A Nonlinear QR Algorithm for Banded Nonlinear Eigenvalue Problems

C. Kristopher Garrett* Zhaojun Bai† Ren-Cang Li‡

January 28, 2016

Abstract

A variation of Kublanovskaya's nonlinear QR method for solving banded nonlinear eigenvalue problems is presented in this paper. The new method is iterative and specifically designed for problems too large to use dense linear algebra techniques. For the unstructurally banded nonlinear eigenvalue problem, a new data structure is used for storing the matrices to keep memory and computational costs low. In addition, an algorithm is presented for computing several nearby nonlinear eigenvalues to already computed ones. Finally, numerical examples are given to show the efficacy of the new methods, and the source code has been made publicly available.

1 Introduction

Given an $n \times n$ (complex) matrix-valued function $H(\lambda)$, the associated *nonlinear eigenvalue problem* (NLEVP) is to determine nonzero n-vectors x and y and a scalar μ such that

$$H(\mu)x = 0, \quad y^*H(\mu) = 0. \tag{1.1}$$

When this equation holds, we call μ a *nonlinear eigenvalue*, x an associated *right eigenvector* (or simply eigenvector), and y an associated *left eigenvector*. The cases $H(\lambda) = A - \lambda I$, $A - \lambda B$, and a matrix polynomial correspond to the linear, generalized, and polynomial eigenvalue problems, respectively.

Recent demands from various applications have been reigniting interest in the numerical analysis and scientific computing community to seek efficient solutions to certain nonlinear eigenvalue problems, as manifested by the establishment of the collection [2] and the survey articles [18, 21].

Kublanovskaya [15] proposed an iterative method for finding a nonlinear eigenvalue and associated eigenvector. The method requires one QR decomposition with column pivoting

*Computational Physics and Methods, Los Alamos National Laboratory, Los Alamos, NM 87544. E-mail: ckgarrett@lanl.gov. The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

†Department of Computer Science and Department of Mathematics, University of California, Davis, CA 95616. E-mail: bai@cs.ucdavis.edu. Supported in part by NSF grants DMR-1035468 and DMS-1115817.

‡Department of Mathematics, University of Texas at Arlington, P.O. Box 19408, Arlington, TX 76019 E-mail: rcli@uta.edu. Supported in part by NSF grants DMS-1115834 and DMS-1317330, and a Research Gift Grant from Intel Corporation.

per iteration step. Thus generally it is very expensive because each QR decomposition usually costs $O(n^3)$ flops. Also column pivoting makes it very nearly impossible to take advantage of any additional structure in $H(\lambda)$. Consequently, the method as is is not suitable for $H(\lambda)$ of large size.

This paper presents a variation of Kublanovskaya's method [15] for efficiently computing solutions of the nonlinear eigenvalue problem for which either $H(\lambda)$ is banded with narrow bandwidths relative to n or can be reduced to such a banded matrix. The new method works not only for regularly banded $H(\lambda)$ but also unstructurally banded $H(\lambda)$, by which we mean each row may have a different left and right bandwidth from another.

The methods presented in this paper basically use the Newton method which is known to usually converge quadratically. A recent work on the convergence analysis of Newton-type methods for general nonlinear eigenvalue problems can be found in [20]. To use Newton-type methods, the only important parameter is the choice of initial guess. This simplicity makes them particularly easy to use and flexible in adaptation into large computational tasks that need to solve certain nonlinear eigenvalue problems.

Recently, two MATLAB software projects were developed for nonlinear eigenvalue problems, namely NLEIGS [10] for a class of rational Krylov methods and CORK [22] for a compact rational Krylov method. These are Krylov subspace projection based methods and are capable of exploiting the underlying problem structure via matrix-vector products. To use these methods, it is necessary to carefully choose a number of critical parameters for fast convergence and high accuracy. However, such methods have the potential to find multiple nonlinear eigenvalues simultaneously. Although a comparison of Krylov type methods such as implemented in NLEIGS and CORK and the Newton-type methods such as the nonlinear QR algorithms presented in this paper is a subject of further study, a brief comparison is presented in the results section.

Throughout this paper we assume that $H(\lambda)$ is (at least) twice differentiable at the desired nonlinear eigenvalues. The $n \times n$ identity matrix will be denoted I_n (or simply I if its dimension is clear from the context) with e_j denoting its jth column. We shall also adopt the MATLAB-like convention to access the entries of vectors and matrices. Let $i:j$ be the set of integers from i to j inclusive. For a vector u and a matrix X, $u_{(j)}$ is u's jth entry, $X_{(i,j)}$ is X's (i,j)th entry; X's submatrices $X_{(k:\ell,i:j)}$, $X_{(k:\ell,:)}$, and $X_{(:,i:j)}$ consist of intersections of row k to row ℓ and column i to column j, row k to row ℓ, and column i to column j, respectively. X^{T} and X^* are the transpose and the complex conjugate transpose of the matrix/vector X, respectively; $\|\cdot\|_2$ denotes the ℓ_2 vector norm or the matrix spectral norm, $\|A\|_{\mathrm{F}}$ the Frobenius norm of matrix A; $\sigma_{\min}(X)$ is the smallest singular value of X. Finally, the term *nonlinear eigenvalue* will be used throughout to refer to any solution μ to (1.1). This is in contrast to using the term eigenvalue, which will be used to denote the classical definition of an eigenvalue for a matrix. The authors note the term *nonlinear eigenvalue* is not standard, but given the term's frequency of use in this paper, it was chosen as a slightly less cumbersome variant of other options.

2 Kublanovskaya's Method and Rank Revealing

2.1 Kublanovskaya's method

The basic idea of Kublanovskaya's method is to apply Newton's method to the last diagonal entry in the R-factor of $H(\lambda)$'s QR decomposition with column pivoting [8, p.258]:

$$H(\lambda)\Pi(\lambda) = Q(\lambda)R(\lambda),$$

where $\Pi(\lambda)$ is a permutation matrix.[1] Rigorously speaking, the decomposition is not continuous because $\Pi(\lambda)$ is not continuous. But this does not pose any numerical difficulty because the role of $\Pi(\lambda)$, as the result of the column pivoting, is to make sure the last diagonal entry $r_{nn}(\lambda)$ of $R(\lambda)$ becomes zero before any other diagonal entry does. Therefore, at any given λ_0, we can keep $\Pi(\lambda)$ constant in a small neighborhood of λ_0 and then Newton's method can be applied to drive $H(\lambda)$ towards being more singular, *provided* $r_{nn}(\lambda)$ *is differentiable at* λ_0.

This last condition for the applicability of Newton's method had caused a stir in the past. As was noticed by Jain and Singhal [13], Kublanovskaya's derivation of $r'_{nn}(\lambda_0)$ is flawed for complex λ. Jain and Singhal gave a 2×2 example

$$H(\lambda) = \begin{bmatrix} \lambda & 1 \\ 1 & \lambda \end{bmatrix} \tag{2.1}$$

to claim that $r'_{nn}(\lambda_0)$ might not exist. Unfortunately they made an error in their calculated QR decomposition of $H(\lambda)$ (near $\lambda_0 = 0$):

$$\alpha \begin{bmatrix} |\lambda|^2 & \bar{\lambda} \\ \bar{\lambda} & -|\lambda|^2 \end{bmatrix} \times \alpha \begin{bmatrix} \lambda(1+|\lambda|^2) & \lambda(\lambda+\bar{\lambda}) \\ 0 & \bar{\lambda}(1-\lambda^2) \end{bmatrix}, \tag{2.2}$$

where $\alpha = \left[|\lambda|^2(1+|\lambda|^2)\right]^{-1/2}$. It can be seen that the $(1,2)$ entry of the product is $\alpha^2\left[\bar{\lambda}^2 + \lambda^3\bar{\lambda}\right] \neq 1$, and thus it is an incorrect QR decomposition for $H(\lambda)$. The correct QR decomposition, however, is

$$H(\lambda) = \beta \begin{bmatrix} \lambda & 1 \\ 1 & -\bar{\lambda} \end{bmatrix} \times \beta \begin{bmatrix} 1+|\lambda|^2 & \lambda+\bar{\lambda} \\ 0 & 1-\lambda^2 \end{bmatrix}, \tag{2.3}$$

where $\beta = \left(1+|\lambda|^2\right)^{-1/2}$. Thus $r_{22}(\lambda) = \beta(1-\lambda^2) = 1+O(\lambda^2)$ which yields the derivative $r'_{22}(0) = 0$ (see [16]).

