mathematics

MDPI

Article

Fast Eigenvalue Decomposition of Arrowhead and Diagonal-Plus-Rank-*k* Matrices of Quaternions

Thaniporn Chaysri, Nevena Jakovčević Stor and Ivan Slapničar *

Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, Rudjera Boškovića 32, 21000 Split, Croatia; thaniporn.chaysri@fesb.hr (T.C.); nevena@fesb.hr (N.J.S.)
* Correspondence: ivan.slapnicar@fesb.hr

Abstract: Quaternions are a non-commutative associative number system that extends complex numbers, first described by Hamilton in 1843. We present algorithms for solving the eigenvalue problem for arrowhead and DPRk (diagonal-plus-rank-*k*) matrices of quaternions. The algorithms use the Rayleigh Quotient Iteration with double shifts (RQIds), Wielandt's deflation technique and the fact that each eigenvector can be computed in $O(n)$ operations. The algorithms require $O(n^2)$ floating-point operations, n being the order of the matrix. The algorithms are backward stable in the standard sense and compare well to the standard QR method in terms of speed and accuracy. The algorithms are elegantly implemented in Julia, using its polymorphism feature.

Keywords: eigenvalue decomposition; matrices of quaternions; arrowhead matrix; diagonal-plus-rank-*k* matrix

MSC: 65F15

Citation: Chaysri, T.; Jakovčević Stor, N.; Slapničar, I. Fast Eigenvalue Decomposition of Arrowhead and Diagonal-Plus-Rank-*k* Matrices of Quaternions. *Mathematics* **2024**, *12*, 1327. https://doi.org/10.3390/math12091327

Academic Editor: Luca Gemignani

Received: 21 February 2024
Revised: 19 April 2024
Accepted: 23 April 2024
Published: 26 April 2024

1. Introduction and Definitions

In linear algebra, there are various methods for computing eigenvalues and eigenvectors of real and complex matrices. When this problem extends to matrices of quaternions, many researchers attempt to solve it [1–8]. For the matrices of quaternions, the majority of classic methods need to be modified since quaternions are non-commutative systems and matrices of quaternions do not have the characteristic polynomial. In this paper, we use the Rayleigh Quotient Iteration with double shifts and a deflation method to solve the eigenvalue problem for the arrowhead and DPRk (diagonal-plus-rank-*k*) matrices of quaternions. Such matrices appear in many applications and they are parts of many significant linear algebra algorithms (see [9–14] for more information).

This paper is organized as follows. In Section 1, we give basic definitions and preliminary results. In Section 2, we give methods for the eigenvalue decomposition of matrices of quaternions. We first briefly describe the standard quaternion QR method for general matrices from [1]. We then describe our new method for arrowhead and DPRk matrices in detail. In Section 3, we analyze the error of our algorithms and show that they are backward stable. In Section 4, we give the numerical examples and compare our method with the standard quaternion QR method. We also present a discussion and conclusions in Section 5.

1.1. Quaternions

Quaternions are a non-commutative associative number system that extends complex numbers, introduced by Hamilton [15,16]. For *basic quaternions* $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$, the quaternions have the form

$$q = a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}, \quad a, b, c, d, \in \mathbb{R}.$$

The *multiplication table* of basic quaternions is the following:

$\times$	$\mathbf{i}$	$\mathbf{j}$	$\mathbf{k}$
$\mathbf{i}$	-1	$\mathbf{k}$	$-\mathbf{j}$
$\mathbf{j}$	$-\mathbf{k}$	-1	$\mathbf{i}$
$\mathbf{k}$	$\mathbf{j}$	$-\mathbf{i}$	-1

Conjugation is given by

$$\bar{q} = a - b\,\mathbf{i} - c\,\mathbf{j} - d\,\mathbf{k}.$$

Then,

$$\bar{q}q = q\bar{q} = |q|^2 = a^2 + b^2 + c^2 + d^2.$$

Also, the *real* and the *imaginary* parts of q are defined by

$$\text{real}(q) = \frac{q + \bar{q}}{2} = a, \quad \text{imag}(q) = \frac{q - \bar{q}}{2},$$

respectively.

The *dot product* of two vectors of quaternions $p, q \in \mathbb{H}^n$, is defined in the standard manner:

$$p \cdot q = \sum_{i=1}^{n} \bar{p}_i q_i.$$

Let $f(x)$ be a complex analytic function. The value $f(q)$, where $q \in \mathbb{H}$, is computed by evaluating the extension of f to the quaternions at q (see [17]). All of the above is implemented in the Julia [18] package Quaternions.jl [19]. Quaternions are homomorphic to $\mathbb{C}^{2\times 2}$:

$$q \to \begin{bmatrix} a + b\,\mathbf{i} & c + d\,\mathbf{i} \\ -c + d\,\mathbf{i} & a - b\,\mathbf{i} \end{bmatrix} \equiv C(q),$$

with eigenvalues q_s and $\bar{q}_s$.

1.2. Matrices of Quaternions

Let $\mathbb{H}$ be a set of quaternions. Any matrix $A = B_1 + B_2\mathbf{i} + B_3\mathbf{j} + B_4\mathbf{k} \in \mathbb{H}^{n\times n}$, where $B_i \in \mathbb{R}^{n\times n}$, $i = 1, 2, 3, 4$ can be uniquely expressed as

$$(B_1 + B_2\mathbf{i}) + (B_3 + B_4\mathbf{i})\mathbf{j} \equiv A_1 + A_2\mathbf{j}, \quad A_1, A_2 \in \mathbb{C}^{n\times n}.$$

Similarly, matrices of quaternions are homomorphic to $\mathbb{C}^{2n\times 2n}$:

$$A \to \begin{bmatrix} A_1 & A_2 \\ -\bar{A}_2 & \bar{A}_1 \end{bmatrix}.$$

Notice that

$$A_1 = \text{real}(A) - \text{real}(A\mathbf{i})\mathbf{i}, \quad A_2 = -\text{real}(A\mathbf{j}) - \text{real}(A\mathbf{k})\mathbf{i}.$$

The *right eigenvalues* and the *standard right eigenvalues* of the matrix A are defined by:

$$\Lambda(A) = \{\lambda \in \mathbb{H} : Ax = x\lambda \quad \text{for some } x \in \mathbb{H}^n,\ x \neq 0\},$$
$$\Lambda_s(A) = \{\lambda \in \mathbb{C} : Ax = x\lambda \quad \text{for some } x \in \mathbb{H}^n,\ x \neq 0,\ \text{imag}(\lambda) \geq 0\}.$$

Let (λ, x) be the right eigenpair of A where λ is the standard eigenvalue. Then, λ is invariant under similarity with complex numbers: if

$$Ax = x\lambda,$$

then

$$A(xa) = (xa)\frac{1}{a}\lambda a = (xa)\lambda,$$

for any non-zero $a \in \mathbb{C}$.

1.3. Arrowhead and Diagonal-Plus-Rank-k Matrices

Let A^* and x^* denote the adjoints of matrix A and vector x, respectively.

The *arrowhead matrix* (Arrow) is a matrix of the form

$$A = \begin{bmatrix} D & u \\ v^* & \alpha \end{bmatrix} \equiv \text{Arrow}(D, u, v, \alpha),$$

where D is a diagonal matrix with diagonal elements $d_i \equiv D_{ii} \in \mathbb{H}$, $\text{diag}(D), u, v \in \mathbb{H}^{n-1}$ and $\alpha \in \mathbb{H}$ or any symmetric permutation of such matrix.

The *diagonal-plus-rank-k matrix* (DPRk) is a matrix of the form

$$A = \Delta + x\rho y^*$$

where Δ is diagonal matrix with diagonal elements $\delta_i = \Delta_{ii} \in \mathbb{H}$, $\text{diag}(\Delta) \in \mathbb{H}^n$, $x, y \in \mathbb{H}^{n\times k}$ and $\rho \in \mathbb{H}^{k\times k}$.

The *diagonal-plus-rank-one matrix* (DPR1) is a special case of a DPRk matrix for $k = 1$.

1.4. Fast Multiplication and Inverses of Arrow and DPRk Matrices

The fast multiplication can be computed in $O(n)$ operations and the product is written in the form $w = Az$. It can be calculated using the following lemma.

Lemma 1 ([20], Lemma 1)**.** *Let $A = \text{Arrow}(D, u, v, \alpha) \in \mathbb{H}^{n\times n}$ be an Arrow matrix with the tip at position $A_{ii} = \alpha$ and let z be a vector. Then, $w = Az$, where*

$$\begin{aligned} w_j &= d_j z_j + u_j z_i, \quad j = 1, 2, \cdots, i-1 \\ w_i &= v^*_{1:i-1} z_{1:i-1} + \alpha z_i + v^*_{i:n-1} z_{i+1:n} \\ w_j &= u_{j-1} z_i + d_{j-1} z_j, \quad j = i+1, i+2, \cdots, n. \end{aligned}$$

Further, let $A = \text{DPRk}(\Delta, x, y, \rho) \in \mathbb{H}^{n\times n}$ be a DPRk matrix, $k \geq 1$ and let $\beta = \rho(y^* z) \equiv \rho(y \cdot z)$. Then, $w = Az$, where

$$w_i = \delta_i z_i + x_i \beta, \quad i = 1, 2, \cdots, n.$$

Inverses of Arrow and DPRk matrices are computed in $O(n)$ and $O(nk^2)$ operations, respectively, and can be calculated using the following lemmas (we usually consider cases where $k << n$, so we can use $O(n)$ instead of $O(nk^2)$).

