Methods for eigenvalue problems with applications in model order reduction

Methoden voor eigenwaardeproblemen
met toepassingen in model orde reductie

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht
op gezag van de rector magnificus, prof.dr. W. H. Gispen,
ingevolge het besluit van het college voor promoties in het openbaar
te verdedigen op maandag 25 juni 2007 des middags te 4.15 uur

door

Joost Rommes

geboren op 29 mei 1979 te Utrecht

Promotor: Prof.dr. H. A. van der Vorst

Dit proefschrift werd mede mogelijk gemaakt door financiële steun van BSIK/BRICKS project MSV1.

Methods for eigenvalue problems with applications in model order reduction

2000 Mathematics Subject Classification: 65F15, 65F50.

Rommes, Joost
Methods for eigenvalue problems with applications in model order reduction
Proefschrift Universiteit Utrecht – Met een samenvatting in het Nederlands.

ISBN 978-90-393-4569-6

Contents

List of Algorithms

Chapter 1

Introduction

1.1 Background

Physical structures and processes are modeled by dynamical systems in a wide range of application areas. In structural analysis, for instance, one is interested in the natural frequency of a structure, that is, the frequency at which the system prefers to vibrate. The natural frequencies are of importance for the design and construction of bridges and large buildings, because precautions must be taken to prevent the structure from resonating with vibrations due to external forces such as pedestrian movements, wind, and earthquakes. Some electrical circuits, on the other hand, are designed to amplify or filter signals at specific frequencies. In chemical engineering, dynamical systems are used to describe heat transfer and convection of chemical processes and reactions. It is often cheaper and safer to simulate the system before actually realizing it, because simulation may give insight in the behavior of the system and may help to check if the design meets the requirements, and to adapt the design if needed.

Of paramount importance in all applications is that the models accurately enough describe the underlying systems. The increasing demand for complex components such as high-speed computer chips with more transistors and decreasing size, and structures such as large airplanes, together with an increasing demand for detail and accuracy, makes the models larger and more complicated. Although computer resources are also getting faster and bigger, direct simulation of the system is often not feasible because the required storage and computer time is proportional to the square and cube, respectively, of the number of elements of the model, which may easily exceed one million. To be able to simulate these large-scale systems, there is need for reduced-order models of much smaller size, that approximate the behavior of the original model and preserve the important characteristics, at least for the frequency or time range of interest.

One can see model order reduction as an application of Ockham's razor. William of Ockham, a medieval logician, stated that in the explanation of a phenomenon,

as few assumptions as possible should be made, and that the simplest explanation is to be preferred. In model order reduction, Ockham's razor is used to "shave off" the less important characteristics of a model. The same goal can be achieved by computing the important characteristics instead.

This thesis presents algorithms for the computation of dominant eigenvalues and corresponding eigenvectors of eigenproblems that are related to large-scale dynamical systems. Here, the interpretation of the adjective *dominant* depends on the application from which the eigenproblem arises. In stability analysis of steady state solutions of discretized Navier-Stokes equations, for example, the dominant eigenvalues are the rightmost eigenvalues, since eigenvalues with positive real part correspond to unstable steady state solutions. In electrical circuit simulation and other applications, one is also interested in the rightmost eigenvalues, that correspond to unstable or slowly damped modes. In structural analysis and engineering, the dominant eigenvalues are the natural frequencies. For general linear time invariant dynamical systems, the dominant eigenvalues are the poles of the transfer function that contribute significantly to the frequency response. In the design of stabilizers and controllers for large-scale systems, the dominant eigenvalues are the zeros of the transfer function that damp unwanted modes.

The dominant eigenvalues and corresponding eigenvectors give information about the behavior of the system or solution, and hence function as a reduced-order model in some sense. The dominant eigenvalues and corresponding eigenvectors, for instance, can be used to construct a reduced-order model for the original system. A large-scale dynamical system can be reduced to a much smaller reduced-order model, which is also a dynamical system, by projecting the state-space on the dominant eigenspace. In stability analysis, insight in the oscillatory or unstable behavior of steady state solutions is obtained by studying the modes corresponding to the rightmost eigenvalues. The construction of the reduced-order model varies with the interpretation of the dominant eigenvalues and the application, but in all cases the quality of the reduced-order model is determined by the degree in which it reflects the characteristic (dominant) behavior of the underlying system or solution. The algorithms presented in this thesis are specialized eigenvalue methods that compute dominant eigenvalues and corresponding eigenvectors for several types of applications.

The remainder of this chapter is organized as follows. Section 1.2 gives a brief introduction to eigenvalue problems. In Section 1.3, essential concepts and results from system and control theory are summarized. Section 1.5 describes methods for eigenvalue problems and the related model order reduction methods. An overview of the contributions of this thesis is given in Section 1.6.

1.2 Eigenvalue problems

This section briefly describes the various types of eigenvalue problems (eigenproblems) that are related to topics discussed in this thesis. For more details the reader is referred to, for instance, [8, 64].

1.2.1 The standard eigenvalue problem

The standard eigenvalue problem is to find $\lambda \in \mathbb{C}$ and $\mathbf{x} \in \mathbb{C}^n$ that satisfy

$$A\mathbf{x} = \lambda\mathbf{x}, \quad \mathbf{x} \neq 0,$$

where A is a complex $n \times n$ matrix. The scalar $\lambda \in \mathbb{C}$, that satisfies $\det(A-\lambda I) = 0$, is an eigenvalue, and the nonzero vector $\mathbf{x} \in \mathbb{C}^n$ is a (right) eigenvector for λ. The pair $(\lambda, \mathbf{x})$ is also referred to as an eigenpair of A. A nonzero vector $\mathbf{y} \in \mathbb{C}^n$ that satisfies $\mathbf{y}^*A = \lambda\mathbf{y}^*$ is a left eigenvector for λ, and $(\lambda, \mathbf{x}, \mathbf{y})$ is called an eigentriplet of A. The set of all eigenvalues of A is called the spectrum of A, denoted by $\Lambda(A)$. Symmetric matrices $A = A^T \in \mathbb{R}^{n\times n}$ have real eigenvalues and an orthonormal basis of real eigenvectors. Hermitian matrices $A = A^* \in \mathbb{C}^{n\times n}$ have real eigenvalues and an orthonormal basis of eigenvectors. If $A \in \mathbb{C}^{n\times n}$ is normal ($AA^* = A^*A$), then there also exists an orthonormal basis of eigenvectors. For symmetric, Hermitian and normal matrices, the right eigenvector for an eigenvalue is also a left eigenvector for the same eigenvalue. For a general matrix $A \in \mathbb{C}^{n\times n}$ there exists a Schur decomposition

$$Q^*AQ = T,$$

where $Q \in \mathbb{C}^{n\times n}$ is unitary ($Q^*Q = I$) and $T \in \mathbb{C}^{n\times n}$ is upper triangular with the eigenvalues of A on its diagonal. A general matrix A is diagonalizable (or nondefective) if and only if there exists a nonsingular $X \in \mathbb{C}^{n\times n}$ such that $X^{-1}AX = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, where λ_i $(i = 1, \ldots, n)$ are the eigenvalues of A and the columns of X are eigenvectors of A. If all λ_i are distinct, then there are n independent eigenvectors. A matrix that is not diagonalizable is called defective, and an eigenvalue λ with algebraic multiplicity (multiplicity of root λ of $\det(A - \lambda I)$) greater than its geometric multiplicity (dimension of the corresponding eigenspace) is called defective. For a general matrix A, there exists a Jordan decomposition $X^{-1}AX = \mathrm{diag}(J_1, \ldots, J_s)$, where the Jordan blocks J_i are $m_i \times m_i$ upper triangular matrices with its single eigenvalue λ_i on the diagonal and ones on the first superdiagonal, and all other elements zero. Each block J_i has a single independent left and right eigenvector, and $m_1 + \ldots + m_s = n$, and each block with $m_i > 1$ corresponds to a defective eigenvalue.

1.2.2 The generalized eigenvalue problem

The generalized eigenvalue problem is of the form

$$A\mathbf{x} = \lambda B\mathbf{x}, \quad \mathbf{x} \neq 0,$$

where A and B are $n \times n$ complex matrices, and reduces to the standard eigenvalue problem if $B = I$. An eigentriplet $(\lambda, \mathbf{x}, \mathbf{y})$ of the pair (A, B) (or pencil $A - \lambda B$) consists of an eigenvalue $\lambda \in \mathbb{C}$ that satisfies $\det(A - \lambda B) = 0$, a nonzero (right) eigenvector $\mathbf{x} \in \mathbb{C}^n$ that satisfies $A\mathbf{x} = \lambda B\mathbf{x}$, and a nonzero left eigenvector $\mathbf{y} \in \mathbb{C}^n$ that satisfies $\mathbf{y}^*A = \lambda\mathbf{y}^*B$. The set of all eigenvalues of (A, B) is called the

spectrum of (A, B), denoted by $\Lambda(A, B)$. If A and B are Hermitian and B is positive definite, then all eigenvalues of (A, B) are real and there exists a nonsingular $X \in \mathbb{C}^{n\times n}$ such that $X^*AX = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ and $X^*BX = I$, where λ_i $(i = 1, \ldots, n)$ are the eigenvalues of (A, B). The columns $\mathbf{x}_i$ of X are B-orthogonal ($\mathbf{x}_i^* B\mathbf{x}_j = 0$ if $i \neq j$), and are both right and left eigenvectors for the same eigenvalue λ_i. If in addition A and B are real, the eigenvectors can be chosen to be real as well. For a general matrix pair (A, B) there exists a generalized Schur decomposition

$$Q^*AZ = T, \quad Q^*BZ = S,$$

where $Q, Z \in \mathbb{C}^{n\times n}$ are unitary ($Q^*Q = Z^*Z = I$) and $S, T \in \mathbb{C}^{n\times n}$ are upper triangular with $t_{ii}/s_{ii} = \lambda_i$ for $s_{ii} \neq 0$. If $s_{ii} = 0$, then $\lambda_i = \infty$, and if both t_{ii} and s_{ii} are zero, then the spectrum $\Lambda(A, B) = \mathbb{C}$. A general matrix pair (A, B) is called diagonalizable (or nondefective) if there are n independent right eigenvectors $\mathbf{x}_i$ and n independent left eigenvectors $\mathbf{y}_i$: if $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ and $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_n]$, then it follows from $\mathbf{y}_i^* B\mathbf{x}_j = 0$ for $i \neq j$ that $Y^*AX = \Lambda_A$ and $Y^*BX = \Lambda_B$, with $\Lambda_A\Lambda_B^{-1} = \Lambda$ (provided Λ_B is nonsingular). If all λ_i are distinct, then there are n independent eigenvectors. Analogous to the Jordan decomposition for the standard eigenvalue problem, there are Weierstrass and Weierstrass-Schur decompositions [8, p. 30] for the generalized eigenvalue problem.

If A (B) is nonsingular, the generalized eigenproblem can be transformed to a standard eigenproblem by multiplying with A^{-1} (B^{-1}), but practically speaking this is often necessary nor advisable.

1.2.3 The polynomial eigenvalue problem

The polynomial eigenvalue problem is of the form

$$(\lambda^p A_p + \lambda^{p-1} A_{p-1} + \cdots \lambda A_1 + A_0)\mathbf{x}, \quad \mathbf{x} \neq 0,$$

where the A_i are $n \times n$ complex matrices, and is a generalization of the standard and generalized eigenvalue problem. Definitions of eigenvalues and left and right eigenvectors follow from the definitions for the generalized eigenvalue problem. Although there are some similarities between polynomial and standard/generalized eigenproblems, the major difference is that the polynomial eigenproblem has np eigenvalues with up to np left and np right eigenvectors, that, if there are more than n eigenvectors, are not independent. If $p = 2$, the polynomial eigenvalue problem is a quadratic eigenvalue problem. See [152] and references therein for more details on quadratic and polynomial eigenproblems.

1.2.4 The singular value problem

For an $m\times n$ matrix A, the singular value problem consists of finding $\sigma \in \mathbb{R}$ $(\sigma \geq 0)$, nonzero $\mathbf{u} \in \mathbb{C}^m$ and nonzero $\mathbf{v} \in \mathbb{C}^n$ with $\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1$ that satisfy

$$\begin{aligned} A\mathbf{v} &= \sigma\mathbf{u}, \\ A^*\mathbf{u} &= \sigma\mathbf{v}. \end{aligned}$$

The singular value σ and corresponding left singular vector $\mathbf{u}$ and right singular vector $\mathbf{v}$ form a singular triplet $(\sigma, \mathbf{u}, \mathbf{v})$. The nonzero singular values are the square roots of the nonzero eigenvalues of AA^* or A^*A, and the left (right) singular vectors of A are the eigenvectors of AA^* (A^*A). Alternatively, the absolute values of the eigenvalues of

$$\begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix}$$

are the singular values of A, and the left (right) singular vectors can be extracted from the first (second) part of the corresponding eigenvectors. The decomposition $A = U\Sigma V^*$, with U and V unitary, is called a singular value decomposition (SVD) of A if the columns of $U \in \mathbb{C}^{m\times m}$ are the left singular vectors, the columns of $V \in \mathbb{C}^{n\times n}$ are the right singular vectors, and $\Sigma \in \mathbb{C}^{m\times n}$ is a diagonal matrix with the singular values on its diagonal.

1.3 System and control theory

This section describes the concepts from system and control theory that are needed in this thesis. Most of the material is adapted from [27, 78, 118, 145], that give detailed introductions to system theory, and [5], that provides a good introduction to system theory from the view point of model order reduction and numerical linear algebra.

1.3.1 State space systems

The internal description of a linear time invariant (LTI) system $\Sigma = (A, B, C, D)$ is

$$\left\{ \begin{array}{rcl} \dot{\mathbf{x}}(t) & = & A\mathbf{x}(t) + B\mathbf{u}(t) \\ \mathbf{y}(t) & = & C^*\mathbf{x}(t) + D\mathbf{u}(t), \end{array} \right. \tag{1.3.1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{n\times p}$, $\mathbf{x}(t) \in \mathbb{R}^n$, $\mathbf{u}(t) \in \mathbb{R}^m$, $\mathbf{y}(t) \in \mathbb{R}^p$, and $D \in \mathbb{R}^{p\times m}$. The matrix A is called the state-space matrix, the matrices B and C are called the input and output map, respectively, and D is the direct transmission map; they are also referred to as system matrices. The vector $\mathbf{u}(t)$ is called the input or control, $\mathbf{x}(t)$ is called the state vector, and $\mathbf{y}(t)$ is called the output of the system. The order of the system is n. If $m, p > 1$, the system is called multi-input multi-output (MIMO). If $m = p = 1$ the system is called single-input single-output (SISO) and reduces to $\Sigma = (A, \mathbf{b}, \mathbf{c}, d)$:

$$\left\{ \begin{array}{rcl} \dot{\mathbf{x}}(t) & = & A\mathbf{x}(t) + \mathbf{b}u(t) \\ y(t) & = & \mathbf{c}^*\mathbf{x}(t) + du(t), \end{array} \right. \tag{1.3.2}$$

where $A \in \mathbb{R}^{n\times n}$, $\mathbf{b}, \mathbf{c}, \mathbf{x}(t) \in \mathbb{R}^n$, and $u(t), y(t), d \in \mathbb{R}$. In a more general setting the matrices A, B, C, D may also be time dependent, and the relations between input, state and output may be nonlinear. The dynamical systems (1.3.1) and (1.3.2) can come directly from linear models, or can be linearizations of nonlinear models.

The transfer function $H : \mathbb{C}^m \longrightarrow \mathbb{C}^p$ of (1.3.1),

$$H(s) = C^*(sI - A)^{-1}B + D, \tag{1.3.3}$$

can be obtained by applying the Laplace transform to (1.3.1) under the condition $\mathbf{x}(0) = 0$. The transfer function relates outputs to inputs in the frequency domain via $Y(s) = H(s)U(s)$, where $Y(s)$ and $U(s)$ are the Laplace transforms of $\mathbf{y}(t)$ and $\mathbf{u}(t)$, respectively, and is important in many engineering applications. It is well known that the transfer function is invariant under state-space transformations $\mathbf{x} \mapsto T\mathbf{x}$ where $T \in \mathbb{R}^{n\times n}$ is nonsingular. Consequently, there exist infinitely many realizations $(TAT^{-1}, TB, T^{-*}C, D)$ of the same LTI transfer function. There exist also realizations of order $\tilde{n}$ with $\tilde{n} > n$ (that are in general not of interest). The lower limit $\hat{n} < n$ on the order of the system is called the McMillan degree of the system and a realization of order $\hat{n}$ is called a minimal realization.

A pole of transfer function $H(s)$ is a $p \in \mathbb{C}$ for which $\lim_{s\to p} \|H(s)\|_2 = \infty$. The set of poles of $H(s)$ is a subset of the eigenvalues $\lambda_i \in \mathbb{C}$ of the state-space matrix A. A (transmission) zero of transfer function $H(s)$ is a $z \in \mathbb{C}$ for which $H(s)$ drops in rank if $s = z$; in the SISO case this means that $H(z) = 0$ for a zero $z \in \mathbb{C}$.

Let $(\lambda_i, \mathbf{x}_i, \mathbf{y}_i)$ $(i = 1, \ldots, n)$ be eigentriplets of A, with all λ_i distinct and the eigenvectors scaled so that $\mathbf{y}_i^*\mathbf{x}_i = 1$ and $\mathbf{y}_i^*\mathbf{x}_j = 0$ for $i \neq j$. The transfer function $H(s)$ can be expressed as a sum of residue matrices $R_j \in \mathbb{C}^{p\times m}$ over the poles [78]:

$$H(s) = D + \sum_{j=1}^{n} \frac{R_j}{s - \lambda_j}, \tag{1.3.4}$$

where

$$R_j = (C^*\mathbf{x}_j)(\mathbf{y}_j^*B).$$

If there are nondefective multiple eigenvalues, then the eigenvectors can be identified by their components in B and C. In the SISO case, for instance, decompose $\mathbf{b}$ as

$$\mathbf{b} = \sum_{i=1}^{k} \sum_{j=1}^{m_i} \beta_i^j \mathbf{x}_i^j,$$

where k is the number of distinct eigenvalues, m_i is the multiplicity of λ_i, β_i^j are coefficients, and $\mathbf{x}_i^j$ are eigenvectors. The right eigenvector $\mathbf{x}_i$ of a pole λ_i of multiplicity m_i is then identified by $\mathbf{x}_i = \sum_{i=1}^{m_i} \beta_i^j \mathbf{x}_i^j$. Note that the summation in (1.3.4) then consists of $k \leq n$ terms. Poles λ_j with large $\|R_j\|_2/|\mathrm{Re}(\lambda_j)|$ (or $\|R_j\|_2$) are called dominant poles and are studied in Chapters 2–6.

A system (A, B, C, D) is called (asymptotically) stable if all eigenvalues of A have strictly negative real parts (A is also called Hurwitz then). This can also be seen in the solution

$$\mathbf{y}(t) = C^*\mathbf{x}(t) + D\mathbf{u}(t), \quad \mathbf{x}(t) = e^{A(t-t_0)}\mathbf{x}_0 + \int_{t_0}^{t} e^{A(t-\tau)}B\mathbf{u}(\tau)d\tau \tag{1.3.5}$$

of system (1.3.1) with initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$: $\mathbf{y}(t)$ becomes unbounded if A has eigenvalues with positive real part. A square $(m = p)$ system is passive if and

only if its transfer function $H(s)$ is positive real, that is, $H(s)$ is analytic for all s with $\mathrm{Re}(s) > 0$, $H(\bar{s}) = H^*(s)$ for all $s \in \mathbb{C}$, and $H(s) + H^*(s) = \mathrm{Re}(H(s)) \geq 0$ for all s with $\mathrm{Re}(s) > 0$ (in the MIMO case, $>$ ($\geq$) denotes positive (semi-)definite). Hence, a real stable system is passive if and only if $\mathrm{Re}(H(s)) \geq 0$ for all s with $\mathrm{Re}(s) > 0$. Passivity [54] means that the system does not generate energy and only absorbs energy from the sources used to excite it.

A system is called causal if for any t, the output $\mathbf{y}(t)$ at time t depends only on inputs $\mathbf{u}(\tilde{t})$ with $\tilde{t} \leq t$. A system is called controllable if for any t_0, starting from zero initial state $\mathbf{x}(t_0) = 0$, every state $\mathbf{x}(t)$ can be reached via a suitable input $\mathbf{u}(t)$. It follows from (1.3.5) that controllability means that the range of $e^{At}B$ is $\mathbb{R}^n$, and by expansion of e^{At} and the Cayley-Hamilton theorem[1] a system (A, B, C, D) is controllable if and only if the controllability matrix

$$C(A, B) = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix}$$

has full rank n. Dually, observability means that the initial state can be uniquely determined from the input and the output, or equivalently, that with $\mathbf{u}(t) = 0$, a zero output $\mathbf{y}(t) = 0$ implies that the initial state is zero. Hence, a system is called observable if and only if the observability matrix

$$O(A, C) = \begin{bmatrix} C & A^*C & \dots & (A^{n-1})^*C \end{bmatrix}^*$$

has full rank n. A system is called complete if it is both controllable and observable, and a realization is minimal if and only if it is both controllable and observable. The set of poles of a minimal realization coincides with the set of eigenvalues of A.

If $\mathbf{u}(t) = 0$, the output (1.3.5) is called the zero-input or transient response $\mathbf{y}_h(t)$, which depends on the initial conditions only. If $\mathbf{x}(t_0) = 0$ the output is called the zero-state response. If $\mathbf{u}(t) = \delta(t)$ (the unit Kronecker delta distribution) and $\mathbf{x}(t_0) = 0$, $\mathbf{h}(t) = C^*e^{At}B + D\delta(t)$ is called the impulse response. The steady state response is the output for $t \to \infty$. If the system is stable it follows that $\lim_{t\to\infty} \|\mathbf{y}_h(t)\| = 0$ and hence the steady state response is $\mathbf{y}_p(t)$, where $\mathbf{y}_p(t)$ is the unique (particular) solution (1.3.5). If $u(t) = 1(t)$, the unit step function ($u(t) = 1$ for $t \geq 0$ and $u(t) = 0$ for $t < 0$), the output is called the step response, and if $u(t) = t1(t)$, the output is called the ramp response.