Despite the flaw in Kublanovskaya's derivation, her expression for $r'_{nn}(\lambda_0)$ was later proved still valid [16]. Let

$$H(\lambda_0)\Pi = Q_0 R_0 \equiv Q_0 \times \begin{matrix} \\ n-1 \\ 1 \end{matrix} \begin{matrix} \begin{matrix} n-1 & 1 \end{matrix} \\ \begin{bmatrix} R_{11}^{(0)} & r_{12}^{(0)} \\ 0 & r_{nn}^{(0)} \end{bmatrix} \end{matrix} \tag{2.4}$$

be the QR decomposition for $H(\lambda_0)\Pi$, where Π is a permutation matrix such that $R_{11}^{(0)}$ is nonsingular. Then in some neighborhood of λ_0, there exists a QR decomposition of $H(\lambda)\Pi$

$$H(\lambda)\Pi = Q(\lambda)\, R(\lambda) \equiv Q(\lambda) \times \begin{matrix} \\ n-1 \\ 1 \end{matrix} \begin{matrix} \begin{matrix} n-1 & 1 \end{matrix} \\ \begin{bmatrix} R_{11}(\lambda) & r_{12}(\lambda) \\ 0 & r_{nn}(\lambda) \end{bmatrix} \end{matrix}, \tag{2.5}$$

such that (from Appendix A, see also Remark 2.1.4)

$$r_{nn}(\lambda) = r_{nn}^{(0)} + (Q_0 e_n)^*\, H'(\lambda_0)\Pi \begin{bmatrix} -z \\ 1 \end{bmatrix} (\lambda - \lambda_0) + O(|\lambda - \lambda_0|^2), \tag{2.6}$$

[1] Kublanovskaya actually did equivalently the QR decomposition for $[H(\lambda)]^*$.

and z is determined by $R_{11}^{(0)} z = r_{12}^{(0)}$. Therefore

$$r'_{nn}(\lambda_0) = (Q_0 e_n)^* \, H'(\lambda_0) \Pi \begin{bmatrix} -z \\ 1 \end{bmatrix}. \tag{2.7}$$

Consequently, the next approximation by Newton's method is

$$\lambda_1 = \lambda_0 - \frac{r_{nn}(\lambda_0)}{r'_{nn}(\lambda_0)} = \lambda_0 - \frac{r_{nn}^{(0)}}{r'_{nn}(\lambda_0)}. \tag{2.8}$$

This process repeats itself to generate a sequence of approximations λ_i until convergence which is usually recognized by, e.g., checking if $|r_{nn}(\lambda_i)|/\|H(\lambda_i)\|_{\mathrm{F}}$ is less than or equal to a given tolerance. At convergence, one can take

$$x = \Pi \begin{bmatrix} -z \\ 1 \end{bmatrix}, \quad y = Q_i e_n \tag{2.9}$$

as corresponding right and left eigenvectors.

Remark 2.1. There are several remarks to be made on the preceding Newton' method.

1. Iteration (2.8) is mathematically the same as the original Kublanovskaya's formula (applied to $[H(\lambda)]^*$). However the right-hand side of (2.8) does not involve the inverse of R_0 but only $R_{11}^{(0)}$ (see (2.7)), unlike Kublanovskaya's formulation, and thus is better suited for numerical use.

2. The permutation matrix Π does not necessarily have to come from the QR decomposition with column pivoting. Its only role, as far as the derivations from (2.4) to (2.8) is to ensure that $R_{11}^{(0)}$ is nonsingular. Of course, practically Π that makes $r_{nn}^{(0)}$ comparable in magnitude to the smallest singular value $\sigma_{\min}(R_0)$ of R_0 should be preferred. This observation, though simple, is numerically significant and what the rest of our development relies on.

3. It is possible that for a given nonlinear eigenvalue, μ, the matrix $R(\mu)$ has a nullspace of dimension greater than one. In this case, $R_{11}^{(0)}$ in (2.4) will become "less and less nonsingular" as $\lambda_0 \to \mu$ and thus it may become more and more difficult to accurately solve for z in (2.6) and (2.7). In [17, Section 3], a more general algorithm is proposed where after a set number of iterations, r_{nn} may be replaced by a block matrix whose size changes iteratively to ultimately match the dimension of the nullspace of $R(\mu)$.

4. There is a subtlety in the definition of $r_{nn}(\lambda)$ and $r'_{nn}(\lambda_0)$ from (2.6) and (2.7) that needs to be addressed since the QR decomposition is not unique. In [16], it was shown that given Q_0 and R_0 from (2.4), there is a QR decomposition such that $H(\lambda)\Pi = Q(\lambda)R(\lambda)$ with Q and R differentiable at $\lambda = \lambda_0$. This is the QR decomposition implicitly referenced in the definition of $r_{nn}(\lambda)$ and $r'_{nn}(\lambda_0)$. However, any QR decomposition may be used when referenced in the rest of this paper. A derivation for finding the differentiable Q and R at $\lambda = \lambda_0$ is given in Appendix A.

2.2 Rank revealing

As we commented above, the role of the column pivoting is to make sure the last diagonal entry $r_{nn}(\lambda)$ of $R(\lambda)$ becomes zero before any other diagonal entry does as well as that[2] $r_{nn}(\lambda)$ has a comparable magnitude to the smallest singular value of $R(\lambda)$. This purpose can also be fulfilled by using any of the rank revealing techniques in [3, 4, 6, 9, 11].

In what follows, we first explain the general idea of modifying Kublanovskaya's method with the use of a rank revealing technique, and then how to efficiently adapt the idea to banded $H(\lambda)$.

The theoretical foundation of rank revealing is the following well-known theorem. A proof is given for completeness.

Theorem 2.1. *Let $A \in \mathbb{C}^{n\times n}$, $x \in \mathbb{C}^n$, $\|x\|_2 = 1$,*

$$k = \operatorname*{argmax}_i |x_{(i)}|, \tag{2.10}$$

$\Pi \in \mathbb{R}^{n\times n}$ be a permutation matrix such that $\Pi e_n = e_k$, and $A\Pi = QR$ be the QR decomposition of $A\Pi$. If $\|Ax\|_2 = \epsilon$, then $|R_{(n,n)}| \le \sqrt{n}\,\epsilon$.

Proof. Notice that $|x_{(k)}| = \max_i |x_{(i)}|$ and $\|x\|_2 = 1$ imply $|x_{(k)}| \ge 1/\sqrt{n}$. We have

$$\epsilon = \|Ax\|_2 = \|A\Pi\Pi^{\mathrm{T}}x\|_2 = \|QR\Pi^{\mathrm{T}}x\|_2 = \|R\Pi^{\mathrm{T}}x\|_2 \ge |R_{(n,n)}x_{(k)}| \ge |R_{(n,n)}|/\sqrt{n},$$

as expected. □

How tiny can ϵ get in this theorem? It is known that $\epsilon \ge \sigma_{\min}(A)$ always and $\epsilon = \sigma_{\min}(A)$ if and only if x is A's (right) singular vector corresponding to $\sigma_{\min}(A)$. Also in this theorem, there is no requirement as to how the rest of the columns of A are permuted, except its kth column has to be permuted to the last! This degree of freedom will become handy to deal with $H(\lambda)$ with special structures, e.g, being banded.

Now return to Kublanovskaya's method. At λ_0, we first compute the QR decomposition without any column pivoting [5, 8, 19]

$$H(\lambda_0) = \widetilde{Q}_0\widetilde{R}_0. \tag{2.11}$$

Next we seek $x \in \mathbb{C}^n$ and $\|x\|_2 = 1$ such that $\|\widetilde{R}_0 x\|_2$ is comparable to $\sigma_{\min}(\widetilde{R}_0)$. This can usually be accomplished by a few steps of inverse iteration on $\widetilde{R}_0^*\widetilde{R}_0$: pick an initial $x_0 \in \mathbb{C}^n$ and normalize[3] it to have $\|x_0\|_2 = 1$, and then perform

$$\widetilde{R}_0^* y = x_i, \quad \widetilde{R}_0 z = y, \quad x_{i+1} = z/\|z\|_2 \quad \text{for } i = 0, 1, \ldots \tag{2.12}$$

Usually no more than 5 iterations are needed in our numerical examples. Each step of (2.12) is very cheap. How do we choose x_0? An obvious one is a random vector or the vector of all ones. This should be good, but as convergence begins to emerge, a better choice will be the x from the previous Newton iteration step.

