AN INNER/OUTER STATIONARY ITERATION FOR COMPUTING PAGERANK

ANDREW P. GRAY[*], CHEN GREIF[†], AND TRACY LAU[‡]

Abstract. We present a stationary iterative scheme for PageRank computation. The algorithm is based on a linear system formulation of the problem, uses inner/outer iterations, and amounts to a simple preconditioning technique. It is simple, can be easily implemented and parallelized, and requires minimal storage overhead. Convergence analysis shows that the algorithm is effective for a crude inner tolerance and is not particularly sensitive to the choice of the parameters involved. Numerical examples featuring matrices of dimensions up to approximately 10^7 confirm the analytical results and demonstrate the accelerated convergence of the algorithm compared to the power method.

Key words. PageRank, power method, stationary method, inner/outer iterations, damping factor

AMS subject classifications.
65F10, 65F15, 65C40

1. Introduction.

PageRank [19] is a method for ranking Web pages whereby a page's 'importance' (or ranking) is determined according to the link structure of the Web. This model has been used by Google as part of its search engine technology. The exact ranking techniques and calculation methods used by Google today are no longer public information, but the PageRank model has taken on a life of its own and has received considerable attention in the scientific community in the last few years. PageRank is essentially the stationary distribution vector of a Markov chain whose transition matrix is a convex combination of the matrix associated with the Web link graph and a certain rank-1 matrix. A key parameter in the model is the *damping factor*, a scalar denoted henceforth by α that determines the weight given to the Web link graph in the model. Due to the great size and sparsity of the matrix, methods based on decomposition are considered infeasible; instead, iterative methods are used, where the computation is dominated by matrix-vector products. Detailed descriptions of the problem and available algorithms can be found, for example, in [5, 17].

In this paper, we propose and investigate a new algorithm for the PageRank problem. It uses the linear system formulation and involves inner/outer iterations. The proposed technique is based on the observation that in general, the smaller the damping factor is, the easier it is to solve the problem. Hence we apply an iterative scheme in which each iteration requires solving another linear system which is similar in its algebraic structure to the original, but with a lower damping factor. In essence, what is proposed here is a simple preconditioning approach that exploits the spectral properties of the matrix involved. We use a technique of inexact solves, whereby the inner iteration is solved only to a crude tolerance. The algorithm is just a few lines long, and can be implemented and parallelized in a straightforward fashion.

The remainder of the paper is structured as follows. In Section 2 we provide a brief description of the PageRank problem. In Section 3 we introduce the proposed algorithm, derive convergence results and provide some guidelines for the selection of the parameters

[*]Faculty of Medicine, 317 - 2194 Health Sciences Mall, Vancouver, BC, V6T 1Z3, Canada, apgray@gmail.com. The work of this author was done in partial fulfillment of the requirements of an Honours B.Sc. degree in mathematics and computer science.

[†]Department of Computer Science, The University of British Columbia, Vancouver B.C. V6T 1Z4, Canada, greif@cs.ubc.ca. The work of this author was supported in part by the Natural Science and Engineering Council of Canada (NSERC).

[‡]Department of Computer Science, The University of British Columbia, Vancouver B.C. V6T 1Z4, Canada, tracylau@cs.ubc.ca. The work of this author was done as part of an NSERC summer undergraduate internship, May-August 2006.

involved. Numerical examples for a few large Web matrices are given in Section 4. Finally, in Section 5 we draw some conclusions.

2. The problem of computing PageRank. The 'raw' PageRank x_i of page i is defined as $x_i = \sum_{j \to i} \frac{x_j}{n_j}$, where $j \to i$ indicates that page j links to page i, and n_j is the outdegree of page j. Each page therefore shares its importance equally among all other pages to which it links; self-links are ignored. The problem in its basic form can thus be formulated as follows: find a vector x that satisfies $x = \bar{P}^T x$, where $\bar{P}$ is given by

$$\bar{P}_{ij} = \begin{cases} \frac{1}{n_i} & \text{if } i \to j, \\ 0 & \text{if } i \nrightarrow j. \end{cases}$$

Pages with no outlinks produce rows of all 0's in $\bar{P}$, hence $\bar{P}$ in its above form is not necessarily a stochastic matrix. This is handled in the model by eliminating zero rows, which is done by replacing $\bar{P}$ with, say,

$$P = \bar{P} + dv^T, \qquad d_i = \begin{cases} 1 & \text{if } n_i = 0, \\ 0 & \text{otherwise,} \end{cases}$$

Here the vector $v \geq 0$ is a probability vector, and now the modified matrix P is a proper stochastic matrix. We note that the correction dv^T is only one of various possibilities. The Ergodic theorem [12, Theorem 6.4.2] tells us that the stationary distribution is unique and is the limiting distribution starting from any initial distribution if the transition matrix is irreducible and aperiodic. In the case of PageRank, a convex combination of P^T with a rank-1 matrix has these desirable properties:

$$A = \alpha P^T + (1-\alpha) v e^T, \tag{2.1}$$

where $\alpha \in (0,1)$ is the damping factor, e is the vector of all 1s, and v is a positive vector. Our final definition of the PageRank vector, then, is $x = Ax$. Considered as a Markov chain, the model assumes that at each time step, a 'random' surfer either follows a link with probability α, or 'teleports' with probability $1-\alpha$, selecting the new page from the probability distribution given by v. The choice of a damping factor significantly smaller than 1 allows for an effective application of the power method. In the original formulation of PageRank [19], the choice $\alpha = 0.85$ was suggested. A higher value of α (i.e., close to 1) yields a model that more closely reflects the actual link structure of the Web, but makes the computation more difficult.

