[13] T.Y. Li, Zhonggang Zeng, and Luan Cong. Solving eigenvalue problems of real nonsymmetric matrices with real homotopies. Preprint, Michigan State University, E. Lansing, MI 48824-1027, 1990.

[14] G. Peters and J.H. Wilkinson. Inverse iteration, ill-conditioned equations and Newton's method. *SIAM Review*, 21:339–360, 1979.

[15] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix Eigensystem Routines – EISPACK Guide*, volume 6 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1976.

[16] J. Wilkinson and C. Reinsch. *Handbook for Automatic Computation: Volume II - Linear Algebra*. Springer-Verlag, New York, 1971.

[17] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, 1965.

References

[1] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen. Rank-one modification of the symmetric eigenproblem. *Numer. Math.*, 31:31–48, 1978.

[2] P. A. Businger. Numerically stable deflation of Hessenberg and symmetric tridiagonal matrices. *BIT*, 11:262–270, 1971.

[3] J. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.*, 36:177–195, 1981.

[4] J. Dongarra. Improving the accuracy of computed matrix eigenvalues. Technical Report ANL-80-84, Argonne National Lab., August 1980.

[5] J. Dongarra, C. Moler, and J. Wilkinson. Improving the accuracy of computed eigenvalues and eigenvectors. *SIAM J. Numer. Anal.*, 20:23–45, 1982.

[6] J. Dongarra and D. Sorensen. A fully parallel algorithm for the symmetric eigenvalue problem. *SIAM J. Sci. Statist. Comput.*, 8:s139–s154, 1987.

[7] J. Dongarra, D. Sorensen, K. Connolly, and J. Patterson. Programming methodology and performance issues for advanced computer architectures. *Parallel Computing*, 8:41–58, 1988.

[8] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, second edition, 1989.

[9] G. Golub and J.H. Wilkinson. Ill-conditioned eigensystems and the computation of the Jordan canonical form. *SIAM Review*, 18:578–619, 1976.

[10] G. H. Golub. Some modified matrix eigenvalue problems. *SIAM Rev.*, 15:318–334, 1973.

[11] L.E Jessup. A case against a divide and conquer approach for the nonsymmetric eigenvalue problem. Preprint, Oak Ridge National Laboratory, Oak Ridge, TN 37831, 1991.

[12] K. Knopp. *Theory of Functions*. Dover Publications, New York, NY, 1945.

In fact in this particular case we have

$$\gamma = \lambda$$

and

$$\delta = 1.$$

Clearly:

$$\sigma(H', U') = \sigma(H, U) - \{\lambda\}.$$

Therefore, having "removed" λ from the spectrum we can get further eigenpairs. We note that, just as in the case $U = I$, H' and U' are very cheap to obtain once N has been determined. Indeed, H' differs from the $n-1 \times n-1$ principal submatrix of H in the last column only, whereas U' is the $n-1 \times n-1$ principal submatrix of U. The computation of N requires a matrix-vector multiply.

It is also easy to verify that deflating two consecutive complex conjugate eigenpairs results in real H' and U'.

The condition number of N might raise concern. We have

$$cond_\infty(N) \approx max(|w_i|)^2.$$

It is therefore easy to detect an ill-conditioned N. We propose to handle this situation in the generalized case in the following manner. Let D be a diagonal matrix with its diagonal elements d_i defined by

$$\begin{cases} d_i = 1, & \text{if } \frac{|w_i|}{|w_n|} \leq 1 \\ d_i = -\frac{w_n}{w_i}, & \text{if } \frac{|w_i|}{|w_n|} > 1 \end{cases}$$

then clearly

$$\sigma(DNHM, DNUM) = \sigma(H, U),$$

since D is non-singular. The multiplication by D cancels all the large entries in the last column of N. It is easy to see that (16) and (17) are satisfied when the pre-multiplication is done with DN instead of N. The disadvantage of having to premultiply by D is that after the deflation of two complex conjugate eigenpairs, the resulting H' and U' might still be complex.

A possible stopping criterion for this scheme is the condition

$$\frac{\|Hx_i - \lambda_i U x_i\|}{\|x_i\|} \leq f(n)\|H\|\|U\|\epsilon,$$

where $f(n)$ is a modest function of n. It is easy to see that starting from two complex conjugate initial guesses the algorithm will compute complex conjugate corrections, therefore allowing similar savings to those in the case $U = I$.

10 Deflation in the generalized case

The method of deflation presented in section 3.1 generalizes to this case in the following manner. Let $Hx = \lambda Ux$ (of course, in practice we only have an approximate eigenpair). Then it is true that if $w = Ux$ then $w_n \neq 0$: Indeed, if $w_n = 0$, then it can be shown that w is zero using the fact that H is unreduced. U being non singular implies that x is zero which is not true.

Let

$$\begin{pmatrix} -v \\ 1 \end{pmatrix} = w/w_n,$$

and

$$u = \begin{pmatrix} v \\ 1 \end{pmatrix}.$$

We know that $\sigma(H, U)$ is the same as $\sigma(NHM, NUM)$ for any non-singular M and N. For our purposes, we take

$$M = [e_1, \ldots, e_{n-1}, x]$$

and

$$N = [e_1, \ldots, e_{n-1}, u].$$

Then it is easy to verify that we have

$$NHM = \left(\begin{array}{c|c} H' & 0 \\ \hline 0 \quad \alpha & \gamma \end{array}\right), \quad NUM = \left(\begin{array}{c|c} U' & 0 \\ \hline 0 & \delta \end{array}\right), \tag{16}$$

where H' is upper-Hessenberg, U' is upper-triangular and

$$\frac{\gamma}{\delta} = \lambda. \tag{17}$$

Without loss of generality, we can assume H to be unreduced and U to be non-singular since otherwise the problem reduces to a smaller problem. Then our algorithm generalizes easily. Indeed, set

$$H_0 = H - \alpha e_{k+1} e_k^T,$$

and consider solutions of (or approximations of these)

$$H_0 x = \lambda U x, \tag{15}$$

as initial approximations to the sought eigenpairs. More precisely, solving (15) reduces to solving

$$H_{11} x = \lambda U_{11} x \quad and \quad H_{22} x = \lambda U_{22} x.$$

Let's denote these solutions by $(\tilde{x}_1, \lambda_1), \ldots, (\tilde{x}_n, \lambda_n)$ (we have n solutions because of our assumption that U is non-singular). These can be used to construct initial approximations $(x_1, \lambda_1), \ldots, (x_n, \lambda_n)$ to the eigenpairs of the original generalized eigenproblem in much the same way we did in the case $U = I$. More precisely, we take

$$(x_i, \lambda_i) = (\begin{pmatrix} \tilde{x}_i \\ 0 \end{pmatrix}, \lambda_i),$$

if $\lambda_i \in \sigma(H_{11}, U_{11})$ and

$$(x_i, \lambda_i) = (\begin{pmatrix} 0 \\ \tilde{x}_i \end{pmatrix}, \lambda_i),$$

if $\lambda_i \in \sigma(H_{22}, U_{22})$ (an appropriate number of zeros in each case).

Finally solving the generalized eigenvalue problem (14) above is the same as solving the problem

$$\begin{cases} Hx - \lambda U x = 0 \\ L(x) = 1 \end{cases}$$

where $L(x)$ is a scalar equation, which we take to be a normalizing condition: $e_s^T x = 1$. Then for each initial eigenpair (x_i, λ_i), successive corrections can be computed via

$$\begin{pmatrix} H - \lambda_i U & -U x_i \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} y \\ \mu \end{pmatrix} = \begin{pmatrix} r_i \\ 0 \end{pmatrix}, \quad \lambda_i \leftarrow \lambda_i + \mu \;\; ; \;\; x_i \leftarrow x_i + y$$

where

$$r_i = \lambda_i U x_i - H x_i.$$

The following are some results from the Intel iPSC/860 implementation.

Order	No. of procs.	Levels	Ratio HQR2/D&C
100	1	1	1.15
	2	1	1.9
	4	2	3.3
	8	3	5.1
400	1	1	1.24
	2	1	2.4
	4	2	3.2
	8	3	6.0
	16	4	8.4
600	8	3	7.5
	16	4	13.5
	32	5	23
	64	6	32

We observe here that the speedups realized by our algorithm over the QR algorithm, do not remain linear for a large number of processors. This is due to the fact that our algorithm is much less efficient on small matrices, and we had to work with small matrices when the number of processors became large. For example, with a matrix of order 600 and using 64 processors, matrices of average size 20 had to be solved on each node at level 5.

9 The generalized eigenvalue problem

We show here how the ideas behind our algorithm can be used to solve the generalized eigenvalue problem.

9.1 Basic algorithm

Given an upper-Hessenberg matrix H and an upper-triangular matrix U:

$$H = \left(\begin{array}{c|c} H_{11} & H_{12} \\ \hline 0 \quad \alpha & H_{22} \end{array}\right), \quad U = \left(\begin{array}{c|c} U_{11} & U_{12} \\ \hline 0 & U_{22} \end{array}\right),$$

we want to solve the generalized eigenvalue problem,

$$Hx = \lambda Ux. \tag{14}$$

Order	HQR2	D&C	Ratio HQR2/D&C	Distinct λ
100	1.04	1.12	.93	99
200	9.31	9.18	1.01	196
300	34.1	28.1	1.2	293
400	94.0	65.3	1.4	395
500	196	136	1.4	490
1000	1741	992	1.7	1000

In an implementation on a shared memory machine, the storage allocated to the Jacobian (in serial mode) is multiplied by the number of processors used; this is meant to prevent concurrent write to the same memory locations. The following are some results from the Alliant FX/8 implementation. The Alliant FX/8 is a parallel machine with 8 vector processors.

Order	No. of procs.	Levels	Ratio HQR2/D&C
100	2	1	1.7
	4	2	2.4
	8	3	4.0

The results on the Alliant were in general disappointing. The storage scheme for the Jacobian that we used on that machine seems to have inhibited the compiler's vector optimizations. HQR2, running on a single processor *was* vector-optimized.

The following are some results from the Intel iPSC/2 implementation. The largest size used was dictated by the memory capacity of a single node.

