Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 2647 – 2650

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

On Mixed Precision Iterative Refinement for Eigenvalue Problems

Karl E. Prikopa[a], Wilfried N. Gansterer[a,*]

[a] *University of Vienna, Research Group Theory and Applications of Algorithms, Austria*

Abstract

We investigate novel iterative refinement methods for solving eigenvalue problems which are derived from Newton's method. In particular, approaches for the solution of the resulting linear system based on saddle point problems are compared and evaluated. The algorithms presented exploit the performance benefits of mixed precision, where the majority of operations are performed at a lower working precision and only critical steps within the algorithm are computed in a higher target precision, leading to a solution which is accurate to the target precision. A complexity analysis shows that the best novel method presented requires fewer floating point operations than the so far only existing iterative refinement eigensolver by Dongarra, Moler and Wilkinson.

Keywords: iterative refinement; eigenvalue problems; mixed precision; saddle point/equilibrium problems

1. Introduction

We investigate novel iterative refinement methods for solving the eigenvalue problem $\mathbf{Ax} = \lambda\mathbf{x}$ with symmetric $\mathbf{A} \in \mathbb{R}^{n\times n}$ based on Newton's method for solving non-linear equations. Iterative refinement for linear systems [1] is a strategy for improving the accuracy of a computed solution by trying to reduce round-off errors. The method iteratively computes a correction term to an approximate solution by solving a linear system using the residual of the result as the right-hand side. *Mixed precision* iterative refinement for solving linear systems of equations [2] is a special performance-oriented case of iterative refinement where the majority of operations is performed in single precision while still achieving double precision accuracy.

This paper introduces a mixed precision iterative refinement method for eigenvalue problems by returning to the origin of iterative refinement, Newton's method, which leads to linear systems changing in each iteration for each eigenpair. The complexity of solving the resulting linear systems for all n eigenpairs directly would be $O(n^4)$, which is too high for an efficient eigenvalue iterative refinement method. Therefore we exploit solvers for equilibrium problems to reduce the complexity and find an efficient solution for the linear systems.

2. Related Work

Dongarra, Moler and Wilkinson [3] described an iterative refinement method for improving the numerical accuracy of eigenvalues and eigenvectors, based on the same concept as iterative refinement for linear systems

*Corresponding author. Tel.: +43-1-4277-78311
E-mail address: wilfried.gansterer@univie.ac.at.

Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.06.002

[1]. Their eigenvalue iterative refinement [3, 4] improves eigenvalues and either improves or computes the corresponding eigenvectors. The method is divided into two parts [4]: the *pre-SICE* phase and the *SICE* phase. In the *pre-SICE* phase the matrix is factored in single precision using the Schur decomposition $\mathbf{A} = \mathbf{QUQ}^*$, where $\mathbf{U}$ is an upper triangular matrix and $\mathbf{Q}$ is a unitary matrix. The *SICE* phase then uses the results from the Schur decomposition in combination with triangularizations using plane rotations to improve the approximate eigenvalues by iteratively solving a linear system for a correction term using the residual $\mathbf{r} = \lambda\mathbf{x} - \mathbf{Ax}$ as the right hand-side.

3. Computational Cost of Existing Eigenvalue Iterative Refinement

In the LAPACK User's Guide [5], the LAPACK eigenvalue solver for symmetric matrices `xSYEVD` is described to have a floating point operation count of $9n^3$ for computing the eigenvalues and the right eigenvectors and $1.33n^3$ operations for computing the eigenvalues only. As described in [4], the *pre-SICE* phase requires $10n^3 + 30n^2$ fused-multiply add (FMA) operations, which corresponds well with the flop count of the LAPACK function for computing the eigenvalues of a general matrix. The *SICE* phase requires $13n^2$ FMA operations per iteration. The author states that on average 3 iterations are needed to improve an eigenpair. Our experiments have shown that while this is correct for small matrices with $n = 10$, the number of iterations required increases with the matrix size, for example, a matrix with $n = 1000$ requires on average 4.77 iterations to reach convergence.

The method described in [3, 4] computes the majority of operations, the Schur decomposition and solving the linear systems, in single precision (SP) and only a few operations, computing the residual and updating the result, use double precision (DP) to achieve the target accuracy of the eigenpairs. To compute all n eigenpairs to DP accuracy, the total number of operations is $(10 + 13k)n^3$ FMA operations with k being the average number of iterations and kn^3 operations being executed using DP to compute the residual. Estimating the performance difference between SP and DP to be a factor of 2, the algorithm would perform $(5 + 7k)n^3$ DP operations. Thus, for $k = 5$ the flop count is higher than for the LAPACK eigenvalue solver for general matrices `xGEEV` [5], which requires $26.33n^3$ FLOPs to compute the eigenvalues and eigenvectors.

