Finding Eigenvalues of Unitary Matrices

By

Peidi Gu

Submitted to the graduate degree program in Mathematics and the
Graduate Faculty of the University of Kansas
in partial fulfillment of the requirements for the degree of
Master of Art

Hongguo Xu, Chairperson

Committee members

Xuemin Tu

Erik Van Vleck

Date defended: December 17, 2013

The Dissertation Committee for Peidi Gu certifies
that this is the approved version of the following dissertation :

Finding Eigenvalues of Unitary Matrices

Hongguo Xu, Chairperson

Date approved: December 17, 2013

Abstract

The study introduces methods of finding eigenvalues for unitary matrices and pencils. Bunse-Gerstner and Elsner([2]) proposed an algorithm of using the Schur parameter pencil to solve eigenproblems for unitary matrices and pencils. This thesis reviews the Schur parameter pencil algorithm. The method is divided into two phases: Reducing a unitary pencil to a Schur parameter form and QR-type shifted iteration. The algorithm is proved to be backward stable and more efficient than the standard QR/QZ algorithm. However, during the process of reduction, norms of vectors are frequently compared for numerical stability, which causes a lot of extra work for computations. Based on the idea in [8], we introduce a modified Schur parameter algorithm to avoid such frequent comparison. The modified algorithm is still divided into two phases similar to the one in [2]. A detailed reduction process and shifted iteration are described in this thesis.

Contents

Chapter 0

Definitions and Notations

0.1 Notations

C^n: n dimensional complex vector space

$C^{m\times n}$: an $m \times n$ matrix in the complex field

$\|x\|$: 2-norm of a vector x

$\bar{c}$: conjugate of the complex number c

e_1: a column vector with 1 at the first component and 0's at others

I: identity matrix

$\mathbf{H}^*$: the complex conjugate transpose of $\mathbf{H} \in \mathbf{C}^{n\times n}$

a_{i*}: the ith row of matrix **A**

a_{*i}: the ith column of matrix **A**

v^T: the transpose of a vector v

v^*: the conjugate transpose of a vector v

$A := B$: assign the value of B to A

$\mathbf{A}(n:m, i:j)$: submatrix of $\mathbf{A}$ by taking the intersection of the entries from nth row to mth row and from ith column to jth column

$\mathbf{A}(n, i:j)$: vector formed by entries from the intersection of the nth row and from ith column to jth column

$\mathbf{A}(i:j, n)$: vector formed by entries from the intersection of the nth column and from ith row to jth row

$\mathbf{A}(n, m)$: the (n, m)th entry of matrix $\mathbf{A}$

0.2 Definitions

0.2.1 Unitary Matrix

A matrix $\mathbf{U} \in \mathbf{C}^{n \times n}$ is a unitary matrix if

$$\mathbf{U}^*\mathbf{U} = \mathbf{U}\mathbf{U}^* = \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{U}^*$ is the complex conjugate transpose of $\mathbf{U}$.

Properties of unitary matrices:

if $\mathbf{U} \in C^{n \times n}$ is a unitary matrix, then:

1. $\mathbf{U}$ and $\mathbf{U}^*$ are invertible,
2. $\mathbf{U}^{-1} = \mathbf{U}^*$ and $(\mathbf{U}^*)^{-1} = \mathbf{U}$,
3. $\mathbf{U}$ is unitary if and only if $\mathbf{U}^*$ is unitary,
4. the columns of $\mathbf{U}$ form an orthonormal basis of C^n with respect to the standard inner product,
5. the rows of $\mathbf{U}$ form an orthonormal basis of C^n with respect to the standard inner product,

6. the eigenvalues of **U** lie on the unit circle in the complex plane.

0.2.2 Hermitian Matrix

A matrix **H** is a Hermitian matrix if $\mathbf{H} = \mathbf{H}^*$.

Example 1 *a Hermitian matrix.*

$$\begin{bmatrix} 1 & 1+i & i & 2-i \\ 1-i & 3 & 5 & 4i \\ -i & 5 & 6 & 0 \\ 2+i & -4i & 0 & 7 \end{bmatrix}$$

The diagonal entries of a Hermitian matrix are always real.

0.2.3 Hessenberg Matrix

An **upper Hessenberg matrix** has all zero entries below the subdiagonal and a **lower Hessenberg matrix** has all zero entries above the first super-diagonal.

Example 2 *An upper Hessenberg matrix:*

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 0 & 9 & 10 & 11 \\ 0 & 0 & 12 & 13 \end{bmatrix}$$

Example 3 *A lower Hessenberg matrix:*

$$\begin{bmatrix} 1 & 5 & 0 & 0 \\ 2 & 6 & 9 & 0 \\ 3 & 7 & 10 & 12 \\ 4 & 8 & 11 & 13 \end{bmatrix}$$

A matrix that is both upper and lower Hessenberg is called **tridiagonal**.

Example 4 *A tridiagonal matrix:*

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 5 & 0 \\ 0 & 6 & 7 & 8 \\ 0 & 0 & 9 & 10 \end{bmatrix}$$

0.2.4 Householder Transformations

Given a vector $v \in C^n, v \neq 0$, a **Householder transformation** is a linear transformation that reflects all the vectors in the space C^n across the hyperplane H orthogonal to v. It is also called a **Householder vector**. This linear transformation is given by a Householder matrix:

$$\mathbf{P} = \mathbf{I} - 2\left(\frac{vv^*}{v^*v}\right)$$

Properties of a Householder matrix:

1. $\mathbf{P}$ is Hermitian: $\mathbf{P} = \mathbf{P}^*$
2. $\mathbf{P}$ is unitary: $\mathbf{P}^{-1} = \mathbf{P}^*$
3. $\mathbf{P}^2 = \mathbf{I}$

Suppose $0 \neq x = [x_1, x_2, \cdots, x_n]^T \in C^n$. One can construct a Householder matrix $\mathbf{P}$ that transforms x to a vector parallel to e_1. The construction can be described as follows. Suppose $x_1 = re^{i\theta}, r = |x_1| > 0$. Define

$$v = x \pm e^{i\theta}\|x\|e_1, \qquad \beta = \frac{2}{v^*v}.$$

Then

$$v = \begin{bmatrix} re^{i\theta} \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \pm \begin{bmatrix} \|x\|e^{i\theta} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} (r \pm \|x\|)e^{i\theta} \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \tag{1}$$

and using $r = |x_1|$,

$$\begin{aligned} v^*v = \|v\|^2 &= (r \pm \|x\|)^2 + \|x_2\|^2 + \cdots + \|x_n\|^2 && (2)\\ &= r^2 \pm 2r\|x\| + \|x\|^2 + \|x_2\|^2 + \cdots + \|x_n\|^2 && (3)\\ &= 2\|x\|^2 \pm 2r\|x\| && (4)\\ &= 2\|x\|(\|x\| \pm r). && (5) \end{aligned}$$

Hence

$$\beta = \frac{2}{v^*v} = \frac{2}{2\|x\|(\|x\| \pm r)} = \frac{1}{\|x\|(\|x\| \pm r)}. \tag{6}$$

For the corresponding Householder matrix $\mathbf{P} = I - \beta vv^*$, we have

$$\mathbf{P}x = (\mathbf{I} - \beta vv^*)x = x - (\beta v^*x)v.$$

Because

$$v^*x = \begin{bmatrix} (r \pm \|x\|)e^{i\theta} \\ x_2 \\ \vdots \\ x_n \end{bmatrix}^* \begin{bmatrix} re^{i\theta} \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \|x\|(\|x\| \pm r) = 1/\beta, \tag{7}$$

one has

$$\mathbf{P}x = x - v = x - (x \pm e^{i\theta}\|x\|e_1) = \mp e^{i\theta}\|x\|e_1 \tag{8}$$

For numerical stability reasons, we will always choose $v = x + e^{i\theta}\|x\|e_1$ so that v has a bigger norm. If $x_1 = 0$, we choose $v = x + \|x\|e_1$.

Algorithm 0.2.1 *Householder Transformation*

Suppose $x \in C^n$, $x \neq 0$, $x_1 = re^{i\theta}$ *with* $r \geq 0$. *This algorithm computes the Householder vector* v, *such that* $\mathbf{P} = I - \beta vv^*$ *and* $\mathbf{P}x = -e^{i\theta}\|x\|e_1$

1. *IF* $r > 0$, *THEN* $v = x + e^{i\theta}\|x\|e_1$; *ELSE* $v = x + \|x\|e_1$

2. $\beta = 2/(v^*v)$

0.2.5 Givens Rotation

A Givens rotation is given by a matrix of the form [3]:

$$\mathbf{G}(i,j,c,s) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\bar{s} & \cdots & \bar{c} & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

where c, s satisfy $|c|^2 + |s|^2 = 1$, and $c, s, \bar{c}, -\bar{s}$ locate at the intersections of ith and jth rows and columns.

If we multiply $\mathbf{G}(i,j,c,s)$ to the left of a matrix $\mathbf{A}$, then only the ith and jth rows of $\mathbf{A}$ would be affected, while if multiply $\mathbf{G}(i,j,c,s)$ to the right of $\mathbf{A}$, then only the ith and jth columns of $\mathbf{A}$ would be affected.

Givens rotations can be constructed to annihilate an element of a vector.

Example 5 *Given a* $2-$*dimensional vector* $\begin{bmatrix} a \\ b \end{bmatrix} \in C^2$, *let*

$$\mathbf{G}(c,s) = \begin{bmatrix} c & s \\ -\bar{s} & \bar{c} \end{bmatrix}$$

with $r = \sqrt{|a|^2 + |b|^2}$, $c = \frac{\bar{a}}{r}$, $s = \frac{\bar{b}}{r}$.

Then

$$\mathbf{G}(c,s) \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

With the current formulas, the computation of r may cause overflow or underflow. The following algorithm provides a practical way to compute c, s.

Algorithm 0.2.2 *Givens Rotation*

Given $x = [x(1), x(2)]^T \in C^2$, *compute* c, s *for the* 2×2 *Givens matrix* $\mathbf{G}$ *such that* $\mathbf{G}x = re_1$, *where* $r = \|x\|$.

If $|x(1)| \geq |x(2)|$

$t = \frac{\overline{x(2)}}{|x(1)|}$;

$r = \sqrt{1 + |t|^2}$;

$c = \frac{\overline{x(1)}}{|x(1)|r}$;

$s = \frac{t}{r}$;

Else

$t = \frac{\overline{x(1)|}}{|x(2)|}$;

$r = \sqrt{1 + |t|^2}$;

$s = \frac{\overline{x(2)}}{|x(2)|r}$;

$c = \frac{t}{r}$;

If $x = 0$*, we simply set* $\mathbf{G} = \mathbf{I}$.

0.2.6 QR Decomposition

The QR decomposition of an $m \times n$ matrix $\mathbf{A} \in C^{m \times n}$ is given by

$$\mathbf{A} = \mathbf{QR}$$

where $\mathbf{Q} \in C^{m \times m}$ is unitary and $\mathbf{R} \in C^{m \times n}$ is upper triangular.

0.2.7 Backward Stability

An algorithm $\tilde{f}$ for a problem f is backward stable if for each $0 \neq x \in X$, where X is a set on which f and $\tilde{f}$ are defined,

$$\tilde{f} = f(\tilde{x}) \text{ for some } \tilde{x} \in X \text{ with } \frac{\|\tilde{x} - x\|}{\|x\|} = \mathcal{O}(\varepsilon_{machine}),$$

where $\varepsilon_{machine}$ is the machine epsilon ([9]). In words,

a backward stable algorithm gives the exact answer to the problem
with input data slightly perturbed from the original data.

Chapter 1

Introduction

1.1 Overview

Finding eigenvalues of matrices has been a popular topic in matrix computations. The QR algorithm is one of the most powerful eigenvalue methods. It reduces a given matrix to an upper Hessenberg matrix with a unitary similarity transformation, then uses a sequence of unitarily similarity transformations, called QR iterations, to reduce it to an upper triangular matrix. In this way, a Schur form will be computed. The QZ algorithm, a generalization of the QR algorithm, is designed for computing a generalized Schur form of a matrix pencil $\mathbf{A} - \lambda\mathbf{B}$. Other eigenvalue methods include power iteration, inverse iteration and Rayleigh quotient iteration.

For the eigenvalue problem of a unitary pencil, which is a matrix pencil $\mathbf{U} - \lambda\mathbf{V}$ where both $\mathbf{U}$ and $\mathbf{V}$ unitary, one may apply the QZ algorithm. Unfortunately, the QZ algorithm does not respect the unitary structures. In [2], Angelika Bunse-Gerstner and Ludwig Elsner proposed a variant of the QZ algorithm. The main difference is that, instead of using a Hessenberg-triangular pencil as the condensed form, they proposed a sparer condensed form called the Schur parameter pencil. With carefully designed iterations, the pencil will eventually converge to a diagonal-diagonal pencil. This algorithm uses unitary structures and is more efficient and reliable. Recently, another Schur parameter pencil reduction procedure was proposed for unitary matrices [8]. The proposed

procedure in [8] is different from that in [2]. The former procedure is more straightforward and logically simpler.

1.2 Goal

We will review the methods introduced by Bunse-Gerstner and Elsner in [2] . We then modify the algorithm by generalizing the strategy used in [8] to the pencil case.

Let us give more details about these two methods. In [2], the authors directly reduce a unitary pencil $\mathbf{U}-\lambda\mathbf{V}$to a Schur parameter pencil $\mathbf{G}_o-\lambda\mathbf{G}_e$ by finding appropriate $n\times n$ unitary matrices $\mathbf{P},\mathbf{Q}$, such that $\mathbf{G}_o=\mathbf{QUP}^*$ and $\mathbf{G}_e=\mathbf{QVP}^*$. Then they apply implicit shifted QR-type iterations to $\mathbf{G}_o-\lambda\mathbf{G}_e$ so that $\mathbf{G}_o-\lambda\mathbf{G}_e$ eventually converges to a diagonal pencil. In contrast to the method in [2], we modify the Schur parameter form reduction procedure in their algorithm. We use a series of Householder reflectors and Givens matrices to simultaneously transform both $\mathbf{U}$ and $\mathbf{V}$ into the identity matrix, meanwhile $\mathbf{G}_o$ and $\mathbf{G}_e$ are recovered from the transformation matrices. We also apply the same idea to the QR-type iterations.

1.3 Outline

This thesis is organized as follows. In Chapter 2, we will briefly introduce the concepts and related properties of eigenvalues and eigenvectors. We will also discuss briefly the standard QR algorithm and QZ algorithm, which are closely related to our study.

In Chapter 3, we will review the algorithm in [2]. It starts with reducing a given unitary pencil $\mathbf{U}-\lambda\mathbf{V}$ to a Schur parameter pencil,

$$\mathbf{Q}(\mathbf{U}-\lambda\mathbf{V})\mathbf{P}^*=\mathbf{G}_o-\lambda\mathbf{G}_e$$

where $\mathbf{Q},\mathbf{P}$ are unitary matrices and $\mathbf{G}_o,\mathbf{G}_e$ are block diagonal matrices. Then QR-type iterations are performed on $\mathbf{G}_o-\lambda\mathbf{G}_e$ so that the pencil will converge to a diagonal pencil. We will also

review the algorithm provided in [2] for a single unitary matrix, or equivalently, the pencil $\mathbf{U} - \lambda \mathbf{I}$.

Chapter 4 contains our modified algorithm. we will provide our modified Schur parameter form reduction method. Instead of reducing $\mathbf{U}, \mathbf{V}$ directly to $\mathbf{G}_o, \mathbf{G}_e$, our goal is to reduce both $\mathbf{U}, \mathbf{V}$ to the identity matrix:

$$\mathbf{G}_o^* \mathbf{Q} \mathbf{U} \mathbf{P}^* = \mathbf{I}$$

$$\mathbf{Q} \mathbf{V} \mathbf{P}^* \mathbf{G}_e^* = \mathbf{I}$$

where $\mathbf{Q}, \mathbf{P}$ are unitary matrices and $\mathbf{G}_o - \lambda \mathbf{G}_e$ is a Schur parameter pencil. Our procedure simplifies the one in [2] by avoiding frequent comparison between the norms of vectors. The procedure for the special case for a single unitary matrix $\mathbf{U}$ will also be provided, which can be considered as a variant of the procedure given in [8]. We will also introduce modified QR-type iteration procedures with single shift and double shifts strategies, respectively, by borrowing the idea used in the modified Schur parameter pencil reduction process.

In Chapter 5, a conclusion is drawn about our study, and future work is discussed.

Chapter 2

The Eigenvalue Problem

2.1 Eigenvalue and Eigenvectors

Definition 2.1.1 *Given a square matrix* $\mathbf{A} \in C^{n\times n}$, $\lambda \in C$ *is an eigenvalue of* $\mathbf{A}$ *if* $\exists\, x \in C^n, x \neq 0$, *such that*

$$\mathbf{A}x = \lambda x.$$

The nonzero vector x is called an eigenvector of $\mathbf{A}$ *corresponding to the eigenvalue* λ. *The characteristic polynomial of the matrix* $\mathbf{A}$ *is denoted by* $p_{\mathbf{A}}(x) = \det(xI - \mathbf{A})$.

Theorem 2.1.1 λ *is an eigenvalue of* $\mathbf{A}$ *if and only if* $p_{\mathbf{A}}(\lambda) = 0$.

Theorem 2.1.2 *If* $\mathbf{A} \in C^{n\times n}$ *is a triangular matrix (either upper-triangular or lower-triangular), then the diagonal entries of* $\mathbf{A}$ *are the eigenvalues of* $\mathbf{A}$.

Theorem 2.1.3 *If* $\mathbf{A} \in C^{n\times n}$ *and* $\mathbf{B} \in C^{n\times n}$ *are similar, i.e., there exists an invertible* $\mathbf{Q} \in C^{n\times n}$ *such that* $\mathbf{A} = \mathbf{Q}^{-1}\mathbf{B}\mathbf{Q}$, *then* $\mathbf{A}$ *and* $\mathbf{B}$ *have the same eigenvalues.*

Definition 2.1.2 *a* **Schur decomposition** *of a matrix* $\mathbf{A}$ *is a factorization*

$$\mathbf{A} = \mathbf{Q}\mathbf{T}\mathbf{Q}^*,$$

where $\mathbf{Q}$ *is unitary and* $\mathbf{T}$ *is upper-triangular. Note that by Theorem 2.1.2, the eigenvalues of* $\mathbf{A}$ *appears on the diagonal of* $\mathbf{T}$.

Theorem 2.1.4 *Every square matrix* $\mathbf{A}$ *has a Schur decomposition.*

2.2 QR Algorithm

As defined in Definition 2.1.2, a Schur decomposition of a matrix $\mathbf{A}$ is $\mathbf{A} = \mathbf{QTQ}^*$, where $\mathbf{Q}$ is unitary and $\mathbf{T}$ is upper-triangular. The eigenvalues of $\mathbf{A}$ appear on the diagonal of $\mathbf{T}$. One of the most popular methods for computing a Schur form of a matrix $\mathbf{A}$ is called the *QR algorithm*. The QR algorithm is an iterative method, which is divided into two phases:

Phase I: Hessenberg Reduction: Compute $\mathbf{H} = \mathbf{U}^*\mathbf{AU}$, where $\mathbf{H}$ is upper-Hessenberg and $\mathbf{U}$ is unitary. This reduction can be done in finite steps.

Phase II: QR Iteration: Repeat the following procedure until convergance:

(a) Determine a (low degree) polynomial $p(t)$

(b) Compute the first column of $p(\mathbf{H})$: $y = p(\mathbf{H})e_1$

(c) Determine a unitary matrix $\mathbf{V}$. such that $\mathbf{V}^*y = \beta e_1$ and $\mathbf{H} := \mathbf{V}^*\mathbf{HV}$ is again upper Hessenberg.

Normally, $\mathbf{H}$ will converge to an upper triangular matrix $\mathbf{T}$, and if all the unitary transformations are accumulated, one has $\mathbf{Q}$ as well. Then a Schur form of $\mathbf{A}$ is computed.

In principle, QR iteration can be applied directly to a matrix for computing the Schur form, which requires $\mathcal{O}(n^3)$ flops in each iteration. Reducing $\mathbf{A}$ to a Hessenberg form before the QR iteration will reduce the cost in each iteration to $\mathcal{O}(n^2)$ flops, because of the zero pattern of Hessengerg matrices. We will give more descriptions about the QR algorithm below.

Phase I: Hessenberg Reduction

The algorithm starts with a Householder reflector $\mathbf{Q}_1^*$ to eliminate the $3rd$ to nth entries the first column of $\mathbf{A} \in C^{n\times n}$. Since a similarity transformation is required, $\mathbf{Q}_1$ is applied to the right of $\mathbf{A}$ as well. Because $\mathbf{Q}_1^*$ doesn't touch the first row of $\mathbf{A}$, then when apply $\mathbf{Q}_1$ to the right of $\mathbf{A}$, the first column is not touched, which preserves the zeros in the first column of $\mathbf{A}$. Then we can proceed to the second column of $\mathbf{A}$ to eliminate the $4th$ to nth entry by another householder reflector $\mathbf{Q}_2^*$, and apply $\mathbf{Q}_2$ to the right of $\mathbf{A}$ at the same time. Continue with these steps until we get a Hessenberg form $\mathbf{H}$ of $\mathbf{A}$. The process of the reduction of a 5×5 matrix $\mathbf{A}$ is shown below [9]:

$$\mathbf{A} \quad \mathbf{Q}_1^*\mathbf{A} \quad \mathbf{Q}_1^*\mathbf{A}\mathbf{Q}_1$$

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \rightarrow \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \mathbf{0} & \times & \times & \times & \times \\ \mathbf{0} & \times & \times & \times & \times \\ \mathbf{0} & \times & \times & \times & \times \end{bmatrix} \rightarrow \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}$$

$$\mathbf{Q}_2^*\mathbf{Q}_1^*\mathbf{A}\mathbf{Q}_1 \quad \mathbf{Q}_2^*\mathbf{Q}_1^*\mathbf{A}\mathbf{Q}_1\mathbf{Q}_2 \quad \mathbf{Q}_3^*\mathbf{Q}_2^*\mathbf{Q}_1^*\mathbf{A}\mathbf{Q}_1\mathbf{Q}_2$$

$$\rightarrow \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \mathbf{0} & \times & \times & \times \\ & \mathbf{0} & \times & \times & \times \end{bmatrix} \rightarrow \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & \times & \times & \times \end{bmatrix} \rightarrow \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & \mathbf{0} & \times & \times \end{bmatrix}$$

$$\mathbf{Q}_3^*\mathbf{Q}_2^*\mathbf{Q}_1^*\mathbf{A}\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3$$

$$\rightarrow \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix} = \mathbf{H}$$

Algorithm 2.2.1 *Householder Reduction to Hessenberg Form*

Given a matrix $\mathbf{A} \in C^{n\times n}$*, this algorithm generates a sequence of Householder reflectors* $\mathbf{Q}_i$ *to reduce* $\mathbf{A}$ *to its Hessenberg form* $\mathbf{H}$ *such that* $\mathbf{A} = \mathbf{U}^*\mathbf{H}\mathbf{U}$*, where* $\mathbf{U} = \mathbf{Q}_1 \ldots \mathbf{Q}_{n-2}$ *is unitary .*

for $i = 1, \cdots, n-2$

$\hat{\mathbf{Q}}^{(i)} = houseg(\mathbf{A}(i+1:n, i))$*; %% generate a Householder reflector for* $\mathbf{A}(i+1:n, i)$

$\mathbf{Q}^{(i)} = diag(\mathbf{I}_i, \hat{\mathbf{Q}}^{(i)})$*; %% adjust the dimension of the Householder matrix for multiplication*

$\mathbf{A} := \mathbf{Q}^{(i)}\mathbf{A}(\mathbf{Q}^{(i)})^*$*;*

end

This algorithm requires $10n^3/3$ flops. Additional $4n^3/3$ flops are needed if $\mathbf{U}$ is aquired.

