Electronic Transactions on Numerical Analysis.
Volume 23, pp. 1-4, 2006.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

A CASE WHERE BALANCING IS HARMFUL*

DAVID S. WATKINS†

Abstract. Balancing is a common preprocessing step for the unsymmetric eigenvalue problem. If a matrix is badly out of scale, balancing can markedly improve the accuracy of the computed eigenvalues. This paper discusses a situation where balancing has the opposite effect. If a matrix that is not out of scale has been transformed to upper Hessenberg form, a subsequent balancing of the Hessenberg matrix will cause the condition numbers of the eigenvalues to be degraded. Consequently the computed eigenvalues will be substantially less accurate than they would have been if the Hessenberg matrix had not been balanced.

Key words. eigenvalues, balancing, condition number, Hessenberg form

AMS subject classifications. 65F15, 15A18

1. Balancing. When an unsymmetric eigenvalue problem is solved using standard software such as MATLAB or LAPACK [1], the first step is often to balance the matrix [2, 4, 5]. The input matrix A is replaced by a rescaled matrix $\tilde{A} = D^{-1}AD$, where D is a diagonal matrix chosen so that, for each i, the ith row and the ith column of $\tilde{A}$ have roughly the same norm.[1] If the rows and columns of a matrix are badly out of scale to begin with, the balancing step can have the effect of greatly reducing the norm of the matrix. After the balancing step, the matrix is transformed to upper Hessenberg form, and its eigenvalues are computed by the QR algorithm [3, 6]. If the matrix was out of scale to begin with, the use of balancing can greatly improve the accuracy of the computed eigenvalues. In some cases balancing is crucial to the success of the computation.

2. Balancing Hessenberg Matrices. In this paper we wish to discuss one situation in which balancing has a detrimental effect on accuracy. Consider a scenario in which the matrix has, for one reason or another, already been reduced to upper Hessenberg form, with or without preliminary balancing. Suppose that the researcher is using MATLAB to do his computations. (This researcher happens to be male.) Now, if he wants to find the eigenvalues of the Hessenberg matrix H, he types `eig(H)`. Unless he thinks to add the optional argument `'nobalance'`, MATLAB will balance the Hessenberg matrix before using the QR algorithm to compute the eigenvalues. This turns out to be a bad idea.

Consider a matrix A that is not normal but is well scaled. Suppose all eigenvalues of A have algebraic multiplicity 1 and are well conditioned. For example, a random matrix (almost always) has these characteristics. If we compute its eigenvalues using MATLAB's `eig` command, the matrix gets balanced first. This step has no effect, because A is already well scaled. The matrix is then reduced to upper Hessenberg form, and its eigenvalues are computed by the QR algorithm. Because all of the eigenvalues are well conditioned, `eig` is able to compute them extremely accurately.

Now suppose that we compute the eigenvalues of A by a nonstandard path. First we transform A to an upper Hessenberg matrix $H = V^*AV$ by a unitary similarity transformation. Then we balance the Hessenberg matrix before computing its eigenvalues. The transformation to Hessenberg form has the effect of seriously unbalancing the matrix. Consider what

*Received April 13, 2005. Accepted for publication November 29, 2005. Recommended by W. B. Gragg.
†Department of Mathematics, Washington State University, Pullman, Washington 99164-3113 (watkins@math.wsu.edu).
[1]Balancing programs in common use [5, 2] also include a sorting step that identifies obvious invariant subspaces. In this paper we do not concern ourselves with the sorting step.

ETNA
Kent State University
etna@mcs.kent.edu

happens on the first step of the reduction [3, 6]. The first column undergoes a transformation

$$\begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ \vdots \\ a_{n1} \end{bmatrix} \to V_1^* a = \begin{bmatrix} h_{11} \\ h_{21} \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

where $h_{11} = a_{11}$ and $h_{21} = (a_{21}^2 + a_{31}^2 + \cdots + a_{n1}^2)^{1/2}$. All of the weight is shifted from the third through nth components onto the second component. This shifts a great deal of weight onto the second row at the expense of the third through nth rows. The transformation $A \to V_1^* A$ also affects columns 2 through n. Weight gets shifted among the rows but not in a systematic way. The 2-norms of the columns are invariant under this transformation. The first step is completed by multiplying by V_1 on the right to complete the similarity transformation: $V_1^* A \to V_1^* A V_1$. This transformation does not touch the first column. It shifts weight around in columns 2 through n, but not in a systematic way. The 2-norms of the rows are unchanged by this transformation. In summary, the first step of the reduction acts systematically on the first column, shifting weight from rows 3 through n onto row 2. The other effects of the reduction are nonsystematic. We expect the norm of the second row to be increased significantly at the expense of the third through nth rows. The larger the matrix is, the bigger the effect will be. We expect the norms of the columns not to undergo any systematic change.