4. Newton's Method for Iterative Refinement Eigensolver

Iterative refinement is based on Newton's method for solving non-linear equations, which finds a root of a function $f(x)$ using the iterative process $x_{k+1} = x_k - f(x_k)/f'(x_k)$. In higher dimensions, $f'(\mathbf{x})$ is the Jacobian matrix $\mathbf{J}_f(\mathbf{x})$ which results in a linear system of equations. In iterative refinement, the function $f(\mathbf{x})$ is the residual of the solution which is being improved. The residual of eigenvalue problems can be expressed for each eigenpair as $\mathbf{Ax} - \lambda\mathbf{x}$. In [6], the function f is expanded by the additional condition $\mathbf{x}^\top\mathbf{x} - 1$ to normalize the eigenvector $\mathbf{x}$. For an eigenpair the function is defined as $f(\mathbf{x}, \lambda) = \begin{pmatrix} \mathbf{Ax} - \lambda\mathbf{x} & \mathbf{x}^\top\mathbf{x} - 1 \end{pmatrix}^\top$. $f(\mathbf{x}, \lambda) = 0$ if and only if $\mathbf{Ax} = \lambda\mathbf{x}$ and $\mathbf{x}^T\mathbf{x} = 1$, which requires $\mathbf{x}$ to be normalized. An alternative approach is $f(\mathbf{x}, \lambda) = \begin{pmatrix} \mathbf{Ax} - \lambda\mathbf{x} & -\frac{\mathbf{x}^\top\mathbf{x}-1}{2} \end{pmatrix}^\top$. Introducing the factor -0.5 leads to a symmetric Jacobian matrix $\mathbf{J}_f(\mathbf{x}, \lambda) = \begin{pmatrix} \mathbf{A} - \lambda\mathbf{I} & -\mathbf{x} \\ -\mathbf{x}^\top & 0 \end{pmatrix}$. This structure can be exploited by special system solvers, as will be shown in Section 5. The correction term $\Delta(\mathbf{x}_k, \lambda_k) = \begin{pmatrix} \Delta\mathbf{x}_k & \Delta\lambda_k \end{pmatrix}^\top$ consists of a correction for the eigenvector and for the eigenvalue and is found by solving the linear system $\mathbf{J}_f(\mathbf{x}_k, \lambda_k)\,\Delta(\mathbf{x}_k, \lambda_k) = f(\mathbf{x}_k, \lambda_k)$. Then, the approximate solution from the previous iteration is updated according to $(\mathbf{x}_{k+1}\ \lambda_{k+1})^\top = (\mathbf{x}_k\ \lambda_k)^\top + \Delta(\mathbf{x}_k, \lambda_k)$.

The eigenvalue iterative refinement takes an approximate eigenvalue and a random vector as its input and each eigenpair is refined separately. Based on experimental observations, compared to [6], the rate of convergence can be improved significantly by normalising the eigenvector $\mathbf{x}_k$ before constructing and solving the linear system. For each eigenpair a new linear system has to be solved in each iteration because the improved eigenvalue and eigenvector appear in the function $f(\mathbf{x}, \lambda)$ and in its Jacobian. Therefore the system of equations changes in each iteration. If the linear system was factorized, this would lead to a complexity of $O(n^3)$ for improving a single eigenpair, resulting in a complexity of $O(n^4)$ for improving all eigenpairs multiplied by the number of iterations required to reach a target accuracy. This is too high for an efficient eigenvalue iterative refinement method. The Jacobian matrices are saddle point matrices [7, 8, 9] and corresponding solvers can be used to reduce the complexity.

5. Saddle Point Problems

The special structure of saddle point matrices can be exploited when solving such systems. There are direct and iterative methods for solving equilibrium problems. A direct method is the *range space method* [9], which assumes a symmetric saddle point matrix. Applying the range space method to iterative refinement solves the eigenvalue problem. This does not yet reduce the complexity of the iterative refinement method because multiple linear systems have to be solved for the range space method. The range space method only requires a solver for the shifted linear system $\mathbf{A} - \lambda\mathbf{I}$.