Lemma 2 ([20], Lemma 4)**.** *Let $A = \text{Arrow}(D, u, v, \alpha) \in \mathbb{H}^{n\times n}$ be a non-singular arrow matrix with the tip at the position $A_{ii} = \alpha$ and let P be the permutation matrix of the permutation $p = (1, 2, \cdots, i-1, n, i, i+1, \cdots, n-1)$.*

If all $d_j \neq 0$, then the inverse of A is a DPR1 matrix

$$A^{-1} = \text{DPR1}(\Delta, x, y, \rho) = \Delta + x\rho y^*, \tag{1}$$

where

$$\Delta = P \begin{bmatrix} D^{-1} & 0 \\ 0 & 0 \end{bmatrix} P^T, \; x = P \begin{bmatrix} D^{-1}u \\ -1 \end{bmatrix}, \; y = P \begin{bmatrix} D^{-*}v \\ -1 \end{bmatrix}, \; \rho = (\alpha - v^* D^{-1} u)^{-1}.$$

If $d_j = 0$, then the inverse of A is an arrow matrix with the tip of the arrow at position (j, j) and zero at position A_{ii} (the tip and the zero on the shaft change places). In particular, let $\hat{P}$ be the

permutation matrix of the permutation $\hat{p} = (1, 2, \cdots, j-1, n, j, j+1, \cdots, n-1)$. *Partition* D, u *and* v *as*

$$D = \begin{bmatrix} D_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & D_2 \end{bmatrix}, \quad u = \begin{bmatrix} u_1 \\ u_j \\ u_2 \end{bmatrix}, \quad v = \begin{bmatrix} v_1 \\ v_j \\ v_2 \end{bmatrix}.$$

Then

$$A^{-1} = P \begin{bmatrix} \hat{D} & \hat{u} \\ \hat{v}^* & \hat{\alpha} \end{bmatrix} P^T, \tag{2}$$

where

$$\hat{D} = \begin{bmatrix} D_1^{-1} & 0 & 0 \\ 0 & D_2^{-1} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \hat{u} = \begin{bmatrix} -D_1^{-1} u_1 \\ -D_2^{-1} u_2 \\ 1 \end{bmatrix} u_j^{-1}, \quad \hat{v} = \begin{bmatrix} -D_1^{-*} v_1 \\ -D_2^{-*} v_2 \\ 1 \end{bmatrix} v_j^{-1},$$
$$\hat{\alpha} = v_j^{-*} \left(-\alpha + v_1^* D_1^{-1} u_1 + v_2^* D_2^{-1} u_2 \right) u_j^{-1}.$$

Lemma 3 ([20], Lemma 5). *Let* $A = \mathrm{DPRk}(\Delta, x, y, \rho) \in \mathbb{H}^{n \times n}$ *be a non-singular DPRk matrix. If all* $\delta_j \neq 0$, *then the inverse of* A *is a DPRk matrix*

$$A^{-1} = \mathrm{DPRk}(\hat{\Delta}, \hat{x}, \hat{y}, \hat{\rho}) = \hat{\Delta} + \hat{x} \hat{\rho} \hat{y}^*, \tag{3}$$

where

$$\hat{\Delta} = \Delta^{-1}, \qquad \hat{x} = \Delta^{-1} x, \quad \hat{y} = \Delta^{-*} y, \quad \hat{\rho} = -\rho (I + y^* \Delta^{-1} x \rho)^{-1}.$$

If $k = 1$ *and* $\delta_j = 0$, *then the inverse of* A *is an arrow matrix with the arrow tip at position* (j, j). *In particular, let* P *be the permutation matrix of the permutation* $p = (1, 2, \cdots, j-1, n, j, j+1, \cdots, n-1)$. *Partition* Δ, x *and* y *as*

$$\Delta = \begin{bmatrix} \Delta_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Delta_2 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_j \\ x_2 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_j \\ y_2 \end{bmatrix}.$$

Then,

$$A^{-1} = P \begin{bmatrix} D & u \\ v^* & \alpha \end{bmatrix} P^T, \tag{4}$$

where

$$D = \begin{bmatrix} \Delta_1^{-1} & 0 \\ 0 & \Delta_2^{-1} \end{bmatrix}, \quad u = \begin{bmatrix} -\Delta_1^{-1} x_1 \\ -\Delta_2^{-1} x_2 \end{bmatrix} x_j^{-1}, \quad v = \begin{bmatrix} -\Delta_1^{-*} y_1 \\ -\Delta_2^{-*} y_2 \end{bmatrix} y_j^{-1},$$
$$\alpha = y_j^{-*} \left(\rho^{-1} + y_1^* \Delta_1^{-1} x_1 + y_2^* \Delta_2^{-1} x_2 \right) x_j^{-1}.$$

2. Methods for Eigenvalue Decomposition

In this section, we first describe the standard QR method for general matrices of quaternions from [1]. In Section 4, we will compare our algorithm with this method. We then describe the method that will be used to compute individual eigenpairs, the Rayleigh Quotient Iteration with double shifts. In Section 2.4, we describe the basic deflation algorithm (Wielandt's deflation) and its adjustment for arrow and DPRk matrices. For each type of matrix, we describe the fast method to compute all eigenvectors and give a complete algorithm.

2.1. A Quaternion QR Algorithm

In [1], the authors proposed a QR type method for computing the Schur decomposition of a general matrix of quaternions. The matrix uses only unitary transformations and is thus stable.

To compute the eigenvalues decomposition of a general matrix of quaternions, we propose the following four-step algorithm based on the algorithm from [1]:

Given a matrix $A \in \mathbb{H}^{n\times n}$, Algorithm 1 has four steps:

Algorithm 1 Computing all eigenpairs of a quaternion matrix

Require: a general matrix $A \in \mathbb{H}^{n\times n}$

1. Reduce A to Hessenberg form by Householder reflectors, $X^*AX = H$, where X is unitary and H is an upper Hessenberg matrix.
2. Compute the Schur decomposition of H, $Q * HQ = T$, where Q is unitary and T is upper triangular matrix with eigenvalues of A on the diagonal, $\Lambda = \mathrm{diag}(T)$.
3. Compute the eigenvectors V of T by solving the Sylvester equation, $TV - V\Lambda = 0$. Then, $V^{-1}TV = \Lambda$.
4. Multiply $U = X * Q * V$. Then, $U^{-1}AU = \Lambda$ is the eigenvalue decomposition of A.

The reduction of a quaternionic matrix to Hessenberg form and the solution of the Sylvester equation are computed similarly to the real or complex case. Details of the Hessenberg reduction can be found in ([1], Appendix, Algorithms 1–3).

Consider a matrix $A \in \mathbb{H}^{n\times n}$ and the shift $\mu \in \mathbb{H}$. Set

$$M(A,\mu) = A^2 - (\mu + \bar{\mu})A + \mu\bar{\mu}I. \tag{5}$$

If $Ax = x\lambda$, then x is also an eigenvector of $M(A,\mu)$:

$$\begin{aligned} M(A,\mu)x &= (A^2 - (\mu+\bar{\mu})A + \mu\bar{\mu}I)x = x\lambda^2 - x(\mu+\bar{\mu})\lambda + x\mu\bar{\mu} \\ &= x(\lambda^2 - (\mu+\bar{\mu})\lambda + \mu\bar{\mu}). \end{aligned}$$

Further, for the perfect shift, $\mu = \lambda$, $M(A,\mu)$ has a zero eigenvalue:

$$\lambda^2 - (\mu+\bar{\mu})\lambda + \mu\bar{\mu} = \lambda^2 - (\lambda+\bar{\lambda})\lambda + \lambda\bar{\lambda} = 0.$$

The QR method from [1] applies non-real shift μ to an upper-Hessenberg matrix H by standard use of Francis double-shift on the matrix $M(H,\mu)$ and applying it implicitly on H. Details of the algorithm can be found in ([1], Appendix, Algorithm 4).

Since Algorithm 1 is used for general matrices, it requires $O(n^3)$ operations. The algorithm is stable and we use it in Section 4 for comparison with our method.

Julia [18] implementation of Algorithm 1 is given in the file `BGBM.jl`, which is available from the GitHub repository [21].

2.2. Rayleigh Quotient Iterations

The standard *Rayleigh Quotient Iteration* (RQI) produces sequences of shifts, approximate eigenvalues and vectors: for $k = 0, 1, 2, \ldots$

$$\lambda_k = \frac{1}{x_k^* x_k} x_k^* A x_k, \quad \mu_k = \mathrm{real}(\lambda_k), \quad y_k = (A - \mu_k I)^{-1} x_k, \quad x_{k+1} = \frac{y_k}{\|y_k\|}.$$

Since the eigenvalues of matrices of quaternions are not shift-invariant, only real shifts can be used. However, this works due to the following: let the matrix $A - \mu I$, $\mu \in \mathbb{R}$, have purely imaginary standard eigenvalue $i\nu$:

$$(A - \mu I)x = x(i\nu), \quad \mu, \nu \in \mathbb{R}.$$

Then

$$Ax = \mu x + xi\nu = x(\mu + i\nu),$$

so, $\lambda = \mu + i\nu$ is an eigenvalue of A and x is its eigenvector.

The method converges, but, since only real shifts are used, the convergence is rather slow and requires a large number of iterations, so "it is too restrictive to limit shifts to real numbers" ([1], p. 89).

Due to the fast computation of inverses from Section 1.4, one step of the RQI requires $O(n)$ operations for arrow and DPRk matrices. For both methods, $\lambda_k \to \lambda$ and $x_k \to x$, where (λ, x) is an eigenpair of A, provided that λ has a unique real part.

The iterations are terminated when

$$\|Ax_k - x_k\lambda_k\| \leq \tau, \tag{6}$$

where τ is a user-prescribed tolerance.