Notice that despite the fact that A is dense because ve^T is dense, it need not be explicitly formed since matrix-vector products of A with x can be efficiently computed by imposing $\|x\|_1 = 1$ (or, equivalently, $e^T x = 1$, since x is nonnegative):

$$Ax = \alpha P^{\mathrm{T}} x + \tilde{v},$$

where for notational convenience we define

$$\tilde{v} = (1-\alpha) v.$$

Thus, the power method for computing PageRank amounts to repeatedly applying $x \leftarrow \alpha P^T x + \tilde{v}$. If the initial guess has a unit 1-norm, then so do all the iterates x throughout this iteration, so normalization is not necessary, but it is carried out in practice for large scale problems in finite precision.

Since Brin and Page's original formulation of the PageRank problem, much work has been done by the scientific community to propose and investigate improvements over this algorithm. In [15] an extrapolation method is presented, which accelerates convergence by calculating and then subtracting off estimates of the contributions of the second and third eigenvectors. Analysis of the eigenstructure of the matrix and interesting results and observations about the sensitivity of the eigenvalue problem are given in [6, 20]. An Arnoldi-type technique is proposed in [10]. Other methods have also been considered: see [2, 4, 13, 16], which contain many additional results and useful references, and [3, 22] for books that include comprehensive overviews of nonnegative matrices and Markov chains.

3. An inner/outer stationary method.

We now present the new algorithm. As pointed out in [1, 9, 18], using $e^T x = 1$ the eigenvalue problem $x = Ax = \alpha P^T x + \tilde{v} e^T x$ can be reformulated as a linear system:

$$(I - \alpha P^T)x = \tilde{v}.$$

Inspired by the fact that the original problem is easier to solve when α is small, let us consider the stationary iteration

$$(I - \beta P^T)x_{k+1} = (\alpha - \beta)P^T x_k + \tilde{v}, \quad k = 1, 2, \dots \tag{3.1}$$

where $0 \leq \beta \leq \alpha$ is some parameter, and $x_0 = v$, the original teleportation vector. To solve (3.1), we will compute x_{k+1} using an inexact inner Richardson iteration as follows:

$$y_{j+1} = \beta P^T y_j + (\alpha - \beta)P^T x_k + \tilde{v}, \tag{3.2}$$

where we take $y_0 = x_k$ as the initial guess and assign the computed vector to which we converge as the new x_{k+1}. We should stress that the inner iteration we present here is just one of many possibilities; in this case we accelerate the power method, since power iterations applied to the eigenvalue problem are equivalent to Richardson-type iterations applied in the linear system setting. In practice, any solution method that one applies for the linear system formulation can be applied in the inner iteration stage. Thus, we can refer to the matrix $I - \beta P^T$ as a preconditioner.

The outer iterative scheme (3.1) is associated with the splitting

$$I - \alpha P^T = M_O - N_O \ ; \quad M_O = I - \beta P^T \ ; \quad N_O = (\alpha - \beta)P^T, \tag{3.3}$$

and the corresponding outer iteration matrix is given by

$$T_O = (\alpha - \beta)(I - \beta P^T)^{-1} P^T. \tag{3.4}$$

The inner scheme (3.2) is associated with the splitting

$$I - \beta P^T = M_I - N_I \ ; \quad M_I = I \ ; \quad N_I = \beta P^T, \tag{3.5}$$

and the corresponding inner iteration matrix is simply

$$T_I = \beta P^T. \tag{3.6}$$

The iterative procedure is presented in Algorithm 1; the parameters η and τ are the inner and outer tolerances respectively. The main challenge is to determine values of β and η that will accelerate the computation. In the extreme case $\beta = 0$ the inner problem can be solved immediately (regardless of η) but the outer iteration is equivalent to the power method. The other extreme leads to a similar situation: if $\beta = \alpha$ then the number of outer iterations is small (one iteration if $\eta = \tau$) but the inner iteration this time is equivalent to power iteration. As for

ALGORITHM 1
basic inner/outer iteration

```
1: $y \leftarrow P^T x$
2: **repeat** until $\|\alpha y + \tilde{v} - x\|_1 < \tau$
3:     $f \leftarrow (\alpha - \beta) y + \tilde{v}$
4:     **repeat** until $\|f + \beta y - x\|_1 < \eta$
5:         $x \leftarrow \beta y + f$
6:         $y \leftarrow P^T x$
7:     **end repeat**
8: **end repeat**
```

η, a value very close to zero (that is, very strict) may result in spending a long computational time performing inner iterations, just to compute a single outer iterate. This may result in slow convergence overall. Setting η very loose, on the other hand, may result in an iterate whose 'quality' is low in the sense that it does not sufficiently approximate the exact solution of the inner iteration. Our hope is that we can find intermediate choices of β and η that significantly reduce the overall work compared to the power method.