Phase II: QR Iteration

The QR iterations are applied to $\mathbf{H}$, which is a Hessenberg matrix obtained from Phase I. In each iteration, $\mathbf{H}^{(i)}$ is transformed into another Hessenberg matrix $\mathbf{H}^{(i+1)}$. The iteration process is shown below:

Algorithm 2.2.2 *QR Iterations*

This algorithm performs shifted QR iterations starting with $\mathbf{H}^{(0)} := \mathbf{H}$ *to generate a sequence of Hessenberg matrices* $\{\mathbf{H}^{(i)}\}$*. Normally* $\{\mathbf{H}^{(i)}\}$ *converges to a triangular matrix* $\mathbf{T}$*.*

FOR $i = 0, 1, \ldots$

$y = p_i(\mathbf{H}^{(i)})e_1$*;*

%% $p_i(t)$ *is a (lower degree) polynomial*

Compute unitary $\mathbf{Q}_0^{(i)}$ *such that* $\mathbf{Q}_0^{(i)}y = \beta_i e_1$

$\mathbf{H}^{(i)} := \mathbf{Q}_0^{(i)*}\mathbf{H}^{(i)}\mathbf{Q}_0^{(i)}$;

$\mathbf{H}^{(i+1)} = \mathbf{Q}^{(i)*}\mathbf{H}^{(i)}\mathbf{Q}^{(i)}$;

%% $\mathbf{Q}^{(i)}$ is a unitary matrix and $\mathbf{H}^{(i+1)}$is Hessenberg

END

When $p_i(t) = t$ for all i, the algorithm is called the **zero shift QR iteration**. If $p_i(t) = t - \mu_i$, we call the algorithm as the **single shift QR iteration**. Usually, we choose μ_i to be equal to $\mathbf{H}^{(i)}(n,n)$, which is called the Rayleigh quotient shift; or μ_i to be the eigenvalue of $\mathbf{H}^{(i)}(n-1:n,n-1:n)$ that is closer to $\mathbf{H}^{(i)}(n,n)$, which is called Wilkinson's shift.

For real matrices, in order to keep the algorithm in real arithmetic, especially in the case where the eigenvalues of $\mathbf{H}^{(i)}(n-1:n,n-1:n)$ are complex, the **double shifts QR iteration** is mostly used, that is when $p_i(t) = (t-\sigma_1^{(i)})(t-\sigma_2^{(i)})$. Normally, $\sigma_1^{(i)}, \sigma_2^{(i)}$ are chosen to be the eigenvalues of $\mathbf{H}^{(i)}(n-1:n,n-1:n)$, which are called Wilkinson's double shifts.

During the iteration process, at a certain step, we may have some $\mathbf{H}^{(i)}(p+1,p)$ that is sufficiently small. In this case we may set it to zero and then

$$\mathbf{H}^{(i)} = \begin{bmatrix} \mathbf{H}_{11}^{(i)} & \mathbf{H}_{12}^{(i)} \\ 0 & \mathbf{H}_{22}^{(i)} \end{bmatrix}$$

where $\mathbf{H}_{11}^{(i)}$ has dimension $p \times p$, $\mathbf{H}_{22}^{(i)}$ has dimension $(n-p)\times(n-p)$. Then the problem decouples into two smaller problems involving $\mathbf{H}_{11}^{(i)}$ and $\mathbf{H}_{22}^{(i)}$. If $p = n-1$ or $n-2$, we call this is a **deflation** procedure.

Normally $\mathbf{H}^{(i)}$ will converge to an upper triangular matrix $\mathbf{T}$. So we will have a Schur from of A. If $\mathbf{A}$ is Hermitian, $\mathbf{T}$ will be a real diagonal matrix.

2.3 QZ Algorithm

The QZ algorithm is a generalized version of the QR algorithm. It is for the generalized eigenvalue problem of a matrix pencil $\mathbf{A} - \lambda\mathbf{B}$. Here we assume $\mathbf{A}$ and $\mathbf{B}$ are both $n \times n$ and $\mathbf{B}$ is invertible.

Definition 2.3.1 *Suppose* $\mathbf{A}$ *and* $\mathbf{B}$ *are complex* $n \times n$ *matrices and* $\mathbf{B}$ *is invertible. a scalar* $x \in C$ *is an eigenvalues of the pencil* $\mathbf{A} - \lambda\mathbf{B}$ *if* x *satisfies*

$$\det(\mathbf{A} - x\mathbf{B}) = 0. \tag{2.1}$$

The set of all eigenvalues of $\mathbf{A} - \lambda\mathbf{B}$ *is denoted by* $\lambda(\mathbf{A}, \mathbf{B})$.

Similar to the QR algorithm, the QZ algorithm has two phases. In phase I, it computes unitary matrices $\mathbf{Q}$ and $\mathbf{Z}$ such that $\mathbf{H} = \mathbf{Q}^*\mathbf{A}\mathbf{Z}$ is upper Hessenberg and $\mathbf{T} = \mathbf{Q}^*\mathbf{B}\mathbf{Z}$ is upper triangular. We call it the **Hessenberg-triangular reduction process**. Phase II is the QZ iteration. Just like the QR iteration, here one QZ iteration is transform one Hessenberg-triangular pencil $\mathbf{H} - \lambda\mathbf{T}$ to another Hessenberg-triangular pencil with a shift strategy. Similarly, deflation is performed during the QZ iteration process. Eventually the pencil will converge to a triangular-triangular pencil, which will give a generalized Schur form of $\mathbf{A} - \lambda\mathbf{B}$.

Algorithm 2.3.1 *QZ Algorithm[3]*

This algorithm computes a generalized Schur form for the pencil $\mathbf{A} - \lambda\mathbf{B}$. *It computes unitary matrices* $\mathbf{Q}$ *and* $\mathbf{Z}$ *and a triangular-triangular pencil* $\mathbf{Q}^*(\mathbf{A} - \lambda\mathbf{B})\mathbf{Z}$.

Phase I. Hessenberg-triangular reduction. *Compute unitary* $\mathbf{Q} := \mathbf{Q}^{(0)}$ *and* $\mathbf{Z} := \mathbf{Z}^{(0)}$, *upper Hessenberg matrix* $\mathbf{H}^{(0)} = \mathbf{Q}^*\mathbf{A}\mathbf{Z}$, *and upper triangular matrix* $\mathbf{T}^{(0)} = \mathbf{Q}^*\mathbf{A}\mathbf{Z}$

Phase II. QZ iteration.

For $i = 1, 2, \cdots$

$\mathbf{M} = \mathbf{H}^{(i)}(\mathbf{T}^{(i)})^{-1}$;

$y = p_i(M)e_1$;

Compute a Householder matrix $\mathbf{P}_0^{(i)}$ *such that* $(\mathbf{P}^{(i)})^* y = \beta e_1$;

$\tilde{\mathbf{H}}^{(i)} = (\mathbf{P}^{(i)})^* \mathbf{H}^{(i)}$, $\tilde{\mathbf{T}}^{(i)} = (\mathbf{P}^{(i)})^* \mathbf{T}^{(i)}$; $\mathbf{Q} := \mathbf{Q}\mathbf{P}^{(i)}$

Compute unitary $\mathbf{Q}^{(i)}$ *and* $\mathbf{Z}^{(i)}$ *with* $\mathbf{Q}^{(i)} e_1 = \mathbf{P}^{(i)} e_1$, *upper Hessenberg* $\mathbf{H}^{(i+1)} := (\mathbf{Q}^{(i)})^* \tilde{\mathbf{H}}^{(i)} \mathbf{Z}^{(i)}$, *and upper triangular* $\mathbf{T}^{(i+1)} = (\mathbf{Q}^{(i)})^* \tilde{\mathbf{T}}^{(i)} \mathbf{Z}^{(i)}$;

update $\mathbf{Q} := \mathbf{Q}\mathbf{Q}^{(i)}$, $\mathbf{Z} = \mathbf{Z}\mathbf{Z}^{(i)}$

end

In the algorithm $p_i(t)$ is the same as defined in algorithm 2.2.2 for using shifts, and the choices of shifts are the same as well.

Chapter 3

Schur Parameter Pencil Algorithm for Unitary Eigenproblems

As introduced in the previous chapter, standard QR/QZ algorithm can be applied to general matrices to solve eigenvalue problems. The reduction of a matrix to a Hessenberg form needs $10n^3/3$ flops, and each iterative step needs $\mathscr{O}(n^2)$ flops. We can also apply the standard QR/QZ algorithm to unitary matrices for eigenvalue problems. However, this algorithm doesn't respect the structure of unitary matrices. It is observed that when a unitary matrix $\mathbf{U}$ is reduced to an upper Hessenberg matrix $\mathbf{H} = \mathbf{Q}_0^*\mathbf{U}\mathbf{Q}_0$, further reduction can be performed on $\mathbf{H}$ to transform it to $\mathbf{G}_o\mathbf{G}_e^*$, where both $\mathbf{G}_o, \mathbf{G}_e$ are unitary and block diagonal with block sizes being 1×1 or 2×2 (Precise definition will be given below). Based on this observation, in [1, 4, 5, 6, 7] eigenvalue algorithms were constructed and convergence analysis was done. In [2], a reduction procedure was proposed that reduces a unitary pencil directly to $\mathbf{G}_o - \lambda\mathbf{G}_e$, which is called Schur parameter pencil. Based on this reduction, a QZ-like algorithm was developed in [2].

In this chapter, we will introduce the Schur parameter pencil algorithm given in [2]. We will first describe their reduction of a general unitary pencil to a Schur parameter pencil. This process requires $8n^3/3$ flops. We will also show how to reduce a single unitary matrix to a Schur parameter pencil. In the end, we will show their implicit QR-type iteration that transforms one Schur

parameter pencil to another. One iteration requires $\mathcal{O}(n)$ flops, if unitary matrices are not updated.

The algorithm introduced in [2] can be divided into two phases:

Phase I: Schur parameter pencil reduction. Reduction from a unitary pencil to a Schur parameter pencil: Compute $\mathbf{QUP} = \mathbf{G}_o, \mathbf{QVP} = \mathbf{G}_e$.

Phase II: QR-type iteration with shifts. Repeat the following steps until both $\mathbf{G}_o, \mathbf{G}_e$ converge to diagonal forms:

(a) Determine the shift and construct a new matrix $\mathbf{G}$ according to the shift.

(b) Determine the first column of $\mathbf{G}$ and compute a householder reflector $\mathbf{Q}_0$ for the column.

(c) Compute $\mathbf{G}_o := \mathbf{Q}_0\mathbf{G}_o$ and $\mathbf{G}_e := \mathbf{Q}_0\mathbf{G}_e$.

(d) Reduce $\mathbf{G}_o, \mathbf{G}_e$ to a Schur parameter form again.

3.1 Reduction from a Unitary Pencil to a Schur Parameter Pencil

Definition 3.1.1 *[2] For $k \in \{1, \cdots, n-1\}$ and $\alpha, \beta, \eta, \xi \in C$, such that* $\begin{bmatrix} \alpha & \xi \\ \eta & \beta \end{bmatrix}$ *is unitary, define*

$$\mathbf{G}_k(\alpha, \beta, \xi, \eta) = \text{diag}(\mathbf{I}_{k-1}, \begin{bmatrix} \alpha & \xi \\ \eta & \beta \end{bmatrix}, \mathbf{I}_{n-k-1}). \tag{3.1}$$

For $\alpha, \beta \in C$ such that $|\alpha| = |\beta| = 1$ define

$$\mathbf{G}_0(\alpha, 1, 0, 0) = \text{diag}(\alpha, \mathbf{I}_{n-1}), \quad \mathbf{G}_n(1, \beta, 0, 0) = diag(\mathbf{I}_{n-1}, \beta), \tag{3.2}$$

i) An $n \times n$ matrix pencil $\mathbf{G}_o - \lambda \mathbf{G}_e$ is called Schur parameter pencil if there exists a set of parameters $\alpha_i, \beta_i, \eta_i, \xi_i \in C, i = 0, 1, \cdots, n$, where $|\beta_0| = |\alpha_n| = 1, \eta_0 = \eta_n = \xi_0 = \xi_n = 0$

and $\begin{bmatrix} \alpha_k & \xi_k \\ \eta_k & \beta_k \end{bmatrix}$ *is unitary for* $k = 0, \cdots, n$, *such that*

$$\mathbf{G}_o = \prod_{j=1}^{[\frac{n+1}{2}]} \mathbf{G}_{2j-1}(\alpha_{2j-1}, \beta_{2j-1}, \xi_{2j-1}, \eta_{2j-1}) \tag{3.3}$$

and

$$\mathbf{G}_e = \prod_{j=0}^{[\frac{n}{2}]} \mathbf{G}_{2j}(\alpha_{2j}, \beta_{2j}, \xi_{2j}, \eta_{2j}). \tag{3.4}$$

We call the parameter pairs $(\alpha_0, \beta_0), (\alpha_1, \beta_1), \cdots, (\alpha_n, \beta_n)$ *Schur parameters of* $\mathbf{G}_o - \lambda \mathbf{G}_e$ *and the parameter pairs* $(\xi_1, \eta_1), \cdots, (\xi_{n-1}, \eta_{n-1})$ *complementary Schur parameters of* $\mathbf{G}_o - \lambda \mathbf{G}_e$.

ii) We call the Schur parameter pencil $\mathbf{G}_o - \lambda \mathbf{G}_e$ *normalized, if all complementary parameters satisfy* $\xi_k = \eta_k$ *and are real and nonnegative, and if* $\alpha_0 = 1, \alpha_k = -\bar{\beta}_k$ *for* $k \in \{1, \cdots, n-1\}$. *In this case there exist* $\gamma_1, \cdots, \gamma_n \in C, \sigma_1, \cdots, \sigma_{n-1} \in R_+ \cup \{0\}$ *such that for all* $k = 1, \cdots, n-1$

$$\begin{bmatrix} -\gamma_k & \sigma_k \\ \sigma_k & \bar{\gamma}_k \end{bmatrix} = \begin{bmatrix} \alpha_k & \xi_k \\ \eta_k & \beta_k \end{bmatrix} \tag{3.5}$$

and $\gamma_n = \bar{\beta}_n$.

$\gamma_1, \cdots, \gamma_n$ *are called normalized Schur parameters and* $\sigma_1, \cdots, \sigma_{n-1}$ *normalized complementary Schur parameters. Note in this case* $\sigma_k = \sqrt{1 - |\gamma_k|^2}$ *for all k, due to the fact that* $\mathbf{G}_k(-\gamma_k, \bar{\gamma}_k, \sigma_k, \sigma_k)$ *is unitary.*

We then define

$$\mathbf{G}_k(\gamma_k) = \mathbf{G}_k(-\gamma_k, \bar{\gamma}_k, \sigma_k, \sigma_k) \qquad k = 1, \cdots, n-1$$

$$\mathbf{G}_n(\gamma_n) = \mathbf{G}_n(1, \bar{\gamma}_n, 0, 0).$$

Theorem 3.1.1 *[2] Let* $\mathbf{U}, \mathbf{V} \in C^{n \times n}$ *be unitary.*

There exist unitary $n \times n$ matrices $\mathbf{Q}$ *and* $\mathbf{P}$*, such that*

$$\mathbf{QUP}^* - \lambda \mathbf{QVP}^* = \mathbf{G}_o - \lambda \mathbf{G}_e$$

is a Schur parameter pencil. The Schur parameter pencil can be computed with a finite number of steps. If $\mathbf{U}$ *and* $\mathbf{V}$ *are real orthogonal matrices, then* $\mathbf{Q}$ *and* $\mathbf{P}$ *can be chosen to be real.*

We now describe the reduction process. Denote

$$\mathbf{P}(j,i,v) = diag(\mathbf{I}_{j-1}, \hat{\mathbf{P}}, \mathbf{I}_{n-i}) \in C^{n\times n}, \quad 1 \le j < i \le n, \quad v \in C^n, \tag{3.6}$$

where $\hat{\mathbf{P}}$ is a Householder reflector such that $\hat{\mathbf{P}}\hat{v} = \alpha e_1$ for $\hat{v} = [v_j, v_{j+1}, \cdots, v_i]^T$. The matrix $\mathbf{P}(j,i,v)$ is still a Householder reflector.

In [2], two cases of the reduction are introduced: reducing a unitary pencil $\mathbf{U} - \lambda \mathbf{V}$ to a Schur parameter pencil and reducing a unitary matrix $\mathbf{U}$, which can also be regarded as a unitary pencil as $\mathbf{U} - \lambda \mathbf{I}$, to a Schur parameter pencil.

3.1.1 Unitary Pencil Case

Consider a pencil $\mathbf{U} - \lambda \mathbf{V}$, where $\mathbf{U}, \mathbf{V} \in C^{n\times n}$ are unitary.

The first step is to determine a Householder reflector $\mathbf{P}_0 := \mathbf{P}(1, n, v_{1*}^*)$ to transform the first row of V to be parallel to e_1^T. Let

$$\mathbf{U}^{(0)} := \mathbf{UP}_0, \quad \mathbf{V}^{(0)} := \mathbf{VP}_0. \tag{3.7}$$

Then $\mathbf{U}^{(0)}$ and $\mathbf{V}^{(0)}$ are of the form:

$$\mathbf{U}^{(0)} = \begin{bmatrix} \times & \times & \cdots & \times \\ \times & \times & \cdots & \times \\ \vdots & \vdots & \cdots & \vdots \\ \times & \times & \cdots & \times \end{bmatrix}, \mathbf{V}^{(0)} = \begin{bmatrix} \times & 0 & \cdots & 0 \\ 0 & \times & \cdots & \times \\ \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \cdots & \times \end{bmatrix}, \tag{3.8}$$

where $\times$ denotes possibly non-zero entries. Because $\mathbf{V}$ is a unitary matrix, $|\mathbf{V}(1,1)| = 1$. Thus zeros in the first column of $\mathbf{V}$ below $\mathbf{V}(1,1)$ are automatically created. If $n = 2$, then $\mathbf{U}^{(0)} - \lambda \mathbf{V}^{(0)}$ is already a Schur parameter pencil. Otherwise, we proceed to the next step.

In order to transform $\mathbf{U}^{(0)} - \lambda \mathbf{V}^{(0)}$ into a Schur parameter form, we need to create the first 2×2 block in $\mathbf{G}_o$. First of all, let $\mathbf{P}_1 = \mathbf{P}(2, n, u_{*1})$ which eliminates the entries from 3 to n in the first column of $\mathbf{U}$, that is

$$\tilde{\mathbf{U}} := \mathbf{P}_1 \mathbf{U}^{(0)} = \mathbf{P}_1 \mathbf{U} \mathbf{P}_0 \tag{3.9}$$

$$\tilde{\mathbf{V}} := \mathbf{P}_1 \mathbf{V}^{(0)} = \mathbf{P}_1 \mathbf{V} \mathbf{P}_0 \tag{3.10}$$

as $\mathbf{P}_1$ only affect the entries from row 2 to row n, then the form of $\mathbf{V}^{(0)}$ is not affected, then

$$\tilde{\mathbf{U}} = \begin{bmatrix} \times & \times & \cdots & \times \\ \times & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \cdots & \times \end{bmatrix}, \tilde{\mathbf{V}} = \begin{bmatrix} \times & 0 & \cdots & 0 \\ 0 & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \cdots & \times \end{bmatrix} \tag{3.11}$$

Next, in order to eliminate the entries from $3rd$ column to nth column in the first and second rows of $\tilde{\mathbf{U}}$, compare the norm of $\tilde{\mathbf{U}}(1,1)$ and $\tilde{\mathbf{U}}(2,1)$ for numerical stability reasons. As the previous steps are associated with unitary matrix multiplications, $\tilde{\mathbf{U}}$ is still unitary. Thus, $|\tilde{u}_{*1}| = 1$, which means at least one of $\tilde{\mathbf{U}}(1,1)$ and $\tilde{\mathbf{U}}(2,1)$ is nonzero. If $|\tilde{\mathbf{U}}(1,1)| \geq |\tilde{\mathbf{U}}(2,1)|$, then $\|\tilde{\mathbf{U}}(1,2:n)\| \leq \|\tilde{\mathbf{U}}(2,2:n)\|$. We take the vector with larger norm, that is $\tilde{\mathbf{U}}(2,2:n)$ to construct a Householder

reflector $\mathbf{P}_2 = \mathbf{P}(2,n,\tilde{u}_{2*}^*)$. Otherwise, if $|\tilde{\mathbf{U}}(1,1)| \leq |\tilde{\mathbf{U}}(2,1)|$, then $\|\tilde{\mathbf{U}}(1,2:n)\| \geq \|\tilde{\mathbf{U}}(2,2:n)\|$. We take the vector $\tilde{\mathbf{U}}(1,2:n)$ to construct the householder reflector $\mathbf{P}_2 = \mathbf{P}(2,n,\tilde{u}_{1*}^*)$. That is

$$\mathbf{U}^{(1)} = \tilde{\mathbf{U}}\mathbf{P}_2 = \begin{bmatrix} \times & \times & 0 & \cdots & 0 \\ \times & \times & 0 & \cdots & 0 \\ 0 & 0 & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \times & \cdots & \times \end{bmatrix}, \mathbf{V}^{(1)} = \tilde{\mathbf{V}}\mathbf{P}_2 = \begin{bmatrix} \times & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \cdots & \times \\ 0 & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \times & \cdots & \times \end{bmatrix} \tag{3.12}$$

The additional zeros in $\mathbf{U}^{(1)}$ is generated automatically because of the unitary structure of $\mathbf{U}^{(1)}$. The first column of $\mathbf{U}^{(1)}$ must be orthogonal to columns from 3 to n, which can only occur if entries from 3 to n are zeros in the 1*st* and 2*nd* rows of $\mathbf{U}^{(1)}$, because after the previous transformation, either entries in the first row or in the second row from 3 to n are zeros already . Now $\mathbf{U}^{(1)}$ can be decoupled into two blocks: one 2×2 block and one $(n-2)\times(n-2)$ block. As $\mathbf{U}^{(1)}$ is unitary, then both blocks are unitary as well. Since $\|\mathbf{U}^{(1)}(1:2,2)\| = 1$, the rest entries in the second column of $\mathbf{U}^{(1)}$ must be zeros due to the fact that $\|u_{*2}^{(1)}\| = 1$.

Because P_2 only affects entries from 3 to n in each row, then the form of $\mathbf{V}^{(0)}$ is not changed.

Now we can apply the same procedure as constructing the 2×2 block in G_o to construct the first 2×2 block in $\mathbf{G}_e$, which locates at $\mathbf{G}_e(2:3,2:3)$. As in this step, only rows and columns from 3 to n is affected, thus the form of $\mathbf{U}^{(1)}$ is not affected. We can do these steps alternatively to $\mathbf{U}$ and $\mathbf{V}$ until the Schur parameter pencil is constructed.

The following algorithm introduced in [2] summarizes the procedure.

Algorithm 3.1.1 *Reducing a Unitary Pencil to a Schur Parameter Pencil*

Given unitary $n\times n$ matrices $\mathbf{U}$ and $\mathbf{V}$, $n>1$, this algorithm computes unitary $n\times n$ matrices $\mathbf{P},\mathbf{Q},\mathbf{G}_o,\mathbf{G}_e \in C^{n\times n}$, such that $\mathbf{G}_o = \mathbf{QUP}^*$ and $\mathbf{G}_e = \mathbf{QVP}^*$ and $\mathbf{G}_o - \lambda\mathbf{G}_e$ is a Schur parameter pencil. $\mathbf{U}$ and $\mathbf{V}$ are overwritten by $\mathbf{G}_o$ and $\mathbf{G}_e$, respectively.

Initialize $\mathbf{Q} = \mathbf{I}$

Compute $\mathbf{Z} = \mathbf{P}(1, n, v_{1*}^*)$

Update $\mathbf{U} = \mathbf{U}\mathbf{Z}^*, \mathbf{V} = \mathbf{V}\mathbf{Z}^*$ *and set* $\mathbf{P} = \mathbf{Z}^*$.

FOR $k = 1, \cdots, n-2$

IF k is odd THEN $x = u_{*k}$ *ELSE* $x = v_{*k}$

Compute $\mathbf{Z} = \mathbf{P}(k+1, n, x)$.

Update $\mathbf{U} = \mathbf{Z}\mathbf{U}, \mathbf{V} = \mathbf{Z}\mathbf{V}, \mathbf{Q} = \mathbf{Z}\mathbf{Q}$.

IF k is odd THEN $x = u_{k*}$ *and* $y = u_{k+1*}$.

ELSE $x = v_{k*}$ *and* $y = v_{k+1*}$.

IF $\|x_k\| \geq \|y_k\|$ *THEN* $\omega = y^*$ *ELSE* $\omega = x^*$

Compute $\mathbf{Z} = \mathbf{P}(k+1, n, \omega)$.

