# Acceleration of inexact inverse iteration for eigenvalue problems

Alan L. Andrew[1] P. H. Foo[2] Roger C. E. Tan[3]
Markus Wächter[4]

(Received 24 July 2008; revised 23 October 2008)

**Abstract**

Many methods have been used to improve the efficiency of iterative numerical algorithms. Combining different methods is not always possible because the performance of acceleration methods usually depends critically on the precise form of the error in successive iterates, and this form often changes when other acceleration methods are used. Inexact implementation methods have proved particularly effective in increasing the efficiency of iterations involving sparse matrices. This article investigates the extent to which the efficiency of inexact inverse iteration and the inexact Rayleigh quotient algorithm, for the numerical computation of eigenvalues and eigenvectors of sparse matrices, may be further increased by the use of the scalar epsilon algorithm, a classical extrapolation technique. Some encouraging numerical results are presented and some pointers are given for future research.

http://anziamj.austms.org.au/ojs/index.php/ANZIAMJ/article/view/1369 gives this article, © Austral. Mathematical Soc. 2008. Published November 6, 2008. ISSN 1446-8735. (Print two pages per sheet of paper.)

# Contents



## 1 Inexact inverse iteration

Inverse iteration [25] is the method of choice [15] for computing eigenvectors of matrices when good approximations of the corresponding eigenvalues are already known. For large matrices, it is also a popular method for simultaneous computation of one or more eigenvalues and the corresponding eigenvectors when initial approximations of the eigenvalues are not available.

We consider the problem of computing the eigenvalue of a $p \times p$ matrix $A$ closest to a given number $\sigma$, and simultaneously computing the corresponding eigenvector. Let the eigenvalues and the corresponding eigenvectors of $A$ be $\lambda_1, \ldots, \lambda_p$ and $x_1, \ldots, x_p$ respectively, and let

$$0 < |\lambda_1 - \sigma| < |\lambda_2 - \sigma| \leq \cdots \leq |\lambda_p - \sigma| \,. \tag{1}$$

The recurrence relation used in inverse iteration is

$$(A - \sigma I)u_{k+1} = \alpha_k u_k \,, \tag{2}$$

where the scalar $\alpha_k$ is a normalizing factor. Then, for all $k \in \mathbb{N}$,

$$u_k = \left[ x_1 + \sum_{j=2}^{p} \left( \frac{a_j}{a_1} \right) \left( \frac{\lambda_1 - \sigma}{\lambda_j - \sigma} \right)^k x_j \right] \frac{a_1 \prod_{j=0}^{k-1} \alpha_j}{(\lambda_1 - \sigma)^k} \,, \tag{3}$$

where $u_0 = \sum_{j=1}^{p} a_j x_j$, so that, for appropriately chosen $\alpha_k$, $u_k \to x_1$ as $k \to \infty$. (In practice, roundoff ensures that $u_k \to x_1$ even in the exceptional case $a_1 = 0$ [25]; the inexact procedures described here are even more advantageous in this case.) If the $\alpha_k$ in (2) are chosen so that, for all $k$,

$$u_{k+1}^* u_k = u_k^* u_k \,, \tag{4}$$

then it follows from (1), (3) and the binomial theorem that $u_k \to x_1$ and $\alpha_k \to \lambda_1 - \sigma$ as $k \to \infty$, and there exist constants $c_j$ and $d_j$, independent of $k$, such that

$$\frac{\lambda_1 - \sigma}{\alpha_k} = 1 + \sum_{j=1}^{\infty} c_j d_j^k \,, \tag{5}$$

and $|d_j| \le |\lambda_1 - \sigma|/|\lambda_2 - \sigma| < 1$, for all $j$.

If $\sigma$ is sufficiently close to $\lambda_1$, a single iteration is often sufficient [25]. We are concerned with the case in which a good initial approximation of $\lambda_1$ is not available. In this case, convergence can be quite slow, and we seek methods of improving efficiency.