3.1. Convergence of the outer iterations. We start our analysis with the following convergence result.

PROPOSITION 3.1. *Given* $0 < \alpha < 1$, *if the inner iterations are solved exactly, the scheme converges for any* $0 \le \beta \le \alpha$. *Furthermore,*

$$\rho(T_O) = \frac{\alpha - \beta}{1 - \beta} < 1, \tag{3.7}$$

and hence the closer β *is to* α, *the greater the asymptotic rate of convergence is for the outer iterations.*

Proof. The matrix $I - \alpha P^T$ is a diagonally dominant M-matrix for any $0 < \alpha < 1$, hence so is M_O. Thus by known results for regular splittings [23, Theorem 3.32] convergence is guaranteed and the larger β is, the faster the outer iterations converge.

To obtain the spectral radius exactly, note that if λ_i is an eigenvalue of P^T then

$$\mu_i = \frac{(\alpha - \beta)\lambda_i}{1 - \beta\lambda_i}$$

is an eigenvalue of T_O. Since $|\lambda_i| \le 1$, we get

$$|\mu_i| = \left| \frac{(\alpha - \beta)\lambda_i}{1 - \beta\lambda_i} \right| \le \frac{(\alpha - \beta)|\lambda_i|}{1 - \beta|\lambda_i|} = \frac{\alpha - \beta}{|\lambda_i|^{-1} - \beta} \le \frac{\alpha - \beta}{1 - \beta},$$

with equality holding for $\lambda_1 = 1$, so $\rho(T_O)$ is as given in (3.7). □

As mentioned above, in practice we will want to accelerate convergence by solving the inner iteration inexactly. Inner/outer stationary iterations for solving linear systems have been considered in [7, 8, 11] and other papers, where possible choices of parameters and estimates of the overall computational work are given. Recent work on Krylov subspace methods can be found in [21].

In the following, we derive bounds on convergence of the inexact iteration. Given a linear system $Bx = g$ and a splitting $B = M - N$, consider a stationary scheme based on inexact solves as follows:

$$Mx_{k+1} = Nx_k + g + \delta_k.$$

Defining $e_k = x_k - x$ as the error in the kth iteration, we have

$$e_{k+1} = Te_k + M^{-1}\delta_k, \tag{3.8}$$

where $T = M^{-1}N$. Suppose

$$\|\delta_k\| \le \eta\|e_k - e_{k-1}\| \tag{3.9}$$

for some η. Note that the bound in (3.9) is computable since $e_k - e_{k-1} = x_k - x_{k-1}$. Combining (3.8) and (3.9) we have

$$\|e_{k+1}\| \le \|T\|\,\|e_k\| + \eta\|M^{-1}\|\,\|e_k - e_{k-1}\|.$$

Defining $\rho = \|T\|$ and $\sigma = \eta\|M^{-1}\|$ and applying the triangular inequality, the resulting inequality

$$\|e_{k+1}\| \le \rho\|e_k\| + \sigma(\|e_k\| + \|e_{k-1}\|)$$

involves a three-term relation. Let ν_k be the solution of the recurrence relation

$$\nu_{k+1} = \rho\nu_k + \sigma(\nu_k + \nu_{k-1}),$$

with $\nu_1 = \|e_1\|$ and $\nu_2 = \|e_2\|$. Then

$$\|e_k\| \le \nu_k = a_1\xi_+^k + a_2\xi_-^k,$$

where

$$\xi_\pm = \frac{\rho+\sigma}{2}\left(1 \pm \sqrt{1 + \frac{4\sigma}{(\rho+\sigma)^2}}\right)$$

and

$$a_1 = \frac{2(\nu_2 - \nu_1\xi_-)}{(\rho+\sigma)^2 s(s+1)}\ ; \qquad a_2 = \frac{2(\nu_2 - \nu_1\xi_+)}{(\rho+\sigma)^2 s(s-1)}\ ,$$

with $s = \sqrt{1 + 4\sigma/(\rho+\sigma)^2}$. Since $\xi_- < 0$, we have that $a_1 \ge 0$ and $\xi_+ > |\xi_-|$. It follows that

$$\|e_k\| \le a_1\xi_+^k + |a_2||\xi_-|^k \le (a_1 + |a_2|)\xi_+^k. \tag{3.10}$$

Hence we have a bound of the type $\|e_k\| \le \vartheta\xi_+^k$, where ϑ is independent of k. To proceed from here, we choose to work with the 1-norm and use the following auxiliary result:

PROPOSITION 3.2. *For the iteration (3.1), we have* $\|M_O^{-1}\|_1 = \frac{1}{1-\beta}$ *and* $\|T_O\|_1 = \frac{\alpha-\beta}{1-\beta}$.