Order	No. of procs.	Levels	Ratio HQR2/D&C
100	1	1	1.22
	2	1	2.2
	4	2	3.7
	8	3	6.0
	16	4	8.2
200	1	1	1.16
	2	1	2.2
	4	2	3.5
	8	3	5.2
	16	4	9.1
300	1	1	1.25
	2	1	2.2
	4	2	3.2
	8	3	6.3
	16	4	9.8
	32	5	13.2
	64	6	21.3

the matrix reside on the same processor. Therefore storage schemes more advantageous for linear system solving, such as wrap mapping of columns or rows, could not be used. The communication between processors for this second implementation is more intensive. Communication is needed when solving the linear systems arising in Newton's iterations as well as for the deflation phase. Also because of the storage scheme, we can expect the processors to become successively idle during the factorization of the Jacobian and the back solve for the correction. However, we have implemented an efficient scheme where the Jacobian is repartitioned by rows before the back solve takes place: the "reshuffling" of the submatrices takes place between processors that became idle after doing their part of the Gaussian elimination.

8 Numerical Results and Performance

In this section we present the results of the implementation of the algorithm on a number of machines. The serial version of the code is available through NETLIB where it's called "nonsymdc".

The same algorithm has been run on the IBM RS/6000-550, the Alliant FX/8, the Intel iPSC/2 and the Intel iPSC/860. We compared our results to those of HQR2 from the EISPACK collection. We have used randomly generated upper-Hessenberg matrices in these tests, with entries uniformly distributed between -1 and 1; deflation was not performed. The storage requirement for our algorithm in a serial implementation is $4n^2 + O(n)$, which is twice the storage requirement for a serial implementation of HQR2 [15].

The following are the results from the IBM RS/6000-550 implementation. The IBM RS/6000-550 is a single processor computer with a RISC-based Architecture. In the last column we have given the number of distinct eigenvalues that was computed by our algorithm (the test matrices had no multiple eigenvalues). Each matrix was divided into two subproblems, i.e., only one zero was introduced on the subdiagonal.

7.2 Distributed memory implementation

The current implementation on distributed memory machines requires that the matrix be stored on each processor. This obviously puts rather severe constraints on the size of problems that can be solved. With this implementation however, communication is needed only during the deflation phase. This implementation is best described through the contribution of a particular processor. Suppose that we have 4 processors at our disposal, $p_0, \ldots, p_3$ and that accordingly the matrix H has been divided into 4 subproblems: $H_0, \ldots, H_3$, that their common size is $n/4$ and that they occur in this order on the diagonal of the matrix. We describe now the contribution of p_2 by steps:

1) Call HQR2 to solve for the eigensystem of the matrix H_2.

2) Refine the output from step 1 to get 1/2 the number of (i.e., $n/4$) eigenpairs of the matrix $H_{\frac{1}{2}}$:

$$H_{\frac{1}{2}} = \left(\begin{array}{c|c} H_2 & B \\ \hline 0 \quad \alpha & H_3 \end{array}\right),$$

where $H_{\frac{1}{2}}$ is a submatrix of H.

3) Refine the output from step 2 to get 1/4 of the number of eigenpairs (i.e., $n/4$) of the matrix H.

As can be readily realized no communication between processors is needed except for checking for eigenpairs to which convergence occurred from more than one initial approximation. For example, the eigenpairs of $H_{\frac{1}{2}}$ are generated on p_2 and p_3, and therefore we need to check for duplicate eigenpairs (on each processor separately which requires no communication, and across both which requires communication).

We are currently developing another implementation where blocks of columns of the matrix are stored on different processors. This storage scheme has been dictated to us by the need to call HQR2 at the lowest level: indeed, the call the serial HQR2 requires that contiguous columns of

7 Parallel Algorithms Details and Performance

It is fairly straightforward to see from section 2 how to obtain a parallel algorithm. We discuss here certain details. The given, generally dense, matrix is first reduced to upper-Hessenberg form using a parallel blocked algorithm. Next comes the partitioning phase or "divide". This phase amounts to constructing a binary tree with each node representing a partition into two subproblems. It has been our practice to partition the matrix into a number of subproblems (at the lowest level) equal to the number of processors available on the target machine. Each of these problems may be spawned independently without fear of data conflicts; the computation at this level (the lowest) consists of calls to the EISPACK routine HQR2. The tree is then traversed in reverse order with Newton's method applied at each node using the results from the children as initial approximations. Note here that the computation at a node does not have to wait for both children to complete in order to start: as a matter of fact, it can start as soon as one child has computed one eigenpair of a subproblem. In order to stress this point, we mention here that this is quite different from the situation in the symmetric case [6], where information from both children is needed before computations can start at the node. However, in practice we have allowed computations to start at a node only after at least one child has completed; the need to check for duplicate eigenvalues and deflate if necessary has imposed further synchronization.

The algorithm has been implemented on computers with a shared memory and computers with distributed memory architectures.

7.1 Shared Memory Implementation

So far we have used SCHEDULE [7] to implement the algorithm on shared memory computers. SCHEDULE is a package of FORTRAN and C subroutines designed to aid in programming explicitly parallel algorithms for numerical calculations. An important part of this package is the provision of a mechanism for dynamically spawning processes even when such a capability is not present within the parallel language extensions provided for a given machine.

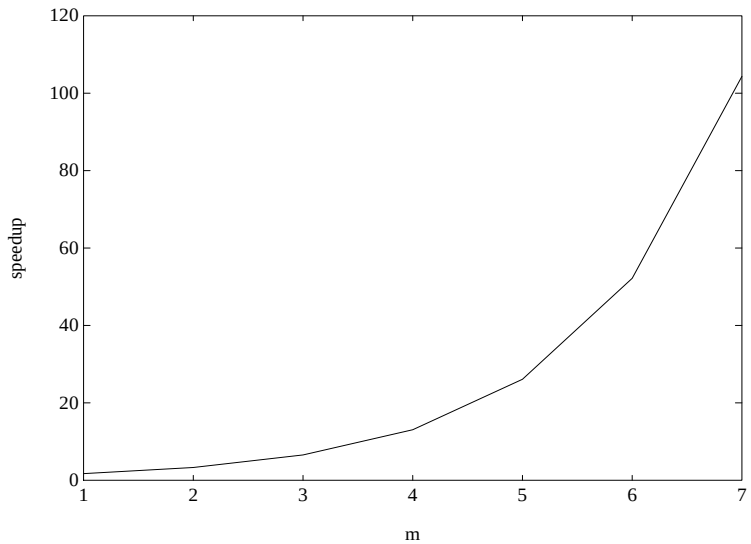

Figure 4: Predicted speedup over QR in terms of number of splits m, with $p = 2^m$ and $k = 3$ (see expression for work).

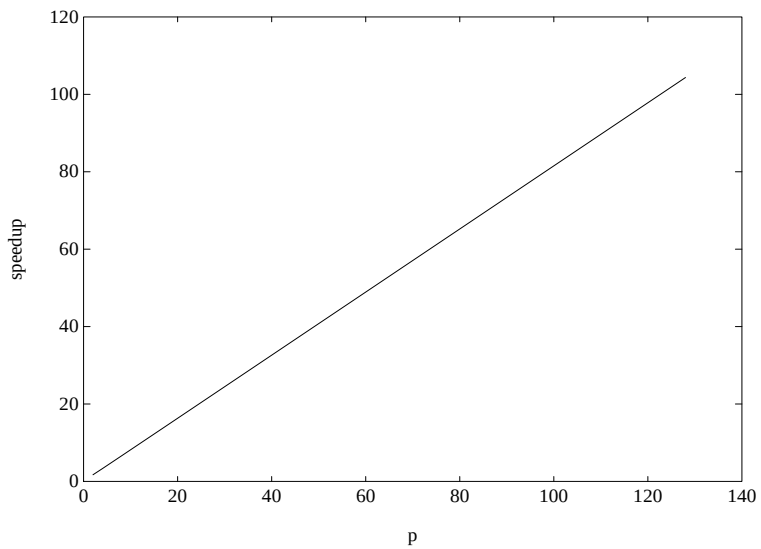

Figure 5: Predicted speedup over QR in terms of the number of processors p, with $m = \log_2 p$ and $k = 3$ (see expression for work).

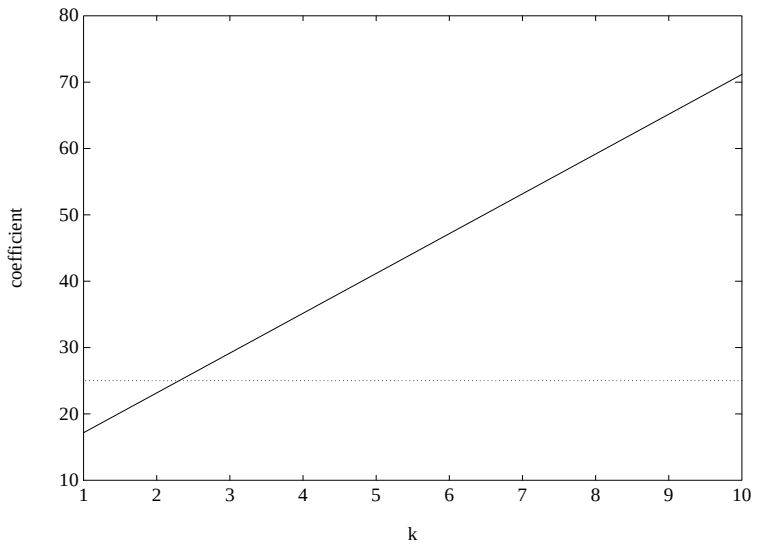

Figure 2: Variation of the Coefficient of n^3 in work estimate model in terms of the number of steps k, with $m = 1$, and $p = 1$ (see expression for work).

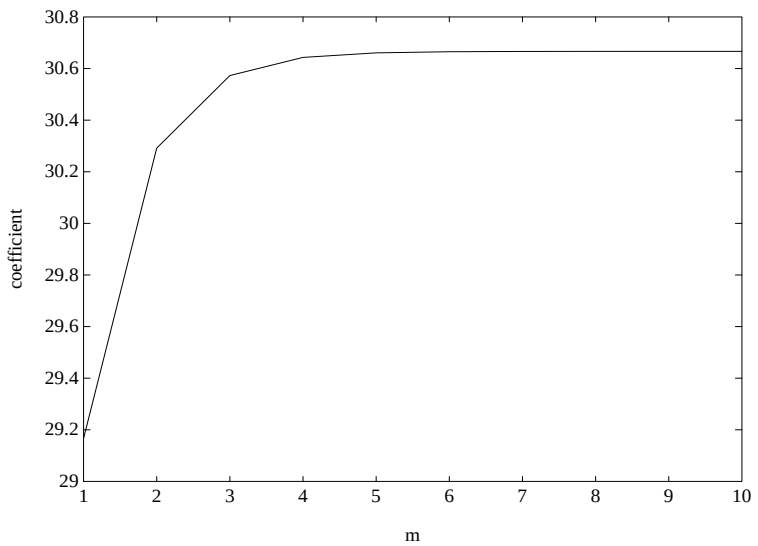

Figure 3: Variation of the Coefficient of n^3 in work estimate model in terms of the number of splits m, with $k = 3$ and $p = 1$ (see expression for work).