When $A$ is large and sparse, (2) is normally solved by iterative methods. Computation of each $u_k$ (a single 'outer iteration') then requires several 'inner iterations', which compute successive approximations of that $u_k$. Computational cost is roughly proportional to the total number of inner iterations. This number can be reduced substantially by using 'inexact inverse iteration' [11, 13, 14, 16, 22]. The convergence rate of iterative methods is usually not significantly affected if the initial iterates are computed less accurately than the later ones. The effect of inexact computation of the initial iterates is comparable to that of using a slightly different initial approximation. The situation is similar with inexact implementations of Newton-like methods [4, 9, 18]. This contrasts with inexact implementation of Krylov methods, which require highest accuracy in the early iterations [6, 20]. With inexact inverse iteration, the exit criteria for the inner iteration are less demanding for the early steps of the outer iteration than for the later ones.

The efficiency of the method depends on the choice of exit criteria for the inner iteration [14, 16].

A popular method of accelerating the convergence of (2) for Hermitian $A$ is Rayleigh quotient iteration (RQI), in which the constant shift $\sigma$ in (2) is replaced by the variable shift $\sigma_k = u_k^* A u_k / u_k^* u_k$. Inexact implementation of the Rayleigh quotient iteration has been considered by various authors [5, 13, 19, 21, 22]. Other variable shifts were considered by Spence et al. [11, 12]. For non-Hermitian $A$, Rayleigh quotient methods using $\sigma_k = w_k^* A u_k / w_k^* u_k$, where $w_k$ is the current approximation to the *left* eigenvector, can also be used, although this nearly doubles the amount of calculation required for each iteration. Provided the eigenvalue is not too ill-conditioned, the shift $\sigma_k = u_k^* A u_k / u_k^* u_k$ can also be useful in the non-Hermitian case. We used $\sigma_k = \hat{u}_k^* A \hat{u}_k / \hat{u}_k^* \hat{u}_k$ in our calculations, where $\hat{u}_k$ denotes the approximation to $u_k$ obtained when (2) is solved inexactly by terminating the inner iteration before convergence is obtained.

The substantial advantages of properly implemented inexact methods are clear, and we are not aware of any serious disadvantages. A possible minor disadvantage might be a reduction in the effectiveness of certain extrapolation techniques, as these are particularly sensitive to the exact asymptotic form of the error. For example, the asymptotic form of the error $u_k - x_1$ in (3) as $k \to \infty$ is ideal for Wynn's $\varepsilon$-algorithm [2, 8, 26]. Inexact implementation destroys this ideal form, making the use of the $\varepsilon$-algorithm more risky. This article describes our experience using the $\varepsilon$-algorithm with inexact methods. Our numerical results support the hypothesis that the $\varepsilon$-algorithm can still be effective when appropriate inexact methods are used, provided that $\|\hat{u}_k - u_k\|$ is sufficiently small compared with $\|u_k - x_1\|$. In this case, the optimum choice of exit criteria for the inner iteration still presents a challenge. If they are too strict then too many inner iterations per step will be demanded, but if they are not strict enough the extrapolation may fail.

We used the $\varepsilon$-algorithm to accelerate the convergence of both the inexact inverse iteration algorithm of Lai et al. [16] and the inexact RQI, both with

various exit criteria, using the same test matrices as Lai et al. These matrices are real but non-symmetric. Our algorithms are described in Section 2 and our numerical results presented in Section 3. We have not attempted to compare the performance of the ε-algorithm with that of other extrapolation methods. We simply used the ε-algorithm to show that inexact implementation does not necessarily prevent the effective use of extrapolation.