2.3. Rayleigh Quotient Iterations with Double Shifts

In order to speed the convergence up, we enhance the RQI with non-real double shifts, using the approach from [1] and Section 2.1.

The *Rayleigh Quotient Iteration with Double Shifts* (RQIds) produces sequences of shifts and vectors

$$\mu_k = \frac{1}{x_k^* x_k} x_k^* A x_k, \quad y_k = [M(A, \mu_k)]^{-1} x_k, \quad x_{k+1} = \frac{y_k}{\|y_k\|}, \quad k = 0, 1, 2, \ldots,$$

where $M(A, \mu_k)$ is defined by (5). Therefore, vector y_k is the solution of the system $M(A, \mu_k)y_k = x_k$, that is

$$(A^2 - \hat{\alpha}A + \beta I)y_k = x_k, \quad \hat{\alpha} = \mu_k + \bar{\mu}_k, \quad \beta = \mu_k\bar{\mu}_k. \tag{7}$$

Notice that $\hat{\alpha}$ and β are real.

In order to have a fast method which requires $O(n^2)$ operations to compute all eigenvalues and eigenvectors, the RQIds should compute one eigenpair in $O(n)$ operations. Therefore, Equation (7) must be solved in $O(n)$ operations.

If A is an arrowhead matrix, one step of the method requires $O(n)$ operations: let $x_k = \begin{bmatrix} x \\ \xi \end{bmatrix}$. Then:

$$\left(\begin{bmatrix} D & u \\ v^* & \alpha \end{bmatrix} \begin{bmatrix} D & u \\ v^* & \alpha \end{bmatrix} - \hat{\alpha} \begin{bmatrix} D & u \\ v^* & \alpha \end{bmatrix} + \beta \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix} \right) y_k = \begin{bmatrix} x \\ \chi \end{bmatrix}.$$

Therefore,

$$\left(\begin{bmatrix} D^2 + uv^* & Du + u\alpha \\ v^*D + \alpha v^* & v^*u + \alpha^2 \end{bmatrix} - \hat{\alpha} \begin{bmatrix} D & u \\ v^* & \alpha \end{bmatrix} + \beta \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix} \right) y_k = \begin{bmatrix} x \\ \chi \end{bmatrix},$$

so

$$\begin{bmatrix} D^2 - \hat{\alpha}D + \beta I + uv^* & Du + u(\alpha - \hat{\alpha}) \\ v^*D + (\alpha - \hat{\alpha})v^* & v^*u + (\alpha - \hat{\alpha})\alpha + \beta \end{bmatrix} y_k = \begin{bmatrix} x \\ \chi \end{bmatrix}. \tag{8}$$

The matrix $C = D^2 - \hat{\alpha}D + \beta I + uv^*$ is a DPRk (DPR1) matrix,

$$C = \mathrm{DPRk}(D^2 - \hat{\alpha}D + \beta I, u, v, 1).$$

Multiplication of Equation (8) by the block matrix $\begin{bmatrix} C^{-1} & \\ & 1 \end{bmatrix}$ from the left yields

$$\begin{bmatrix} I & C^{-1}(Du + u(\alpha - \hat{\alpha})) \\ v^*D + (\alpha - \hat{\alpha})v^* & v^*u + (\alpha - \hat{\alpha})\alpha + \beta \end{bmatrix} y_k = \begin{bmatrix} C^{-1}x \\ \xi \end{bmatrix}. \tag{9}$$

Let B denote the matrix on the left hand side of Equation (9). Then, B is an arrowhead matrix and, finally, $y_k = B^{-1} \begin{bmatrix} C^{-1}x \\ \xi \end{bmatrix}$. Due to fast computation of inverses of arrowhead

and DPRk matrices from Lemmas 2 and 3, respectively, one step of the RQIds method requires $O(n)$ operations.

If A is a DPRk matrix, Equation (7) becomes

$$[(\Delta + x\rho y^*)(\Delta + x\rho y^*) - \hat{\alpha}(\Delta + x\rho y^*) + \beta I]y_k = x_k,$$

or

$$[\Delta(\Delta - \hat{\alpha}I) + \beta I + x\rho y^*\Delta + (\Delta x + x\rho(y^*x) - \hat{\alpha}x)\rho y^*]y_k = x_k. \tag{10}$$

Similarly to the arrowhead case, we solve Equation (10) accurately in two steps: let $\hat{\Delta} = \Delta(\Delta - \hat{\alpha}I) + \beta I$ and let

$$C = \hat{\Delta} + x\rho y^*\Delta \equiv \text{DPRk}(\hat{\Delta}, x, \Delta^* y, \rho).$$

Then, Equation (10) becomes

$$C^{-1}My_k = C^{-1}x_k,$$

or

$$Dy_k = \hat{x}_k,$$

where $D = C^{-1}M$ and $\hat{x}_k = C^{-1}x_k$, provided C is non-singular. Notice that D is a DPRk matrix,

$$D = \text{DPRk}(I, C^{-1}[\Delta x + x\rho(y^*x) - \hat{\alpha}x], y, \rho).$$

Since both C and D are DPRk matrices, due to the fast computation of inverses of DPRk matrices from Lemma 3, one step of the RQIds method requires $O(n)$ operations.

2.4. Wielandt's Deflation

Theorem 1. *Let $A \in \mathbb{H}^{n\times n}$ and (λ, u) be a right eigenpair of A. Choose z such that $z^*u = 1$, say $z^* = \begin{bmatrix}1/u_1 & 0 & \cdots & 0\end{bmatrix}$. Compute the deflated matrix $\tilde{A} = (I - uz^*)A$. Then, $(0, u)$ is an eigenpair of $\tilde{A}$. Further, if (μ, v) is an eigenpair of A, then $(\mu, \tilde{v})$, where $\tilde{v} = (I - uz^*)v$ is an eigenpair of $\tilde{A}$.*

Proof. The proof can be found in [13], but we repeat it for the sake of completeness. Using $Au = u\lambda$ and $z^*u = 1$, the first statement holds since

$$\tilde{A}u = (I - uz^*)Au = Au - uz^*Au = u\lambda - uz^*u\lambda = 0.$$

Further,

$$\begin{aligned}\tilde{A}\tilde{v} &= (I - uz^*)A(I - uz^*)v \\ &= (I - uz^*)Av - Auz^*v + uz^*Auz^*v \\ &= (I - uz^*)v\mu - u\lambda z^*v + uz^*u\lambda z^*v \\ &= \tilde{v}\mu\end{aligned}.$$

□

2.5. Deflation for Arrow Matrices

Lemma 4. *Let $A \in \mathbb{H}^{n\times n}$ be an arrow matrix partitioned as*

$$A = \begin{bmatrix}\delta & 0 & \chi \\ 0 & \Delta & x \\ \bar{v} & y^* & \alpha\end{bmatrix},$$

where χ, v *and* α *are scalars,* x *and* y *are vectors and* Δ *is a diagonal matrix. Let* $\left(\lambda, \begin{bmatrix} \nu \\ u \\ \psi \end{bmatrix}\right)$, *where* ν *and* ψ *are scalars and* u *is a vector, be an eigenpair of* A. *Then, the deflated matrix* $\tilde{A}$ *has the form*

$$\tilde{A} = \begin{bmatrix} 0 & 0^T \\ w & \hat{A} \end{bmatrix} \tag{11}$$

where

$$w = \begin{bmatrix} -u\frac{1}{\nu}\delta \\ -\psi\frac{1}{\nu}\delta + \bar{v} \end{bmatrix},$$

and $\hat{A}$ *is an arrow matrix*

$$\hat{A} = \begin{bmatrix} \Delta & -u\frac{1}{\nu}\chi + x \\ y^* & -\psi\frac{1}{\nu}\chi + \alpha \end{bmatrix}. \tag{12}$$

Proof. We have

$$\begin{aligned} \tilde{A} &= \left(\begin{bmatrix} 1 & 0^T & 0 \\ 0 & I & 0 \\ 0 & 0^T & 1 \end{bmatrix} - \begin{bmatrix} \nu \\ u \\ \psi \end{bmatrix} \begin{bmatrix} \frac{1}{\nu} & 0^T & 0 \end{bmatrix} \right) \begin{bmatrix} \delta & 0 & \chi \\ 0 & \Delta & x \\ \bar{v} & y^* & \alpha \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0^T & 0 \\ -u\frac{1}{\nu} & I & 0 \\ -\psi\frac{1}{\nu} & 0^T & 1 \end{bmatrix} \begin{bmatrix} \delta & 0 & \chi \\ 0 & \Delta & x \\ \bar{v} & y^* & \alpha \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0^T & 0 \\ -u\frac{1}{\nu}\delta & \Delta & -u\frac{1}{\nu}\chi + x \\ -\psi\frac{1}{\nu}\delta + \bar{v} & y^* & -\psi\frac{1}{\nu}\chi + \alpha \end{bmatrix}, \end{aligned} \tag{13}$$

as desired. □

Lemma 5. *Let* A *and* $\hat{A}$ *be as in Lemma 4. If* $\left(\mu, \begin{bmatrix} \hat{z} \\ \hat{\xi} \end{bmatrix}\right)$ *is an eigenpair of* $\hat{A}$, *then the eigenpair of* A *is*

$$\left(\mu, \begin{bmatrix} \zeta \\ \hat{z} + u\frac{1}{\nu}\zeta \\ \hat{\xi} + \psi\frac{1}{\nu}\zeta \end{bmatrix}\right), \tag{14}$$

where ζ *is the solution of the scalar Sylvester equation*

$$\left(\delta + \chi\psi\frac{1}{\nu}\right)\zeta - \zeta\mu = -\chi\hat{\xi}. \tag{15}$$