Update $\mathbf{U} = \mathbf{U}\mathbf{Z}^*, \mathbf{V} = \mathbf{V}\mathbf{Z}^*$ *and* $\mathbf{P} = \mathbf{Z}\mathbf{P}$

END

Example 6 *We illustrate the procedure with an example: let* $\mathbf{U}, \mathbf{V} \in C^{5\times 5}$ *be unitary matrices. First, set* $\mathbf{P}_0 = \mathbf{P}(1, 5, v_{1*}^*)$*, then*

$$\mathbf{U}^{(0)} = \mathbf{U}\mathbf{P}_0 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}, \mathbf{V}^{(0)} = \mathbf{V}\mathbf{P}_0 = \begin{bmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}$$

Next, set $\mathbf{P}_1 = \mathbf{P}(2,5,u_{*1}^{(0)})$ *such that the* 3*rd to* 5*th entries in the first column of* $\mathbf{U}^{(0)}$ *is eliminated, that is*

$$\tilde{\mathbf{U}}^{(1)} = \mathbf{P}_1\mathbf{U}\mathbf{P}_0 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}, \tilde{\mathbf{V}}^{(1)} = \mathbf{P}_1\mathbf{V}\mathbf{P}_0 = \begin{bmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}$$

Then compare $|\tilde{\mathbf{U}}(1,1)|$ *and* $|\tilde{\mathbf{U}}^{(1)}(2,1)|$. *If* $|\tilde{\mathbf{U}}^{(1)}(1,1)| \geq |\tilde{\mathbf{U}}^{(1)}(2,1)|$, *then take* $\mathbf{P}_2 = \mathbf{P}(2,5,(\tilde{u}_{2*}^{(1)})^*)$; *otherwise, take* $\mathbf{P}_2 = \mathbf{P}(2,5,(\tilde{u}_{1*}^{(1)})^*)$. *One has*

$$\mathbf{U}^{(1)} = \mathbf{P}_1\mathbf{U}\mathbf{P}_0\mathbf{P}_2 = \begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{bmatrix}, \mathbf{V}^{(1)} = \mathbf{P}_1\mathbf{V}\mathbf{P}_0\mathbf{P}_2 = \begin{bmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}$$

We start to eliminate entries $\mathbf{V}^{(1)}(4,2)$ *and* $\mathbf{V}^{(1)}(5,2)$ *by setting* $\mathbf{P}_3 = \mathbf{P}(3,5,v_{*2}^{(1)})$. *Then*

$$\tilde{\mathbf{U}}^{(2)} = \mathbf{P}_3\mathbf{P}_1\mathbf{U}\mathbf{P}_0\mathbf{P}_2 = \begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{bmatrix}, \tilde{\mathbf{V}}^{(2)} = \mathbf{P}_3\mathbf{P}_1\mathbf{V}\mathbf{P}_0\mathbf{P}_= \begin{bmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{bmatrix}$$

Compare $|\tilde{\mathbf{V}}^{(2)}(2,2)|$ *and* $|\tilde{\mathbf{V}}^{(2)}(3,2)|$: *if* $|\tilde{\mathbf{V}}^{(2)}(2,2)| \geq |\tilde{\mathbf{V}}^{(2)}(3,2)|$, *set* $\mathbf{P}_4 = \mathbf{P}(3,5,(\tilde{v}_{3*}^{(2)})^*)$; *other-*

wise, set $\mathbf{P}_4 = \mathbf{P}(3,5,(\tilde{v}_{2*}^{(2)})^*)$. *Then*

$$\mathbf{U}^{(2)} = \mathbf{P}_3\mathbf{P}_1\mathbf{U}\mathbf{P}_0\mathbf{P}_2\mathbf{P}_4 = \begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{bmatrix}, \mathbf{V}^{(2)} = \mathbf{P}_3\mathbf{P}_1\mathbf{V}\mathbf{P}_0\mathbf{P}_2\mathbf{P}_4 = \begin{bmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & \times & 0 & 0 \\ 0 & \times & \times & 0 & 0 \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$$

Now $\mathbf{V}^{(2)}$ *has already had the form as defined for* $\mathbf{G}_e$. *Next step is to finish constructing* $\mathbf{G}_o$ *by eliminating entries in* $\mathbf{U}^{(2)}(5,3:4)$ *and* $\mathbf{U}^{(2)}(3:4,5)$. *First, set* $\mathbf{P}_5 = \mathbf{P}(4,5,u_{*3})$ *to eliminate* $\mathbf{U}^{(2)}(5,3)$. *We have*

$$\tilde{\mathbf{U}}^{(3)} = \mathbf{P}_5\mathbf{P}_3\mathbf{P}_1\mathbf{U}\mathbf{P}_0\mathbf{P}_2\mathbf{P}_4 = \begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}, \tilde{\mathbf{V}}^{(3)} = \mathbf{P}_5\mathbf{P}_3\mathbf{P}_1\mathbf{V}\mathbf{P}_0\mathbf{P}_2\mathbf{P}_4 = \begin{bmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & \times & 0 & 0 \\ 0 & \times & \times & 0 & 0 \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$$

Compare $|\tilde{\mathbf{U}}^{(3)}(3,3)|$ *and* $|\tilde{\mathbf{U}}^{(3)}(4,3)|$*: if* $|\tilde{\mathbf{U}}^{(3)}(3,3)| \geq |\tilde{\mathbf{U}}^{(3)}(4,3)|$, *set* $\mathbf{P}_6 = \mathbf{P}(4,5,(\tilde{u}_{4*}^{(3)})^*)$*; otherwise, set* $\mathbf{P}_6 = \mathbf{P}(4,5,(\tilde{u}_{3*}^{(3)})^*)$. *Then*

$$\mathbf{G}_o = \mathbf{U}^{(3)} = \mathbf{P}_5\mathbf{P}_3\mathbf{P}_1\mathbf{U}\mathbf{P}_0\mathbf{P}_2\mathbf{P}_4\mathbf{P}_6 = \begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 \\ 0 & 0 & \times & \times & 0 \\ 0 & 0 & \times & \times & 0 \\ 0 & 0 & 0 & 0 & \times \end{bmatrix},$$

$$\mathbf{G}_e = \mathbf{V}^{(3)} = \mathbf{P}_5\mathbf{P}_3\mathbf{P}_1\mathbf{V}\mathbf{P}_0\mathbf{P}_2\mathbf{P}_4\mathbf{P}_6 = \begin{bmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & \times & 0 & 0 \\ 0 & \times & \times & 0 & 0 \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$$

Finally, we find

$$\mathbf{G}_o = \mathbf{U}^{(3)}, \mathbf{G}_e = \mathbf{V}^{(3)}, \mathbf{Q} = \mathbf{P}_5\mathbf{P}_3\mathbf{P}_1, \mathbf{P} = \mathbf{P}_6\mathbf{P}_4\mathbf{P}_2\mathbf{P}_0$$

such that

$$\mathbf{G}_o = \mathbf{Q}\mathbf{U}\mathbf{P}^*, \mathbf{G}_e = \mathbf{Q}\mathbf{V}\mathbf{P}^*$$

According to [2], this algorithm requires essentially $\frac{8}{3}n^3$ flops.

3.1.2 Unitary Matrix Case

Algorithm 3.1.1 may apply to a single unitary matrix $\mathbf{U}$ by considering it as a pencil $\mathbf{U} - \lambda\mathbf{I}$. Since here $\mathbf{V} = \mathbf{I}$, it is possible to use this special form to reduce the cost. In [2] a reduction process was proposed that reduces $\mathbf{U}$ to the pentadiagonal matrix $\mathbf{G}_o\mathbf{G}_e^*$. This procedure only needs half of the flops needed with Algorithm 3.1.1.

The main idea of this reduction is to reduce $\mathbf{U}$ to a pentadiagonal form with unitarily similarity transformation. In practice $\mathbf{U}$ is reduced to $\mathbf{G}_o$, and $\mathbf{G}_e$ is formed during the reduction process. We give a description of the process.

Let $\mathbf{U} \in C^{n\times n}$ be a unitary matrix. First, we eliminate entries in u_{*1} from 3 to n by $\mathbf{P}_1 = \mathbf{P}(2, n, u_{*1})$. Since we need to perform a similarity transformation, we multiply $\mathbf{P}_1^*$ to the right of

$\mathbf{U}$ at the same time to get

$$\mathbf{U}^{(1)} = \mathbf{P}_1 \mathbf{U} \mathbf{P}_1^* = \begin{bmatrix} \times & \times & \times & \cdots & \times \\ \times & \times & \times & \cdots & \times \\ 0 & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \times & \cdots & \times \end{bmatrix}$$

Next, in order to construct the pentadiagonal form, we need to eliminate entries from 4 to n in the 1st and 2nd rows of $\mathbf{U}^{(1)}$. For numerical stability, we compare $|\mathbf{U}^{(1)}(1,1)|$ and $|\mathbf{U}^{(1)}(2,1)|$. If $|\mathbf{U}^{(1)}(1,1)| \geq |\mathbf{U}^{(1)}(2,1)|$, set $\mathbf{P}_2 = \mathbf{P}(3,n,(u_{2*}^{(1)})^*)$; otherwise, set $\mathbf{P}_2 = \mathbf{P}(3,n,(u_{1*}^{(1)})^*)$. Then

$$\mathbf{U}^{(2)} = \mathbf{P}_2 \mathbf{U}^{(1)} \mathbf{P}_2^* = \begin{bmatrix} \times & \times & \times & 0 & \cdots & 0 \\ \times & \times & \times & 0 & \cdots & 0 \\ 0 & \times & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \times & \times & \cdots & \times \end{bmatrix}$$

Extra zeros in the 1*st* and 2*nd* rows of $\mathbf{U}^{(2)}$ are generated automatically due to the orthogonality.

In order to generate the form of $\mathbf{G}_o$, we need to continue with the previous step to eliminate entries $\mathbf{U}^{(2)}(1,3)$ and $\mathbf{U}^{(2)}(2,3)$. If $|\mathbf{U}^{(2)}(1,1)| \geq |\mathbf{U}^{(2)}(2,1)|$, set $\mathbf{P}_3 = \mathbf{P}(2,3,(u_{2*}^{(2)})^*)$; otherwise, let $\mathbf{P}_3 = \mathbf{P}(2,3,(u_{1*}^{(2)})^*)$, we have

$$\mathbf{U}^{(3)} = \mathbf{U}^{(2)} \mathbf{P}_3^* = \begin{bmatrix} \times & \times & 0 & \cdots & 0 \\ \times & \times & 0 & \cdots & 0 \\ 0 & 0 & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \times & \cdots & \times \end{bmatrix}$$

now we set

$$\mathbf{V}^{(3)} = \mathbf{P}_3^* = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \cdots & 0 \\ 0 & \times & \times & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Till now, we have

$$\mathbf{U}^{(2)} = \mathbf{U}^{(3)}\mathbf{V}^{(3)*}$$

and the eigenvalue problem

$$\mathbf{U} - \lambda\mathbf{I} = 0$$

has been changed to a pencil problem, such as

$$\begin{aligned} \mathbf{U} - \lambda\mathbf{I} &\rightarrow \mathbf{U}^{(3)}\mathbf{V}^{(3)*} - \lambda\mathbf{I} \\ &\rightarrow \mathbf{U}^{(3)} - \lambda\mathbf{V}^{(3)} \\ &= 0 \end{aligned}$$

Now the unitary matrix case has been translated into a unitary pencil case. To complete the Schur parameter form construction, we continue in the same way as above for the rest of $\mathbf{U}^{(3)}$ until $\mathbf{U}$ is reduced to $\mathbf{G}_o$, and $\mathbf{V}$ is constructed to be $\mathbf{G}_e$.

The algorithm in [2] is summarized as below:

Algorithm 3.1.2 *Reduction of a Unitary Matrix to a Schur Parameter Pencil*

Given a unitary $n \times n$ matrix $\mathbf{U}$, $n > 2$, this algorithm computes unitary $n \times n$ matrices $\mathbf{Q}, \mathbf{G}_o, \mathbf{G}_e \in C^{n \times n}$, such that $\mathbf{G}_o - \lambda\mathbf{G}_e$ is a Schur parameter pencil. $\mathbf{U}$ is overwritten by $\mathbf{G}_o$.

Initialize $\mathbf{Q} = \mathbf{I}$ *and* $\mathbf{G}_e = \mathbf{I}$

FOR $k = 1, 3, \cdots, 2[\frac{n+1}{2}] - 3$

Copmpute $\mathbf{Z} = \mathbf{P}(k+1, n, u_{*k})$

Update $\mathbf{U} = \mathbf{ZUZ}^*$ *and* $\mathbf{Q} = \mathbf{ZQ}$

IF $|u_{kk}| \geq |u_{k+1,k}|$, *THEN* $K = k+1$ *ELSE* $K = k$

Compute $\mathbf{Z} = \mathbf{P}(k+2, n, u^*_{K*})$.

$\mathbf{U} = \mathbf{ZUZ}^*$ *and* $\mathbf{Q} = \mathbf{ZQ}$.

$\mathbf{N} = \mathbf{P}(k+1, k+2, u^*_{K*})$

Update $\mathbf{U} = \mathbf{UN}^*$ *and* $\mathbf{G}_e = \mathbf{G}_e\mathbf{N}^*$

We will also go through the algorithm by an example.

Example 7 *Let* $\mathbf{U} \in C^{5\times 5}$ *be a unitary matrix.*

First, eliminate entries from 3 *to* 5 *by setting* $\mathbf{P}_1 = \mathbf{P}(2, 5, u_{*1})$, *that is*

$$\mathbf{U}^{(1)} = \mathbf{P}_1\mathbf{U}\mathbf{P}_1^* = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}$$

Then we need to eliminate entries from 4 *to* 5 *in row* 1 *and row* 2 *of* $\mathbf{U}^{(1)}$. *For numerical stability, compare* $|\mathbf{U}^{(1)}(1,1)|$ *and* $|\mathbf{U}^{(1)}(2,1)|$. *If* $|\mathbf{U}^{(1)}(1,1)| \geq |\mathbf{U}^{(1)}(2,1)|$, *set* $\mathbf{P}_2 = \mathbf{P}(3, 5, (u^{(1)}_{2*})^*)$;

otherwise, set $\mathbf{P}_2 = \mathbf{P}(3,5,(u_{1*}^{(1)})^*)$*, then*

$$\mathbf{U}^{(2)} = \mathbf{P}_2\mathbf{U}^{(1)}\mathbf{P}_2^* = \begin{bmatrix} \times & \times & \times & 0 & 0 \\ \times & \times & \times & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}$$

Additional zeros in the 1*st and* 2*nd rows of* $\mathbf{U}^{(2)}$ *are generated automatically due to the orthogonality between the columns of unitary matrices.*

Then, if $|\mathbf{U}^{(2)}(1,1)| \geq |\mathbf{U}^{(2)}(2,1)|$*, set* $\mathbf{P}_3 = \mathbf{P}(2,3,(u_{2*}^{(2)})^*)$*; otherwise, let* $\mathbf{P}_3 = \mathbf{P}(2,3,(u_{1*}^{(2)})^*)$*, we have*

$$\mathbf{U}^{(3)} = \mathbf{U}^{(2)}\mathbf{P}_3^* = \begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{bmatrix}$$

now we set

$$\mathbf{V}^{(3)} = \mathbf{P}_3^* = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & 0 & 0 \\ 0 & \times & \times & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

To complete the Schur parameter form construction, we continue in the same way as above for the

rest 3×3 *block of* $\mathbf{U}^{(3)}$. *Set* $\mathbf{P}_4 = \mathbf{P}(4,5,u_{*3}^{(3)})$ *to eliminate* $\mathbf{U}^{(3)}(5,3)$

$$\mathbf{U}^{(4)} = \mathbf{P}_4\mathbf{U}^{(3)}\mathbf{P}_4^* = \begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$$

At the same time, in principle we need to update $V^{(4)} = \mathbf{P}_4\mathbf{V}^{(3)}\mathbf{P}_4^*$ *to ensure that the pencil* $\mathbf{U}^{(4)} - \lambda\mathbf{V}^{(4)}$ *has the same eigenvalues as* $\mathbf{U}$. *However, as* $\mathbf{P}_4$ *has the leading* 3×3 *principal submatrix as identity and* $\mathbf{V}^{(3)}$ *has the last* 2×2 *principal submatrix as identity, then we have* $\mathbf{V}^{(4)} = \mathbf{V}^{(3)}$. *The last step is to eliminates either* $\mathbf{U}^{(4)}(3,5)$ *or* $\mathbf{U}^{(4)}(4,5)$ *by multiplying a Householder* $\mathbf{P}_5$ *to the right of* $\mathbf{U}^{(5)}$. *The additional zeros will be generated as* $\mathbf{U}^{(5)}$ *is still unitary. We have*

$$\mathbf{U}^{(5)} = \mathbf{U}^{(4)}\mathbf{P}_5^* = \begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 \\ 0 & 0 & \times & \times & 0 \\ 0 & 0 & \times & \times & 0 \\ 0 & 0 & 0 & 0 & \times \end{bmatrix}$$

Update $V^{(5)} = \mathbf{V}^{(4)}\mathbf{P}_5^* = \mathbf{V}^{(3)}\mathbf{P}_5^*$. *As we can see here, the multiplication does not require any arithmetic operation. We just need to store the essential part of* $\mathbf{P}_5^*$ *to the last* 2×2 *principal submatrix of* $\mathbf{V}^{(3)}$. *Finally, we complete transforming a single unitary eigenvalue problem to a Schur parameter pencil eigenvalue problem.*

In [2], the transformation of a unitary pencil $\mathbf{U} - \lambda\mathbf{V}$ to a Schur parameter pencil $\mathbf{G}_o - \lambda G_e = \mathbf{Q}(\mathbf{U} - \lambda\mathbf{V})\mathbf{P}^*$ is uniquely determined up to scaling with unitary diagonal matrices, if the first column of $\mathbf{Q}$ or $\mathbf{P}$ is fixed and the complementary Schur parameters are nonzeros, which means the Schur parameter pencil is unreduced. This transformation is also proved to be backward stable

in [2]. In [2], it also provides a shifted iterative method to converge $\mathbf{G}_o, \mathbf{G}_e$ to diagonal forms. The choice of the value of shifts is similar to that in the standard shifted QR iteration. The 2×2 diagonal block form of $\mathbf{G}_o$ and $\mathbf{G}_e$ is broken by applying shifts to the original $\mathbf{G}_o, \mathbf{G}_e$. Then the bulge-chase method is applied to retain the 2×2 diagonal block form. Now, we will use some examples to demonstrate the bulge-chase process in both single shifted and double shifted cases.

3.2 Shift and Iteration

For Phase II of the method [2], two kinds of shifted iteration are discussed. Similar to the shift introduced in the standard QR algorithm, Angelika Bunse-Gerstner and Ludwig Elsner([2]) described the iteration process of how to converge $\mathbf{G}_o, \mathbf{G}_e$ to diagonal forms with single shift and double shifts.

3.2.1 Single Shift Iteration

[2] introduced the single shifted iterative method to converge $\mathbf{G}_o, \mathbf{G}_e$ to diagonal forms. In each iteration step, $\mathbf{G}_o^{(i-1)}, \mathbf{G}_e^{(i-1)}$ are transformed into $\mathbf{G}_o^{(i)}, \mathbf{G}_e^{(i)}$. For simplicity, we omit the super- and subscript i in our description below.

Let

$$z = (\mathbf{G}_o - \mu \mathbf{G}_e)e_1 = \begin{bmatrix} \times \\ \times \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where μ can be chosen to be the last diagonal entry of $\mathbf{G}_o \mathbf{G}_e^*$. Then as described in Algorithm 2.2.2, compute $\mathbf{Q}_0 = \mathbf{P}(1,2,z)$ such that $\mathbf{Q}_0 z = ke_1$. Multiply $\mathbf{Q}_0$ to the left of the pencil $\mathbf{G}_o - \lambda \mathbf{G}_e$, then

the Schur parameter form is destroyed as

$$\mathbf{Q}_0\mathbf{G}_o = \begin{bmatrix} \times & \times & & & & & \\ \times & \times & & & & & \\ & & \times & \times & & & \\ & & \times & \times & & & \\ & & & & \times & \times & \\ & & & & \times & \times & \\ & & & & & & \ddots \end{bmatrix}, \mathbf{Q}_0\mathbf{G}_e = \begin{bmatrix} \times & + & + & & & & \\ + & \times & \times & & & & \\ & \times & \times & & & & \\ & & & \times & \times & & \\ & & & \times & \times & & \\ & & & & & \ddots & \end{bmatrix}$$

where $+$ stands for some additional possibly nonzero entry. Now we can apply Algorithm 3.1.1 to the newly generated unitary pencil $\mathbf{Q}_0(\mathbf{G}_o - \lambda\mathbf{G}_e)$. In [2], the reduction is proved to require $\mathscr{O}(n)$ flops because of the additional zeros compared with the initial unitary pencil $\mathbf{U} - \lambda\mathbf{V}$.

The process of the reduction is similar to Algorithm 3.1.1. First we need to use a Householder reflector $\mathbf{P}_0$ to eliminate $\mathbf{G}_e(1,2)$ and $\mathbf{G}_e(1,3)$. In this step, only columns 1 to 3 are involved, so the number of flops required is less than that of the full matrix. We will have $\mathbf{G}_{o,1} = \mathbf{Q}_0\mathbf{G}_o\mathbf{P}_0$ and $\mathbf{G}_{e,1} = \mathbf{Q}_0\mathbf{G}_e\mathbf{P}_0$

$$\mathbf{G}_{o,1} = \begin{bmatrix} \times & \times & + & & & & \\ \times & \times & + & & & & \\ + & + & \times & \times & & & \\ + & + & \times & \times & & & \\ & & & & \times & \times & \\ & & & & \times & \times & \\ & & & & & & \ddots \end{bmatrix}, \mathbf{G}_{e,1} = \begin{bmatrix} \times & & & & & & \\ & \times & \times & & & & \\ & \times & \times & & & & \\ & & & \times & \times & & \\ & & & \times & \times & & \\ & & & & & \ddots & \end{bmatrix}$$

then we are going to eliminate the $(3,1)$ and $(4,1)$ entries in $\mathbf{G}_{o,1}$ by a Householder matrix $\mathbf{Q}_1$. In

this step, rows 2 to 4 are involved

$$\mathbf{Q}_1\mathbf{G}_{o,1} = \begin{bmatrix} \times & \times & + & & & & \\ \times & \times & + & & & & \\ & + & \times & \times & & & \\ & + & \times & \times & & & \\ & & & & \times & \times & \\ & & & & \times & \times & \\ & & & & & & \ddots \end{bmatrix}, \mathbf{Q}_1\mathbf{G}_{e,1} = \begin{bmatrix} \times & & & & & & \\ & \times & \times & + & + & & \\ & \times & \times & + & + & & \\ & + & + & \times & \times & & \\ & & & \times & \times & & \\ & & & & & \ddots & \end{bmatrix}$$

the additional zeros at $(1,4)$ and $(2,4)$ in $\mathbf{Q}_1\mathbf{G}_{o,1}$ are due to the orthogonality between the columns of unitary matrices.

Next we need to zero out $(1,3)$ and $(2,3)$ entries in $\mathbf{Q}_1\mathbf{G}_{o,1}$ by a unitary transformation $\mathbf{P}_1$ where only columns 2 to 3 will be affected, then we have $\mathbf{G}_{o,2} = \mathbf{Q}_1\mathbf{G}_{o,1}\mathbf{P}_1$ and $\mathbf{G}_{e,2} = \mathbf{Q}_1\mathbf{G}_{e,1}\mathbf{P}_1$, that is

$$\mathbf{G}_{o,2} = \begin{bmatrix} \times & \times & & & & & \\ \times & \times & & & & & \\ & & \times & \times & & & \\ & & \times & \times & & & \\ & & & & \times & \times & \\ & & & & \times & \times & \\ & & & & & & \ddots \end{bmatrix}, \mathbf{G}_{e,2} = \begin{bmatrix} \times & & & & & & \\ & \times & \times & + & + & & \\ & \times & \times & + & + & & \\ & + & + & \times & \times & & \\ & & & \times & \times & & \\ & & & & & \ddots & \end{bmatrix}$$

Then we will repeat to eliminate the extra zeros from $\mathbf{G}_{o,i}$ and $\mathbf{G}_{e,i}$ until we regain the Schur parameter form.

3.2.2 Double Shifts Iteration

As introduced in the previous chapter, the double shifts algorithm will keep the arithmetic real for real matrices. Therefore, we assume that $\mathbf{G}_o - \lambda \mathbf{G}_e$ is a real Schur parameter pencil, where $\mathbf{G}_o, \mathbf{G}_e \in R^{n\times n}$, and that σ_1, σ_2 are shifts chosen, where $\sigma_1 + \sigma_2$ and $\sigma_1\sigma_2$ are real, then all matrices in the reduction process are real.