## 2 Using the ε-algorithm

A good introduction to the theory of the ε-algorithm is given by Brezinski and Redivo Zaglia [8]. Given a sequence $\{\beta_k\}$ of scalars, the scalar ε-algorithm (SEA) computes the double sequence $\varepsilon_n^{(k)}$ defined by

$$\varepsilon_{n+1}^{(k)} = \varepsilon_{n-1}^{(k+1)} + \frac{1}{\varepsilon_n^{(k+1)} - \varepsilon_n^{(k)}}, \quad k, n = 0, 1, \ldots, \tag{6}$$

where, for all $k$, $\varepsilon_{-1}^{(k)} = 0$ and $\varepsilon_0^{(k)} = \beta_k$. When applied to sequences of the form (5), this algorithm effectively eliminates the most slowly decaying error terms. Vector variants of the algorithm can also be applied to $u_k$, and have been used to compute matrix eigenvalues and eigenvectors [7] and their sensitivities [2]. However, these vector variants require much more computational effort, and in this case it is better to apply the original scalar ε-algorithm (SEA) to the sequence $(\alpha_k)^{-1}$. This produces a sequence, $\left(\varepsilon_k^{(0)}\right)^{-1} + \sigma_{k+1}$, which converges more rapidly to $\lambda_1$. Having a more accurate eigenvalue estimate then enables eigenvectors to be computed more efficiently. Our aim is to test whether the ε-algorithm can also be useful with inexact methods, when the $\alpha_k$ no longer satisfy (5) exactly.

All our algorithms are readily derived from Algorithm 1 below, which describes our implementation of the inexact Rayleigh quotient algorithm with the SEA. To facilitate comparison of our algorithms with that of Lai et al. [16], we use a similar format. In particular, we use generic parameters $\text{crit}_{\text{stop}}$

and $\rho_k$ for the stopping criteria for the outer and inner iterations respectively and a generic linear functional $\ell$ for scaling. We tried various choices of $\text{crit}_{\text{stop}}$ in step 22 of Algorithm 1, some of them using $|\varepsilon_k^{(0)} - \varepsilon_{k-2}^{(2)}|/|\varepsilon_k^{(0)}|$, and all led to the same conclusion on the relative merits of the methods. Results reported in Section 3 used $\text{crit}_{\text{stop}} = \text{res}$, where "res" is defined in step 20 of Algorithm 1. We tested three choices of $\rho_k$ which, following Lai et al. [16], we labelled R1-INVIT, R2-INVIT and R3-INVIT. We tested four different linear functionals $\ell$. Best results were obtained with

$$\ell(v_{k+1}) = v_{k+1}^* u_k / u_k^* u_k \,, \quad k \geq 0 \,, \tag{7}$$

which is consistent with (4), and which was used to obtain the results reported here. Detailed results are reported elsewhere [24].

The algorithm for simple inexact inverse iteration, is the same as Algorithm 1, except that steps 5 and 14 are omitted, and, for all $k$, $\sigma_k$ is replaced by the constant $\sigma$. We compared these algorithms with the corresponding algorithms *without* the $\varepsilon$-algorithm, that is with steps 8 and 11–13 omitted and $\varepsilon_k^{(0)}$ replaced by $\varepsilon_0^{(k)}$ in step 19.

# 3 Numerical results

For step 2 and step 7 (the inner loop), we used the BI-CGSTAB algorithm [23], with different preconditioners for our two examples. We computed the eigenvalue of smallest magnitude, and the corresponding eigenvector, using $\sigma = 0$. It is quite likely that, with a choice of $\sigma$ closer to $\lambda_1$, the RQI (but not the $\varepsilon$-algorithm) would have produced a smaller gain than reported here. We generated $u_0$ randomly using the `rand` command of MATLAB.

For all three tested methods used to compute $\rho_{k+1}$, and for both examples, the $\varepsilon$-algorithm generally improved the performance of both simple inverse iteration and RQI, though the improvement produced by the $\varepsilon$-algorithm

**Algorithm 1**: Inexact Rayleigh quotient iteration

**Data**: $A$, $\sigma$, TOL $> 0$, $k_{max}$, a vector $u_0$ (initial approximation) and a linear functional $\ell$. (Typically $\ell(v_{k+1}) = u_k^* v_{k+1} / u_k^* u_k$.)