But $p' = 2^m$ and therefore $p'^2 = 4^m$ and hence

$$W_p = \frac{n^3}{p}\left(\frac{16}{3} + 8k + (18 - 8k)\frac{1}{4^m}\right), \tag{13}$$

where for ease of reference we redefine the various parameters: n is the order of the matrix; p is the number of processors; k is the average number of Newton iterations needed before an eigenpair is accepted; and m is the number of zeros introduced on the subdiagonal.

The cost of getting the eigenvalues and eigenvectors by the QR algorithm is $15n^3$ [8]. A reasonable value for k is 3, however there are cases when k is 2 or less. There are also cases where k is larger than 3, mostly with matrices of small order or defective matrices.

It is easy to verify that for $k = 2$ and for $m = 1$ (one split) the model for the cost of the algorithm predicts that a sequential implementation of our algorithm is faster than RG. Our model predicts that a sequential implementation of our algorithm is slower than RG for problems where k equals or exceeds 2 (see Fig. 2). Here are some sample values of W_p, assuming that $k = 3$ and $p = p' = 2^m$. which means that the original problem is subdivided into a number of problems equal to the number of available processors.

$$p = 128 \rightarrow W_p = 0.2291n^3 \;\; ; \;\; p = 1024 \rightarrow W_p = 0.0286n^3.$$

Here we have assumed that the problem is large enough to allow the efficient use of that many processors. Note that the ratio of the work estimate from our model to the work estimate of QR is independent of n. This is due to the simplifying assumption on k made at the beginning of this section. That assumption has in effect "hidden" the dependency on n of the coefficient of n^3 in our model. Figures 2, 3, 4 and 5 show plots of the work estimate for various values of the parameters involved.

We note lastly that the cost of our algorithm will increase when the matrix or the subproblems obtained in the divide process are ill-conditioned (see section 5 and [11]). Furthermore, repeated deflations will also contribute to an increase in the cost.

The cost of computing one correction at the level ℓ is roughly $6s_\ell^2$: indeed, one Newton iteration involves the solution of a linear system that is upper-Hessenberg, but for possible non-zeros in the last row; the order of this linear system is $s_\ell + 1$. Therefore 2 multipliers at most need be computed per column and, when updating the matrix, each of these will be used in $2(s_\ell + 1 - i)$ multiplications and additions, where i is the index of the column. The factorization of the Jacobian requires then $2s_\ell^2$ operations in addition to $O(s_\ell)$ operations (including divisions and comparisons). The forward solve is $O(s_\ell)$ work and is negligible. The backsolve requires s_ℓ^2 operations and computing the residual requires another s_ℓ^2 operations. Therefore for a real current approximation one correction comes at the cost of $4s_\ell^2$. For a complex current eigenpair, this becomes $16s_\ell^2$ since the dominant operations are a roughly equal number of multiplications and additions. Since, as we indicated in section 2, only one of a conjugate pair of complex eigenpairs need be corrected, we can assume that $8s_\ell^2$ operations are required for correcting a complex eigenpair. Assuming again that the matrix has an equal number of real and complex eigenvalues, our estimate for the amount of work required for computing one correction at the level ℓ becomes $6s_\ell^2$ operations.

We shall use $\frac{n}{p}$ initial guesses to start $\frac{n}{p}$ Newton processes on each processor. We now have the following work estimate on one processor, assuming all processors share equally the cost of all the stages of the algorithm

$$W_p = \frac{10n^3}{3p} + 18\left(\frac{p'}{p}\right)\left(\frac{n}{p'}\right)^3 + \sum_{\ell=0}^{m-1} 6(\frac{n}{p})ks_\ell^2 + 2\frac{n^3}{p},$$

where: $\frac{10n^3}{3p}$ is the processor's contribution to the reduction of the original matrix to upper-Hessenberg form; $18\left(\frac{p'}{p}\right)\left(\frac{n}{p'}\right)^3$ is the cost of applying the QR algrithm to $\frac{p'}{p}$ matrices of size $\frac{n}{p'}$; $\sum_{\ell=0}^{m-1} 6(\frac{n}{p})ks_l^2$ is the cost of solving k linear systems with matrices of size s_ℓ, repeating this for $\frac{n}{p}$ initial guesses and for $\ell = 0, \cdots, m-1$; $2\frac{n^3}{p}$ is the processor's contribution to the computation of the eigenvectors of the original dense matrix A once those of H have been computed. The expression for the work can be rewritten:

$$W_p = \frac{n^3}{p}\left(\frac{16}{3} + \frac{18}{p'^2} + \frac{24k}{3}(1 - (\frac{1}{4})^m)\right).$$

fact that it is the structure of the matrix that caused the poor approximations and the failures, and not some inherent difficulty with the spectrum of these matrices.

6 Work Estimates

We assume in this section that we are given a real dense matrix A. Then the first task in our algorithm is to reduce A to an upper-Hessenberg matrix H. This requires $\frac{10n^3}{3}$ operations (plus lower order terms) if elementary transformations are used, since these matrices are to be accumulated.

The dividing process is now applied to H. Assume $n = 2^m r$, for some $m \geq 1$ where r is not necessarily relatively prime to 2. If we repeat the dividing process m times, we end up with 2^m matrices of size r, each of which is upper-Hessenberg. A submatrix of size $\frac{n}{2^i}$ obtained in the dividing process will be referred to as a matrix at the level i. The matrix H itself is at the level 0, and the r matrices referred to above are at the level m. Thus, we have $m+1$ levels in total. Let $p' = 2^m$ be the number of submatrices at the lowest level, and p the number of processors; we will assume that $p = 2^q, q \leq m$. The cost of finding the eigenpairs of a (Hessenberg) matrix at the lowest level (by the QR algorithm) is roughly $18\left(\frac{n}{p'}\right)^3$: this figure is very approximate and assumes among other things that two QR steps are needed before a real or two complex conjugate eigenvalues are identified and that the matrix has an equal number of real and complex eigenvalues [8]. Let $s_\ell = \frac{n}{2^\ell}$ be the size of one matrix at the level ℓ. Let k_ℓ be the average number of iterations needed to get one eigenpair of a matrix at the level ℓ. k_ℓ depends on the matrix and on its size of course. We will make the following simplifying assumption however: $k_\ell = k$, $\ell = 0, \cdots, m$, i.e., we assume that the average number of steps required for convergence is the same at all levels. Our experience with the algorithm suggests that this is a realistic assumption as long as the size of the submatrices remains moderate. If the number of levels is increased to the point where we are left with small submatrices (less than 20×20 say), then k_ℓ becomes significantly larger as ℓ increases (for very small matrices it can be more than 10 on the average).

H, namely:

$$(\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, 0), \quad (\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, 0),$$

if the zero is introduced in the $(3,2)$ position.

Furthermore, the Jacobian is exactly singular at each of these initial approximations. Indeed the Jacobian at $(\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, 0)$ is

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

which is clearly of rank three since the first and the last columns are linearly dependent, and the fourth column is zero. Similarly it can be seen that the Jacobian at $(\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, 0)$ is also singular. A remedy to this situation is to perturb the initial guess zero by a small amount thereby making the Jacobian non singular, and indeed this has been successful in practice. The problem in this case, as in most other similar cases, results from the particular structure of the matrix. In order to obtain further eigenvalues, we need to deflate the matrix each time a new eigenpair is computed which makes the algorithm almost serial.

5.3 Known failures

Some matrices of the structure mentioned at the end of section 5.2 (companion-like matrices), provided us with the only cases where the algorithm failed in practice to converge to the desired eigenpairs, i.e., failed to produce eigenpairs with small residuals after a fixed number of iterations. When these matrices were subjected to random orthogonal similarity transformations however and then reduced back to upper-Hessenberg form, the dividing process provided us with much better initial approximations and indeed, the algorithm converged for all initial approximations. We are certainly not advocating this as a general viable scheme: we wanted to emphasize the

unchanged after a zero has been introduced in the subdiagonal. An example, is the 2×2 matrix

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

After the dividing process we have

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

The starting eigenpairs are then $(\begin{pmatrix} 1 \\ 0 \end{pmatrix}, 1)$ and $(\begin{pmatrix} 0 \\ 1 \end{pmatrix}, 1)$. The second of these is of course the exact eigenpair. Now, even though the first starting eigenvector is orthogonal to the desired one, the equations arising during the first of Newton's iterations can be solved in a way to produce the desired eigenvector from the first step: zero pivots need to be replaced by small numbers on the order of the machine unit roundoff (as is done in inverse iteration, INVIT [15]).

Finally, we mention that the deflation process can contribute to the improvement of the condition of the eigenvalues. Indeed, if λ and λ' are pathologically close (and fairly distant from the rest of the eigenvalues), then by deflating λ, λ' will have a better condition number as an eigenvalue of the resulting matrix. Indeed, λ' is no longer part of a cluster.