Let $x = x_i$ be the last x_i from the iteration process (2.12), and then k as in (2.10), and let

$$\Pi = \big[e_1, \ldots, e_{k-1}, e_{k+1}, \ldots, e_n, e_k\big] \in \mathbb{R}^{n\times n},$$

[2] This is not always guaranteed. One example is the Kahan matrix [14, p.791].

[3] For computational efficiency, x_i can be scaled to $\|x_i\|_\infty = 1$ because all we need is the index of its largest entry in magnitude, where $\|\cdot\|_\infty$ is the ℓ_∞ vector norm.

i.e., Π will move the kth column of $\widetilde{R}_0$ in (2.13) to the last while shifting its $(k+1)$st to nth columns forward each by 1. Compute the QR decomposition

$$\widetilde{R}_0\Pi = \widehat{Q}_0 R_0, \tag{2.13}$$

and finally

$$H(\lambda_0)\Pi = \widetilde{Q}_0\widetilde{R}_0\Pi = (\widetilde{Q}_0\widehat{Q}_0)R_0 \equiv Q_0 R_0, \tag{2.14}$$

for which the last diagonal entry $r_{nn}^{(0)}$ of R_0 is usually of comparable magnitude to $\sigma_{\min}(R_0)$.

3 Banded QR Decompositions and Rank Revealing

In general, computing the QR decomposition (2.4) costs $O(n^3)$ flops, even when $H(\lambda)$ has certain favorable structure, including being banded with very small (left and right) bandwidths. This occurs because the column pivoting can and will destroy the structure and make ensuing computations cost $O(n^3)$ flops, and that is too expensive. For an example of this phenomenon, we modify the loaded string problem [1, 12] to give the following NLEVP:

$$H(\lambda)x = \big(A - \lambda B + e^{-\lambda}D\big)x = 0, \tag{3.1}$$

where $h = 1/n$ and

$$A = \frac{1}{h}\begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & 2 & -1 \\ & & -1 & 1 \end{bmatrix}, \quad B = \frac{h}{6}\begin{bmatrix} 4 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & 4 & 1 \\ & & 1 & 2 \end{bmatrix}, \quad D = e_n e_n^*.$$

Figure 3.1 shows the sparsity pattern in the R-factors for the QR decompositions of $H(1)$ for $n = 100$. It is evident that the R-factor from the QR decomposition with column pivoting is sufficiently dense that the QR decomposition with column pivoting will require $O(n^3)$ flops to complete, whereas for the R-factor by the rank revealing QR decomposition in subsections 3.2 and 3.3 below, it costs only $O(n)$. That is where the computational savings come from.

3.1 Structurally banded QR decomposition

Write $H(\lambda) = \big(h_{ij}(\lambda)\big)$. If

$$h_{ij}(\lambda) \equiv 0 \quad \text{for } j < i - p \text{ or } j > i + q, \tag{3.2}$$

we say that $H(\lambda)$ is *banded* with *left bandwidth* p and *right bandwidth* q.

In principle, the following development is valid for either $p = n-1$ or $q = n-1$ or both. However, it is only in the case $p \ll n$ that it takes much less work than $O(n^3)$ flops to obtain the desired QR decomposition (2.14). In the case when $q \ll n$ but $p \not\ll n$, we should work with the transpose of $H(\lambda)$, instead.

For such $H(\lambda)$, the R-factor in its QR decomposition without any column pivoting is an upper triangular matrix with *right bandwidth* $\min\{n-1, p+q\}$, namely $\widetilde{R}_0$ in (2.11). But in general the upper triangular part as well as the p subdiagonals of Q_0 are possibly nonzero. This still makes the overall cost $O(n^3)$ if Q_0 is formed explicitly. Fortunately Q_0 does not have to be formed explicitly and it should not be, especially when $p \ll n$. In fact,

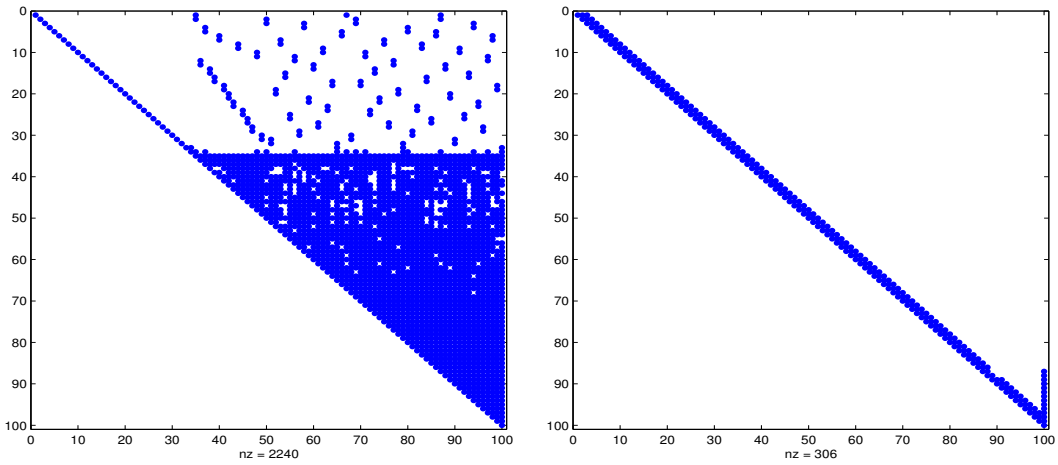

Figure 3.1: The sparsity pattern of the R-factors for the QR decompositions of $H(1)$ for the modified loaded string problem (3.1). Left: with standard column pivoting; Right: with the rank-revealing technique in subsection 3.3.

Q_0 can stay in its factored form as the product of either $n-1$ Householder transformations each of which takes $p+2$ floating point numbers of storage or $n-1$ Givens rotations each of which takes 2 floating point numbers of storage if $p=1$, see for examples [5, 8, 19].

Specifically, in (2.11) we have

$$\widetilde{Q}_0 = U_1^* U_2^* \cdots U_{n-1}^*, \quad U_i = \begin{bmatrix} I_{i-1} & & \\ & W_i & \\ & & I_{n-p-i} \end{bmatrix},$$

where I_j with $j \le 0$ is understood as an empty matrix, $W_i \in \mathbb{C}^{(p+1)\times(p+1)}$ is either a Householder matrix (when $p > 1$) or a Givens rotation (when $p = 1$). Each W_i can be recorded by no more than $p+2$ floating point numbers. In (2.13),

$$\widehat{Q}_0 = V_k^* V_{k+1}^* \cdots V_{n-1}^*, \quad V_i = \begin{bmatrix} I_{i-1} & & \\ & Z_i & \\ & & I_{n-i-1} \end{bmatrix},$$

where $Z_i \in \mathbb{C}^{2\times 2}$ is a Givens rotation. Finally (2.14) holds with

$$Q_0 = \widetilde{Q}_0 \widehat{Q}_0 = U_1^* U_2^* \cdots U_{n-1}^* V_k^* V_{k+1}^* \cdots V_{n-1}^*$$

which will be used to compute $Q_0 e_n$. The matrix R_0 in (2.13) is very well structured: its first $n-1$ columns are upper triangular with the right bandwidth $p+q$, and its last column has possible nonzero entries from the $\min\{k-p-q, 1\}$ position to the last position.