**Result**: the eigenvalue $\lambda$ of $A$ closest to $\sigma$, and the corresponding eigenvector $u_k$, using SEA and RQI, with accuracy satisfying $\text{crit}_{\text{stop}} <$ TOL.

Initialize $\rho_0 = \rho_1 = 1$, $k = 1$, res $=$ TOL $+ 1$, and $\varepsilon_{-1}^{(0)} = \varepsilon_{-1}^{(1)} = 0$;
Compute $v_1$ such that $\|(A - \sigma I)v_1 - u_0\| \le \rho_0$;
$\varepsilon_0^{(0)} = \ell(v_1)$;
$u_1 = v_1 / \varepsilon_0^{(0)}$;
$\sigma_1 = \sigma$;
**repeat**
    Compute $v_{k+1}$ such that $\|(A - \sigma_k I)v_{k+1} - u_k\| \le \rho_k$;
    $\varepsilon_{-1}^{(k+1)} = 0$;
    $\varepsilon_0^{(k)} = \ell(v_{k+1})$;
    $u_{k+1} = v_{k+1} / \varepsilon_0^{(k)}$;
    **for** $i = 1, \ldots, k$ **do**
        Compute $\varepsilon_i^{(k-i)}$ by the scalar $\varepsilon$-algorithm as given in (6);
    **end**
    $\sigma_{k+1} = u_{k+1}^* A u_{k+1} / u_{k+1}^* u_{k+1}$;
    Compute $\rho_{k+1}$. This is
        $|\varepsilon_0^{(k)} - \varepsilon_0^{(k-1)}| / (k|\varepsilon_0^{(k)}|)$ for R1-INVIT;
        $\|u_{k+1} - u_k\| / (k|\varepsilon_0^{(k)}|)$ for R2-INVIT;
        $\|u_{k+1} - u_k\|$ for R3-INVIT;
    $\lambda = 1/\varepsilon_k^{(0)} + \sigma_{k+1}$;
    res $= \|Au_{k+1} - \lambda u_{k+1}\| / \|u_{k+1}\|$;
    $k = k + 1$;
**until** $k >= k_{max}$ *or* $crit_{stop} <$ TOL ;
**Output** $\lambda$, $u_k$, res and $k$. **Stop.**

without the RQI was generally less than the improvement to simple inverse iteration produced by the RQI without the the $\varepsilon$-algorithm. In general the improvement produced by the $\varepsilon$-algorithm was slightly greater with the method R1-INVIT than with R2-INVIT or R3-INVIT, and R1-INVIT was used to obtain the numerical results reported here.

Our first example is the $n^3 \times n^3$ banded block-Toeplitz matrix SA3D [16]. This matrix arises in the classical finite difference solution, with mesh length $h = 1/(n+1)$, of the three dimensional problem

$$\begin{aligned} -\Delta\phi(x,y,z) + \frac{\partial\phi(x,y,z)}{\partial x} &= \lambda\,\phi(x,y,z) \quad \text{in} \quad \Omega\,, \\ \phi(x,y,z) &= 0 \quad \text{on} \quad \partial\Omega\,, \end{aligned} \tag{8}$$

where $\Omega = (0,1)^3$. (The definition of SA3D by Lai et al. [16] contains a typographical error. The block $C_n$ should be tridiag$[-1-h/2, 6, -1+h/2]$.) Eigenvalues of $C_n$ are found by symmetrizing [25, page 336] [1, Remark 2]. Separation of variables then shows that the eigenvalues of SA3D are

$$6 - 2\cos(q\pi h) - 2\cos(r\pi h) - 2\sqrt{1-(h/2)^2}\cos(s\pi h)\,,$$

$q, r, s = 1, \ldots, n$. Following Lai et al. [16], we take $n = 15$, so that SA3D is $3375 \times 3375$, it has 22275 nonzero elements, and its five smallest eigenvalues are 0.11624635, 0.2300023, 0.2300578, 0.2300578 and 0.3438138.