Define

$$z = (\mathbf{G}_o - \sigma_1 \mathbf{G}_e)(\mathbf{G}_e^T - \sigma_2 \mathbf{G}_o^T)e_1 = \begin{bmatrix} \times \\ \times \\ \times \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where σ_1, σ_2 can be chosen to be the eigenvalues of the trailing 2×2 principal submatrix of $\mathbf{G}_o\mathbf{G}_e^T$. Let $\mathbf{Q}_0 = \mathbf{P}(1,3,z)$ such that

$$\mathbf{Q}_0 z = \beta e_1$$

Then apply $\mathbf{Q}_0$ to the right of $\mathbf{G}_o$ and $\mathbf{G}_e$, we have

$$\mathbf{Q}_0\mathbf{G}_o = \begin{bmatrix} \times & \times & + & + & & & \\ \times & \times & + & + & & & \\ + & + & \times & \times & & & \\ & & \times & \times & & & \\ & & & & \times & \times & \\ & & & & \times & \times & \\ & & & & & & \ddots \end{bmatrix}, \mathbf{Q}_0\mathbf{G}_e = \begin{bmatrix} \times & + & + & & & & \\ + & \times & \times & & & & \\ + & \times & \times & & & & \\ & & & \times & \times & & \\ & & & \times & \times & & \\ & & & & & \ddots & \end{bmatrix}$$

Then we are going to use the Algorithm 3.1.1 again to regain the Schur parameter form from $\mathbf{Q}_0(\mathbf{G}_o - \lambda \mathbf{G}_e)$, which we will not specifically show here.

During the whole process of iteration steps, we need to monitor the values of complementary Schur parameters. Deflation need to be considered if there is zero among the values.

Chapter 4

Modified Schur Parameter Pencil Method

In the previous chapter, we have introduced a Schur parameter pencil method for unitary eigenproblems [2]. Compared with the standard QR algorithm, the Schur parameter method provides a more efficient way to reduce a unitary pencil to diagonal forms by Schur parameter pencil reduction.

Definition 4.0.1 *A matrix* $\mathbf{A} \in C^{n\times n}$ *is called a modified Householder reflector if*

$$\mathbf{A} = \mathbf{DP},$$

where

$$\mathbf{D} = \mathrm{diag}(d_1, d_2, \cdots, d_n), \qquad |d_i| = 1, \quad \forall\, i \in N.$$

and $\mathbf{P}$ *is a standard Householder reflector. It is obvious to see that* $\mathbf{A}$ *is still unitary.*

For numerical stability, during the process of Schur parameter reduction, norms between vectors are frequently compared, which brings up a lot of extra work in computation. To avoid the frequent comparison, we use another approach as introduced in [8] to reduce a unitary pencil $\mathbf{U} - \lambda\mathbf{V}$ to a Schur parameter form $\mathbf{G}_o - \lambda\mathbf{G}_e$ by reducing $\mathbf{U}$ and $\mathbf{V}$ with unitary matrices $\mathbf{Q}, \mathbf{P}, \mathbf{G}_o, \mathbf{G}_e$ to

identity matrices through a sequence of Householder transformations and Givens rotations, that is:

$$\mathbf{G}_o^*\mathbf{QUP}^* = \mathbf{I}, \quad \mathbf{QVP}^*\mathbf{G}_e^* = \mathbf{I} \tag{4.1}$$

where

$$\mathbf{Q} = \mathbf{Q}_s\mathbf{Q}_{s-1}\cdots\mathbf{Q}_2\mathbf{Q}_1; \tag{4.2}$$

$$\mathbf{P}^* = \mathbf{P}_1^*\mathbf{P}_2^*\cdots\mathbf{P}_{t-1}^*\mathbf{P}_t^*; \tag{4.3}$$

where $\mathbf{Q}_i, \mathbf{P}_i$ are modified Hessenberg reflectors, and

$$\mathbf{G}_o^* = \mathbf{G}_{2n-1}\cdots\mathbf{G}_3\mathbf{G}_1; \tag{4.4}$$

$$\mathbf{G}_e^* = \mathbf{G}_0\mathbf{G}_2\cdots\mathbf{G}_{2m}; \tag{4.5}$$

$\mathbf{G}_i$ are Givens matrices affecting the intersection of ith to $i+1th$ rows and columns of a $k \times k$ matrix, such that

$$\mathbf{G}_i = \begin{bmatrix} \ddots & & & \\ & g_{i,i} & g_{i,i+1} & \\ & g_{i+1,i} & g_{i+1,i+1} & \\ & & & \ddots \end{bmatrix} \tag{4.6}$$

and we define $\mathbf{G}_0 = \mathbf{I}$.

Our modified method will still follow the steps as in [2]:

Phase I: Schur parameter pencil reduction. Compute $\mathbf{G}_o^*\mathbf{QUP}^* = \mathbf{I}, \mathbf{QVP}^*\mathbf{G}_e^* = \mathbf{I}$, such that the unitary matrix pencil $\mathbf{U} - \lambda\mathbf{V}$ can be replaced by $\mathbf{G}_o - \lambda\mathbf{G}_e$ for finding eigenvalues.

Phase II: QR-type iteration with shifts. Repeat the following steps until $\mathbf{G}_o, \mathbf{G}_e$ converge to diagonal forms:

(a) Determine the shift and construct a new matrix $\mathbf{G} = f(\mathbf{G}_o, \mathbf{G}_e)$, where f(x,y) is a lower degree function, according to the shift.

(b) Determine the first column of $\mathbf{G}$ and compute a Householder reflector $\mathbf{Q}_0$ for the column.

(c) Compute $\mathbf{G}_o := \mathbf{Q}_0\mathbf{G}_o$ and $\mathbf{G}_e := \mathbf{Q}_0\mathbf{G}_e$.

(d) Reduce $\mathbf{G}_o, \mathbf{G}_e$ to Schur parameter forms again.

4.1 Reducing from a Unitary Pencil to a Schur Parameter Pencil

Given a unitary pencil $\mathbf{U} - \lambda\mathbf{V}$, our goal is to find matrices $\mathbf{Q}, \mathbf{P}, \mathbf{G}_o, \mathbf{G}_e$ such that equation (4.1) is satisfied, then

$$\det(\mathbf{U} - \lambda\mathbf{V}) = 0 \quad \rightarrow \quad \det(\mathbf{QUP} - \lambda\mathbf{QVP}) = 0 \tag{4.7}$$

$$\rightarrow \quad \det(\mathbf{G}_o - \lambda\mathbf{G}_e) = 0 \tag{4.8}$$

we will also introduce two cases of the reduction: unitary pencil case and unitary matrix case.

4.1.1 Unitary Pencil Case

In contrast with the reduction method introduced in [2], the modified reduction method aims at reducing $\mathbf{U}$ and $\mathbf{V}$ into identity matrices at the same time by a sequence of Householder reflectors and Givens matrices. Then $\mathbf{G}_o$ and $\mathbf{G}_e$ can be recovered from the transformation matrices. The reduction procedure is as following:

let $\mathbf{U}, \mathbf{V} \in C^{n\times n}$ be unitary, $\mathbf{P}(j, i, v)$ be the same as defined in (3.6); $\mathbf{G}(v)$ be the Givens matrix for the given vector v. We will use the sub-note $\mathbf{Q}_{\geq i}$ to symbolize that the transformation only affect the rows below the ith row of a given matrix if $\mathbf{Q}_{\geq i}$ is applied to the left of the given matrix, and the columns after the ith column of the given matrix if $\mathbf{Q}_{\geq i}$ is applied to the right hand side.

During the process of reducing $\mathbf{U}, \mathbf{V}$ into identity matrices, we also need to find a sequence of Givens matrices that follow the pattern of $\mathbf{G}_o$ and $\mathbf{G}_e$ as defined in (3.3) and (3.4).

First, we need to eliminate the zeros from $2nd$ to nth rows of $\mathbf{V}$. Define $\mathbf{Q}_{\geq 1} = \mathbf{H}_1\mathbf{P}(1,n,v_{*1})$, and $\mathbf{H}_1$ is a diagonal matrix to keep $v_{*1}(1)$ real, then we have

$$\mathbf{U}^{(0)} = \mathbf{Q}_{\geq 1}\mathbf{U} = \begin{bmatrix} \times & \times & \times & \cdots & \times \\ \times & \times & \times & \cdots & \times \\ \times & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \times & \times & \times & \cdots & \times \end{bmatrix}, \mathbf{V}^{(0)} = \mathbf{Q}_{\geq 1}\mathbf{V} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \cdots & \times \\ 0 & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \times & \cdots & \times \end{bmatrix}. \tag{4.9}$$

The additional zeros on the first row of $\mathbf{V}^{(0)}$ are generated since $\mathbf{V}^{(0)}$ is still unitary. Then we turn to $\mathbf{U}$ and define $\mathbf{Q}_{\geq 2} = \mathbf{P}(2,n,u_{*1}^{(0)})$ to eliminate entries from 3 to n in the first column of $\mathbf{U}$, we have

$$\tilde{\mathbf{U}}^{(1)} = \mathbf{Q}_{\geq 2}\mathbf{U}^{(0)} = \begin{bmatrix} \times & \times & \times & \cdots & \times \\ \times & \times & \times & \cdots & \times \\ 0 & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \times & \cdots & \times \end{bmatrix}, \tilde{\mathbf{V}}^{(1)} = \mathbf{Q}_{\geq 2}\mathbf{V}^{(0)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \cdots & \times \\ 0 & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \times & \cdots & \times \end{bmatrix} \tag{4.10}$$

As $\mathbf{Q}_{\geq 2}$ only affect rows from 2 to 3, so the multiplication won't change the zeros in the first row of $\mathbf{V}^{(0)}$. And because the entries from 2 to n in the first column of $\mathbf{V}^{(0)}$ are all zeros, the first column of $\mathbf{V}^{(0)}$ is not affected. Next, to eliminate the $(1,2)th$ entry of $\tilde{\mathbf{U}}^{(1)}$, let $\mathbf{G}_1^* = \mathbf{G}(\tilde{\mathbf{U}}^{(1)}(1:2,1))$, such that

$$\mathbf{U}^{(1)} = \mathbf{G}_1^*\tilde{\mathbf{U}}^{(1)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \cdots & \times \\ 0 & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \times & \cdots & \times \end{bmatrix}, \mathbf{V}^{(1)} = \tilde{\mathbf{V}}^{(1)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \cdots & \times \\ 0 & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \times & \cdots & \times \end{bmatrix} \tag{4.11}$$

The additional zeros in $\mathbf{U}^{(1)}$ are generated as $\mathbf{U}^{(1)}$ is still unitary. By Definition (4.4), next Givens matrix we need to construct is $\mathbf{G}_3$, which only affects the $3rd$ and $4th$ rows and columns of the original matrix. Thus, we need to reduce entry $\mathbf{U}^{(1)}(2,2)$ to 1 in advance. Define $\mathbf{P}_{\geq 2} = \mathbf{H}_2\mathbf{P}(2,n,u^{(1)}_{*2})$, where $\mathbf{H}_2$ is diagonal and unitary, and apply its conjugate transpose to the right of $\mathbf{U}^{(1)}$ and $\mathbf{V}^{(1)}$, then

$$\mathbf{U}^{(2)} = \mathbf{U}^{(1)}\mathbf{P}^*_{\geq 2} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \times & \cdots & \times \end{bmatrix}, \mathbf{V}^{(2)} = \mathbf{V}^{(1)}\mathbf{P}^*_{\geq 2} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \cdots & \times \\ 0 & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \times & \cdots & \times \end{bmatrix} \tag{4.12}$$

Because $\mathbf{P}^*_{\geq 2}$ only affects the columns from 2 to n, then the zeros in the first column of $\mathbf{V}^{(1)}$ are preserved. And as $\mathbf{V}^{(1)}$ is still unitary, zeros in the first row of $\mathbf{V}^{(1)}$ are preserved, too.

In order to construct $\mathbf{G}_2$, we need to eliminate entries from 4 to n on the second row of $\mathbf{V}^{(2)}$ by a Householder reflector $\mathbf{P}_{\geq 3} = \mathbf{P}(3,n,v^{*(2)}_{2*})$, that is

$$\tilde{\mathbf{U}}^{(3)} = \mathbf{U}^{(2)}\mathbf{P}^*_{\geq 3} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \times & \cdots & \times \end{bmatrix}, \tilde{\mathbf{V}}^{(3)} = \mathbf{V}^{(2)}\mathbf{P}^*_{\geq 3} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & 0 & \cdots & 0 \\ 0 & \times & \times & \times & \cdots & \times \\ 0 & \times & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \times & \times & \cdots & \times \end{bmatrix} \tag{4.13}$$

Then let $\mathbf{G}_2 = \mathbf{G}(\tilde{\mathbf{V}}^{*(3)}(2,2:3))$ and apply its conjugate transpose to the right of $\tilde{\mathbf{V}}^{(3)}$, we get

$$\mathbf{U}^{(3)} = \tilde{\mathbf{U}}^{(3)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \times & \cdots & \times \end{bmatrix}, \mathbf{V}^{(3)} = \tilde{\mathbf{V}}^{(3)}\mathbf{G}_2^* = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \times & \cdots & \times \end{bmatrix} \tag{4.14}$$

Then we can repeat the previous steps to find $\mathbf{G}_3, \mathbf{G}_4, \cdots, \mathbf{G}_n$ until both $\mathbf{U}$ and $\mathbf{V}$ are converted into identities. The 2×2 non-identity diagonal block of each $\mathbf{G}_n$ can be stored directly to $\mathbf{G}_o$ and $\mathbf{G}_e$ if we can write

$$\cdots \mathbf{G}_{2s-1}^* \mathbf{Q}_{\geq k} \cdots \mathbf{Q}_{\geq 2} \mathbf{G}_1 \mathbf{Q}_{\geq 1} \mathbf{U} \mathbf{P}_{\geq 1}^* \mathbf{P}_{\geq 3}^* \cdots \mathbf{P}_{\geq t}^* = \mathbf{I} \tag{4.15}$$

$$\cdots \mathbf{Q}_{\geq k} \cdots \mathbf{Q}_{\geq 2} \mathbf{Q}_{\geq 1} \mathbf{V} \mathbf{P}_{\geq 1}^* \mathbf{P}_{\geq 3}^* \mathbf{G}_2^* \cdots \mathbf{P}_{\geq t}^* \mathbf{G}_{2m}^* = \mathbf{I} \tag{4.16}$$

where $k, m, s, t \in N$ and $2s-1, 2m, k, t \leq n$, as

$$\cdots \mathbf{G}_{2s-1}^* \mathbf{G}_{2s-3}^* \cdots \mathbf{G}_3^* \mathbf{G}_1^* \mathbf{Q}_{\geq k} \cdots \mathbf{Q}_{\geq 2} \mathbf{Q}_{\geq 1} \mathbf{U} \mathbf{P}_{\geq 3}^* \cdots \mathbf{P}_{\geq t}^* = \mathbf{I} \tag{4.17}$$

$$\mathbf{Q}_{\geq k} \cdots \mathbf{Q}_{\geq 2} \mathbf{Q}_{\geq 1} \mathbf{V} \mathbf{P}_{\geq 3}^* \cdots \mathbf{P}_{\geq t}^* \mathbf{G}_0^* \mathbf{G}_2^* \cdots \mathbf{G}_{2m-2}^* \mathbf{G}_{2m}^* \cdots = \mathbf{I} \tag{4.18}$$

so that let

$$\mathbf{G}_o^* = \cdots \mathbf{G}_{2s-1}^* \mathbf{G}_{2s-3}^* \cdots \mathbf{G}_3^* \mathbf{G}_1^* \qquad \mathbf{G}_e^* = \mathbf{G}_0^* \mathbf{G}_2^* \cdots \mathbf{G}_{2m-2}^* \mathbf{G}_{2m}^*$$

$$\mathbf{Q} = \mathbf{Q}_{\geq k} \cdots \mathbf{Q}_{\geq 2} \mathbf{Q}_{\geq 1} \qquad \mathbf{P}^* = \mathbf{P}_{\geq 3}^* \cdots \mathbf{P}_{\geq t}^*$$

then we can rewrite (4.17) and (4.18) as

$$\mathbf{G}_o^* \mathbf{Q} \mathbf{U} \mathbf{P}^* = \mathbf{I} \qquad \mathbf{Q} \mathbf{V} \mathbf{P}^* \mathbf{G}_e^* = \mathbf{I}$$

Then we have

$$\mathbf{QUP} = \mathbf{G}_o \tag{4.19}$$

$$\mathbf{QVP} = \mathbf{G}_e \tag{4.20}$$

The algorithm is provided below, $houseg(v)$ is a function that generates a modified Householder reflector for a vector v and ensures the first entry to be real positive, $givensg(v)$ is a function that generates a Givens matrix for a vector v:

Algorithm 4.1.1 *Reducing a Unitary Pencil to a Schur Parameter Pencil*

Given unitary $n \times n$ matrices $\mathbf{U}$ *and* $\mathbf{V}$*, $n > 1$, this algorithm computes unitary $n \times n$ matrices* $\mathbf{P}, \mathbf{Q}, \mathbf{G}_o, \mathbf{G}_e \in C^{n \times n}$*, such that* $\mathbf{G}_o = \mathbf{QUP}^*$ *and* $\mathbf{G}_e = \mathbf{QVP}^*$ *and* $\mathbf{G}_o - \lambda \mathbf{G}_e$ *is a Schur parameter pencil.* $\mathbf{U}$ *and* $\mathbf{V}$ *are overwritten by* $\mathbf{G}_o$ *and* $\mathbf{G}_e$*, respectively.* $\mathbf{H}$ *is the matrix used to store intermediate Householder reflectors, Givens matrices or diagonal matrices.*

Initialize $\mathbf{Q} = \mathbf{P} = \mathbf{G}_o = \mathbf{G}_e = \mathbf{I}$, $\mathbf{G}_0 = \mathbf{I}$

FOR $j = 1, \cdots, \lfloor n/2 \rfloor$

i = 2(j-1)+1;

COMPUTE $\mathbf{H} = houseg(\mathbf{V}(i:n, i))$*;*

UPDATE $\mathbf{Q} := \mathbf{HQ}, \mathbf{U} := \mathbf{HU}, \mathbf{V} := \mathbf{HV}$*;*

COMPUTE $\mathbf{H} = houseg(\mathbf{U}(i+1:n, i))$*;*

UPDATE $\mathbf{Q} := \mathbf{HQ}, \mathbf{U} := \mathbf{HU}, \mathbf{V} := \mathbf{HV}$*;*

COMPUTE $\mathbf{H} = givensg(\mathbf{U}(i:i+1, i))$*;*

UPDATE $\mathbf{G}_o^*(i:i+1, i:i+1) := \mathbf{H}(i:i+1, i:i+1), \mathbf{U} := \mathbf{HU}$*;*

%% for even rows of $\mathbf{U}$ *and* $\mathbf{V}$

COMPUTE $\mathbf{H} = houseg(\mathbf{U}^*(i+1, i+1:n))^*$*;*

UPDATE $\mathbf{P}^* := \mathbf{P}^*\mathbf{H}, \mathbf{U} := \mathbf{U}\mathbf{H}, \mathbf{V} := \mathbf{V}\mathbf{H}$*;*

COMPUTE $\mathbf{H} = houseg(\mathbf{V}(i+1, i+2:n))^*$*;*

UPDATE $\mathbf{P}^* := \mathbf{P}^*\mathbf{H}, \mathbf{U} := \mathbf{U}\mathbf{H}, \mathbf{V} := \mathbf{V}\mathbf{H}$*;*

COMPUTE $\mathbf{H} = givensg(\mathbf{V}^*(i+1, i+2:n))$*;*

UPDATE $\mathbf{G}_e^*(i+1:i+2, i+1:i+2) = \mathbf{G}^*(i+1:i+2, i+1:i+2)$*;*

COMPUTE $\mathbf{V} := \mathbf{V}\mathbf{G}^*$*;*

We are going to use a 6×6 unitary matrix pencil to show how (4.15) and (4.16) can be rewritten as (4.17) and (4.18).

Example 8 *Given a* 6×6 *unitary matrix pencil* $\mathbf{U} - \lambda\mathbf{V}$*, where* $\mathbf{U}, \mathbf{V} \in C^{6\times 6}$ *and both of them are unitary. We are going to reduce* $\mathbf{U}, \mathbf{V}$ *to identity matrices by a series of modified Householder reflectors and Givens matrices. We initialize* $\mathbf{G}_0 = \mathbf{I}$*. Let*

$$\mathbf{U}^{(0)} = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{bmatrix}, \mathbf{V}^{(0)} = \mathbf{V}\mathbf{G}_0 = \mathbf{V} = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{bmatrix},$$

First, we are going to introduce zeros from 2 *to* 6 *in the first column of* $\mathbf{V}$ *by a modified Householder*

reflector $\mathbf{Q}_{\geq 1} = \mathbf{H}_1\mathbf{P}(1,6,v_{*1})$*, where* $\mathbf{H}_1$ *is a diagonal matrix, then*

$$\mathbf{U}^{(1)} = \mathbf{Q}_{\geq 1}\mathbf{U} = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{bmatrix}, \mathbf{V}^{(1)} = \mathbf{Q}_{\geq 1}\mathbf{V} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \end{bmatrix},$$

Then define $\mathbf{Q}_{\geq 2} = \mathbf{P}(2,6,u^{(1)}_{*1})$ *to eliminate entries from* 3 *to* 6 *in the first column of* $\mathbf{U}^{(1)}$*, we have*

$$\tilde{\mathbf{U}}^{(2)} = \mathbf{Q}_{\geq 2}\mathbf{U}^{(1)} = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \end{bmatrix}, \tilde{\mathbf{V}}^{(2)} = \mathbf{Q}_{\geq 2}\mathbf{V}^{(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \end{bmatrix}.$$

Next, we are going to use a givens matrix to eliminate the second entry in the first column of $\tilde{\mathbf{U}}^{(2)}$*. So define* $\mathbf{G}^*_1 = \mathbf{G}(\tilde{\mathbf{U}}^{(2)}(1:2,1)$*, then we only apply* $\mathbf{G}^*_1$ *to the left of* $\tilde{\mathbf{U}}^{(2)}$*:*

$$\mathbf{U}^{(2)} = \mathbf{G}^*_1\tilde{\mathbf{U}}^{(2)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \end{bmatrix}, \mathbf{V}^{(2)} = \tilde{\mathbf{V}}^{(2)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \end{bmatrix}.$$

Then we need to eliminate entries from 3 *to* 6 *in the second row of* $\mathbf{U}^{(2)}$*. Define* $\mathbf{P}_{\geq 2} = \mathbf{H}_2\mathbf{P}(2,6,u^{*(2)}_{2*})$

and apply its conjugate transpose to the right of $\mathbf{U}^{(2)}$ *and* $\mathbf{V}^{(2)}$*, that is:*

$$\mathbf{U}^{(3)} = \mathbf{U}^{(2)}\mathbf{P}^{*}_{\geq 2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \end{bmatrix}, \mathbf{V}^{(3)} = \mathbf{V}^{(2)}\mathbf{P}^{*}_{\geq 2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \end{bmatrix}.$$

Next, we define $\mathbf{P}_{\geq 3} = \mathbf{P}(3, 6, v^{*(3)}_{2*})$ *to eliminate entries from* 4 *to* 6 *in the second row of* $\mathbf{V}^{(3)}$*:*

$$\tilde{\mathbf{U}}^{(4)} = \mathbf{U}^{(3)}\mathbf{P}^{*}_{\geq 3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \end{bmatrix}, \tilde{\mathbf{V}}^{(4)} = \mathbf{V}^{(3)}\mathbf{P}^{*}_{\geq 3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \end{bmatrix}.$$

To zero out $\tilde{\mathbf{V}}^{(4)}(2,3)$ *and transform* $\tilde{\mathbf{V}}^{(4)}(2,2)$ *into* 1*, we use a Givens matrix* $\mathbf{G}_2 = \mathbf{G}(\tilde{\mathbf{V}}^{(4)}(2, 2:3))$ *and apply its conjugate transpose only to the right of* $\tilde{\mathbf{V}}^{(4)}$*, we have:*

$$\mathbf{U}^{(4)} = \tilde{\mathbf{U}}^{(4)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \end{bmatrix}, \mathbf{V}^{(4)} = \tilde{\mathbf{V}}^{(4)}\mathbf{G}^{*}_{2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \end{bmatrix}.$$

Till now, we have reduced the first two rows and columns of $\mathbf{U}$ *and* $\mathbf{V}$ *into identities. We are going*

to repeat the previous steps for the rest part of $\mathbf{U}$ *and* $\mathbf{V}$.

