Matlab 7 (R14) on an Intel Xeon 2.80Ghz biprocessor.

We applied the algorithms to the two test cases reported in [4]. The first one (E1) is an MBT of size $n = 9$ depending on a parameter λ, which is critical for $\lambda \approx 0.85$ and supercritical for larger values of λ. The second one (E2) is an MBT of size $n = 3$ depending on a parameter λ, which is critical for $\lambda \approx 0.34$ and $\lambda \approx 0.84$, and supercritical for the values inbetween.

The only noteworthy issue in the implementation of PI is the method used for the computation of the maximal eigenvector. The classical methods are usually optimized for matrices of much larger size; however, here we deal with matrices of size $n = 3$ and $n = 9$, for which the complexity constants matter. We compared several candidates (`eigs`, `eig`, the power method, a power method accelerated by repeated squaring of the matrix) and found that in our examples the fastest method to find the maximal eigenvector is computing the full eigenvector basis with `[V,Lambda]=eig(P)` and then selecting the maximal eigenvector. The picture should change for problems of larger size: `eig` takes $O(n^3)$ operations, while, for instance, `eigs` should take only $O(n^2)$ in typical cases. On the other hand, we point out that in absence of any structure (such as sparsity) in $b(\cdot,\cdot)$, forming the matrix $b(v,\cdot)$ or $b(\cdot,v)$ for a new vector v, an operation which is required at every step in all known iterative algorithms, requires $O(n^3)$ operations. Therefore, the CPU times are somehow indicative of the real complexity of the algorithms but should be taken with a grain of salt.

The stopping criterion was chosen to be $\|x - a + b(x,x)\| \leq n\varepsilon$ with $\varepsilon = 10^{-13}$ for all algorithms.

Table 7.1 shows the results for several choices of λ. The algorithm TH is clearly the slowest, taking far more CPU time than the two competitors. The different behavior of PI when approaching the critical cases is apparent: while the iterations for TH and NM increase, PI seems to be unaffected by the near-singularity of the problem, and in fact the iteration count decreases slightly.

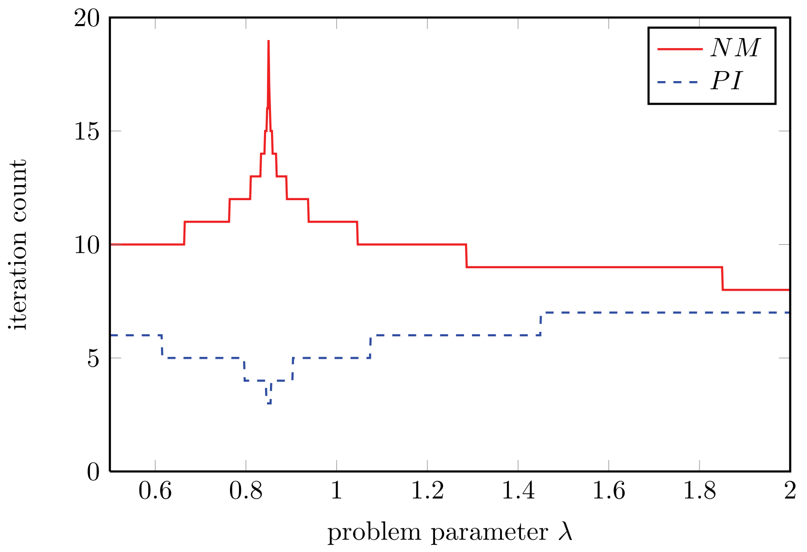
TABLE 7.1
CPU time in seconds (and number of iterations in brackets) for TH, NM, and PI on several choices of λ for $E1$ (top) and $E2$ (bottom).

λ	TH	NM	PI
0.86	$2.39 \times 10^{+00}$(11879)	5.09×10^{-03}(14)	4.93×10^{-03}(7)
0.90	6.54×10^{-01}(3005)	4.29×10^{-03}(12)	5.58×10^{-03}(8)
1.00	2.80×10^{-01}(1149)	3.90×10^{-03}(11)	5.51×10^{-03}(8)
2.00	9.26×10^{-02}(191)	2.85×10^{-03}(8)	5.51×10^{-03}(8)

λ	TH	NM	PI
0.50	7.70×10^{-02}(132)	2.33×10^{-03}(8)	5.70×10^{-03}(11)
0.70	7.65×10^{-02}(135)	2.18×10^{-03}(8)	5.61×10^{-03}(11)
0.80	9.36×10^{-02}(313)	2.45×10^{-03}(9)	4.62×10^{-03}(9)
0.84	7.31×10^{-01}(4561)	4.00×10^{-03}(13)	4.11×10^{-03}(8)

To show further results on the comparison between NM and PI, we report a number of graphs comparing the iteration count and CPU times of the two algorithms. The graphs are not cut at the critical values, but they extend to subcritical cases as well. It is an interesting point to note that when the MBT is subcritical, and thus the minimal solution x^* (vector of extinction probabilities) is e, the two algorithms have a different behavior: NM (and TH as well) converges to e, while PI skips this solution and converges to a different solution $x > e$. This is because in the derivation of the Perron iteration we chose the solution $\alpha \neq 0$ for the normalization equation, thus explicitly excluding the solution $y = 0$ (i.e., $x = e$).