For stable systems, the controllability gramian $P \in \mathbb{R}^{n\times n}$ for a linear time-invariant system is defined as

$$P = \int_0^\infty e^{A\tau}BB^*e^{A^*\tau}d\tau,$$

and can be interpreted in the following way: the minimum energy $J(\mathbf{u}) = \int_{-\infty}^0 \mathbf{u}^*(t)\mathbf{u}(t)dt$ of the input to arrive at $\mathbf{x}(0) = \mathbf{x}_0$ is $J(\mathbf{u}) = \mathbf{x}_0^*P^{-1}\mathbf{x}_0$. Hence, states in the span of the eigenvectors corresponding to small eigenvalues of P are difficult to reach. Dually, the observability gramian $Q \in \mathbb{R}^{n\times n}$ is defined as

$$Q = \int_0^\infty e^{A^*\tau}CC^*e^{A\tau}d\tau.$$

[1] The Cayley-Hamilton theorem states that a matrix A satisfies its characteristic equation $p(\lambda) = \det(A - \lambda I) = 0$.

A system released from $\mathbf{x}(0) = \mathbf{x}_0$ with $\mathbf{u}(t) = 0, t \geq 0$ has

$$\int_0^\infty \mathbf{y}^*(t)\mathbf{y}(t)dt = \mathbf{x}_0^* Q \mathbf{x}_0,$$

and it follows that states in the span of the eigenvectors corresponding to small eigenvalues of Q are difficult to observe. A stable system is controllable if and only if P is positive definite, and it is observable if and only if Q is positive definite. Substitution shows that for stable systems, the gramians are solutions of the Lyapunov equations

$$AP + PA^* + BB^* = 0, \quad \text{and} \quad A^*Q + QA + CC^* = 0.$$

Although the gramians are not similar under state-space transformation $\mathbf{x} \mapsto T\mathbf{x}$ with $T \in \mathbb{R}^{n\times n}$ nonsingular, their product PQ is.

The Hankel operator $\mathcal{H}$ maps past inputs $\mathbf{u}(t)$ $(t < 0)$ to future outputs $\mathbf{y}(t)$ $(t > 0)$:

$$\mathcal{H} : L_2^m((-\infty, 0)) \longrightarrow L_2^p((0, \infty)) : u(t) \mapsto \int_{-\infty}^0 Ce^{A(t-\tau)}Bu(\tau)d\tau,$$

where $L_p^n(I)$ denotes the Lebesque space $L_p^n(I) = \{\mathbf{x}(t) : I \to \mathbb{R}^n | \|\mathbf{x}(t)\|_p < \infty\}$, and is of importance in control theory and model order reduction. The singular values of the Hankel operator are the square roots of the eigenvalues of $\mathcal{H}^*\mathcal{H}$, and are called the Hankel singular values. It can be shown that for stable complete systems, the Hankel singular values $\sigma_i^{\mathcal{H}}$ can be computed as the positive square roots of the eigenvalues of the product of the controllability gramian P and observability gramian Q:

$$\sigma_i^{\mathcal{H}} \equiv \sigma_i(\mathcal{H}) = \sqrt{\lambda_i(PQ)}, \quad i = 1, 2, \ldots, n.$$

The fact that the Hankel operator has a finite number n of singular values partly explains its importance and usefulness in control theory and model order reduction, since this allows, amongst others, for the definition of the Hankel norm of a system (A, B, C, D) with transfer function $H(s)$:

$$\|H\|_{\mathcal{H}} \equiv \sup_{\mathbf{u}\in L_2(-\infty,0)} \frac{\|\mathcal{H}\mathbf{u}\|_2}{\|\mathbf{u}\|_2} = \sigma_{\max}(\mathcal{H}) = \lambda_{\max}^{1/2}(PQ).$$

The Hankel singular values for a system and its transfer function are defined to be the same. The Hankel singular values are invariant under state-space transformations, since similarity of PQ is preserved under state-space transformations, and are so called input-output invariants. The Hankel singular values and Hankel norm are of great importance for certain types of model order reduction, as is described in Section 1.4. See [62] for more details on the Hankel operator and Hankel singular values.

1.3.2 Descriptor systems

Linear time invariant descriptor systems (E, A, B, C, D) are of the form

$$\begin{cases} E\dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t) \\ \mathbf{y}(t) &= C^*\mathbf{x}(t) + D\mathbf{u}(t), \end{cases} \tag{1.3.6}$$

where $E, A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{n\times p}$, $\mathbf{x}(t) \in \mathbb{R}^n$, $\mathbf{u}(t) \in \mathbb{R}^m$, $\mathbf{y}(t) \in \mathbb{R}^p$, and $D \in \mathbb{R}^{p\times m}$. Terminology is similar as for state-space systems, but here the descriptor matrix E may be singular. If E is nonsingular, the descriptor can be transformed to the state-space system $(E^{-1}A, E^{-1}B, C, D)$, although this usually is not attractive from a computational viewpoint. The transfer function $H(s)$ of (1.3.6) is defined as

$$H(s) = C^*(sE - A)^{-1}B + D. \tag{1.3.7}$$

The transfer function $H(s)$ can be expressed as a sum of residue matrices $R_j \in \mathbb{C}^{p\times m}$ over the finite first order poles (if (A, E) is nondefective) [78], cf. (1.3.4):

$$H(s) = D + R_\infty + \sum_{j=1}^{\tilde{n}} \frac{R_j}{s - p_j}, \tag{1.3.8}$$

where $\tilde{n} \leq n$ is the number of finite poles (eigenvalues), and R_∞ is the contribution due to poles (eigenvalues) at $\pm\infty$ ($R_\infty = 0$ often). Poles p_j with large $\|R_j\|_2/|\mathrm{Re}(p_j)|$ (or $\|R_j\|_2$) are also called dominant poles and are studied in Chapters 2–6.

Concepts such as controllability and observability, the corresponding gramians, and Hankel singular values can be generalized to descriptor systems, but there is no uniform approach in the literature. Interpretation of these concepts in terms of states that are hard to reach and to observe, however, remains valid and is sufficient for the material in this thesis; most of the systems considered in this thesis are descriptor systems. The reader is referred to [149] and references therein for a clear generalization and applications to model order reduction.

1.4 Model order reduction

The model order reduction problem is to find, given an n-th order (descriptor) dynamical system (E, A, B, C, D) (1.3.6), a k-th order system $(\widetilde{E}, \widetilde{A}, \widetilde{B}, \widetilde{C}, D)$:

$$\begin{cases} \widetilde{E}\dot{\tilde{\mathbf{x}}}(t) &= \widetilde{A}\tilde{\mathbf{x}}(t) + \widetilde{B}\mathbf{u}(t) \\ \tilde{\mathbf{y}}(t) &= \widetilde{C}^*\tilde{\mathbf{x}}(t) + D\mathbf{u}(t), \end{cases} \tag{1.4.1}$$

where $k < n$, and $\widetilde{E}, \widetilde{A} \in \mathbb{R}^{k\times k}$, $\widetilde{B} \in \mathbb{R}^{k\times m}$, $\widetilde{C} \in \mathbb{R}^{k\times p}$, $\tilde{\mathbf{x}}(t) \in \mathbb{R}^k$, $\mathbf{u}(t) \in \mathbb{R}^m$, $\tilde{\mathbf{y}}(t) \in \mathbb{R}^p$, and $D \in \mathbb{R}^{p\times m}$. The number of inputs and outputs are the same as for the original system, and the corresponding transfer function becomes

$$\widetilde{H}(s) = \widetilde{C}^*(s\widetilde{E} - \widetilde{A})^{-1}\widetilde{B} + D.$$

The reduced-order model (1.4.1) should satisfy some or all of the following requirements [6]:

1. The approximation error must be small, and a global error bound should exist. Usually this means that the output error $\|\mathbf{y}(t) - \tilde{\mathbf{y}}(t)\|$ should be minimized for some or even all inputs $\mathbf{u}(t)$ in an appropriate norm.
2. The order of the reduced system is much smaller than the order of the original system: $k \ll n$.
3. Preservation of (physical and numerical) properties such as stability and passivity.
4. The procedure must be computationally stable and efficient.
5. The procedure must be based on some error tolerance (automatically) and a cheap measurement for the error is desired.

Depending on the application area, there may be some additional requirements:

- The reduced-order model should preserve the structure of the original model. In many practical situations, the original model is a linearization of a second order system, so that the system matrices (E, A, B, C) have a specific structure that needs to be preserved in the system matrices $(\widetilde{E}, \widetilde{A}, \widetilde{B}, \widetilde{C})$. See also [11, 55].
- A step further, the reduced-order model itself must be realizable, that is, the model should be realized as physical device. In general, reduced-order models have no clear physical meaning.
- The procedure must fit in existing simulation software, for instance in a hierarchical electric circuit simulator.

All the model order reduction methods discussed in this thesis have the common property that the reduced-order model is constructed via a Petrov-Galerkin type of projection

$$(\widetilde{E}, \widetilde{A}, \widetilde{B}, \widetilde{C}, D) \equiv (Y^*EX, Y^*AX, Y^*B, X^*C, D),$$

where $X, Y \in \mathbb{R}^{n\times k}$ are matrices whose columns form bases for relevant subspaces of the state-space.

Concerning the first requirement, the approximation error $\mathbf{y}(t) - \tilde{\mathbf{y}}(t)$ can be measured in several norms:

- The "eye-norm": a Bode magnitude (phase) plot of a transfer function plots the magnitude (phase) of $H(i\omega)$, usually in decibel, for a number of frequencies ω in the frequency range of interest, see Figure 1.1. If the transfer function of the original system can be evaluated at enough $s = i\omega$ to produce an accurate Bode plot, the original frequency response can be compared

with the frequency response of the reduced model. The degree in which the responses match may give an indication of the quality of the reduced model and may be sufficient for certain applications, but it should be noted that the relative errors may give a different view. In practical situations the experience of a domain specialist, for instance an electrical engineer, may be helpful to judge the quality and usefulness of the reduced-order model.

- The induced $\|\cdot\|_2$ norm or $\|\cdot\|_\infty$ norm: via Parseval's identity, the operator norm induced by the 2-norm in the frequency domain is defined as [62]

$$\|H\|_\infty \equiv \sup_{\omega\in\mathbb{R}} \sigma_{\max}(H(i\omega)),$$

where $\sigma_{\max}$ is the maximum singular value. This gives for the approximation error

$$\|\mathbf{y} - \tilde{\mathbf{y}}\|_2 = \|Y - \widetilde{Y}\|_2 \leq \|H - \widetilde{H}\|_\infty \|\mathbf{u}\|_2,$$

where Y and $\widetilde{Y}$ are the Laplace transforms of $\mathbf{y}$ and $\tilde{\mathbf{y}}$, respectively. For balanced truncation and modal truncation methods, to be described in the next section, there exist upper bounds for this error, although not always easily computable.

- The Hankel norm $\|\cdot\|_{\mathcal{H}}$: it is possible to construct a realization for the error transfer function $H(s) - \widetilde{H}(s)$, see [62], and the error can be measured in the Hankel norm

$$\|H - \widetilde{H}\|_{\mathcal{H}} \leq \|H - \widetilde{H}\|_\infty.$$

Also for Hankel norm approximation there exist upper bounds for this error, although not always easily computable.

1.5 Methods for eigenvalue problems and model order reduction

Roughly speaking, the methods for eigenvalue problems can be divided in two categories (see [156] for a historical background of eigenproblems): *full space* methods based on the QR method [52] for the standard eigenproblem, and the QZ method [101] for the generalized eigenproblem (see [64, Chapter 7] for efficient implementations), and *iterative subspace* methods based on the Lanczos [83], Arnoldi [7], Davidson [37], and Jacobi-Davidson [140] methods. Although the full space methods are sometimes called direct methods, they are in fact iterative as well. The complexity of the full space methods is $O(n^3)$, where n is the order of the matrix, irrespective of the sparsity, and hence they are only applicable to moderately sized problems. The complexity of the iterative subspace methods, on the other hand, usually depends on the number of nonzero elements in A (and B), and are especially applicable to large-scale sparse matrices of practically unlimited order. Full space methods usually compute the complete spectrum (and corresponding

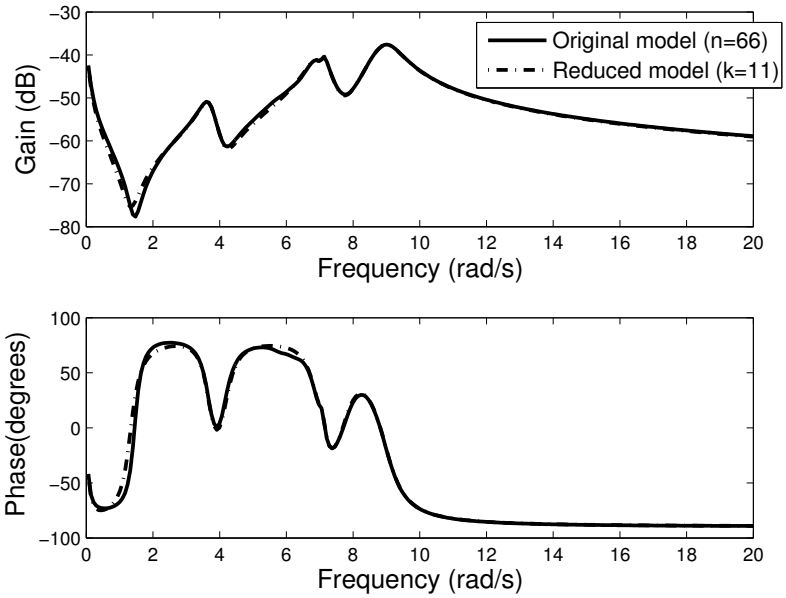


Figure 1.1: Bode magnitude (upper) and phase plot of original 66-th order model (solid), and 11-th order reduced model (dash-dot). The frequency response of the reduced model shows small deviations from the exact frequency response.

eigenvectors), while iterative subspace methods typically focus on the computation of a few specific eigenvalues and eigenvectors.

Basically, there are three main approaches for model order reduction: (1) methods based on balanced truncation [102] and Hankel norm approximation [62], (2) Padé [14] and moment matching [67] type methods, and (3) modal approximation methods [38]. The balanced truncation based methods can also be interpreted as (iterative) full space methods and have complexity $O(n^3)$, although there are developments that make these methods applicable to large sparse systems (see [18] and references therein). Moment matching methods, on the other hand, are usually based on Krylov subspace techniques such as Lanczos and Arnoldi (or rational variants [131]), and applicable to large-scale systems. Modal approximation requires selection of dominant eigenvalues (and eigenvectors) and these can be computed via full space methods (QR, QZ) or iterative subspace methods. One of the contributions of this thesis is an efficient and effective iterative subspace method to compute specifically the dominant poles and corresponding eigenvectors (Chapters 2–6).

Without loss of generality, $D = 0$ in the following, unless stated otherwise. The moments are the coefficients of a power series expansion of $H(s)$ around a finite $s_0 \notin \Lambda(A, E)$:

$$H(s) = C^*(sE - A)^{-1}B = \sum_{i=0}^{\infty} M_i(s - s_0)^i,$$

where $M_i = -C^*((s_0E - A)^{-1}E))^i(s_0E - A)^{-1}B$ for $i \geq 0$ are called the shifted moments, and, if $s_0 = 0$, the $M_i = -C^*(A^{-1}E))^iA^{-1}B$ are simply called the moments of $H(s)$. If E is nonsingular, a Neumann expansion around $s_0 = \infty$ gives $M_i = C^*(E^{-1}A)^{-i-1}E^{-1}B$ for $i < 0$, called the Markov parameters.

For a SISO system $(E, A, \mathbf{b}, \mathbf{c})$, a reduced-order system $(\widetilde{E}, \widetilde{A}, \tilde{\mathbf{b}}, \tilde{\mathbf{c}})$ with transfer function $\widetilde{H}$ and moments $\widetilde{M}$ is called a k-th Padé approximant if [14]

$$H(s) = \widetilde{H}(s) + O((s - s_0)^{2k}), \quad (s \to s_0),$$

with $M_i = \widetilde{M}_i$ for $i = 0, 1, \ldots, 2k-1$. A reduced-order model whose first $2k$ Markov parameters are equal to the first $2k$ Markov parameters of the original system is called a partial realization [67]. A multipoint Padé approximation or rational interpolant is a reduced-order model whose moments $M_i^{(j)}$ match the first $2J_j$ moments at σ_j, with $i = 0, \ldots, J_j-1$ and $J_0+\cdots+J_{l-1} = k$ for l interpolation points σ_j $(j = 0, 1, \ldots, l-1)$ [67, 14]. At the interpolation points σ_j, Padé approximations are exact, but accuracy is lost away from σ_j, even more rapidly if σ_j is close to a pole [14, 30]. Therefore, state-of-the-art moment matching techniques employ multiple interpolation points in order to match the frequency response for a large frequency range of interest [58, 70]. However, the choice of the interpolation points σ_j, and number of moments to match around each point, is usually not easy, except possibly in applications such as electric circuit simulation, where the operating frequency seems to be a good interpolation point [18, 48].

The Arnoldi and Lanczos based methods, and related model order reduction methods, construct bases for Krylov subspaces. A k-th order Krylov subspace for a square matrix A and a vector $\mathbf{v}$ is defined as

$$\mathcal{K}^k(A, \mathbf{v}) = \text{span}(\mathbf{v}, A\mathbf{v}, \ldots, A^{k-1}\mathbf{v}),$$

and can be efficiently computed if applications of A to $\mathbf{v}$ are cheap. Together with the moment matching property (see Section 1.5.3 and Section 1.5.4) this makes Krylov subspace methods successful for large sparse matrices. The methods differ in the way the bases are computed: theoretically the subspaces are the same, but numerically the condition of the bases may vary considerably. Note that Krylov subspaces are invariant under shift and scaling of A, i.e. $\mathcal{K}^k(\alpha A+\sigma I, \mathbf{v}) = \mathcal{K}^k(A, \mathbf{v})$, but that in general $\mathcal{K}^k((\sigma I - A)^{-1}, \mathbf{v}) \neq \mathcal{K}^k(A, \mathbf{v})$. Methods based on rational interpolation construct bases for rational Krylov subspaces [131]. For a regular matrix pencil (A, E), l vectors $\mathbf{v}_j$ and l shifts σ_j, a rational Krylov subspace is defined as [131]

$$\sum_{j=1}^{l} \mathcal{K}^{J_j}((\sigma_j E - A)^{-1}E, \mathbf{v}_j),$$

where $J_1 + \cdots + J_l = k$.

Each of the following subsections briefly describes an eigenvalue method together with (closely) related model order reduction methods, and provides references for further reading. Other overviews of model order reduction methods are given in [5, 18].

1.5.1 Full space methods and balanced truncation

Full space methods such as QR and QZ can be used to computed complete eigendecompositions of eigenproblems. The SVD [63], that computes the singular value or spectral decomposition $A = U\Sigma V^*$ of a general matrix A, is an important ingredient for balanced truncation based model order reduction methods. Given a state-space system (A, B, C, D), balanced truncation constructs a reduced-order model by truncating a balanced realization of (A, B, C, D). A balanced realization (A_b, B_b, C_b, D_b) is a realization for which the controllability and observability gramians are equal diagonal matrices $P_b = Q_b = \mathrm{diag}(\sigma_1^{\mathcal{H}}, \ldots, \sigma_n^{\mathcal{H}})$, where (for stable systems) P_b and Q_b are the solutions of the Lyapunov equations

$$A_b P_b + P_b A_b^* + B_b B_b^* = 0, \quad \text{and} \quad A_b^* Q_b + Q_b A_b + C_b C_b^* = 0, \tag{1.5.1}$$

and $\sigma_i^{\mathcal{H}} = \sqrt{\lambda_i(P_b Q_b)}$ are the Hankel singular values (see also Section 1.3). Recall that the Hankel singular values are input-output invariants because PQ is similar under state-space transformations $\mathbf{x} \to T\mathbf{x}$. A state-space transformation defined by T that transforms (A, B, C, D) to a balanced realization

$$(TAT^{-1}, TB, T^{-T}C, D) = \left(\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}, D \right),$$

with $A_{11} \in \mathbb{R}^{k\times k}$, $B_1 \in \mathbb{R}^{k\times m}$, $C_1 \in \mathbb{R}^{k\times p}$ and $k < n$, is called a balancing transformation. For minimal (complete) systems there exists such a balancing transformation. Balanced truncation [102] constructs a reduced k-order model of (A, B, C, D) by truncating:

$$(\widetilde{A}, \widetilde{B}, \widetilde{C}, \widetilde{D}) = (A_{11}, B_1, C_1, D).$$

Important properties [62, 102] of this truncated balanced realization are that the gramians $\widetilde{P}$ and $\widetilde{Q}$ are balanced and equal to $\widetilde{P} = \widetilde{Q} = \mathrm{diag}(\sigma_1^{\mathcal{H}}, \ldots, \sigma_k^{\mathcal{H}})$, and that there is the absolute error bound [45, 62]

$$\|H - \widetilde{H}\|_\infty \leq 2 \sum_{i=k+1}^{n} \sigma_i^{\mathcal{H}}. \tag{1.5.2}$$

It follows that the reduced-order model preserves the largest Hankel singular values, and moreover, that k can be chosen such that the error (1.5.2) is smaller than the required tolerance. This all is applicable, however, under the assumption that all Hankel singular values are known, or at least computed in order of decreasing magnitude.