Proof. If μ is an eigenvalue of $\hat{A}$, it is also an eigenvalue of $\tilde{A}$ and then also of A. Assume that the corresponding eigenvector of A is partitioned as $\begin{bmatrix} \zeta \\ z \\ \xi \end{bmatrix}$. By combining (11)–(13), we have

$$\begin{bmatrix} 0 & 0^T & 0 \\ -u\frac{1}{\nu}\delta & \Delta & -u\frac{1}{\nu}\chi + x \\ -\psi\frac{1}{\nu}\delta + \bar{v} & y^* & -\psi\frac{1}{\nu}\chi + \alpha \end{bmatrix} \begin{bmatrix} 0 & 0^T & 0 \\ -u\frac{1}{\nu} & I & 0 \\ -\psi\frac{1}{\nu} & 0^T & 1 \end{bmatrix} \begin{bmatrix} \zeta \\ z \\ \xi \end{bmatrix} = \begin{bmatrix} 0 & 0^T & 0 \\ -u\frac{1}{\nu} & I & 0 \\ -\psi\frac{1}{\nu} & 0^T & 1 \end{bmatrix} \begin{bmatrix} \zeta \\ z \\ \xi \end{bmatrix} \mu,$$

or

$$\begin{bmatrix} 0 & 0^T & 0 \\ -u\frac{1}{\nu}\delta & \Delta & -u\frac{1}{\nu}\chi + x \\ -\psi\frac{1}{\nu}\delta + \bar{v} & y^* & -\psi\frac{1}{\nu}\chi + \alpha \end{bmatrix} \begin{bmatrix} 0 \\ -u\frac{1}{\nu}\zeta + z \\ -\psi\frac{1}{\nu}\zeta + \xi \end{bmatrix} = \begin{bmatrix} 0 \\ -u\frac{1}{\nu}\zeta + z \\ -\psi\frac{1}{\nu}\zeta + \xi \end{bmatrix} \mu.$$

Since the bottom right 2×2 matrix is $\hat{A}$, the above equation implies

$$\begin{bmatrix} \hat{z} \\ \hat{\xi} \end{bmatrix} = \begin{bmatrix} -u\frac{1}{\nu}\zeta + z \\ -\psi\frac{1}{\nu}\zeta + \xi \end{bmatrix},$$

which proves (14). It remains to compute ζ. The first component of equality

$$A \begin{bmatrix} \zeta \\ z \\ \xi \end{bmatrix} = \begin{bmatrix} \delta & 0 & \chi \\ 0 & \Delta & x \\ \bar{v} & y^* & \alpha \end{bmatrix} \begin{bmatrix} \zeta \\ z \\ \xi \end{bmatrix} = \begin{bmatrix} \zeta \\ z \\ \xi \end{bmatrix} \mu$$

implies

$$\delta\zeta + \chi\xi = \zeta\mu,$$

or

$$\delta\zeta + \chi(\hat{\xi} + \psi\frac{1}{\nu}\zeta) = \zeta\mu,$$

which is exactly Equation (15). □

2.5.1. Computing the Eigenvectors of Arrow Matrices

Let $\left(\lambda, \begin{bmatrix} \nu \\ u \\ \psi \end{bmatrix}\right)$ be an eigenpair of the matrix A partitioned according to Lemma 4, that is

$$\begin{bmatrix} \delta & 0 & \chi \\ 0 & \Delta & x \\ \bar{v} & y^* & \alpha \end{bmatrix} \begin{bmatrix} \nu \\ u \\ \psi \end{bmatrix} = \begin{bmatrix} \nu \\ u \\ \psi \end{bmatrix} \lambda.$$

If λ and ψ are known, then the other components of the eigenvector are solutions of scalar Sylvester equations

$$\begin{aligned} \delta\nu - \nu\lambda &= -\chi\psi, \\ \Delta_{ii}u_i - u_i\lambda &= -x_i\psi, \quad i = 1, \ldots, n-2. \end{aligned} \tag{16}$$

By setting

$$\gamma = \delta + \chi\psi\frac{1}{\nu}$$

the Sylvester Equation (15) becomes

$$\gamma\zeta - \zeta\mu = -\chi\hat{\xi}. \tag{17}$$

Dividing first the equation in (16) by ν from the right and combining with the expression for γ gives

$$\gamma = \nu\lambda\frac{1}{\nu}. \tag{18}$$

2.5.2. Complete Algorithm for Arrow Matrices

In the first (forward) pass, the absolute largest eigenvalue and its eigenvector are computed by RQIds in each step. The first element of the current vector x and the first and the last elements of the current eigenvector are stored. The current value γ is computed using (18) and stored. The deflation is then performed according to Lemma 4.

The eigenvectors are reconstructed bottom-up, that is from the smallest matrix to the original one (a backward pass). In each iteration, we need to have access to the first element of the vector x which was used to define the current arrow matrix, its absolutely largest eigenvalue and the first and the last elements of the corresponding eigenvector.

In the ith step, for each $j = i+1, \ldots, n$ the following steps are performed:

1. Equation (17) is solved for ζ (the first element of the eigenvector of the larger matrix). The quantity $\hat{\zeta}$ is the last element of the eigenvectors and was stored in the forward pass.
2. The first element of the eigenvector of the super-matrix is updated (set to ζ).
3. The last element of the eigenvectors of the super-matrix is updated using (14).

All iterations are completed in $O(n^2)$ operations. After all iterations are completed, we have:

- the absolutely largest eigenvalue and its eigenvector (unchanged from the first run of the RQIds);
- all other eigenvalues and the last elements of their corresponding eigenvectors.

The rest of the elements of the remaining eigenvectors are computed using the procedure in Section 2.5.1. This step also requires $O(n^2)$ operations.

Finally, due to floating-point error in operations with quaternions, the computed eigenpairs have larger residuals that required. This is successfully remedied by running again few steps of the RQIds, but starting from the computed eigenvectors. This has the effect of using nearly perfect shifts, so typically just a few additional iterations are needed to attain the desired accuracy. This step also requires $O(n^2)$ operations.

Algorithm 2 is implemented in the function `eigen()` in the Code section of the notebook `ED_Arrow.jl`, which is available from the GitHub repository [21]. The file also contains the code which is used to generate and run examples which are presented in Section 4. The implementation is written in the programming language Julia [18], which enables elegant and efficient manipulation with quaternions.

Algorithm 2 Computing all eigenpairs of an arrow matrix

Require: an arrow matrix $A \in \mathbb{H}^{n\times n}$
Compute and store the first eigenpair (λ_1, u) using RQIds
Compute the deflated matrix $\hat{A}$ according to Lemma 4
Compute $\gamma = \nu\lambda\frac{1}{\nu}$
Compute and store ν, χ, ψ according to Lemma 4
for $i = 2, 3, \ldots, n-1$ **do**
  Compute $g = \frac{1}{\nu}\chi$
  Compute w from (12): $w = x - ug$
  Compute the new matrix $\hat{A}$ from (12)
  Compute and store an eigenpair (λ_i, u) of $\hat{A}$ using RQIds
  Update and store $\gamma_i, \nu_i, \chi_i, \psi_i$ according to Lemma 4
end for
Compute and store the last eigenvalue
for $i = n-1, n-2, \ldots, 1$ **do**
  for $j = i+1, i+2, \ldots, n$ **do**
    Solve the Sylvester equation $\gamma_i\zeta - \zeta\lambda_j = -\chi_i\psi_j$ for ζ
    Update ν_j and ψ_j, the first and the last element of the eigenvector of the super-matrix, respectively:

$$\nu_j = \zeta, \quad \psi_j = \psi_j + \psi_i\frac{\zeta}{\nu_i}$$

  end for
end for
Reconstruct all eigenvectors from the computed eigenvalues and respective first and last elements using (16)
Correct the computed eigenpairs by running few steps of RQIds with nearly perfect shifts.

2.6. Deflation for DPRk Matrices

Lemma 6. *Let $A \in \mathbb{H}^{n\times n}$ be a DPRk matrix partitioned as*

$$A = \begin{bmatrix} \delta & 0^T \\ 0 & \Delta \end{bmatrix} + \begin{bmatrix} \chi \\ x \end{bmatrix} \rho \begin{bmatrix} \bar{v} & y^* \end{bmatrix},$$

and let $\left(\lambda, \begin{bmatrix} \nu \\ u \end{bmatrix}\right)$ be the eigenpair of A. Then, the deflated matrix $\tilde{A}$ has the form

$$\tilde{A} = \begin{bmatrix} 0 & 0^T \\ w & \hat{A} \end{bmatrix} \tag{19}$$

where

$$w = -u\frac{1}{\nu}\delta - u\frac{1}{\nu}\chi\rho\bar{v} + x\rho\bar{v}$$

and $\hat{A}$ is a DPRk matrix

$$\hat{A} = \Delta + \hat{x}\rho y^*, \quad \hat{x} = x - u\frac{1}{\nu}\chi. \tag{20}$$