Define $\mathbf{Q}_{\geq 3} = \mathbf{H}_3\mathbf{P}(3,6,v_{*3}^{(4)})$ *and apply it to the left of* $\mathbf{U}^{(4)}, \mathbf{V}^{(4)}$*, then*

$$\mathbf{U}^{(5)} = \mathbf{Q}_{\geq 3}\mathbf{U}^{(4)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \end{bmatrix}, \mathbf{V}^{(5)} = \mathbf{Q}_{\geq 3}\mathbf{V}^{(4)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix}$$

Now let's turn to $\mathbf{U}^{(5)}$*. Define* $\mathbf{Q}_{\geq 4} = \mathbf{P}(4,6,u_{*3}^{(5)})$ *and apply it to the left of* $\mathbf{U}^{(5)}$ *to eliminate entries from 5 to 6 in the third column of* $\mathbf{U}^{(5)}$*. Apply* $\mathbf{Q}_{\geq 4}$ *to the left of* $\mathbf{V}^{(5)}$ *as well, then we get*

$$\tilde{\mathbf{U}}^{(6)} = \mathbf{Q}_{\geq 4}\mathbf{U}^{(5)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix}, \tilde{\mathbf{V}}^{(6)} = \mathbf{Q}_{\geq 4}\mathbf{V}^{5)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix}.$$

Then use a Givens matrix $\mathbf{G}_3^* = \mathbf{G}(\tilde{\mathbf{U}}^{(6)}(3:4,3))$ *to eliminate the* $(4,3)th$ *entry of* $\tilde{\mathbf{U}}^{(6)}$*, so*

$$\mathbf{U}^{(6)} = \mathbf{G}_3^*\tilde{\mathbf{U}}^{(6)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix}, \mathbf{V}^{(6)} = \tilde{\mathbf{V}}^{(6)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix}.$$

Find $\mathbf{P}_{\geq 4} = \mathbf{H}_4\mathbf{P}(4,6,u_{*4}^{*(6)})$ *and apply its conjugate transpose to the right of* $\mathbf{U}^6$ *and* $\mathbf{V}^{(6)}$*, we have*

$$\mathbf{U}^{(7)} = \mathbf{U}^{(6)}\mathbf{P}_{\geq 4}^* = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}, \mathbf{V}^{(7)} = \mathbf{V}^{6)}\mathbf{P}_{\geq 4}^* = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix},$$

To eliminate the $(4,6)$*th entry of* $\mathbf{V}^{(7)}$*, let* $\mathbf{P}_{\geq 5} = \mathbf{P}(5,6,v_{4*}^{*(7)})$*, we have*

$$\tilde{\mathbf{U}}^{(8)} = \mathbf{U}^{(7)}\mathbf{P}_{\geq 5}^* = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}, \tilde{\mathbf{V}}^{(8)} = \mathbf{V}^{(7)}\mathbf{P}_{\geq 5}^* = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & \times & 0 \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix},$$

then use a Givens matrix $\mathbf{G}_4^* = \mathbf{G}(\tilde{\mathbf{V}}^{*(8)}(4,4:5))$ *to zero out the* $(4,5)$*th entry of* $\tilde{\mathbf{V}}^{(8)}$*, so that*

$$\mathbf{U}^{(8)} = \tilde{\mathbf{U}}^{(8)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}, \mathbf{V}^{(8)} = \tilde{\mathbf{V}}^{(8)}\mathbf{G}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix},$$

Now we have finished transforming the first 4 *rows and columns of the original unitary matrices into identities. For the last* 2×2 *submatrix, we can follow the same steps as the previous ones.*

We can use Givens matrices instead of householder reflectors as the submatrix is $2-$*dimensional. Define* $\mathbf{Q}_{\geq 5} = \mathbf{G}(v_{*5}^{(8)})$ *and apply it to the left of* $\mathbf{U}^{(8)}, \mathbf{V}^{(8)}$*, we have*

$$\mathbf{U}^{(9)} = \mathbf{Q}_{\geq 5}\mathbf{U}^{(8)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}, \mathbf{V}^{(9)} = \mathbf{Q}_{\geq 5}\mathbf{V}^{(8)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \times \end{bmatrix}.$$

Define $\mathbf{G}_5^* = \mathbf{G}(u_{*5}^{(9)})$ *and apply it only to the left of* $\mathbf{U}^{(9)}$*, then*

$$\mathbf{U}^{(10)} = \mathbf{G}_5^*\mathbf{U}^{(9)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \times \end{bmatrix}, \mathbf{V}^{(10)} = \mathbf{V}^{(9)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \times \end{bmatrix}.$$

The last step is very important, because now both $\mathbf{U}^{(10)}$ *and* $\mathbf{V}^{(10)}$ *have the last diagonal entry that might not be equal to 1. As the dimension of the original pencil is 6, which is an even number,* $\mathbf{G}_o, \mathbf{G}_e$ *should have the form as defined in Definition (3.3) and (3.4):*

$$\mathbf{G}_o = \begin{bmatrix} \times & \times & 0 & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & 0 & 0 \\ 0 & 0 & \times & \times & 0 & 0 \\ 0 & 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}, \mathbf{G}_e = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & 0 & 0 & 0 \\ 0 & \times & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & \times & 0 \\ 0 & 0 & 0 & \times & \times & 0 \\ 0 & 0 & 0 & 0 & 0 & \times \end{bmatrix},$$

so we need to construct $\mathbf{G}_6$ *as the last block of* $\mathbf{G}_e$*. Define* $\mathbf{P}_{\geq 6} = \text{diag}(1,1,1,1,1,\overline{\mathbf{U}^{(10)}(6,6)})$ *to turn* $\mathbf{U}^{(10)}(6,6)$ *into 1, we have*

$$\mathbf{U}^{(11)} = \mathbf{U}^{(10)}\mathbf{P}_{\geq 6} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{V}^{(11)} = \mathbf{V}^{(10)}\mathbf{P}_{\geq 6} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \times \end{bmatrix}.$$

The last step is to construct $\mathbf{G}_6 = \text{diag}(1,1,1,1,1,\overline{\mathbf{V}^{(11)}(6,6)})$ *and apply it only to* $\mathbf{V}^{(11)}$*, we have*

$$\mathbf{U}^{(12)} = \mathbf{U}^{(11)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{V}^{(12)} = \mathbf{V}^{(11)}\mathbf{G}_6 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Now, we have finished reducing both $\mathbf{U}$ *and* $\mathbf{V}$ *into identities. To sum up the whole process, we have*

$$\mathbf{G}_5^*\mathbf{Q}_{\geq 5}\mathbf{G}_3^*\mathbf{Q}_{\geq 4}\mathbf{Q}_{\geq 3}\mathbf{G}_1^*\mathbf{Q}_{\geq 2}\mathbf{Q}_{\geq 1}\mathbf{U}\mathbf{P}_{\geq 2}^*\mathbf{P}_{\geq 3}^*\mathbf{P}_{\geq 4}^*\mathbf{P}_{\geq 5}^*\mathbf{P}_{\geq 6} = \mathbf{I}$$

$$\mathbf{Q}_{\geq 5}\mathbf{Q}_{\geq 4}\mathbf{Q}_{\geq 3}\mathbf{Q}_{\geq 2}\mathbf{Q}_{\geq 1}\mathbf{V}\mathbf{G}_0^*\mathbf{P}_{\geq 2}^*\mathbf{P}_{\geq 3}^*\mathbf{G}_2^*\mathbf{P}_{\geq 4}^*\mathbf{P}_{\geq 5}^*\mathbf{G}_4^*\mathbf{P}_{\geq 6}\mathbf{G}_6^* = \mathbf{I}$$

First of all, let's look at the left side of $\mathbf{U}$*,* $\mathbf{Q}_{\geq 5}$ *only affects the rows* 5 *and* 6*, but* $\mathbf{G}_3^*$ *only affects rows* 3 *and* 4*, so the multiplication is commutative, that is*

$$\mathbf{Q}_{\geq 5}\mathbf{G}_3^* = \mathbf{G}_3^*\mathbf{Q}_{\geq 5},$$

then the left side of $\mathbf{U}$ *becomes* $\mathbf{G}_5^*\mathbf{G}_3^*\mathbf{Q}_{\geq 5}\mathbf{Q}_{\geq 4}\mathbf{Q}_{\geq 3}\mathbf{G}_1\mathbf{Q}_{\geq 2}\mathbf{Q}_{\geq 1}$*; if we look at* $\mathbf{Q}_{\geq 5}\mathbf{Q}_{\geq 4}\mathbf{Q}_{\geq 3}$*, they only affect rows from* 3 *to* 6*, while* $\mathbf{G}_1^*$ *only deals with rows* 1 *and* 2*, thus, their place can be exchanged, too. Then we have*

$$\mathbf{G}_5^*\mathbf{G}_3^*\mathbf{G}_1^*\mathbf{Q}_{\geq 5}\mathbf{Q}_{\geq 4}\mathbf{Q}_{\geq 3}\mathbf{Q}_{\geq 2}\mathbf{Q}_{\geq 1}\mathbf{U}\mathbf{P}_{\geq 2}^*\mathbf{P}_{\geq 3}^*\mathbf{P}_{\geq 4}^*\mathbf{P}_{\geq 5}^*\mathbf{P}_{\geq 6} = \mathbf{I}$$

The same idea applies to the right side of $\mathbf{V}$*, we can also have*

$$\mathbf{Q}_{\geq 5}\mathbf{Q}_{\geq 4}\mathbf{Q}_{\geq 3}\mathbf{Q}_{\geq 2}\mathbf{Q}_{\geq 1}\mathbf{V}\mathbf{P}_{\geq 2}^*\mathbf{P}_{\geq 3}^*\mathbf{P}_{\geq 4}^*\mathbf{P}_{\geq 5}^*\mathbf{P}_{\geq 6}\mathbf{G}_0^*\mathbf{G}_2^*\mathbf{G}_4^*\mathbf{G}_6^* = \mathbf{I}$$

Then we let

$$\mathbf{Q} = \mathbf{Q}_{\geq 5}\mathbf{Q}_{\geq 4}\mathbf{Q}_{\geq 3}\mathbf{Q}_{\geq 2}\mathbf{Q}_{\geq 1}$$

$$\mathbf{P}^* = \mathbf{P}_{\geq 2}^*\mathbf{P}_{\geq 3}^*\mathbf{P}_{\geq 4}^*\mathbf{P}_{\geq 5}^*\mathbf{P}_{\geq 6}$$

$$\mathbf{G}_o^* = \mathbf{G}_5^*\mathbf{G}_3^*\mathbf{G}_1^*$$

$$\mathbf{G}_e^* = \mathbf{G}_0^*\mathbf{G}_2^*\mathbf{G}_4^*\mathbf{G}_6^*$$

then

$$\mathbf{G}_o^*\mathbf{Q}\mathbf{U}\mathbf{P}^* = \mathbf{I}, \qquad \mathbf{Q}\mathbf{V}\mathbf{P}^*\mathbf{G}_e^* = \mathbf{I}$$

$$\mathbf{Q}\mathbf{U}\mathbf{P}^* = \mathbf{G}_o, \qquad \mathbf{Q}\mathbf{V}\mathbf{P}^* = \mathbf{G}_e$$

now the original unitary pencil $\mathbf{U} - \lambda\mathbf{V}$ *has been transformed into a Schur parameter pencil* $\mathbf{G}_o - \lambda\mathbf{G}_e$*. The multiplication in* $\mathbf{G}_o^* = \mathbf{G}_5^*\mathbf{G}_3^*\mathbf{G}_1^*$ *and* $\mathbf{G}_e^* = \mathbf{G}_0^*\mathbf{G}_2^*\mathbf{G}_4^*\mathbf{G}_6^*$ *can be replaced by directly assigning the* 2×2 *nonidentity block to an identity matrix in place, where the computation flops will be decreased.*

4.1.2 Unitary Matrix Case [8]

For a single unitary matrix $\mathbf{U}$, we can also reduce $\mathbf{U}-\lambda\mathbf{I}$ into a Schur parameter pencil $\mathbf{G}_o-\lambda\mathbf{G}_e$ by Algorithm 4.1.1. However, as introduced in [2], we also modified the method of reducing a single unitary eigenvalue problem into a Schur parameter eigenproblem. The reduction idea is similar to Algorithm 4.1.1; the difference is that for a single unitary matrix , we need to consider similarity transformation. We describe the reduction steps as below.

Given a unitary matrix $\mathbf{U}\in C^{n\times n}$, we need to find unitary matrices $\mathbf{Q},\mathbf{G}_o,\mathbf{G}_e$ such that $\mathbf{G}_o^*\mathbf{Q}\mathbf{U}\mathbf{Q}^*\mathbf{G}_e=\mathbf{I}$, so that $\mathbf{Q}\mathbf{U}\mathbf{Q}^*=\mathbf{G}_o\mathbf{G}_e^*$ and the single unitary eigenproblem $\det(\mathbf{U}-\lambda\mathbf{I})=0$ can be replaced by $\det(\mathbf{G}_o-\lambda\mathbf{G}_e)=0$.

First , let

$$\mathbf{U}^{(0)}=\mathbf{U}=\begin{bmatrix}\times&\times&\times&\cdots&\times\\ \times&\times&\times&\cdots&\times\\ \times&\times&\times&\cdots&\times\\ \vdots&\vdots&\vdots&\cdots&\vdots\\ \times&\times&\times&\cdots&\times\end{bmatrix}.$$

Then define $\mathbf{Q}_{\geq 2}=\mathbf{P}(2,n,u_{*1}^{(0)})$ to eliminate entries from 3 to 5 in the first column of $\mathbf{U}^{(0)}$, and to keep similarity, we need to apply $\mathbf{Q}_{\geq 2}^*$ to the right of $\mathbf{U}^{(0)}$ as well, we have

$$\mathbf{U}^{(1)}=\mathbf{Q}_{\geq 2}\mathbf{U}\mathbf{Q}_{\geq 2}^*=\begin{bmatrix}\times&\times&\times&\cdots&\times\\ \times&\times&\times&\cdots&\times\\ 0&\times&\times&\cdots&\times\\ \vdots&\vdots&\vdots&\cdots&\vdots\\ 0&\times&\times&\cdots&\times\end{bmatrix}.$$

To eliminate the $(1,2)th$ entry of $\mathbf{U}^{(1)}$, let $\mathbf{G}_1=\mathbf{G}(\mathbf{U}^{(1)}(1:2,1))$ and apply it only to the left of

$\mathbf{U}^{(1)}$, then

$$\mathbf{U}^{(2)} = \mathbf{G}_1\mathbf{U}^{(1)} = \mathbf{G}_1\mathbf{Q}_{\geq 2}\mathbf{U}\mathbf{Q}_{\geq 2}^* = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \cdots & \times \\ 0 & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \times & \cdots & \times \end{bmatrix}.$$

The additional zeros in the first row of $\mathbf{U}^{(2)}$ are generated since $\mathbf{U}^{(2)}$ is still unitary. Next, define $\mathbf{Q}_{\geq 3} = \mathbf{P}(3, n, u_{2*}^{*(2)})$ to eliminate entries from 4 to n in the second row of $\mathbf{U}^{(2)}$. So

$$\mathbf{U}^{(3)} = \mathbf{Q}_{\geq 3}\mathbf{U}^{(2)}\mathbf{Q}_{\geq 3}^* = \mathbf{Q}_{\geq 3}\mathbf{G}_1\mathbf{Q}_{\geq 2}\mathbf{U}\mathbf{Q}_{\geq 2}^*\mathbf{Q}_{\geq 3}^* = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \cdots & 0 \\ 0 & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \times & \times & \cdots & \times \end{bmatrix}.$$

Then we use a Givens matrix $\mathbf{G}_2^* = \mathbf{G}(\mathbf{U}^{*(3)}(2, 2:3))$ to make the $(2,3)th$ entry of $\mathbf{U}^{(3)}$ as 0:

$$\mathbf{U}^{(4)} = \mathbf{U}^{(3)}\mathbf{G}_2 = \mathbf{Q}_{\geq 3}\mathbf{G}_1^*\mathbf{Q}_{\geq 2}\mathbf{U}\mathbf{Q}_{\geq 2}^*\mathbf{Q}_{\geq 3}^*\mathbf{G}_2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \times & \cdots & \times \end{bmatrix}.$$

Then we can repeat the previous steps until $\mathbf{U}$ is reduced to identity. The algorithm is shown below:

Algorithm 4.1.2 *Reducing a Unitary Matrix to a Schur Parameter Pencil*

Given unitary $n \times n$ matrix $\mathbf{U}$, $n > 1$, this algorithm computes unitary $n \times n$ matrices $\mathbf{Q}, \mathbf{G}_o, \mathbf{G}_e \in C^{n \times n}$, such that $\mathbf{QUQ}^* = \mathbf{G}_o\mathbf{G}_e^*$ and $\mathbf{G}_o - \lambda\mathbf{G}_e$ is a Schur parameter pencil. $\mathbf{H}$ is the matrix used

to store intermediate Householder reflectors, Givens matrices or diagonal matrices; the function houseg(v) *generates a modified Householder reflector for the vector* $v \in C^n$.

Initialize $\mathbf{Q} = \mathbf{G}_o = \mathbf{G}_e = \mathbf{I}$, $\mathbf{G}_0 = \mathbf{I}$

FOR $j = 1, \cdots, \lfloor n/2 \rfloor$

i = 2(j-1)+1;

COMPUTE $\mathbf{H} = houseg(\mathbf{U}(i+1:n, i))$*;*

UPDATE $\mathbf{U} := \mathbf{HUH}^*, \mathbf{Q} := \mathbf{HQ}$*;*

COMPUTE $\mathbf{H} = givensg(\mathbf{U}(i:i+1, i))$*;*

UPDATE $\mathbf{U} := \mathbf{HU}, \mathbf{G}_o^* := \mathbf{HG}_o^*$*;*

COMPUTE $\mathbf{H} = houseg(\mathbf{U}^*(i+1, i+2:n))$*;*

UPDATE $\mathbf{U} := \mathbf{HUH}^*, \mathbf{Q} := \mathbf{HQ}$*;*

COMPUTE $\mathbf{H} = givensg(\mathbf{U}^*(i+1, i+1:i+2))$*;*

UPDATE $\mathbf{U} := \mathbf{UH}^*, \mathbf{G}_e^* = \mathbf{G}_e^*\mathbf{H}^*$*;*

We also use a 5×5 example to illustrate the algorithm above:

Example 9 *Given a unitary matrix* $\mathbf{U} \in C^{5\times 5}$*, we need to find unitary matrices* $\mathbf{Q}, \mathbf{G}_o, \mathbf{G}_e$ *such that* $\mathbf{G}_o^*\mathbf{QUQ}^*\mathbf{G}_e = \mathbf{I}$*, so that* $\mathbf{QUQ}^* = \mathbf{G}_o\mathbf{G}_e^*$ *and the single unitary eigenproblem* $\det(\mathbf{U} - \lambda\mathbf{I}) = 0$ *can be replaced by* $\det(\mathbf{G}_o - \lambda\mathbf{G}_e) = 0$.

First of all, we initialize $\mathbf{G}_0^* = \mathbf{I}$ *because* $\mathbf{U}$ *is unitary. And*

$$\mathbf{U}^{(0)} = \mathbf{U}\mathbf{G}_0^* = \mathbf{U} = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}.$$

Then define $\mathbf{Q}_{\geq 2} = \mathbf{P}(2,5,u_{*1}^{(0)})$ *to eliminate entries from 3 to 5 in the first column of* $\mathbf{U}^{(0)}$*, we have*

$$\mathbf{U}^{(1)} = \mathbf{Q}_{\geq 2}\mathbf{U}\mathbf{Q}_{\geq 2}^* = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}.$$

To eliminate the $(2,1)th$ *entry of* $\mathbf{U}^{(1)}$*, let* $\mathbf{G}_1^* = \mathbf{G}(\mathbf{U}^{(1)}(1:2,1))$ *and apply it only to the left of* $\mathbf{U}^{(1)}$*, then*

$$\mathbf{U}^{(2)} = \mathbf{G}_1^*\mathbf{U}^{(1)} = \mathbf{G}_1^*\mathbf{Q}_{\geq 2}\mathbf{U}\mathbf{Q}_{\geq 2}^* = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}.$$

Next, define $\mathbf{Q}_{\geq 3} = \mathbf{P}(3,5,u_{2*}^{*(2)})$ *to eliminate entries from 4 to 5 in the second row of* $\mathbf{U}^{(2)}$*. So*

$$\mathbf{U}^{(3)} = \mathbf{Q}_{\geq 3}\mathbf{U}^{(2)}\mathbf{Q}_{\geq 3}^{*} = \mathbf{Q}_{\geq 3}\mathbf{G}_1^{*}\mathbf{Q}_{\geq 2}\mathbf{U}\mathbf{Q}_{\geq 2}^{*}\mathbf{Q}_{\geq 3}^{*} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}.$$

Then we use a Givens matrix $\mathbf{G}_2^{*} = \mathbf{G}(\mathbf{U}^{*(3)}(2,2:3))$ *to make the* $(2,3)$*th entry of* $\mathbf{U}^{(3)}$ *as* 0*:*

$$\mathbf{U}^{(4)} = \mathbf{U}^{(3)}\mathbf{G}_2 = \mathbf{Q}_{\geq 3}\mathbf{G}_1^{*}\mathbf{Q}_{\geq 2}\mathbf{U}\mathbf{Q}_{\geq 2}^{*}\mathbf{Q}_{\geq 3}^{*}\mathbf{G}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{bmatrix}.$$

Now we have a 3×3 *submatrix left. Define* $\mathbf{Q}_{\geq 4} = \mathbf{P}(4,5,u_{*3}^{(4)})$*, we have*

$$\mathbf{U}^{(5)} = \mathbf{Q}_{\geq 4}\mathbf{U}^{(4)}\mathbf{Q}_{\geq 4}^{*} = \mathbf{Q}_{\geq 4}\mathbf{Q}_{\geq 3}\mathbf{G}_1^{*}\mathbf{Q}_{\geq 2}\mathbf{U}\mathbf{Q}_{\geq 2}^{*}\mathbf{Q}_{\geq 3}^{*}\mathbf{G}_2\mathbf{Q}_{\geq 4}^{*} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}.$$

Then let $\mathbf{G}_3^* = \mathbf{G}(\mathbf{U}^{(5)}(3:4,3))$ *and apply it to the left of* $\mathbf{U}^{(5)}$*:*

$$\mathbf{U}^{(6)} = \mathbf{G}_3^*\mathbf{U}^{(5)} = \mathbf{G}_3^*\mathbf{Q}_{\geq 4}\mathbf{Q}_{\geq 3}\mathbf{G}_1^*\mathbf{Q}_{\geq 2}\mathbf{U}\mathbf{Q}_{\geq 2}^*\mathbf{Q}_{\geq 3}^*\mathbf{G}_2\mathbf{Q}_{\geq 4}^* = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}.$$

As we only have a 2×2 *submatrix left, we can directly let* $\mathbf{G}_4^* = \mathbf{G}(\mathbf{U}^{*(6)}(4,4:5))$ *to zero out the* $(4,5)$*th entry of* $\mathbf{U}^{(6)}$*:*

$$\mathbf{U}^{(7)} = \mathbf{U}^{(6)}\mathbf{G}_4 = \mathbf{G}_3^*\mathbf{Q}_{\geq 4}\mathbf{Q}_{\geq 3}\mathbf{G}_1^*\mathbf{Q}_{\geq 2}\mathbf{U}\mathbf{Q}_{\geq 2}^*\mathbf{Q}_{\geq 3}^*\mathbf{G}_2\mathbf{Q}_{\geq 4}^*\mathbf{G}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \times \end{bmatrix}.$$

For the last diagonal entry, define $\mathbf{G}_5^* = \mathrm{diag}(1,1,1,1,\overline{\mathbf{U}^{(6)}(5,5)})$*, we have:*

$$\mathbf{U}^{(8)} = \mathbf{G}_5^*\mathbf{U}^{(7)} = \mathbf{G}_5^*\mathbf{G}_3^*\mathbf{Q}_{\geq 4}\mathbf{Q}_{\geq 3}\mathbf{G}_1^*\mathbf{Q}_{\geq 2}\mathbf{U}\mathbf{Q}_{\geq 2}^*\mathbf{Q}_{\geq 3}^*\mathbf{G}_2\mathbf{Q}_{\geq 4}^*\mathbf{G}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \mathbf{I}.$$

Now if we look at the left side of $\mathbf{U}$*,* $\mathbf{G}_1^*$ *can be moved to the left of* $\mathbf{Q}_{\geq 4}$*, and for the right hand side of* $\mathbf{U}$*,* $\mathbf{G}_2$ *can be moved to the right of* $\mathbf{Q}_{\geq 4}^*$*, then we have*

$$\mathbf{G}_5^*\mathbf{G}_3^*\mathbf{G}_1^*\mathbf{Q}_{\geq 4}\mathbf{Q}_{\geq 3}\mathbf{Q}_{\geq 2}\mathbf{U}\mathbf{Q}_{\geq 2}^*\mathbf{Q}_{\geq 3}^*\mathbf{Q}_{\geq 4}^*\mathbf{G}_2\mathbf{G}_4 = \mathbf{I},$$

let

$$\mathbf{G}_o^* = \mathbf{G}_5^*\mathbf{G}_3^*\mathbf{G}_1^*, \qquad \mathbf{G}_e = \mathbf{G}_0\mathbf{G}_2\mathbf{G}_4, \qquad \mathbf{Q} = \mathbf{Q}_{\geq 4}\mathbf{Q}_{\geq 3}\mathbf{Q}_{\geq 2},$$

then we have

$$\mathbf{G}_o^*\mathbf{Q}\mathbf{U}\mathbf{Q}^*\mathbf{G}_e = \mathbf{I} \Longrightarrow \mathbf{Q}\mathbf{U}\mathbf{Q}^* = \mathbf{G}_o^*\mathbf{G}_e^*$$

4.2 Modified Shift and Iteration

As introduced in [2], we adopt the same method for the choice of the single shift value for μ and double shift values for σ_1, σ_2.