The first step in balanced truncation of a stable minimal system is to compute the unique positive definite solutions P and Q of the Lyapunov equations (1.5.1). Given P and Q, a balancing transformation T can be computed as $T = \Sigma^{\frac{1}{2}} U^T R^{-T}$, where $P = R^T R$ is a Cholesky factorization and $U\Sigma^2 U^T = RQR^T$ is an SVD of RQR^T. The state-space transformation $\mathbf{x} \to T\mathbf{x}$ is a balancing transformation, $P_b = Q_b = \Sigma$, and the balanced system can be truncated up to required accuracy

by using the bound in (1.5.2). Note that this truncated system is not an optimal approximation. In [62] it is shown how, using a balancing transformation, an approximation can be constructed that is optimal in the Hankel norm, that is, $\sigma_{k+1}^{\mathcal{H}} \leq \|H - \widetilde{H}\|_{\mathcal{H}} < \sigma_k^{\mathcal{H}}$ (this approximation also has error bound (1.5.2) and hence is not necessarily more accurate than the balanced truncation, in the $\|\cdot\|_\infty$ norm).

There are several ways to compute full solutions P and Q of the Lyapunov equations, see for instance [5, Chapter 6] and references therein, but these (iterative) full space solution methods have complexity $O(n^3)$ (except in some cases for the ADI iteration [162]). Also the SVD and Cholesky factorization have complexity $O(n^3)$ for general matrices [64]. Hence, when relying on full space methods, balanced truncation is only applicable to systems of moderate order n and not feasible for large-scale sparse systems. The full space balanced truncation methods are also called SVD-type model order reduction methods.

There are methods that compute low-rank solutions of the Lyapunov equations [87, 115] or combine SVD-type methods with (rational) Krylov approaches [71], and these methods make it possible to apply balanced truncation to large sparse systems as well (see [5, 18, 19] for more details and references). Generalizations and methods for Lyapunov equations arising from descriptor systems are described in [149]. Balanced truncation is a concept in the context of proper orthogonal decomposition (POD), also known as the method of empirical eigenfunctions, a method that tries to extract the dominant dynamics from the time response and that is also applicable to nonlinear systems (see [5, Section 9.1]).

1.5.2 The Power method and Asymptotic Waveform Evaluation

Given a matrix $A \in \mathbb{R}^{n\times n}$ and a vector $\mathbf{v}_1$ with $\|\mathbf{v}_1\|_2 = 1$, the power method computes the sequence $\{\mathbf{v}_i\}_{i>0}$ as $\mathbf{v}_1, \mathbf{v}_i = A\mathbf{v}_{i-1}/\|A\mathbf{v}_{i-1}\|_2$ $(i > 1)$, and corresponding approximate eigenvalues via the Rayleigh quotient $\mathbf{v}_i^* A\mathbf{v}_i/\mathbf{v}_i^*\mathbf{v}_i$, as is shown in Algorithm 1.1. It is well known that the sequence converges to the eigenvector corresponding to the dominant (in absolute value) eigenvalue (if it is simple) of A if $\mathbf{v}_1$ has a component in that direction, see for example [156, Chapter 4].

Algorithm 1.1 The Power method

INPUT: $n \times n$ matrix A, initial vector $\mathbf{v}_1 \neq 0$
OUTPUT: Dominant eigenpair $(\lambda_1, \mathbf{x}_1)$

1: $\mathbf{v}_1 = \mathbf{v}_1/\|\mathbf{v}_1\|_2$
2: **for** $i = 1, 2, \ldots,$until convergence **do**
3: $\quad \mathbf{v}_{i+1} = A\mathbf{v}_i$
4: $\quad \theta_i = \mathbf{v}_i^*\mathbf{v}_{i+1}$
5: $\quad \mathbf{v}_{i+1} = \mathbf{v}_{i+1}/\|\mathbf{v}_{i+1}\|_2$
6: **end for**

Asymptotic Waveform Evaluation (AWE) [30, 117] computes the $2k$ moments m_i of a SISO transfer function $H(s)$ explicitly by applying the power method to $(s_0E - A)^{-1}E$ and $\mathbf{v}_1 = \mathbf{b}$, for an expansion point s_0 (typically $s_0 = 0$ or $s_0 = \infty$). In the second step, the transfer function of the approximate k-th order realization is forced to match the $2k$ moments m_i of the original impulse response, which, although not mentioned in the original AWE literature, can be achieved by using a projector based on the vectors generated by the power method [57].

AWE suffers from numerical problems that are caused by the explicit moment matching via the power method: since the $\mathbf{v}_i$ converge to the eigenvector corresponding to the (absolutely) largest eigenvalue, the $\mathbf{v}_i$ and computed moments $m_i = \mathbf{c}^*\mathbf{v}_i$ practically contain dominant information on the largest eigenvalue, resulting in a poor reduced-order model. These effects are already notable for small values of i, and although they can be handled by using several different expansion points s_i, AWE is rarely applicable in practice. For more details on the numerical problems with AWE see [48, 57].

After k iterations, the vectors $\mathbf{v}_i$ generated by the power method span the Krylov subspace

$$\mathcal{K}^k(A, \mathbf{v}_1) = \text{span}(\mathbf{v}_1, A\mathbf{v}_1, \ldots, A^{k-1}\mathbf{v}_1),$$

but from a numerical viewpoint they form an ill-conditioned basis. The Lanczos and Arnoldi methods, to be discussed in the next subsections, are numerically more stable methods for the construction of orthonormal bases for $\mathcal{K}^k(A, \mathbf{v}_1)$.

1.5.3 The Lanczos method and Padé Via Lanczos

Lanczos [83] proposes a numerically more stable method to construct a basis $(\mathbf{v}_1, \ldots, \mathbf{v}_{k+1})$ for the Krylov subspace $\mathcal{K}^{k+1}(A, \mathbf{v}_1)$ for a general matrix A and vector $\mathbf{v}_1$. In fact, the unsymmetric or two-sided Lanczos method computes bi-orthogonal bases $(\mathbf{v}_1, \ldots, \mathbf{v}_{k+1})$ and $(\mathbf{w}_1, \ldots, \mathbf{w}_{k+1})$ for the Krylov subspaces $\mathcal{K}^{k+1}(A, \mathbf{v}_1)$ and $\mathcal{K}^{k+1}(A^*, \mathbf{w}_1)$, respectively, by the following two three term recurrence relations ($\mathbf{v}_0 = \mathbf{w}_0 = 0$):

$$\begin{aligned} \rho_{i+1}\mathbf{v}_{i+1} &= A\mathbf{v}_i - \alpha_i\mathbf{v}_i - \beta_i\mathbf{v}_{i-1}, \quad (i = 1, \ldots, k), \\ \eta_{i+1}\mathbf{w}_{i+1} &= A^*\mathbf{w}_i - \bar{\alpha}_i\mathbf{w}_i - \gamma_i\mathbf{w}_{i-1}, \quad (i = 1, \ldots, k), \end{aligned}$$

as is also shown in Alg. 1.2.

Algorithm 1.2 The two-sided Lanczos method

INPUT: $n \times n$ matrix A, nonzero vectors $\mathbf{v}_1, \mathbf{w}_1$ with $\mathbf{w}_1^* \mathbf{v}_1 \neq 0$
OUTPUT: $n \times k$ matrices $V = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$, $W = [\mathbf{w}_1, \ldots, \mathbf{w}_k]$,
$k \times k$ matrices $T = \text{tridiag}(\rho_{2:k}, \alpha_{1:k}, \beta_{2:k})$, $S = \text{tridiag}(\eta_{2:k}, \bar{\alpha}_{1:k}, \gamma_{2:k})$,
vectors $\mathbf{v}_{k+1}, \mathbf{w}_{k+1}$, scalars $t_{k+1,k}, s_{k+1,k}$

1: $\rho_1 = \|\mathbf{v}_1\|_2$, $\eta_1 = \|\mathbf{w}_1\|_2$, $\mathbf{v}_1 = \mathbf{v}_1/\rho_1$, $\mathbf{w}_1 = \mathbf{w}_1/\eta_1$
2: $\mathbf{v}_0 = 0$, $\mathbf{w}_0 = 0$, $\delta_0 = 1$
3: **for** $i = 1, 2, \ldots, k$ **do**
4: $\quad \delta_i = \mathbf{w}_i^* \mathbf{v}_i$
5: $\quad \mathbf{x} = A\mathbf{v}_i$
6: $\quad \alpha_i = \mathbf{w}_i^* \mathbf{x}/\delta_i$
7: $\quad \beta_i = \delta_i \eta_i / \delta_{i-1}$
8: $\quad \gamma_i = \delta_i \rho_i / \delta_{i-1}$
9: $\quad \mathbf{x} = \mathbf{x} - \alpha_i \mathbf{v}_i - \beta_i \mathbf{v}_{i-1}$
10: $\quad \mathbf{y} = A^* \mathbf{w}_i - \bar{\alpha}_i \mathbf{w}_i - \gamma_i \mathbf{w}_{i-1}$
11: $\quad \rho_{i+1} = \|\mathbf{x}\|_2$, $\eta_{i+1} = \|\mathbf{y}\|_2$
12: $\quad \mathbf{v}_{i+1} = \mathbf{x}/\rho_{i+1}$, $\mathbf{w}_{i+1} = \mathbf{y}/\eta_{i+1}$
13: **end for**

The coefficients $\alpha_j, \beta_j, \gamma_j, \delta_j, \eta_j, \rho_j$ are computed such that the basis vectors are biorthogonal, i.e. $\mathbf{w}_i^* \mathbf{v}_j = 0$ if $i \neq j$ and $\mathbf{w}_i^* \mathbf{v}_j = \delta_j$ if $j = i$, for $i, j = 1, \ldots, k+1$. Then the following relations hold:

$$\begin{aligned} AV_k &= V_k T_k + \rho_{k+1} \mathbf{v}_{k+1} \mathbf{e}_k^T, \\ A^* W_k &= W_k S_k + \eta_{k+1} \mathbf{w}_{k+1} \mathbf{e}_k^T, \end{aligned}$$

where $V_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k] \in \mathbb{R}^{n \times k}$ and $W_k = [\mathbf{w}_1, \ldots, \mathbf{w}_k] \in \mathbb{R}^{n \times k}$, and $T_k = \text{tridiag}(\rho_{2:k}, \alpha_{1:k}, \beta_{2:k})$ and $S_k = \text{tridiag}(\eta_{2:k}, \bar{\alpha}_{1:k}, \gamma_{2:k})$ are tridiagonal $k \times k$ matrices. The eigenvalues of T (S) are approximate eigenvalues of A (A^*), also known as Petrov values. The process can suffer from break down if $\mathbf{w}_i^* \mathbf{v}_i = 0$ for nonzero $\mathbf{v}_i$ and $\mathbf{w}_i$, and remedies are described in for example [56]. See [64, 156] and references therein for more details and robust implementations of Lanczos methods.

Padé Via Lanczos (PVL), derived independently in [48, 57], uses the two-sided Lanczos method to deal with the problems of AWE. For a single interpolation point $\sigma_0 \notin \Lambda(A, E)$, the transfer function of the SISO system $(E, A, \mathbf{b}, \mathbf{c})$ can be rewritten as

$$H(s) = \mathbf{c}^*(sE - A)^{-1}\mathbf{b} = \mathbf{c}^*(I + (s - \sigma_0)(\sigma_0 E - A)^{-1}E)^{-1}((\sigma_0 E - A)^{-1}\mathbf{b}).$$

Two-sided Lanczos can be used to compute bi-orthogonal bases $(\mathbf{v}_1, \ldots, \mathbf{v}_k)$ and $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$ for the Krylov spaces $\mathcal{K}((s_0 E - A)^{-1}E, (s_0 E - A)^{-1}\mathbf{b})$ and $\mathcal{K}((s_0 E - A)^{-*}E^*, \mathbf{c})$, and the reduced-order model is constructed as $(T_k, \sigma_0 T_k - I, W_k^*(\sigma_0 E - A)^{-1}\mathbf{b}, V_k^* \mathbf{c})$. It can be shown that the reduced-order model preserves $2k$ moments [48, Section 3] and [57, Thm. 1]. In general PVL is not stability preserving, since the Petrov values (eigenvalues of T_k) are not necessarily located in the left half

plane, even if that is the case for the eigenvalues of (E, A). Passivity preserving PVL like methods exist for specific applications as *RLC*-circuits (SyPVL) [53], and there are also PVL methods for MIMO systems (MPVL) [54]. Because of its short recurrences, PVL is an efficient way to create a reduced model, but it may have numerical difficulties due to break down; see [49] for a more robust implementation (via block Lanczos). In [13] an estimate is given for the approximation error of the model computed by PVL.

As mentioned before, single interpolation point methods usually produce reduced-order models that are accurate in the neighborhood of the interpolation point. Rational Krylov methods [131] can be used to construct bases for rational Krylov subspaces, that may lead to improved reduced-order models. In [58, 70], a model order reduction method based on a rational Lanczos scheme is presented. In the SISO case, a reduced-order model is constructed as $(W^*EV, W^*AV, W^*\mathbf{b}, V^*\mathbf{c})$, where the columns of the $n \times k$ matrices V and W form bases for the rational Krylov subspaces

$$\sum_{j=1}^{l} \mathcal{K}^{J_j}((\sigma_j E - A)^{-1}E, (\sigma_j E - A)^{-1}\mathbf{b}),$$

and

$$\sum_{j=1}^{l} \mathcal{K}^{J_j}((\sigma_j E - A)^{-*}E^*, (\sigma_j E - A)^{-*}\mathbf{c}),$$

respectively, with $2J_j$ the number of moments to match around interpolation point σ_j, with $J_1 + \cdots + J_l = k$. The reader is referred to Section 3.8 and [70] for more information about two-sided methods. For a two-sided interpolation approach for MIMO systems, see [59].

1.5.4 The Arnoldi method and PRIMA

Arnoldi [7] computes an orthonormal basis $(\mathbf{v}_1, \ldots, \mathbf{v}_{k+1})$ for the Krylov subspace $\mathcal{K}^{k+1}(A, \mathbf{v}_1)$ for a general matrix A and vector $\mathbf{v}_1$. Given a basis $(\mathbf{v}_1, \ldots, \mathbf{v}_i)$, the next basis vector $\mathbf{v}_{i+1}$ is computed as the normalized orthogonal complement of $A\mathbf{v}_i$ in $\text{span}(\mathbf{v}_1, \ldots, \mathbf{v}_i)$. Algorithm 1.3 shows the Arnoldi process with repeated modified Gram-Schmidt orthogonalization (MGS): if the new vector after orthogonalization is a fraction $\gamma < 1$ of the vector before orthogonalization, it is orthogonalized again to deal with cancellation effects (a practical choice is $\gamma = 0.1$) [36, 64].

Algorithm 1.3 The Arnoldi method

INPUT: $n \times n$ matrix A, nonzero vector $\mathbf{v}_1$
OUTPUT: $n \times k$ matrix $V = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$, $k \times k$ matrix $H = [h_{ij}]$, vector $\mathbf{v}_{k+1}$, scalar $h_{k+1,k}$

$\mathbf{v}_1 = \mathbf{v}_1/\|\mathbf{v}_1\|_2$, $H = 0^{k\times k}$
for $i = 1, 2, \ldots, k$ **do**
  $\mathbf{w} = A\mathbf{v}_i$
  $(\mathbf{w}, \mathbf{g})$ =MGS($[\mathbf{v}_1, \ldots, \mathbf{v}_i], \mathbf{w}$) {Alg. 1.4}
  $h_{ji} = g_j$ $(j = 1, \ldots, i)$
  $h_{i+1,i} = \|\mathbf{w}\|_2$
  $\mathbf{v}_{i+1} = \mathbf{w}/h_{i+1,i}$
end for

Algorithm 1.4 Modified Gram-Schmidt (MGS)

INPUT: $n \times k$ matrix $V = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$, nonzero vector $\mathbf{v}$, $\gamma < 1$
OUTPUT: vector $\mathbf{w} \perp V$, $k \times 1$ vector $\mathbf{h}$

$\mathbf{w} = \mathbf{v}$
$\tau_0 = \|\mathbf{w}\|_2$
for $i = 1, \ldots, k$ **do**
  $h_i = \mathbf{v}_i^*\mathbf{w}$
  $\mathbf{w} = \mathbf{w} - h_i\mathbf{v}_i$
end for
if $\|\mathbf{w}\|_2 \leq \gamma\tau_0$ **then**
  for $i = 1, \ldots, k$ **do**
    $\alpha = \mathbf{v}_i^*\mathbf{w}$
    $\mathbf{w} = \mathbf{w} - \alpha\mathbf{v}_i$
    $h_i = h_i + \alpha$
  end for
end if

The basis vectors are related by

$$AV_k = V_k H_k + h_{k+1,k}\mathbf{v}_{k+1}\mathbf{e}_k^T = V_{k+1}\underline{H}_k, \quad V_{k+1}^T V_{k+1} = I, \tag{1.5.3}$$

where $V_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k] \in \mathbb{R}^{n\times k}$, $H_k \in \mathbb{R}^{k\times k}$, and $\underline{H}_k = [H_k^T, h_{k+1,k}\mathbf{e}_k]^T \in \mathbb{R}^{(k+1)\times k}$ are upper Hessenberg[2]. Relation (1.5.3) characterizes a k-step Arnoldi factorization. Approximate eigenpairs $(\theta_i, V_k\mathbf{y}_i)$, called Ritz pairs, can be computed from eigenpairs $(\theta_i, \mathbf{y}_i)$ of H_k. See Chapter 7, and [64, 156] and references therein for more details and efficient implementations of the Arnoldi method. If A is symmetric, it follows that H_k is a symmetric tridiagonal matrix, and the Arnoldi algorithm reduces (mathematically) to the three term recurrence known as the Lanczos method [83].

[2]Barred identifiers $\underline{H}_k$ are elements of $\mathbb{R}^{(k+1)\times k}$, whereas $H_k \in \mathbb{R}^{k\times k}$.

Consider a single interpolation point $\sigma_0 \notin \Lambda(A, E)$ and rewrite the transfer function of the square (p inputs - p outputs) system (E, A, B, C) as

$$H(s) = C^*(sE - A)^{-1}B = C^*(I + (s - \sigma_0)(\sigma_0 E - A)^{-1}E)^{-1}((\sigma_0 E - A)^{-1}B).$$

Most of the Arnoldi-based model order reduction methods use a (block) Arnoldi method [132] to construct an orthonormal basis $(\mathbf{v}_1, \ldots, \mathbf{v}_{kp})$ for the (block) Krylov space $\mathcal{K}((s_0E - A)^{-1}E, (s_0E - A)^{-1}B)$, but they differ in how the reduced-order system matrices $(\widetilde{E}, \widetilde{A}, \widetilde{B}, \widetilde{C})$ are computed. In [136] the reduced-order model is constructed as $(H_k, \sigma_0 H_k - I, V_k^* B, V_k^* C)$. This reduced-order model matches k moments, but does not preserve passivity. The Passive Reduced-Order Interconnect Macromodeling Algorithm (PRIMA) [105] constructs the reduced model as $(V_k^* E V_k, V_k^* A V_k, V_k^* B, V_k^* C)$ and is passivity preserving (if E is symmetric semi-positive definite and $B = C$, as is often the case in simulation of RLC-circuits). In [73] efficient techniques for building the bases for the block Krylov subspaces are described. Structure-preserving variants are described in [11, 55].

In [70], a model order reduction method based on a two-sided rational Arnoldi scheme is presented. In the SISO case, a reduced-order model is constructed as $(W^*EV, W^*AV, W^*\mathbf{b}, V^*\mathbf{c})$, where the columns of the $n \times k$ matrices V and W form bases for the rational Krylov subspaces

$$\sum_{j=1}^{l} \mathcal{K}^{J_j}((\sigma_j E - A)^{-1}E, (\sigma_j E - A)^{-1}\mathbf{b}),$$

and

$$\sum_{j=1}^{l} \mathcal{K}^{J_j}((\sigma_j E - A)^{-*}E^*, (\sigma_j E - A)^{-*}\mathbf{c}),$$

respectively, with $2J_j$ the number of moments to match around interpolation point σ_j, with $J_1 + \cdots + J_l = k$. Note that in contrast with the standard (single-sided) Arnoldi methods such as PRIMA, here twice as many moments are matched (k vs. $2k$), at the costs of (in total) k additional matrix vector products with $(\sigma_j E - A)^{-*}E^*$. The reader is referred to Section 3.8 of Chapter 3 and [70] for more information on two-sided methods.

1.5.5 Jacobi-Davidson, the Dominant Pole Algorithm and modal approximation

The Jacobi-Davidson method [140] combines two principles to compute eigenpairs of eigenvalue problems $A\mathbf{x} = \lambda\mathbf{x}$. The first (Davidson) principle is to apply a Ritz-Galerkin approach with respect to the search space spanned by the orthonormal columns of $V_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$:

$$AV_k\mathbf{s} - \theta V_k\mathbf{s} \perp \{\mathbf{v}_1, \ldots, \mathbf{v}_k\},$$

which leads to k Ritz pairs $(\theta_i, \mathbf{q}_i = V_k\mathbf{s}_i)$, where $(\theta_i, \mathbf{s}_i)$ are eigenpairs of $V_k^* A V_k$. The second (Jacobi) principle is to compute a correction $\mathbf{t}$ orthogonal to the selected

eigenvector approximation $\mathbf{q}$ (for instance, corresponding to the largest Ritz value θ) from the Jacobi-Davidson correction equation

$$(I - \mathbf{q}\mathbf{q}^*)(A - \theta I)(I - \mathbf{q}\mathbf{q}^*)\mathbf{t} = -(A\mathbf{q} - \theta\mathbf{q}).$$

The search space is expanded with (an approximation of) $\mathbf{t}$. A Ritz pair is accepted if $\|\mathbf{r}\|_2 = \|A\mathbf{q} - \theta\mathbf{q}\|_2$ is smaller than a given tolerance. A two-sided variant, called two-sided Jacobi-Davidson [74, 148], is shown in Alg. 1.5 (see Sections 3.4–3.6 of Chapter 3 for more details).