We present results for Algorithm 1 (Rayleigh quotient iteration and scalar $\varepsilon$-algorithm) and two variants, all using a diagonal preconditioner:

- RQI-SEA (Algorithm 1)
- RQI-NOSEA (Rayleigh quotient iteration but without the SEA)
- NORQI-NOSEA (Simple inexact inverse iteration without the SEA)

TABLE 1: Number of inner (and outer) iterations for SA3D with diagonal preconditioner

| TOL | $10^{-4}$ | $10^{-6}$ | $10^{-8}$ | $10^{-10}$ | $10^{-12}$ |
|---|---|---|---|---|---|
| RQI-SEA | 48 (3) | 63 (4) | 63 (4) | 91 (5) | 91 (5) |
| RQI-NOSEA | 48 (3) | 63 (4) | 91 (5) | 91 (5) | 140 (6) |
| NORQI-NOSEA | 89 (8) | 153 (15) | 218 (22) | 275 (28) | 350 (35) |

TABLE 2: Number of inner (and outer) iterations for JPWH991 with SSOR preconditioner

| TOL | $10^{-4}$ | $10^{-6}$ | $10^{-8}$ | $10^{-10}$ | $10^{-12}$ |
|---|---|---|---|---|---|
| RQI-SEA | 19 (3) | 32 (4) | 54 (5) | 54 (5) | 54 (5) |
| RQI-NOSEA | 32 (4) | 54 (5) | 54 (5) | 54 (5) | 102 (6) |
| NORQI-NOSEA | 34 (5) | 87 (9) | 143 (12) | 239 (16) | 326 (19) |

Table 1 shows the number of inner iterations (followed in brackets by the number of outer iterations) needed for various values of the accuracy TOL.

Our next example is the matrix JPWH 991 from the *Harwell–Boeing Sparse Matrix Collection*. This $991 \times 991$ matrix is derived from a circuit physics model [10]. It can be downloaded from the Matrix Market web site [17], which also gives a structure diagram and other information. Figure 1 shows the distribution of all its eigenvalues. The seven eigenvalues of smallest magnitude are $-0.1206708$, $-0.43112$, $-0.435934$, $-0.453105$, $-0.497937$, $-0.4998651$ and $-0.686086$.

For this example we used an SSOR preconditioner [3]. We present results gained with a fixed relaxation parameter, $\omega = 0.8$. We also examined variable relaxation parameters [24]. Table 2 gives the number of inner iterations (and outer iterations in brackets) used by three methods for various values of the accuracy TOL.

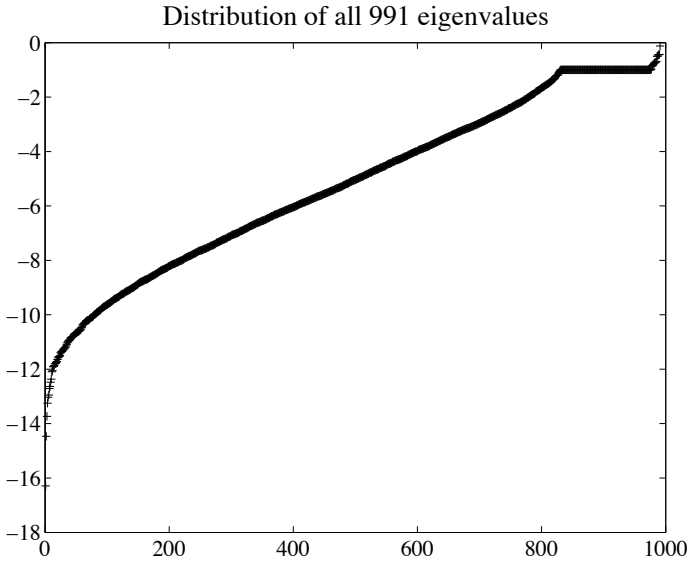


FIGURE 1: Eigenvalues of JPWH 991.

