INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING, *SERIES B*
Volume 1, Number 1, Pages 109–122

MULTIPLE NONLINEAR EIGENVALUES OF SMOOTH RANK-DEFICIENT MATRICES

ANDREW BINDER AND JORGE REBAZA

Abstract. A smooth block LU factorization, coupled with Newton's method, is used to compute multiple nonlinear eigenvalues of smooth rank-deficient matrix functions $A(\lambda)$. We provide conditions for such factorizations to exist and show that the algorithm for the computation of multiple nonlinear eigenvalues converges quadratically, and is more efficient than one using QR factorizations. A possible approach for cubic convergence is also discussed. Several numerical examples are given for general and random nonlinear matrix functions $A(\lambda)$.

Key words. Smooth factorizations, multiple nonlinear eigenvalues.

1. Introduction

The factorization of matrix functions $A(\lambda)$, where λ is a parameter in general complex, has received great attention due to its importance and applications in several areas including linear algebra, matrix computations and dynamical systems. Smooth factorizations in particular [3], where the factors are as smooth as $A(\lambda)$, are a central tool for the computation of heteroclinic and homoclinic orbits in dynamical systems [4], [8].

We say λ is a nonlinear eigenvalue of $A(\lambda)$ if it satisfies

$$A(\lambda)v = 0. \tag{1}$$

The vector $v \neq 0$ is called the corresponding nonlinear eigenvector. In the very special case when $A(\lambda) = B - \lambda I$, for some constant matrix B, then λ and v are just the ordinary (linear) eigenvalue and eigenvector respectively. Nonlinear eigenvalues come from a long list of applications including the dynamical analysis of structures, singularities in elastic materials, and acoustic emissions of high speed trains. For an extensive list of problems related to nonlinear eigenvalues, see [1].

There is reliable software (e.g. MATLAB) that computes such nonlinear eigenvalues for the case when $A(\lambda)$ is a polynomial function. Some algorithms for the cases where $A(\lambda)$ is a general, large, and sparse matrix function have also been developed [11]. One of the main ideas of these methods is to linearize the problem in order to apply linear tools (such as generalized Schur factorizations) to the linearized system and preserve the structure of the original problem.

We explore the idea of computing multiple nonlinear eigenvalues for general, rank-deficient, and smooth matrix functions $A(\lambda)$ via rank-revealing LU factorizations coupled with Newton's method. We assume that $A(\lambda)$ is relatively small and dense. We first introduce the necessary theory to make sure that such factorizations involve smooth factors; we then develop an algorithm, study its convergence properties, and explore a higher order of convergence. A similar approach was considered in [7] using the classical QR factorization, and we take care of comparing their

Received by the editors June 29, 2010 and, in revised form, September 20, 2010.
2000 *Mathematics Subject Classification.* 65F99, 35P30.
This research was supported in part by NSF Grant DMS–0552573.

corresponding efficiencies. A generalization of this QR approach to compute multiple nonlinear eigenvalues of matrix functions $A(\alpha, \lambda)$, where λ is the eigenvalue parameter, is considered in [2].

2. LU Factorization of a smooth nonsingular matrix

Lemma 1. *All full rank square matrices $A(\lambda) \in C^1$ with nonsingular leading principal submatrices have a unique $L(\lambda)U(\lambda)$ factorization, where $L(\lambda) \in C^1$ is unit lower triangular, and $U(\lambda) \in C^1$ is upper triangular.*

Proof. Let $A(0) = L(0)U(0)$, with $L(0)$ unit lower triangular and $U(0)$ upper triangular. If the sought factorization is feasible, let $A(\lambda) = L(\lambda)U(\lambda)$. Taking the derivative of $A(\lambda)$ and solving for $U'(\lambda)$, we get:

$$U'(\lambda) = L^{-1}(\lambda)A'(\lambda) - L^{-1}(\lambda)L'(\lambda)U(\lambda). \tag{2}$$

Since $U'(\lambda)$ must be upper triangular,

$$\begin{aligned}
(L^{-1}(\lambda)A'(\lambda))_{i,1} &= (L^{-1}(\lambda)L'(\lambda))_{i,1}U(\lambda)_{1,1}, \quad i \geq 2 \\
(L^{-1}(\lambda)A'(\lambda))_{i,2} &= (L^{-1}(\lambda)L'(\lambda))_{i,1}U(\lambda)_{1,2} + (L^{-1}(\lambda)L(\lambda))_{i,2}U(\lambda)_{2,2}, \quad i \geq 3. \\
&\vdots
\end{aligned}$$

This system of linear equations is solvable for $B(\lambda) = (L^{-1}(\lambda)L'(\lambda))_{i,j}$ such that $i > j$. Similarly, we have

$$L'(\lambda) = A'(\lambda)U^{-1}(\lambda) - L(\lambda)U'(\lambda)U^{-1}(\lambda). \tag{3}$$

Since $L'(\lambda)$ must be lower triangular,

$$\begin{aligned}
(A'(\lambda)U^{-1}(\lambda))_{1,i} &= L(\lambda)_{1,1}(U'(\lambda)U^{-1}(\lambda))_{1,i}, \quad i \geq 2 \\
(A'(\lambda)U^{-1}(\lambda))_{2,i} &= L(\lambda)_{2,1}(U'(\lambda)U^{-1}(\lambda))_{1,i} + L(\lambda)_{2,2}(U'(\lambda)U^{-1}(\lambda))_{2,i}, \quad i \geq 3. \\
&\vdots
\end{aligned}$$

This system of equations is solvable for $C(\lambda) = (U'(\lambda)U^{-1}(\lambda))_{i,j}$ such that $i < j$. The diagonal entries of $U(\lambda)$ are completely determined from $A(\lambda) = L(\lambda)U(\lambda)$. Therefore, the diagonal entries of $C(\lambda)$ depend smoothly on $A(\lambda)$ and $L(\lambda)$. Thus, the system (2), (3) with $A(0) = L(0)U(0)$ is uniquely solvable and provides the sought smooth factorization. □

3. LU factorization for a smooth rank-deficient matrix

Terminology. We say a matrix A has a block LU factorization when $A = LU$ and the matrices L and U satisfy: $L = \begin{bmatrix} L_{1,1} & 0 \\ L_{2,1} & I \end{bmatrix}$, $L_{1,1}$ is a unit lower triangular matrix, $U = \begin{bmatrix} U_{1,1} & U_{1,2} \\ 0 & U_{2,2} \end{bmatrix}$, and $U_{1,1}$ is upper triangular.