Proof. The result for $\|M_O^{-1}\|_1$ readily follows by [14, Lemma 5]. For T_O we notice that since P^T and $(I - \alpha P^T)^{-1}$ commute, we have $\|T_O\|_1 = (\alpha - \beta)\|P^T(I - \alpha P^T)^{-1}\|_1$. We can again use the strategy in [14, lemma 5], as follows. Let $\tilde{A}$ be the matrix defined in (2.1) but with the specific choice $v = e_i$, the ith standard basis vector. Suppose $\tilde{x}$ is the eigenvector we seek: $\tilde{A}\tilde{x} = \tilde{x}$. Then, if $\|\tilde{x}\|_1 = 1$ we have $\tilde{x} = (1-\beta)(I - \beta P^T)^{-1}e_i$. Multiplying both sides by P^T and taking norms gives

$$1 = \|P^T\tilde{x}\|_1 = (1-\beta)\|P^T(I - \beta P^T)^{-1}e_i\|_1.$$

Hence $\|T_O\|_1 = \frac{\alpha-\beta}{1-\beta}$, as claimed. □

The bound (3.10) motivates us to find η such that $\xi_+ < 1$. By Prop. 3.2 we substitute $\rho = \frac{\alpha-\beta}{1-\beta}$ and $\sigma = \frac{\eta}{1-\beta}$. Defining $\gamma = \alpha - \beta$ and using $\sqrt{1+x} \lesssim 1 + \frac{x}{2}$ for $x \ll 1$, we obtain

$$\xi_+ = \frac{\rho+\sigma}{2}\left(1+\sqrt{1+\frac{4\sigma}{(\rho+\sigma)^2}}\right) \lesssim \frac{\gamma+\eta}{1-\beta} + \frac{\eta}{\gamma+\eta}.$$

Requiring $\xi_+ < 1$, we obtain the quadratic inequality $\eta^2 + 2\gamma\eta + \gamma^2 + \gamma(\beta-1) < 0$, which yields

$$0 < \eta < -(\alpha-\beta) + \sqrt{(\alpha-\beta)(1-\beta)}. \tag{3.11}$$

We can obtain a simpler expression which approximates the bound by assuming that η is sufficiently small and ignoring the second order term η^2 in the quadratic inequality. Solving the corresponding linear inequality, we get $0 < \eta < \frac{1-\alpha}{2}$. From this we make the observation that the closer α is to 1, the smaller η should be chosen to be.

In Fig. 3.1 we experimentally examine how ξ_+ relates to β and η. These graphs show that convergence is expected for a large range of values of these parameters. It is worth noting that our analysis only provides sufficient conditions for convergence, and so for $\xi_+ > 1$ we may still have convergence.

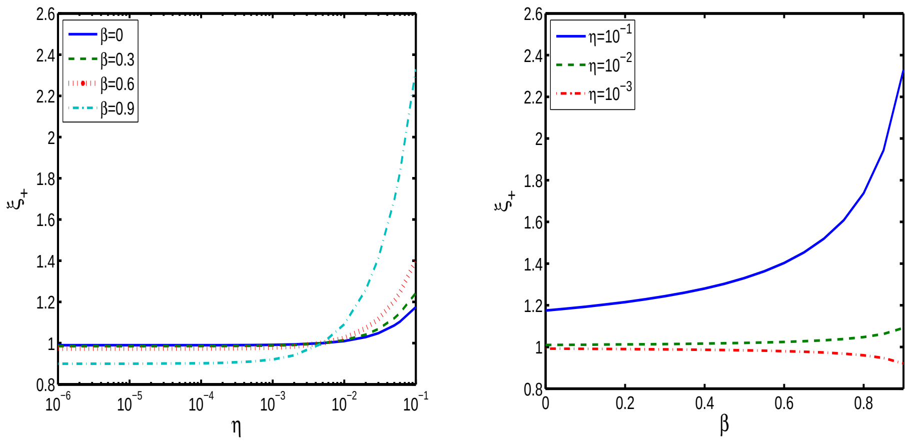


FIG. 3.1. ξ_+ *for* $\alpha = 0.99$*, and various values of* η *and* β*. The algorithm is guaranteed to converge when* $\xi_+ < 1$.