5.2 Case when H_0 is defective

It can happen in this case (e.g., if H is non-defective) that the initial dividing process would leave us with a number of initial approximations that is smaller than n. This will inhibit the parallelism of the algorithm in that fewer Newton processes can be started simultaneously. Furthermore, whatever eigenpairs we have can be extremely poor approximations to the desired ones. An extreme situation is illustrated by the matrix

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

No matter where the zero is introduced on the subdiagonal, the resulting matrix has 0 as its only eigenvalue, and we only have two initial approximations to the four distinct eigenpairs of

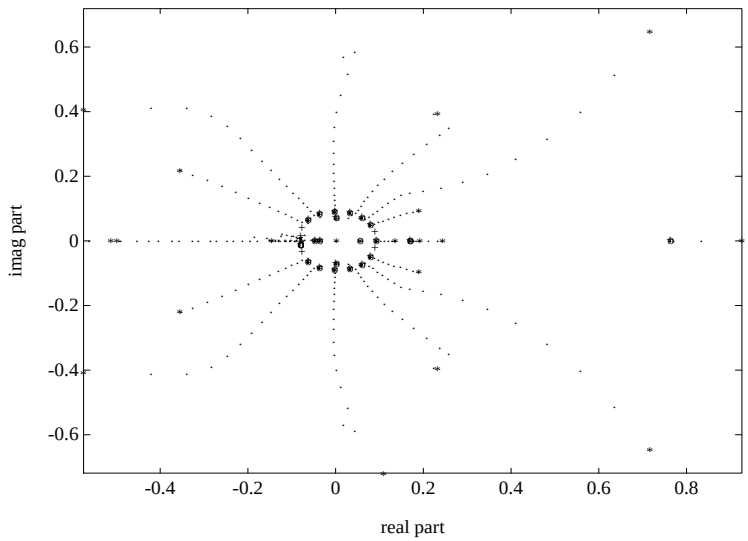

Figure 1: *The behavior of the algorithm for an almost defective 25×25 matrix. The crosses are the eigenvalues of the original matrix; the stars are the initial guesses; the circles are the eigenvalues computed by our algorithm; the dots are the iterates arising in Newton's iterations.*

to λ can be expected to converge to the space generated by $x_1, \ldots, x_\ell$. As the eigenvalue λ approaches the cluster however, continued corrections to the eigenvector cannot be expected to refine it. We refer the reader to the particularly lucid account in [14] for a justification of these claims. Solving as in inverse iteration (see INVIT [15]) is a possible way around this problem. Computing the residual with extended precision arithmetic is also an obvious approach, and has been successful in practice.

When approaching a singular solution, Newton's method loses its quadratic convergence rate. We have proved earlier (Thm. 4.1) that the Jacobian is singular at multiple eigenpairs and therefore we can expect slower convergence when these are the target (see Fig. 1).

Recall the rate of change of λ that we derived in section 2:

$$|\lambda'(0)| = \frac{|\alpha||y_{k+1}||x_k|}{|y^H x|}.$$

We wish to caution from hastily drawing conclusions about the sensitivity of the eigenvalues of a defective matrix to our dividing process from this expression. Indeed, as an extreme case which will help illustrate our point, the eigenvalues of a defective matrix can remain virtually

Therefore, letting $b = \begin{pmatrix} r \\ 0 \end{pmatrix}$, we have,

$$\begin{pmatrix} y \\ \mu \end{pmatrix} = J^{-1}b = J_0^{-1}b - \frac{v^T J_0^{-1} b}{1 + v^T J_0^{-1} e_{n+1}} J_0^{-1} e_{n+1}. \tag{11}$$

It can be easily checked that

$$J_0^{-1}b = \begin{pmatrix} -x \\ 0 \end{pmatrix}$$

and that

$$J_0^{-1}e_{n+1} = \begin{pmatrix} \tilde{x} \\ 1 \end{pmatrix}$$

where $(H - \lambda I)\tilde{x} = x$. Now if we let (x_1, λ_1) be the next eigenpair we have from (11) and the subsequent equalities,

$$\begin{pmatrix} x_1 \\ \lambda_1 \end{pmatrix} = \begin{pmatrix} x \\ \lambda \end{pmatrix} + \begin{pmatrix} y \\ \mu \end{pmatrix} = \begin{pmatrix} x \\ \lambda \end{pmatrix} + \begin{pmatrix} -x \\ 0 \end{pmatrix} - \frac{v^T J_0^{-1} b}{1 + v^T J_0^{-1} e_{n+1}} \begin{pmatrix} \tilde{x} \\ 1 \end{pmatrix}.$$

Furthermore,

$$\frac{v^T J_0^{-1} b}{1 + v^T J_0^{-1} e_{n+1}} = \frac{-1}{\tilde{x}_s}$$

and so finally we see that our scheme reduces to the following: Given (x, λ) compute the next iterate (x_1, λ_1) via

$$\begin{aligned} (H - \lambda I)\tilde{x} &= x \\ x_1 &= \tfrac{1}{\tilde{x}_s}\tilde{x} \\ \lambda_1 &= \lambda + \tfrac{1}{\tilde{x}_s}. \end{aligned} \tag{12}$$

5.1 Case when H is defective

When the algorithm is expressed as in (12) we can readily see some of the difficulties that arise when the matrix H is defective or almost defective, which is more likely in general due to round-off errors. These problems are similar to the kind of problems that face the application of inverse iteration. More precisely, assume that H is almost defective with $x_1, \ldots, x_n$ as a complete set of eigenvectors. Let $\lambda_1, \ldots, \lambda_\ell$ be a cluster of eigenvalues of H and suppose that the initial approximate eigenvalue λ corresponds to one of these. The eigenvectors $x_1, \ldots, x_\ell$ corresponding to these eigenvalues are then almost linearly dependent. In general, the eigenvector corresponding

this section with a particular emphasis on the case when either the matrix H or the modified matrix H_0 is defective, i.e., when either one of these matrices does not have a complete set of eigenvectors. As we remarked earlier, an eigenvalue of H (with no zeros on the subdiagonal) can only have geometric multiplicity one, and therefore H is defective whenever it has a multiple eigenvalue. An eigenvalue of H_0 on the other hand, can have geometric multiplicity one or two. In what follows n is the order of H.

The connection between Newton's method and inverse iteration is well known [14]. We present a proof which motivates the subsequent analysis: we let J be the Jacobian of the map F_s at (x, λ) defined in section 2,

$$J = \begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix},$$

and we assume that $x_s = 1$. The order of the Jacobian is $n + 1$. Then,

$$J = J_0 + e_{n+1} v^T$$

with

$$J_0 = \begin{pmatrix} H - \lambda I & -x \\ 0 & 1 \end{pmatrix}$$

and,

$$v = e_s - e_{n+1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ -1 \end{pmatrix}.$$

Assume J_0 it is not singular; this is true if and only if λ is not an eigenvalue. Assume also that J is not singular. The correction to (x, λ) is computed in the following manner:

$$\begin{pmatrix} y \\ \mu \end{pmatrix} = J^{-1} \begin{pmatrix} r \\ 0 \end{pmatrix},$$

where $r = \lambda x - Hx$. Using the Sherman-Morrison-Woodbury formula [8] we can write J^{-1} as,

$$J^{-1} = (J_0 + e_{n+1} v^T)^{-1} = J_0^{-1} - \frac{1}{1 + v^T J_0^{-1} e_{n+1}} J_0^{-1} e_{n+1} v^T J_0^{-1}.$$

direction since then the eigenvalue would be acting like an eigenvalue of geometric multiplicity more than one, which is impossible since the matrix is unreduced (recall that $x_s = 1$).

The second derivative of F_s is a constant bilinear operator with norm equal to 2. In fact,

$$F_s''(x,\lambda) = \left(\begin{array}{cc|cc} 0 & -\mathrm{I} & -\mathrm{I} & 0 \\ 0 & 0 & 0 & 0 \end{array}\right).$$

Suppose now that our procedure is started with initial guess (x_0, λ_0). We now give sufficient conditions for the convergence of our procedure with the given initial guess. This is provided by the classical Kantorovich theorem [5]. Let us first introduce

$$K = \|F_s'^{-1}(x_0, \lambda_0)\|$$

and

$$c_1 = \|(x_1, \lambda_1) - (x_0, \lambda_0)\|,$$

where (x_1, λ_1) is the first iterate, i.e.,

$$(x_1, \lambda_1)^T = (x_0, \lambda_0)^T - F_s'^{-1}(x_0, \lambda_0) F_s(x_0, \lambda_0).$$

We call (x_0, λ_0), ...,(x_k, λ_k) the sequence of iterates produced by the algorithm.

Theorem 4.3 *If $\beta_0 = Kc_1 < 1/4$, then the sequence (x_k, λ_k) converges quadratically starting from (x_0, λ_0).*

The process can be regarded as starting from any of the iterates (x_i, λ_i) and in fact will often converge even when the conditions of the theorem are not satisfied at (x_0, λ_0). These conditions will then be met for some (x_k, λ_k) at which stage convergence becomes quadratic.

5 Defective Case

Ours being a non-stationary iteration (the iterating map is not fixed), it is not easy to analyze the behavior of the successive approximations. We try however to address this problem in

Theorem 4.2 *Assume (x, λ) is not an exact zero of F_s. Then* $\begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix}$ *is singular if and only if at least one of the following is true:*

1) λ is an eigenvalue of H and has an eigenvector whose s^{th} component is zero.

2) x belongs to the space generated by $(c_1, \ldots, c_{s-1}, c_{s+1}, \ldots, c_n)$, where c_i is the i^{th} column of $H - \lambda I$ (λ may or may not be an eigenvalue).

Proof. Assume first that 1) is true and let y be an eigenvector, $y_s = 0$. Then $\begin{pmatrix} y \\ 0 \end{pmatrix}$ is clearly in the null space of the Jacobian. If 2) holds, and $x = (H - \lambda I)y$, with $y_s = 0$, then $\begin{pmatrix} y \\ 1 \end{pmatrix}$ is in the null space of the Jacobian.

Conversely, if the Jacobian is singular then there exists a vector $\begin{pmatrix} y \\ \mu \end{pmatrix}$ such that,

$$\begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} y \\ \mu \end{pmatrix} = 0.$$

This implies that

$$\begin{cases} (H - \lambda I)y = \mu x \\ e_s^T y = 0 \end{cases}$$

and clearly $y_s = 0$. We now consider two cases according to whether μ is zero or not. If μ is zero, then λ is an eigenvalue with corresponding eigenvector y, therefore we are in case 1). If μ is not zero then μx and hence x is in the range of $H - \lambda I$ and therefore we are in case 2) since y_s=0. Note that λ may or may not be an eigenvalue in this last case. □

Remarks: The theorem tells us that more often than not, a singular Jacobian is an indication that we already have an eigenvalue, and this has been our experience indeed. The singularity of the Jacobian is also an indication of an ill-conditioned eigensystem. In fact, if we accept that the current eigenvector was moving in the "right" direction then if it satisfies condition 2 of the theorem, we can say that the eigenvalue is acting like a multiple one since the eigenvector is also in the range of $(H - \lambda I)$. In practice we have not encountered a situation where condition 1 applied: this is understandable again if we accept that the eigenvector is moving in the right

Theorem 4.1 (Wilkinson) *Assume (x, λ) is an exact zero of F_s. Then*

$$\begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix}$$

is singular if and only if λ is multiple.