Theorem 2. *Let $A(\lambda)$ be an $n \times n$ C^1 matrix function on some $D \subset \mathbb{C}$ such that for some $\lambda_0 \in D$, $A(\lambda_0)$ has rank $n - m$, $m < n$. Assume there are permutation matrices P_1, P_2 such that $P_1A(\lambda_0)P_2$ has a block LU factorization $P_1A(\lambda_0)P_2 = L_0U_0$. Then, there is a neighborhood $N(\lambda_0) \subset D$ such that $P_1A(\lambda)P_2$ has a block LU factorization*

$$P_1A(\lambda)P_2 = L(\lambda)U(\lambda), \ \ \forall\, \lambda \in N(\lambda_0), \tag{4}$$

with $L(\lambda)$, $U(\lambda) \in C^1(D)$, satisfying that $L(\lambda_0) = L_0$ and $U(\lambda_0) = U_0$.

Proof. Write $A(\lambda)$ as

$$A(\lambda) = A(\lambda_0) + A'(\lambda_0)(\lambda - \lambda_0) + o(|\lambda - \lambda_0|^2) = A(\lambda_0) + R(\lambda), \tag{5}$$

where $R(\lambda) = A'(\lambda_0)(\lambda - \lambda_0) + o(|\lambda - \lambda_0|^2)$, and $R(\lambda_0) = 0$. Then,

$$L_0^{-1}P_1A(\lambda)P_2 = L_0^{-1}P_1A(\lambda_0)P_2 + L_0^{-1}P_1R(\lambda)P_2 = U_0 + R_1(\lambda), \tag{6}$$

where $R_1(\lambda) = L_0^{-1}P_1R(\lambda)P_2$. Partition the matrices:

$$U_0 = \begin{bmatrix} U_{1,1} & U_{1,2} \\ 0 & U_{2,2} \end{bmatrix}, \quad R_1(\lambda) = \begin{bmatrix} R_{1,1}(\lambda) & R_{1,2}(\lambda) \\ R_{2,1}(\lambda) & R_{2,2}(\lambda) \end{bmatrix}. \tag{7}$$

The blocks $U_{1,1}$ and $R_{1,1}(\lambda)$ have dimension $(n-m)\times(n-m)$. Define accordingly a smooth unit block triangular matrix

$$L_1^{-1}(\lambda) = \begin{bmatrix} L_{1,1}(\lambda) & 0 \\ L_{2,1}(\lambda) & I(\lambda) \end{bmatrix}. \tag{8}$$

Then,

$$\begin{aligned} L_1^{-1}(\lambda)L_0^{-1}P_1A(\lambda)P_2 &= L_1^{-1}(\lambda)(U_0 + R_1(\lambda)) \\ &= \begin{bmatrix} L_{1,1}(\lambda) & 0 \\ L_{2,1}(\lambda) & I(\lambda) \end{bmatrix} \begin{bmatrix} U_{1,1} + R_{1,1}(\lambda) & U_{1,2} + R_{1,2}(\lambda) \\ R_{2,1}(\lambda) & U_{2,2} + R_{2,2}(\lambda) \end{bmatrix} \\ &= \begin{bmatrix} C_{1,1}(\lambda) & C_{1,2}(\lambda) \\ C_{2,1}(\lambda) & C_{2,2}(\lambda) \end{bmatrix}, \end{aligned} \tag{9}$$

where

$$\begin{aligned} C_{1,1}(\lambda) &= L_{1,1}(\lambda)(U_{1,1} + R_{1,1}(\lambda)), \\ C_{1,2}(\lambda) &= L_{1,1}(\lambda)(U_{1,2} + R_{1,2}(\lambda)), \\ C_{2,1}(\lambda) &= L_{2,1}(\lambda)(U_{1,1} + R_{1,1}(\lambda)) + R_{2,1}(\lambda), \\ C_{2,2}(\lambda) &= L_{2,1}(\lambda)(U_{1,2} + R_{1,2}(\lambda)) + (U_{2,2} + R_{2,2}(\lambda)). \end{aligned} \tag{10}$$

We want to have $C_{2,1}(\lambda) = 0$. Since $U_{1,1}$ is invertible, for small $|\lambda - \lambda_0|$, $U_{1,1} + R_{1,1}(\lambda)$ is also invertible. Then, with the matrix function

$$L_{2,1}(\lambda) = -R_{2,1}(\lambda)(U_{1,1} + R_{1,1}(\lambda))^{-1}, \tag{11}$$

we get

$$L_1^{-1}(\lambda)L_0^{-1}P_1A(\lambda)P_2 = \begin{bmatrix} L_{1,1}(\lambda)(U_{1,1} + R_{1,1}(\lambda)) & C_{1,2}(\lambda) \\ 0 & C_{2,2}(\lambda) \end{bmatrix}. \tag{12}$$

Notice that the matrix $L_{1,1}(\lambda)(U_{1,1} + R_{1,1}(\lambda))$ is smooth and invertible in some neighborhood $N(\lambda_0)$. By Lemma 1, we have a smooth LU factorization

$$L_{1,1}(\lambda)(U_{1,1} + R_{1,1}(\lambda)) = \hat{L}_2(\lambda)\hat{U}_2(\lambda). \tag{13}$$

Expand $\hat{L}_2(\lambda)$ to an $n \times n$ matrix $L_2(\lambda) = \begin{bmatrix} \hat{L}_2(\lambda) & 0 \\ 0 & I \end{bmatrix}$, so that

$$\begin{aligned} L_2^{-1}(\lambda)L_1^{-1}(\lambda)L_0^{-1}P_1A(\lambda)P_2 &= \begin{bmatrix} \hat{L_2}^{-1}(\lambda) & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{L}_2(\lambda)\hat{U}_2(\lambda) & C_{1,2}(\lambda) \\ 0 & C_{2,2}(\lambda) \end{bmatrix} \\ &= \begin{bmatrix} \hat{U}_2(\lambda) & \hat{L_2}^{-1}(\lambda)C_{1,2}(\lambda) \\ 0 & C_{2,2}(\lambda) \end{bmatrix} = U(\lambda). \end{aligned}$$

Thus, $P_1A(\lambda)P_2$ has a C^1 block LU factorization, and $L(\lambda) = L_0L_1(\lambda)L_2(\lambda)$. □

Note: We say (4) is an LU factorization of $A(\lambda)$, with complete pivoting.