The second step of the reduction is just like the first, except that it piles weight onto the third row at the expense of rows 4 through n. In general the kth step adds weight to row $k+1$ at the expense of rows $k+2$ through n. After $n-2$ steps, a great imbalance can be built up. The larger the matrix is, the more potential for imbalance there is.

We haven't proven anything; we have just argued that it is plausible that the Hessenberg matrix could be seriously unbalanced. That such an unbalance does build up in practice is confirmed by numerical experiments, some of which will be presented below.

Even though the Hessenberg matrix may be seriously "out of balance" in a certain sense, its eigenvalues are well conditioned. The unitary similarity transformation $H = V^* A V$ leaves the condition numbers unchanged.

When we apply the balancing algorithm to H, it effects a diagonal similarity transformation $\tilde{H} = D^{-1} H D$, where D may have a very large condition number. This has a bad effect on the condition numbers of the eigenvalues. Recall that the condition number of an eigenvalue λ of H, is determined by the relationship between its left and right eigenvectors. Suppose $w^T A = \lambda w^T$ and $Av = v\lambda$, where w and v are normalized so that $w^T v = 1$. Then the condition number of λ is $\kappa(\lambda, H) = \| w \|_2 \| v \|_2$ [3, 6]. A large condition number is bad. The best possible condition number is 1. We are assuming our matrix has well conditioned eigenvalues, so none of these numbers is very large. Now consider the eigenvectors of $\tilde{H}$. Corresponding to eigenvalue λ we have left and right eigenvectors $w^T D$ and $D^{-1} v$, which satisfy $(w^T D)(D^{-1} v) = 1$, so the condition number of λ is $\kappa(\lambda, \tilde{H}) = \| Dw \|_2 \| D^{-1} v \|_2$. If D is ill conditioned, this quantity can be large. If some of the larger components of D line up with nontrivial components of w, while some of the larger components of D^{-1} line up with nontrivial components of v, then $\kappa(\lambda, \tilde{H})$ will certainly be large. Again we have not proved anything, but it seems inevitable that some of the eigenvalues of $\tilde{H}$ will have large condition numbers. If so, the QR algorithm applied to $\tilde{H}$ will not be able to compute them accurately. These expectations are borne out by the numerical experiments.

3. Numerical Results. We will present just two examples that are typical of the many that confirmed the expectations raised in the previous section. We chose fairly large examples

ETNA
Kent State University
etna@mcs.kent.edu

in order to get dramatic effects. Both of our matrices are of dimension 800.

EXAMPLE 3.1. Our first example was generated by the MATLAB command

```
A = randn(n) + i * randn(n);
```

with $n = 800$. Thus its entries were iid random variables whose real and imaginary parts have mean 0 and standard deviation 1. Although we did not know the exact eigenvalues of this matrix, we were able approximate them very accurately, because they were all well conditioned. According to MATLAB's `condeig` function, the average eigenvalue condition number of this matrix was 17, and the worst condition number was about 180. Thus our computed eigenvalues could be in error by no more than a modest multiple of $180u$, where $u \approx 10^{-16}$ is the unit roundoff for double-precision arithmetic.

We transformed A to Hessenberg form H, then balanced H to produce another upper Hessenberg matrix $\tilde{H} = D^{-1}HD$. As predicted, the diagonal scaling matrix D turned out be quite ill conditioned. Its largest main diagonal entry was 1.6×10^4 and its smallest was 7.5×10^{-9}, yielding a condition number of 2.2×10^{12}. This means that the condition numbers of the eigenvalues of $\tilde{H}$ could be up to 10^{12} times as large as those of H. In fact they turned out to be somewhat less than that but bad nevertheless. Using MATLAB's `condeig` function we found that the average eigenvalue condition number of $\tilde{H}$ was 2.9×10^8, and the largest condition number was 3.4×10^{10}.