Figure 7.1 shows a plot of the iteration count of the two methods vs. different values of the parameter λ. While in close-to-critical cases the iteration count for NM has a spike, the one for PI seems to decrease. However, the iteration count comparison is not fair since the steps of the two iterations require a different machine time. Figure 7.2 shows a similar plot, considering the CPU time instead of the iteration count. In order to achieve better accuracy, the plotted times are averages over 100 consecutive runs.



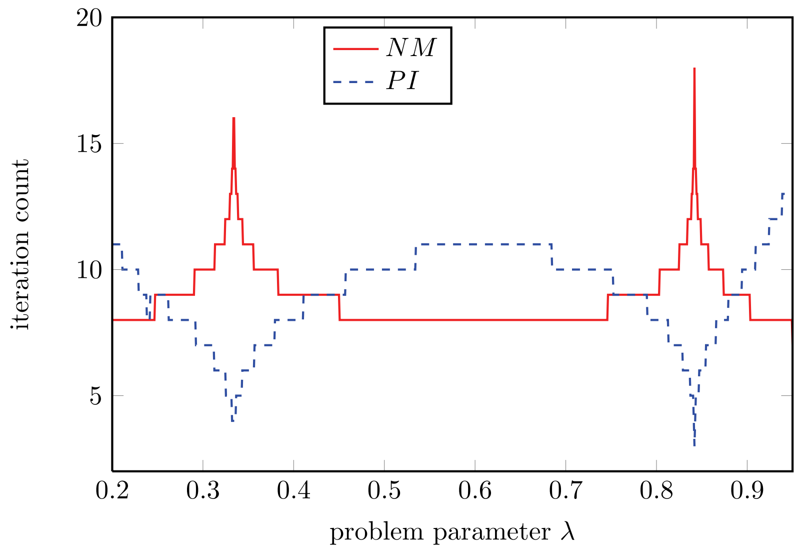
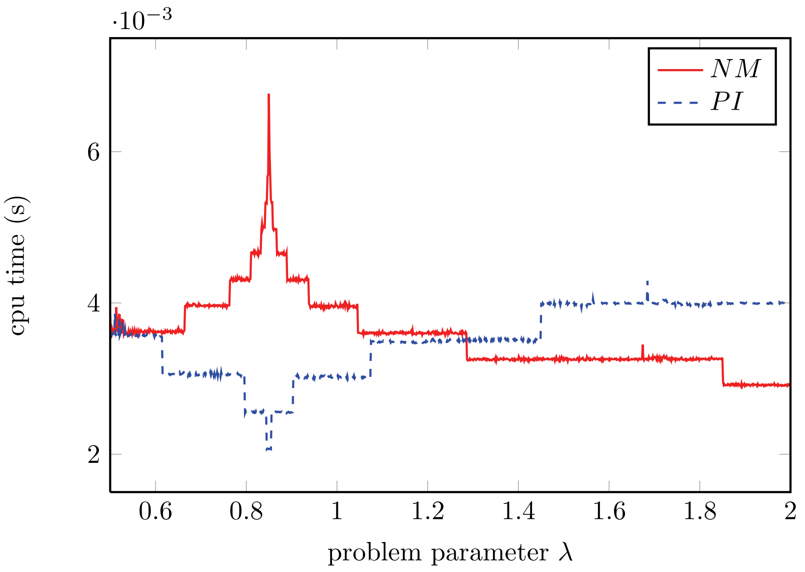


FIG. 7.1. *Number of iterations needed for* $E1$ *(top) and* $E2$ *(bottom).*

The results now favor the Newton method in most experiments, but in close-to-critical cases the new method achieves better performance. The results are very close to each other, though, so it is to be expected that for larger input sizes or different implementations the differences in the performance of the eigensolver could lead to significant changes in the results.

The Jacobian (6.6) had spectral radius less than 1 in all the above experiments, a condition which is needed to ensure the convergence of PI. However, this is not true for all possible MBTs. In fact, by setting the parameter λ for E1 to much larger values, we encountered problematic cases in which PI did not converge. Specifically, starting from $\lambda \approx 78$ the spectral radius of the Jacobian (6.6) is larger than 1 and PI does not converge. However, such cases are of little practical interest since they