4. Algorithm for the Computation of a Multiple Nonlinear Eigenvalue

The next objective is to devise an algorithm to calculate a multiple nonlinear eigenvalue λ_* of a matrix function $A(\lambda) \in C^2$. Let $||\cdot||_F$ denote the Frobenius norm. For a matrix $A = [a_1 \cdots a_n]$, where a_i are the columns of the matrix, we define the vector

$$\text{col } A \equiv \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}. \tag{14}$$

Now, let $\lambda_* \in \mathbb{C}$ be a multiple nonlinear eigenvalue and rank $A(\lambda_*) = n - m$ $(1 \leq m < n)$. From [5], there exists a rank revealing LU factorization

$$P_1 A(\lambda_*) P_2 = L(\lambda_*) \begin{bmatrix} U_{1,1}^{(*)}(\lambda_*) & U_{1,2}^{(*)}(\lambda_*) \\ 0 & 0 \end{bmatrix}, \tag{15}$$

where rank $U_{1,1}^{(*)}(\lambda_*) = n - m =$ rank $A(\lambda_*)$. Suppose λ_0 is sufficiently close to λ_*, and let

$$P_1 A(\lambda_0) P_2 = L(\lambda_0) U(\lambda_0) \tag{16}$$

be a block LU factorization of $P_1 A(\lambda_0) P_2$. Following Theorem 2, we can get a smooth block LU factorization

$$P_1 A(\lambda) P_2 = L(\lambda) U(\lambda), \quad \text{for } \lambda \in N(\lambda_0), \tag{17}$$

and

$$U(\lambda) = \begin{bmatrix} \hat{U_2}(\lambda) & \hat{L_2}^{-1}(\lambda) C_{1,2}(\lambda) \\ 0 & C_{2,2}(\lambda) \end{bmatrix},$$

where

$$C_{2,2}(\lambda) = C_{2,2}(\lambda_0) + C'_{2,2}(\lambda_0)(\lambda - \lambda_0) + O(|\lambda - \lambda_0|^2). \tag{18}$$

Dropping the higher order terms, we have

$$C_{2,2}(\lambda) \approx C_{2,2}(\lambda_0) + C'_{2,2}(\lambda_0)(\lambda - \lambda_0). \tag{19}$$

Thus, the next iterate λ_1 is chosen so that

$$||C_{2,2}(\lambda_0) + C'_{2,2}(\lambda_0)(\lambda_1 - \lambda_0)||_F^2 = \min_{\lambda} ||C_{2,2}(\lambda_0) + C'_{2,2}(\lambda_0)(\lambda - \lambda_0)||_F^2. \tag{20}$$

We need to determine $C'_{2,2}(\lambda_0)$. Recall from Eq. (10) that

$$C_{2,2}(\lambda) = L_{2,1}(\lambda)(U_{1,2} + R_{1,2}(\lambda)) + (U_{2,2} + R_{2,2}(\lambda)). \tag{21}$$

Substituting Eq. (11) into the above equation, we get

$$C_{2,2}(\lambda) = -R_{2,1}(\lambda)(U_{1,1} + R_{1,1}(\lambda))^{-1}(U_{1,2} + R_{1,2}(\lambda)) + (U_{2,2} + R_{2,2}(\lambda)). \tag{22}$$

Again, dropping the higher order terms, we have

$$C_{2,2}(\lambda) \approx -R'_{2,1}(\lambda_0)(\lambda - \lambda_0)(U_{1,1})^{-1}(U_{1,2}) + (U_{2,2} + R'_{2,2}(\lambda_0)(\lambda - \lambda_0)). \tag{23}$$

But observing (6), we can write

$$\begin{aligned} C_{2,2}(\lambda) \quad \approx \quad & -(L_0^{-1} P_1 A'(\lambda_0) P_2)_{s \times (n-s)} (\lambda - \lambda_0)(U_{1,1})^{-1}(U_{1,2}) \\ & + U_{2,2} + (L_0^{-1} P_1 A'(\lambda_0) P_2)_{s \times s} (\lambda - \lambda_0), \end{aligned}$$

where the subindices denote the corresponding blocks being used. Then,

$$C'_{2,2}(\lambda) \approx (L_0^{-1} P_1 A'(\lambda_0) P_2)_{s \times s} - (L_0^{-1} P_1 A'(\lambda_0) P_2)_{s \times (n-s)} (U_{1,1})^{-1}(U_{1,2}). \tag{24}$$

Going back to (20) and defining

$$f(\lambda) = ||C_{2,2}(\lambda_0) + C'_{2,2}(\lambda_0)(\lambda - \lambda_0)||_F^2, \tag{25}$$

we have

$$f'(\lambda_0) = 2(\text{col } C'_{2,2}(\lambda_0))^H \cdot \text{col } C_{2,2}(\lambda_0), \qquad f''(\lambda_0) = 2||C'_{2,2}(\lambda_0)||_F^2. \tag{26}$$

Thus, a Newton iteration reads

$$\lambda_1 = \lambda_0 - \frac{(\text{col } C'_{2,2}(\lambda_0))^H \cdot \text{col } C_{2,2}(\lambda_0)}{||C'_{2,2}(\lambda_0)||_F^2}. \tag{27}$$

We have then the following algorithm:

Step 1: Given an initial approximation λ_0 to λ_*.

Step 2: Compute

$$A(\lambda_i) \text{ and } A'(\lambda_i),\ i = 0, 1, \cdots .$$

Step 3: Compute the LU factorization with complete column pivoting of $A(\lambda_i)$:

$$P_1 A(\lambda_i) P_2 = L(\lambda_i) U(\lambda_i) \tag{28}$$

where $L(\lambda_i)$ is block unit lower triangular and $U(\lambda_i)$ is block upper triangular.

Step 4: Compute

$$C'_{2,2}(\lambda_i) = (L_i^{-1} P_1 A'(\lambda_i) P_2)_{s\times s} - (L_i^{-1} P_1 A'(\lambda_i) P_2)_{s\times(n-s)} (U_{1,1}^{(i)^{-1}} U_{1,2}^{(i)}). \tag{29}$$

Step 5: Compute

$$\lambda_{i+1} = \lambda_i - \frac{(\text{col } C'_{2,2}(\lambda_i))^H \cdot \text{col } C_{2,2}(\lambda_i)}{||C'_{2,2}(\lambda_i)||_F^2}. \tag{30}$$

Step 6: If the desired accuracy is attained, stop the iteration. Otherwise, repeat steps 2-6.

Estimating the Numerical Rank. An important piece of information necessary to perform the algorithm is the rank of the matrix $A(\lambda_i)$. This rank can be approximated numerically using a rank revealing factorization. While the LU rank revealing factorization does not offer a perfect ordering of the diagonal entries of U (as a QR factorization would for the entries of R), it provides sufficient information to determine the numerical rank due to the property

$$\min_{1\le i\le n-m} |u_{ii}| \ \gg \max_{n-m+1\le i,j\le n} |u_{ij}|.$$

Thus, we can choose $\epsilon > 0$ and find the largest t such that

$$\frac{|u_{t+1,t+1}|}{|u_{1,1}|} \le \epsilon \le \frac{|u_{t,t}|}{|u_{1,1}|}. \tag{31}$$

Our tests using this rank method have proved that it is very consistent and accurate.