Proof. We have

$$\begin{aligned} \tilde{A} &= \left(\begin{bmatrix} 1 & 0^T \\ 0 & I \end{bmatrix} - \begin{bmatrix} \nu \\ u \end{bmatrix} \begin{bmatrix} \frac{1}{\nu} & 0^T \end{bmatrix}\right)\left(\begin{bmatrix} \delta & 0^T \\ 0 & \Delta \end{bmatrix} + \begin{bmatrix} \chi \\ x \end{bmatrix} \rho \begin{bmatrix} \bar{v} & y^* \end{bmatrix}\right) \\ &= \begin{bmatrix} 0 & 0^T \\ -u\frac{1}{\nu} & I \end{bmatrix} \cdot \begin{bmatrix} \delta + \chi\rho\bar{v} & \chi\rho y^* \\ x\rho\bar{v} & \Delta + x\rho y^* \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0^T \\ -u\frac{1}{\nu}\delta - u\frac{1}{\nu}\chi\rho\bar{v} + x\rho\bar{v} & -u\frac{1}{\nu}\chi\rho y^* + \Delta + x\rho y^* \end{bmatrix}, \end{aligned} \tag{21}$$

as desired. □

Lemma 7. *Let A, $\tilde{A}$ and $\hat{A}$ be as in Lemma 6. If $(\mu, \hat{z})$ is an eigenpair of $\hat{A}$, then the eigenpair of A is*

$$\left(\mu, \begin{bmatrix} \zeta \\ \hat{z} + u\frac{1}{\nu}\zeta \end{bmatrix}\right),$$

where ζ is the solution of the Sylvester equation

$$(\delta + \chi\rho\bar{v} + \chi\rho y^* u\frac{1}{\nu})\zeta - \zeta\mu = -\chi\rho y^*\hat{z}. \tag{22}$$

Proof. If μ is an eigenvalue of $\hat{A}$, it is also an eigenvalue of $\tilde{A}$ and then also of A. Assume that the corresponding eigenvector of A is partitioned as $\begin{bmatrix} \zeta \\ z \end{bmatrix}$. By combining (19) and (21) and the previous results, it must hold that

$$\begin{bmatrix} 0 & 0^T \\ w & \hat{A} \end{bmatrix} \begin{bmatrix} 0 & 0^T \\ -u\frac{1}{\nu} & I \end{bmatrix} \begin{bmatrix} \zeta \\ z \end{bmatrix} = \begin{bmatrix} 0 & 0^T \\ -u\frac{1}{\nu} & I \end{bmatrix} \begin{bmatrix} \zeta \\ z \end{bmatrix} \mu,$$

or

$$\begin{bmatrix} 0 & 0^T \\ w & \hat{A} \end{bmatrix} \begin{bmatrix} 0 \\ -u\frac{1}{\nu}\zeta + z \end{bmatrix} = \begin{bmatrix} 0 \\ -u\frac{1}{\nu}\zeta + z \end{bmatrix} \mu.$$

Therefore, $\hat{z} = -u\frac{1}{\nu}\zeta + z$ or

$$z = \hat{z} + u\frac{1}{\nu}\zeta, \tag{23}$$

and it remains to compute ζ. From the equality

$$\left(\begin{bmatrix}\delta & 0^T\\ 0 & \Delta\end{bmatrix} + \begin{bmatrix}\chi\\ x\end{bmatrix}\rho\begin{bmatrix}\bar{v} & y^*\end{bmatrix}\right)\begin{bmatrix}\zeta\\ z\end{bmatrix} = \begin{bmatrix}\zeta\\ z\end{bmatrix}\mu$$

it follows that

$$\begin{bmatrix}\delta+\chi\rho\bar{v} & \chi\rho y^*\\ x\rho\bar{v} & \Delta+x\rho y^*\end{bmatrix}\begin{bmatrix}\zeta\\ z\end{bmatrix} = \begin{bmatrix}\zeta\\ z\end{bmatrix}\mu.$$

Equating the first elements and using (23) gives

$$(\delta+\chi\rho\bar{v})\zeta + \chi\rho y^*\hat{z} + \chi\rho y^* u\frac{1}{\nu}\zeta = \zeta\mu,$$

which is exactly the Sylvester Equation (22). □

2.6.1. Computing the Eigenvectors of DPRk Matrices

Let the DPRk matrix A and its eigenpair be defined as in Lemma 6. Let x be partitioned row-wise as

$$x = \begin{bmatrix}x_1\\ x_2\\ \vdots\\ x_{n-1}\end{bmatrix}.$$

Set $\alpha = \rho\begin{bmatrix}\bar{v} & y^*\end{bmatrix}\begin{bmatrix}\nu\\ u\end{bmatrix}$. From

$$\left(\begin{bmatrix}\delta & 0^T\\ 0 & \Delta\end{bmatrix} + \begin{bmatrix}\chi\\ x\end{bmatrix}\rho\begin{bmatrix}\bar{v} & y^*\end{bmatrix}\right)\begin{bmatrix}\nu\\ u\end{bmatrix} = \begin{bmatrix}\nu\\ u\end{bmatrix}\lambda,$$

it follows that the elements of the eigenvector satisfy scalar Sylvester equations

$$\begin{aligned}\delta\nu - \nu\lambda &= -\chi\alpha,\\ \Delta_{ii}u_i - u_i\lambda &= -x_i\alpha, \quad i = 1,\ldots,n-1.\end{aligned} \tag{24}$$

If λ, ν and the first $k-1$ components of u are known, then α is computed from the first k equations in (24), that is, by solving the system

$$\begin{bmatrix}\chi\\ x_1\\ \vdots\\ x_{k-1}\end{bmatrix}\alpha = \begin{bmatrix}\nu\lambda-\delta\nu\\ u_1\lambda-\Delta_{11}u_1\\ \vdots\\ u_{k-1}\lambda-\Delta_{k-1,k-1}u_{k-1}\end{bmatrix},$$

and u_i, $i = k,\ldots,n-1$ are computed by solving the remaining Sylvester equations in (24).

In special cases, for DPR1 matrices, we can compute their eigenvectors via the following method: let the DPR1 matrix A and its eigenpair be defined as in Lemma 6. Set $\alpha = \rho\begin{bmatrix}\bar{v} & y^*\end{bmatrix}\begin{bmatrix}\nu\\ u\end{bmatrix}$. From

$$\left(\begin{bmatrix}\delta & 0^T\\ 0 & \Delta\end{bmatrix} + \begin{bmatrix}\chi\\ x\end{bmatrix}\rho\begin{bmatrix}\bar{v} & y^*\end{bmatrix}\right)\begin{bmatrix}\nu\\ u\end{bmatrix} = \begin{bmatrix}\nu\\ u\end{bmatrix}\lambda,$$

it follows that the elements of the eigenvector satisfy the scalar Sylvester equations

$$\begin{aligned}\delta\nu - \nu\lambda &= -\chi\alpha,\\ \Delta_{ii}u_i - u_i\lambda &= -x_i\alpha, i = 1,\ldots,n-1.\end{aligned} \tag{25}$$

If λ and ν are known, then α is computed from the first equation in (25),

$$\alpha = \frac{1}{\chi}(\nu\lambda - \delta\nu),$$

and u_i, $i = 1, \ldots, n-1$ are computed by solving the remaining Sylvester equations in (25).

Lemma 8. *Assume A and its eigenpair are given as in Lemma 6 and $\hat{A}$ and its eigenpair are given as in Lemma 7. Set in (22)*

$$\gamma = \delta + \chi\rho\bar{v} + \chi\rho y^* u\frac{1}{\nu}, \alpha = \rho y^* \hat{z}.$$

Then,

$$\gamma = \nu\lambda\frac{1}{\nu} \tag{26}$$

and α is the solution of the system

$$\begin{bmatrix} x_1 - u_1\frac{1}{\nu}\chi \\ \vdots \\ x_k - u_k\frac{1}{\nu}\chi \end{bmatrix}\alpha = \begin{bmatrix} \hat{z}_1\mu - \Delta_{11}\hat{z}_1 \\ \vdots \\ \hat{z}_k\mu - \Delta_{kk}\hat{z}_k \end{bmatrix}. \tag{27}$$

Proof. The formula for γ follows by multiplying the first elements of the equation

$$\left(\begin{bmatrix} \delta & 0^T \\ 0 & \Delta \end{bmatrix} + \begin{bmatrix} \chi \\ x \end{bmatrix}\rho\begin{bmatrix} \bar{v} & y^* \end{bmatrix}\right)\begin{bmatrix} \nu \\ u \end{bmatrix} = \begin{bmatrix} \nu \\ u \end{bmatrix}\lambda$$

with $\frac{1}{\nu}$ from the right.

Consider the equation $\hat{A}\hat{z} = \hat{z}\mu$, that is,

$$[\Delta + (x - u\frac{1}{\nu}\chi)\rho y^*]\hat{z} = \hat{z}\mu.$$

The i-th component is

$$\Delta_{ii}\hat{z}_i + (x_i - u_i\frac{1}{\nu}\chi)\alpha = \hat{z}_i\mu,$$

which gives (27). □

2.6.2. Complete Algorithm for DPRk Matrices

Lemmas 6 and 8 are used as follows. In the first (forward) pass, the absolute largest eigenvalue and its eigenvector are computed by RQIds in each step. The first two elements of the current vector x and the current eigenvector are stored. The current value γ is computed using (26) and stored. The deflation is then performed according to Lemma 6.

The eigenvectors are reconstructed bottom-up, that is from the smallest 2×2 matrix to the original one (a backward pass). In each iteration, we need to have access to the first two elements of the vector x which was used to define the current DPRk matrix, its absolutely largest eigenvalue and the first two elements of the corresponding eigenvector.

In the i-th step, for each $j = i+1, \ldots, n$, the following steps are performed:

1. The value α is computed from (27).
2. The Equation (22), which now reads $\gamma\zeta - \zeta\mu = -\chi\alpha$, is solved for ζ (the first element of the eigenvector of the larger matrix).
3. First element of the eigenvector of super-matrix is updated (set to ζ).

Iterations are completed in $O(kn^2)$ operations. After all iterations are completed, we have:

- The absolutely largest eigenvalue and its eigenvector (unchanged from the first run of the RQIds);
- All other eigenvalues and the first elements of their corresponding eigenvectors.

The rest of the elements of the remaining eigenvectors are computed using the procedure described in Section 2.6.1. This step requires $O(n^2)$ operations.