Proof. Let us assume first that λ is a multiple eigenvalue of H and that x is the corresponding (exact) eigenvector, then there exists a nonzero vector y such that

$$y^H(H - \lambda I) = 0 \text{ and } y^H x = 0;$$

y is simply a left eigenvector of H corresponding to λ. Thus, $(y^H\ 0)$ is a nonzero vector and

$$(y^H\ 0)\begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix} = 0.$$

This proves that the Jacobian is singular.

Now conversely, if $\begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix}$ is singular then there exists a nonzero vector $\begin{pmatrix} v \\ \mu \end{pmatrix}$ such that

$$\begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix}\begin{pmatrix} v \\ \mu \end{pmatrix} = 0.$$

But then $v_s = 0$ and $(H - \lambda I)v = \mu x$. If $\mu = 0$, then v is nonzero since $\begin{pmatrix} v \\ \mu \end{pmatrix} \neq 0$, and so v is an eigenvector that is linearly independent of x, since $x_s = 1$ and $v_s = 0$. Hence λ is a multiple eigenvalue in this case. If $\mu \neq 0$, then v is also nonzero; if it were then we would have

$$\mu x = (H - \lambda I)v = (H - \lambda I)0 = 0$$

and hence $x = 0$, contradicting our assumption on x. But $(H - \lambda I)^2 v = \mu(H - \lambda I)x = 0$, and thus v is a nonzero vector that has grade 2. Therefore λ is multiple in this case as well. □

Remark: Since we are assuming that H is upper-Hessenberg and unreduced, an eigenvalue can only be non-derogatory, i.e., the associated eigenspace has dimension one.

The previous result applies to the Jacobian at a zero of F_s. We wish to know more about the Jacobian at those approximations arising during Newton's iteration and before convergence is declared.

task: in particular, they will not be detected if their difference is compared to *tol* introduced earlier. Conversely, it can happen that under certain conditions the tolerated size of the residual be much larger than the distance between certain exact eigenvalues and therefore between certain computed eigenvalues. In this case, it could happen that distinct computed eigenvalues will be declared as duplicates if their difference is compared to *tol*.

Example 3.2 *We illustrate this last case with the following matrix*

$$H = \begin{pmatrix} 2 \cdot 10^{-7} & 0 & 0 \\ 2 & 10^{-7} & 0 \\ 0 & 2 & 10^{10} \end{pmatrix}.$$

The tolerated size of the residuals for this matrix as chosen in our algorithm is $tol \approx \|H\|\epsilon > 10^{-6}$. *Therefore if we decide to declare as duplicates those eigenvalues whose difference is less than tol then the first two distinct eigenvalues of* H *will be declared as duplicates.*

The problems we have just raised do not have easy solutions [9], and indeed more research is needed here.

4 Convergence

In section 2, we mentioned that computing an eigenpair of H reduces to computing a zero of

$$F_s(x, \lambda) = \begin{pmatrix} Hx - \lambda x \\ e_s^T x - 1 \end{pmatrix}.$$

The Jacobian of F_s at (x, λ) is

$$D_{(x,\lambda)} F_s(x, \lambda) = \begin{pmatrix} D_x(Hx - \lambda x) & D_\lambda(Hx - \lambda x) \\ D_x(e_s^T - 1) & D_\lambda(e_s^T - 1) \end{pmatrix} = \begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix},$$

where $D_x(F)$ denotes the derivative with respect to x of the function F. In this section, we give sufficient conditions for the convergence of our procedure. The result is a version of the Kantorovich theorem as it applies to our case.

the definition of H'). However, this will in general result in introducing non-zero entries below the subdiagonal of H and therefore H needs to be reduced to upper-Hessenberg form again. We refer the reader to [2] for an example of such an algorithm; the generalization of that algorithm to the case where the eigenvalue to be deflated is complex is straightforward.

In addition to the difficulty just mentioned, the deflation with plane rotations, suffers from the fact that the deflated matrix will be complex if the eigenvalue to be deflated is complex. Indeed, it is unfortunately not true in general that when an eigenvalue and its complex conjugate are deflated by this method, the resulting matrix is real.

Example 3.1 *When the two complex eigenvalues of the* 4×4 *upper-Hessenberg matrix,*

$$\begin{pmatrix} 0.2190 & -0.0756 & 0.6787 & -0.6391 \\ -0.9615 & 0.9032 & -0.4571 & 0.8804 \\ 0 & -0.3822 & 0.4526 & -0.0641 \\ 0 & 0 & -0.1069 & -0.0252 \end{pmatrix}$$

are deflated using plane rotations as just described we obtain

$$\begin{pmatrix} 1.1593 - 0.0000i & 0.0000 + 1.0129i & 0.0000 + 0.0000i & 0.0000 + 0.0000i \\ 0.0000 - 0.3053i & 0.1740 - 0.0000i & 0.0000 + 0.0000i & 0.0000 + 0.0000i \\ -0.1534 - 0.3540i & 0.5717 + 0.0112i & 0.1082 - 0.4681i & 0.0000 + 0.0000i \\ -0.4640 + 0.1988i & 0.2187 - 0.3465i & 0.3964 + 0.0611i & 0.1082 + 0.4681i \end{pmatrix}.$$

After the deflation, we will be working with the upper 2×2 *block of* H' *which clearly is complex.*

3.4 Remarks on identifying duplicate eigenvalues

We have already remarked that the computed eigenpairs from our algorithm are the exact eigenpairs of a nearby matrix. Under those conditions ([8]),

$$|\lambda - \lambda_e| \leq \|E\|/|s(\lambda_e)|,$$

where E is the error in the matrix, and $|s(\lambda_e)|$ is the condition number of the exact eigenvalue λ_e of the original matrix H. When λ_e is ill-conditioned we can expect unpredictably large errors (compared to the tolerated size of the residual) in the computed approximations to λ_e and therefore in the duplicates if any and identifying these duplicates becomes a rather daunting

end of the $n-1$ steps just described the last column is zero. Let us set $\mathcal{G} = G_1 \ldots G_{n-1}$ to simplify the notation. Then,

$$\mathcal{G}^{-1} = \mathcal{G}^H = G_{n-1}^H \ldots G_1^H,$$

and it is straightforward to verify that the zeros of the last column of $(H - \lambda I)\mathcal{G}$ will be preserved when it is pre-multiplied by $\mathcal{G}^{-1}$, because the successive pre-multiplications by $G_r^T, r = 1, \ldots, n-1$, will preserve those zeros. Therefore the last column of

$$\widetilde{H} = \mathcal{G}^H (H - \lambda I)\mathcal{G} + \lambda I$$

is equal to λe_n. The eigenvector of $\widetilde{H}$ corresponding to λ is e_n. Note that $\widetilde{H}$ is not upper-Hessenberg in this case though. Indeed, non zero elements will be introduced in the last row of $\widetilde{H}$; in fact ,

$$\widetilde{H} = \begin{pmatrix} * & \cdots & * & \\ * & & & 0 \\ & \ddots & \vdots & \\ & & * * & \\ * & \cdots & * & \lambda \end{pmatrix}.$$

This is not disturbing since if H' is the leading principal submatrix of $\widetilde{H}$ of order $n-1$, then H' is upper-Hessenberg and $\sigma(H') = \sigma(H) - \{\lambda\}$.

If λ is a multiple eigenvalue of H of (algebraic) multiplicity m then λ is an eigenvalue of H' of multiplicity $m-1$.

So far we have assumed that the eigenpair (x, λ) is exact. In practice, (x, λ) will only be approximate, in the sense that $Hx - \lambda x = r$, is of the order of the machine ϵ. In this case round-off errors will in general prevent G_{n-1} from annihilating the $(1, n)$ entry. In fact, the accuracy of the computed eigenvalue λ will come into play. If λ corresponds to an ill-conditioned eigenvalue of H, then it is possible that λ be a rather poor approximation of the exact eigenvalue. As a consequence the $(1, n)$ entry might not be negligible at all and examples do exist where this is indeed the case [17]. A way around this difficulty is to construct the plane rotations $G_1, \ldots, G_{n-1}$ in a way to reduce the vector x to e_n and then apply the corresponding similarity transformation to H. Inequality (10) from section 3.1 holds when this is done (with obvious modification in

The strategy consists of $n-1$ major steps, at each of which a new zero is introduced in the last column of $H-\lambda I$ starting from the bottom. The configuration at the beginning of the r^{th} step looks like

$$\tilde{H} = \begin{pmatrix} * & \cdots & * & * \\ * & & & \vdots \\ & \ddots & \vdots & * \\ & & * & 0 \end{pmatrix}$$

with zeros in the last $r-1$ components of the last column. The r^{th} step then consists in a post-multiplication by G_r, where G_r is the (possibly complex) rotation in the plane $(n-r,n)$ designed to annihilate the $(n-r,n)$ element:

$$G_r = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & c_r & & & -s_r \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & s_r & & & 1 \\ & & & & & & & c_r \end{pmatrix},$$

where $\|c_r\|^2 + \|s_r\|^2 = 1$. This post-multiplication will affect columns $n-r$ and n only and therefore will not disturb zeros previously introduced in the last column. At the $(n-1)$ step, G_{n-1} which is constructed to zero the $(2,n)$ element, will also zero the $(1,n)$ element. Indeed, assume that after the $(2,n)$ element has been zeroed we have some value α in the $(1,n)$ position; then we have

$$(H-\lambda I)G_1 \ldots G_{n-1} = \begin{pmatrix} * & \cdots & * & \alpha \\ * & & & \\ & \ddots & \vdots & \\ & & * & 0 \end{pmatrix}.$$

Now if we develop the determinant of this matrix by the last column we get

$$\det[(H-\lambda I)G_1 \ldots G_{n-1}] = \alpha b_1 \ldots b_{n-1},$$

where b_r is the r^{th} subdiagonal element of $(H-\lambda I)G_1 \ldots G_{n-1}$.

$$\det[(H-\lambda I)G_1 \ldots G_{n-1}] = \det(H-\lambda I) = 0,$$

since $(\det(G_r) = 1)$ for $r = 1, \ldots, n-1$. But $b_r \neq 0$ for all r, since we have assumed that H had no zeros on the subdiagonal and since post-multiplication by a G_r can only increase the modulus of a subdiagonal element in $H-\lambda I$. Thus we must have $\alpha = 0$ and therefore at the

$n-1$, then,

$$(I - yy^H)z = z - (y^H z)y = \alpha x + Xv - (\alpha y^H x)y \approx z,$$

showing that z is hardly modified by the projection and therefore suggesting that the algorithm will not necessarily prevent a second convergence to (x, λ).