5. Convergence Analysis

We begin the convergence analysis of the preceding algorithm by proving some preliminary results. Ultimately, we will prove that the algorithm is locally quadratic convergent.

Theorem 3. *Let $A(\lambda) \in \mathbb{C}^{n\times n}$ have rank $n-m$, $1 \leq m < n$. Assume there exist two distinct block LU factorizations $A(\lambda) = L_1(\lambda)U_1(\lambda) = L_2(\lambda)U_2(\lambda)$. Then,*

$$L_1(\lambda) = L_2(\lambda)D(\lambda) \quad \textit{and} \quad U_1(\lambda) = D^{-1}(\lambda)U_2(\lambda) \tag{32}$$

where $D(\lambda)$ is an invertible, block diagonal matrix.

Proof. Define $U_i(\lambda)$ to be

$$U_i(\lambda) = \begin{bmatrix} U_{1,1}^{(i)}(\lambda) & U_{1,2}^{(i)}(\lambda) \\ 0 & U_{2,2}^{(i)}(\lambda) \end{bmatrix}, \quad i = 1, 2 \tag{33}$$

where $U_{1,1}^{(i)}(\lambda) \in \mathbb{C}^{n-m\times n-m}$ is invertible. Define

$$\hat{U}_1(\lambda) = \begin{bmatrix} U_{1,1}^{(1)}(\lambda) & U_{1,2}^{(1)}(\lambda) \\ 0 & U_{2,2}^{(1)}(\lambda) + \epsilon I \end{bmatrix}, \tag{34}$$

such that $U_{2,2}^{(1)}(\lambda) + \epsilon I$ is invertible, for some $\epsilon \in \mathbb{C}$. From the assumptions, we have

$$L_2^{-1}(\lambda)L_1(\lambda)U_1(\lambda) = U_2(\lambda). \tag{35}$$

Define

$$\hat{U}_2(\lambda) = L_2^{-1}(\lambda)L_1(\lambda)\hat{U}_1(\lambda) = \begin{bmatrix} U_{1,1}^{(2)}(\lambda) & U_{1,2}^{(2)}(\lambda) \\ 0 & \hat{U}_{2,2}^{(2)}(\lambda) \end{bmatrix}. \tag{36}$$

Then,

$$L_2^{-1}(\lambda)L_1(\lambda) = \hat{U}_2(\lambda)\hat{U}_1^{-1}(\lambda) =: D(\lambda). \tag{37}$$

As $D(\lambda)$ is the product of two invertible block lower triangular matrices as well as two invertible block upper triangular matrices, $D(\lambda)$ must be an invertible block diagonal matrix. From (35) and (37), the theorem is proved. □

Theorem 4. *Let $A(\lambda) \in C^2$, and let P_1, P_2 and $\widehat{P}_1$, $\widehat{P}_2$ be two sets of permutation matrices that result from two distinct LU factorizations with complete pivoting of $A(\lambda_*)$ with rank $A(\lambda_*) = n - m$. Then,*

$$||U'_{2,2}(\lambda_*)||_F^2 > 0 \iff ||\widehat{U}'_{2,2}(\lambda_*)||_F^2 > 0. \tag{38}$$

Proof. Assume by way of contradiction (and without loss of generality) that

$$||U'_{2,2}(\lambda_*)||_F^2 > 0 \quad \text{and} \quad ||\widehat{U}'_{2,2}(\lambda_*)||_F^2 = 0. \tag{39}$$

By assumption, we have

$$P_1A(\lambda_*)P_2 = L^*U^* \quad \text{and} \quad \widehat{P}_1A(\lambda_*)\widehat{P}_2 = \widehat{L}^*\widehat{U}^*, \tag{40}$$

with

$$U^* = \begin{bmatrix} U_{1,1}^* & U_{1,2}^* \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad \widehat{U}^* = \begin{bmatrix} \widehat{U}_{1,1}^* & \widehat{U}_{1,2}^* \\ 0 & 0 \end{bmatrix}. \tag{41}$$

By Theorem 2, there exist complete pivoting LU factorizations of $A(\lambda)$ in a neighborhood of λ_*:

$$P_1A(\lambda)P_2 = L(\lambda)U(\lambda) \quad \text{and} \quad \widehat{P}_1A(\lambda)\widehat{P}_2 = \widehat{L}(\lambda)\widehat{U}(\lambda). \tag{42}$$

For $|\lambda-\lambda_*|$ sufficiently small, these factorizations have the following properties:

$$\begin{aligned} &U_{1,1}(\lambda), \widehat{U}_{1,1}(\lambda) \text{ are invertible,} \\ &U_{2,2}(\lambda) = U'_{2,2}(\lambda_*)(\lambda-\lambda_*) + O(|\lambda-\lambda_*|^2), \quad \widehat{U}_{2,2}(\lambda) = O(|\lambda-\lambda_*|^2), \\ &U'_{2,2}(\lambda_*) \neq 0. \end{aligned} \tag{43}$$

From Theorem 3, we let

$$V(\lambda) = \begin{bmatrix} U_{1,1}(\lambda) & U_{1,2}(\lambda) \\ 0 & I^{(m)} \end{bmatrix}, \widehat{V}(\lambda) = \begin{bmatrix} \widehat{U}_{1,1}(\lambda) & \widehat{U}_{1,2}(\lambda) \\ 0 & I^{(m)} \end{bmatrix},$$
$$D(\lambda) = \begin{bmatrix} I^{(n-m)} & 0 \\ 0 & U_{2,2}(\lambda) \end{bmatrix}, \widehat{D}(\lambda) = \begin{bmatrix} I^{(n-m)} & 0 \\ 0 & \widehat{U}_{2,2}(\lambda) \end{bmatrix}.$$

Then, from (42), we have

$$\begin{aligned} & P_1 A(\lambda) P_2 = L(\lambda) D(\lambda) V(\lambda), \;\; \widehat{P}_1 A(\lambda) \widehat{P}_2 = \widehat{L}(\lambda)\widehat{D}(\lambda)\widehat{V}(\lambda) \\ \Rightarrow \quad & A(\lambda) = P_1^T L(\lambda) D(\lambda) V(\lambda) P_2^T, \;\; A(\lambda) = \widehat{P}_1^T \widehat{L}(\lambda)\widehat{D}(\lambda)\widehat{V}(\lambda)\widehat{P}_2^T \\ \Rightarrow \quad & P_1^T L(\lambda) D(\lambda) M(\lambda) = \widehat{P}_1^T \widehat{L}(\lambda)\widehat{D}(\lambda)\widehat{M}(\lambda), \end{aligned} \tag{44}$$

where $M(\lambda) = V(\lambda)P_2^T$ and $\widehat{M}(\lambda) = \widehat{V}(\lambda)\widehat{P}_2^T$. For $|\lambda-\lambda_*|$ sufficiently small,

$$D(\lambda)M(\lambda)\widehat{M}(\lambda)^{-1} = (P_1^T L(\lambda))^{-1} \widehat{P}_1^T \widehat{L}(\lambda)\widehat{D}(\lambda). \tag{45}$$