3.2 Unstructurally banded QR decomposition

In this subsection, we shall consider an unstructurally banded QR decomposition by which we mean $H(\lambda) = (h_{ij}(\lambda))$ in (1.1) characterized by two sets of integers

$$\partial_{\mathrm{U}} = \{\ell_1^{\mathrm{U}}, \ell_2^{\mathrm{U}}, \ldots, \ell_n^{\mathrm{U}}\} \quad \text{(upper column boundaries)}, \tag{3.3a}$$

$$\partial_{\mathrm{L}} = \{\ell_1^{\mathrm{L}}, \ell_2^{\mathrm{L}}, \ldots, \ell_n^{\mathrm{L}}\} \quad \text{(lower column boundaries)}, \tag{3.3b}$$

in such a way that

$$h_{ij}(\lambda) \equiv 0 \quad \text{for } i > \ell_j^{\text{L}} \text{ or } i < \ell_j^{\text{U}}. \tag{3.4}$$

For convenience, we will assume

$$\ell_j^{\text{U}} \le j \le \ell_j^{\text{L}}. \tag{3.5}$$

All (structurally) banded $H(\lambda)$ as defined in subsection 3 are examples of unstructurally banded matrices. For example, for tridiagonal $H(\lambda)$,

$$\partial_{\text{U}} = \{1, 1, 2, \ldots, n-1\}, \quad \partial_{\text{L}} = \{2, 3, \ldots, n, n\}.$$

On the other hand, an unstructurally banded $H(\lambda)$ with upper and lower column boundaries (3.3) is a (structurally) banded NLEVP with left bandwidth and right bandwidth given by

$$p = \max_j\{\ell_j^{\text{L}} - j\}, \quad q = \max_j\{j - \ell_j^{\text{U}}\}.$$

For ease of implementation, the third set of integers:

$$\partial_{\text{R}} = \{\ell_1^{\text{R}}, \ell_2^{\text{R}}, \ldots, \ell_n^{\text{R}}\} \quad \text{(right row boundaries)} \tag{3.6}$$

should also be made available, where ∂_{R} is defined by

$$h_{ij}(\lambda) \equiv 0 \quad \text{for } j > \ell_i^{\text{R}}. \tag{3.7}$$

The right row boundary can be computed from the upper column boundary ∂_{U} in (3.3a) as follows.

```
1  ℓ_i^R := i for i = 1 : n;
2  For j = 2 : n
3      ℓ_i^R := max{j, ℓ_i^R} for i = ℓ_j^U : j − 1;
4  EndFor.
```

In what follows, we will explain how to efficiently compute the QR decomposition of an unstructurally banded matrix with upper and lower column boundaries (3.3) and right row boundary (3.6).

Suppose that $T = (t_{ij}) \in \mathbb{C}^{n\times n}$ is unstructurally banded with upper column, lower column, and right row boundaries

$$\partial_{\text{U}} = \{\ell_1^{\text{U}}, \ell_2^{\text{U}}, \ldots, \ell_n^{\text{U}}\},\ \partial_{\text{L}} = \{\ell_1^{\text{L}}, \ell_2^{\text{L}}, \ldots, \ell_n^{\text{L}}\},\ \partial_{\text{R}} = \{\ell_1^{\text{R}}, \ell_2^{\text{R}}, \ldots, \ell_n^{\text{R}}\}, \tag{3.8}$$

i.e.,

$$t_{ij} = 0 \quad \text{for } i > \ell_j^{\text{L}} \text{ or } i < \ell_j^{\text{U}} \text{ or } j > \ell_i^{\text{R}}.$$

We are interested in efficient ways to compute T's QR decomposition: $T = QR$. The interesting case is when $\max_j\{\ell_j^{\text{L}} - \ell_j^{\text{U}}\} \ll n$, and thus T is very sparse.

As in the (structurally) banded case, the QR decomposition $T = QR$ can still be computed as

$$Q = U_1^* U_2^* \cdots U_{n-1}^*, \quad U_i = \begin{bmatrix} I_{i-1} & & \\ & W_i & \\ & & \ddots \end{bmatrix}, \tag{3.9}$$

where W_i is either the scalar 1 or a Householder transformation whose dimension will be determined at runtime[4] and can be seen to be no more than

$$\hat{m} = \max_j \{\ell_j^{\text{L}} - j\} + 1$$

at the beginning. At the same time, R is upper triangular (so its lower column boundary must be $\{1, 2, \ldots, n\}$) and unstructurally banded with its upper column boundary to be determined at runtime but with the guarantee that each column has no more than

$$m = \hat{m} + \max_j \{j - \ell_j^{\text{U}}\}$$

nontrivial consecutive entries starting at the diagonal entry upwards.

Each Householder transformation $W = I - 2uu^* \in \mathbb{C}^{i\times i}$ that transforms x to αe_1 can be represented by $i + 1$ complex values as:

$$u = \frac{x - \alpha e_1}{\|x - \alpha e_1\|_2}, \quad \text{if } x \neq \alpha e_1,$$

where α is chosen so that $|\alpha| = \|x\|_2$ (since W is unitary) and $|x_{(1)} - \alpha| = |x_{(1)}| + |\alpha|$ (to avoid possible cancellation in $x_{(1)} - \alpha$), and then

$$w = x - \alpha e_1, \quad \beta = \frac{2}{\|x - \alpha e_1\|_2^2} = \frac{1}{\|x\|_2(\|x\|_2 + |x_{(1)}|)}$$

to give $W = I - \beta ww^*$. If, however $x = \alpha e_1$ for some $\alpha \in \mathbb{C}$, including the cases $x = 0$ or x of dimension 1, simply taking $\beta = 0$ will do the job.

Based on the above discussion, we propose to store T into an $m \times n$ array T^{wk} to begin with in such a way that the nontrivial entries of T's jth column are stored as the first $\ell_j^{\text{L}} - \ell_j^{\text{U}} + 1$ entries of T^{wk}'s jth column:

$$T^{\text{wk}}_{(1:\ell_j^{\text{L}} - \ell_j^{\text{U}} + 1, j)} = T_{(\ell_j^{\text{U}}:\ell_j^{\text{L}}, j)}. \tag{3.10}$$

For an example of the sparse storage and indexing, consider the 5×5 matrix

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 4 & 0 & 0 \\ 0 & 3 & 5 & 7 & 0 \\ 0 & 0 & 6 & 8 & 9 \\ 0 & 0 & 0 & 0 & 10 \end{pmatrix}.$$

Then

$$\partial_{\text{U}} = \{1, 2, 2, 3, 4\}, \quad \partial_{\text{L}} = \{1, 3, 4, 4, 5\}, \quad \partial_{\text{R}} = \{1, 3, 4, 5, 5\},$$

and

$$T^{\text{wk}} = \begin{pmatrix} 1 & 2 & 4 & 7 & 9 \\ 0 & 3 & 5 & 8 & 10 \\ 0 & 0 & 6 & 0 & 0 \end{pmatrix}.$$

During the process of computing $T = QR$,

[4] If $p = 1$, then Givens rotations can be used.

1. $\partial_{\mathrm{U}} = \{\ell_1^{\mathrm{U}}, \ell_2^{\mathrm{U}}, \ldots, \ell_n^{\mathrm{U}}\}$ and $\partial_{\mathrm{R}} = \{\ell_1^{\mathrm{R}}, \ell_2^{\mathrm{R}}, \ldots, \ell_n^{\mathrm{R}}\}$ are updated to be the upper column and right row boundaries of the R-factor, respectively;

2. T^{wk} is updated to contain the R-factor in such a way

$$T^{\mathrm{wk}}_{(1:j-\ell_j^{\mathrm{U}}+1,j)} = R_{(\ell_j^{\mathrm{U}}:j,j)};$$

3. $\partial_{\mathrm{L}} = \{\ell_1^{\mathrm{L}}, \ell_2^{\mathrm{L}}, \ldots, \ell_n^{\mathrm{L}}\}$ is updated to yield the dimensions of the $n-1$ Householder transformations W_j in (3.9): $\ell_j^{\mathrm{L}} - j + 1$;

4. The Householder transformations W_j are compactly stored as an $(\hat{m}+1) \times (n-1)$ array Q^{wk} in such a way that

$$W_j = I_{\ell_j^{\mathrm{L}}-j+1} - \beta_j w_j w_j^*, \quad Q^{\mathrm{wk}}_{(1:\ell_j^{\mathrm{L}}-j+2,j)} = \begin{bmatrix} \beta_j \\ w_j \end{bmatrix}.$$

3.3 Rank-revealing of the R-factor

Suppose that $R = (r_{ij}) \in \mathbb{C}^{n\times n}$ is unstructurally banded and upper triangular, typically as the result of unstructurally banded QR decomposition, with upper column and right row boundaries

$$\partial_{\mathrm{U}} = \{\ell_1^{\mathrm{U}}, \ell_2^{\mathrm{U}}, \ldots, \ell_n^{\mathrm{U}}\},\ \partial_{\mathrm{R}} = \{\ell_1^{\mathrm{R}}, \ell_2^{\mathrm{R}}, \ldots, \ell_n^{\mathrm{R}}\}, \tag{3.11}$$

i.e.,

$$r_{ij} = 0 \quad \text{for } i < \ell_j^{\mathrm{U}} \text{ or } j > \ell_i^{\mathrm{R}}.$$

Since R is upper triangular, its lower column boundary is $\partial_{\mathrm{L}} = \{1, 2, \ldots, n\}$. Performing a rank-revealing QR decomposition as outlined in subsection 2.2 requires efficient ways to

1. solve linear systems $Rx = b$ and $R^*x = b$, and

2. compute the QR decomposition of the matrix $\left[R_{(:,1:k-1)}, R_{(:,k+1:n)}, R_{(:,k)}\right]$ permuted from R by moving its kth column to the last and shifting columns $k+1$ to n one to the left.