Algorithm 1.5 The two-sided Jacobi-Davidson method

INPUT: $n \times n$ matrix A, initial vectors $\mathbf{v}_1, \mathbf{w}_1$ ($\mathbf{w}_1^*\mathbf{v}_1 \neq 0$), tolerance ϵ
OUTPUT: approximate eigentriplet $(\theta, \mathbf{v}, \mathbf{w})$ with
$\min(\|A\mathbf{v} - \theta\mathbf{v}\|, \|A^*\mathbf{w} - \theta^*\mathbf{w}\|) \leq \epsilon$

1: $\mathbf{s} = \mathbf{v}_1, \mathbf{t} = \mathbf{w}_1$
2: $U_0 = V_0 = W_0 = H_0 = []$
3: **for** $i = 1, 2, \ldots$ **do**
4: $\quad \mathbf{v}_i = \mathrm{MGS}(V_{i-1}, \mathbf{s})$, $\mathbf{w}_i = \mathrm{MGS}(W_{i-1}, \mathbf{t})$ {Alg. 1.4}
5: $\quad \mathbf{v}_i = \mathbf{v}_i / \|\mathbf{v}_i\|_2$, $V_i = [V_{i-1}, \mathbf{v}_i]$
6: $\quad \mathbf{w}_i = \mathbf{w}_i / \|\mathbf{w}_i\|_2$, $W_i = [W_{i-1}, \mathbf{w}_i]$
7: $\quad \mathbf{u}_i = A\mathbf{v}_i$, $U_i = [U_{i-1}, \mathbf{u}_i]$
8: $\quad H_i = \begin{bmatrix} H_{i-1} & W_{i-1}^*\mathbf{u}_i \\ \mathbf{w}_i^*U_{i-1} & \mathbf{w}_i^*\mathbf{u}_i \end{bmatrix}$
9: $\quad$ Select suitable eigentriplet $(\theta, \mathbf{s}, \mathbf{t})$ of H_i
10: $\quad \mathbf{v} = V_i\mathbf{s}/\|V_i\mathbf{s}\|_2$, $\mathbf{w} = W_i\mathbf{t}/\|W_i\mathbf{t}\|_2$
11: $\quad \mathbf{r}_v = A\mathbf{v} - \theta\mathbf{v}$, $\mathbf{r}_w = A^*\mathbf{w} - \theta^*\mathbf{w}$
12: $\quad$ **if** $\min(\|\mathbf{r}_v\|_2, \|\mathbf{r}_w\|_2) \leq \epsilon$ **then**
13: $\quad\quad$ Stop
14: $\quad$ **end if**
15: $\quad$ Solve (approximately) $\mathbf{s} \perp \mathbf{v}$, $\mathbf{t} \perp \mathbf{w}$ from correction equations

$$\begin{aligned} (I - \frac{\mathbf{v}\mathbf{w}^*}{\mathbf{w}^*\mathbf{v}})(A - \theta I)(I - \mathbf{v}\mathbf{v}^*)\mathbf{s} &= -\mathbf{r}_v \\ (I - \frac{\mathbf{w}\mathbf{v}^*}{\mathbf{v}^*\mathbf{w}})(A - \theta I)^*(I - \mathbf{w}\mathbf{w}^*)\mathbf{t} &= -\mathbf{r}_w \end{aligned}$$

16: **end for**

The search space V_k in general is *not* a Krylov subspace, but for certain choices it can be shown to be a rational Krylov subspace (see Chapter 3 and [131]). If the correction equations are solved exactly, Jacobi-Davidson is an exact Newton method [139, 140], but one of the properties that makes Jacobi-Davidson powerful is that it is often sufficient for convergence to solve the correction equation up to moderate accuracy only, using, for instance, a (preconditioned) linear solver such as GMRES [133]. Jacobi-Davidson is especially effective for computing a number of eigenvalues near a target in the complex plane, or that satisfy a certain criterion

(such as lying in the right half-plane in stability analysis). The fact that no exact solves are required at all makes Jacobi-Davidson well-suited to large-scale standard and generalized eigenproblems. Jacobi-Davidson QR (QZ) methods, that compute partial (generalized) Schur forms for standard (generalized) eigenproblems, are described in [51, 139]. In Chapter 7 a scheme based on Jacobi-Davidson QZ is presented for the computation of right half-plane eigenvalues in the presence of eigenvalues at infinity (stability analysis), and in Chapter 8 a Jacobi-Davidson QZ variant for the computation of a partial generalized *real* Schur form is described.

The Dominant Pole Algorithm (DPA) [91] is a specialized eigenvalue method for the computation of dominant poles of a transfer function $H(s) = \mathbf{c}^*(sE - A)^{-1}\mathbf{b}$. Here, a dominant pole is a pole with a large residue (observable via peaks in the Bode plot). DPA uses the Newton method to compute the poles λ_i as zeros of the function $1/H(s)$. Starting with initial estimate s_1, a sequence of estimates is computed via

$$s_{k+1} = s_k - \frac{\mathbf{c}^*(s_kE - A)^{-1}\mathbf{b}}{\mathbf{c}^*(s_kE - A)^{-1}E(s_kE - A)^{-1}\mathbf{b}}.$$

The DPA is shown in Alg. 2.1 and its typical convergence to more dominant poles, even for initial estimates in the neighborhood of less dominant poles, is studied in detail in Chapter 2. An initial MIMO variant of DPA (MDP) is described in [93], and a block variant (DPSE) for the computation of more than one pole is presented in [90].

Although DPA is a single-pole algorithm (it computes one pole per run), different initial estimates can be used to find different dominant poles and corresponding left and right eigenvectors (with the risk of finding duplicate poles, see Chapter 3 for the multi-pole variant SADPA). Similarly, two-sided Jacobi-Davidson [74, 148] can be used to compute dominant poles and corresponding left and right eigenvectors, as is also described in Chapter 3. These dominant eigenspaces can be used for the construction of reduced-order models in the form of modal approximations. Note that the idea is to compute only the dominant eigentriplets, and not to compute a complete eigendecomposition (which is not feasible for large-scale systems). In principle, implicitly restarted Arnoldi [85, 146] and Lanczos methods can also be used for the computation of (dominant) eigentriplets. Their typical convergence to well-separated eigenvalues at the outer-edge of the spectrum, however, makes these methods less efficient than Jacobi-Davidson methods for selection criteria other than closest to a specific target [51, 139, 140].

Let (A, E) be stable and diagonalizable, and let $R_\infty = 0$. If $Y = [Y_1, Y_2] \in \mathbb{C}^{n\times n}$ and $X = [X_1, X_2] \in \mathbb{C}^{n\times n}$ are such that $Y^*AX = \Lambda = \mathrm{diag}(\Lambda_1, \Lambda_2)$ and $Y^*EX = I$ are diagonal, and ordered such that $\Lambda_1 = \mathrm{diag}(\lambda_1, \ldots, \lambda_k)$ has the k most dominant poles on its diagonal, then a modal approximation can be constructed as $(Y_1^*EX_1 = I, Y_1^*AX_1 = \Lambda_1, Y_1^*B, X_1^*C, D)$. If H_k is the transfer function of the reduced-order

model, then the truncation error becomes [68, lemma 9.2.1]

$$\begin{aligned} \|H - H_k\|_\infty &= \| \sum_{j=k+1}^{n} \frac{R_j}{s - \lambda_j} \|_\infty \\ &\leq \sum_{j=k+1}^{n} \frac{\|R_j\|_2}{|\mathrm{Re}(\lambda_j)|}, \end{aligned}$$

where $R_j = (C^* X_j)(Y_j^* B)$ are the residues. Note that

$$\|R_j\|_2 \leq \|X\|_2 \|Y\|_2 \|B\|_2 \|C\|_2,$$

and in general *all* eigenvalues and eigenvectors are needed to compute the error. However, this information is usually not available for large-scale systems. The advantages of modal truncation are that it is conceptually simple, and that the poles of the reduced-order model are also poles of the original system, so that they keep their physical interpretation as, for instance, resonance frequencies [68, p. 317] and stability is preserved.

The Dominant Pole Algorithm (DPA) plays an important role in Chapter 2 to Chapter 6, where it is studied in more detail and extended to SADPA (SAMDP) for the efficient computation of specifically the dominant spectra (and zeros) of SISO (MIMO) transfer functions, and for the construction of modal approximations.

1.6 Overview

Eigenvalues play an important role throughout this thesis. In each chapter algorithms are presented for the computation of specific eigenvalues and corresponding right (and left) eigenvectors of large sparse matrices. Furthermore, there is always a connection with model order reduction in some sense, varying from the reduction of large-scale dynamical systems, where the dominant poles are of interest, to the stability analysis of steady state solutions of discretized Navier-Stokes equations, where the rightmost eigenvalues are of importance.

Chapter 2 gives a detailed analysis of the convergence behavior of the dominant pole algorithm (DPA). For symmetric matrices, a similar method was already considered briefly by Ostrowski in 1958 [106], for the computation of a single eigenvalue, but Ostrowski brought up this iteration primarily as an introduction to the well-known Rayleigh quotient iteration (RQI), which has cubic instead of quadratic rate of convergence in the neighborhood of an eigenvalue. Both iterations use a new shift (the eigenvalue estimate) every iteration, but DPA keeps the right-hand side fixed, while RQI updates the right-hand side every iteration. Keeping the right-hand side fixed forces convergence to eigenvalues with eigenvectors in the direction of the right-hand side. The asymptotic convergence rate drops to quadratic for DPA, but in practice this only costs one or two additional iterations. The convergence behavior of DPA makes it an effective method for the computation of dominant poles of large-scale dynamical systems, and the significantly better

convergence (with respect to dominance) compared to two-sided Rayleigh quotient iteration is illustrated by numerical examples. This chapter is also available as [126]:

> Joost Rommes and Gerard L. G. Sleijpen, *Convergence of the dominant pole algorithm and Rayleigh quotient iteration*, Preprint 1356, Utrecht University, 2006,

and has been submitted for publication.

DPA is a single-pole algorithm. Given an initial estimate, it computes one dominant pole using Newton's method. In Chapter 3, DPA is extended with subspace acceleration to obtain better global convergence, and with deflation to compute more than one pole without recomputing already computed poles: subspace accelerated DPA (SADPA). Deflation can be implemented very efficiently via the fixed right-hand sides of the DPA iteration, so no explicit orthogonalizations of the search space expansion vectors are needed. Under certain conditions, SADPA is equivalent to two-sided Jacobi-Davidson and rational Krylov methods, and if the exact solves of linear systems are not feasible, inexact two-sided Jacobi-Davidson can be used to compute dominant poles. The modal approximations that can be constructed with the right and left eigenvectors of the dominant poles are compared to reduced-order models computed by rational Krylov methods, and it is shown how both methods can improve each other. This chapter is based on [123]:

> Joost Rommes and Nelson Martins, *Efficient computation of transfer function dominant poles using subspace acceleration*, IEEE Transactions on Power Systems **21** (2006), no. 3, 1218–1226,

and [120]:

> Joost Rommes, *Modal Approximation and Computation of Dominant Poles*, Model Order Reduction: Theory, Research Aspects and Applications, (H. A. van der Vorst and W. H. A. Schilders, eds.), Springer, To Appear,

but is almost completely rewritten, extended with new results, relations to Jacobi-Davidson and rational Krylov, additional numerical experiments and reflections to rational Krylov based model order reduction methods.

In Chapter 4, DPA and SADPA are generalized to algorithms for the computation of dominant poles of multi-input multi-output (MIMO) transfer functions: subspace accelerated MIMO dominant pole algorithm (SAMDP). The basic idea is the same as for (SA)DPA, but the fact that MIMO transfer functions are square or non-square matrix valued functions leads to additional algorithmic and numerical considerations. This chapter is published as [122]:

> Joost Rommes and Nelson Martins, *Efficient computation of multivariable transfer function dominant poles using subspace acceleration*, IEEE Transactions on Power Systems **21** (2006), no. 4, 1471–1483.

Chapter 5 focuses on the computation of dominant zeros of transfer functions. Like dominant poles, the dominant zeros of a transfer function are of importance for stability analysis and design of large-scale control systems. The zeros are generalized eigenvalues of a matrix pair related to the system matrices, and the dominant zeros cause the dips in the Bode magnitude plot of the transfer function. If the inverse of the transfer function exists, the zeros are equal to the poles of the inverse transfer function. Because the system matrices of the inverse transfer function are closely related to the system matrices of the original transfer function, SADPA and SAMDP can be used to compute the dominant zeros via the dominant poles of the inverse transfer function. This chapter is also available as [125]:

> Joost Rommes, Nelson Martins, and Paulo C. Pellanda, *Efficient computation of large-scale transfer function dominant zeros*, Preprint 1358, Utrecht University, 2006,

and has been submitted for publication.

In Chapter 6, the Dominant Pole Algorithm is generalized to an algorithm for the computation of dominant poles of transfer functions of second-order dynamical systems: Quadratic DPA (QDPA). In this case, the dominant poles are specific eigenvalues of a quadratic eigenvalue problem. Since QDPA works with the original system matrices, no linearization to a generalized eigenproblem is required. Furthermore, the modal approximations that are constructed using the dominant eigenspaces preserve the second-order structure of the original system. It is shown that the dominant poles can be used to improve reduced-order models computed by second-order Krylov methods [11, 12]. Generalizations to higher-order systems and MIMO systems are described, and QDPA can also be used for the computation of dominant zeros. This chapter is available as [124]:

> Joost Rommes and Nelson Martins, *Efficient computation of transfer function dominant poles of large second-order dynamical systems*, Preprint 1360, Utrecht University, 2007,

and has been submitted for publication.

In Chapter 7 algorithms based on Arnoldi and Jacobi-Davidson methods are considered for the computation of the rightmost eigenvalues of large-scale generalized eigenproblems $A\mathbf{x} = \lambda B\mathbf{x}$ with singular B. These eigenproblems arise, for instance, in stability analysis of discretized Navier-Stokes equations and large-scale power systems. Eigenvalues with positive real parts imply instability of the steady state solution and are therefore of importance. The computation of these rightmost eigenvalues is, however, complicated by the presence of eigenvalues at infinity caused by the singularity of B. These eigenvalues at infinity have no physical relevance. Standard Arnoldi and Jacobi-Davidson approaches may fail because they may interpret approximations of eigenvalues at infinity as approximations to finite eigenvalues. In this chapter, strategies and algorithms are presented for the successful computation of the finite rightmost eigenvalues. This chapter (without appendix) is also available as [121]:

> Joost Rommes, *Arnoldi and Jacobi-Davidson methods for generalized eigenvalue problems $A\mathbf{x} = \lambda B\mathbf{x}$ with singular B*, Preprint 1339, Utrecht University, 2005 (revised 2007),

and has been accepted for publication in Mathematics of Computation.

Chapter 8 presents a variant of the Jacobi-Davidson method that is specifically designed for real unsymmetric matrix pencils: real Jacobi-Davidson QZ (RJDQZ). Because the search and test space are kept purely real, this method has lower memory and computational costs than standard JDQZ. Numerical experiments confirm that also the convergence is accelerated. This chapter is published as [158]:

> Tycho van Noorden and Joost Rommes, *Computing a partial generalized real Schur form using the Jacobi-Davidson method*, Numerical Linear Algebra with Applications **14** (2007), no. 3, 197–215.

Except for Chapter 3, all chapters are available as separate papers. The notation for this thesis has been made uniform, and addenda have been added with additional remarks. Chapter 7 has been extended with an appendix.

Chapter 2

Convergence of the Dominant Pole Algorithm and Rayleigh Quotient Iteration

Abstract. The dominant poles of a transfer function are specific eigenvalues of the state-space matrix of the corresponding dynamical system. In this chapter, two methods for the computation of the dominant poles of a large-scale transfer function are studied: two-sided Rayleigh Quotient Iteration (RQI) and the Dominant Pole Algorithm (DPA). Firstly, a local convergence analysis of DPA will be given, and the local convergence neighborhoods of the dominant poles will be characterized for both methods. Secondly, theoretical and numerical results will be presented that indicate that for DPA the basins of attraction of the dominant pole are larger than for two-sided RQI. The price for the better global convergence is only a few additional iterations, due to the asymptotically quadratic rate of convergence of DPA, against the cubic rate of two-sided RQI.

Key words. eigenvalues, eigenvectors, dominant poles, two-sided Rayleigh quotient iteration, dominant pole algorithm, Newton's method, rate of convergence, transfer function, modal model reduction

2.1 Introduction

The transfer function of a large-scale dynamical system often only has a small number of dominant poles compared to the number of state variables. The dominant behavior of the system can be captured by projecting the state-space on the subspace spanned by the eigenvectors corresponding to the dominant poles. This type of model reduction is known as modal approximation, see for instance [159]. The computation of the dominant poles, that are specific eigenvalues of the system matrix, and the corresponding modes, requires specialized eigenvalue methods.

In [91] Newton's method is used to compute a dominant pole of single-input single-output (SISO) transfer function: the Dominant Pole Algorithm (DPA). In two recent publications this algorithm is improved and extended to a robust and efficient method for the computation of the dominant poles, modes and modal approximates of large-scale SISO [123] (see Chapter 3) and MIMO transfer functions [122] (see Chapter 4).

This chapter is concerned with the convergence behavior of DPA. Firstly, DPA will be related to two-sided or generalized Rayleigh quotient iteration [107, 110]. A local convergence analysis will be given, showing the asymptotically quadratic rate of convergence. Furthermore, for systems with a symmetric state-space matrix, a characterization of the local convergence neighborhood of the dominant pole will be presented for both DPA and RQI. The results presented in this chapter are sharper than the results by Ostrowski [106, 107] and Beattie and Fox [16]. Secondly, theoretical and numerical results indicate that for DPA the basins of attraction of the most dominant poles are larger than for two-sided RQI. In practice, the asymptotically quadratic (DPA) instead of cubic rate (two-sided RQI) of convergence costs about two or three iterations.

The outline of this chapter is as follows. Definitions and properties of transfer functions and dominant poles, and further motivation are given in Section 2.2. The Dominant Pole Algorithm and its relation to two-sided Rayleigh quotient iteration are discussed in Section 2.3. In Section 2.4 the local convergence of DPA is analyzed. The basins of attraction of DPA and two-sided RQI are studied in Section 2.5. Section 2.6 concludes.

2.2 Transfer functions and poles

The motivation for this chapter comes from dynamical systems $(E, A, \mathbf{b}, \mathbf{c}, d)$ of the form

$$\left\{ \begin{array}{rcl} E\dot{\mathbf{x}}(t) & = & A\mathbf{x}(t) + \mathbf{b}u(t) \\ y(t) & = & \mathbf{c}^*\mathbf{x}(t) + du(t), \end{array} \right. \tag{2.2.1}$$

where $A, E \in \mathbb{R}^{n\times n}$, E may be singular, $\mathbf{b}, \mathbf{c}, \mathbf{x}(t) \in \mathbb{R}^n$, $u(t), y(t), d \in \mathbb{R}$. The vectors $\mathbf{b}$ and $\mathbf{c}$ are called the input, and output vector, respectively. The transfer function $H : \mathbb{C} \longrightarrow \mathbb{C}$ of (2.2.1) is defined as

$$H(s) = \mathbf{c}^*(sE - A)^{-1}\mathbf{b} + d. \tag{2.2.2}$$

The poles of transfer function (2.2.2) are a subset of the eigenvalues $\lambda_i \in \mathbb{C}$ of the matrix pencil (A, E). An eigentriplet $(\lambda_i, \mathbf{x}_i, \mathbf{y}_i)$ is composed of an eigenvalue λ_i of (A, E) and corresponding right and left eigenvectors $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{C}^n$ (identified by their components in $\mathbf{b}$ and $\mathbf{c}$):

$$\begin{array}{rcll} A\mathbf{x}_i & = & \lambda_i E\mathbf{x}_i, & \mathbf{x}_i \neq 0, \\ \mathbf{y}_i^* A & = & \lambda_i \mathbf{y}_i^* E, & \mathbf{y}_i \neq 0, \qquad (i = 1, \ldots, n). \end{array}$$

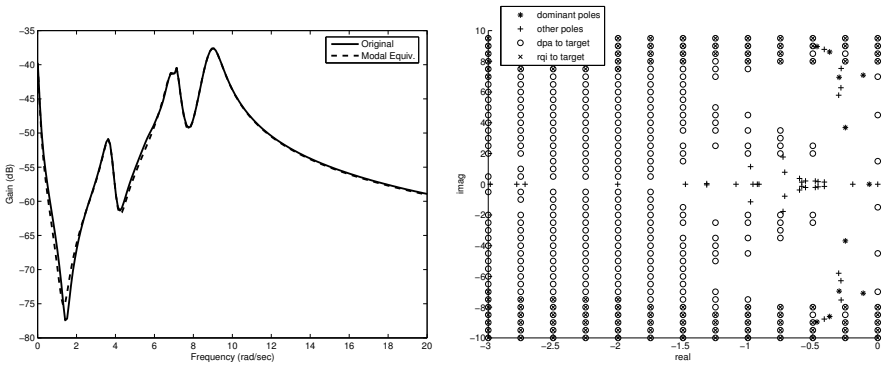


Figure 2.1: The left figure shows the Bode plot of the transfer function ($n = 66$ states) of the New England test system [91], together with the Bode plot of the $k = 11$th order modal equivalent, constructed by projecting the system onto the modes of the 6 most dominant poles, which may belong to complex conjugate pairs. The right figure shows part of the pole spectrum together with the initial shifts for which DPA (marked by circles) and two-sided RQI (x-es) converge to the most dominant pole $\lambda \approx -0.467 \pm 8.96i$.