The initial approximation for the eigenvalues can be computed using the *Schur factorization* $\mathbf{A} = \mathbf{QUQ}^\top$ in single precision. The Schur factors can then be used to solve the linear systems by applying the shift to $\mathbf{U}$ and inverting $\mathbf{U} - \lambda\mathbf{I}$. This removes the necessity of a decomposition in each iteration and reduces the complexity for improving an eigenpair to $O(n^2)$. As we consider $\mathbf{A}$ being symmetric in this paper, the Schur factor $\mathbf{U}$ is a diagonal matrix and the shifted system can be inverted at a very low cost of n operations. The orthogonal matrix $\mathbf{Q}$ from the Schur factorization can be used as the initial approximation of the eigenvectors, but random values can also be used instead although it will increase the number of iterations required until convergence. This behaviour will be shown in Section 6. When computing $\mathbf{U} - \lambda\mathbf{I}$, the approximate eigenvalue is subtracted from the diagonal of $\mathbf{U}$. This results in $\mathbf{U}$ becoming singular, causing the inversion to fail. A small correction δ has to be introduced when subtracting the eigenvalue from the diagonal of $\mathbf{U}$ to ensure non-singularity.

Acquiring the initial approximate eigenvalues through the Schur decomposition requires $9n^3$ SP operations [10]. For each eigenpair, computing the residual costs n^2 DP operations. The range space method requires the solution of three linear systems in SP using the already available Schur factors, each solution therefore consisting of two matrix-vector operations ($2n^2$) and a back-substitution ($n^2/2$) to invert $\mathbf{U} - \lambda\mathbf{I}$. The total number of operations is $8.5n^2$ per each iteration for each eigenpair. Taking into account the mixed precision computation with a speed-up factor 2 for SP, the total number of operations would be further reduced to $(4.5 + 4.75k)n^3$ DP operations. Due to $\mathbf{A}$ being symmetric, the Schur factor $\mathbf{U}$ becomes a diagonal matrix and a matrix-vector operation is reduced to n multiplications, leading to $(9 + 7k)n^3$ operations and considering SP to $(4.5 + 4k)n^3$ DP operations. Thus, the new method has a lower complexity than the algorithm in [4] with $(5 + 7k)n^3$ DP operations as shown in Section 2.

Another factorization of a symmetric matrix A is the *Householder tridiagonalization* $\mathbf{A} = \mathbf{QTQ}^\top$, with $\mathbf{T}$ being tridiagonal and $\mathbf{Q}$ the product of the Householder transformations. As described previously for the Schur decomposition, the shifted linear systems can be solved analogously by applying the shift to $\mathbf{T}$, again resulting in the reduced complexity of $O(n^2)$ for each improved eigenpair.

The approximate eigenvalues are obtained computing the Householder tridiagonalization in $4n^3/3$ SP operations followed by the Pan-Walker-Kahan QR algorithm with a complexity of $O(n^2)$ [10]. The product Q of the Householder transformations is needed explicitly for solving the shifted linear systems, which requires $4n^3/3$ SP operations. The three linear systems solved for the range space method consist of two matrix-vector operations ($2n^2$) and a bidiagonalization to solve the tridiagonal shifted system $\mathbf{T} - \lambda\mathbf{I}$ with a complexity of $O(n)$. This totals to $8n^3/3 + 7kn^3$ operations. Applying mixed precision, only the computation of the residual requires DP with n^2 operations, reducing the total number of DP operations to $(4/3 + 4k)n^3$. If the number of iterations k was to be the same for all discussed approaches, the Householder approach would lead to the lowest number of operations.

6. Experiments and Initial Results

The experiments compare the eigenvalue iterative refinement by Dongarra, Moler and Wilkinson [4] (SICEDR), and the saddle point approaches with Schur factorization (SPSIR) and Householder tridiagonalization (SPHIR). Almost all experiments were conducted using Matlab 2010a with the exception of SICEDR which was implemented in C. The initial experiments summarized in this section focus on the number of iterations required for convergence and the accuracy of the results, which is compared based on the relative residual. The iterative process terminates if the correction term $\|\Delta(\mathbf{x}, \lambda)\|_\infty$ is less than a defined threshold ϵ and a predefined maximum number of iterations is set as an additional termination criterion. For our experiments random, symmetric matrices are used, $\epsilon = 10^{-12}$ and the maximum number of iterations is set to 20.