Assume that R is compactly stored as an $m \times n$ array R^{wk} in such a way that

$$R^{\mathrm{wk}}_{(1:j-\ell_j^{\mathrm{U}}+1,j)} = R_{(\ell_j^{\mathrm{U}}:j,j)}.$$

The triangular linear system $Rx = b$ can be solved by a column-oriented version of backward substitution [8, Section 3.1] to overwrite b in place:

```
0  Solve Rx = b.
1      For i = n : −1 : 2
2          b_(i) = b_(i) / R^wk_(i−ℓ_i^U+1,i);
3          b_(ℓ_i^U:i−1) = b_(ℓ_i^U:i−1) − b_(i) R^wk_(1:i−ℓ_i^U,i);
4      EndFor
5      b_(1) = b_(1) / R^wk_(1,1).
```
(3.12)

The key point is to update the relevant portion of b after each new entry of the solution is computed. The same idea is applicable to solving $R^*x = b$, except in this case, we use a row-oriented version of forward substitution. Here is an outline of the implementation:

$$
\begin{array}{ll}
0 & \textbf{\textit{Solve}}\ R^*x = b. \\
1 & \quad b_{(1)} = b_{(1)} / \operatorname{conj}(R^{\mathrm{wk}}_{(1,1)}); \\
2 & \quad \text{For } i = 2:n \\
3 & \quad\quad b_{(i)} = b_{(i)} - \operatorname{conj}(R^{\mathrm{wk}}_{(1:i-\ell^{\mathrm{U}}_i,i)})^{\mathrm{T}} b_{(\ell^{\mathrm{U}}_i:i-1)}; \\
4 & \quad\quad b_{(i)} = b_{(i)} / \operatorname{conj}(R^{\mathrm{wk}}_{(i-\ell^{\mathrm{U}}_i+1,i)}); \\
5 & \quad \text{EndFor}.
\end{array}
\tag{3.13}
$$

Next, we consider computing the QR decomposition of the matrix

$$
\left[R_{(:,1:k-1)}, R_{(:,k+1:n)}, R_{(:,k)}\right] = \widetilde{Q}\widetilde{R} \tag{3.14}
$$

for a given k, where R is stored as R^{wk}. As in the (structurally) banded case, this can be done by $n-k$ Givens rotations

$$
\widetilde{Q} = V_k^* V_{k+1}^* \cdots V_{n-1}^*, \quad V_i = \begin{bmatrix} I_{i-1} & & \\ & Z_i & \\ & & I_{n-i-1} \end{bmatrix}, \quad Z_j = \begin{bmatrix} c_j & s_j \\ -s_j^* & c_j \end{bmatrix}, \tag{3.15}
$$

where $c_j \in \mathbb{R}$ and $s_j \in \mathbb{C}$, $c_j^2 + |s_j|^2 = 1$. At its completion,

1. the first $n-1$ columns in R^{wk} compactly store the first $n-1$ columns of $\widetilde{R}$;
2. ℓ^{U}_j is updated to yield the upper column boundary for the first $n-1$ columns of $\widetilde{R}$;
3. $\widetilde{Q}$ is stored in the product form of $n-k$ Givens rotations in an array of $2 \times (n-k)$;
4. $r \in \mathbb{C}^n$ is the last column of $\widetilde{R}$.

4 Banded Kublanovskaya's Method

An application of Kublanovskaya's method (2.8) to the banded $H(\lambda)$ now critically relies on computing z and the last column of the Q-factor in the final QR decomposition computed as described in subsections 3.2 and 3.3.

The final R-factor as in subsection 3.3 is stored as two pieces – the first $n-1$ columns in R^{wk} and a vector r. Then z can be computed similarly to (3.12). The last column of the Q-factor is

$$
U_1^* U_2^* \cdots U_{n-1}^* V_k^* V_{k+1}^* \cdots V_{n-1}^* e_n,
$$

where U_i and V_j are from (3.9) and (3.15), respectively. Given U_i and V_j are compactly stored, this vector is easily and efficiently computed.

Algorithm 1 summarizes our variant of Kublanovskaya's method for an unstructurally banded nonlinear eigenvalue problem.

Algorithm 1 Nonlinear QR Algorithm for Unstructurally Banded NLEVP

Given unstructurally banded $H(\lambda)$ and its upper column boundary $\partial_{\text{U}}^{(0)}$, lower column boundary $\partial_{\text{L}}^{(0)}$, and (optionally) right row boundary $\partial_{\text{R}}^{(0)}$, initial guess $\lambda_0 \in \mathbb{C}$, and a relative tolerance `rtol`, this algorithm computes a nonlinear eigenvalue μ of $H(\lambda)$, and optionally (left and/or right) eigenvectors x and y as in (1.1).

for $i = 0, 1, \ldots$ **do**
  if $i > 0$ and $|r_{nn}(\lambda_i)|/\|H(\lambda_i)\|_{\text{F}} \leq$ `rtol` **then**
    break;
  else
    $\partial_{\text{U}} = \partial_{\text{U}}^{(0)}$, $\partial_{\text{L}} = \partial_{\text{L}}^{(0)}$, $\partial_{\text{R}} = \partial_{\text{R}}^{(0)}$;
    $T = H(\lambda_i)$ but compactly stored as T^{wk} to satisfy (3.10);
    compute $T = QR$ as outlined at the end of subsection 3.2;
    compute a rank-revealing QR of R as in subsection 3.3;
    compute λ_{i+1} according to (2.7) and (2.8), as outlined in section 4;
  end if
end for
(optionally) compute right and/or left eigenvectors x and y as by (2.9);
return λ_{i+1} as an approximate nonlinear eigenvalue to μ, and optionally x and/or y as approximate right and/or left eigenvectors.

4.1 Some convertible cases

There are several examples in the collection [2] that have banded $H(\lambda)$ to begin with. This section discusses how to convert $H(\lambda)$ of the following form

$$H(\lambda) = A + \sum_{i=1}^{k} \phi_i(\lambda) u_i v_i^* + \psi(\lambda) B, \tag{4.1}$$

into the banded case (including the possibility of the right bandwidth being $q = n - 1$) at $O(n^3)$ cost, where A, $B \in \mathbb{C}^{n\times n}$, $u_i, v_i \in \mathbb{C}^n$, and ϕ_i and ψ are scalar functions of λ. The cost $O(n^3)$ makes the conversions feasible for small to medium sized $H(\lambda)$ in the form of (4.1).

Case 1: $A = A^*$, $B = \alpha I_n$, and $u_i = v_i$ for $1 \leq i \leq k$. Possibly $\alpha = 0$, or equivalently the last term $\psi(\lambda)B$ in (4.1) drops out. First, we find $U_0 \in \mathbb{C}^{n\times n}$ as the product of k Householder transformations such that

$$U_0^* \big[u_1, u_2, \ldots, u_k\big] = \big[w_1, w_2, \ldots, w_k\big] \tag{4.2}$$

is upper triangular in the sense that $(w_i)_{(j)} = 0$ for all $j > i$. Next, similarly to the standard technique for reducing a Hermitian matrix to a real symmetric tri-diagonal matrix, we find $U_1 \in \mathbb{C}^{n\times n}$ as the product of $n - k$ Householder transformations such that

$$U_1^*(U_0^* A U_0) U_1 = T$$

is a banded Hermitian matrix with its left and right bandwidth k. Finally with $U = U_0 U_1$

$$U^* H(\lambda) U = T + \sum_{i=1}^{k} \phi_i(\lambda) w_i w_i^* + \alpha\psi(\lambda) I_n,$$

which is also banded with left and right bandwidth k.

It is possible that the rank ℓ of $\big[u_1, u_2, \ldots, u_k\big]$ is less than k. Then in (4.2), we will have $(w_i)_{(j)} = 0$ for all i and $j > \ell$. Consequently U_1 is now the product of $n - \ell$ Householder transformations, and finally $U^*H(\lambda)U$ has left and right bandwidth ℓ.