Assuming that the pencil is nondefective, the right and left eigenvectors corresponding to finite eigenvalues can be scaled so that $\mathbf{y}_i^* E \mathbf{x}_i = 1$. Furthermore, it is well known that left and right eigenvectors corresponding to distinct eigenvalues are E-orthogonal: $\mathbf{y}_i^* E \mathbf{x}_j = 0$ for $i \neq j$. The transfer function $H(s)$ can be expressed as a sum of residues $R_i \in \mathbb{C}$ over the $\tilde{n} \leq n$ finite first order poles [78]:

$$H(s) = \sum_{i=1}^{\tilde{n}} \frac{R_i}{s - \lambda_i} + R_\infty + d, \tag{2.2.3}$$

where the residues R_i are

$$R_i = (\mathbf{c}^* \mathbf{x}_i)(\mathbf{y}_i^* \mathbf{b}),$$

and R_∞ is the constant contribution of the poles at infinity (often zero).

Although there are different indices of modal dominance [4, 68, 123, 159], the following will be used in this chapter.

Definition 2.2.1. *A pole λ_i of $H(s)$ with corresponding right and left eigenvectors $\mathbf{x}_i$ and $\mathbf{y}_i$ ($\mathbf{y}_i^* E \mathbf{x}_i = 1$) is called the dominant pole if $|R_i| > |R_j|$, for all $j \neq i$.*

More generally, a pole λ_i is called dominant if $|R_i|$ is not very small compared to $|R_j|$, for all $j \neq i$. A dominant pole is well observable and controllable in the transfer function. This can also be seen in the corresponding Bode-plot (see Figure 2.1), which is a plot of $|H(i\omega)|$ against $\omega \in \mathbb{R}$: peaks occur at frequencies ω close to the imaginary parts of the dominant poles of $H(s)$. An approximation of $H(s)$ that consists of $k < n$ terms with $|R_j|$ above some value, determines the effective transfer

function behavior [144] and is also known as transfer function modal equivalent:

$$H_k(s) = \sum_{j=1}^{k} \frac{R_j}{s - \lambda_j} + d.$$

The dominant poles are specific (complex) eigenvalues of the pencil (A, E) and usually form a small subset of the spectrum of (A, E). They can be located anywhere in the spectrum, see also Figure 2.1. The two algorithms to compute poles (eigenvalues) that will be discussed in this chapter, the Dominant Pole Algorithm (DPA) and two-sided Rayleigh Quotient Iteration (two-sided RQI), both start with an initial shift s_0, but behave notably differently: as can be seen in Figure 2.1, DPA converges to the most dominant pole for many more initial shifts than two-sided RQI (marked by circles and x-es, respectively). In Section 2.5 more of such figures will be presented and for all figures it holds: the more circles (compared to x-es), the better the performance of DPA over two-sided RQI. The typical behavior of DPA will be discussed in more detail in Sections 2.4 and 2.5.

Since the dominance of a pole is independent of d, without loss of generality $d = 0$ in the following.

2.3 The Dominant Pole Algorithm (DPA)

The poles of transfer function (2.2.2) are the $\lambda \in \mathbb{C}$ for which $\lim_{s\to\lambda} |H(s)| = \infty$. Consider now the function $G : \mathbb{C} \longrightarrow \mathbb{C}$

$$G(s) = \frac{1}{H(s)}.$$

For a pole λ of $H(s)$, $\lim_{s\to\lambda} G(s) = 0$. In other words, the poles are the roots of $G(s)$ and a good candidate to find these roots is Newton's method. This idea is the basis of the Dominant Pole Algorithm (DPA) [91] (and can be generalized to MIMO systems as well, see [93, 122]).

The derivative of $G(s)$ with respect to s is given by

$$G'(s) = -\frac{H'(s)}{H^2(s)}. \tag{2.3.1}$$

The derivative of $H(s)$ with respect to s is

$$H'(s) = -\mathbf{c}^*(sE - A)^{-1}E(sE - A)^{-1}\mathbf{b}. \tag{2.3.2}$$

Equations (2.3.1) and (2.3.2) lead to the following Newton scheme:

$$\begin{aligned} s_{k+1} &= s_k - \frac{G(s_k)}{G'(s_k)} \\ &= s_k + \frac{1}{H(s_k)}\frac{H^2(s_k)}{H'(s_k)} \\ &= s_k - \frac{\mathbf{c}^*(s_kE - A)^{-1}\mathbf{b}}{\mathbf{c}^*(s_kE - A)^{-1}E(s_kE - A)^{-1}\mathbf{b}}. \end{aligned} \tag{2.3.3}$$

The formula (2.3.3) was originally derived in [20]. Using $\mathbf{v}_k = (s_k E - A)^{-1}\mathbf{b}$ and $\mathbf{w}_k = (s_k E - A)^{-*}\mathbf{c}$, the Newton update (2.3.3) can also be written as the generalized two-sided Rayleigh quotient $\rho(\mathbf{v}_k, \mathbf{w}_k)$:

$$\begin{aligned} s_{k+1} &= s_k - \frac{\mathbf{c}^*(s_k E - A)^{-1}\mathbf{b}}{\mathbf{c}^*(s_k E - A)^{-1}E(s_k E - A)^{-1}\mathbf{b}} \\ &= \frac{\mathbf{c}^*(s_k E - A)^{-1}A(s_k E - A)^{-1}\mathbf{b}}{\mathbf{c}^*(s_k E - A)^{-1}E(s_k E - A)^{-1}\mathbf{b}} \\ &= \frac{\mathbf{w}_k^* A \mathbf{v}_k}{\mathbf{w}_k^* E \mathbf{v}_k}. \end{aligned}$$

An implementation of this Newton scheme is represented in Algorithm 2.1. It is also known as the Dominant Pole Algorithm [91].

Algorithm 2.1 Dominant Pole Algorithm (DPA)

INPUT: System $(E, A, \mathbf{b}, \mathbf{c})$, initial pole estimate s_0, tolerance $\epsilon \ll 1$
OUTPUT: Approximate dominant pole λ and corresponding right and left eigenvectors $\mathbf{x}$ and $\mathbf{y}$

1: Set $k = 0$
2: **while** not converged **do**
3: Solve $\mathbf{v}_k \in \mathbb{C}^n$ from $(s_k E - A)\mathbf{v}_k = \mathbf{b}$
4: Solve $\mathbf{w}_k \in \mathbb{C}^n$ from $(s_k E - A)^* \mathbf{w}_k = \mathbf{c}$
5: Compute the new pole estimate

$$s_{k+1} = s_k - \frac{\mathbf{c}^*\mathbf{v}_k}{\mathbf{w}_k^* E \mathbf{v}_k} = \frac{\mathbf{w}_k^* A \mathbf{v}_k}{\mathbf{w}_k^* E \mathbf{v}_k}$$

6: The pole $\lambda = s_{k+1}$ with $\mathbf{x} = \mathbf{v}_k / \|\mathbf{v}_k\|_2$ and $\mathbf{y} = \mathbf{w}_k / \|\mathbf{w}_k\|_2$ has converged if

$$\|A\mathbf{x} - s_{k+1} E \mathbf{x}\|_2 < \epsilon$$

7: Set $k = k + 1$
8: **end while**

The two linear systems that need to be solved in step 3 and 4 of Algorithm 2.1 can be efficiently solved using one LU-factorization $LU = s_k E - A$, by noting that $U^* L^* = (s_k E - A)^*$. In this chapter it will be assumed that an exact LU-factorization is available, although this may not always be the case for real-life examples, depending on the size and condition of the system. If an exact LU-factorization is not available, one has to use inexact Newton schemes, such as inexact Rayleigh Quotient Iteration and Jacobi-Davidson style methods [74, 140, 148], a topic that is studied in more detail in Chapter 3.

2.3.1 DPA and two-sided Rayleigh quotient iteration

The generalized two-sided Rayleigh quotient is defined as follows:

Definition 2.3.1. *The generalized two-sided Rayleigh quotient* $\rho(\mathbf{x}, \mathbf{y})$ *is given by [107, 110]* $\rho(\mathbf{x}, \mathbf{y}) \equiv \rho(\mathbf{x}, \mathbf{y}, A, E) \equiv \mathbf{y}^* A\mathbf{x}/\mathbf{y}^* E\mathbf{x}$*, provided* $\mathbf{y}^* E\mathbf{x} \neq 0$.

The two-sided Rayleigh quotient iteration [107, 110] is shown in Alg. 2.2. The only difference with DPA is that the right-hand sides in step 3 and 4 of Alg. 2.1 are kept fixed, while the right-hand sides in step 4 and 5 of Alg. 2.2 are updated every iteration.

Algorithm 2.2 Two-sided Rayleigh quotient iteration

INPUT: System $(E, A, \mathbf{b}, \mathbf{c})$, initial pole estimate s_0, tolerance $\epsilon \ll 1$
OUTPUT: Approximate eigenvalue λ and corresponding right and left eigenvectors $\mathbf{x}$ and $\mathbf{y}$

1: $\mathbf{v}_0 = (s_0 E - A)^{-1}\mathbf{b}$, $\mathbf{w}_0 = (s_0 E - A)^{-*}\mathbf{c}$, and $s_1 = \rho(\mathbf{v}_0, \mathbf{w}_0)$
2: Set $k = 1$, $\mathbf{v}_0 = \mathbf{v}_0/\|\mathbf{v}_0\|_2$, and $\mathbf{w}_0 = \mathbf{w}_0/\|\mathbf{w}_0\|_2$
3: **while** not converged **do**
4: Solve $\mathbf{v}_k \in \mathbb{C}^n$ from $(s_k E - A)\mathbf{v}_k = E\mathbf{v}_{k-1}$
5: Solve $\mathbf{w}_k \in \mathbb{C}^n$ from $(s_k E - A)^* \mathbf{w}_k = E^* \mathbf{w}_{k-1}$
6: Compute the new pole estimate

$$s_{k+1} = \rho(\mathbf{v}_k, \mathbf{w}_k) = \frac{\mathbf{w}_k^* A \mathbf{v}_k}{\mathbf{w}_k^* E \mathbf{v}_k}$$

7: Set $\mathbf{v}_k = \mathbf{v}_k/\|\mathbf{v}_k\|_2$ and $\mathbf{w}_k = \mathbf{w}_k/\|\mathbf{w}_k\|_2$
8: The pole $\lambda = s_{k+1}$ with $\mathbf{x} = \mathbf{v}_k$ and $\mathbf{y} = \mathbf{w}_k$ has converged if

$$\|A\mathbf{v}_k - s_{k+1} E \mathbf{v}_k\|_2 < \epsilon$$

9: Set $k = k + 1$
10: **end while**

While the use of the fixed right-hand sides in DPA drops the asymptotic convergence rate from cubic to quadratic, it is exactly this use of fixed right-hand sides that causes the typical better convergence to dominant poles, as will be shown later. In that light the quadratic instead of cubic local convergence, that in practice only makes a small difference in the number of iterations, is even more acceptable. Moreover, based on techniques in [16, 150] one can switch from DPA to two-sided RQI in the final phase of the process, to save some iterations. However, such techniques are not considered in this chapter, since the primary goal is to study the convergence behavior.

2.4 Local convergence analysis

The generalized two-sided Rayleigh quotient (Def. 2.3.1) has some well known basic properties, see [107, 110]:

- Homogeneity: $\rho(\alpha\mathbf{x}, \beta\mathbf{y}, \gamma A, \delta E) = (\gamma/\delta)\rho(\mathbf{x}, \mathbf{y}, A, E)$ for $\alpha, \beta, \gamma, \delta \neq 0$.
- Translation Invariance: $\rho(\mathbf{x}, \mathbf{y}, A - \alpha E, E) = \rho(\mathbf{x}, \mathbf{y}, A, E) - \alpha$.
- Stationarity (all directional derivatives are zero): $\rho = \rho(\mathbf{x}, \mathbf{y}, A, E)$ is stationary if and only if $\mathbf{x}$ and $\mathbf{y}$ are right and left eigenvectors of (A, E), respectively, with eigenvalue ρ and $\mathbf{y}^* E\mathbf{x} \neq 0$.

2.4.1 Asymptotically quadratic rate of convergence

In [110, p. 689] it is proved that the asymptotic convergence rate of two-sided RQI is cubic for nondefective matrices. Along the same lines it can be shown that the asymptotic convergence rate of DPA is quadratic. For the eigenvalues, this also follows from the fact that DPA is an exact Newton method, but for the corresponding left and right eigenvectors the following lemma is needed, which gives a useful expression for $(\rho_{k+1} - \lambda)$ (using $s_k \equiv \rho_k \equiv \rho(\mathbf{v}_k, \mathbf{w}_k, A, E)$ from now on).

Lemma 2.4.1. *Let* $\mathbf{x}$ *and* $\mathbf{y}$ *be right and left eigenvectors of* (A, E) *with eigenvalue* λ*, i.e.* $(A - \lambda E)\mathbf{x} = 0$ *and* $\mathbf{y}^*(A - \lambda E) = 0$*, and* $\mathbf{y}^* E\mathbf{x} = 1$*. Let* $\tau_k, \omega_k \in \mathbb{C}$ *be scaling factors so that the solutions* $\mathbf{v}_k$ *and* $\mathbf{w}_k$ *of*

$$(\rho_k E - A)\mathbf{v}_k = \tau_k \mathbf{b} \qquad \text{and} \qquad (\rho_k E - A)^* \mathbf{w}_k = \omega_k \mathbf{c} \tag{2.4.1}$$

are of the form

$$\mathbf{v}_k = \mathbf{x} + \mathbf{d}_k \qquad \text{and} \qquad \mathbf{w}_k = \mathbf{y} + \mathbf{e}_k, \tag{2.4.2}$$

where $\mathbf{y}^* E\mathbf{d}_k = \mathbf{e}_k^* E\mathbf{x} = 0$*. Then with* $\mathbf{u} \equiv (I - E\mathbf{x}\mathbf{y}^*)\frac{\mathbf{b}}{\mathbf{y}^*\mathbf{b}}$ *and* $\mathbf{z} \equiv (I - E^*\mathbf{y}\mathbf{x}^*)\frac{\mathbf{c}}{\mathbf{x}^*\mathbf{c}}$*, it follows that*

$$\mathbf{u} = (\rho_k - \lambda)^{-1}(\rho_k E - A)\mathbf{d}_k \perp \mathbf{y} \qquad \text{and} \qquad \mathbf{z} = (\rho_k - \lambda)^{-*}(\rho_k E - A)^* \mathbf{e}_k \perp \mathbf{x},$$

and with $\rho_{k+1} = \mathbf{w}_k^* A\mathbf{v}_k / (\mathbf{w}_k^* E\mathbf{v}_k)$*, one has that*

$$\rho_{k+1} - \lambda = (\rho_k - \lambda)\mu, \text{ where } \mu = \frac{\mathbf{e}_k^*\mathbf{u} + \mathbf{e}_k^* E\mathbf{d}_k}{1 + \mathbf{e}_k^* E\mathbf{d}_k}. \tag{2.4.3}$$

Note that $\mathbf{u}$ *and* $\mathbf{z}$ *do not change during the iteration.*

Proof. Substitution of (2.4.2) into (2.4.1) and multiplication from the left by $\mathbf{y}^*$ and $\mathbf{x}^*$, respectively, gives

$$\tau_k = \frac{\rho_k - \lambda}{\mathbf{y}^*\mathbf{b}} \qquad \text{and} \qquad \omega_k = \frac{(\rho_k - \lambda)^*}{\mathbf{x}^*\mathbf{c}}.$$

It follows that

$$(\rho_k E - A)\mathbf{d}_k \quad = \quad (\rho_k - \lambda)(I - E\mathbf{x}\mathbf{y}^*)\frac{\mathbf{b}}{\mathbf{y}^*\mathbf{b}} \equiv (\rho_k - \lambda)\mathbf{u} \perp \mathbf{y}$$

and

$$(\rho_k E - A)^*\mathbf{e}_k \quad = \quad (\rho_k - \lambda)^*(I - E^*\mathbf{y}\mathbf{x}^*)\frac{\mathbf{c}}{\mathbf{x}^*\mathbf{c}} \equiv (\rho_k - \lambda)^*\mathbf{z} \perp \mathbf{x},$$

where $\mathbf{u}$ and $\mathbf{z}$ are independent of the iteration. With $\rho_{k+1} = \mathbf{w}_k^* A\mathbf{v}_k/(\mathbf{w}_k^* E\mathbf{v}_k)$, it follows that

$$\rho_{k+1} - \lambda = \frac{\mathbf{w}_k^*(A - \lambda E)\mathbf{v}_k}{\mathbf{w}_k^* E\mathbf{v}_k} = \frac{\mathbf{e}_k^*(A - \lambda E)\mathbf{d}_k}{1 + \mathbf{e}_k^* E\mathbf{d}_k}.$$

Note that $\mathbf{e}_k^*(A-\lambda E)\mathbf{d}_k = \mathbf{e}_k^*(A-\rho_k E)\mathbf{d}_k+(\rho_k-\lambda)\mathbf{e}_k^* E\mathbf{d}_k = (\rho_k-\lambda)(\mathbf{e}_k^*\mathbf{u}+\mathbf{e}_k^* E\mathbf{d}_k)$, which shows (2.4.3). □

This lemma will be used in the proof of the following theorem, that shows the asymptotically quadratic rate of convergence of DPA, and expression (2.4.3) in particular will be used to derive the local convergence neighborhoods of DPA and RQI in Section 2.4.2.

Theorem 2.4.2. *Let* $\mathbf{x}$ *and* $\mathbf{y}$ *be right and left eigenvectors of* (A, E) *with eigenvalue* λ*, i.e.* $(A - \lambda E)\mathbf{x} = 0$ *and* $\mathbf{y}^*(A - \lambda E) = 0$*, and* $\mathbf{y}^* E\mathbf{x} = 1$*. Then* $\lim_{k\to\infty} \mathbf{v}_k = \mathbf{x}$ *and* $\lim_{k\to\infty} \mathbf{w}_k = \mathbf{y}$ *if and only if* $s_{k+1} = \rho_k = \rho(\mathbf{v}_k, \mathbf{w}_k)$ *approaches* λ*, and the convergence rate is asymptotically quadratic.*

Proof. The proof is an adaptation of the proofs in [110, p. 689] and [74, p. 150]. The main difference here is that for DPA the right-hand-sides of the linear systems are kept fixed during the iterations. Let the iterates $\mathbf{v}_k$ and $\mathbf{w}_k$, see lemma 2.4.1, be of the form

$$\mathbf{v}_k = \mathbf{x} + \mathbf{d}_k \qquad \text{and} \qquad \mathbf{w}_k = \mathbf{y} + \mathbf{e}_k,$$

where $\mathbf{y}^* E\mathbf{d}_k = \mathbf{e}_k^* E\mathbf{x} = 0$ and $\mathbf{y}^* E\mathbf{x} = 1$. Put $\mathbf{d}_k = (\rho_k-\lambda)\tilde{\mathbf{d}}_k$ with $(\rho_k E-A)\tilde{\mathbf{d}}_k = \mathbf{u}$, and $\mathbf{e}_k = (\rho_k-\lambda)^*\tilde{\mathbf{e}}_k$ with $(\rho_k E-A)^*\tilde{\mathbf{e}}_k = \mathbf{z}$. Since $(\lambda E-A)^{-1} : \mathbf{y}^\perp \to (E^*\mathbf{y})^\perp$ is bounded on $\mathbf{y}^\perp$ and $(\lambda E-A)^{-*} : \mathbf{x}^\perp \to (E\mathbf{x})^\perp$ is bounded on $\mathbf{x}^\perp$, it follows that as $\rho_k \to \lambda$, then

$$\begin{aligned} \|\mathbf{d}_k\| &= \|(\rho_k - \lambda)(\rho_k E - A)^{-1}\mathbf{u}\| \\ &= |\rho_k - \lambda|\|((\lambda E - A)|_{(E^*\mathbf{y})^\perp})^{-1}\|\|\mathbf{u}\| + O((\rho_k - \lambda)^2), \end{aligned} \tag{2.4.4}$$

and similarly

$$\begin{aligned} \|\mathbf{e}_k\| &= \|(\rho_k - \lambda)^*(\rho_k E - A)^{-*}\mathbf{z}\| \\ &= |\rho_k - \lambda|\|((\lambda E - A)|_{(E\mathbf{x})^\perp})^{-*}\|\|\mathbf{z}\| + O((\rho_k - \lambda)^2), \end{aligned} \tag{2.4.5}$$

and $\mathbf{d}_k$ and $\mathbf{e}_k$, and $\tilde{\mathbf{d}}_k$ and $\tilde{\mathbf{e}}_k$, are bounded. Hence, $\rho_k \to \lambda$ if and only if $\mathbf{v}_k \to \mathbf{x}$ and $\mathbf{w}_k \to \mathbf{y}$.