The algorithm generalizes to the case when λ is multiple in the following way: let V be a right invariant subspace corresponding to λ. Let (V, V_c) be such that

$$(V, V_c)^{-1} H (V, V_c) = \begin{pmatrix} J_\lambda & 0 \\ 0 & J \end{pmatrix},$$

where the right hand side is again the Jordan canonical form of H. J_λ is the Jordan block corresponding to λ; since H is assumed to be unreduced, there can be only one such block. If we set

$$(V, V_c)^{-1} = \begin{pmatrix} U^H \\ W \end{pmatrix},$$

then clearly U^H is a left invariant subspace corresponding to λ and furthermore

$$U^H V_c = 0.$$

This last property will allow U^H to be used in much the same way the left eigenvector was used earlier. However, the practical usefulness of this method is restricted to the case when the eigenvalue λ is simple: indeed, the problem of determining the invariant subspace associated with a multiple eigenvalue λ is an extremely difficult one and can be prohibitively expensive.

This method in its simplest form (using the left eigenvector) adds $O(n^2)$ work to the cost of finding one eigenpair distinct from (x, λ): this is the cost of computing the left eigenvector corresponding to λ; the cost of a single projection is $O(n)$.

3.3 Deflation with orthogonal transformations

We present now a very stable method for obtaining an upper-Hessenberg matrix H' with the property that it has all the eigenvalues of H except for λ [17]. We assume for now that the eigenpair (x, λ) is *exact*.

where the right hand side is the Jordan canonical form of H. Now set

$$(x, X)^{-1} = \begin{pmatrix} y^H \\ Y \end{pmatrix}.$$

Then it is clear that y^H is a left eigenvector of H corresponding to λ and furthermore that

$$y^H X = 0.$$

This property can be used to modify Newton's method to avoid converging to the eigenpair (x, λ) a second time. Indeed, given λ we can compute the left eigenvector y^H corresponding to it and use it to confine the current eigenvector to the space X. Therefore we can expect to converge to an eigenvector linearly independent of x and hence corresponding to a different eigenpair (since (x, λ) was assumed to be simple). When (x, λ) has been obtained once, our algorithm for avoiding it then consists of the following major steps:

Compute the left eigenvector y^H corresponding to λ, with $\|y\|_2 = 1$;
Given the current eigenpair (z, μ) compute the Newton correction from algorithm 2.1;
Let z' be the approximate eigenvector obtained after adding the Newton correction
to z, choose the next eigenvector z'' as:

$$z'' = (I - yy^H)z';$$

In the last step we are just projecting z' onto the space X. We note that when this algorithm is applied, it could happen (as with all the deflation methods we are describing) that we obtain another eigenpair of H, (x', λ'), that was already computed. There will be a need then to do two successive projections when the process is restarted, in order to avoid both (x, λ) and (x', λ').

It is obvious why we need the known eigenpair to be simple for the algorithm just outlined to work. If λ is multiple then left eigenvectors are no longer necessarily orthogonal to X. In fact, the algorithm will be adversely affected if the eigenvalue λ is ill-conditioned, i.e., if $y^H x$ is very small. Indeed, in this case, if the current eigenvector $z = \alpha x + Xv$, where v is a vector of length

to upper-Hessenberg form increases when s approaches 1, it is more advantageous to choose the largest s for which the ratios x_i/x_s are moderate. We wish therefore to define a threshold t for the size of these ratios on the basis of which s will be determined.

Let $\widetilde{H}$ be the computed form of the matrix in (9). In [17], chap. 9, Wilkinson established that if $\|x\|_\infty \leq 1$, then the eigenvalues of $\widetilde{H}$ are the exact eigenvalues of a matrix $\widetilde{H}'$ satisfying

$$\|H - \widetilde{H}'\|_\infty \leq \|r\|_\infty + (n-1)\epsilon,$$

where r is the residual $\lambda x - Hx$ and ϵ is the machine epsilon. With no assumptions on the infinity norm of x, this inequality becomes

$$\|H - \widetilde{H}'\|_\infty \leq \|r\|_\infty + (n-1)\|x\|_\infty \epsilon. \tag{10}$$

Since in our algorithm, our computed eigenpairs have residuals on the order of $n\|H\|_\infty\epsilon$, we propose $\|H\|_\infty$ as a value for t.

In addition to destroying the structure of the matrix, this last method of deflation suffers from the fact that in the case when the eigenvalue to be deflated is non-real, the resulting matrix H' is complex and therefore will considerably increase the cost of finding subsequent eigenpairs if a Newton process is restarted from a real initial guess.

3.2 Deflation with help from the left eigenvector

The method we introduce now is different in spirit from the ones in the previous section, in that no attempt is made at modifying the matrix.

Assume that (x, λ) is an exact eigenpair of H and that λ is simple:

$$Hx = \lambda x.$$

No assumption is made about the remaining eigenvalues of H.

Let (x, X) be such that

$$(x, X)^{-1} H (x, X) = \begin{pmatrix} \lambda & 0 \\ 0 & J \end{pmatrix}$$

let P_{ns} be the matrix obtained from the identity matrix by permuting the n^{th} and s^{th} columns. Then $(P_{ns}x, \lambda)$ is an eigenpair of the matrix $P_{ns}HP_{ns}$, since

$$(P_{ns}HP_{ns})(P_{ns}x) = \lambda P_{ns}x.$$

Now scale the vector $P_{ns}x$ so that the last component is 1 and call that vector $\tilde{x}$. Then, as above, the elementary transformation

$$M = \begin{pmatrix} 1 & & \tilde{x}_1 \\ & \ddots & \vdots \\ 0 & & \tilde{x}_{n-1} \\ & & 1 \end{pmatrix}$$

can be used to deflate the matrix $P_{ns}HP_{ns}$. The fact that here $P_{ns}HP_{ns}$ is not upper-Hessenberg is of no consequence. In fact, we can make the following general statement: Given any matrix A of order n, and any eigenpair (x, λ) of A, then a matrix Q satisfying

$$Q^{-1}x = e_1 \quad \text{or} \quad Q^{-1}x = e_n,$$

can be used to deflate A, in the sense that

$$Q^{-1}AQ = [\lambda e_1, B_1] \quad \text{or} \quad Q^{-1}AQ = [B_2, \lambda e_n],$$

respectively; where B_1 and B_2 are $n \times (n-1)$ matrices.

Having thus deflated the matrix $P_{ns}HP_{ns}$, the leading principal submatrix of order $n-1$ of

$$M^{-1}P_{ns}HP_{ns}M, \tag{9}$$

call it H', has all the eigenvalues of H except λ (if λ is simple). However, H' is not upper-Hessenberg in general and therefore will be reduced back to Hessenberg form before applying Newton's iterations; this is meant to save on the cost of factorizing the Jacobian when solving the linear systems arising at each step of Newton's iteration. Note that it is only the trailing diagonal submatrix of order $n-s+1 \times n-s+1$ of H' that needs to be reduced and that if $s = n-1$ or $s = n$, then H' is upper-Hessenberg. Moreover s need not be chosen so that x_s is the largest component of x (in absolute value). Indeed, since the size of the matrix to be reduced

where D is the diagonal matrix with the following entries

$$\begin{cases} d_i = 1, & \text{if } \frac{|x_i|}{|x_n|} \leq 1 \\ d_i = -\frac{x_n}{x_i}, & \text{if } \frac{|x_i|}{|x_n|} > 1 \end{cases}$$

i.e., D is chosen so that all the entries in N are less than or equal to one. Then we have

$$NHM = \left(\begin{array}{c|c} H' & 0 \\ \hline \overset{\alpha}{0} & \gamma \end{array}\right), \quad NM = \left(\begin{array}{c|c} D' & 0 \\ \hline 0 & \delta \end{array}\right), \tag{7}$$

with

$$\lambda = \gamma/\delta.$$

Also,

$$\sigma(H', D') = \sigma(H, I) - \{\lambda\}$$

and therefore by this transformation λ has been "removed" from the spectrum. Working on the solution of a generalized eigenvalue problem from this point on will not in general cause any dramatic increase in the cost of the algorithm mainly because D' is diagonal. A Newton step with this problem involves the following computation

$$\begin{pmatrix} H' - \lambda_i D' & -D'x_i \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} y \\ \mu \end{pmatrix} = \begin{pmatrix} r_i \\ 0 \end{pmatrix}, \quad \lambda_i \leftarrow \lambda_i + \mu \ ; \ x_i \leftarrow x_i + y \tag{8}$$

where

$$r_i = \lambda_i D' x_i - H x_i.$$

Clearly the previous computation involves an $O(n)$ increase in the cost of one step: this comes from the multiplications by D'. The details on how equation (8) is derived are given in section 9. If λ is complex, then after deflating λ and its conjugate $\bar{\lambda}$ the resulting matrices H' and D' are complex in general. This is the major drawback of this method.

Finally we mention another approach that can be of interest when the similarity transformation (6) involves a very ill-conditioned M (4). This approach consists of interchanging two components of x and the corresponding columns and rows in H so that the last component of x is large enough. More precisely, let x_s be the largest component of x (in absolute value) and

of H'' is $h'_{n-2} - h_{n-1,n-2}x''$, where h'_{n-2} is the last column of the leading principal submatrix of H' and x'' is the vector whose components are the first $n-2$ components of $\tilde{x}$. Since all of the quantities involved are real, H'' is real.

We remark here, as can be readily realized, that H' (or H'') is quite cheap to obtain in practice once an eigenpair of H is available. It requires $O(n)$ operations consisting of a vector normalization, a scalar-vector multiplication and a vector-vector addition. However the conditioning of the matrix M might raise concern. Indeed:

$$cond_\infty(M) = \|M\|_\infty \|M^{-1}\|_\infty \approx max(|x_i|) max\left(\frac{|x_i|}{|x_n|}\right)$$

which can be large if $x_n \ll x_i$. Having noted this, it is clear that the ill-conditioning of M can be easily detected, and therefore one of the more stable (and costlier) methods which we introduce next and in the following sections can be used.