Let $N(\lambda) = M(\lambda)\widehat{M}(\lambda)^{-1}$ and $\widehat{N}(\lambda) = (P_1^T L(\lambda))^{-1}\widehat{P}_1^T\widehat{L}(\lambda)$, which we partition as

$$N(\lambda) = \begin{bmatrix} N_{1,1}(\lambda) & N_{1,2}(\lambda) \\ N_{2,1}(\lambda) & N_{2,2}(\lambda) \end{bmatrix}, \quad \widehat{N}(\lambda) = \begin{bmatrix} \widehat{N}_{1,1}(\lambda) & \widehat{N}_{1,2}(\lambda) \\ \widehat{N}_{2,1}(\lambda) & \widehat{N}_{2,2}(\lambda) \end{bmatrix}. \tag{46}$$

Eq. (45) in matrix block form becomes

$$\begin{bmatrix} N_{1,1}(\lambda) & N_{1,2}(\lambda) \\ U_{2,2}(\lambda)N_{2,1}(\lambda) & U_{2,2}(\lambda)N_{2,2}(\lambda) \end{bmatrix} = \begin{bmatrix} \widehat{N}_{1,1}(\lambda) & \widehat{N}_{1,2}(\lambda)\widehat{U}_{2,2}(\lambda) \\ \widehat{N}_{2,1}(\lambda) & \widehat{N}_{2,2}(\lambda)\widehat{U}_{2,2}(\lambda) \end{bmatrix}. \tag{47}$$

The properties of (43) imply that $\lim_{\lambda\to\lambda_*} U_{2,2}(\lambda) = \lim_{\lambda\to\lambda_*} \widehat{U}_{2,2}(\lambda) = 0$. Combining this fact with Eq. (47) gives

$$\begin{aligned} \lim_{\lambda\to\lambda_*} \widehat{N}_{2,1}(\lambda) &= \lim_{\lambda\to\lambda_*} U_{2,2}(\lambda)N_{2,1}(\lambda) = 0 \quad \text{and} \\ \lim_{\lambda\to\lambda_*} N_{1,2}(\lambda) &= \lim_{\lambda\to\lambda_*} \widehat{N}_{1,2}(\lambda)\widehat{U}_{2,2}(\lambda) = 0. \end{aligned}$$

From Eq. (47), $U_{2,2}(\lambda)N_{2,2}(\lambda) = \widehat{N}_{2,2}(\lambda)\widehat{U}_{2,2}(\lambda)$. Then,

$$\begin{aligned} & U_{2,2}(\lambda) = \widehat{N}_{2,2}(\lambda)\widehat{U}_{2,2}(\lambda)N_{2,2}(\lambda)^{-1} \\ \Rightarrow \quad & \frac{U_{2,2}(\lambda)}{\lambda-\lambda_*} = \widehat{N}_{2,2}(\lambda)\frac{\widehat{U}_{2,2}(\lambda)}{\lambda-\lambda_*}N_{2,2}(\lambda)^{-1}. \end{aligned}$$

Letting $\lambda \to \lambda_*$ gives

$$\begin{aligned} 0 \neq U'_{2,2}(\lambda_*) &= \lim_{\lambda\to\lambda_*} \frac{U_{2,2}(\lambda)}{\lambda-\lambda_*} \\ &= \lim_{\lambda\to\lambda_*} \widehat{N}_{2,2}(\lambda)\frac{\widehat{U}_{2,2}(\lambda)}{\lambda-\lambda_*}N_{2,2}(\lambda)^{-1} \\ &= \lim_{\lambda\to\lambda_*} \widehat{N}_{2,2}(\lambda)\frac{O(|\lambda-\lambda_*|^2)}{\lambda-\lambda_*}N_{2,2}(\lambda)^{-1} = 0. \end{aligned} \tag{48}$$

The above statement is a contradiction and completes the proof. □

Theorem 5. *Let $A(\lambda) \in C^2$. Let the permutation matrices $P_i^{(1)} = P_*^{(1)}$ and $P_i^{(2)} = P_*^{(2)}$ independently of i for λ_i sufficiently close to λ_*. If $||U'_{2,2}(\lambda_*)||_F^2 > 0$, then the algorithm from Section 4 is locally quadratically convergent.*

Proof.

$$U_{2,2}(\lambda) = U_{2,2}(\lambda_i) + U'_{2,2}(\lambda_i)(\lambda - \lambda_i) + O(|\lambda - \lambda_i|^2). \tag{49}$$

At the nonlinear eigenvalue λ_*,

$$\begin{aligned} & 0 = U_{2,2}(\lambda_i) + U'_{2,2}(\lambda_i)(\lambda_* - \lambda_i) + O(|\lambda_* - \lambda_i|^2) \\ \Rightarrow \quad & 0 = \text{col } U_{2,2}(\lambda_i) + \text{col } U'_{2,2}(\lambda_i)(\lambda_* - \lambda_i) + O(|\lambda_* - \lambda_i|^2). \end{aligned} \tag{50}$$

Then,

$$0 = [\text{col } U'_{2,2}(\lambda_i)]^H \text{col } U_{2,2}(\lambda_i) + ||U'_{2,2}(\lambda_i)||_F^2(\lambda_* - \lambda_i) + O(|\lambda_* - \lambda_i|^2). \tag{51}$$

From the given assumptions and Theorem 4, we know that $||U'_{2,2}(\lambda_*)||_F^2 > 0$ for any set of permutation matrices. We may choose a $k > 0$ small enough for λ_i sufficiently close to λ_* such that

$$||U'_{2,2}(\lambda_i)||_F^2 > k||U'_{2,2}(\lambda_*)||_F^2 > 0. \tag{52}$$

Thus,

$$0 = \frac{[\text{col}U'_{2,2}(\lambda_i)]^H \text{col}U_{2,2}(\lambda_i)}{||U'_{2,2}(\lambda_i)||_F^2} + (\lambda_* - \lambda_i) + O(|\lambda_* - \lambda_i|^2). \tag{53}$$

With the help of (30), this simplifies to

$$0 = -(\lambda_{i+1} - \lambda_i) + (\lambda_* - \lambda_i) + O(|\lambda_* - \lambda_i|^2). \tag{54}$$

Therefore,

$$\lambda_* - \lambda_{i+1} = O(|\lambda_* - \lambda_i|^2). \tag{55}$$

□

6. Numerical Tests

Numerical tests were performed to compare the efficiency of the LU algorithm versus the QR algorithm for computing multiple nonlinear eigenvalues of a smooth, nonlinear matrix function. The algorithms described in the previous section were used, including the numerical rank approximation. These algorithms were run using MATLAB.