3.2. Convergence rate of the inner iterations, and an improved algorithm. We now consider the rate of convergence of the inner iterations (3.2) and the dependence on the parameters α and β. From (3.6) it follows that asymptotically, the error is reduced by a factor of approximately β in each inner iteration. It is possible to derive a formula for the error between a given inner iterate and the PageRank vector as follows. Define

$$\epsilon_j = y_j - x.$$

By the linear system formulation, $x = \alpha P^T x + \tilde{v}$. Subtracting this from (3.2), we get

$$\begin{aligned}\epsilon_{j+1} &= \beta P^T y_j - \beta P^T x_k + \alpha P^T e_k \\ &= \beta P^T \epsilon_j + (\alpha-\beta) P^T e_k,\end{aligned}$$

which leads to

$$\epsilon_j = \beta^j (P^T)^j \epsilon_0 + (\alpha - \beta) \sum_{i=0}^{j} \beta^{i-1} (P^T)^i e_k.$$

Notice that $y_0 = x_k$, and hence $\epsilon_0 = e_k$. This can further simplify the expression for the error and shows that ϵ_j is a j-th degree polynomial in P^T, dependent on α and β and multiplied by ϵ_0.

While we cannot easily prove that the iteration counts for the inner solves monotonically decrease as we get closer to the solution of the problem, the above expression for the error indicates that when $y_0 = x_k$ is sufficiently close to x, the inner iterations will rapidly converge, to the point of immediate convergence. This motivates us to incorporate an improvement in Algorithm 1: when the inner iterations start converging immediately we switch to the power method, which spares us the need to check the inner convergence criterion. The modified scheme is presented in Algorithm 2. This is the algorithm we use henceforth in our numerical examples.

ALGORITHM 2
inner/outer iteration

```
1: $y \leftarrow P^T x$
2: **repeat** until $\|\alpha y + \tilde{v} - x\|_1 < \tau$
3:     $f \leftarrow (\alpha - \beta) y + \tilde{v}$
4:     **for** $i = 1$ until $\|f + \beta y - x\|_1 < \eta$
5:         $x \leftarrow \beta y + f$
6:         $y \leftarrow P^T x$
7:     **end for**
8:     **if** i=1, power($\alpha y + \tilde{v}$); **break**
9: **end repeat**
```

3.3. Computational cost of the algorithm. Each iteration in Algorithm 2 is computationally inexpensive, but it does nevertheless involve some additional work compared to the power method, so it is useful to quantify this in precise terms.

Lines 2 and 3 entail three SAXPY operations (here $\alpha y + \tilde{v}$ is computed once and can be used twice), and one 1-norm computation. The inner convergence criterion (line 4) entails one SAXPY operation and one 1-norm computation. The operations inside the inner loop (lines 5–6 in Algorithm 2) include a computation of $P^T x$, which typically accounts for the most computationally costly component. (Notice that $f + \beta y$ is used twice, in lines 4 and 5, but is computed only once.) As noted in [15, Algorithm 1], the computation of $P^T x$ involves more than mere a matrix-vector product, because for dealing with dangling nodes, our matrix is in fact of the form $P = \bar{P} + dv^T$, as explained in Section 2. A computationally efficient way to form matrix-vector products with this matrix is done in three steps [15]: $y \leftarrow P^T x$, set $w = ||x||_1 - ||y||_1$, and finally $y \leftarrow y + wv$. Thus, computing y essentially requires one matrix-vector product, two 1-norm computations, and one SAXPY operation. (Operations between scalars are ignored.)

Thus, when both an outer and an inner iteration are carried out, we have one matrix-vector product, four 1-norm computations, and six SAXPY operations. And when only inner iterations are performed, the cost is one matrix-vector product, three 1-norm computations, and three SAXPY operations per iteration. We do this only until the inner iterations start converging within one iteration, at which point we switch to the power method. In the next section we show experimentally that this typically happens quite quickly.

Comparing this to cost of the power method can be done straightforwardly as follows. In one power iteration we have one matrix-vector product, three 1-norm computations, and two SAXPY operations per iteration (see [15, Algorithm 2]). In our algorithm, when the inner iteration is being carried out and has not terminated yet, we have only one extra 1-norm computation and one extra SAXPY operation. When we compute the next outer iteration, we have one additional 1-norm computation and four extra SAXPY operations.

Given that matrix-vector products require additions as well as multiplications approximately equal to the number of nonzeros in the matrix, a realistic estimate of the cost of a single inner/outer iteration leads to the conclusion that the overhead is typically marginal. For example, if a matrix has 10 nonzeros per row on average, the overhead of an inner iteration is approximately 15% of a power iteration; if an outer iteration is also performed, the overhead rises to approximately 35%. However, this overhead is only incurred in the first few iterations, before we switch to the power method, so the overall overhead is expected to be considerably less than 15%. In the next section, data from numerical experiments (CPU execution times in particular) confirm that the overhead is, on the whole, quite small compared to the savings in matrix-vector products.