It is possible to prevent the ill-conditioning of M from bearing on the algorithm by avoiding a similarity transformation. More precisely, the eigenvalue problem we want to solve can be thought of as a generalized eigenvalue problem,

$$Hx = \lambda Bx,$$

with $B = I$. We want to find $\sigma(H) = \sigma(H, I)$. Now we know that given any non-singular M and N,

$$\sigma(H, I) = \sigma(NHM, NM).$$

Given a particular eigenpair (x, λ) we would like to choose M and N in a way that solves the problem we set for ourselves at the beginning of this section, namely, we want to reduce the problem to one where λ is no longer in the spectrum. A closer look at the similarity transformation 6, reveals that its deflating property is due to the fact that $M^{-1}x = e_n$. But then M^{-1} is not the only matrix that can be used to accomplish this. In fact the following matrix N can be chosen to reduce x to a multiple of e_n:

$$N = DM^{-1},$$

then it is straightforward to verify that the last column of H' is $h_{n-1} - h_{n,n-1}x'$, where

$$x' = \begin{pmatrix} \frac{x_1}{x_n} \\ \vdots \\ \frac{x_{n-1}}{x_n} \end{pmatrix}.$$

The strategy we have just described is given in [17] and applies regardless of whether the eigenpair (x, λ) is real or not. However, when (x, λ) is real we get a real matrix H'. In the case when (x, λ) is not real, the last column of H' alone is not real since, as we remarked earlier, the other columns are those of a leading principal submatrix of H. Hence the leading principal submatrix of H' of order $n-2$ is real. Also in this case the complex conjugate of (x, λ), $(\bar{x}, \bar{\lambda})$, is an eigenpair of H, and therefore $\bar{\lambda}$ is an eigenvalue of $\widetilde{H}$; a corresponding eigenvector is

$$M^{-1}\bar{x} = \begin{pmatrix} \bar{x}_1 - x_1\frac{\bar{x}_n}{x_n} \\ \vdots \\ \bar{x}_{n-1} - x_{n-1}\frac{\bar{x}_n}{x_n} \\ \frac{\bar{x}_n}{x_n} \end{pmatrix}.$$

Since $\sigma(H') = \sigma(H) - \{\lambda\}$, $\bar{\lambda}$ is an eigenvalue of H' and a corresponding eigenvector is the vector $\tilde{x}$ of length $n-1$, whose components are the first $n-1$ components of a scalar multiple of $M^{-1}\bar{x}$:

$$\tilde{x} = \begin{pmatrix} (\bar{x}_1 x_n - x_1 \bar{x}_n)/i \\ \vdots \\ (\bar{x}_{n-1} x_n - x_{n-1}\bar{x}_n)/i \end{pmatrix},$$

where $i^2 = -1$. This is due to the special structure of $\widetilde{H}$. Now recall that H' has no zeros on the subdiagonal, and so we know that the last component of $\tilde{x}$ is nonzero. Note that $\tilde{x}$ is real. We can carry out a deflation that produces a matrix H'' of order $n-2$ with the property that

$$\sigma(H'') = \sigma(H') - \{\bar{\lambda}\},$$

in the same way we obtained H' from H using the elementary transformation of order $n-1$

$$M' = \begin{pmatrix} 1 & & \tilde{x}_1 \\ & \ddots & \vdots \\ 0 & & \tilde{x}_{n-2} \\ & & \tilde{x}_{n-1} \end{pmatrix}.$$

By a previous remark about this deflation strategy, we know that H'' differs from the leading principal submatrix of H' of order $n-2$ (which is real) in the last column only. The last column

A common drawback that these methods have is that they tend to serialize the computation. However, it has been our experience that the need to deflate arises infrequently, less than 5% of the time in our tests.

3.1 Deflation using elementary transformations

We now describe one possible deflating similarity transformation. We assume that H is an unreduced upper-Hessenberg matrix, λ is an eigenvalue of H and x the corresponding eigenvector.

Since we are assuming H to be upper-Hessenberg with no zeros on the subdiagonal, then $x_n \neq 0$ and the elementary transformation

$$M = [e_1, \ldots, e_{n-1}, x]$$

i.e.,

$$M = \begin{pmatrix} 1 & & & x_1 \\ & \ddots & & \vdots \\ & & 1 & x_{n-1} \\ 0 & & & x_n \end{pmatrix}, \tag{4}$$

is non-singular. The inverse of this matrix is

$$M^{-1} = \begin{pmatrix} 1 & & & -\frac{x_1}{x_n} \\ & \ddots & & \vdots \\ & & 1 & -\frac{x_{n-1}}{x_n} \\ 0 & & & \frac{1}{x_n} \end{pmatrix}. \tag{5}$$

It is easy to see that $M^{-1}x = e_n$. Now we let

$$\widetilde{H} = M^{-1} H M. \tag{6}$$

Then it is easy to verify that the last column of $\widetilde{H}$ is λe_n. Furthermore the leading principal submatrix of order $n-1$ of $\widetilde{H}$, which we will call H', is upper Hessenberg, has the property that

$$\sigma(H') = \sigma(H) - \{\lambda\}$$

and differs from the leading $n-1 \times n-1$ principal submatrix of H in the last column only. In fact, if we let h_{n-1} be the last column of the leading principal submatrix of H of order $n-1$,

Lastly when starting from a real initial guess, only *real* corrections are computed. Therefore the imaginary part of the eigenvalue should be perturbed if convergence to a complex eigenpair is to be made possible. In practice we have done this when real arithmetic does not lead to convergence after a pre-specified number of corrections. To illustrate the situation when perturbing the imaginary part of the eigenvalue is necessary we mention the very simple case of a 2×2 real matrix whose eigenvalues are complex.

3 Deflation

When Newton's method is applied to solve the eigensystem of a matrix H, it is possible that starting from distinct initial guesses we converge to the same eigenpair of H. In fact a naive implementation of the algorithm in section 2 may result in many eigenvalues being found multiple times and consequently some eigenvalues not being found at all since the number of initial guesses is n (at most). To avoid this unwanted situation, we included a deflation step in our algorithm which is designed to obtain further zeros using Newton's method. In this section we propose various methods for doing this. Basically, these methods fall into two classes.

Assume that when algorithm 2.1 is applied to the $n \times n$ matrix H (we will assume for simplicity that it has no multiple eigenvalues) the eigenpair (x, λ) of H is obtained more than once, i.e., the algorithm converges to (x, λ) from several distinct initial guesses $(x_0^{(i)}, \lambda_0^{(i)}), i = 1, \ldots, r, r > 1$. Then the methods of one class produce an $n-1 \times n-1$ matrix H' such that $\sigma(H') = \sigma(H) - \{\lambda\}$. Algorithm 2.1 is then applied to H' starting from $r-1$ of these initial guesses; indeed, we can expect $r-1$ eigenpairs of H to have been missed. If the algorithm converges, then it will do so to eigenpairs different from (x, λ) since λ is no longer in the spectrum.

The other class of methods will reapply algorithm 2.1 to the original matrix H starting from $r-1$ of the initial guesses mentioned above, but will force convergence away from (x, λ) by ensuring at all steps that the current eigenvector forms a non zero angle with x.

processors. However, care must be exercised if the algorithm is to be kept efficient. More will be said about this in section 6 (work estimate). In practice, the original matrix will be dense and we will need to reduce it to upper-Hessenberg form as a first step; this can be done in a stable fashion through a sequence of orthogonal similarity transformations, although elementary transformations can also be used with confidence as in RG [15].

A brief study of some sufficient conditions guaranteeing the convergence of our method will be touched upon in section 4. Now assuming that the algorithm converges, it could happen that the same eigenpair of H is obtained more than once: i.e.,starting from two (or more) distinct initial approximations Newton's method converges to the same eigenpair of H. We have investigated methods to obtain further eigenpairs of H should this happen (see section 3).

We end this section with some implementational details. Our algorithm will accept (x, λ) as an eigenpair of H when $\|Hx - \lambda x\|/\|x\|\|H\| < tol$, where tol is some specified tolerance of order ϵ, the machine unit roundoff. Under these conditions ([8]), (x, λ) is an exact eigenpair of a matrix obtained from H by a slight perturbation: Indeed,

$$(H + \frac{1}{x^H x} r x^H)x = \lambda x,$$

where $r = \lambda x - Hx$.

Starting from two complex conjugate initial approximations, Newton's method will converge to two complex conjugate zeros of F_s, or the same real zero. To prove this it suffices to look at one iteration and show that when the current approximations are complex conjugate, Newton's method yields two complex conjugate corrections, or the same real correction. The system we have to solve starting from (x, λ) is (1). For $(\bar{x}, \bar{\lambda})$ this becomes

$$\begin{pmatrix} H - \bar{\lambda} I & -\bar{x} \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} y' \\ \mu' \end{pmatrix} = \begin{pmatrix} \bar{r} \\ 0 \end{pmatrix}.$$

But this is the same linear system obtained when the conjugate of both sides in (1) is taken, knowing that the matrix is real and hence $\bar{H} = H$. Therefore: $y' = \bar{y}$ and $\mu' = \bar{\mu}$. This will allow significant savings in the computations.

The quantities that vary with k in this expression for the rate of change of λ, are the factors in the numerator. However, $|\alpha|, |y_{k+1}|$ and $|x_k|$ are not really independent of one another: indeed, if $\alpha = 0$, then at least one of y_{k+1} or x_k is zero. Hence for α small, we can expect one of y_{k+1} and x_k to be correspondingly small, since the components of the eigenvectors vary continuously. Therefore, we have found it sufficient in practice to look for the smallest subdiagonal entry in a pre-specified range, and accept it as the subdiagonal entry (in that range) with respect to which the eigenvalues of the matrix are least sensitive, and set it equal to zero.

An outline of the algorithm follows:

Algorithm 2.1 *Given an upper-Hessenberg matrix H, the following algorithm computes the eigensystems of two submatrices of H and uses them as initial guesses for starting Newton iterations for determining the eigensystem of H.*

Determine subdiagonal element α where $H = \left(\begin{array}{c|c} H_{11} & H_{12} \\ \hline 0 \quad \alpha & H_{22} \end{array}\right)$, should be split;

Determine initial guesses from eigensystems of the 2 diagonal blocks H_{11} and H_{22};

For each initial guess (λ_i, x_i) iterate until convergence:

$$\begin{pmatrix} H - \lambda_i I & -x_i \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} y \\ \mu \end{pmatrix} = \begin{pmatrix} r_i \\ 0 \end{pmatrix}; \lambda_i \leftarrow \lambda_i + \mu \ ; \ x_i \leftarrow x_i + y;$$

end;

Check for duplicates and deflate if necessary;

More will be said about the last step, deflation, in section 3. The algorithm just described is inherently parallel. Indeed, the eigensystems of H_{11} and H_{22} can be computed in parallel. After which Newton's method can be started with different initial guesses on different processors. Furthermore, the dividing process can be applied recursively to obtain yet smaller subproblems, i.e., H_{11} and H_{22} can be divided each into subproblems. This will permit the use of more

should satisfy the same condition, then (y, μ) is the solution of

$$\begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} y \\ \mu \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix},$$

with r as above. But, this is the same equation that we obtain when a Newton iteration is applied to the function F_s to find a correction to (x, λ), as we can see from (1). We note here a result from [4], where the author studied an iterative refinement technique to compute the correction (y, μ) to (x, λ) from equation (2) (i.e., without ignoring the second order term) and proved that in *exact* arithmetic, that method and Newton's method produce the same final iteration.