The nonlinear matrices used in the experiments were of the form:

$$A(\lambda) = A_0 + A_1\lambda + A_2\lambda^2 + A_3 \sin(\lambda) + A_4 \cos(\lambda) + A_5 e^{\lambda}, \tag{56}$$

where the matrices $A_i,\ i = 0, 1, 2, 3, 4, 5$, are random and constant. Eigenvalues calculated were deemed acceptable if both algorithms converged to the same eigenvalue and $\det A(\lambda)$ at the eigenvalue approximation was sufficiently close to zero. The initial guess of the eigenvalue was determined by adding a constant perturbation to the calculated eigenvalue.

Tables 1, 2, and 3 display the time results for the LU and QR algorithms for various nonlinear matrices tested in more than 100 trials in each case. The letters Q, E, and S signify the combination of nonlinear matrices that were used in the tests. The letters represent

$$Q = A_0 + A_1\lambda + A_2\lambda^2, \quad S = A_3 \sin(\lambda) + A_4 \cos(\lambda), \quad E = A_5 e^{\lambda}. \tag{57}$$

As can be seen from the tables, the average LU algorithm performed several times faster than its QR counterpart in all instances. Also, the average running time for the LU algorithm was less than the minimum running time required by the QR algorithm in each case. Observe that the maximum time required by the LU algorithm was less than the minimum time required by the QR algorithm to compute the nonlinear eigenvalues for the 10×10 matrices and a couple of the 4×4 matrices.

TABLE 1. Time [ms] Comparison of 4 x 4 Algorithm Performance

Nonlinear Matrix	LU			QR			Ratio of Averages (QR / LU)
	Max	Average	Min	Max	Average	Min	
Q	3.937	1.121	0.666	9.946	5.021	3.096	4.478
Q, E	2.767	1.113	0.665	8.370	5.040	3.100	4.526
Q, S	3.531	1.137	0.668	9.455	5.027	3.122	4.421
Q, E, S	1.907	1.122	0.665	8.403	5.044	3.106	4.495

TABLE 2. Time [ms] Comparison of 10 x 10 Algorithm Performance

Nonlinear Matrix	LU			QR			Ratio of Averages (QR / LU)
	Max	Average	Min	Max	Average	Min	
Q	5.106	2.552	1.634	33.479	13.986	9.366	5.481
Q, E	5.778	2.570	1.651	38.528	13.620	9.365	5.299
Q, S	7.826	2.688	1.655	35.637	14.903	9.614	5.543
Q, E, S	4.533	2.568	1.634	25.445	14.626	9.483	5.555

TABLE 3. Time [ms] Comparison of 100 x 100 Algorithm Performance

Nonlinear Matrix	LU			QR			Ratio of Averages (QR / LU)
	Max	Average	Min	Max	Average	Min	
Q	708.512	221.234	131.962	3021.232	911.267	398.420	4.119
Q, E	590.827	217.775	127.710	2556.985	927.271	475.985	4.258
Q, S	705.574	258.276	132.617	3813.738	1058.852	430.225	4.100
Q, E, S	623.669	233.597	125.844	3673.734	950.936	497.605	4.071

Tables 4, 5, and 6 display the average number of iterations required for the LU and QR algorithms to converge to the nonlinear eigenvalue, the average time [ms]/iteration in the computation, and the QR/LU ratio of the average time [ms]/iteration. In all cases, the LU factorization required slightly more iterations on average to converge to the nonlinear eigenvalue than the QR decomposition. Tables 4, 5, and 6 demonstrate however that the LU algorithm is more time efficient per iteration than the QR algorithm.

7. Application

We will next apply the LU algorithm to a nonlinear eigenvalue problem with multiple eigenvalues that comes from the analysis of a vibrating railtrack resting

TABLE 4. Average Iteration Comparison of 4 x 4 Algorithm Performance

Nonlinear Matrix	LU		QR		Time [ms] Per Iteration Ratio
	Number	Time/Iteration [ms]	Number	Time/Iteration [ms]	
Q	4.73	.24	4.64	1.08	4.57
Q, E	4.77	.23	4.69	1.07	4.60
Q, S	4.83	.24	4.64	1.08	4.60
Q, E, S	4.83	.23	4.71	1.07	4.61

TABLE 5. Average Iteration Comparison of 10 x 10 Algorithm Performance

Nonlinear Matrix	LU		QR		Time [ms] Per Iteration Ratio
	Number	Time/Iteration [ms]	Number	Time/Iteration [ms]	
Q	4.47	.57	4.30	3.26	5.70
Q, E	4.48	.57	4.21	3.24	5.64
Q, S	4.65	.58	4.49	3.32	5.74
Q, E, S	4.51	.57	4.34	3.28	5.77

TABLE 6. Average Iteration Comparison of 100 x 100 Algorithm Performance

Nonlinear Matrix	LU		QR		Time [ms] Per Iteration Ratio
	Number	Time/Iteration [ms]	Number	Time/Iteration [ms]	
Q	3.22	68.72	3.10	293.76	4.27
Q, E	3.11	70.08	3.08	301.36	4.30
Q, S	3.44	75.07	3.22	329.08	4.38
Q, E, S	3.25	75.96	3.08	309.05	4.29

on sleepers (lateral supports under the track) given in [6]. The discretization of the problem leads to a quadratic eigenvalue problem of the form:

$$A(\lambda) = A_0 + A_1\lambda + A_2\lambda^2, \tag{58}$$

where $A_i,\ i = 0, 1, 2$, are constant matrices with dimension 10×10. The exact eigenvalues of the problem are known and can be determined with an explicit equation. One multiple eigenvalue of the system is $\lambda_* = -0.5729 - 0.6600i$, with an algebraic multiplicity of 2. Starting with a guess of $\lambda_0 = -1 - 0.75i$, the U matrix given by the algorithm after 5 iterations and approximately 0.00586 s is of the form

$$U = \begin{bmatrix} U_{1,1} & U_{1,2} \\ 0 & 0 \end{bmatrix} \tag{59}$$

where $U_{1,1}$ is an 8×8 matrix. The approximate eigenvalue given by the algorithm matches the known nonlinear eigenvalue within an error tolerance of 10^{-15}. The multiplicity of the nonlinear eigenvalue is determined from the algorithm directly by the dimension of the lower right 0 block. Both methods reveal that the nonlinear eigenvalue has an algebraic multiplicity of 2. The LU and QR algorithms both required 5 iterations to compute the nonlinear eigenvalue. The LU and QR algorithms required 0.00586 and 0.0199 seconds to perform the calculations respectively, confirming again the superiority of the LU factorization over the QR factorization in terms of execution time.