To prove the asymptotically quadratic rate of convergence, first note that

$$\rho_{k+1} - \lambda = \rho(\mathbf{v}_k, \mathbf{w}_k) = (\rho_k - \lambda)^2 \frac{\tilde{\mathbf{e}}_k^*(A - \lambda E)\tilde{\mathbf{d}}_k}{1 + (\rho_k - \lambda)^2 \tilde{\mathbf{e}}_k^* E \tilde{\mathbf{d}}_k},$$

and hence

$$|\rho_{k+1} - \lambda| = (\rho_k - \lambda)^2 |\tilde{\mathbf{e}}_k^*(A - \lambda E)\tilde{\mathbf{d}}_k| + O((\rho_k - \lambda)^4). \tag{2.4.6}$$

Let $\kappa(A|_{x^\perp})$ denote the condition number of A restricted to $x^\perp$. By (2.4.4) and (2.4.6), it follows that as $k \to \infty$, then

$$\begin{aligned} \|\mathbf{x} - \mathbf{x}_{k+1}\| &= \|(\rho_{k+1} - \lambda)\tilde{\mathbf{d}}_{k+1}\| \\ &\leq |\rho_k - \lambda|^2 \|\tilde{\mathbf{d}}_k\| \|\tilde{\mathbf{e}}_k\| (\kappa((A - \lambda E)|_{(E^*\mathbf{y})^\perp}) \|\mathbf{u}\|) + O((\rho_k - \lambda)^3), \end{aligned}$$

and similarly, by (2.4.5) and (2.4.6),

$$\begin{aligned} \|\mathbf{y} - \mathbf{y}_{k+1}\| &= \|(\rho_{k+1} - \lambda)\tilde{\mathbf{e}}_{k+1}\| \\ &\leq |\rho_k - \lambda|^2 \|\tilde{\mathbf{e}}_k\| \|\tilde{\mathbf{d}}_k\| (\kappa((A - \lambda E)|_{(E\mathbf{x})^\perp}) \|\mathbf{z}\|) + O((\rho_k - \lambda)^3), \end{aligned}$$

which proves the asymptotically quadratic convergence. □

2.4.2 Convergence neighborhood

In this section it will be assumed that A is a symmetric matrix. In [106] Ostrowski characterizes the convergence neighborhood of the iteration

$$(A - \rho_k I)\mathbf{v}_k = \tau_k \mathbf{b}, \quad k = 0, 1, \ldots, \tag{2.4.7}$$

for symmetric matrices A, where ρ_0 arbitrary, $\rho_{k+1} = \rho(A, \mathbf{v}_k)$ $(k > 0)$ and τ_k is a scalar so that $\|\mathbf{v}_k\|_2 = 1$. It can be seen that DPA for symmetric matrices (with $E = I$, $\mathbf{b} = \mathbf{c}$),

$$(\rho_k I - A)\mathbf{v}_k = \tau_k \mathbf{b}, \quad k = 0, 1, \ldots, \tag{2.4.8}$$

is similar and hence Ostrowski's approach can be used to characterize the local convergence neighborhood of DPA for symmetric matrices A with $\mathbf{c} = \mathbf{b} = (b_1, \ldots, b_n)^T$. In fact, a larger convergence neighborhood of DPA will be derived here. This result gives insight in the typical convergence behavior of DPA.

Since the two-sided Rayleigh quotient and (2.4.7, 2.4.8) are invariant under unitary similarity transforms, without loss of generality A will be a diagonal matrix $\mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ with $\lambda_1 < \ldots < \lambda_n$. Note that $R_j = b_j^2$ and the λ_j with $j = \mathrm{argmax}_j(b_j^2)$ is the dominant pole. The main results of this chapter, sharp bounds for the convergence neighborhoods of DPA and RQI, respectively, are stated in theorem 2.4.3 and theorem 2.4.4, respectively. The proofs are given in Section 2.4.2.

Theorem 2.4.3. *Let $(\lambda, \mathbf{x})$ be an eigenpair of A. In the DPA iteration for A and $\mathbf{b}$ with initial shift ρ_0, let $\mathbf{v}_k$ and τ_k be such that*

$$\|\mathbf{v}_k\| = 1, \quad (\rho_k I - A)\mathbf{v}_k = \tau_k \mathbf{b}, \text{ with } \rho_{k+1} \equiv \mathbf{v}_k^* A \mathbf{v}_k, \quad (k \geq 0),$$

and put $\gamma = \min_{\lambda_i \neq \lambda} |\lambda_i - \lambda|$. *If*

$$\alpha_{dpa} \equiv \frac{|\rho_0 - \lambda|}{\gamma} \leq \frac{1}{1+\zeta^2} \text{ with } \zeta \equiv \tan \angle(\mathbf{x}, \mathbf{b}), \tag{2.4.9}$$

then, with $c \equiv \cos \angle(\mathbf{x}, \mathbf{b})$, *it follows that*

$$\rho_k \to \lambda \quad \text{and} \quad \frac{|\rho_{k+1} - \lambda|}{c^2 \gamma} \leq \left(\frac{|\rho_k - \lambda|}{c^2 \gamma} \right)^2 \qquad (k \geq 0).$$

The bound given by Ostrowski [106, p. 235 (eqn. (19))] is

$$|\rho_k - \lambda| < \frac{\gamma}{2} \min\left(\frac{b^2}{2(1-b^2)}, 1\right),$$

and it is clear that the neighborhood in theorem 2.4.3 is larger.

In [106, p. 239] also the convergence neighborhood of standard RQI,

$$(A - \rho_k I)\mathbf{v}_k = \tau_k \mathbf{x}_{k-1}, \quad k = 0, 1, \ldots \tag{2.4.10}$$

where $\mathbf{x}_{-1}$ arbitrary, $\rho_{k+1} = \rho(A, \mathbf{v}_k)$ $(k > 0)$ and τ_k is a scalar so that $\|\mathbf{v}_k\|_2 = 1$, is derived. Here a sharper bound is derived.

Theorem 2.4.4. *Let* $(\lambda, \mathbf{x})$ *be an eigenpair of* A. *In the RQI iteration for* A *and* $\mathbf{b}$ *with initial shift* ρ_0 *and* $\mathbf{x}_{-1} = \mathbf{b}$, *let* $\mathbf{v}_k$ *and* τ_k *be such that*

$$\|\mathbf{v}_k\| = 1, \quad (\rho_k I - A)\mathbf{v}_k = \tau_k \mathbf{x}_{k-1}, \text{ with } \rho_{k+1} \equiv \mathbf{v}_k^* A \mathbf{v}_k, \quad (k \geq 0),$$

and put $\gamma = \min_{\lambda_i \neq \lambda} |\lambda_i - \lambda|$. *If*

$$\alpha_{rqi} \equiv \frac{|\rho_0 - \lambda|}{\gamma} \leq \frac{1}{1+\zeta} \text{ with } \zeta \equiv \tan \angle(\mathbf{x}, \mathbf{b}), \tag{2.4.11}$$

then $|\rho_1 - \lambda| < \gamma/2$, *and*

$$\rho_k \to \lambda \quad \text{and} \quad \frac{|\rho_{k+1} - \lambda|}{\gamma - |\rho_{k+1} - \lambda|} \leq \left(\frac{|\rho_k - \lambda|}{\gamma - |\rho_k - \lambda|} \right)^2 \qquad (k > 0).$$

In particular, the results of theorem 2.4.3 and 2.4.4 are sharp in the sense that if, in the two-dimensional case, condition (2.4.9) (condition (2.4.11)) is not fulfilled for λ_1, then it is fulfilled for λ_2. This follows from the fact that if $\alpha_0^{(1)} = |\rho_0 - \lambda_1|/\gamma$, then $\alpha_0^{(2)} = 1 - \alpha_0^{(1)}$, and $\zeta_0^{(1)} = 1/\zeta_0^{(2)}$.

In [16, Thm. 1] it is shown that, with $\gamma_b = \beta - \alpha$ a known gap in the spectrum of A (for instance, $\gamma = \min_{\lambda_i \neq \lambda} |\lambda - \lambda_i|$), if $\rho_1 < (\alpha+\beta)/2$ *and* $\|\mathbf{r}_1\| = \|A\mathbf{x}_0 - \rho_1 \mathbf{x}_0\| \leq \gamma_b$, then $\rho_k < (\alpha+\beta)/2$ for $k \geq 1$, and similarly for the case $\rho_1 > (\alpha+\beta)/2$. It can be shown that the conditions of this theorem imply the second step of theorem 2.4.4. On the other hand, the conditions of theorem 2.4.4 do not imply the conditions of [16, Thm. 1]. To see this, consider the two-dimensional example $A = \mathrm{diag}(-1, 1)$.

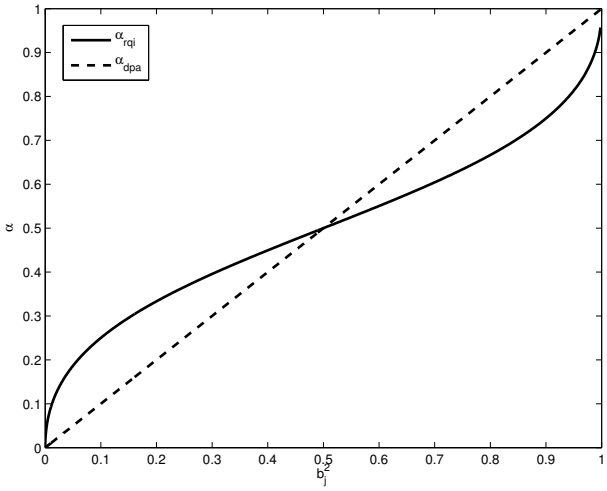


Figure 2.2: Bounds of the local convergence neighborhood for DPA (dashed) and best-case RQI (solid). If $|\lambda_j - \rho_k| < \alpha\gamma$, with $\gamma = \min_{i \neq j} |\lambda_i - \lambda_j|$, there is convergence to λ_j.

With $\rho_0 = 0.01$, $\mathbf{x}_{-1} = \mathbf{b} = [\sqrt{2}/2, \sqrt{2}/2]$ and $\mathbf{x}_0 = (A - \rho_0 I)^{-1}\mathbf{x}_{-1}$, it follows that $|\lambda - \rho_0| < 1$ and condition (2.4.11) is satisfied, while $\|\mathbf{r}_1\| = \|A\mathbf{x}_0 - \rho_1\mathbf{x}_0\| \approx 1.03 > 1$. Hence, the result in theorem 2.4.4 is sharper.

In Figure 2.2, α_{dpa} and α_{rqi}, see equations (2.4.9) and (2.4.11), respectively, are plotted for $0 < b_j^2 < 1$, $\|\mathbf{b}\|_2 = 1$. As b_j^2 increases, i.e. as mode j becomes more dominant, both local convergence neighborhoods increase and $\alpha \to 1$, while the bound for the DPA neighborhood is larger for $b^2 > 1/2$, or $\angle(\mathbf{x}, \mathbf{b}) < 45°$.

The price one has to pay for the cubic convergence, is the smaller local convergence neighborhood of the dominant pole, as it becomes more dominant, for RQI. While DPA emphasizes the dominant mode every iteration by keeping the right-hand side fixed, RQI only takes advantage of this in the first iteration, and for initial shifts too far from the dominant pole, the dominant mode may be damped out from the iterates $\mathbf{v}_k$. In that sense, RQI is closer to the inverse power method or inverse iteration, which converges to the eigenvalue closest to the shift, while DPA takes advantage of the information in the right-hand side $\mathbf{b}$.

Because the results are in fact lower bounds for the local convergence neighborhood, theoretically speaking no conclusions can be drawn about the global basins of attraction. But the results strengthen the intuition that for DPA the basin of attraction of the dominant pole is larger than for RQI.

Proofs of theorem 2.4.3 and theorem 2.4.4

The following two lemmas provide expressions and bounds that are needed for the proofs of theorem 2.4.3 and theorem 2.4.4.

Lemma 2.4.5. *Let* $\mathbf{x}$ *be an eigenvector of* $A = A^T$ *with eigenvalue* λ *with* $\|\mathbf{x}\| = 1$, *and let* $\tau_k \in \mathbb{R}$ *be a scaling factor so that the solution* $\mathbf{v}_k$ *of*

$$(\rho_k I - A)\mathbf{v}_k = \tau_k \mathbf{b}$$

is of the form

$$\mathbf{v}_k = \mathbf{x} + \mathbf{d}_k, \tag{2.4.12}$$

where $\mathbf{x}^*\mathbf{d}_k = 0$, *and let* $\mathbf{z} = (\rho_k E - A)\mathbf{d}_k$. *Then* $\rho_{k+1} = \mathbf{v}_k^* A \mathbf{v}_k / (\mathbf{v}_k^* \mathbf{v}_k)$ *satisfies*

$$\rho_{k+1} - \lambda = (\rho_k - \lambda)\mu,$$

where

$$\mu = \frac{\mathbf{d}_k^* \mathbf{z} + \mathbf{d}_k^* \mathbf{d}_k}{1 + \mathbf{d}_k^* \mathbf{d}_k}. \tag{2.4.13}$$

Proof. The result follows from lemma 2.4.1, by noting that $A = A^T$ and $E = I$. □

Lemma 2.4.6. *Under the assumptions of lemma 2.4.5, put* $\gamma = \min_{\lambda_i \neq \lambda} |\lambda_i - \lambda|$, $c = \cos\angle(\mathbf{x}, \mathbf{b})$, $\zeta = \|\mathbf{z}\|$, $\alpha_k = \frac{|\rho_k - \lambda|}{\gamma}$, *and* $\tilde{\alpha}_k = \alpha_k/(1 - \alpha_k)$. *The following statements hold:*

$$\zeta_{k+1} \equiv \|\mathbf{d}_k\| \leq \frac{\alpha_k}{1 - \alpha_k}\zeta. \tag{2.4.14}$$

If $\alpha_k \leq c = 1/\sqrt{1 + \zeta^2}$, *then*

$$|\mu| \leq \frac{\tilde{\alpha}_k + \tilde{\alpha}_k^2}{1 + \tilde{\alpha}_k^2 \zeta^2}\zeta^2 = \frac{\alpha_k \zeta^2}{(1 - \alpha_k)^2 + \alpha_k^2 \zeta^2}, \tag{2.4.15}$$

$$|\mu| \leq 1 \text{ if } \tilde{\alpha}_k \zeta^2 \Leftrightarrow \alpha_k \leq \frac{1}{1 + \zeta^2} = c^2, \tag{2.4.16}$$

$$\alpha_{k+1} \leq \alpha_k |\mu| \leq \alpha_k^2 (1 + \zeta^2), \tag{2.4.17}$$

$$\text{and } \tilde{\alpha}_{k+1} \equiv \frac{\alpha_{k+1}}{1 - \alpha_{k+1}} \leq (\tilde{\alpha}_k \zeta)^2. \tag{2.4.18}$$

Proof. Put $\zeta_k = \|\mathbf{d}_k\|$. Then by (2.4.13)

$$|\mu| \leq \phi(\zeta_k) \text{ where } \phi(\tau) \equiv \frac{\zeta\tau + \tau^2}{1 + \tau^2} \quad (\tau \in \mathbb{R}).$$

The function ϕ is increasing on $[0, \tau_{\max}]$, where $\tau_{\max} = (1 + \sqrt{1 + \zeta^2})/\zeta$, or, using $c \equiv \cos\angle(\mathbf{x}, \mathbf{b}) = 1/\sqrt{1 + \zeta^2}$, $\tau_{\max} = \sqrt{(1 + c)/(1 - c)}$, and $0 \leq \phi \leq \frac{1+c}{2c}$ on $(0, \infty)$.

Since $\|(A-\rho_k)^{-1}|_{\mathbf{x}^\perp}\| \leq |1/(\gamma - |\lambda - \rho_k|)|$, it follows that

$$\zeta_k \equiv \|\mathbf{d}_k\| \leq |\rho_k - \lambda| \|(A-\rho_k)^{-1}|_{\mathbf{x}^\perp}\| \|\mathbf{z}\| \leq \frac{|\rho_k - \lambda|}{|\gamma - |\rho_k - \lambda||} = \frac{\alpha}{1-\alpha}\zeta,$$

which proves (2.4.14). Statement (2.4.15) now follows from the observation that $\alpha_k\zeta/(1-\alpha) \leq \tau_{\max}$ if and only if $\alpha_k \leq 1/\sqrt{1+\zeta^2} = c$. Statement (2.4.16) follows readily from statement (2.4.14) and the definition $\tilde{\alpha}_k = \alpha_k/(1-\alpha_k)$.

For statement (2.4.17), first note that $(1-\alpha_k)^2 + \alpha_k^2\zeta^2 \geq \zeta^2/(1+\zeta^2)$ for all $\alpha_k \geq 0$, and therefore $|\mu| \leq \alpha_k(1+\zeta^2)$. Hence, with $\alpha_{k+1} \equiv |\rho_{k+1} - \lambda|/\gamma$, inequality (2.4.17) follows by (2.4.13).

Finally, statement (2.4.18) follows the fact that (2.4.15) and (2.4.18) imply $\alpha_{k+1} \leq (\tilde{\alpha}_k\zeta)^2/(1+(\tilde{\alpha}_k\zeta)^2)$. □

Note that it is essential that the function ϕ is increasing, since this allows to use upper bound (2.4.14) also to handle the denominator in (2.4.13), leading to (2.4.15).

In the two-dimensional case, the estimate in (2.4.15) is sharp (equality), since both $\mathbf{z}$ and $\mathbf{d}_k$ are in the same direction (orthogonal to $\mathbf{x}$). Furthermore, in statement 2, $|\mu| \leq 1$ if and only if $\tilde{\alpha}_k\zeta^2 \leq 1$.

Proof of theorem 2.4.3. Note that ζ is the same in all iterations, and recall that $\alpha_k \equiv |\rho_k - \lambda|/\gamma$. Since $c^2 = 1/(1+\zeta^2)$, condition (2.4.9) implies $\alpha_0(1+\zeta^2) < 1$, and by induction and (2.4.17) of lemma 2.4.6, $\alpha_k\sqrt{1+\zeta^2} \leq \alpha_k(1+\zeta^2) < 1$. Again by (2.4.17) of lemma 2.4.6, it follows that

$$\alpha_{k+1}(1+\zeta^2) \leq (\alpha_k(1+\zeta^2))^2,$$

which implies the quadratic convergence. □

Note that result (2.4.18) implies $\tilde{\alpha}_{k+1}\zeta^2 \leq (\tilde{\alpha}_k\zeta^2)^2$, which guarantees quadratic convergence as soon as $\tilde{\alpha}_0\zeta^2 < 1$. This condition is equivalent to $\alpha_0 < 1/(1+\zeta^2)$, the condition (2.4.9) of the theorem.

Proof of theorem 2.4.4. Note that $\zeta_k = \tan\angle(\mathbf{x}, \mathbf{v}_k)$ changes every iteration, and recall that $\alpha_k \equiv |\rho_k - \lambda|/\gamma$, and $\tilde{\alpha}_k = \alpha_k/(1-\alpha_k)$. Condition (2.4.11) implies $\alpha_0 < 1/(1+\zeta)$, or, equivalently, $\tilde{\alpha}_0\zeta_0 < 1$. By (2.4.18) it follows that $\alpha_1 < 1/2$, or, equivalently, $|\rho_1 - \lambda| < \gamma/2$. Since $\tilde{\alpha}_k\zeta_k < 1$ implies that $\alpha_k < 1/\sqrt{1+\zeta_k^2}$, results (2.4.14) and (2.4.18) of lemma 2.4.6 can be applied to obtain

$$\tilde{\alpha}_{k+1} \leq (\tilde{\alpha}_k\zeta_k)^2 \qquad \text{and} \qquad \zeta_{k+1} \leq \tilde{\alpha}_k\zeta_k,$$

if $\tilde{\alpha}_k\zeta_k < 1$. It follows that $\zeta_{k+2} \leq \tilde{\alpha}_{k+1}\zeta_{k+1} \leq (\tilde{\alpha}_k\zeta_k)^3 < 1$. Therefore, since $\tilde{\alpha}_0\zeta_0 < 1$, the sequences $(\tilde{\alpha}_k)$ and (ζ_k) converge dominated cubically, and, for $k > 0$, $\tilde{\alpha}_{k+1} \leq \tilde{\alpha}_k^2$. □

2.4.3 General systems

Theorems 2.4.3 and 2.4.4 can readily be generalized for normal matrices, but it is difficult to obtain such bounds for general matrices without making specific assumptions. To see this, note that it is difficult to give sharp bounds for (2.4.3) in lemma 2.4.1. However, the following theorem states that DPA is invariant under certain transformations and helps in getting more insight in DPA for general, nondefective systems $(E, A, \mathbf{b}, \mathbf{c})$.

Theorem 2.4.7. *Let (A, E) be a nondefective matrix pencil, and let $X, Y \in \mathbb{C}^{n\times n}$ be of full rank. If DPA$(E, A, \mathbf{b}, \mathbf{c}, s_0)$ produces the sequence $(\mathbf{v}_k, \mathbf{w}_k, s_{k+1})$, then DPA$(Y^*EX, Y^*AX, Y^*\mathbf{b}, X^*\mathbf{c}, s_0)$ produces the sequence $(X^{-1}\mathbf{v}_k, Y^{-1}\mathbf{w}_k, s_{k+1})$, and vice versa.*

Proof. If $\mathbf{v} = \mathbf{v}_k$ is the solution of

$$(sE - A)\mathbf{v} = \mathbf{b},$$

then $\tilde{\mathbf{v}} = \tilde{\mathbf{v}}_k = X^{-1}\mathbf{v}$ is the solution of

$$(sY^*EX - Y^*AX)\tilde{\mathbf{v}} = Y^*\mathbf{b},$$

and vice versa. Similar relations hold for $\mathbf{w} = \mathbf{w}_k$ and $\tilde{\mathbf{w}} = \tilde{\mathbf{w}}_k = Y^{-1}\mathbf{w}$. Noting that

$$s_{k+1} = \frac{\tilde{\mathbf{w}}^*Y^*AX\tilde{\mathbf{v}}}{\tilde{\mathbf{w}}^*Y^*EX\tilde{\mathbf{v}}} = \frac{\mathbf{w}^*A\mathbf{v}}{\mathbf{w}^*E\mathbf{v}} = s_{k+1}.$$

completes the proof. □

Let W and V have as their columns the left and right eigenvectors of (A, E), respectively, i.e. $AV = EV\Lambda$ and $W^*A = \Lambda W^*E$, with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. Furthermore, let W and V be scaled so that $W^*EV = \Delta$, where Δ is a diagonal matrix with $\delta_{ii} = 1$ for finite λ_i and $\delta_{ii} = 0$ for $|\lambda_i| = \infty$. According to theorem 2.4.7, DPA$(E, A, \mathbf{b}, \mathbf{c})$ and DPA$(\Delta, \Lambda, W^*\mathbf{b}, V^*\mathbf{c})$ produce the same pole estimates s_k. In $\tilde{\mathbf{b}} = W^*\mathbf{b}$ and $\tilde{\mathbf{c}} = V^*\mathbf{c}$, the new right-hand sides, one recognizes the contributions to the residues $R_i = \tilde{\mathbf{c}}_i\tilde{\mathbf{b}}_i = (\mathbf{c}^*\mathbf{x}_i)(\mathbf{y}_i^*\mathbf{b})$. The more dominant pole λ_i is, the larger the corresponding coefficients $\tilde{\mathbf{b}}_i$ and $\tilde{\mathbf{c}}_i$ are, and, since (Λ, Δ) is a diagonal pencil, the larger the chance that DPA converges to the unit vectors $\tilde{\mathbf{x}} = \mathbf{e}_i$ and $\tilde{\mathbf{y}} = \mathbf{e}_i$, that correspond to the right and left eigenvectors $\mathbf{x}_i = V\mathbf{e}_i$ and $\mathbf{y}_i = W\mathbf{e}_i$, respectively.