Algorithm 3 is implemented in the function `eigen()` in the Code section of the notebook `ED_DPRk.jl`, which is available from the GitHub repository [21]. The file also contains the code which is used to generate and run examples which are presented in Section 4.

Algorithm 3 Computing all eigenpairs of a DPRk matrix

Require: a DPRk matrix $A \in \mathbb{H}^{n\times n}$
  Compute and store the first eigenpair (λ_1, u) using RQIds
  Compute the deflated matrix $\hat{A}$ according to Lemma 6
  Compute $\gamma = \nu\lambda\frac{1}{\nu}$
  Compute and store ν, u_1, χ according to Lemma 6
  for $i = 2, 3, \ldots, n$ **do**
    Compute $g = \frac{1}{\nu}\chi$.
    Compute $\hat{x}$ from (20): $\hat{x} = x - ug$
    Compute $\hat{A} = \Delta + \hat{x}\rho y^*$
    Compute and store an eigenpair (λ_i, u) of $\hat{A}$ using RQIds
    Update and store $\gamma_i, \nu_i, \chi_i, u_{1_i}, x_{1_i}$ according to Lemma 6
  end for
  for $i = n-1, n-2, \ldots, 1$ **do**
    for $j = i+1, i+2, \ldots, n$ **do**
      Compute α from (27): $\alpha = \frac{1}{x_{1_i} - u_{1_i}\frac{1}{\nu_i}\chi_i}(\nu_j\lambda_j - \Delta_{i+1}\nu_j)$
      Solve the Sylvester equation $\gamma_i\zeta - \zeta\lambda_j = -\chi_i\alpha$ for ζ
      Update the first element of the eigenvector of the super-matrix: $\nu_j = \zeta$
    end for
  end for
  Reconstruct all eigenvectors from the computed eigenvalues and respective first elements using (25)
  Correct the computed eigenpairs by running few steps of RQIds with nearly perfect shifts.

3. Perturbation Theory, Error Analysis and Error Bounds

In this section, we first state several Bauer–Fike-type perturbation results for eigenpairs and complete eigenvalue decompositions of matrices of quaternions. In Section 3.2, we give errors of basic operation with quaternions (addition and multiplication of two quaternions), the dot product of two vectors of quaternions and the multiplication of two matrices of quaternions. In Section 3.3, we combine results from Sections 3.1 and 3.2 and obtain residual error bounds which can be efficiently used to check the accuracy of the computed eigenvalue decomposition.

3.1. Perturbation Theory

In 2021, Ahmad et al. [2] stated and proved the Bauer–Fike-type theorem for the right eigenvalues of a perturbed quaternionic matrix.

Theorem 2 ([2], Theorem 3.7). *Let $A \in \mathbb{H}^{n\times n}$ be a diagonalizable matrix, with $A = XDX^{-1}$, where $X \in \mathbb{H}^{n\times n}$ is invertible and $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ with λ_i being the standard right eigenvalues of A. If μ is a standard right eigenvalue of $A + \Delta A$, then*

$$\operatorname{dist}(\mu, \Lambda_s(A)) = \min_{\lambda_i \in \Lambda_s(A)} \{|\lambda_i - \mu|\} \leq \kappa(X)\|\Delta A\|_2.$$

Moreover, we have

$$\operatorname{dist}(\xi, \Lambda(A)) = \inf_{\eta_j \in \Lambda(A)} \{|\eta_j - \xi|\} \leq \kappa(X)\|\Delta A\|_2,$$

where $\xi \in \Lambda(A + \Delta A)$ *and* $\kappa(\cdot)$ *is the condition number with respect to the matrix* 2*-norm.*

The following corollary and theorem are given when a matrix is normal or Hermitian, respectively.

Corollary 1 ([2], Corollary 3.8)**.** *Let* $A \in \mathbb{H}^{n\times n}$ *be a normal matrix and let* μ *be a standard right eigenvalue of the perturbed quaternionic matrix* $A + \Delta A$*. Then,*

$$\operatorname{dist}(\mu, \Lambda_s(A)) = \min_{\lambda_i \in \Lambda_s(A)} \{|\lambda_i - \mu|\} \leq \|\Delta A\|_2.$$

Moreover, we have

$$\operatorname{dist}(\xi, \Lambda(A)) = \inf_{\eta_j \in \Lambda(A)} \{|\eta_j - \xi|\} \leq \|\Delta A\|_2,$$

where $\xi \in \Lambda(A + \Delta A)$*.*

Theorem 3 ([2], Theorem 3.12)**.** *Let* $A \in \mathbb{H}^{n\times n}$ *be a Hermitian matrix. For some* $\tilde{\mu} \in \mathbb{R}$ *and* $\tilde{x} \in \mathbb{H}^n$ *with* $\|\tilde{x}\|_2 = 1$*, define the residual vector* $r = A\tilde{x} - \tilde{\mu}\tilde{x}$*. Then,* $|\tilde{\mu} - \mu| \leq \|r\|_2$ *for some* $\mu \in \Lambda(A)$*.*

We now prove two residual bounds for general (non-Hermitian) matrices:

Theorem 4. *Let* $(\tilde{\lambda}, \tilde{x})$ *be the approximate eigenpair of the matrix* A*, where* $\|\tilde{x}\|_2 = 1$*. Let*

$$r = A\tilde{x} - \tilde{x}\tilde{\lambda}, \quad \Delta A = -r\tilde{x}^*.$$

Then, $(\tilde{\lambda}, \tilde{x})$ *is the eigenpair of the matrix* $A + \Delta A$ *and* $\|\Delta A\|_2 \leq \|r\|_2$*.*

Proof. We have

$$(A + \Delta A)\tilde{x} - \tilde{x}\tilde{\lambda} = A\tilde{x} - \tilde{x}\tilde{\lambda} + \Delta A\tilde{x} = r - r(\tilde{x}^*\tilde{x}) = r - r = 0,$$

which proves the first statement. Obviously,

$$\|\Delta A\|_2 \leq \|r\|_2\|\tilde{x}\|_2 = \|r\|_2.$$

□

In the case of several computed eigenvalues and corresponding eigenvectors, we have the following result:

Theorem 5. *Let* $(\tilde{\lambda}_i, \tilde{x}_i)$*,* $i = 1, \ldots, m$ *be approximate eigenpairs of the matrix* A*, where* $\|\tilde{x}_i\|_2 = 1$*. Set* $\tilde{\Lambda} = \operatorname{diag}(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_m)$ *and* $\tilde{X} = \begin{bmatrix} \tilde{x}_1 & \cdots & \tilde{x}_m \end{bmatrix}$*. We assume that eigenvectors are linearly independent. Let*

$$R = A\tilde{X} - \tilde{X}\tilde{\Lambda}, \quad \Delta A = -R(\tilde{X}^*\tilde{X})^{-1}\tilde{X}^*.$$

Then, $(\tilde{\lambda}_i, \tilde{x}_i)$*,* $i = 1, \ldots, m$ *are the eigenpairs of the matrix* $A + \Delta A$ *and*

$$\|\Delta A\|_2 \leq \|R\|_2\|(\tilde{X}^*\tilde{X})^{-1}\tilde{X}^*\|_2.$$

Proof. We have

$$\begin{aligned}(A+\Delta A)\tilde{X}-\tilde{X}\tilde{\lambda} &= A\tilde{X}-\tilde{X}\tilde{\lambda}+\Delta A\tilde{X} = R+[-R(\tilde{X}^*\tilde{X})^{-1}\tilde{X}^*]\tilde{}X\\ &= R-R(\tilde{X}^*\tilde{X})^{-1}(\tilde{X}^*\tilde{X}) = R-R=0,\end{aligned}$$

which proves the first statement. Obviously,

$$\|\Delta A\|_2 \le \|-R(\tilde{X}^*\tilde{X})^{-1}\tilde{X}^*\|_2 = \|R\|_2\|(\tilde{X}^*\tilde{X})^{-1}\tilde{X}^*\|_2.$$

□

3.2. Errors of Basic Operations

We are assuming the standard model of floating-point arithmetic [9]: for $\odot \in \{+,*,/\}$, we have

$$fl(a\odot b) = (1+\varepsilon_\odot)(a\odot b), \qquad |\varepsilon_\odot| \le \varepsilon,$$

where a and b are floating-point numbers and ε is the machine precision (twice the round-off unit).

An error of the addition of quaternions is bounded as follows:

$$|fl((p+q)-(p+q)| \le 2|p+q|\varepsilon, \tag{28}$$

or, equivalently

$$fl(p+q) = p+q+\gamma, \quad |\gamma| \le 2|p+q|\varepsilon. \tag{29}$$

An error of the product of two quaternions is bounded as follows [22]:

Lemma 9 ([22], Lemma 1). *Let $p,q \in \mathbb{H}$. Then*

$$|fl(pq)-pq| \le (5.75\varepsilon+\varepsilon^2)|p||q|. \tag{30}$$

Using (28) and (30), we obtain the following error bound for the dot product of two vectors of quaternions:

Lemma 10. *Let $p,q \in \mathbb{H}^n$, that is, $p=(p_1,p_2,\ldots,p_n)$ and $q=(q_1,q_2,\ldots,q_n)$, where p_i, $q_i \in \mathbb{H}$ for $i=1,\ldots,n$. Let $|p| \equiv (|p_1|,|p_2|,\ldots,|p_n|)$ and $|q| \equiv (|q_1|,|q_2|,\ldots,|q_n|)$ denote the corresponding vectors of component-wise absolute values. Then*

$$|fl(p\cdot q)-p\cdot q| \le (2n+5.75)\varepsilon|p|\cdot|q|+\mathcal{O}(\varepsilon^2). \tag{31}$$

Proof. From (30), for each n it follows that

$$fl(\bar{p}_n\cdot q_n) = \bar{p}_n q_n+\zeta_n, \quad |\zeta_n| \le 5.75\varepsilon|p_n||q_n|+\mathcal{O}(\varepsilon^2). \tag{32}$$

The proof is by induction on n. For $n=1$, the bound (31) follows from (32). Assume that the bound (31) holds for all vectors of length $n-1$.