The LU and QR algorithms were also compared in an application examining the eigenvibrations of a loaded string from [1] and [9]. From the discretization of the problem of a vibrating string with a load of mass m attached by a spring of stiffness k, we find the equation

$$R(\lambda)x = \left(A - B\lambda + \frac{\lambda}{\lambda - \sigma}C\right)x = 0 \tag{60}$$

where $\sigma = \frac{k}{m}$. In the MATLAB tests, the matrices had dimensions of 100 x 100, and the constants σ and k were both set equal to 1. Five eigenvalues of the problem for these parameter values are provided in [9]. In particular, $\lambda_* = 4.482176546$ is an eigenvalue of this problem. The initial eigenvalue guess for both methods was $\lambda_0 = \lambda_* + (2 + 2i)$. The LU algorithm calculated the correct eigenvalue in 5 iterations in 148.91 ms. The QR algorithm calculated the correct eigenvalue in 4 iterations in 760.83 ms. The LU algorithm required one more iteration yet was 5 times faster than the QR algorithm. Once again, the LU algorithm has proven to be superior to the QR method in regards to the execution time.

Further tests using these algorithms were performed for larger square matrices up to dimensions of 1000 x 1000. For square matrices with dimensions greater than 100 x 100, the LU algorithm's advantage over the QR algorithm in terms of the average total running time and the average time per iteration seemed to increase. For example, the average time per iteration required in the few tests of 500 x 500 matrices for the LU and QR algorithms was about 10.79 s and 122.65 s respectively. The QR algorithm ran approximately 11.4 times longer than the LU algorithm per iteration. The average total running time measured in these trials saw a similar discrepancy. This increasing trend continued in the few tests performed for larger matrices up to a size of 1000 x 1000.

8. Cubic Convergence Algorithm

The algorithm presented in Section 5 is locally quadratically convergent near the nonlinear eigenvalue. This algorithm can be modified to allow for cubic convergence. This is possible by a Newton-Steffensen method as described in [10]. When solving for the roots of $f(x) = 0$, the Newton-Steffensen iterative formula is of the form

$$x_{n+1} = x_n - \frac{f^2(x_n)}{f'(x_n)(f(x_n) - f(x_n^*))}, \tag{61}$$

where

$$x_n^* = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{62}$$

Recall that

$$f'(\lambda_i) = 2(\operatorname{col} U'_{2,2}(\lambda_i))^H \cdot \operatorname{col} U_{2,2}(\lambda_i), \qquad f''(\lambda_i) = 2||U'_{2,2}(\lambda_i)||_F^2. \tag{63}$$

Therefore, after adapting Newton-Steffensen to our problem, the iterations to minimize such a norm reads

$$\lambda_{i+1} = \lambda_n - \frac{((\operatorname{col} U'_{2,2}(\lambda_i))^H \cdot \operatorname{col} U_{2,2}(\lambda_i))^2}{||U'_{2,2}(\lambda_i)||_F^2((\operatorname{col} U'_{2,2}(\lambda_i))^H \cdot \operatorname{col} U_{2,2}(\lambda_i) - (\operatorname{col} U'_{2,2}(\lambda_i^*))^H \cdot \operatorname{col} U_{2,2}(\lambda_i^*))}, \tag{64}$$

where

$$\lambda_i^* = \lambda_i - \frac{(\operatorname{col} U'_{2,2}(\lambda_i))^H \cdot \operatorname{col} U_{2,2}(\lambda_i)}{||U'_{2,2}(\lambda_i)||_F^2}. \tag{65}$$

Thus, if λ_* is a nonlinear eigenvalue of $A(\lambda) \in \mathbb{C}^{n\times n}$ with rank $A(\lambda_*) = n - m$, known in advance, the corresponding algorithm for cubic convergence is

Step 1: Given an initial approximation λ_0 to λ_*.

Step 2: Compute

$$A(\lambda_i) \text{ and } A'(\lambda_i),\ i = 0, 1, \cdots .$$

Step 3: Compute a block LU factorization with complete column pivoting of $A(\lambda_i)$:

$$P_1 A(\lambda_i) P_2 = L(\lambda_i) U(\lambda_i) \tag{66}$$

Step 4: Compute

$$U'_{2,2}(\lambda_i) = (L_i^{-1} P_1 A'(\lambda_i) P_2)_{m\times m} - (L_i^{-1} P_1 A'(\lambda_i) P_2)_{m\times(n-m)} (U_{1,1}^{(i)^{-1}} U_{1,2}^{(i)}). \tag{67}$$

Step 5: Compute the Newton method iteration

$$\lambda_i^* = \lambda_i - \frac{(\operatorname{col} U'_{2,2}(\lambda_i))^H \cdot \operatorname{col} U_{2,2}(\lambda_i)}{||U'_{2,2}(\lambda_i)||_F^2}. \tag{68}$$

Step 6: Compute

$$A(\lambda_i^*) \text{ and } A'(\lambda_i^*),\ i = 0, 1, \cdots .$$

Step 7: Compute a block LU decomposition with complete column pivoting of $A(\lambda_i^*)$:

$$P_1 A(\lambda_i^*) P_2 = L(\lambda_i^*) U(\lambda_i^*) \tag{69}$$

Step 8: Compute

$$U'_{2,2}(\lambda_i^*) = (L_i^{-1}(\lambda_i^*) P_1 A'(\lambda_i^*) P_2)_{m\times m} - (L_i^{-1}(\lambda_i^*) P_1 A'(\lambda_i^*) P_2)_{m\times(n-m)} (U_{1,1}^{(i)^{-1}}(\lambda_i^*) U_{1,2}^{(i)}(\lambda_i^*)). \tag{70}$$

Step 9: Compute the Newton-Steffensen iteration

$$\lambda_{i+1} = \lambda_i - \frac{((\operatorname{col} U'_{2,2}(\lambda_i))^H \cdot \operatorname{col} U_{2,2}(\lambda_i))^2}{||U'_{2,2}(\lambda_i)||_F^2 ((\operatorname{col} U'_{2,2}(\lambda_i))^H \cdot \operatorname{col} U_{2,2}(\lambda_i) - (\operatorname{col} U'_{2,2}(\lambda_i^*))^H \cdot \operatorname{col} U_{2,2}(\lambda_i^*))}. \tag{71}$$

Step 10: If the desired accuracy is attained, stop the iteration. Otherwise, repeat steps 2-10.