Case 2: $A = A^*$, $B = B^*$, $u_i = v_i$ for $1 \le i \le k$, and one of A and B is positive definite. Suppose B is positive definite and has the Cholesky factorization

$$B = LL^*.$$

Then

$$L^{-1}H(\lambda)L^{-*} = L^{-1}AL^{-*} + \sum_{i=1}^{k} \phi_i(\lambda) w_i w_i^* + \psi(\lambda) I_n$$

becomes **Case 1** above, where $Lw_i = u_i$.

Case 3: $B = \alpha I_n$. Now A is not necessarily Hermitian, and possibly $\alpha = 0$. First we find unitary $U_0 \in \mathbb{C}^{n\times n}$ such that

$$U_0^*\big[u_1, u_2, \ldots, u_k, v_1, v_2, \ldots, v_k\big] = \big[x_1, x_2, \ldots, x_k, y_1, y_2, \ldots, y_k\big] \tag{4.3}$$

with $(x_i)_{(j)} = (y_i)_{(j)} = 0$ for $1 \le i \le k$ and $j > \ell$, where ℓ is the (numerical) rank of $\big[u_1, u_2, \ldots, u_k, v_1, v_2, \ldots, v_k\big]$. Next, similarly to the standard technique for reducing a non-Hermitian matrix to an upper Hessenberg matrix, we find $U_1 \in \mathbb{C}^{n\times n}$ as the product of $n - \ell$ Householder transformations such that

$$U_1^*(U_0^*AU_0)U_1 = T$$

is banded with its left bandwidth ℓ and right bandwidth $n - 1$. Finally with $U = U_0U_1$

$$U^*H(\lambda)U = T + \sum_{i=1}^{k} \phi_i(\lambda) x_i y_i^* + \alpha\psi(\lambda) I_n,$$

which is also banded with its left bandwidth ℓ and right bandwidth $n - 1$.

4.2 Computation of several nearby nonlinear eigenvalues

We have presented a variation of Kublanovskaya's method with the goal of making it efficient for (structurally/unstructurally) banded nonlinear eigenvalue problems. Like the application of any Newton method, it starts at an initial guess μ_0 and ends at an approximate nonlinear eigenvalue if the process converges. Often all nonlinear eigenvalues nearby a particular point or within a region are sought. A straightforward way would be to repeat the Newton process at various initial points μ_0 near the point or the region. Sometimes starting at different initial points μ_0 does not lead to different approximate nonlinear eigenvalues, and in general it is hard, if not impossible, to know in advance how to pick initial points to avoid such repeated convergence to the same approximate nonlinear eigenvalue.

In what follows, we will explain a method to deflate out any known nearby nonlinear eigenvalues as in [24, Section 7.48]. Consider solving $f(t) = 0$ for t near a given point $t_0 \in \mathbb{C}$, where $f(t)$ is a nonlinear scalar function and, for our purpose, differentiable.

Suppose we know or have computed m zeros r_i $(i = 1 : m)$ of $f(t)$ near t_0, each of multiplicity m_i

$$f(r_i) = 0.$$

Usually $m_i = 1$. The simple idea is just to apply the underlying root finder to

$$g(t) := \frac{f(t)}{\prod_{i=1}^{m}(t - r_i)^{m_i}}. \tag{4.4}$$

In the case of Newton's method

$$\frac{g(t)}{g'(t)} = \frac{f(t)}{f'(t) - f(t)\sum_{i=1}^{m} \frac{m_i}{t-r_i}}. \tag{4.5}$$

So for our nonlinear QR algorithm, it is simple to modify (2.8) to

$$\lambda_1 = \lambda_0 - \frac{r_{nn}(\lambda_0)}{r'_{nn}(\lambda_0) - r_{nn}(\lambda_0)\sum_{i=1}^{m} \frac{m_i}{\lambda_0-\mu_i}}, \tag{4.6}$$

where μ_i for $1 \le i \le m$ are known or already computed zeros with multiplicity m_i, respectively, although usually $m_i = 1$. This idea works rather well in our numerical examples later.

5 Numerical Examples

Our variant of Kublanovskaya's method in combination with the unstructurally banded data storage format is indices intensive. It is hard to take advantage of MATLAB's built-in pre-compiled functions, and thus not surprisingly the algorithm runs slowly if coded in MATLAB for n in the thousands or larger because MATLAB is an interpreted language. So, we also provide an implementation in the C++ programming language. Both implementations may be found at [7].

Example 5.1. This is a problem from a finite element model of a vibrating structure with nonproportional damping [23]. The λ-matrix $H(\lambda)$ takes the form

$$H(\lambda) = \lambda^2 M + K - \frac{1}{1 + \beta\lambda}\Delta K,$$

where K, M, and ΔK are 9376×9376 real matrices and the parameter β is set to be $\beta = 2 \times 10^{-5}$.

The matrix $H(\lambda)$ is not an unstructurally banded matrix function to begin with, but it becomes one after a symmetric row and column permutation computed by MATLAB's `symrcm` as `perm=symrcm(A)`, where $A = |K| + |M| + |\Delta K|$ and the absolute value of a matrix is understood in the entrywise sense[5]. Figure 5.1 shows plots of the sparsity patterns of these matrices before and after the permutation. With the permutation, $H(\lambda)$ becomes unstructurally banded with maximum left and right bandwidths $p = q = 212$, but the bandwidths for most rows are much smaller than 212.

In Table 5.1, we show data for finding one nonlinear eigenvalue starting with an initial guess of $\lambda_0 = -1 - 10i$. The table shows at each iteration:

[5]We actually used Octave's implementation of `symrcm`, which gives a different permutation than the MATLAB implementation.

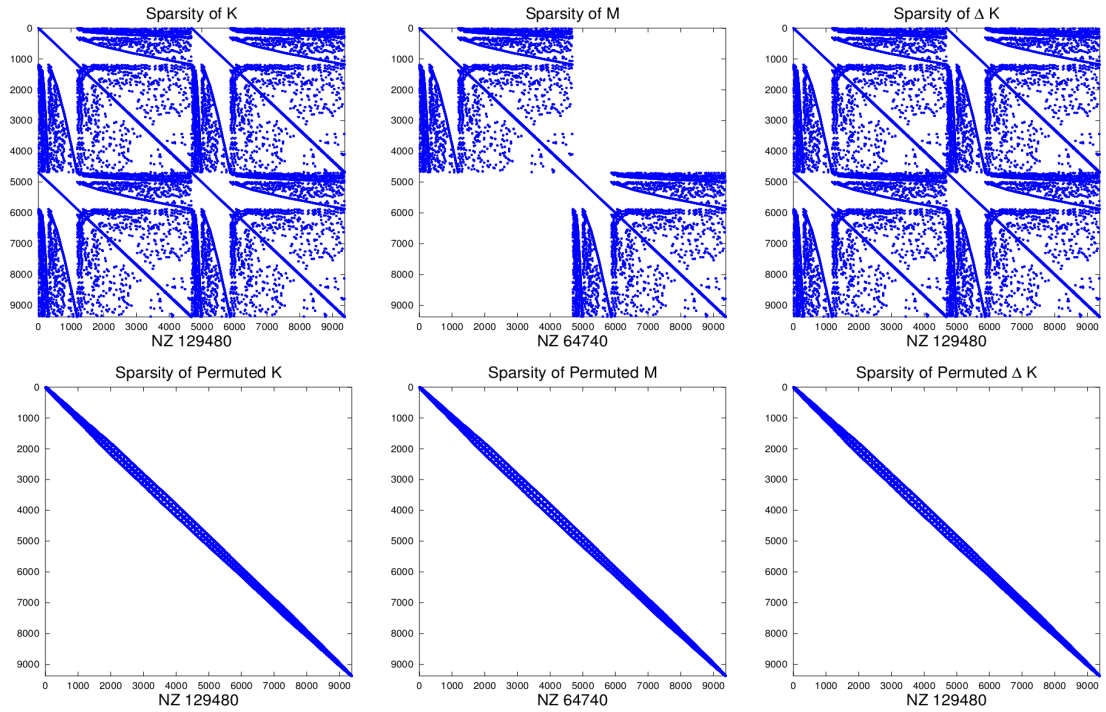

Figure 5.1: Sparsity patterns of K, M, and ΔK before and after the permutation for Example 5.1. Top row: before permutation; Bottom row: after permutation. NZ below each figure stands for the number of non-zero elements in the matrix.