So far we have not attempted to answer the question of which subdiagonal entry to introduce the zero. We will use first order perturbation theory in order to shed some light on this issue. Let $E = -\alpha e_{k+1} e_k^T$, where as above: $\alpha = h_{k+1,k}$ (note that $H + E = H_0$, introduced above). Then classical results from function theory ([12], v. 2, pp. 119-134) allow us to state that in a small neighborhood of zero, we have

$$(H + \epsilon E)x(\epsilon) = \lambda(\epsilon)x(\epsilon),$$

for all ϵ in that neighborhood and for differentiable $(x(\epsilon), \lambda(\epsilon))$ corresponding to a simple eigenpair (x, λ) of H. Clearly: $x(0) = x$ and $\lambda(0) = \lambda$. Let y^H be the left eigenvector of H corresponding to λ. Then differentiating both sides we have,

$$H D_\epsilon x(\epsilon) + Ex(\epsilon) + \epsilon E D_\epsilon x(\epsilon) = D_\epsilon \lambda(\epsilon)x(\epsilon) + \lambda(\epsilon) D_\epsilon x(\epsilon),$$

where, D_ϵ denotes the diffenrentiation with respect to ϵ. Multiplying by y^H and setting $\epsilon = 0$ we get

$$y^H E x = D_\epsilon \lambda(0) y^H x,$$

and therefore

$$|D_\epsilon \lambda(0)| = \frac{|y^H E x|}{|y^H x|} = \frac{|\alpha||y_{k+1}||x_k|}{|y^H x|}. \tag{3}$$

H_0 corresponding to λ. On the other hand, if λ is an eigenvalue of H_{22} and x is the corresponding eigenvector, then $\begin{pmatrix} 0 \\ x \end{pmatrix}$, where k zeros are prefixed to x is taken as an approximate eigenvector of H corresponding to the approximate eigenvalue λ. If λ is not an eigenvalue of H_{11} then the last $n-k$ components of $\begin{pmatrix} 0 \\ x \end{pmatrix}$, i.e., those of x, are the trailing components of an exact eigenvector of H_0 corresponding to λ, namely: $\begin{pmatrix} (H_{11}-\lambda)^{-1}H_{12}x \\ x \end{pmatrix}$. However, if λ is an eigenvalue of H_{11}, and hence a necessarily multiple eigenvalue of H_0, then if λ has geometric multiplicity one, no eigenvector of H_0 will have the components of x as its trailing components.

Having constructed these initial guesses, we now use them as starting points for Newton's method to find the zeros of F_s, i.e., the eigenpairs of H. Hence, given an approximate zero (x, λ) of F_s the next approximation is

$$x' = x + y; \lambda' = \lambda + \mu,$$

where (y, μ) is the solution of the system

$$\begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} y \\ \mu \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}, \tag{1}$$

where $r = \lambda x - Hx$.

Newton's method comes into this problem in a rather "natural" way. Indeed, suppose that (x, λ) is an approximate eigenpair of H,

$$Hx \approx \lambda x,$$

and let us find a way to compute a correction (y, μ) to this approximate eigenpair. Clearly, (y, μ) should satisfy

$$H(x + y) = (\lambda + \mu)(x + y).$$

Rearranging the latter equation yields

$$(H - \lambda)y - \mu x = \lambda x - Hx + \mu y. \tag{2}$$

Now we ignore the second order term μy, and we impose a normalization condition on x, say $x_s = 1$, where x_s is the s^th component of x. If we also assume that the desired eigenvector

2 The Algorithm

Given a matrix H, an eigenpair (x_0, λ_0) of H can be thought of as a solution to the following polynomial system

$$(S)\begin{cases} Hx - \lambda x = 0 \\ L(x) = 1 \end{cases}$$

where $L(x)$ is a scalar equation. Here, we set $L(x) = e_s^T x$, where e_s is the s^{th} unit vector. Let

$$F_s(x, \lambda) = \begin{pmatrix} Hx - \lambda x \\ e_s^T x - 1 \end{pmatrix}.$$

Then, finding an eigenpair of H reduces to finding a zero of F_s. In what follows, unless otherwise mentioned, H is assumed to be a real, unreduced (no zeros on the subdiagonal), upper-Hessenberg matrix of order n. This is no restriction on the type of problems we want to solve, since if H has a zero on the subdiagonal then finding its eigenvalues reduces to finding those of the blocks on the diagonal. We note also that as a consequence of our assumption that H is unreduced, an eigenvalue of H can only have geometric multiplicity one: this is quite easy to see since the first $n-1$ columns of $H - \lambda I$ are linearly independent. We will assume for now that H has a simple spectrum. We can write H as

$$H \quad = \quad \left(\begin{array}{c|c} H_{11} & H_{12} \\ \hline \alpha e_1^{(k)} e_k^{(k)T} & H_{22} \end{array}\right),$$

where H_{11} and H_{22} are upper-Hessenberg of dimensions $k \times k$ and $n-k \times n-k$, respectively; $\alpha = h_{k+1,k}$, and $e_i^{(k)}$ is the i^{th} unit vector of length k.

Let $H_0 = H - \alpha e_{k+1}^{(n)} e_k^{(n)T}$, then $H_0 = \left(\begin{array}{c|c} H_{11} & H_{12} \\ \hline 0 & H_{22} \end{array}\right)$, and $\sigma(H_0) = \sigma(H_{11}) \cup \sigma(H_{22})$ (where $\sigma(M)$ is the spectrum of M). The algorithm can then be described as follows. We first find the k eigenpairs of H_{11} and the $n-k$ eigenpairs of H_{22} by some method, perhaps the QR algorithm. These eigenpairs are then used to construct initial approximations to the eigenpairs of H in the following way: if λ is an eigenvalue of H_{11} and x is the corresponding eigenvector, then λ is viewed as an approximate eigenvalue of H with the corresponding approximate eigenvector taken to be $\begin{pmatrix} x \\ 0 \end{pmatrix}$, where $n-k$ zeros are appended to x. Note that $\begin{pmatrix} x \\ 0 \end{pmatrix}$ is an exact eigenvector of

nal form). This approach in turn enables rapid and accurate determination, in parallel, of the eigenvalues and the associated eigenvectors.

In this paper we propose a parallel algorithm for the solution of the non-symmetric eigenvalue problem. The approach uses some of the features of the divide-and-conquer algorithm for the symmetric case mentioned earlier. In particular, the original problem is divided into two smaller and independent subproblems by a rank-one modification of the matrix. (We assume that the matrix has already been reduced to Hessenberg form and that the rank-one modification removes a subdiagonal element). Once the eigensystems of the smaller subproblems are known, it is possible to compute those of the original matrix. In the non-symmetric case, however, the eigenvalues of the modified matrix do not interlace with those of the original matrix. Indeed, the eigenvalues can scatter anywhere in the complex plane.

In our algorithm for the non-symmetric case, the eigensystem of the subproblem is used only to construct initial guesses for an iterative process that yields the desired eigensystem of the original problem. Under suitable conditions, Newton's method or a continuation method can be used to find the eigenpairs of the original problem. We report here on our application of an iterative refinement approach based on Newton's method; we shall not pursue the continuation method in this paper. Work on the continuation approach has been reported by [13].

In section 2 we describe the algorithm using an iterative refinement procedure based on Newton's method. Section 3 covers the deflation step required to overcome multiple convergence to a particular eigenvalue. In section 4 the convergence behavior is discussed. In section 5 we discuss the case when the matrix or its rank one modification has a defective system of eigenvectors. Section 6 estimates the amount of work the parallel algorithm requires and compares this to the standard techniques. Section 7 describes the parallel algorithm and the different parallel implementations of the new algorithm, and gives numerical results. Section 9 describes how our ideas extend to the generalized eigenvalue problem.

A Parallel Algorithm for the Non-Symmetric Eigenvalue Problem *

Jack J. Dongarra
and
Majed Sidani
Computer Science Department
University of Tennessee
Knoxville, TN 37996-1301

and

Mathematical Sciences Section
Oak Ridge National Laboratory
Oak Ridge, TN 37831

September 13, 1991

Abstract

This paper describes a parallel algorithm for computing the eigenvalues and eigenvectors of a non-symmetric matrix. The algorithm is based on a divide-and-conquer procedure and uses an iterative refinement technique.

1 Introduction

The algebraic eigenvalue problem is one of the fundamental problems in computational mathematics. It arises in many applications and therefore represents an important area of algorithmic research. The problem has received considerable attention which has resulted in reliable methods [17, 16, 15]. However it is reasonable to expect that calculations might be accelerated through the use of parallel algorithms. A fully parallel algorithm for the symmetric eigenvalue problem was recently proposed in [6]. This algorithm is based on a divide-and-conquer procedure outlined in [3]. The latter was based on work in [10] and [1]. The fundamental principle behind this algorithm is that the partitioning by rank-one tearing interlaces the eigenvalues of the modified problem with the eigenvalues of the original problem (the matrix is first reduced to tridiago-

*This work was supported in part by the National Science Foundation Science and Technology Center Cooperative Agreement No. CCR-8809615, and in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S Department of Energy, under Contract DE-AC05-84OR21400.

CS - 91 - 137

A Parallel Algorithm for the Non-Symmetric Eigenvalue Problem

*Jack J. Dongarra**

and

Majed Sidani

Computer Science Department
University of Tennessee
Knoxville, TN 37996-1301

and

Mathematical Sciences Section
Oak Ridge National Laboratory
Oak Ridge, TN 37831

CS - 91 - 137

September 13, 1991

* Electronic mail address: dongarra@cs.utk.edu and sidani@cs.utk.edu.

This work was supported in part by the National Science Foundation Science and Technology Center Cooperative Agreement No. CCR-8809615, and in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S Department of Energy, under Contract DE-AC05-84OR21400.