An obvious disadvantage of this algorithm is that it requires the computation of two LU factorizations per step versus the single factorization required in the algorithm that provides quadratic convergence. This will make each iteration in the process more expensive. A similar algorithm can be created using QR decomposition with the appropriate substitutions in the algorithm. Table 7 displays the time results for the cubic LU and QR algorithms for various nonlinear matrices tested in more than 100 trials in each case. As can be seen from the table, the average LU algorithm performed several times faster than its QR counterpart in all instances. For the 10×10 comparisons, the LU algorithm was approximately 5.3 times faster than the QR algorithm on average for each nonlinear matrix function. This ratio is slightly less than in the quadratic case. The maximum time required by the LU algorithm was in all cases less than the minimum time required by the QR algorithm to compute the nonlinear eigenvalues for the 10×10 matrices with cubic convergence.

TABLE 7. Time [ms] Comparison of 10 x 10 Cubic Convergence Algorithm Performance

Nonlinear Matrix	LU			QR			Ratio of Averages (QR / LU)
	Max	Average	Min	Max	Average	Min	
Q	10.630	4.013	2.333	41.275	21.344	13.036	5.319
Q, E	9.572	3.889	2.342	46.831	20.919	13.045	5.379
Q, S	7.346	4.000	2.377	47.283	20.898	13.044	5.224
Q, E, S	10.610	4.090	2.351	53.448	21.608	13.037	5.283

Table 8 displays the average number of iterations required for the LU and QR algorithms to converge to the nonlinear eigenvalue, the average time [ms]/iteration in the computation, and the QR/LU ratio of the average time [ms]/iteration. In all cases, the LU factorization required slightly more iterations on average to converge to the nonlinear eigenvalue than the QR decomposition. Table 8 demonstrates that the LU algorithm is more time efficient per iteration than the QR algorithm. Comparing Table 8 with Table 5, the data shows that the average number of iterations required to converge to the nonlinear eigenvalue in the cubic convergence algorithm decreased on average by slightly more than one. However, the time per iteration increased substantially. This change was expected as each iteration now requires two decompositions. Table 9 demonstrates the cubic convergence of the LU and QR algorithm for two random 10×10 matrices.

TABLE 8. Average Iteration Comparison of 10 x 10 Cubic Convergence Algorithm Performance

Nonlinear Matrix	LU		QR		Time [ms] Per Iteration Ratio
	Number	Time/Iteration [ms]	Number	Time/Iteration [ms]	
Q	3.26	1.230	3.12	6.832	5.56
Q, E	3.22	1.206	3.14	6.670	5.53
Q, S	3.28	1.218	3.14	6.651	5.46
Q, E, S	3.31	1.235	3.16	6.829	5.53

TABLE 9. Cubic Algorithm Error Convergence for a 10 x 10 Matrix Function

Iteration	LU Error	QR Error
0	0.7071	0.6403
1	0.3212	0.2253
2	0.1340	0.0938
3	$1.7791 * 10^{-4}$	$2.2825 * 10^{-4}$
4	$5.1172 * 10^{-13}$	$1.8925 * 10^{-12}$

9. Final Remarks

We have presented a complete pivoting LU factorization of smooth rank-deficient matrices as a more efficient alternative to the QR factorization to compute multiple nonlinear eigenvalues when the matrix is in general dense and relatively small. The given algorithm, which converges quadratically, can also be useful within iterative projection methods for large sparse problems. There are additional ways the current

algorithms could be modified to improve their effectiveness. As they are now, the algorithms do not take advantage of any structure that may be inherent in the problem. Modifying the algorithm to use properties such as symmetric structure would be very beneficial.

The cubic convergence algorithm for the general case has important areas where it could be improved. The cubic convergence algorithm is more expensive per iteration due to the necessity of the computation of two factorizations per iteration compared to the single decomposition required in the quadratic decomposition. If there were a way to approximate the second LU decomposition or just the components necessary to calculate the next iterative guess, the method could become much more efficient. One simple approach to approximating the $f'(x_n^*)$ component in the method is to use the approximation

$$\frac{f'(x_n^*) - f'(x_n)}{x_n^* - x_n} \approx f''(x_n). \tag{72}$$

This equation can be solved for $f'(x_n^*)$ to yield

$$f'(x_n^*) \approx f''(x_n)(x_n^* - x_n) + f'(x_n). \tag{73}$$

The advantage to using this equation would be that the second LU decomposition does not have to be calculated. An increased rate of convergence could be gained for minimal cost as all the values are already known. Of course, this approximation could only be applied when x_n^* is sufficiently close to x_n. Initial testing, however, has shown negligible improvement in the speed of the algorithm.

References

[1] T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder and F. Tisseur, NLEVP: A collection of nonlinear eigenvalue problems, MIMS Eprints 40 (2008).

[2] H. Dai and P. Lancaster, Numerical methods for finding multiple eigenvalues of matrices depending on parameters, Numerische Math., 76 (1997) 189–208.

[3] L. Dieci and T. Eirola, On smooth decompositions of matrices, SIAM J. Matrix Anal. Appl., 20 (1999) 800–819.

[4] E. J. Doedel, B.W. Kooi, Y. A. Kuznetzov, and G. A. Voorn, Continuation of connecting orbits in 3D ODEs, (I) Point to cycle connections, Int. J. Bifurc. & Chaos 18 (2008) 1889–1903.

[5] G. Golub and C. Van Loan, Matrix Computations, 3rd ed. Johns Hopkins University Press (1996).

[6] P. Lancaster and Rózsa, The spectrum and stability of a vibrating rail supported by sleepers, Computers Math. Applic., 31 (1996) 201-213.

[7] R. C. Li, Compute multiple nonlinear eigenvalues, J. Comp. Math., 10 (1992) 1–20.

[8] J. Rebaza, Smooth Schur factorizations in the continuation of separatrices, Lin. Alg. & its Applic., 421 (2007) 138–156.

[9] S. I. Solovëv, Preconditioned iterative methods for a class of nonlinear eigenvalue problems, Lin. Alg. & its Applic. 415 (2006) 210-229.

[10] J. R. Sharma, A composite third order Newton Steffensen method for solving nonlinear equations, Appl. Math and Comp., 169 (2005) 242-246.

[11] V. Mehrmann and H. Voss, Nonlinear eigenvalue problems: A challenge for modern eigenvalue methods, GAMM Mitt. Ges. Angew. Math. Mech. 27 (2004) 121–152.

Department of Mathematics, University of Arizona, Tucson, AZ 85721, USA
E-mail: abinder@email.arizona.edu

Department of Mathematics, Missouri State University, Springfield, MO 65897, USA
E-mail: jrebaza@missouristate.edu
URL: http://math.missouristate.edu/jrebaza.htm