We computed the eigenvalues of $\tilde{H}$ using the QR algorithm and compared the computed eigenvalues with the eigenvalues of A computed earlier, which are accurate and can be considered to be "exact". The average error of the computed eigenvalues of $\tilde{H}$ was 2.2×10^{-7}, and the worst error was 2.7×10^{-5}. Thus balancing caused the eigenvalues of the benign matrix A to be computed with surprisingly little accuracy. The errors were a good million times bigger than they should have been.

EXAMPLE 3.2. Our second example was a random non-normal 800×800 matrix with known eigenvalues. An upper-triangular matrix T was constructed as follows. The main-diagonal entries, the eigenvalues, were chosen to be independent random complex numbers whose real and imaginary parts were normally distributed with mean 0 and standard deviation 1. The entries above the main diagonal were constructed just as the main diagonal entries were, except that the standard deviation was $1/10$ instead of 1. This gave the matrix a significant but not-too-severe departure from normality. A random unitary similarity transformation was applied to the upper-triangular matrix to yield a full matrix with known eigenvalues.[2] The matrix was then reduced to upper Hessenberg form H. Because of the departure from normality, the eigenvalues of this matrix were not as well conditioned as those of the matrix in Example 3.1. See Table 3.1 . The Hessenberg matrix H was balanced to produce another upper Hessenberg matrix $\tilde{H} = D^{-1}HD$. In this example the condition number of D was even worse than in Example 3.1. The largest main-diagonal entry of D was 1.3×10^5 and the smallest was 1.8×10^{-12}, yielding a condition number of 7.2×10^{16}. This suggests that some of the eigenvalues of $\tilde{H}$ may be so ill conditioned that they cannot be computed with any accuracy at all.

The actual computed condition numbers are given in Table 3.1. As we mentioned earlier, the eigenvalues of H are somewhat ill conditioned, but they are not so bad that the eigenvalues cannot be computed with at least a few digits accuracy. On the other hand, the eigenvalues of $\tilde{H}$ are really badly conditioned.

We computed the eigenvalues of H and $\tilde{H}$ by the QR algorithm and checked their accuracy by comparing them with the eigenvalues of the original upper-triangular matrix T. The results are given in Table 3.2. The eigenvalues of H were computed with errors around 10^{-6}

[2]Technically speaking, we know the eigenvalues of T but not of A. Because of roundoff errors, the eigenvalues of A are not exactly the same as those of T.

TABLE 3.1
Eigenvalue condition numbers for Example 3.2

	average	maximum
H (unbalanced)	6.3×10^7	9.3×10^9
$\tilde{H}$ (balanced)	4.5×10^{12}	6.1×10^{14}

TABLE 3.2
Errors in computed eigenvalues for Example 3.2

	average	maximum
H (unbalanced)	1.1×10^{-8}	2.1×10^{-6}
$\tilde{H}$ (balanced)	2.0×10^{-2}	8.7×10^{-1}

or less. This is about what we would expect, given the condition numbers. On the other hand, but also in line with the condition numbers, we were unable to compute the eigenvalues of $\tilde{H}$ with any accuracy at all. The average error was about 0.02. Again balancing has degraded the accuracy.

4. Conclusions. Balancing sometimes seriously degrades accuracy. In particular, one should not balance a matrix after it has been transformed to Hessenberg form. However, we must emphasize that balancing is usually not harmful and often very beneficial. When in doubt, balance.

Acknowledgement. The phenomenon discussed in this brief paper was not dicovered in the way it was presented here. The author spent several days wondering what was wrong with his codes. He is indebted to David Day for pointing out that balancing was the issue.

REFERENCES

[1] E. ANDERSON ET AL., *LAPACK Users' Guide*, SIAM, Philadelphia, Third ed., 1999. Available also at http://www.netlib.org/lapack/lug/lapack_lug.html.
[2] T. Y. CHEN AND J. W. DEMMEL, *Balancing sparse matrices for computing eigenvalues*, Linear Algebra Appl., 309 (2000), pp. 261–287.
[3] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Third ed., 1996.
[4] E. E. OSBORNE, *On pre-conditioning of matrices*, J. Assoc. Comput. Mach., 7 (1960), pp. 338–345.
[5] B. N. PARLETT AND C. REINSCH, *Balancing a matrix for calculation of eigenvalues and eigenvectors*, Numer. Math., 13 (1969), pp. 293–304.
[6] D. S. WATKINS, *Fundamentals of Matrix Computations*, John Wiley and Sons, Second ed., 2002.