As observed earlier, DPA emphasizes the dominant mode every iteration by keeping the right-hand sides fixed, and thereby can be expected to enlarge the convergence neighborhood also for general systems, compared to two-sided RQI. In practice, the quadratic instead of cubic rate of local convergence costs at most 2 or 3 iterations. Numerical experiments confirm that the basins of attraction of the dominant eigenvalues are larger for DPA, as will be discussed in the following section.

2.5 Basins of attraction and typical convergence behavior

It is not straightforward to characterize the global convergence of DPA, even not for symmetric matrices (see [106, p. 236-237]). Basins of attraction of RQI in the three-dimensional case are studied in [3, 15, 109], while in [16, 150] local convergence neighborhoods are described. Because the DPA residuals $\mathbf{r}_k = (A - \rho_k I)\mathbf{b}$ are not monotonically decreasing (in contrast to the inverse iteration residuals $\mathbf{r}_k = (A - \sigma I)\mathbf{v}_k$ and the RQI residuals $\mathbf{r}_k = (A - \rho_k I)\mathbf{v}_k$, see [16, 109, 110]), it is not likely that similar results can be obtained for DPA. Numerical experiments, however, may help to get an idea of the typical convergence behavior of DPA and may show why DPA is to be preferred over two-sided RQI for the computation of dominant poles.

An unanswered question is how to choose the initial shift of DPA. An obvious choice is the two-sided Rayleigh quotient $s_0 = (\mathbf{c}^* A\mathbf{b})/(\mathbf{c}^* E\mathbf{b})$. This choice will work in the symmetric case $A = A^*, E = I, \mathbf{c} = \mathbf{b}$. In the general nonsymmetric case this choice will not always be possible: the vectors $\mathbf{b}$ and $\mathbf{c}$ are often very sparse (only $O(1)$ nonzero entries) and moreover, it may happen that $\mathbf{c}^* E\mathbf{b} = 0$. In that case the initial shift should be based on heuristics. For two-sided RQI, an obvious choice is to take as the initial vectors $\mathbf{v}_0 = \mathbf{b}$ and $\mathbf{w}_0 = \mathbf{c}$, but similarly, if $\mathbf{w}_0^* E\mathbf{v}_0 = 0$, this fails. Therefore, in the following experiments an initial shift s_0 will be chosen and the (normalized) initial vectors for two-sided RQI are $\mathbf{v}_0 = (A - s_0 E)^{-1}\mathbf{b}$ and $\mathbf{w}_0 = (A - s_0 E)^{-1}\mathbf{c}$, see Alg. 2.2.

All experiments were executed in Matlab 7 [151]. The criterion for convergence was $\|A\mathbf{v}_k - s_{k+1} E\mathbf{v}_k\|_2 < 10^{-8}$.

2.5.1 Three-dimensional symmetric matrices

Because RQI and DPA are shift and scaling invariant, the region of all 3×3 symmetric matrices can be parametrized by $A = \mathrm{diag}(-1, s, 1)$, with $0 \leq s < 1$ due to symmetry (see [109]). In order to compute the regions of convergence of RQI and DPA (as defined in (2.4.7, 2.4.8)), the algorithms are applied to A for initial shifts in the range $(-1, 1)\backslash\{s\}$, with $\mathbf{c} = \mathbf{b} = (b_1, b_2, b_3)^T$, where $0 < b_2 \leq 1$ and $b_1 = b_3 = \sqrt{(1 - b_2^2)/2}$. In Figure 2.3 the results are shown for $s = 0$ and $s = 0.8$. The intersections $\rho = \rho_{\lambda_1}$ and $\rho = \rho_{\lambda_3}$ at $b_2 = b$ with the borders define the convergence regions: for $-1 \leq \rho_0 < \rho_{\lambda_1}$ there is convergence to $\lambda_1 = -1$, for $\rho_{\lambda_1} \leq \rho_0 < \rho_{\lambda_3}$ there is convergence to $\lambda_2 = s$, while for $\rho_{\lambda_3} \leq \rho_0 \leq 1$ there is convergence to $\lambda_3 = 1$.

For the case $s = 0$ it can be observed that (see vertical lines) for $0 \leq b_2 \lesssim 0.5$, the convergence region to the dominant extremal eigenvalues is larger for DPA. For $0.5 \lesssim b_2 \leq 1/\sqrt{3} \approx 0.577$, the point at which λ_2 becomes dominant, the convergence region of RQI is larger. However, for $b_2 \gtrsim 0.5$, the convergence region of λ_2 is clearly larger for DPA. Note also that the theoretical (lower bound $\alpha_{dpa}\gamma$ of the) local convergence neighborhood for DPA (Thm. 2.4.3) is even larger than

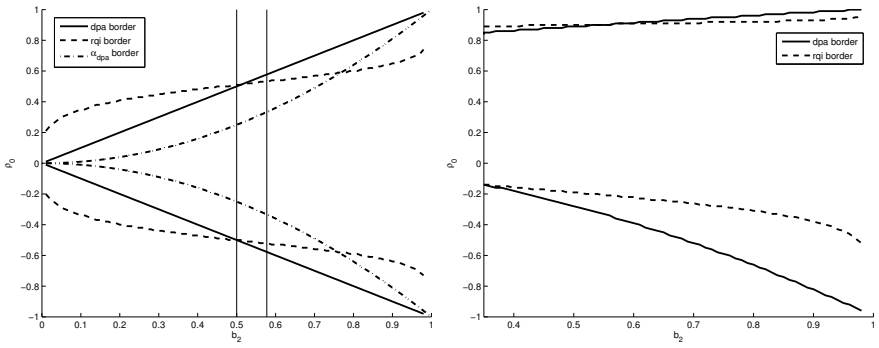


Figure 2.3: Convergence regions for DPA (solid borders) and RQI (dashed), and the theoretical DPA border (dash-dot, see Thm. 2.4.3), for the matrix $A = \mathrm{diag}(-1, s, 1)$, for $s = 0$ (left) and $s = 0.8$. The regions of convergence to $\lambda_2 = s$ for DPA and RQI respectively are enclosed between the lower and upper borders of DPA and RQI respectively. The regions of convergence to $\lambda_1 = -1$ ($\lambda_3 = 1$) are below (above) the lower (upper) border.

the practical convergence neighborhood of two-sided RQI for $b_2 \gtrsim 0.8$.

A similar observation can be made for the case $s = 0.8$. There, due to the decentralized location of λ_2, the figure is not symmetric and the region of convergence of λ_2 is clearly larger for DPA. For $0 \leq b_2 \lesssim 0.35$, DPA and RQI appear to be very sensitive to the initial shift. While the convergence region for λ_1 was similar to the case $s = 0$, convergence for $-0.1 \lesssim \rho_0 \lesssim 0.8$ was irregular in the sense that for initial shifts in this interval both λ_2 and λ_3 could be computed; hence the regions are only shown for $b_2 \gtrsim 0.35$. Because the theoretical lower bounds are much smaller, since $d = \min_{i \neq j} |\lambda_i - \lambda_j| = 0.2$, and make the figure less clear, they are not shown (the theoretical DPA border still crosses the practical two-sided RQI border around $b_2 \approx 0.9$).

It is almost generic, that apart from a small interval of values of b_2, the area of convergence of the dominant eigenvalue is larger for DPA than for RQI. The following example discusses a large-scale general system.

2.5.2 A large-scale example

This example is a test model of the Brazilian Interconnect Power System (BIPS) [123, 122]. The sparse matrices A and E are of dimension $n = 13,251$ and E is singular. The input and output vectors $\mathbf{b}$ and $\mathbf{c}$ only have one nonzero entry and furthermore $\mathbf{c}^* E \mathbf{b} = 0$; the choice $\mathbf{v}_0 = \mathbf{b}$ and $\mathbf{w}_0 = \mathbf{c}$ is not practical, see the beginning of this section. The pencil (A, E) is non-normal and the most dominant poles appear in complex conjugate pairs. It is not feasible to determine the converge regions for the entire complex plane, but the convergence behavior in the neighborhood of a dominant pole can be studied by comparing the found poles for a number of initial shifts in the neighborhood of the pole, for both DPA and

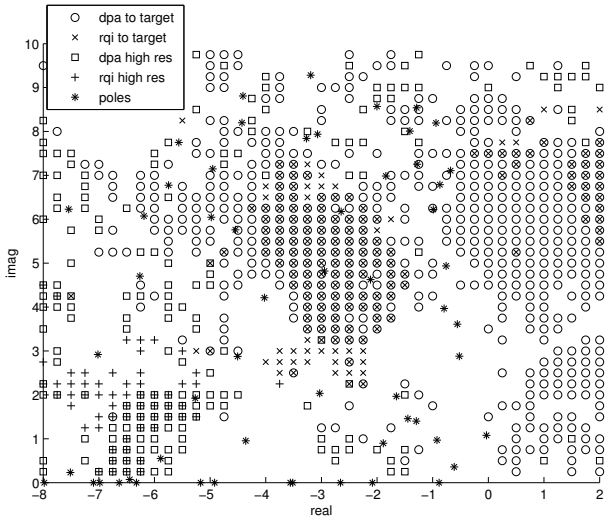


Figure 2.4: Convergence regions for DPA and two-sided RQI, for the example of Section 2.5.2. The center of the domain is the pole $\lambda \approx -2.9 \pm 4.8i$ with residue norm $|R| \approx 3.0 \cdot 10^{-3}$. Circles (x-es) mark initial shifts for which convergence to the target takes place for DPA (two-sided RQI). Horizontal and vertical stride are both 0.25.

two-sided RQI (Alg. 2.1 and Alg. 2.2). The results, for two areas of the complex plane, are shown in Figure 2.4 and Figure 2.5.

Initial shifts for which DPA and two-sided RQI converge to the target (the most dominant pole, in the center of the domain) or its complex conjugate are marked by a circle and an x, respectively. In Figure 2.4, occasionally there is converge to a more dominant pole outside the depicted area (marked by a square and a +, respectively). Grid points with no marker denote convergence to a less dominant pole.

Figure 2.4 shows rather irregular convergence regions for both DPA and two-sided RQI. This is caused by the presence of many other (less) dominant poles in this part of the complex plane. Nevertheless, the basins of attraction of the dominant poles are notably larger for DPA. Moreover, it can be observed that even for initial shifts very close to another less dominant pole, DPA converges to a more dominant pole, while two-sided RQI converges to the nearest pole. For example, for initial shift $s_0 = -2 + 4.5i$, DPA converges to $\lambda \approx -2.9 + 4.8i$ with $|R| \approx 3.0 \cdot 10^{-3}$, while two-sided RQI converges to $\lambda \approx -2.1 + 4.6i$ with $|R| \approx 1.0 \cdot 10^{-5}$.

In Figure 2.5 the target is the most dominant pole of the system. It can be clearly observed that for DPA the number of initial shifts that converge to the dominant pole is larger than for two-sided RQI. The basin of attraction of the dominant pole is larger for DPA: except for regions in the neighborhood of other relatively dominant poles (see, for instance, the poles in the interval $(-28, -24)$ on the real axis), there is convergence to the most dominant pole. For DPA typically

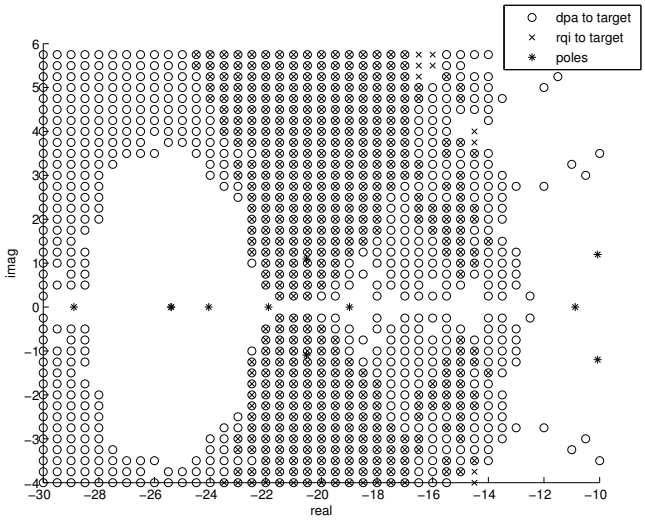


Figure 2.5: Convergence regions for DPA and two-sided RQI, for the example of Section 2.5.2. The center of the domain is the pole $\lambda \approx -20.5 \pm 1.1i$, with residue norm $|R| \approx 6.2 \cdot 10^{-3}$. Circles (x-es) mark initial shifts for which convergence to the target takes place for DPA (two-sided RQI). Horizontal and vertical stride are 0.5 and 0.25.

the size of the basin of attraction increases with the relative dominance of the pole, while for two-sided RQI the effect is less strong, cf. theorem 2.4.3, theorem 2.4.4 and the discussion in Section 2.4.2. The symmetry with respect to the real axis can be explained by the fact that if for initial shift s_0, DPA (two-sided RQI) produces the sequence $(\mathbf{v}_k, \mathbf{w}_k, s_{k+1})$ converging to $(\mathbf{x}, \mathbf{y}, \lambda)$, then for $\bar{s}_0$ it produces the sequence $(\bar{\mathbf{x}}_k, \bar{\mathbf{y}}_k, \bar{s}_{k+1})$ converging to $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})$.

In both figures it can be seen that for many initial shifts DPA converges to the most dominant pole, but two-sided RQI does not. On the other hand, for a very small number of initial shifts, two-sided RQI converges to the most dominant pole while DPA does not. This is a counterexample for the obvious thought that if two-sided RQI converges to the dominant pole, then also DPA converges to it.

The average number of iterations needed by DPA to converge to the most dominant pole was 7.2, while two-sided RQI needed an average number of 6.0 iterations. The average numbers over the cases where both DPA and two-sided RQI converged to the most dominant pole were 6.1 and 5.9 iterations, respectively.

Similar behavior is observed for other systems and transfer functions. Although the theoretical and experimental results do not provide hard evidence in the sense that they prove that the basin of attraction of the dominant pole is larger for DPA than for two-sided RQI, they indicate at least an advantage of DPA over two-sided RQI.

2.5.3 PEEC example

The PEEC system [28] is a well known benchmark system for model order reduction applications. One of the difficulties with this system of order $n = 480$ is that it has many equally dominant poles that lie close to each other in a relatively small part, $[-1, 0] \times [-10i, 10i]$, of the complex plane. This explains why in Figure 2.6 for only a relatively small part of the plane there is convergence (marked by circles and x-es for DPA and two-sided RQI, respectively) to the most dominant pole $\lambda \approx -0.14 \pm 5.4i$ (marked by a $*$).

Although the difference is less pronounced than in the previous examples, DPA still converges to the most dominant pole in more cases than two-sided RQI, and the average residue norm of the found poles was also larger: $R_{avg}^{dpa} \approx 5.2 \cdot 10^{-3}$ vs. $R_{avg}^{rqi} \approx 4.5 \cdot 10^{-3}$. Again a remarkable observation is that even for some initial shifts very close to another pole, DPA converges to the most dominant pole, while two-sided RQI converges to the nearest pole: e.g., for initial shift $s_0 = 5i$ DPA converges to the most dominant pole $\lambda \approx -0.143 + 5.38i$ with $|R| \approx 7.56 \cdot 10^{-3}$, while two-sided RQI converges to less dominant pole $\lambda \approx -6.3 \cdot 10^{-3} + 4.99i$ with $|R| \approx 3.90 \cdot 10^{-5}$.

The average number of iterations needed by DPA to converge to the most dominant pole was 9.8, while two-sided RQI needed an average number of 7.9 iterations. The average numbers over the cases where both DPA and two-sided RQI converged to the most dominant pole were 9.4 and 7.7 iterations, respectively.

2.6 Conclusions

The theoretical and numerical results confirm the intuition, and justify the conclusion, that the Dominant Pole Algorithm has better global convergence than two-sided Rayleigh quotient iteration to the dominant poles of a large-scale dynamical system. The derived local convergence neighborhoods of dominant poles are larger for DPA, as the poles become more dominant, and numerical experiments indicate that the local basins of attraction of the dominant poles are larger for DPA than for two-sided RQI.

Both DPA and two-sided RQI need to solve two linear systems at every iteration. The difference between DPA and two-sided RQI is that DPA keeps the right-hand sides fixed to the input and output vector of the system, while two-sided RQI updates the right-hand sides every iteration. The more dominant a pole is, the bigger the difference in convergence behavior between DPA and two-sided RQI. The other way around, for considerably less dominant poles, the basins of attraction are much smaller for DPA than for two-sided RQI. This could be observed in cases where the initial shift was very close to a less dominant pole and DPA converged to a more dominant pole, while two-sided RQI converged to the nearest, less dominant pole.

The fact that DPA has a asymptotically quadratic rate of convergence, against a

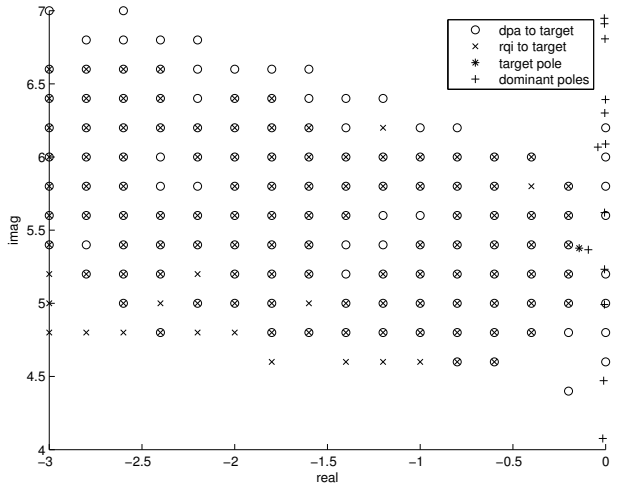


Figure 2.6: Convergence regions for DPA and two-sided RQI. The center of the domain is the pole $\lambda \approx -0.14 \pm 5.4i$, with residue norm $|R| \approx 7.6 \cdot 10^{-3}$. Circles (x-es) mark initial shifts for which convergence to the target takes place for DPA (two-sided RQI). Horizontal and vertical stride are both 0.2.

cubic rate for two-sided RQI, is of minor importance, since this has only a very local effect and hence leads to a small difference in the number of iterations (typically a difference of 1 or 2 iterations). Furthermore, there exist criteria to switch from DPA to two-sided RQI.

Addendum

This chapter is also available as [126]

> Joost Rommes and Gerard L. G. Sleijpen, *Convergence of the dominant pole algorithm and Rayleigh quotient iteration*, Preprint 1356, Utrecht University, 2006.

and has been submitted for publication.

Chapter 3

Subspace accelerated DPA and Jacobi-Davidson style methods

Abstract. This chapter describes a new algorithm for the computation of the dominant poles of a high-order scalar transfer function. The algorithm, called the Subspace Accelerated Dominant Pole Algorithm (SADPA), is able to compute the full set of dominant poles and to produce good modal equivalents automatically, without any human interaction. It is shown that SADPA is equivalent to two-sided Jacobi-Davidson, if the (correction) equations for both are solved exactly. The effects of subspace acceleration on local and global convergence are studied. The modal approximations computed by SADPA are compared to reduced-order models computed by rational Krylov methods, and it is shown how both methods can be applied to improve each other.

Key words. small-signal stability, poorly-damped oscillations, power system dynamics, transfer function, system poles, model reduction, dominant pole spectrum, large-scale systems, sparse eigenanalysis, modal equivalents, two-sided Jacobi-Davidson

3.1 Introduction

Recent work on power system stability, controller design and electromagnetic transients has used several advanced model reduction techniques [29, 108, 119, 134, 153], that produce good results but impose high computational costs. Modal model reduction is a cost-effective alternative for large-scale systems, when only a fraction of the system pole spectrum is controllable-observable for the transfer function of interest. The concept of pole dominance, as utilized in this chapter, implies high controllability and observability in the chosen transfer function. Modal reduction produces transfer function modal equivalents from the knowledge of the dominant poles and their corresponding residues, but requires specialized eigensolution methods. Existing methods are still not capable enough to produce automatically the full set of truly dominant poles [90, 91, 93] for large system models. A good sur-

vey on model reduction methods employing either singular value decompositions or moment matching based methods can be found in [5, 6]. A good introduction on modal model reduction on state-space models can be found in [68].

In this chapter, a new variant of the Dominant Pole Algorithm (DPA) [91] and the Dominant Pole Spectrum Eigensolver (DPSE) [90] will be proposed: Subspace Accelerated DPA (SADPA). Instead of computing the dominant poles of a scalar transfer function simultaneously, the dominant poles and corresponding residues are computed one by one by selecting the most dominant approximation every iteration. This approach leads to a faster, more robust and more flexible algorithm. To avoid repeated computation of the same dominant poles, a deflation strategy is used. SADPA directly operates on implicit state-space systems, also known as descriptor systems, which are very sparse in practical power system applications.