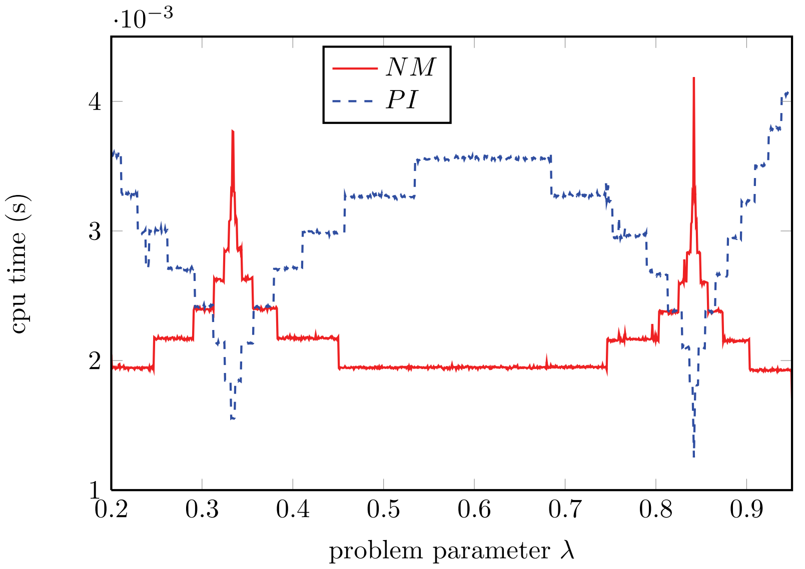


FIG. 7.2. *CPU time needed for solving E1 (top) and E2 (bottom).*

are highly supercritical MBTs, distant from the critical case, and thus they are easily solved with the traditional methods (NM or the customary functional iterations [2]) with a small number of iterations.

The problem E2 is well-posed only for $0 \leq \lambda \leq 1$. Otherwise negative entries appear in b; thus the above discussion does not apply.

In all the experiments reported above, all the matrices H_y appearing in the PI steps always turned out to have positive entries, even in the subcritical problems; thus their Perron vector and values were always well-defined and real.

8. Conclusions and open issues. We have proposed a new algorithm for solving the quadratic vector equation (1.1), based on a Perron iteration. The algorithm performs well, both in terms of speed of convergence and accuracy, for close-to-critical problems where the classical methods are slower.

Along the framework that we have exposed, several different choices are possible in the practical implementation of the new algorithm.

One of them is the choice of the bilinear form $b(\cdot,\cdot)$, as discussed in section 2. Like the *order* algorithm, our algorithm depends truly on the bilinear form, not only on its quadratic form restriction. Therefore, we may modify the bilinear form given by the problem in any way that keeps it compatible with the quadratic form. For instance, we may swap $\widetilde{b}(u,v) := b(v,u)$ as suggested above, or we may symmetrize it to $\widehat{b}(u,v) = \frac{1}{2}(b(u,v) + \widetilde{b}(u,v))$. This is discussed in more detail in [3], with several examples.

A second choice is the normalization of the computed Perron vector: different approaches may be attempted—for instance, minimization of the 1-norm, of the 2-norm, or orthogonality of the residual of (4.2) with respect to a suitably chosen vector—although it is not clear whether we can improve the results of the normalization presented here.

A third choice, crucial in the computational experiments, is the method used to compute the Perron vector. For moderate sizes of the problem, it is cheaper to do a full eigendecomposition of the matrix and extract the eigenvalue with maximum modulus, but for larger problems it pays to use different specific methods for its computation.

All these variants deserve to be better understood and are now under investigation.

Acknowledgment. The authors wish to thank Dario Bini and Guy Latouche for the pleasant discussions on the topics of this paper.

REFERENCES

[1] K. B. Athreya and P. E. Ney, *Branching Processes*, Dover, Mineola, NY, 2004. Reprint of the 1972 original [Springer, New York; MR0373040].

[2] N. G. Bean, N. Kontoleon, and P. G. Taylor, *Markovian trees: Properties and algorithms*, Ann. Oper. Res., 160 (2008), pp. 31–50.

[3] D. A. Bini, B. Meini, and F. Poloni, *On the solution of a quadratic vector equation arising in Markovian binary trees*, Numer. Linear Algebra Appl., submitted.

[4] S. Hautphenne, G. Latouche, and M.-A. Remiche, *Newton's iteration for the extinction probability of a Markovian binary tree*, Linear Algebra Appl., 428 (2008), pp. 2791–2804.

[5] S. Hautphenne, G. Latouche, and M.-A. Remiche, *Algorithmic approach to the extinction probability of branching processes*, Methodol. Comput. Appl. Probab., 13 (2011), pp. 171–192.

[6] S. Hautphenne and B. Van Houdt, *On the link between Markovian trees and tree-structured Markov chains*, European J. Oper. Res., 201 (2010), pp. 791–798.

[7] C. D. Meyer and G. W. Stewart, *Derivatives and perturbations of eigenvectors*, SIAM J. Numer. Anal., 25 (1988), pp. 679–691.

[8] F. Poloni, *Quadratic vector equations*, Linear Algebra Appl., submitted.

[9] R. S. Varga, *Matrix Iterative Analysis*, expanded ed., Springer Ser. Comput. Math., Springer, Berlin, New York, 2000.

[10] J. H. Wilkinson, *The algebraic eigenvalue problem*, in Monographs on Numerical Analysis, Clarendon, Oxford University Press, New York, 1988.