From the assumption and (31), it follows that

$$fl\left(\sum_{i=1}^{n-1}\bar{p}_i q_i\right) = \sum_{i=1}^{n-1}\bar{p}_i q_i+\eta_{n-1}, \quad |\eta_{n-1}| \le (2(n-1)+5.75)\varepsilon\sum_{i=1}^{n-1}|p_i||q_i|+\mathcal{O}(\varepsilon^2). \tag{33}$$

Using (29)–(33), we have

$$\begin{aligned} fl\left(\sum_{i=1}^{n} \bar{p}_i q_i\right) &= fl\left(\sum_{i=1}^{n-1} \bar{p}_i q_i + \bar{p}_n q_n\right) = fl\left(fl\left(\sum_{i=1}^{n-1} \bar{p}_i q_i\right) + fl(\bar{p}_n q_n)\right) \\ &= fl\left(\sum_{i=1}^{n-1} \bar{p}_i q_i + \eta_{n-1} + \bar{p}_n q_n + \zeta_n\right) \\ &= \sum_{i=1}^{n} \bar{p}_i q_i + \eta_{n-1} + \zeta_n + \gamma_n \equiv \sum_{i=1}^{n} \bar{p}_i q_i + \eta_n, \end{aligned}$$

where γ_n accounts for the final addition. Using (33), (32) and (29) for addition, we have

$$\begin{aligned} |\eta_n| &\leq |\eta_{n-1}| + |\zeta_n| + |\gamma_n| \leq (2(n-1) + 5.75)\varepsilon \sum_{i=1}^{n-1} |p_i||q_i| + 5.75\varepsilon |p_n||q_n| \\ &\quad + 2\left|\sum_{i=1}^{n} \bar{p}_i q_i + \eta_{n-1} + \zeta_n\right|\varepsilon + \mathcal{O}(\varepsilon^2) \\ &\leq 2(n-1)\varepsilon \sum_{i=1}^{n-1} |p_i||q_i| + 2\varepsilon \sum_{i=1}^{n} |p_i||q_i| + 5.75\varepsilon \sum_{i=1}^{n-1} |p_i||q_i| + 5.75\varepsilon |p_n||q_n| + \mathcal{O}(\varepsilon^2) \\ &\leq 2n\varepsilon \sum_{i=1}^{n} |p_i||q_i| + 5.75\varepsilon \sum_{i=1}^{n} |p_i||q_i| + \mathcal{O}(\varepsilon^2), \end{aligned}$$

as desired. □

We can extend the above lemma for matrices of quaternions as follows:

Corollary 2. *Let $A, B \in \mathbb{H}^{n\times n}$ be matrices of quaternions and ε and let $|A|$ and $|B|$ denote the corresponding matrices of component-wise absolute values. Then*

$$|fl(A \cdot B) - A \cdot B| \leq (2n + 5.75)\varepsilon |A| \cdot |B| + \mathcal{O}(\varepsilon^2). \tag{34}$$

3.3. Error Bounds for Algorithms 2 and 3

Since the errors of basic operations with quaternions from Section 3.2 are of the same order as the errors in the real or complex cases, we can use these bounds to analyze Algorithms 2 and 3. To estimate the accuracy of the computed eigenvalues, these error bounds can further be plugged into the perturbation bounds of Section 3.1 (Theorems 2 and 3, Corollary 1).

However, since the algorithms use deflation, this analysis would be too cumbersome and useless. Instead, we can estimate the accuracy of the computed eigenvalues a posteriori by using residual bounds from Theorems 4 and 5 and inserting those bounds into Theorem 2, Corollary 1 or Theorem 3, respectively.

For example, let $(\tilde{\mu}, \tilde{x})$ be the computed eigenpair of the matrix A, where $\tilde{\mu}$ is in the standard form and $\|\tilde{x}\|_2 = 1$. Then, we can compute the residual r as in Theorems 2 and 4; this implies that

$$\min_{\lambda_i \in \Lambda_s(A)} \{|\lambda_i - \tilde{\mu}|\} \leq \kappa(X)\|r\|_2.$$

We can use the bound effectively if the matrix is diagonalizable and we can approximate the condition of the eigenvector matrix $\kappa(X)$ by the condition of the computed eigenvector matrix $\tilde{X}$.

If we computed all eigenvalues and all eigenvectors of a diagonalizable matrix, $\tilde{\Lambda} = \mathrm{diag}(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n)$ and $\tilde{X}$, respectively, then we can compute the residual R as in Theorem 5. Inserting the bound for $\|\Delta A\|_2$ from Theorem 5 into Theorem 2 yields

$$\max_j \min_{\lambda_i \in \Lambda_s(A)} \{|\lambda_i - \tilde{\lambda}_j|\} \leq \kappa(\tilde{X})\|R\|_2 \|(\tilde{X}^*\tilde{X})^{-1}\tilde{X}^*\|_2.$$

If the matrix is normal or Hermitian, then, instead of Theorem 2, we can use Corollary 1 of Theorem 3, yielding sharper bounds.

We demonstrate the efficacy of the bounds in the experiments in the next section.

4. Numerical Results

In this section, we present the results of tests performed on several sets of moderate-size random matrices ($n = 10, 20, 40, 100$). For the smaller dimensions ($n \leq 20$), the actual errors are computed by comparing the eigenvalues computed using Julia's 64-bit arithmetic (`Float64`) with the eigenvalues computed using Julia's type `BigFloat`. When using `BigFloat` type, all operations are computed with a 512 bit mantissa. This is equivalent to computing with some 77 decimal digits, that is, with the machine precision equal to $\varepsilon \approx 1.7272 \times 10^{-77}$.

Experiments were performed on an Intel Omen computer with Intel(R) Core(TM) i7-8700K CPU @ 3.70 GHz processor and 16 GB RAM.

Errors are computed as

$$\max_i \frac{|\lambda_i^{Float64} - \lambda_i^{BigFloat}|}{|\lambda_i^{BigFloat}|},$$

where care was taken to properly order the eigenvalues. In each case, the residual R was computed in a standard manner as in Theorem 5 and the error bound was computed by plugging in ΔA from Theorem 5 and the condition of the computed eigenvector matrix $\tilde{X}$ into Theorem 2. The Julia notebooks used to run the experiments and make graphs and tables in this section are located in the GitHub repository [21] (see the files `ED_Arrow.jl`, `ED_DPRk.jl` and `Plotting.jl`).

The results for arrow matrices, displayed in Figure 1 and Table 1, are obtained as follows: we generated 10 random quaternionic arrow matrices of each size and computed their eigenvalue decompositions using Algorithm 2 and Algorithm 1 for comparison.

The results for DPRk matrices, displayed in Figure 2 and Table 2, are obtained by generating 10 random quaternionic DPRk matrices of each size, with $k \in \{2, 3, 4\}$ and we computed their eigenvalue decompositions using Algorithm 3 and Algorithm 1 for comparison. In all examples, the tolerance was set to 1×10^{-12}.

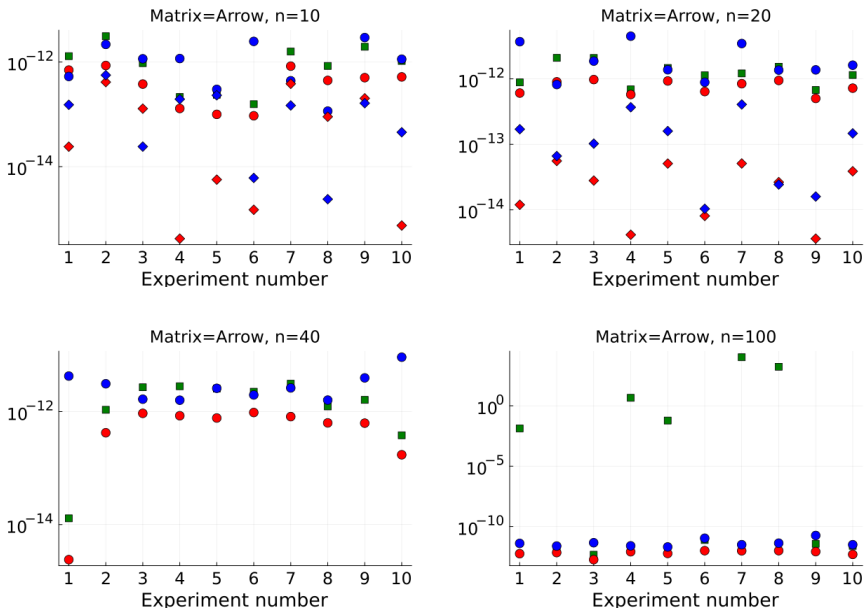

Figure 1. Error bounds (green squares), residuals and actual errors computed by Algorithm 2 (red dots and diamonds, respectively) and residuals and actual errors computed by Algorithm 1 (blue dots and diamonds, respectively). For $n = 40$ and $n = 100$, the actual errors are not computed.

Table 1. Mean number of iterations per eigenvalue and mean total running times for arrow matrices of orders $n = 10, 20, 40, 100$, using Algorithm 2 (RQIds) and Algorithm 1 (QR), respectively.