4. Numerical experiments. We have implemented the proposed inner/outer method using MATLAB MEX files. We ran experiments on a Linux workstation with a 2.53 GHz P4 processor and 2 GB of main memory, and used Web matrices whose dimensions and number of nonzeros are provided in Table 4.1. We used outer tolerances τ ranging from 10^{-3} to 10^{-7}, and damping factors from $\alpha = 0.85$ to $\alpha = 0.99$. As an initial guess we took $x_0 = v = \frac{1}{n}e$, and ran the algorithm for various values of β and η. As our speedup criterion, we will refer below to a *relative percentage gain* measure given by

$$\frac{I_p - I_s}{I_p} \cdot 100\%, \tag{4.1}$$

where I_p and I_s represent the power and inner/outer stationary iteration counts, respectively.

name	size	nz	avg nz per row
`UBC-CS`	51,681	673,010	13.0
`Stanford`	281,903	2,312,497	8.2
`UBC`	339,147	4,203,811	12.4
`Stanford-Berkeley`	683,446	7,583,376	11.1
`edu`	2,024,716	14,056,641	6.9
`wb-edu`	9,845,725	57,156,537	5.8

TABLE 4.1

Dimensions and number of nonzeros of a few test matrices. The UBC matrices were generated using a Web crawler developed by the first author. The Stanford and Stanford-Berkeley matrices were retrieved from `http://www.stanford.edu/~sdkamvar`*. The edu and wb-edu matrices were provided by Yahoo! Research Laboratory.*

Results for the inner/outer method applied to the `Stanford-Berkeley` matrix are presented in Fig. 4.1. On the left-hand graph we see that for a loose inner tolerance η there is only a narrow range of values of β for which the inner/outer method converges much more quickly than the power method; see the convergence graph for $\eta = 0.1$. When η is very strict the overall computational work is large for almost all values of β, due to a large number of inner iterations; see the graph for $\eta = 10^{-5}$. Significant gains are observed for

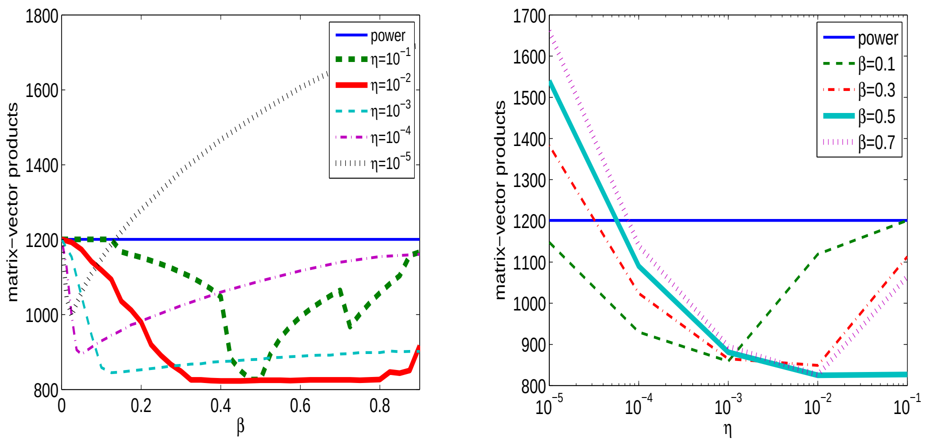


FIG. 4.1. *Total number of matrix-vector products required for convergence of the inner/outer scheme, for the* `Stanford-Berkeley` *matrix. (Outer Tolerance* 10^{-7}*,* $\alpha = 0.99$*,* β *and* η *varied.)*

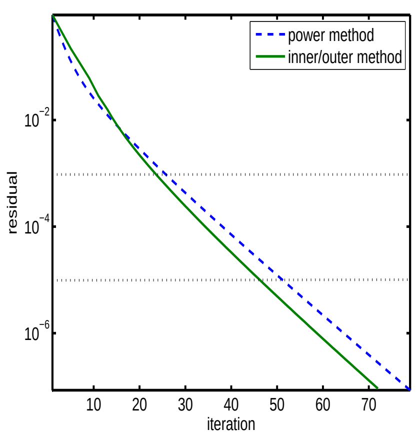
moderate values of η; see, for example, the graph for $\eta = 0.01$. In this case, the performance of the scheme is not sensitive to the choice of β. We have observed similar behavior for our other test matrices: choosing $\eta \approx 10^{-2}$ and $0.4 \lesssim \beta \lesssim 0.8$ has consistently led to accelerated convergence. The right-hand graph in Fig. 4.1 shows similar behavior for various fixed values of β, with η varying. The conclusion is that moderate values of both β and η reduce the overall computational work and are fairly insensitive to small perturbations.

We now compare our inner/outer method to the power method. In our experiments we use the values $\beta = 0.5$ and $\eta = 10^{-2}$. We plot in Fig. 4.2 the norms of the residuals for both methods run on the large `wb-edu` matrix for two values of α. As discussed in Section 3.2, the gains in the inner/outer method are made in the *initial* iterates, and the gains are most significant when $\alpha = 0.99$ and the outer tolerance is loose. This can be observed in Fig. 4.2, where for $\tau = 10^{-3}$ we have a relative gain of 57%. From Table 4.3 we can see that for the other matrices the savings range from 17% to 41% for $\tau = 10^{-7}$, and from 38% to 74% for $\tau = 10^{-3}$. When the outer tolerance is stricter (10^{-7}), the inner/outer scheme achieves a relative gain of 9% for $\alpha = 0.85$ (72 matrix-vector products compared to 79 for the power method), which is fairly marginal. On the other hand, when $\alpha = 0.99$ the inner/outer stationary method achieves a substantial relative gain of 28%: 328 fewer matrix-vector products than 1159.