Figure 1 shows the average number of iterations per eigenpair for different methods. The comparison cannot be limited to the number of iterations but has to also include the total floating point operations. The DP operations

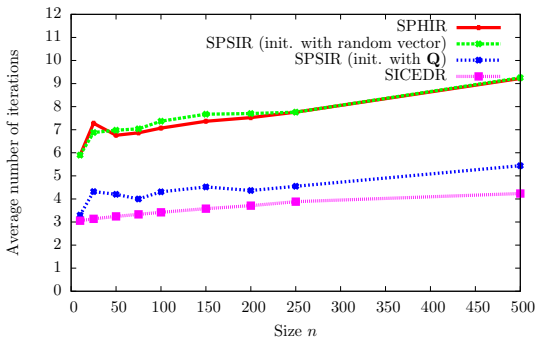

Fig. 1. Average number of iterations for different system sizes n

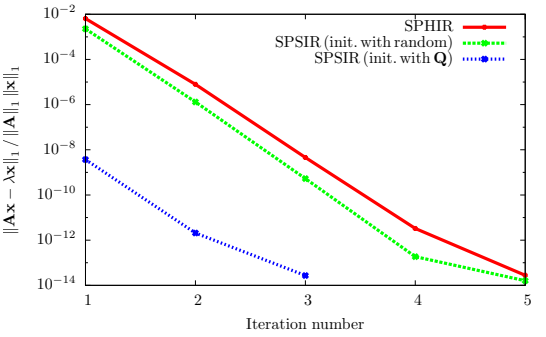

Fig. 2. Comparison of the convergence history for the eigenpair $(\lambda_0, \mathbf{x}_0)$ of a symmetric matrix with $n = 1000$

for the mixed precision methods for $n = 500$ based on the operations count described in the previous sections are: SICEDR with $\bar{k} = 4.23$ requires $34.61n^3$ FLOPs, SPSIR with $\bar{k} = 5.44$ requires $30.34n^3$ FLOPs and SPHIR with $\bar{k} = 9.22$ requires $38.21n^3$ FLOPs. Note that the average iteration count $\bar{k}$ for SPSIR initialised with $\mathbf{Q}$ is slightly higher compared to SICEDR, but the total operations count is lower.

In Figure 2, the convergence history for the first eigenpair $\lambda_0, \mathbf{x}_0$ of a matrix with $n = 1000$ is shown for all methods in terms of the relative residual. SPSIR initialized with the Schur factor $\mathbf{Q}$ only requires 3 iterations. A random vector as the starting vector leads to a higher iteration count, achieving the target precision in 5 iterations. SPHIR does not have an approximation for the eigenvectors and therefore starts with a random vector, converging in 5 iterations.

7. Conclusion

New approaches for mixed precision eigenvalue iterative refinement have been presented based on the solution of saddle point problems. The range space method is used in combination with the Schur factorization and Householder tridiagonalization to solve the resulting non-constant linear systems. It has been shown that the number of floating point operations is lower than the previously described iterative refinement method by Dongarra, Moler and Wilkinson [4], even though the number of iterations is higher. Future research will have to analyse the convergence, the numerical accuracy and the performance of the presented methods. The behaviour for non-symmetric system matrices and other saddle point solvers are future topics of interest.

Acknowledgements

This work has been partly supported by the Austrian Science Fund (FWF) in project S10608 (NFN SISE).

References

[1] J. Wilkinson, Rounding Errors in Algebraic Processes, Her Majesty's Stationery Office, London, 1963.
[2] A. Buttari, J. Dongarra, J. Kurzak, P. Luszczek, S. Tomov, Using mixed precision for sparse matrix computations to enhance the performance while achieving 64-bit accuracy, ACM Trans. Math. Softw. 34 (4) (2008) 17:1–17:22.
[3] J. J. Dongarra, C. B. Moler, J. H. Wilkinson, Improving the Accuracy of Computed Eigenvalues and Eigenvectors, SIAM Journal on Numerical Analysis 20 (1) (1983) 23.
[4] J. Dongarra, Algorithm 589: SICEDR: A FORTRAN subroutine for improving the accuracy of computed matrix eigenvalues, ACM Transactions on Mathematical Software (TOMS) 8 (4) (1982) 371–375.
[5] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. J. Dongarra, J. Du Croz, S. Hammarling, A. Greenbaum, A. McKenney, D. Sorensen, LAPACK Users' guide (third ed.), Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.
[6] H. Röck, Finding an Eigenvector and Eigenvalue, with Newtons Method for Solving Systems of Nonlinear Equations, Tech. rep., Department of Scientific Computing, University of Salzburg, Salzburg (2003).
[7] S. A. Vavasis, Stable numerical algorithms for equilibrium systems, SIAM J. Matrix Anal. Appl 15 (1992) 1108–1131.
[8] M. Benzi, G. H. Golub, J. Liesen, Numerical solution of saddle point problems, Acta Numerica 14 (2005) 1–137.
[9] W. Gansterer, J. Schneid, C. W. Ueberhuber, Mathematical properties of equilibrium systems, Technical report (2003).
[10] G. Golub, C. Van Loan, Matrix Computations, Johns Hopkins University Press, 1996.