4.2.1 Modified Single Shift Iteration

We will also describe one such iteration step, which transforms $\mathbf{G}_o^{(i-1)} - \mathbf{G}_e^{(i-1)}$ to $\mathbf{G}_o^{(i)} - \mathbf{G}_e^{(i)}$. For simplicity, we omit the super- and subscript i and let $n = size(\mathbf{G}_o)$.

Let

$$z = (\mathbf{G}_o - \mu\mathbf{G}_e)e_1 = \begin{bmatrix} \times \\ \times \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where μ can be chosen to be the last diagonal entry of $\mathbf{G}_o\mathbf{G}_e^*$. Compute $\mathbf{Q}_0 = \mathbf{P}(1,2,z)$ such that $\mathbf{Q}_0 z = ke_1$. Multiply $\mathbf{Q}_0$ to the left of the pencil $\mathbf{G}_o - \lambda\mathbf{G}_e$, then the original Schur parameter form

becomes

$$\mathbf{Q}_0\mathbf{G}_o = \begin{bmatrix} \times & \times & & & & & \\ \times & \times & & & & & \\ & & \times & \times & & & \\ & & \times & \times & & & \\ & & & & \times & \times & \\ & & & & \times & \times & \\ & & & & & & \ddots \end{bmatrix}, \mathbf{Q}_0\mathbf{G}_e = \begin{bmatrix} \times & + & + & & & & \\ + & \times & \times & & & & \\ & \times & \times & & & & \\ & & & \times & \times & & \\ & & & \times & \times & & \\ & & & & & \ddots & \end{bmatrix},$$

where $+$ stands for some additional possibly nonzero entry. Now we can apply Algorithm 4.1.1 to the newly generated unitary pencil $\mathbf{Q}_0(\mathbf{G}_o - \lambda\mathbf{G}_e)$. Because of the condensed forms for $\mathbf{G}_o, \mathbf{G}_e$, for the first step of the reduction, instead of applying $\mathbf{Q}_{\geq 1}$ to the left of $\mathbf{G}_o, \mathbf{G}_e$ which tries to eliminate entries from 2 to n in the first column of $\mathbf{Q}_0\mathbf{G}_e$, we define $\mathbf{P}_{\geq 1} = \mathbf{HP}(1,3,(\mathbf{G}_e)_{1*})$ to eliminate entries from 2 to 3 in the first row of $\mathbf{Q}_0\mathbf{G}_e$, that is

$$\mathbf{G}_{o,1} = \mathbf{Q}_0\mathbf{G}_o\mathbf{P}_{\geq 1}^* = \begin{bmatrix} \times & \times & + & & & & \\ \times & \times & + & & & & \\ + & + & \times & \times & & & \\ + & + & \times & \times & & & \\ & & & & \times & \times & \\ & & & & \times & \times & \\ & & & & & & \ddots \end{bmatrix}, \mathbf{G}_{e,1} = \mathbf{Q}_0\mathbf{G}_e\mathbf{P}_{\geq 1}^* = \begin{bmatrix} 1 & & & & & & \\ & \times & \times & & & & \\ & \times & \times & & & & \\ & & & \times & \times & & \\ & & & \times & \times & & \\ & & & & & \ddots & \end{bmatrix}.$$

Then define $\mathbf{Q}_{\geq 2} = \mathbf{P}(2,4,\mathbf{G}_{o,1}(2:4,1))$ to eliminate the entries from 2 to 4 in the first column of

$\mathbf{G}_{o,1}$, we have:

$$\tilde{\mathbf{G}}_{o,2} = \mathbf{Q}_{\geq 2}\mathbf{G}_{o,1} = \begin{bmatrix} \times & \times & + & & & & \\ \times & \times & + & + & & & \\ & + & \times & \times & & & \\ & + & \times & \times & & & \\ & & & & \times & \times & \\ & & & & \times & \times & \\ & & & & & & \ddots \end{bmatrix}, \tilde{\mathbf{G}}_{e,2} = \mathbf{Q}_{\geq 2}\mathbf{G}_{e,1} = \begin{bmatrix} 1 & & & & & & \\ & \times & \times & + & + & & \\ & \times & \times & + & + & & \\ & + & + & \times & \times & & \\ & & & \times & \times & & \\ & & & & & \ddots & \end{bmatrix}.$$

Next we are going to use a Givens matrix $\mathbf{G}_1 = \mathbf{G}(\tilde{\mathbf{G}}_{o,2}(1:2,1))$ to zero out the first entry in the second row of $\tilde{\mathbf{G}}_{o,2}$, that is:

$$\mathbf{G}_{o,2} = \mathbf{G}_1\tilde{\mathbf{G}}_{o,2} = \begin{bmatrix} 1 & & & & & & \\ & \times & + & + & & & \\ & + & \times & \times & & & \\ & + & \times & \times & & & \\ & & & & \times & \times & \\ & & & & \times & \times & \\ & & & & & & \ddots \end{bmatrix}, \mathbf{G}_{e,2} = \tilde{\mathbf{G}}_{e,2} = \begin{bmatrix} 1 & & & & & & \\ & \times & \times & + & + & & \\ & \times & \times & + & + & & \\ & + & + & \times & \times & & \\ & & & \times & \times & & \\ & & & & & \ddots & \end{bmatrix}.$$

Then define $\mathbf{P}_{\geq 2} = \mathbf{HP}(2,4,\mathbf{G}_{o,2}(2,2:4))$ to eliminate the entries from 3 to 4 in the second row of

$\mathbf{G}_{o,2}$, so:

$$\mathbf{G}_{o,3} = \mathbf{G}_{o,2}\mathbf{P}^*_{\geq 2} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \times & \times & & & \\ & & \times & \times & & & \\ & & & & \times & \times & \\ & & & & \times & \times & \\ & & & & & & \ddots \end{bmatrix}, \mathbf{G}_{e,3} = \mathbf{G}_{e,2}\mathbf{P}^*_{\geq 2} = \begin{bmatrix} 1 & & & & & \\ & \times & \times & + & + & \\ & \times & \times & + & + & \\ & + & + & \times & \times & \\ & + & + & \times & \times & \\ & & & & & \ddots \end{bmatrix}.$$

In order to eliminate entries from 4 to 5 in the second row of $\mathbf{G}_{e,3}$, define $\mathbf{P}_{\geq 3} = \mathbf{P}(3,5,\mathbf{G}_{e,3}(2,3:5))$ and apply its conjugate transpose to the right of $\mathbf{G}_{o,3}, \mathbf{G}_{e,3}$, then

$$\tilde{\mathbf{G}}_{o,4} = \mathbf{G}_{o,3}\mathbf{P}^*_{\geq 3} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \times & \times & + & & \\ & & \times & \times & + & & \\ & & + & + & \times & \times & \\ & & + & + & \times & \times & \\ & & & & & & \ddots \end{bmatrix}, \tilde{\mathbf{G}}_{e,4} = \mathbf{G}_{e,3}\mathbf{P}^*_{\geq 3} = \begin{bmatrix} 1 & & & & & \\ & \times & \times & & & \\ & \times & \times & + & + & \\ & + & + & \times & \times & \\ & + & + & \times & \times & \\ & & & & & \ddots \end{bmatrix}.$$

Then we can use a Givens matrix $\mathbf{G}_2 = \mathbf{G}(\tilde{\mathbf{G}}_{e,4}(2,2:3))$ to eliminate the $(2,3)th$ entry of $\tilde{\mathbf{G}}_{e,4}$, we

have

$$\mathbf{G}_{o,4} = \tilde{\mathbf{G}}_{o,4} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \times & \times & + & & \\ & & \times & \times & + & & \\ & & + & + & \times & \times & \\ & & + & + & \times & \times & \\ & & & & & & \ddots \end{bmatrix}, \mathbf{G}_{e,4} = \tilde{\mathbf{G}}_{e,4}\mathbf{G}_2^* = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \times & + & + & & \\ & & + & \times & \times & & \\ & & + & \times & \times & & \\ & & & & & \ddots & \end{bmatrix}.$$

Then we can follow exactly the process as in Algorithm 4.1.1 to convert $\mathbf{G}_o^{(i)}, \mathbf{G}_e^{(i)}$ back to identities, where we can get a new pair of $\mathbf{G}_o^{(i+1)}$ and $\mathbf{G}_e^{(i+1)}$. In each iterative step, the maximum length of the vectors dealing with Householder reflectors is reduced to 3 because of the tridiagonal structure of $\mathbf{G}_o, \mathbf{G}_e$. After each iterative step, deflation needs to be considered if any zero complementary Schur parameters appears. If that is the case, then we subdivide $\mathbf{G}_o^{(i)}, \mathbf{G}_e^{(i)}$ accordingly and apply the algorithm separately until all complementary Schur parameters become zero. Then the diagonal entries of $\mathbf{G}_o(i), \mathbf{G}_e^{(i)}$ provide the information for the eigenvalues of the original unitary matrix pencil.

4.2.2 Modified Double Shifts Iteration

When $\mathbf{G}_o - \lambda \mathbf{G}_e$ is a real Schur parameter pencil, we can also consider double shifts iteration as introduced in previous chapter to keep the arithmetic real. The process of the modified double shift iteration steps are similar to the one introduced in previous chapter.

Define

$$z = (\mathbf{G}_o - \sigma_1 \mathbf{G}_e)(\mathbf{G}_e^* - \sigma_2 \mathbf{G}_o^*)e_1 = \begin{bmatrix} \times \\ \times \\ \times \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where σ_1, σ_2 can be chosen to be the eigenvalues of the trailing 2×2 principal submatrix of $\mathbf{G}_o \mathbf{G}_e^*$. Let $\mathbf{Q}_0 = \mathbf{P}(1,3,z)$ such that

$$\mathbf{Q}_0 z = \beta e_1$$

Then apply $\mathbf{Q}_0$ to the right of $\mathbf{G}_o$ and $\mathbf{G}_e$, we have

$$\mathbf{Q}_0\mathbf{G}_o = \begin{bmatrix} \times & \times & + & + & & & \\ \times & \times & + & + & & & \\ + & + & \times & \times & & & \\ & & \times & \times & & & \\ & & & & \times & \times & \\ & & & & \times & \times & \\ & & & & & & \ddots \end{bmatrix}, \mathbf{Q}_0\mathbf{G}_e = \begin{bmatrix} \times & + & + & & & & \\ + & \times & \times & & & & \\ + & \times & \times & & & & \\ & & & \times & \times & & \\ & & & \times & \times & & \\ & & & & & \ddots & \end{bmatrix}$$

Then we are going to use the algorithm 4.1.1 again to regain the Schur parameter form from $\mathbf{Q}_0(\mathbf{G}_o - \lambda \mathbf{G}_e)$. The difference is that, similar to the modified single shift iteration, we need to eliminate the entries from 2 to 3 in the first row of $Q_0\mathbf{G}_e$ instead of the entries from 2 to 3 in the first column of $\mathbf{Q}_0\mathbf{G}_e$; and in each iteration step, we can use a vector with length at most 3 to construct Householder reflectors. We will not specifically show the reduction process here.

During the whole process of iteration steps, we need to monitor the values of complementary Schur parameters. Deflation may be considered if there is zero among the values as well.

4.3 Numerical Examples

In this section, all codes are run by MATLAB R2010a, Version 7.10.0.499, 64-bit. Unitary matrices are generated by taking the unitary matrix of QR decomposition of a random square matrix from MATLAB. We are going to give some examples for the modified Schur parameter pencil method.

Example 10 *Given a complex unitary pencil* $\mathbf{U}-\lambda\mathbf{V}$*, where* $\mathbf{U},\mathbf{V}\in C^{4\times 4}$*,*

$$\mathbf{U}=\begin{bmatrix} 0.2808+0.4609i & -0.0897-0.1053i & 0.0696-0.3070i & -0.2179-0.7369i \\ -0.5684+0.2176i & -0.2769-0.5602i & -0.0767-0.2492i & -0.3431+0.2313i \\ 0.4910-0.1601i & -0.1793-0.7465i & 0.0152+0.2524i & 0.2812+0.0286i \\ 0.2653+0.0355i & 0.0226+0.0214i & 0.6955-0.5341i & 0.0298+0.3969i \end{bmatrix},$$

$$\mathbf{V}=\begin{bmatrix} -0.1016+0.7520i & 0.1092+0.2283i & 0.1730+0.2537i & -0.4746-0.2015i \\ -0.0378+0.1693i & -0.4063-0.0611i & 0.3537+0.5791i & 0.4198+0.4054i \\ -0.5325-0.0157i & -0.1767-0.2210i & 0.4524-0.4792i & -0.2934+0.3402i \\ -0.2848+0.1707i & 0.8046-0.1991i & 0.1013+0.0257i & 0.4221+0.1164i \end{bmatrix}.$$

After Phase I, $\mathbf{U},\mathbf{V}$ *are reduced to identities, and associated* $\mathbf{G}_o,\mathbf{G}_e$ *are found:*

$$\mathbf{G}_o=\begin{bmatrix} 0.0479-0.1325i & -0.9560-0.2572i & 0 & 0 \\ 0.9560-0.2572i & 0.0479+0.1325i & 0 & 0 \\ 0 & 0 & -0.1085+0.8416i & 0.3617-0.3863i \\ 0 & 0 & -0.3617-0.3863i & -0.1085-0.8416i \end{bmatrix},$$

$$\mathbf{G}_e=\begin{bmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & -0.0056-0.1983i & -0.2399+0.9503i & 0 \\ 0 & 0.2399+0.9503i & -0.0056+0.1983i & 0 \\ 0 & 0 & 0 & -0.9992+0.0391i \end{bmatrix}.$$

Then after Phase II, both $\mathbf{G}_o, \mathbf{G}_e$ *are converged to diagonals with single shift iteration:*

$$\mathbf{G}_o = \begin{bmatrix} -0.9124+0.4093i & 0 & 0 & 0 \\ 0 & 0.1843-0.9829i & 0 & 0 \\ 0 & 0 & -0.0817+0.9967i & 0 \\ 0 & 0 & 0 & -0.0817-0.9967i \end{bmatrix},$$

$$\mathbf{G}_e = \begin{bmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0.2341+0.9722i & 0 \\ 0 & 0 & 0 & -0.9992+0.0391i \end{bmatrix}.$$

Thus, the eigenvalue problem of $\mathbf{U}-\lambda\mathbf{V}$ *becomes the eigenvalue problem of* $\mathbf{G}_o-\lambda\mathbf{G}_e$*, where the eigenvalues appear on the diagonal of* $\mathbf{G}_o\mathbf{G}_e^*$*:*

$$\mathbf{G}_o\mathbf{G}_e^* = \begin{bmatrix} 0.1843-0.9829i & 0 & 0 & 0 \\ 0 & 0.9498+0.3128i & 0 & 0 \\ 0 & 0 & -0.9124+0.4093i & 0 \\ 0 & 0 & 0 & 0.0427+0.9991i \end{bmatrix}.$$

Figure 4.1 is the error information provided by semilogy() between eigenvalues σ^{sch} *found with the modified Schur parameter pencil method and those* σ^{eig} *obtained by eig*$(\mathbf{UV}^*)$ *in MATLAB .*

Example 11 *Given a single complex unitary matrix* $\mathbf{U}$*,where* $\mathbf{U} \in C^{4\times4}$*,*

$$\mathbf{U} = \begin{bmatrix} 0.0097+0.5203i & -0.6294+0.2355i & -0.4589-0.2055i & -0.1186-0.1041i \\ -0.0110-0.5016i & -0.3910-0.2316i & -0.4087+0.4651i & 0.3521-0.1858i \\ 0.0548+0.6318i & 0.2936-0.4265i & -0.0625+0.4621i & 0.0947-0.3214i \\ 0.0928+0.2583i & -0.2291+0.1462i & 0.3773+0.0643i & 0.7569+0.3625i \end{bmatrix}$$

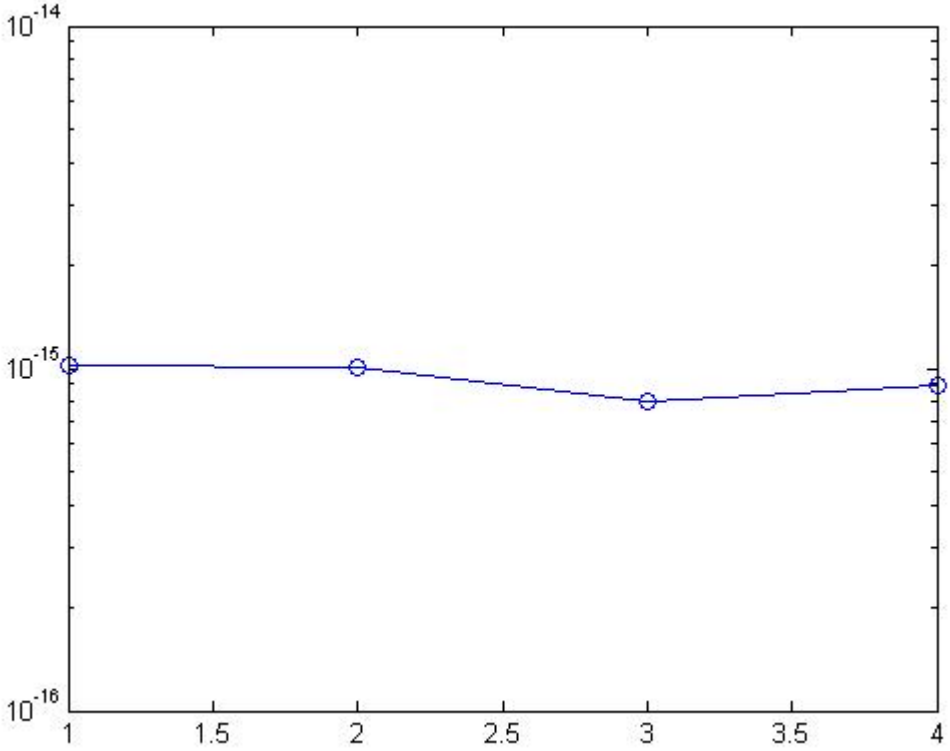

Figure 4.1: $y = |\sigma_i^{sch} - \sigma_i^{eig}|, x = i$

After Phase I, $\mathbf{U}$ *is reduced to identity, and associated* $\mathbf{G}_o, \mathbf{G}_e$ *are:*

$$\mathbf{G}_o = \begin{bmatrix} 0.0097+0.5203i & -0.0187+0.8537i & 0 & 0 \\ 0.0187+0.8537i & 0.0097-0.5203i & 0 & 0 \\ 0 & 0 & 0.6460-0.0949i & -0.2380+0.7190i \\ 0 & 0 & 0.2380+0.7190i & 0.6460+0.0949i \end{bmatrix},$$

$$\mathbf{G}_e = \begin{bmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & -0.3368+0.0389i & -0.0465-0.9396i & 0 \\ 0 & 0.0465-0.9396i & -0.3368-0.0389i & 0 \\ 0 & 0 & 0 & 0.8607+0.5091i \end{bmatrix}.$$

After Phase II, $\mathbf{G}_o, \mathbf{G}_e$ *are reduced to diagonal matrices with single shift iteration. We found the eigenvalues of* $\mathbf{U}$ *according to the diagonal entries of* $\mathbf{G}_o\mathbf{G}_e^*$*:*

$$\mathbf{G}_o\mathbf{G}_e' = \begin{bmatrix} -0.8158-0.5784i & 0 & 0 & 0 \\ 0 & 0.9967+0.0810i & 0 & 0 \\ 0 & 0 & 0.6532+0.7572i & 0 \\ 0 & 0 & 0 & -0.5211+0.8535i \end{bmatrix}.$$

Figure 4.2 is the error information provided by semilogy() between eigenvalues σ^{sch} *found with the modified Schur parameter pencil method and those* σ^{eig} *obtained by eig*$(\mathbf{UV}^*)$ *in MATLAB .*

Example 12 *Given a real unitary pencil* $\mathbf{U}-\lambda\mathbf{V}$*, where* $\mathbf{U}, \mathbf{V} \in R^{5\times 5}$*,*

$$\mathbf{U} = \begin{bmatrix} 0.8000 & 0.1969 & 0.3397 & 0.4537 & 0.0012 \\ -0.0032 & -0.1559 & -0.5200 & 0.4608 & 0.7021 \\ -0.5386 & 0.4191 & 0.1279 & 0.6726 & -0.2561 \\ -0.2392 & 0.1251 & 0.6971 & -0.1562 & 0.6456 \\ -0.1131 & -0.8635 & 0.3344 & 0.3241 & -0.1572 \end{bmatrix},$$

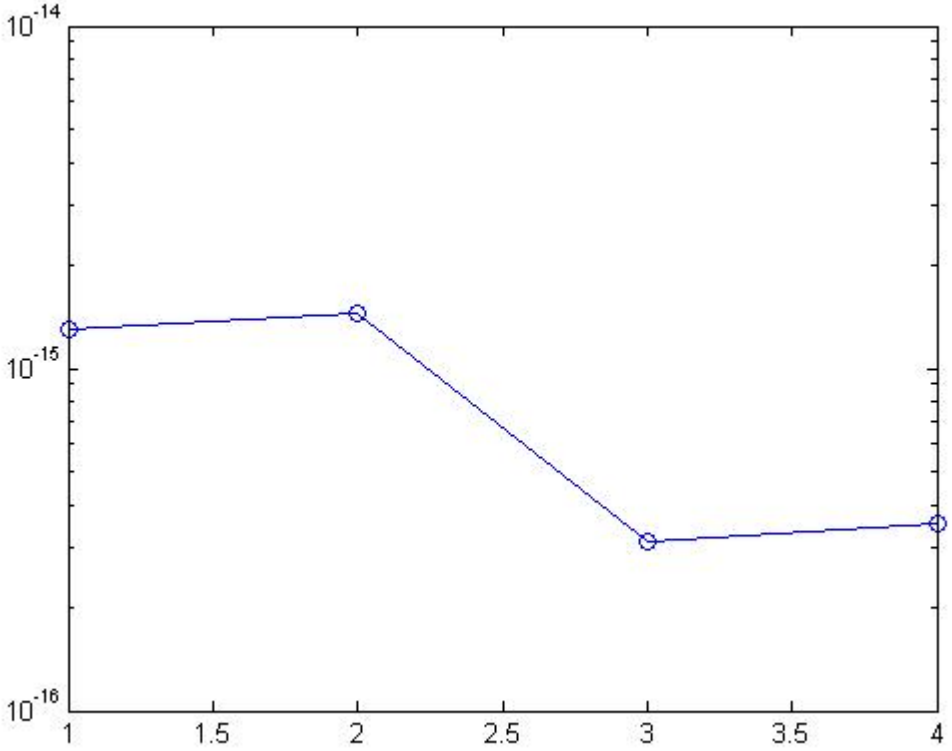

Figure 4.2: $y = |\sigma_i^{sch} - \sigma_i^{eig}|, x = i$

$$\mathbf{V} = \begin{bmatrix} -0.5949 & -0.2246 & -0.3492 & -0.3896 & -0.5674 \\ 0.0617 & 0.2304 & -0.3520 & -0.7166 & 0.5529 \\ -0.7382 & 0.1748 & -0.1117 & 0.4212 & 0.4844 \\ 0.2978 & -0.3338 & -0.8122 & 0.3686 & 0.0666 \\ -0.0928 & -0.8686 & 0.2865 & -0.1462 & 0.3652 \end{bmatrix}.$$