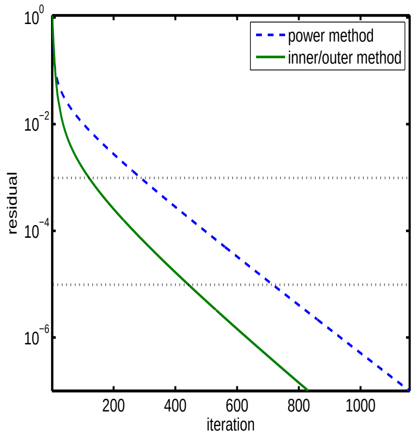
Comparing the savings in iteration counts (Table 4.3) to the savings in CPU time (Table 4.4) confirms our claim in Section 3.3 that the overhead introduced by the inner/outer method is quite small compared to the savings.

Table 4.2 shows that the inner iteration counts per outer iteration decrease monotonically in practice and reach one fairly quickly. From the table we can see that it takes 24 inner iterations overall (within 9 outer iterations) until the inner iterates start converging immediately, at which point we switch to the power method. Tables 4.3 and 4.4 show that overall, the gains are substantial and do indeed strongly dominate the overhead.

We end this section with a brief reference to the merits of our scheme in comparison with the well known Quadratic Extrapolation scheme [15]. The speed of convergence for both methods is similar; see for example [15, Fig. 7], and in both methods the gains are made in initial iterates. However, our outer/inner method has two distinct advantages. It is simple, relying exclusively on matrix-vector products and norm computations. It also has lower space



$\alpha = 0.85$



$\alpha = 0.99$

FIG. 4.2. *Convergence of the computation for the* $9,845,725 \times 9,845,725$ `wb-edu` *matrix* ($\tau = 10^{-7}$, $\beta = 0.5$ *and* $\eta = 10^{-2}$ *in the inner/outer method)*

outer iteration	# inner iterations
1-2	4
3-4	3
5-9	2
10-...	1

TABLE 4.2

Number of inner iterations required for each outer iteration when applying the inner/outer stationary method to `wb-edu` *matrix with* $c = 0.99$, $b = 0.5$, *and* $\eta = 10^{-2}$. *Total iteration counts and CPU times for this example can be found in the last rows of Table 4.3 and Table 4.4.*

requirements: it involves only three working vectors (x, y, f in Algorithms 1 and 2), whereas in Quadratic Extrapolation six vectors are required for recording iterates and solving the least squares problem.

5. Conclusions.

We have presented an inner/outer stationary method for accelerating PageRank computations. Our algorithm is simple, fast, and introduces minimal overhead. Because no permutations, projections, orthogonalizations or decompositions of any sort are involved, programming it is straightforward, and it is highly parallelizable.

The algorithm is parameter-dependent, but an effective choice of the parameters can be easily made. We have provided a detailed convergence analysis and have shown analytically and experimentally that the proposed technique is effective for a large range of inner tolerances. Observing that the gains are made in the initial iterates, our scheme switches to the power method once the inner iterations start converging immediately. For $\alpha = 0.85$ our algorithm marginally outperforms the power method, but for values of α closer to 1 the gains are quite substantial.

The mechanism of inner/outer iterations allows for much flexibility, and a Richardson-type inner iteration is only one possibility; in fact, any linear solver that is effective for the linear system formulation of PageRank computations can be incorporated into our inner/outer scheme. It is most natural to think of our approach as a preconditioning technique, where the preconditioner is strongly connected with the underlying spectral properties of the matrix.

outer tolerance	10^{-3}			10^{-5}			10^{-7}		
matrix	power	in/out	savings	power	in/out	savings	power	in/out	savings
`UBC-CS`	226	140	38.1%	574	431	24.9%	986	814	17.4%
`Stanford`	281	120	57.3%	716	426	40.5%	1165	790	32.2%
`UBC`	242	140	42.1%	676	483	28.6%	1121	855	23.7%
`Stan-Berk`	309	120	61.2%	751	448	40.3%	1202	825	31.4%
`edu`	426	111	73.9%	882	410	53.5%	1339	786	41.3%
`wb-edu`	287	120	58.2%	714	441	38.2%	1159	831	28.3%

TABLE 4.3

Total number of matrix-vector products required for convergence to three different outer tolerances τ, and the corresponding relative gains defined by (4.1). The parameters used here are $\alpha = 0.99,\ \beta = 0.5,\ \eta = 10^{-2}$.