- permuting iterations: the number of iterations used to compute the column index, k as in (2.10), to be permuted,
- permuted column: the column index, k as in (2.10), of the column to be permuted to the end of the matrix,
- residual: $|r_{nn}(\lambda_i)|/\|H(\lambda_i)\|_{\mathrm{F}}$, where λ_i is the ith iteration's approximation of the nonlinear eigenvalue,
- eigenvalue: the nonlinear eigenvalue approximation of the ith iteration.

The final left and right normalized residuals $\|y^*H(\lambda)\|_2/\|H(\lambda)\|_{\mathrm{F}}$ and $\|H(\mu)x\|_2/\|H(\mu)\|_{\mathrm{F}}$ were `8.5e-18` and `5.4e-17`, respectively.

Our current implementation attempts to solve for a nonlinear eigenvalue within a preset maximum number of iterations (default is 10), and if successful, it deflates the computed nonlinear eigenvalue as described in subsection 4.2 and continues to seek the next nonlinear eigenvalue with the same initial guess. The process continues until it fails to find a nonlinear eigenvalue within the preset maximum number of iterations. With the initial guess `-1408 - 21493i` (chosen because it is close to the nonlinear eigenvalue previously computed), our code is able to find five nonlinear eigenvalues before the maximum number of iterations is surpassed in searching for the sixth nonlinear eigenvalue. Figure 5.2 shows the residuals for each iteration of the five computed nonlinear eigenvalues. The final left and right normalized residuals for each computed nonlinear eigenvalue, as well as each computed nonlinear eigenvalue, are shown in Table 5.2.

iteration	permuting iterations	permuted column	residual	eigenvalue
0	2	7	1.1e-05	-1.000000e0 - i1.000000e1
1	3	2455	8.9e-05	-1.578073e3 - i2.145839e4
2	2	2787	2.4e-05	-1.374332e3 - i2.143317e4
3	2	2787	2.1e-05	-1.410348e3 - i2.153883e4
4	1	2787	1.5e-06	-1.405421e3 - i2.149456e4
5	1	2787	1.4e-08	-1.408698e3 - i2.149333e4
6	1	2787	1.0e-12	-1.408685e3 - i2.149336e4
7	1	2787	1.5e-17	-1.408685e3 - i2.149336e4

Table 5.1: Data for Example 5.1 with an initial guess of $\lambda_0 = -1 - 10i$.

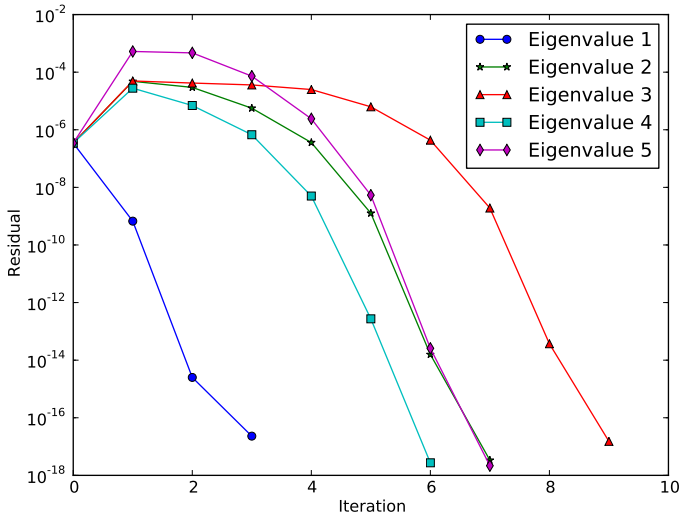

Figure 5.2: Figure containing the residuals at each iteration and each nonlinear eigenvalue for Example 5.1 with an initial guess of $\lambda_0 =$ -1408.0 - i21493. The different lines represent the different nonlinear eigenvalues.

index	left residual	right residual	eigenvalue
1	8.4e-18	5.5e-17	-1.408685e3 - i2.149336e4
2	8.4e-18	5.2e-17	-1.337498e3 - i2.122305e4
3	8.5e-18	5.7e-17	-1.304014e3 - i2.175030e4
4	8.2e-18	5.0e-17	-1.330369e3 - i2.043782e4
5	8.3e-18	4.2e-17	-4.711966e3 - i4.539375e4

Table 5.2: Left and right normalized residuals with the corresponding computed nonlinear eigenvalue for Example 5.1 with an initial guess of $\lambda_0 =$ `-1408 - 21493i`.

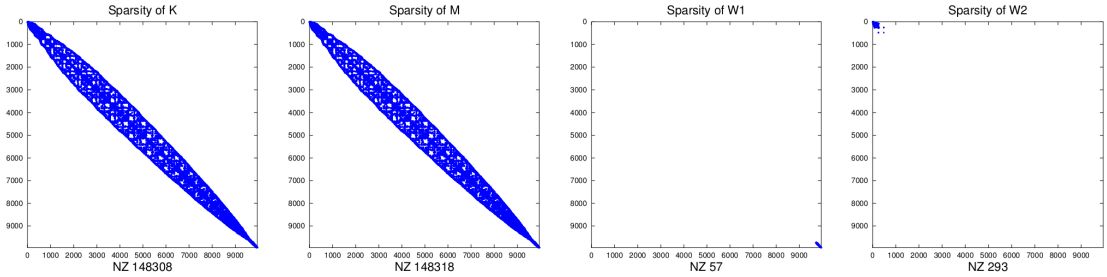

Figure 5.3: Sparsity patterns of K, M, W_1, and W_2 for Example 5.2. NZ below each figure stands for the number of non-zero elements in the matrix.

Remark 5.1. In this example, raising the maximum number of iterations does not yield more nonlinear eigenvalues. However, after reaching the maximum number of iterations, we tried new starting guesses by randomly choosing a point in

$$[\mathrm{Re}(\lambda_0) - 1000, \mathrm{Re}(\lambda_0) + 1000] + i[\mathrm{Im}(\lambda_0) - 1000, \mathrm{Im}(\lambda_0) + 1000].$$

Using this trick, we were able to find more nonlinear eigenvalues.

Example 5.2. For this example, we use the *gun* problem taken from the NLEVP collection [2], which models a radio-frequency gun cavity. This problem is the biggest banded problem currently available in the collection. The problem takes the form

$$H(\lambda) = K - \lambda M + i(\lambda - \sigma_1^2)^{1/2} W_1 + i(\lambda - \sigma_2^2)^{1/2} W_2,$$

where K, M, W_1, and W_2 are real symmetric 9956×9956 matrices. This $H(\lambda)$ is already an unstructurally banded matrix function, so no permutations need be done as in Example 5.1. Figure 5.3 shows plots of the sparsity patterns of these matrices. The matrices have bandwidths within $p = q = 843$, but the bandwidths for most rows are much smaller than this. This is especially true for W_1 and W_2 which have only 57 and 293 nonzero elements respectively.

When running this example, we set $\sigma_1 =$ `0.0` and $\sigma_2 =$ `108.8774`. Setting the initial condition, λ_0, anywhere between 20,000 and 25,000 leads to convergence of the nonlinear eigenvalue `2.234512e4 + i6.449986e-1`. Solving for another nonlinear eigenvalue leads to the program trying to find a nonlinear eigenvalue near zero, which makes the Newton steps fail since $H'(\lambda)$ becomes unbounded as $\lambda \to 0$. In Figure 5.4, we show the residuals at each iteration given various initial conditions.

Finally, to show that the unstructurally banded algorithm is much faster than the dense Kublanovskaya algorithm, we show timings for both algorithms run on a personal

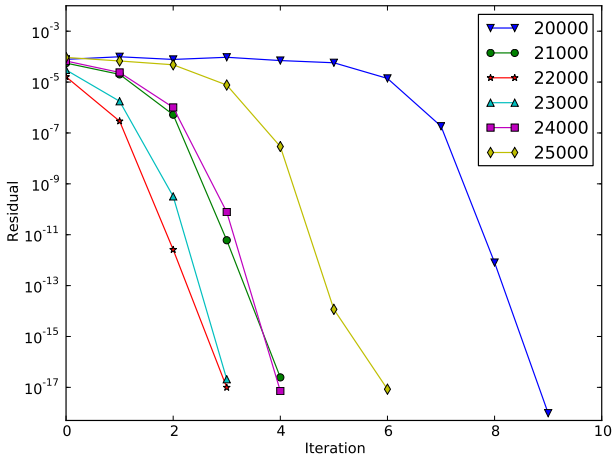

Figure 5.4: Figure containing the residuals at each iteration for computing the nonlinear eigenvalue `2.234512e4 + i6.449986e-1` in Example 5.2. The different lines represent the different initial guesses.