# 4 Concluding remarks

Although our numerical tests considered only the numerical examples of Lai et al. [16], we have no reason to believe that our results are any better than can be expected for most examples. Indeed, in both our examples, $|\lambda_2|$, $|\lambda_3|$ and $|\lambda_4|$ are very close to each other, but not close to $|\lambda_1|$, thus making $\mathtt{d}_2$ and $\mathtt{d}_3$ in (5) very close to $\mathtt{d}_1$, and reducing the effectiveness of eliminating the most slowly decaying terms. This suggests that the $\varepsilon$-algorithm may be even more effective when there are no such tight clusters of eigenvalues. Extrapolation methods must always be implemented with caution, but this still applies when exact methods are used. In spite of the extreme sensitivity of extrapolation methods to errors, our results indicate that any reduction in the usefulness of extrapolation with inexact methods is minor, and certainly does not offset the known advantages of inexact methods. Of course, any general recommendation on extrapolation must be justified by theoret-

ical analysis. This will require a bound on the truncation error caused by the early termination of the inner iterations. Since this error depends on the method used in steps 2 and 7 of Algorithm 1, different methods may require separate analysis. However, our results are sufficiently encouraging to suggest such an analysis would be worthwhile. Another question of interest is whether changes in the method of implementing the $\varepsilon$-algorithm produce improvement. For example, since the later $u_k$ are calculated more accurately than the earlier ones, is it better to apply the $\varepsilon$-algorithm to the sequence $\{u_k\}_{k=K}^{k=\infty}$ for some $K > 0$, rather than to the entire sequence $\{u_k\}_{k=0}^{k=\infty}$, as in our calculations? If so, are there simple criteria for finding the optimal $K$? Some of us are continuing work on the questions raised here and would be interested in hearing from others who may be doing the same.

**Acknowldegment** Part of this work was supported by NUS academic research grants R-146-000-087-112 and R-146-000-064-112.

# References

[1] A. L. Andrew. The accuracy of Numerov's method for eigenvalues. *BIT*, **26**:251–253 1986. doi:10.1007/BF01933751 C244

[2] A. L. Andrew and R. C. E. Tan. Iterative computation of derivatives of repeated eigenvalues and the corresponding eigenvectors. *Numer. Linear Algebra Appl.*, **7**:151–167, 2000. MR2001g:65036 doi:10.1002/1099-1506(200005)7:4¡151::AID-NLA191¿3.0.CO;2-M C240, C241

[3] O. Axelsson. A survey of preconditioned iterative methods for linear systems of algebraic equations. *BIT*, **25**:166–187, 1985. doi:10.1007/BF01934996 C245

[4] Z.-J. Bai, R. H. Chan and B. Morini. An inexact Cayley transform method for inverse eigenvalue problems. *Inverse Problems*, **20**:1675–1689, 2004. doi:10.1088/0266-5611/20/5/022 C239

[5] J. Berns-Müller, I. G. Graham and A. Spence. Inexact inverse iteration for symmetric matrices. *Linear Algebra Appl.*, **416**:389–413, 2006. doi:10.1016/j.laa.2005.11.019 C240

[6] A. Bouras and V. Frayssé. Inexact matrix-vector products in Krylov methods for solving linear systems: a relaxation strategy. *SIAM J. Matrix Anal. Appl.*, **26**:660–678 2005. doi:10.1137/S0895479801384743 C239

[7] C. Brezinski. Computation of the eigenelements of a matrix by the ε-algorithm. *Linear Algebra Appl.*, **11**:7–20, 1975. doi:10.1016/0024-3795(75)90113-5 C241

[8] C. Brezinski and M. Redivo Zaglia. *Extrapolation Methods: Theory and Practice*. North Holland: Amsterdam, 1991. C240, C241

[9] P. Deuflhard. *Newton Methods for Nonlinear Problems. Affine invariance and adaptive algorithms*. Springer-Verlag: Berlin, 2004. C239