Mean Number of Iterations per Eigenvalue		**Mean Total Running Time (s)**		
n	**RQIds**	***n***	**RQIds**	**QR**
10	8	10	0.00081	0.00079
20	9	20	0.0026	0.011
40	16	40	0.014	0.039
100	32	100	0.17	0.47

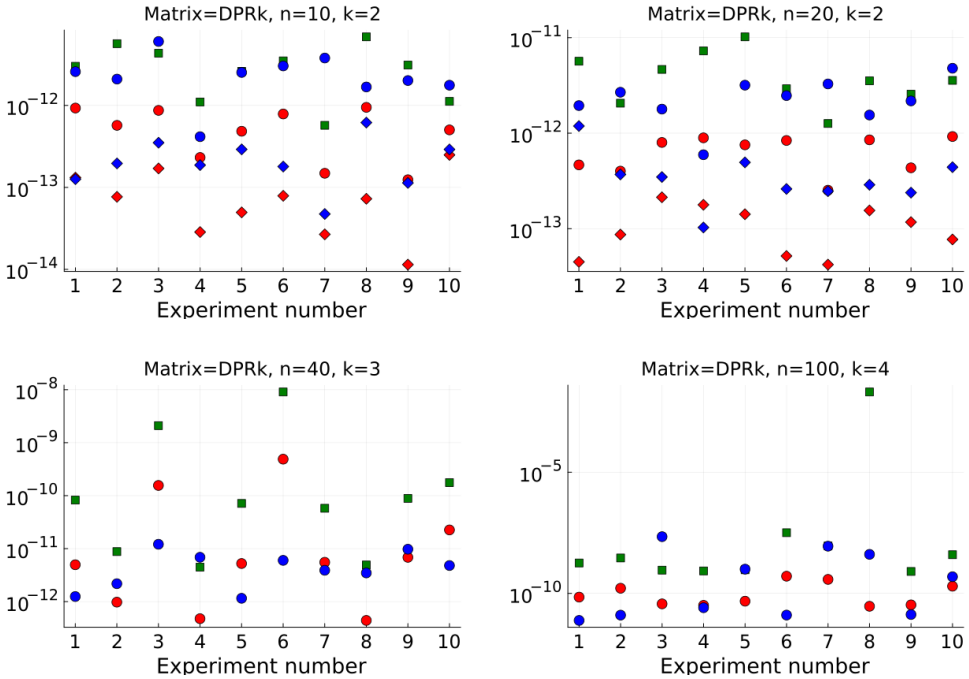

Figure 2. Error bounds (green squares), residuals and actual errors computed by Algorithm 2 (red dots and diamonds, respectively) and residuals and actual errors computed by Algorithm 3 (blue dots and diamonds, respectively). For $n = 40$ and $n = 100$ the actual errors are not computed.

Table 2. Mean number of iterations per eigenvalue and mean total running times for DPRk matrices of orders $n = 10, 20, 40, 100$, using Algorithm 3 (RQIds) and Algorithm 1 (QR), respectively.

Mean number of Iterations per Eigenvalue			**Mean Total Running Time (s)**			
n	***k***	**RQIds**	***n***	***k***	**RQIds**	**QR**
10	2	7	10	2	0.0018	0.00075
20	2	9	20	2	0.0077	0.011
40	3	16	40	3	0.031	0.071
100	4	27	100	4	0.25	0.85

5. Discussion and Conclusions

From Figures 1 and 2, we conclude that the errors approximately coincide with the tolerance and are smaller than the respective residuals, which are, in turn, smaller, but well approximated by the computed error bounds. Since the computations using `BigFloat` type are slow, the actual errors are computed only for the dimensions $n = 10$ and $n = 20$. We can also conclude that residuals and error of our methods from Algorithms 2 and 3 are similar to (or even slightly smaller than) those of the quaternion QR method from Algorithm 1.

In Tables 1 and 2, we see that the running time of our algorithms grows approximately with the square of the dimension, as expected by the $O(n^2)$ algorithm, while the running

time for the quaternion QR method increases with the cube of the dimension, as expected from the general $O(n^3)$ algorithm.

The key contributions of the paper are the following:

- Efficient algorithms for computing eigenvalue decompositions of arrow and DPRk matrices of quaternions;
- The algorithms require $O(n^2)$ arithmetic operations, n being the order of the matrix;
- Algorithms have proven error bounds;
- Experiments demonstrate that the computable residual is a good estimate of actual errors;
- Experiments demonstrate that actual errors are even smaller than predicted by the residuals;
- In all experiments, errors and residuals are of the order of tolerance from Algorithms 2 and 3;
- Experiments demonstrate that Rayleigh Quotient Iteration with double-shifts is efficient for non-Hermitian matrices;
- Algorithms 2 and 3 compare favorably in terms of accuracy and speed to the quaternion QR method for general matrices from Algorithm 1.

Author Contributions: Conceptualization, N.J.S. and I.S.; formal analysis, T.C., N.J.S. and I.S.; funding acquisition, I.S.; investigation, T.C., N.J.S. and I.S.; project administration, I.S.; resources, T.C., N.J.S. and I.S.; software, T.C., N.J.S. and I.S.; writing—original draft, T.C., N.J.S. and I.S.; writing—review & editing, T.C., N.J.S. and I.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been fully supported by the Croatian Science Foundation under project IP-2020-02-2240.

Data Availability Statement: The notebooks which contain the codes for Algorithms 2 and 3 and are used to run the experiments and make graphs and tables from Section 4 are located in the GitHub repository https://github.com/ivanslapnicar/MANAA/, accessed on 9 February 2024.

Acknowledgments: The authors wish to express their thanks to the editor and the reviewers for their succinct and valuable comments.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bunse-Gerstner, A.; Byers, R.; Mehrmann, V. A quaternion QR algorithm. *Numer. Math.* **1989**, *55*, 83–95. [CrossRef]
2. Ahmad, S.S.; Ali, I.; Slapničar, I. Perturbation analysis of matrices over a quaternion division algebra. *Electron. Trans. Numer. Anal.* **2021**, *54*, 128–149. [CrossRef]
3. Baker, A. Right eigenvalues for quaternionic matrices: A topological approach. *Linear Algebra Appl.* **1999**, *286*, 303–309. [CrossRef]
4. Farenick, D.R.; Pidkowich, B.A. The spectral theorem in quaternions. *Linear Algebra Appl.* **2003**, *371*, 75–102. [CrossRef]
5. Jia, Z.; Wang, Q.; Pang, H.K.; Zhao, M. Computing Partial Quaternion Eigenpairs with Quaternion Shift. *J. Sci. Comput.* **2023**, *97*, 41:1–41:21. [CrossRef]
6. Lee, H.C. Eigenvalues and canonical forms of matrices with quaternion coefficients. *Proc. R. Irish. Acad.* **1949**, *52*, 253–260.
7. De Leo, S.; Scolarici, G. Right eigenvalue equation in quaternionic quantum mechanics. *J. Phys. A* **2000**, *33*, 2971–2995. [CrossRef]
8. Tisseur, F.; Meerbergen, K. Quadratic eigenvalue problem. *SIAM Rev.* **2001**, *43*, 235–286. [CrossRef]
9. Golub, G.H.; Van Loan, C.F. *Matrix Computations*, 4th ed.; The John Hopkins University Press: Baltimore, MD, USA, 2013.
10. Najafi, H.S.; Edalatpanah, S.A.; Gravvanis, G.A. An efficient method for computing the inverse of arrowhead matrices. *Appl. Math. Lett.* **2014**, *33*, 1–5. [CrossRef]
11. Jakovčević Stor, N.; Slapničar, I.; Barlow, J.L. Accurate eigenvalue decomposition of real symmetric arrowhead matrices and applications. *Linear Algebra Appl.* **2015**, *464*, 62–89. [CrossRef]
12. Jakovčević Stor, N.; Slapničar, I.; Barlow, J.L. Forward stable eigenvalue decomposition of rank-one modifications of diagonal matrices. *Linear Algebra Appl.* **2015**, *487*, 301–315. [CrossRef]
13. Saad, Y. *Numerical Methods for Large Eigenvalue Problems*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2011. [CrossRef]
14. Watkins, D.S. Unsymmetric Matrix Eigenvalue Techniques. In *Handbook of Linear Algebra*; Hogben, L., Ed.; CRC Press: Boca Raton, FL, USA, 2014; Chapter 56, pp. 1–12. [CrossRef]
15. Hamilton, W.R., Sr. *Lectures on Quaternions*; Hodges and Smith: London, UK, 1853.

16. Hamilton, W.R., Sr. *Elements of Quaternions*; Longmans, Green and Co.: London, UK, 1866.
17. Sudbery, A. Quaternionic analysis. *Math. Proc. Camb. Philos. Soc.* **1979**, *85*, 199–225. [CrossRef]
18. The Julia Language. Version 1.10.2. Available online: http://julialang.org/ (accessed on 9 February 2024).
19. Quaternions.jl. Version 0.7.4. Available online: https://github.com/JuliaGeometry/Quaternions.jl (accessed on 9 February 2024).
20. Jakovčević Stor, N.; Slapničar, I. Fast multiplication, determinants and inverses of arrowhead and diagonal-plus-rank-one matrices over associative fields. *arXiv* **2022**, arXiv:2212.10966.
21. Matrix Algorithms for Non-Commutative Associative Algebras. Available online: https://github.com/ivanslapnicar/MANAA/ (accessed on 9 February 2024).
22. Joldeş, M.; Muller, J.-M. Algorithms for manipulating quaternions in floating-point arithmetic. In Proceedings of the 2020 IEEE 27th Symposium on Computer Arithmetic (ARITH), Portland, OR, USA, 7–10 June 2020; pp. 48–55. [CrossRef]