Example	Initial Guess	Algorithm	Time (seconds)
5.1	`-1408 - 21493i`	Sparse	20s
5.1	`-1408 - 21493i`	Dense	1411s
5.2	`22000`	Sparse	283s
5.2	`22000`	Dense	1665s

Table 5.3: Timings for finding one nonlinear eigenvalue for each example. Sparse indicates using the unstructurally banded algorithm and dense indicates using the dense matrix algebra with the Kublanovskaya method.

workstation. The dense algorithm we use is basically the original Kublanovskaya method without exploiting any unstructurally banded structure. The QR decomposition with column pivoting uses LAPACK's `zgeqp3` which calculates the QR decomposition of a matrix with column pivoting such that the diagonal elements of the upper triangular matrix are decreasing in absolute value. Table 5.3 has the timings for computing one nonlinear eigenvalue for both examples using both the sparse and dense algorithms. Clearly the dense version of the algorithm takes much longer to compute. Also, Example 5.1 takes significantly less time than Example 5.2 for the sparse solver because the bandwidth is considerably smaller in Example 5.1.

Remark 5.2. We compared results for the *gun* problem using the method described in this paper against CORK and NLEIGS referenced in the introduction, since this problem is implemented as an example by those packages. Both packages converged to the nonlinear eigenvalue given previously, and both took less time for the computation even when compared to our C++ implementation. At least one reason for this is our code has not been tuned to take advantage of level 2 and 3 BLAS routines yet given the complication of the unstructured nature of the algorithm. The CORK and NLEIGS software packages implicitly take advantage of level 2 and 3 BLAS routines in MATLAB, since their algorithms are easily written in vectorized form.

6 Concluding Remarks

We have presented new methods based on Kublanovskaya's method for solving the nonlinear eigenvalue problem for banded problems. In particular, new data structures were presented for the unstructurally banded case. While the new data structures are more complicated than most sparse or banded matrix data structures, they enable more efficient use of memory. We also explained how to find several nearby nonlinear eigenvalues to already calculated ones. Finally, we presented two examples to show the capabilities of the new methods, and we have made the source code publicly available [7] to reproduce the results and solve other unstructurally banded nonlinear eigenvalue problems.

7 Acknowledgements

We are grateful to the anonymous referees for their careful review of the manuscript and valuable comments.

A Appendix on Differentiability

What follows is taken from [16] for deriving a $Q(\lambda)$ and $R(\lambda)$ used to find $r_{nn}(\lambda)$ in (2.6), where $Q(\lambda)$ and $R(\lambda)$ are differentiable at $\lambda = \lambda_0$ with $Q(\lambda_0) = Q_0$ and $R(\lambda_0) = R_0$.

Write $Q_0^* H(\lambda)\Pi = R_0 + E$, where

$$E = Q_0^* H'(\lambda_0)\Pi\,(\lambda - \lambda_0) + O(|\lambda - \lambda_0|^2).$$

Partition R_0 as in (2.4) and E conformably as

$$E = \begin{array}{c} \\ n-1 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} n-1 & 1 \end{array} \\ \begin{bmatrix} E_{11} & e_{12} \\ e_{21} & e_{nn} \end{bmatrix} \end{array}.$$

For sufficiently tiny $|\lambda - \lambda_0|$, $R_{11}^{(0)} + E_{11}$ is nonsingular. Assume that $|\lambda - \lambda_0|$ is sufficiently tiny, and let

$$p = e_{21}[R_{11}^{(0)} + E_{11}]^{-1}, \quad Q_1^* = \begin{bmatrix} (I + p^*p)^{-1/2} & 0 \\ 0 & (1 + pp^*)^{-1/2} \end{bmatrix} \begin{bmatrix} I & p^* \\ -p & 1 \end{bmatrix}.$$

We have

$$\begin{aligned} &Q_1^* Q_0^* H(\lambda)\Pi \\ &\quad = \begin{bmatrix} (I + p^*p)^{-1/2} & 0 \\ 0 & (1 + pp^*)^{-1/2} \end{bmatrix} \begin{bmatrix} R_{11}^{(0)} + E_{11} + p^* e_{21} & r_{12}^{(0)} + e_{12} + p^*(r_{nn}^{(0)} + e_{nn}) \\ 0 & r_{nn}^{(0)} + e_{nn} - p(r_{12}^{(0)} + e_{12}) \end{bmatrix} \\ &\quad =: \begin{bmatrix} M_{11}(\lambda) & m_{12}(\lambda) \\ 0 & r_{nn}(\lambda) \end{bmatrix}. \end{aligned}$$

The same procedure can now be used inductively on $M_{11}(\lambda)$ to find $Q(\lambda)$ such that $H(\lambda)\Pi = Q(\lambda)R(\lambda)$. However, it is not necessary to calculate any further as we already have $r_{nn}(\lambda)$ which is the one given in (2.6).

References

[1] T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder, and F. Tisseur, *NLEVP: A collection of nonlinear eigenvalue problems*, MIMS EPrint 2008.40, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, Apr. 2008.

[2] T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder, and F. Tisseur, *NLEVP: A collection of nonlinear eigenvalue problems*, ACM Trans. Math. Software, 39 (2013), pp. 7:1–7:28.

[3] T. Chan, *Rank-revealing QR factorizations*, Linear Algebra Appl., 88/89 (1987), pp. 67–82.

[4] S. Chandrasekaran and I. C. F. Ipsen, *On rank-revealing factorisations*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 592–622.

[5] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

[6] L. Foster, *Rank and null space calculations using matrix decomposition without column interchanges*, Linear Algebra Appl., 74 (1986), pp. 47–71.

[7] C. K. Garrett and R.-C. Li, *Unstructurally Banded Nonlinear Eigenvalue Solver Software*, `http://www.csm.ornl.gov/newsite/software.html`, (2013).

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, 3rd ed., 1996.

[9] M. Gu and S. C. Eisenstat, *An efficient algorithm for computing a strong rank revealing QR factorization*, SIAM J. Sci. Comput., 17 (1996), pp. 848–869.

[10] S. Güttel and R. Van Beeumen and K. Meerbergen and W. Michiels, *NLEIGS: A class of fully rational Krylov methods for nonlinear eigenvalue problems*, SIAM J. Sci. Comput., 36(6) (2014), pp. A2842–A2864.

[11] Y. P. Hong and C.-T. Pan, *Rank-revealing QR factorizations and the singular value decomposition*, Math. Comp., 58 (1992), pp. 213–232.

[12] X. Huang, Z. Bai, and Y. Su, *Nonlinear rank-one modification of the symmetric eigenvalue problem*, J. Comp. Math., 28 (2010), pp. 218–234.

[13] N. K. Jain and K. Singhal, *On Kublanovskaya's approach to the solution of the generalized latent value problems for functional λ-matrices*, SIAM J. Numer. Anal., 20 (1983), pp. 1062–1067.

[14] W. Kahan, *Numerical linear algebra*, Canad. Math. Bull., 9 (1966), pp. 757–801.

[15] V. N. Kublanovskaya, *On an approach to the solution of the generalized latent value problems for λ-matrices*, SIAM J. Numer. Anal., 7 (1970), pp. 532–537.

[16] R.-C. Li, *QR decomposition and nonlinear eigenvalue problems*, Math. Numer. Sinica, 11 (1989), pp. 374–385. In Chinese.

[17] ———, *Compute multiple nonlinear eigenvalues*, J. Comp. Math., 10 (1992), pp. 1–20.

[18] V. Mehrmann and H. Voss, *Nonlinear eigenvalue problems: A challenge for modern eigenvalue methods*, GAMM-Mitteilungen (GAMM-Reports), 27:121-152, 2004.

[19] G. W. Stewart, *Matrix Algorithms Vol. I: Basic Decompositions*, SIAM, Philadelphia, 1998.

[20] D. B. Szyld and F. Xue, *Local convergence analysis of several inexact Newton-type algorithms for general nonlinear eigenvalue problems*, Numer. Math. 123 (2013), pp. 333–362.

[21] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Rev., 43 (2001), pp. 235–386.

[22] R. Van Beeumen and K. Meerbergen and W. Michiels, *Compact rational Krylov methods for nonlinear eigenvalue problems*, SIAM. J. Matrix Anal. & Appl., 36(2) (2015), pp. 820–838.

[23] H. Voss, *An Arnoldi method for nonlinear eigenvalue problems*, BIT, 44 (2004), pp. 387–401.

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