[10] A. M. Erisman, R. G. Grimes, J. G. Lewis, W. G. Poole, Jr. and H. D. Simon. Evaluation of orderings for unsymmetric sparse matrices, *SIAM J. Sci. Comput.*, **8**:600-624, 1987. doi:10.1137/0908054 C245

[11] M. A. Freitag and A. Spence. Convergence theory for inexact inverse iteration applied to the generalised nonsymmetric eigenproblem, *ETNA* **28**:40–64, 2007. `http://etna.mcs.kent.edu/` C239, C240

[12] M. A. Freitag and A. Spence. Rayleigh quotient iteration and simplified Jacobi–Davidson method with preconditioned iterative solves, *Linear Algebra Appl.*, **428**:2049–2060, 2008. doi:10.1016/j.laa.2007.11.013 C240

[13] M. A. Freitag and A. Spence. A tuned preconditioner for inexact inverse iteration applied to Hermitian eigenvalue problems, *IMA J. Numer. Anal.*, **28**:522–551, 2008. doi:10.1093/imanum/drm036 C239, C240

[14] G. H. Golub and Q. Ye. Inexact inverse iteration for generalized eigenvalue problems, *BIT*, **40**:671–684, 2000. doi:10.1023/A:1022388317839 C239, C240

[15] I. C. F. Ipsen. Computing an eigenvector with inverse iteration, *SIAM Review*, **39**:254–291, 1997. doi:10.1137/S0036144596300773 C238

[16] Y. L. Lai, K. Y. Lin and W. W. Lin. An inexact inverse iteration for large sparse eigenvalue problems, *Numer. Linear Algebra Appl.*, **4**:425-437, 1997. MR98f:65042 doi:10.1002/(SICI)1099-1506(199709/10)4:5¡425::AID-NLA117¿3.0.CO;2-G C239, C240, C241, C242, C244, C246

[17] Matrix Market. [Online] `http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/cirphys/jpwh_991.html` C245

[18] S. G. Nash. A survey of truncated-Newton methods. *J. Comput. Appl. Math.* **124**:45–59 2000. doi:10.1016/S0377-0427(00)00426-X C239

[19] Y. Notay. Convergence analysis of inexact Rayleigh quotient iteration. *SIAM J. Matrix Anal. Appl.* **24**:627–644, 2003. doi:10.1137/S0895479801399596 C240

[20] V. Simoncini. Variable accuracy of matrix-vector products in projection methods for eigencomputation. *SIAM J. Numer. Anal.*, **43**:1155–1174, 2005. doi:10.1137/040605333 C239

[21] V. Simoncini and L. Eldén. Inexact Rayleigh quotient-type methods for eigenvalue computations. *BIT*, **42**:159–182, 2002. doi:10.1023/A:1021930421106 C240

[22] P. Smit and M. H. C. Paardekooper. The effects of inexact solvers in algorithms for symmetric eigenvalue problems, *Linear Algebra Appl.* **287**:337–357, 1999. doi:10.1016/S0024-3795(98)10201-X C239, C240

[23] H. A. Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Comput.* **13**:631-644, 1992. doi:10.1137/0913035 C242

[24] M. Wächter. *Survey on inexact inverse iteration for eigenvalue problems* Technical Report: Centre for Industrial Mathematics, National University of Singapore, 2005. C242, C245

[25] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press: Oxford, 1965. C238, C239, C244

[26] P. Wynn. On a device for computing the $e_m(S_n)$ transfromation. *Math. Tables Aids Comput.*, **10**:91–96, 1956. MR18,801e C240

## Author addresses

1. **Alan L. Andrew**, Mathematics Department, La Trobe University, Victoria 3086, Australia.
mailto:a.andrew@latrobe.edu.au

2. **P. H. Foo**, Mathematics Department, National University of Singapore, Singapore 117546

3. **Roger C. E. Tan**, Mathematics Department, National University of Singapore, Singapore 117546.
mailto:scitance@nus.edu.sg

4. **Markus Wächter**, German Institute of Science and Technology, Singapore 609916