After Phase I, $\mathbf{U}, \mathbf{V}$ *are reduced to identities, and associated* $\mathbf{G}_o, \mathbf{G}_e$ *are found:*

$$\mathbf{G}_o = \begin{bmatrix} -0.1393 & 0.9903 & 0 & 0 & 0 \\ -0.9903 & -0.1393 & 0 & 0 & 0 \\ 0 & 0 & -0.6768 & -0.7362 & 0 \\ 0 & 0 & 0.7362 & -0.6768 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix},$$

$$\mathbf{G}_e = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 0 & -0.1991 & -0.9800 & 0 & 0 \\ 0 & 0.9800 & -0.1991 & 0 & 0 \\ 0 & 0 & 0 & 0.1209 & 0.9927 \\ 0 & 0 & 0 & -0.9927 & 0.1209 \end{bmatrix}.$$

Then after Phase II, both $\mathbf{G}_o, \mathbf{G}_e$ *are converged to diagonals with double shifts iteration:*

$$\mathbf{G}_o = \begin{bmatrix} 0.4892 - 0.8722i & 0 & 0 & 0 & 0 \\ 0 & 0.4892 + 0.8722i & 0 & 0 & 0 \\ 0 & 0 & -0.9581 - 0.2865i & 0 & 0 \\ 0 & 0 & 0 & -0.9581 + 0.2865i & 0 \\ 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix},$$

$$\mathbf{G}_e = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Thus, the eigenvalue problem of $\mathbf{U} - \lambda \mathbf{V}$ *becomes the eigenvalue problem of* $\mathbf{G}_o - \lambda \mathbf{G}_e$*, where the eigenvalues appear on the diagonal of* $\mathbf{G}_o \mathbf{G}_e^*$*:*

$$\mathbf{G}_o \mathbf{G}_e^* = \begin{bmatrix} 0.4892 - 0.8722i & 0 & 0 & 0 & 0 \\ 0 & 0.4892 + 0.8722i & 0 & 0 & 0 \\ 0 & 0 & -0.9581 - 0.2865i & 0 & 0 \\ 0 & 0 & 0 & -0.9581 + 0.2865i & 0 \\ 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix}.$$

Figure 4.3 is the error information provided by semilogy() between eigenvalues σ^{sch} *found with the modified Schur parameter pencil method and those* σ^{eig} *obtained by eig*$(\mathbf{U}\mathbf{V}^*)$ *in MATLAB .*

Example 13 *Given a real unitary matrix* $\mathbf{U}$*, where* $\mathbf{U} \in R^{5\times 5}$*,*

$$\mathbf{U} = \begin{bmatrix} 0.6683 & -0.0590 & -0.1033 & -0.2860 & 0.6763 \\ 0.4695 & -0.6019 & -0.3716 & 0.1836 & -0.4955 \\ -0.1960 & -0.2385 & -0.1227 & 0.8035 & 0.4939 \\ 0.1898 & -0.3528 & 0.9098 & 0.1026 & -0.0359 \\ -0.5085 & -0.6730 & -0.0920 & -0.4778 & 0.2277 \end{bmatrix},$$

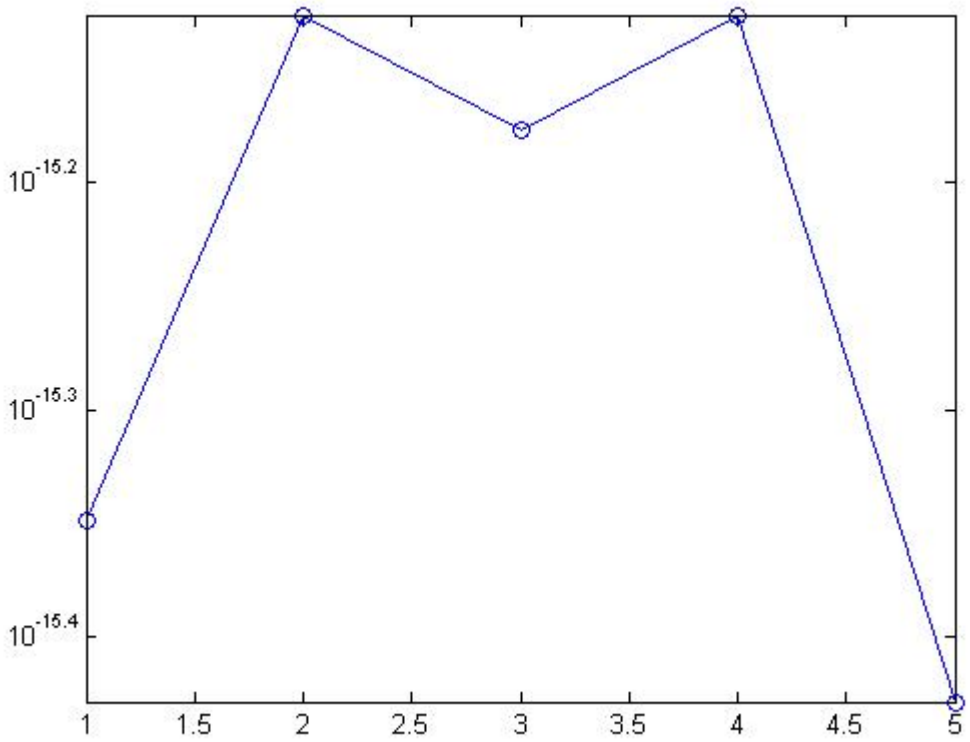

Figure 4.3: $y = |\sigma_i^{sch} - \sigma_i^{eig}|, x = i$

After Phase I, $\mathbf{U}$ *is reduced to identities, and associated* $\mathbf{G}_o, \mathbf{G}_e$ *are found:*

$$\mathbf{G}_o = \begin{bmatrix} 0.6683 & 0.7439 & 0 & 0 & 0 \\ -0.7439 & 0.6683 & 0 & 0 & 0 \\ 0 & 0 & -0.1567 & -0.9876 & 0 \\ 0 & 0 & 0.9876 & -0.1567 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix},$$

$$\mathbf{G}_e = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0.7330 & 0.6803 & 0 & 0 \\ 0 & -0.6803 & 0.7330 & 0 & 0 \\ 0 & 0 & 0 & -0.9123 & 0.4095 \\ 0 & 0 & 0 & -0.4095 & -0.9123 \end{bmatrix}.$$

Then after Phase II, both $\mathbf{G}_o, \mathbf{G}_e$ *are converged to diagonals with double shifts iteration:*

$$\mathbf{G}_o = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0.6036 - 0.7973i & 0 & 0 & 0 \\ 0 & 0 & 0.6036 + 0.7973i & 0 & 0 \\ 0 & 0 & 0 & -0.9667 - 0.2560i & 0 \\ 0 & 0 & 0 & 0 & -0.9667 + 0.2560i \end{bmatrix},$$

$$\mathbf{G}_e = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Thus, the eigenvalue problem of $\mathbf{U} - \lambda \mathbf{I}$ *becomes the eigenvalue problem of* $\mathbf{G}_o - \lambda \mathbf{G}_e$*, where the*

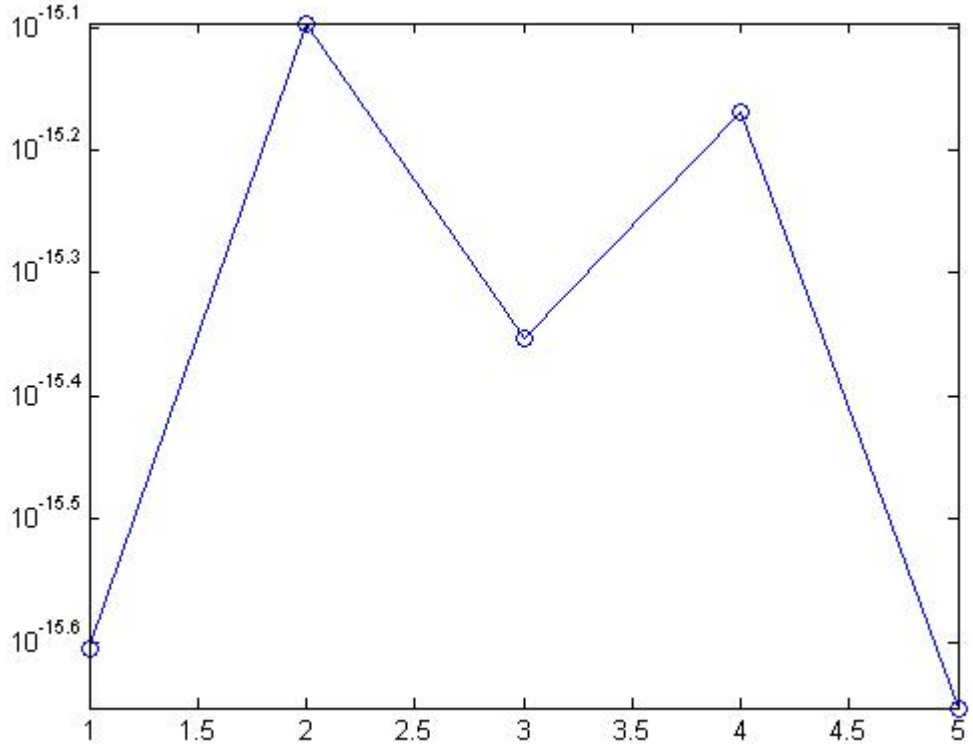

Figure 4.4: $y = |\sigma_i^{sch} - \sigma_i^{eig}|, x = i$

eigenvalues appear on the diagonal of $\mathbf{G}_o\mathbf{G}_e^*$*:*

$$\mathbf{G}_o\mathbf{G}_e^* = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0.6036 - 0.7973i & 0 & 0 & 0 \\ 0 & 0 & 0.6036 + 0.7973i & 0 & 0 \\ 0 & 0 & 0 & -0.9667 - 0.2560i & 0 \\ 0 & 0 & 0 & 0 & -0.9667 + 0.2560 \end{bmatrix}.$$

Figure 4.4 is the error information provided by semilogy() between eigenvalues σ^{sch} *found with the modified Schur parameter pencil method and those* σ^{eig} *obtained by eig*$(\mathbf{U})$ *in MATLAB .*

Chapter 5

Conclusion and Future Work

This paper reviews the method of using the Schur parameter pencil to solve eigenvalue problems for unitary matrices introduced by [2]. The method is divided into two phases:

Phase I: Reducing a unitary pencil $\mathbf{U} - \lambda\mathbf{V}$ or unitary matrix $\mathbf{U}$ to a Schur parameter pencil $\mathbf{G}_o - \lambda\mathbf{G}_e$.

Phase II: Converge $\mathbf{G}_o, \mathbf{G}_e$ to diagonal forms with shifted iterative method.

In Phase I, [2] tries to find appropriate unitary matrices $\mathbf{P}, \mathbf{Q}$ to reduce $\mathbf{U}, \mathbf{V}$ simultaneously to $\mathbf{G}_o, \mathbf{G}_e$, i.e. $\mathbf{QUP} = \mathbf{G}_o, \mathbf{QVP} = \mathbf{G}_e$. During the process of the reduction, norms between vectors are frequently compared for the purpose of computational stability. In Phase II, both single-shifted and double-shifted iterative methods are discussed to converge $\mathbf{G}_o, \mathbf{G}_e$ obtained from Phase I to diagonal forms. In the case of the single shift iteration, μ is chosen to be the last diagonal entry of $\mathbf{G}_o\mathbf{G}_e^*$ in each iterative step, while in the case of the double shifts iteration, σ_1, σ_2 are chosen to be the eigenvalues of the trailing 2×2 principal submatrix of $\mathbf{G}_o\mathbf{G}_e^*$. A new Householder reflector $\mathbf{Q}_0$ is constructed according to the shift chosen. $\mathbf{Q}_0$ is then applied to the left of $\mathbf{G}_o, \mathbf{G}_e$,which breaks the Schur parameter form of the original $\mathbf{G}_o, \mathbf{G}_e$. Then the reduction method in Phase I is applied to $\mathbf{Q}_0\mathbf{G}_o, \mathbf{Q}_0\mathbf{G}_e$ to regain the Schur parameter form. This algorithm ends when both $\mathbf{G}_o, \mathbf{G}_e$ converge to diagonal forms.

Based on the algorithm introduced in [2] and combined with the idea from [8], we modified the method of reducing a unitary pencil to a Schur parameter pencil to avoid frequent comparison between the norms of vectors. Our modified algorithm still can be divided into two phases similar to [2]. In phase I, instead of directly converting $\mathbf{U},\mathbf{V}$ to $\mathbf{G}_o,\mathbf{G}_e$ by $\mathbf{Q},\mathbf{P}$, we aim at converting $\mathbf{U},\mathbf{V}$ simultaneously to identities by a sequence of modified Householder reflectors and Givens matrices. Then we can extract $\mathbf{G}_o,\mathbf{G}_e$ from the sequence of Givens matrices, i.e. $\mathbf{G}_o\mathbf{QUP} = I, \mathbf{QVPG}_e = \mathbf{I}$; in phase II, we adopt the same choice of shifts for iterative steps. The difference is that in the process of regaining the Schur parameter form of $\mathbf{G}_o,\mathbf{G}_e$,we use the reduction algorithm from our phase I again. In each iterative step, we extract new $\mathbf{G}_o^{(n+1)},\mathbf{G}_e^{(n+1)}$ from the sequence of Givens matrices after converting $\mathbf{G}_o^{(n)},\mathbf{G}_e^{(n)}$ to identities.

In the future, we need to prove the stability of our algorithm which should be backward stable. In addition, MATLAB coding can be improved to decrease flops required in our algorithm and to enhance stability for computation as well.

References

[1] G.S. Ammar, W.B. Gragg, L. Reichel, On the Eigenproblem for Orthogonal Matrices. *in Proc. 25th IEEE Conf. on Decision and Control*, Athens 1986.

[2] A. Bunse-Gerstner, L. Elsner, Schur Parameter Pencils for the Solution of the Unitary Eigenproblem. *Linear Algebra and its Applications*, Volumes 154-156, Pages 741-778, August - October 1991.

[3] G. H. Golub, C. F. Van Loan. Matrix Computations, Third Edition. Baltimore, MD: The John Hopkins University Press, 96. Print.

[4] W.B. Gragg, The QR Algorithm for Unitary Hessenberg Matrices. *J. Comput. Appl. Math*, 16, Pages 1-8, 1986.

[5] W.B. Gragg, L. Reichel. A Divide and Conquer Algorithm for Unitary and Orthogonal Eigenproblems. *Numer. Math*, 57, Pages 695-718, 1990.

[6] W.B. Gragg, T.L. Wang. Convergence of the Shifted QR Algorithm for Unitary Hessenberg Matrices. *Report NPS-53-90-008 Naval Postgraduate School*, Monterey, California, 1990.

[7] W.B. Gragg, T.L. Wang. Convergence of the Unitary Hessenberg QR Algorithm with Unimodular Shifts. *Report NPS-53-90-008 Naval Postgraduate School*, Monterey, California, 1990.

[8] L. Reichel, H. Xu. A CS decomposition method for eigenvalue problem of orthogonal matrices. March, 29th, 2013.

[9] L. N. Trefethen, D. Bau III. Numerical Linear Algebra. Philadelphia, PA: Society for Industrial and Applied Mathematics, 97. Print.

Appendix A

Programming Codes in MATLAB

A.1 Package Instruction

This MATLAB programming code package contains:

four main functions:

unitarypencil − single.m: main function to find eigenvalue pairs for a unitary pencil with single shifted iteration.

unitarymatrix − single.m: main function to find eigenvalues for a unitary matrix with single shifted iteration.

unitarypencil − double.m: main function to find eigenvalue pairs for a unitary pencil with double shifted iteration.

unitarymatrix − double.m: main function to find eigenvalues for a unitary matrix with double shifted iteration.

other functions:

decomposition.m: reduce unitary matrices $\mathbf{U}, \mathbf{V}$ from the unitary pencil $\mathbf{U} - \lambda\mathbf{V}$ to identities. Find $\mathbf{G}_o, \mathbf{G}_e$ for the Schur parameter pencil.

singledecom.m:reduce unitary matrix$\mathbf{U}$ to identy. Find $\mathbf{G}_o, \mathbf{G}_e$ for the Schur parameter pencil.

my2by2matrix.m: directly find eigenvalues for 2×2 matrices.

myloop.m: function for single-shifted iterative method to converge $\mathbf{G}_o, \mathbf{G}_e$ to diagonal forms.

doubleloop.m: function for double-shifted iterative method to converge $\mathbf{G}_o, \mathbf{G}_e$ to diagonal forms.

chase.m: function for a single iterative step with single-shifted method.

doublechase.m: function for a single iterative step with double-shifted method.

givensg.m: function that generates a givens matrix for a given vector.

houseg.m: function that generates a householder reflector for a given vector.

A.2 Programming Codes

A.2.1 unitarypencil-single.m

```
-----------------------Generating random unitary U and V -----------------
U = eye(5);                     % The dimension of U can be changed if needed.
X1 = (randn(n))/sqrt(2);
[Q1,R1] = qr(X1);
U = Q1;


X2 = (randn(n))/sqrt(2);
[Q2,R2] = qr(X2);
V = Q2;
------------------end of generating random unitary U and V -----------------
```

```
n = length(U);
myDiag_U = eye(length(U));     % Initialize the diagonal matrix G_o converges to.
myDiag_V = myDiag_U;           % Initialize the diagonal matrix G_e converges to.

if n ==2                       % If the dimension of U and V is 2, then find
                               % the eigenvalue pairs directly.
 [Q,P,e1,e2] = my2by2matrix(U,V);
 Q
 P
 myDiag_U = [e2 0;0 e1]
 myDiag_V = eye(2)

elseif n >2                    % If the dimension of U and V is bigger than 2,
                               % then do the following.
  [G_o,G_e,Q,P] = decomposition(U,V);       % Phase I, reduce U, V to identities
                                           % and find G_o,G_e.

-----------------------check for deflation before iteration -----------------
   j = 1;
   for i = 1:n-1                 % Trace from top to bottom about the zeros in
                                 % G_o and G_e, whenever meet a zero, pick the
                                 % submatrix with the previous block, then
                                 % continue with the rest of the matrix.

     if ((mod(i,2)==1)&&(abs(G_o(i+1,i))<10^(-14)))||
        ((mod(i,2)==0)&&(abs(G_e(i+1,i))<10^(-14)))
```

```
    if j == i                    % If the block is 1 by 1 matrix, then
                                 % directly assign the scalar to the
                                 % diagonal matrix.
       myDiag_U(j,j) = G_o(j,j);
       myDiag_V(j,j) = G_e(j,j);

    else                         % If the block is not a scalar matrix.
       if mod(j,2)==1            % If the row starts in odd number.
          G_o1 = G_o(j:i,j:i);
          G_e1 = G_e(j:i.j:i);
       else                      % If the row starts in even number, then
                                 % switch the place of G_o and G_e.
          G_o1 = G_e(j:i,j:i);
          G_e1 = G_o(j:i,j:i);
       end
       [G_o2,G_e2,myQ,myP] = myloop(G_o1,G_e1);
                                 % Do chase for the block found.
       myDiag_U(j:i,j:i) = G_o2;
       myDiag_V(j:i,j:i) = G_e2;
       Q = blkdiag(eye(j-1),myQ,eye(n-i))*Q;
       P = P*blkdiag(eye(j-1),myP,eye(n-1));
     end
  j = i+1;

elseif (i==n-1)&&(abs(G_o(n,n-1))>10^(-15)||abs(G_e(n,n-1))>10^(-15))
                                 % If there is no zeros among complimentary
```

```
                                        % Schur parameters, then directly begin the
                                        % iteration

      [G_o2,G_e2,myQ,myP] = myloop(G_o,G_e);     % Do chase for the block found
      myDiag_U(j:n,j:n) = G_o2   % Display the diagonal matrix converged by G_o.
      myDiag_V(j:n,j:n) = G_e2   % Display the diagonal matrix converged by G_e.
      Q = myQ*Q                       % Display the unitary matrix Q.
      P = P*myP                       % Display the unitary matrix P.
    end
  end
end
```

A.2.2 unitarymatrix-single.m

This function is similar to 'unitarypencil-single.m', so we omit some explaination here.

```
-----------------------Generating a random unitary matrix U -----------------
U = eye(2);
X1 = (randn(n))/sqrt(2);
[Q1,R1] = qr(X1);
U = Q1;
-----------------end of Generating random unitary matrix U ----------------
n = length(U);
myDiag_U = eye(length(U));     % Initialize the diagonal matrix G_o converges to.
myDiag_V = myDiag_U;           % Initialize the diagonal matrix G_e converges to.

if n ==2                         % If dim(U)=2, then find eigenvalues directly.
   [Q,P,e1,e2] = my2by2matrix(U,V);
```

```
   myDiag_U = [e2 0;0 e1];
   myDiag_V = eye(2);
   display('Q*U*P - myDiag_U =')
   disp(Q*U*P - myDiag_U)

elseif n >2
   [G_o,G_e,Q] = singledecom(U);        % Reduce U to identities, find G_o,G_e.
   P = eye(n);                          % Initialize P such that PUP^*=G_oG_e^*.
   G_e = G_e';
   j = 1;

-----------------------check for deflation before iteration -----------------
   for i = 1:n-1
      if ((mod(i,2)==1)&&(abs(G_o(i+1,i))<10^(-15)))||
         ((mod(i,2)==0)&&(abs(G_e(i+1,i))<10^(-15)))


         if j == i
            myDiag_U(j,j) = G_o(j,j);
            myDiag_V(j,j) = G_e(j,j);

         else
            if mod(j,2)==1
               G_o1 = G_o(j:i,j:i);
               G_e1 = G_e(j:i.j:i);
            else
               G_o1 = G_e(j:i,j:i);
```

```
                G_e1 = G_o(j:i,j:i);
            end
            [G_o2,G_e2,myQ,myP] = myloop(G_o1,G_e1);
            myDiag_U(j:i,j:i) = G_o2;
            myDiag_V(j:i,j:i) = G_e2;
            Q = blkdiag(eye(j-1),myQ,eye(n-i))*Q;
            P = P*blkdiag(eye(j-1),myP,eye(n-1));
        end
     j = i+1;

    elseif (i==n-1)&&(abs(G_o(n,n-1))>10^(-15)||abs(G_e(n,n-1))>10^(-15))
        [G_o2,G_e2,myQ,myP] = myloop(G_o,G_e);
        myDiag_U(j:n,j:n) = G_o2;
        myDiag_V(j:n,j:n) = G_e2;
        Q = myQ*Q;
     end
   end
 end

 display('eigenvalues for the matrix U are:')
 myDiag_U*myDiag_V       % Eigenvalues of U are displayed in a diagonal matrix.
```

A.2.3 unitarypencil-double.m

This function is similar to 'unitarypencil-single.m', so we omit some explaination here.

```
----------------------Generating random unitary U and V -------------------
U = eye(5);                     % The dimension of U can be changed if needed.
X1 = (randn(n))/sqrt(2);
```

```
[Q1,R1] = qr(X1);
U = Q1;

X2 = (randn(n))/sqrt(2);
[Q2,R2] = qr(X2);
V = Q2;
------------------end of generating random unitary U and V -----------------
n = length(U);
myDiag_U = eye(length(U));
myDiag_V = myDiag_U;

if n ==2
 [Q,P,e1,e2] = my2by2matrix(U,V);
 Q
 P
 myDiag_U = [e2 0;0 e1]
 myDiag_V = eye(2)

elseif n >2
  [G_o,G_e,Q,P] = decomposition(U,V);

----------------------check for deflation before iteration ----------------
   j = 1;
   for i = 1:n-1

     if ((mod(i,2)==1)&&(abs(G_o(i+1,i))<10^(-14)))||
        ((mod(i,2)==0)&&(abs(G_e(i+1,i))<10^(-14)))
```

```
    if j == i
       myDiag_U(j,j) = G_o(j,j);
       myDiag_V(j,j) = G_e(j,j);

    else
       if mod(j,2)==1
          G_o1 = G_o(j:i,j:i);
          G_e1 = G_e(j:i.j:i);
       else
          G_o1 = G_e(j:i,j:i);
          G_e1 = G_o(j:i,j:i);
       end
       [G_o2,G_e2,myQ,myP] = doubleloop(G_o1,G_e1);
       myDiag_U(j:i,j:i) = G_o2;
       myDiag_V(j:i,j:i) = G_e2;
       Q = blkdiag(eye(j-1),myQ,eye(n-i))*Q;
       P = P*blkdiag(eye(j-1),myP,eye(n-1));
     end
  j = i+1;

elseif (i==n-1)&&(abs(G_o(n,n-1))>10^(-15)||abs(G_e(n,n-1))>10^(-15))

  [G_o2,G_e2,myQ,myP] = doubleloop(G_o,G_e);
  myDiag_U(j:n,j:n) = G_o2
  myDiag_V(j:n,j:n) = G_e2
```

```
        Q = myQ*Q
        P = P*myP
    end
  end
end
```