outer tolerance	10^{-3}			10^{-5}			10^{-7}		
matrix	power	in/out	savings	power	in/out	savings	power	in/out	savings
`UBC-CS`	3.0	2.0	33.3%	7.6	6.0	21.1%	13.0	11.3	13.1%
`Stanford`	43.3	19.7	54.5%	110.9	69.5	37.3%	180.7	124.5	31.1%
`UBC`	21.1	12.8	39.3%	59.1	43.3	26.7%	97.2	77.6	20.2%
`Stan-Berk`	42.4	17.6	58.5%	104.8	64.3	38.6%	166.5	117.2	29.6%
`edu`	175.1	48.6	72.2%	363.7	173.2	52.4%	546.6	333.0	39.1%
`wb-edu`	452.5	198.8	56.1%	1128.5	704.4	37.6%	1814.3	1318.0	27.4%

TABLE 4.4

CPU times (in seconds) required for convergence to three different outer tolerances τ, and the corresponding relative gains defined by (4.1). The parameters used here are $\alpha = 0.99,\ \beta = 0.5,\ \eta = 10^{-2}$.

Future work may include investigating how to dynamically determine the parameters β and η, and exploring the performance of the algorithm as an acceleration technique for other methods of PageRank computation. It may also be possible that the proposed technique is applicable as a preconditioner for general Markov chains.

The excellent performance of the method for the large matrices we have tested, along with its simplicity and modest storage requirements, suggest that this scheme may be very effective for PageRank computations.

References.

[1] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin. PageRank computation and the structure of the Web: Experiments and algorithms. In *The Eleventh International WWW Conference*, May 2002.

[2] P. Berkhin. A survey on PageRank computing. *Internet Math.*, 2:73–120, 2005.

[3] A. Berman and R.J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. SIAM, Philadelphia, 1994.

[4] C. Brezinski and M. Redivo-Zaglia. The PageRank vector: properties, computation, approximation, and acceleration. *SIAM J. Matrix Anal. Appl.*, 28(2):551–575, 2006.

[5] K. Bryan and T. Leise. The $25,000,000,000 eigenvector: The linear algebra behind Google. *SIAM Review*, 48(3):569–581, 2006.

[6] L. Eldén. A note on the eigenvalues of the Google matrix. *Report LiTH-MAT-R-04-01, Linköping University*, 2003.

[7] H.C. Elman and G.H. Golub. Inexact and preconditioned Uzawa algorithms for saddle point problems. *SIAM J. Numer. Anal.*, 31(6):1645–1661, 1994.

[8] E. Giladi, G.H. Golub, and H.B. Keller. Inner and outer iterations for the Chebyshev algorithm. *SIAM J. Numer. Anal.*, 35(1):300–319, 1998.

[9] D. Gleich, L. Zhukov, and P. Berkhin. Fast parallel PageRank: a linear sys-

tem approach. *Yahoo! Research Technical Report YRL-2004-038, available via http://research.yahoo.com/publication/YRL-2004-038.pdf*, 2004.

[10] G.H. Golub and C. Greif. An Arnoldi-type algorithm for computing PageRank. *BIT*, 46(4):759–771, 2006.

[11] G.H. Golub and M.L. Overton. The convergence of inexact Chebyshev and Richardson iterative methods for solving linear systems. *Numer. Math.*, 53:571–593, 1988.

[12] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, Oxford, third edition, 2001.

[13] I.C.F. Ipsen and R. S. Wills. Mathematical properties and analysis of Google's PageRank. *Bol. Soc. Esp. Mat. Apl.*, to appear.

[14] S.D. Kamvar and T.H. Haveliwala. The condition number of the PageRank problem. *Technical Report, Stanford University, available via http://dbpubs.stanford.edu:8090/pub/2003-36*, 2003.

[15] S.D. Kamvar, T.H. Haveliwala, C.D. Manning, and G.H. Golub. Extrapolation methods for accelerating PageRank computations. In *Proceedings of the 12th International Conference on the World Wide Web*, Stanford Technical Report: http://dbpubs.stanford.edu:8090/pub/2003-17, 2003.

[16] A.N. Langville and C.D. Meyer. A survey of eigenvector methods for Web information retrieval. *SIAM Review*, 47(1):135–161, 2005.

[17] A.N. Langville and C.D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.

[18] C.B. Moler. *Numerical Computing with Matlab*. SIAM, 2003.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the web. *Stanford Digital Libraries, available via http://dbpubs.stanford.edu:8090/pub/1999-66*, 1999.

[20] S. Serra-Capizzano. Jordan canonical form of the Google matrix: a potential contribution to the PageRank computation. *SIAM J. Matrix Anal. Appl.*, 27:305–312, 2005.

[21] V. Simoncini and D.B. Szyld. Theory of inexact krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.*, 25:454–477, 2003.

[22] W.J. Stewart. *Introduction to the numerical solution of Markov chains*. Princeton University Press, Princeton, NJ, 1994.

[23] R.S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, 1962.