A.2.4 unitarymatrix-double.m

This function is similar to 'unitarymatrix-single.m', so we omit some explaination here.

```
-----------------------Generating a random unitary matrix U ----------------
U = eye(2);
X1 = (randn(n))/sqrt(2);
[Q1,R1] = qr(X1);
U = Q1;
-----------------end of Generating random unitary matrix U -----------------
n = length(U);
myDiag_U = eye(length(U));
myDiag_V = myDiag_U;

if n ==2
   [Q,P,e1,e2] = my2by2matrix(U,V);
   myDiag_U = [e2 0;0 e1];
   myDiag_V = eye(2);
   display('Q*U*P - myDiag_U =')
   disp(Q*U*P - myDiag_U)

elseif n >2
   [G_o,G_e,Q] = singledecom(U);
```

```
   P = eye(n);
   G_e = G_e';
   j = 1;

-----------------------check for deflation before iteration -----------------
   for i = 1:n-1
      if ((mod(i,2)==1)&&(abs(G_o(i+1,i))<10^(-15)))||
         ((mod(i,2)==0)&&(abs(G_e(i+1,i))<10^(-15)))



          if j == i
             myDiag_U(j,j) = G_o(j,j);
             myDiag_V(j,j) = G_e(j,j);

          else
             if mod(j,2)==1
                G_o1 = G_o(j:i,j:i);
                G_e1 = G_e(j:i.j:i);
             else
                G_o1 = G_e(j:i,j:i);
                 G_e1 = G_o(j:i,j:i);
             end
             [G_o2,G_e2,myQ,myP] = doubleloop(G_o1,G_e1);
             myDiag_U(j:i,j:i) = G_o2;
             myDiag_V(j:i,j:i) = G_e2;
             Q = blkdiag(eye(j-1),myQ,eye(n-i))*Q;
             P = P*blkdiag(eye(j-1),myP,eye(n-1));
```

```
          end
      j = i+1;

    elseif (i==n-1)&&(abs(G_o(n,n-1))>10^(-15)||abs(G_e(n,n-1))>10^(-15))
        [G_o2,G_e2,myQ,myP] = doubleloop(G_o,G_e);
        myDiag_U(j:n,j:n) = G_o2;
        myDiag_V(j:n,j:n) = G_e2;
        Q = myQ*Q;
      end
    end
  end

  display('eigenvalues for the matrix U are:')
  myDiag_U*myDiag_V
```

A.2.5 decomposition.m

```
function [G_o,G_e,Q,P] = decomposition(X,Y)
% X,Y are the unitary matrices in the original pencil to be reduced
% Output G_o,G_e,Q,P are as such G_oQXP=I,QYPG_e=I

U = X;
V = Y;                          % Both U and V are unitary matrices.

n = size(U);

Q = eye(n);                     % Left mult matrix
P = Q;                          % Right mult matrix
```

```
G_o = Q;                        % Left mult to U
G_e = Q;                        % Right mult to V
H = Q;                          % Store the householder reflector
G = Q;                          % Store the givens rotation matrix

for j = 1:floor(n/2)
   i = 2*(j-1)+1;
   if n-2*j>=1
      H = houseg(V(i:n,i),n);          % Find the Householder reflector of the
                                       % vector V(i:n,i).
      Q = H*Q;
      U = H*U;
      V = H*V;

--------------------This block is used to ensure the diagonal V(i,i) = 1-------
      H = blkdiag(eye(i-1),conj(V(i,i)),eye(n-i));
      Q = H*Q;
      U = H*U;
      V = H*V;
-------------------------------------------------------------------------------

      H = houseg(U(i+1:n,i),n);         % Left mult to U such that the elements
                                        % of U(i+2:n,i) becomes zero.
      Q = H*Q;
      U = H*U;
      V = H*V;
      G = givensg(U(i:i+1,i),n,i);      % Left mult to U such that the element of
```

```
                                        % U(i+1,i) becomes zero.
    G_o(i:i+1,i:i+1) = G(i:i+1,i:i+1);
    U = G*U;

    H = (houseg(U(i+1,i+1:n)',n))';      % Right mult to U such that the
                                         % elements of U(i+1, i+2:n)
                                         % becomes zero.
    P = P*H;
    U = U*H;
    V = V*H;

    H = blkdiag(eye(i), conj(U(i+1,i+1)),eye(n-i-1));
    P = P*H;
    U = U*H;
    V = V*H;

    H = (houseg(V(i+1,i+2:n)',n))';
    P = P*H;
    U = U*H;
    V = V*H;
    G = givensg(V(i+1,i+1:i+2)',n,i+1);
    G_e(i+1:i+2,i+1:i+2) = G(i+1:i+2,i+1:i+2)';
    V = V*G';


-------------------------------------------------------------------------

% Adjusting the value of the last diag element in U and V if size of U is an odd
```

```
% number to guarantee they are 1.


      if n-2*j == 1
         H = blkdiag(eye(n-1),conj(V(2*j+1,2*j+1)));
         Q = H*Q;
         V = H*V;
         U = H*U;
         m = conj(U(2*j+1,2*j+1));
         G_o(n,n) = m;
         H = blkdiag(eye(n-1),m);
         U = H*U;
      end


% Now we finish transforming the ith and i+1th rows and coloums of U and V into I.
-------------------------------------------------------------------------------
   elseif n-2*j == 0
      H = houseg(V(i:n,i),n);              % Find the Householder reflector of the
                                           % vector V(i, i:n).
      Q = H*Q;
      U = H*U;
      V = H*V;


      H = blkdiag(eye(i-1),conj(V(i,i)),1);
      Q = H*Q;
      U = H*U;
      V = H*V;
```

```
      G = givensg(U(i:i+1,i),n,i);
      G_o(i:i+1,i:i+1) = G(i:i+1,i:i+1);
      U = G*U;
      H = blkdiag(eye(n-1),conj(U(2*j,2*j)));      %turn U(n,n) into 1
      P = P*H;
      U = U*H;
      V = V*H;

      H = blkdiag(eye(n-1),conj(V(2*j,2*j)));
      V = V*H;
      G_e(2*j, 2*j) = H(2*j,2*j);
   end
end

  G_o = G_o';
  G_e = G_e';
end
```

A.2.6 singledecom.m

```
function [G_o,G_e,Q] = singledecom(X)
% X is the unitary matrix to be transformed
% Output G_o,G_e,Q are as such QXQ^*=G_oG_e

H = X;
n = length(H);
Q = eye(n);
G_o = Q;
```

```
G_e = Q;
K = Q;                          %K is used to store generated householder matrix.

for i=1:floor(n/2)
   j = 2*(i-1)+1;
   if n-j>2                             % If there are more than 3 rows left.
      K = houseg(H(j+1:n,j),n);         % Constructing G_o.
      H = K*H*K';
      Q = K*Q;
      K = givensg(H(j:j+1,j),n,j);
      H = K*H;
      G_o = K*G_o;
      K = houseg(H(j+1,j+2:n)',n)';     % Constructing G_e.
      H = K'*H*K;
      Q = K'*Q;
      K = givensg(H(j+1,j+1:j+2)',n,j+1)';
      H = H*K;
      G_e = G_e*K;

   elseif n-j== 2                       % If there are exactly 3 rows left.
      K = houseg(H(j+1:n,j),n);         % Constructing G_o.
      H = K*H*K';
      Q = K*Q;
      K = givensg(H(j:j+1,j),n,j);
      H = K*H;
      G_o = K*G_o;
```

```
      K = givensg(H(j+1,j+1:j+2)',n,j+1)';
      H = H*K;
      G_e = G_e*K;
      G_o = blkdiag(eye(n-1),sign(H(n,n))*1)*G_o;
      H = blkdiag(eye(n-1), sign(H(n,n))*1)*H;

   elseif n-j == 1                  % If there are only 2 rows left
      K = givensg(H(j:j+1,j),n,j);
      G_o = K*G_o;
      H = K*H;
      G_e = G_e*blkdiag(eye(n-1), conj(H(n,n)));
      H = H*blkdiag(eye(n-1), conj(H(n,n)));
   end
end

   G_o = G_o';
   G_e = G_e';
end
```

A.2.7 my2by2matrix.m

```
function [Q,P,e1,e2] = my2by2matrix(X,Y)
% Directly find the decomposition of 2 by 2 unitary matrices.
% Output Q,P,e1,e2 are as such QXP = diag[e1,e2], QYP=I.

   U = X;
   V = Y;
   W = V'*U;
```

```
   a = W(1,1);
   b = W(1,2);
   c = W(2,1);
   d = W(2,2);

   eigen1 = (a+d+sqrt((a-d)^2+4*b*c))/2;                % First eigenvalue of W
   eigen2 = (4*a*d-4*b*c)/(2*(a+d+sqrt((a-d)^2+4*b*c)));% Second eigenvalue of W
   A = W-eigen1*eye(2);
   if norm(A(:,1))>=norm(A(:,2))          % choose the column with bigger norm.
      v = A(:,1);
  else
      v = A(:,2);
  end

   P = givensg(v,2,1);
   Q = P*V';
   P = P';
   e1 = eigen1;
   e2 = eigen2;
end
```

A.2.8 myloop.m

```
function [G_o2,G_e2,Q,P] = myloop(X1,Y1)
% X1,Y1 are the G_o,G_e from phase I
% Output G_o2,G_e2,Q,P are as such QX1P=G_o2,QX2P=G_e2.
% G_o2,G_e2 are diagonal matrices
```

```
G_o = X1;
G_e = Y1;
n = length(G_o);
Q = eye(n);
P = Q;
G_o2 = Q;
G_e2 = Q;

i = n;

while i>2                          % If the dimension of the matrix is bigger than 2,
                                   % then we do chase.

   [subG_o1,subG_e1,subG_o2,subG_e2,myQ,myP] = chase(G_o,G_e);
                                              % One iteration step.
   Q = blkdiag(myQ,eye(n-i))*Q;
   P = P*blkdiag(myP, eye(n-i));
   k = length(subG_o2);

   if k == 2
      [Q1,P1,e1,e2] = my2by2matrix(subG_o2,subG_e2);
      Q = blkdiag(eye(i-2),Q1,eye(n-i))*Q;
      P = P*blkdiag(eye(i-2),P1,eye(n-i));
      G_o2(i-1,i-1) = e2;
      G_o2(i,i) = e1;

   elseif k == 1
```

```
      G_o2(i,i) = subG_o2(1,1);
      G_e2(i,i) = subG_e2(1,1);

   end
      G_o = subG_o1;
      G_e = subG_e1;
      i = i -k;
end

if i == 2                    % If after the chase, the dimension is 2, the we
                             % directly find the eigenvalues .
   [Q1,P1,e1,e2] = my2by2matrix(G_o,G_e);
   Q = blkdiag(Q1, eye(n-2))*Q;
   P = P*blkdiag(P1, eye(n-2));
   G_o2(1,1) = e2;
   G_o2(2,2) = e1;

elseif i ==1                 % If after the chase, the dimension is 1, then we
                             % directly assign the value to G_o2, G_e2
      G_o2(1,1) = G_o(1,1);
      G_e2(1,1) = G_e(1,1);
end
end
```

A.2.9 doubleloop.m

This function is similar to 'myloop.m', so we omit some explaination here.

```
function [G_o2,G_e2,Q,P] = doubleloop(X1,Y1)
```

```
G_o = X1;

G_e = Y1;

n = length(G_o);

Q = eye(n);

P = Q;

G_o2 = Q;

G_e2 = Q;


i = n;


while i>2

   [subG_o1,subG_e1,subG_o2,subG_e2,myQ,myP] = doublechase(G_o,G_e);

   Q = blkdiag(myQ,eye(n-i))*Q;

   P = P*blkdiag(myP, eye(n-i));

   k = length(subG_o2);


   if k == 2

      [Q1,P1,e1,e2] = my2by2matrix(subG_o2,subG_e2);

      Q = blkdiag(eye(i-2),Q1,eye(n-i))*Q;

      P = P*blkdiag(eye(i-2),P1,eye(n-i));

      G_o2(i-1,i-1) = e2;

      G_o2(i,i) = e1;


   elseif k == 1

      G_o2(i,i) = subG_o2(1,1);

      G_e2(i,i) = subG_e2(1,1);
```

```
   end
      G_o = subG_o1;
      G_e = subG_e1;
      i = i -k;
end

if i == 2
   [Q1,P1,e1,e2] = my2by2matrix(G_o,G_e);
   Q = blkdiag(Q1, eye(n-2))*Q;
   P = P*blkdiag(P1, eye(n-2));
   G_o2(1,1) = e2;
   G_o2(2,2) = e1;

elseif i ==1
      G_o2(1,1) = G_o(1,1);
      G_e2(1,1) = G_e(1,1);
end
end
```

A.2.10 chase.m

```
function [subG_o1,subG_e1,subG_o2,subG_e2,myQ,myP] = chase(X,Y)
% X is G_o
% Y is G_e
% m is the first row of the matrix located in the originl matrix

G_o = X;
G_e = Y;
```

```
G_o2 = X;
G_e2 = Y;
minRow = 0;                          % The smaller number between a and b.
n = length(G_o);
m = 0;

i = 0;
myflag = 0;
G_o1 = eye(n);
G_e1 = G_o1;
myQ = G_o1;
myP = G_o1;

while (myflag == 0)
   G = G_o*G_e';
   s = G(n,n);

   G1 = G_o-s*G_e;
   Q0 = houseg(G1(:,1),n);
   G_o = Q0*G_o;
   G_e = Q0*G_e;
   H = eye(n);

   Q = H;
   P = H;
   G_o1 = H;
   G_e1 = H;
```

```
for j = 1:floor(n/2)
   row = 2*(j-1)+1;
   if row+2<=n                  % If there are at least 3 rows left, then we
                                % can transform the first two diagonal
                                % elements into 1 through this block.

      if row == 1               % If it is the first transformation, then
                                % start with the right multi of G_e,
                                % otherwise, start with left multi of G_e.

         H = (blkdiag(houseg((G_e(1,1:3))',3),eye(n-3)))';
         G_e = G_e*H;
         G_o = G_o*H;
         P = P*H;

         H = blkdiag(conj(G_e(1,1)), eye(n-1));
         G_e = G_e*H;
         G_o = G_o*H;
         P = P*H;

      else
         H = blkdiag(houseg(G_e(row:row+2,row),row+2),eye(n-row-2));
         G_e = H*G_e;
         G_o = H*G_o;
         Q = H*Q;
```

```
  H = blkdiag(eye(row-1),conj(G_e(row,row)), eye(n-row));
  G_e = H*G_e;
  G_o = H*G_o;
  Q = H*Q;
end

minRow = min([n row+3]);
H = blkdiag(houseg(G_o(row+1:minRow,row),minRow),eye(n-minRow));
G_o = H*G_o;
G_e = H*G_e;
Q = H*Q;

H = givensg(G_o(row:row+1,row),n,row);
G_o1(row:row+1,row:row+1) = H(row:row+1,row:row+1);
G_o = H*G_o;

H = blkdiag(houseg((G_o(row+1,row+1:minRow))',minRow),eye(n-minRow))';
P = P*H;
G_o = G_o*H;
G_e = G_e*H;

H = blkdiag(eye(row),conj(G_o(row+1,row+1)), eye(n-row-1));
G_e = G_e*H;
G_o = G_o*H;
P = P*H;

if row+2<n
```

```
      minRow = min([n row+4]);
      H = (blkdiag(houseg((G_e(row+1,row+2:minRow))',minRow),
            eye(n-minRow)))';
      P = P*H;
      G_o = G_o*H;
      G_e = G_e*H;
   end

   H = (givensg((G_e(row+1,row+1:row+2))',n,row+1))';
   G_e1(row+1:row+2,row+1:row+2) = H(row+1:row+2,row+1:row+2);
   G_e = G_e*H;

   if row+2 == n
      H = blkdiag(eye(n-1),conj(G_e(n,n))) ;
      Q = H*Q;
      G_o = H*G_o;
      G_e = H*G_e;
      H = blkdiag(eye(n-1),conj(G_o(n,n)));
      G_o = H*G_o;
      G_o1(n,n) = H(n,n);
   end

elseif row+1 == n            % If there are only 2 rows and 2 columns left.
   H = houseg(G_e(row:row+1,row),n);
   G_e = H*G_e;
   G_o = H*G_o;
   Q = H*Q;
```

```
      H = blkdiag(eye(row-1),conj(G_e(row,row)), 1);
      G_e = H*G_e;
      G_o = H*G_o;
      Q = H*Q;

      H = givensg(G_o(row:row+1,row),n,row);
      G_o1(row:row+1,row:row+1) = H(row:row+1,row:row+1);
      G_o = H*G_o;

      H = blkdiag(eye(n-1),conj(G_o(n,n)));
      G_o = G_o*H;
      G_e = G_e*H;
      P = P*H;

      H = blkdiag(eye(n-1),conj(G_e(n,n)));
      G_e = G_e*H;
      G_e1(n,n) = H(n,n);
   end

end

G_o = G_o1';
G_e = G_e1';
myQ = Q*Q0*myQ;
myP = myP*P;
i = i+1;
```

```
   k = 0;

   while (k<n-1)                                % check for deflation
      k = k+1;
      if (mod(k,2)==1)&&(abs(G_o(k+1,k))<10^(-16))
         m = k;
         myflag = 1;
         k = n-1;
      elseif (mod(k,2)==0)&&(abs(G_e(k+1,k))<10^(-16))
         m = k;
         myflag = 1;
         k = n-1;
      end
   end
 end

subG_o1 = G_o(1:m,1:m);
subG_o2 = G_o(m+1:n,m+1:n);
subG_e1 = G_e(1:m,1:m);
subG_e2 = G_e(m+1:n,m+1:n);

end
```

A.2.11 doublechase.m

This function is similar to 'chase.m', so we omit some explaination here.

```
function [subG_o1,subG_e1,subG_o2,subG_e2,myQ,myP] = doublechase(X,Y)
```

```
G_o = X;
G_e = Y;
G_o2 = X;
G_e2 = Y;
minRow = 0;
n = length(G_o);
m = 0;

i = 0;
myflag = 0;
G_o1 = eye(n);
G_e1 = G_o1;
myQ = G_o1;
myP = G_o1;

while (myflag == 0)
   G = G_o*G_e';

   a = G(n-1,n-1);
   b = G(n-1,n);
   c = G(n,n-1);
   d = G(n,n);

   G1 = G_o*G_e'+(a*d-b*c)*G_e*G_o'-(a+d)*eye(n);
   Q0 = houseg(G1(:,1),n);
   G_o = Q0*G_o;
   G_e = Q0*G_e;
```

```
H = eye(n);


Q = H;
P = H;
G_o1 = H;
G_e1 = H;


for j = 1:floor(n/2)
   row = 2*(j-1)+1;
   if row+2<=n
      if row == 1
         H = (blkdiag(houseg((G_e(1,1:3))',3),eye(n-3)))';
         G_e = G_e*H;
         G_o = G_o*H;
         P = P*H;


         H = blkdiag(conj(G_e(1,1)), eye(n-1));
         G_e = G_e*H;
         G_o = G_o*H;
         P = P*H;


      else
         H = blkdiag(houseg(G_e(row:row+2,row),row+2),eye(n-row-2));
         G_e = H*G_e;
         G_o = H*G_o;
         Q = H*Q;
```

```
  H = blkdiag(eye(row-1),conj(G_e(row,row)), eye(n-row));
  G_e = H*G_e;
  G_o = H*G_o;
  Q = H*Q;
end

minRow = min([n row+3]);
H = blkdiag(houseg(G_o(row+1:minRow,row),minRow),eye(n-minRow));
G_o = H*G_o;
G_e = H*G_e;
Q = H*Q;

H = givensg(G_o(row:row+1,row),n,row);
G_o1(row:row+1,row:row+1) = H(row:row+1,row:row+1);
G_o = H*G_o;

H = blkdiag(houseg((G_o(row+1,row+1:minRow))',minRow),eye(n-minRow))';
P = P*H;
G_o = G_o*H;
G_e = G_e*H;

H = blkdiag(eye(row),conj(G_o(row+1,row+1)), eye(n-row-1));
G_e = G_e*H;
G_o = G_o*H;
P = P*H;

if row+2<n
```

```
        minRow = min([n row+4]);
        H = (blkdiag(houseg((G_e(row+1,row+2:minRow))',minRow),
              eye(n-minRow)))';
        P = P*H;
        G_o = G_o*H;
        G_e = G_e*H;
      end

      H = (givensg((G_e(row+1,row+1:row+2))',n,row+1))';
      G_e1(row+1:row+2,row+1:row+2) = H(row+1:row+2,row+1:row+2);
      G_e = G_e*H;

      if row+2 == n
        H = blkdiag(eye(n-1),conj(G_e(n,n))) ;
        Q = H*Q;
        G_o = H*G_o;
        G_e = H*G_e;
        H = blkdiag(eye(n-1),conj(G_o(n,n)));
        G_o = H*G_o;
        G_o1(n,n) = H(n,n);
      end

elseif row+1 == n
   H = houseg(G_e(row:row+1,row),n);
   G_e = H*G_e;
   G_o = H*G_o;
   Q = H*Q;
```

```
      H = blkdiag(eye(row-1),conj(G_e(row,row)), 1);
      G_e = H*G_e;
      G_o = H*G_o;
      Q = H*Q;

      H = givensg(G_o(row:row+1,row),n,row);
      G_o1(row:row+1,row:row+1) = H(row:row+1,row:row+1);
      G_o = H*G_o;

      H = blkdiag(eye(n-1),conj(G_o(n,n)));
      G_o = G_o*H;
      G_e = G_e*H;
      P = P*H;

      H = blkdiag(eye(n-1),conj(G_e(n,n)));
      G_e = G_e*H;
      G_e1(n,n) = H(n,n);
   end

end

G_o = G_o1';
G_e = G_e1';
myQ = Q*Q0*myQ;
myP = myP*P;
i = i+1;
```

```
   k = 0;

   while (k<n-1)
      k = k+1;
      if (mod(k,2)==1)&&(abs(G_o(k+1,k))<10^(-16))
         m = k;
         myflag = 1;
         k = n-1;
      elseif (mod(k,2)==0)&&(abs(G_e(k+1,k))<10^(-16))
         m = k;
         myflag = 1;
         k = n-1;
      end
   end
 end

subG_o1 = G_o(1:m,1:m);
subG_o2 = G_o(m+1:n,m+1:n);
subG_e1 = G_e(1:m,1:m);
subG_e2 = G_e(m+1:n,m+1:n);

end
```

A.2.12 givens.m

```
function givens = givensg(x,n,m)
% Function to generate a Givens rotation matrix
% x is the vector to be transformed, n is the dimension of the orginal
```

```
% matrix, m is the starting position of the x located in the original
% matrix

G = zeros(2);                       % Initialize Givens matrix

if x(2,1)== 0
   givens = eye(n);                 % If x is a multiple of e1, then givens = I.

else
   if norm(x(1,1))>=norm(x(2,1))
      t = norm(x(2,1))/norm(x(1,1));
      r = sqrt(1+(t)^2);
      c = conj(x(1,1))/(norm(x(1,1))*r);
      s =(x(1,1)/(norm(x(1,1)))^2)*c*conj(x(2,1));

   else
      t = norm(x(1,1))/norm(x(2,1));
      r = sqrt(1+(t)^2);
      s =conj(x(2,1))/(norm(x(2,1))*r);
      c = (x(2,1)/(norm(x(2,1)))^2)*s*conj(x(1,1));
   end

   G(1,1)= c;
   G(2,2) = conj(c);
   G(1,2) = s;
   G(2,1) = -conj(s);
   givens = blkdiag(eye(m-1),G,eye(n-m-1));
```

```
end
end
```

A.2.13 houseg.m

```
function householder = houseg(x,n)
% function to generate a Householder reflector such that householder*v =
% norm(v)e1
% x is the vector to be transformed, n is the dimension of the original
% matrix

i = length(x);
j = n - i;                  % Determine the dimension of the identity matrix
                            % such that subI direct sum with the Householder
                            % reflector is what we want
I = eye(i);
subI = eye(j);

if norm(x)<= 10^(-15)
   householder = eye(n);                % If x is 0, then householder = I

else
   if abs(x(1,1))<=10^(-15)
      a = norm(x);
   else
      a = x(1,1)/abs(x(1,1))*norm(x);
   end
```

```
    b = a*I(:,1);
    v = b+x;
    beta = 2/(v'*v);
    housereflector = I - beta*(v*v');
    householder =blkdiag(subI, housereflector);
end
end
```