The subspace accelerated dominant pole algorithm SADPA (originally presented in [123], see also the Addendum of this chapter) will be related to two-sided Jacobi-Davidson [74]. It will be shown that if the linear systems that arise in SADPA, and the correction equations that arise in two-sided JD, are solved exactly, and if the same Ritz value selection criterion is used, then both methods are equivalent and compute the same dominant poles. This may be surprising at first sight, since *without* subspace acceleration, DPA and two-sided JD in general do not converge to the same pole: it was shown in Chapter 2 that DPA tends to converge to the most dominant pole in the neighborhood of the initial guess, while RQI tends to converge to the pole closest to the initial guess. Subspace acceleration in combination with a proper selection criterion makes both methods equivalent and hence enables two-sided Jacobi-Davidson to converge to dominant poles.

SADPA is a generalization of DPA to compute more than one dominant pole, by using deflation, subspace acceleration, and a proper selection strategy. Deflation is needed to avoid repeated computation of already found poles and hence can be seen as a necessary ingredient. Subspace acceleration, on the other hand, has the strategic advantage of better global convergence, but the computational drawback of having to keep orthogonal search spaces. If deflation is organized as deflation by restriction [112, Sect. 5.2], then DPA can easily and efficiently be extended to a multi-pole algorithm, without subspace acceleration. Practically speaking, however, this approach has some disadvantages, and subspace acceleration is an effective way to deal with these difficulties.

In the large-scale setting it is not always feasible to solve the linear systems and correction equations exactly. For Jacobi-Davidson style methods, it is well known that it is sufficient for local convergence to solve the correction equation up to a certain accuracy [51, 140]. In SADPA, however, the use of inexact solves may cause stagnation, even if subspace acceleration is used. This behavior is also observed for inexact (subspace accelerated) Rayleigh quotient iteration [74, 154]. Motivated by this observation and other well known arguments, two-sided Jacobi-Davidson is to be preferred if only inexact solves are feasible.

Besides presenting subspace accelerated DPA, the goal of this chapter is threefold. Firstly, is shown that SADPA and two-sided Jacobi-Davidson are equivalent methods if the equations are solved exactly. Secondly, the role of subspace ac-

celeration (and a proper selection criterion) with respect to global convergence is analyzed, and the influence of subspace acceleration on the asymptotic rate of convergence in general is studied. Thirdly, it will be shown that two-sided Jacobi-Davidson is an effective method to compute dominant poles if only inexact solves are feasible. All results are illustrated by numerical experiments.

The chapter is organized as follows. Section 3.2 gives definitions and properties of transfer functions and dominant poles, and describes the basic DPA algorithm. In Section 3.3 it is described how DPA can be extended to subspace accelerated DPA (SADPA), to compute more than one dominant pole, and in Section 3.4 it is shown how two-sided Jacobi-Davidson can be formulated to become equivalent to SADPA. The role of subspace acceleration with respect to global and local convergence is analyzed in Section 3.5. Inexact variants of SADPA and two-sided JD are discussed in Section 3.6. Numerical results are presented in Section 3.7. In Section 3.8 SADPA is compared to rational Krylov based model reduction methods. Section 3.9 concludes.

3.2 Transfer functions and dominant poles

In this chapter, the dynamical systems $(E, A, \mathbf{b}, \mathbf{c}, d)$ are of the form

$$\left\{ \begin{array}{rcl} E\dot{\mathbf{x}}(t) & = & A\mathbf{x}(t) + \mathbf{b}u(t) \\ y(t) & = & \mathbf{c}^*\mathbf{x}(t) + du(t), \end{array} \right. \tag{3.2.1}$$

where $A, E \in \mathbb{R}^{n\times n}$, E may be singular, $\mathbf{b}, \mathbf{c}, \mathbf{x}(t) \in \mathbb{R}^n$, $u(t), y(t), d \in \mathbb{R}$. The vectors $\mathbf{b}$ and $\mathbf{c}$ are called the input, and output map, respectively. The transfer function $H : \mathbb{C} \longrightarrow \mathbb{C}$ of (3.2.1) is defined as

$$H(s) = \mathbf{c}^*(sE - A)^{-1}\mathbf{b} + d. \tag{3.2.2}$$

The poles of transfer function (3.2.2) are a subset of the eigenvalues $\lambda_i \in \mathbb{C}$ of the matrix pencil (A, E). An eigentriplet $(\lambda_i, \mathbf{x}_i, \mathbf{y}_i)$ is composed of an eigenvalue λ_i of (A, E) and corresponding right and left eigenvectors $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{C}^n$ (identified by their components in $\mathbf{b}$ and $\mathbf{c}$):

$$\begin{array}{rcll} A\mathbf{x}_i & = & \lambda_i E\mathbf{x}_i, & \mathbf{x}_i \neq 0, \\ \mathbf{y}_i^*A & = & \lambda_i \mathbf{y}_i^*E, & \mathbf{y}_i \neq 0, \qquad (i = 1, \ldots, n). \end{array}$$

Assuming that the pencil is nondefective, the right and left eigenvectors corresponding to finite eigenvalues can be scaled so that $\mathbf{y}_i^*E\mathbf{x}_i = 1$. Furthermore, it is well known that left and right eigenvectors corresponding to distinct eigenvalues are E-orthogonal: $\mathbf{y}_i^*E\mathbf{x}_j = 0$ for $i \neq j$. The transfer function $H(s)$ can be expressed as a sum of residues $R_i \in \mathbb{C}$ over the $\tilde{n} \leq n$ finite first order poles [78]:

$$H(s) = \sum_{i=1}^{\tilde{n}} \frac{R_i}{s - \lambda_i} + R_\infty + d, \tag{3.2.3}$$

where the residues R_i are

$$R_i = (\mathbf{c}^*\mathbf{x}_i)(\mathbf{y}_i^*\mathbf{b}),$$

and R_∞ is the constant contribution of the poles at infinity (often zero).

Although there are different indices of modal dominance [4, 68, 159], the following will be used in this chapter.

Definition 3.2.1. *A pole λ_i of $H(s)$ with corresponding right and left eigenvectors $\mathbf{x}_i$ and $\mathbf{y}_i$ ($\mathbf{y}_i^*E\mathbf{x}_i = 1$) is called* the *dominant pole if $|R_i|/|Re(\lambda_i)| > |R_j|/|Re(\lambda_j)|$, for all $j \neq i$.*

More generally, a pole λ_i is called dominant if $|R_i|/|\mathrm{Re}(\lambda_i)|$ is not small compared to $|R_j|/|\mathrm{Re}(\lambda_j)|$, for all $j \neq i$. A dominant pole is well observable and controllable in the transfer function. This can also be seen in the corresponding Bode-plot, which is a plot of $|H(i\omega)|$ against $\omega \in \mathbb{R}$: peaks occur at frequencies ω close to the imaginary parts of the dominant poles of $H(s)$. An approximation of $H(s)$ that consists of $k < n$ terms with $|R_j|/|\mathrm{Re}(\lambda_j)|$ above some value, determines the effective transfer function behavior [144] and is also known as transfer function modal equivalent:

Definition 3.2.2. *A transfer function modal equivalent $H_k(s)$ is an approximation of a transfer function $H(s)$ that consists of $k < n$ terms:*

$$H_k(s) = \sum_{j=1}^{k} \frac{R_j}{s - \lambda_j} + d. \tag{3.2.4}$$

A modal equivalent that consists of the most dominant terms determines the effective transfer function behavior [144]. If $X \in \mathbb{C}^{n\times k}$ and $Y \in \mathbb{C}^{n\times k}$ are matrices having the left and right eigenvectors $\mathbf{y}_i$ and $\mathbf{x}_i$ of (A, E) as columns, such that $Y^*AX = \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_k)$, with $Y^*EX = I$, then the corresponding (complex) reduced system follows by setting $\mathbf{x} = X\tilde{\mathbf{x}}$ and multiplying from the left by Y^*:

$$\begin{cases} \dot{\tilde{\mathbf{x}}}(t) &= \Lambda\tilde{\mathbf{x}}(t) + (Y^*\mathbf{b})u(t) \\ \tilde{y}(t) &= (\mathbf{c}^*X)\tilde{\mathbf{x}}(t) + du(t). \end{cases}$$

In practice, it is advisable to make a real reduced model in the following way: for every complex pole triplet $(\lambda, \mathbf{x}, \mathbf{y})$, construct real bases for the right and left eigenspace via $[\mathrm{Re}(\mathbf{x}), \mathrm{Im}(\mathbf{x})]$ and $[\mathrm{Re}(\mathbf{y}), \mathrm{Im}(\mathbf{y})]$, respectively. Let the columns of X_r and Y_r be such bases, respectively. Because the complex conjugate eigenvectors are also in this space, the real bases for the eigenspaces are still (at most) k dimensional. The real reduced model can be formed by using X_r and Y_r in $(Y_r^*EX_r, Y_r^*AX_r, Y_r^*\mathbf{b}, X_r^*\mathbf{c}, d)$.

The dominant poles are specific (complex) eigenvalues of the pencil (A, E) and usually form a small subset of the spectrum of (A, E), so that rather accurate modal equivalents may be possible for $k \ll n$. The dominant poles can be located anywhere in the spectrum. Since the dominance of a pole is independent of d, without loss of generality $d = 0$ in the following.

3.2.1 The Dominant Pole Algorithm (DPA)

The poles of transfer function (3.2.2) are the $\lambda \in \mathbb{C}$ for which $\lim_{s\to\lambda} |H(s)| = \infty$ and, as was described in Section 2.3, can be computed via the roots of $G(s) = 1/H(s)$. Applying Newton's method leads to the scheme

$$s_{k+1} = s_k - \frac{\mathbf{c}^* \mathbf{v}_k}{\mathbf{w}_k^* E \mathbf{v}_k}, \tag{3.2.5}$$

where $\mathbf{v}_k = (s_k E - A)^{-1}\mathbf{b}$ and $\mathbf{w}_k = (s_k E - A)^{-*}\mathbf{c}$. The algorithm based on this scheme, also known as the Dominant Pole Algorithm (DPA) [91], is shown in Algorithm 2.1 in Chapter 2. Note that strictly speaking the definition of dominance used here is Definition 2.2.1.

The two linear systems that need to be solved in step 3 and 4 of Algorithm 2.1 to compute the Newton update in (3.2.5) can be efficiently solved using one LU-factorization $LU = s_k E - A$, by noting that $U^* L^* = (s_k E - A)^*$. If an exact LU-factorization is not available, one has to use inexact Newton schemes, such as inexact Rayleigh Quotient Iteration and Jacobi-Davidson style methods [74, 140, 148], a topic that will be studied in more detail in Section 3.6. In the next section, extensions of DPA are presented that are able to compute more than one eigenvalue in an effective and efficient way.

3.3 Deflation and subspace acceleration

In Chapter 2 the convergence of DPA is analyzed and compared to the convergence of two-sided Rayleigh quotient iteration. For symmetric matrices, it is shown that the convergence neighborhood of the dominant pole is larger for DPA than for RQI, and numerical experiments confirm that this is also true for general descriptor systems. DPA also has better global convergence to the dominant poles than two-sided RQI. The price one has to pay for this is that the asymptotic rate of convergence is quadratic, against the cubic rate of convergence of RQI. In practice, however, the difference is hardly noticeable, since the number of additional iterations is often only 1 or 2.

The situation may be different if deflation and subspace acceleration are used, and selection strategies are required to select a Ritz value at every iteration. In practical applications often more than one dominant pole is wanted: one is interested in all the dominant poles, no matter what definition of dominance is used. Simply running the single pole algorithm DPA for a number of different initial shifts will most likely result in duplicate dominant poles. In [90] the Dominant Pole Spectrum Eigensolver (DPSE) was presented as a block variant of DPA to compute multiple dominant poles, by running k DPA iterations in parallel for k different shifts. Every iteration, this leads to k approximate right and left eigenvectors $\mathbf{v}^{(i)}$ and $\mathbf{w}^{(i)}$ $(i = 1, \ldots, k)$. The shifts for the new iteration are the eigenvalues of the projected pencil (W^*AV, W^*EV), where the columns of V and W are the approximate right and left eigenvectors $\mathbf{v}^{(i)}$ and $\mathbf{w}^{(i)}$, respectively. DPSE, however, requires blocks of size $n \times k$, where k is the number of wanted poles, and this

may be impractical when k is large, since it leads to projected pencils of order k. Besides that, there may be simultaneous convergence to complex conjugate poles (redundancy), and, like in DPA, components in the direction of other eigenvectors are discarded at the end of every iteration.

A well known strategy to avoid computation of already found eigenvalues is deflation, see for instance [132]. It is also known that subspace acceleration may improve the global convergence: for an $n \times n$ problem, the subspace accelerated algorithm converges within at most n iterations, although in practice it may need only $k \ll n$ iterations and will almost never build a full search space of dimension n, but restart every $k_{max} \ll n$ iterations. The use of subspace acceleration requires that every iteration an approximate pole has to be selected from the available approximations. This also may improve the global convergence, since better approximations than the initial estimate, which may be a rather crude approximation, become available during the process. The effect on the local convergence, however, is less understood. The role of subspace acceleration will be analyzed in more detail in Section 3.5.

In Section 3.3.1, first a variant of DPA for the computation of more than one pole that does *not* use subspace acceleration is discussed. Although this variant can be implemented very efficiently with only constant costs for deflation, it also has some disadvantages that may make it of less use in practice. The problems that arise in this variant are an additional motivation for the use of subspace acceleration in Section 3.3.2. In Section 3.4, a variant based on two-sided Jacobi-Davidson is presented, and it is shown that subspace accelerated DPA (SADPA) and two-sided Jacobi-Davidson are equivalent methods.

Throughout the rest of this chapter, let the $(n \times k)$ matrices X_k and Y_k have as their columns the normalized (found) right and left eigenvectors $\mathbf{x}_i$ and $\mathbf{y}_i$ $(i = 1, \ldots, k)$ of (A, E), respectively, and let Λ_k be a diagonal $(k \times k)$ matrix with on its diagonal the corresponding eigenvalues, i.e. $\Lambda_k = \mathrm{diag}(\lambda_1, \ldots, \lambda_k)$, $Y_k^* A X_k = \Lambda_k$ and $Y_k^* E X_k = I$. For ease of notation, the subscript k will be omitted if this does not lead to confusion.

3.3.1 DPA with deflation by restriction

It can be verified that if $X \equiv X_k$ and $Y \equiv Y_k$ have as their columns exact eigenvectors (normalized so that $Y^* E X = I$), then the system $(E_d, A_d, \mathbf{b}_d, \mathbf{c}_d)$, where

$$\begin{aligned} E_d &= (I - EXY^*)E(I - XY^*E), \\ A_d &= (I - EXY^*)A(I - XY^*E), \\ \mathbf{b}_d &= (I - EXY^*)\mathbf{b}, \\ \mathbf{c}_d &= (I - E^*YX^*)\mathbf{c}, \end{aligned}$$

has the same poles, eigenvectors and residues, but with the found λ_i $(i = 1, \ldots, k)$ and corresponding R_i transformed to 0. So in order to avoid recomputing the same pole, DPA could be applied to the deflated system $(E_d, A_d, \mathbf{b}_d, \mathbf{c}_d)$ after having found one or more poles. This would require solves with $(sE_d - A_d)$ and $(sE_d - A_d)^*$

in step 4 and 5 of Algorithm 3.1[1], but the following theorem shows that it is sufficient to only replace $\mathbf{b}$ by $\mathbf{b}_d$ and $\mathbf{c}$ by $\mathbf{c}_d$ to ensure deflation.

Theorem 3.3.1. *The deflated transfer function* $H_d(s) = \mathbf{c}_d^*(sE - A)^{-1}\mathbf{b}_d$, *where*

$$\mathbf{b}_d = (I - EXY^*)\mathbf{b} \quad \text{and} \quad \mathbf{c}_d = (I - E^*YX^*)\mathbf{c},$$

has the same poles λ_i *and corresponding residues* R_i *as* $H(s) = \mathbf{c}^*(sE-A)^{-1}\mathbf{b}$, *but with the residues* R_i *corresponding to the found poles* λ_i $(i = 1, \ldots, k)$ *transformed to* $R_i = 0$.

Proof. Recall that the residue corresponding to an eigentriplet $(\lambda_i, \mathbf{x}_i, \mathbf{y}_i)$ of (A, E) is defined as $R_i = (\mathbf{c}^*\mathbf{x}_i)(\mathbf{y}_i^*\mathbf{b})$. Let $(\lambda, \mathbf{x}, \mathbf{y})$ be a found eigentriplet with $R = (\mathbf{c}^*\mathbf{x})(\mathbf{y}^*\mathbf{b})$ and hence $X(Y^*E\mathbf{x}) = \mathbf{x}$ and $(\mathbf{y}^*EX)Y^* = \mathbf{y}^*$. Note that clearly $Y^*\mathbf{b}_d = X^*\mathbf{c}_d = 0$, and it follows immediately that the corresponding residue in $H_d(s)$ is

$$R_d = (\mathbf{c}_d^*\mathbf{x})(\mathbf{y}^*\mathbf{b}_d) = (\mathbf{c}^*(I - XY^*E)\mathbf{x})(\mathbf{y}^*(I - EXY^*)\mathbf{b}) = 0.$$

Similarly, if $(\lambda, \mathbf{x}, \mathbf{y})$ is not deflated from $H(s)$ and hence $Y^*E\mathbf{x} = 0$ and $\mathbf{y}^*EX = 0$, then the corresponding residue is

$$R_d = (\mathbf{c}_d^*\mathbf{x})(\mathbf{y}^*\mathbf{b}_d) = (\mathbf{c}^*(I - XY^*E)\mathbf{x})(\mathbf{y}^*(I - EXY^*)\mathbf{b}) = (\mathbf{c}^*\mathbf{x})(\mathbf{y}^*\mathbf{b}) = R.$$

Since A and E are left unchanged, so are the eigenvalues of (A, E) and hence the poles. □

In other words, by using $\mathbf{b}_d$ and $\mathbf{c}_d$ the found dominant poles are degraded to non dominant poles of $H_d(s)$, while not changing the dominance of the remaining poles. Graphically, the peaks caused by the found poles are 'flattened' in the Bode plot (see also Figure 3.1).

Corollary 3.3.2. *Since the residues of the found poles are transformed to zero, the found poles are no longer dominant poles of* $H_d(s)$. *Hence these poles will not be recomputed by DPA applied to* $H_d(s)$.

Note that if $H(s) = \mathbf{c}^*(sE-A)^{-1}\mathbf{b}+d$ with $d = 0$, then the deflated poles in fact become zeros of $H_d(s)$. Finally, the following theorem shows that DPA applied to $H_d(s) = \mathbf{c}_d^*(sE - A)^{-1}\mathbf{b}_d$ and DPA applied to $H_{\tilde{d}}(s) = \mathbf{c}_d^*(sE_d - A_d)^{-1}\mathbf{b}_d$ produce the same results.

Theorem 3.3.3. *Given an initial shift* s_0, *DPA(*E, A, $\mathbf{b}_d$, $\mathbf{c}_d$, s_0*) and DPA(*E_d, A_d, $\mathbf{b}_d$, $\mathbf{c}_d$, s_0*) produce, in exact arithmetic, the same iterates and results.*

Proof. Denote the pole estimates produced by DPA(E, A, $\mathbf{b}_d$, $\mathbf{c}_d$, s_0) by s_k, and the pole estimates produced by DPA(E_d, A_d, $\mathbf{b}_d$, $\mathbf{c}_d$, s_0) by $\tilde{s}_k$. Let $\mathbf{v} = (s_kE - A)^{-1}\mathbf{b}_d$ and $\mathbf{w} = (s_kE - A)^{-*}\mathbf{c}_d$. It follows that $Y^*E\mathbf{v} = Y^*E(s_kE - A)^{-1}\mathbf{b}_d =$

[1] Note that $(sE_d - A_d)$ would never be computed explicitly, and that systems $(sE_d - A_d)\mathbf{x} = \mathbf{b}_d$ can be solved efficiently, see also Thm. 3.3.3.

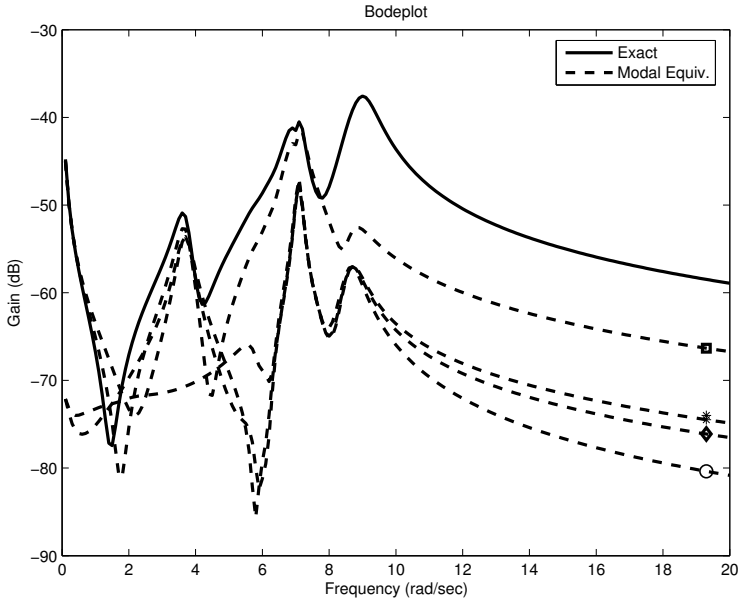


Figure 3.1: Exact transfer function (solid) of the New England test system [91], and modal equivalents where the following dominant pole (pairs) are removed one by one: $-0.467 \pm 8.96i$ (square), $-0.297 \pm 6.96i$ (asterisk), -0.0649 (diamond), and $-0.249 \pm 3.69i$ (circle). Note that the corresponding peaks are removed from the Bode plot as well (to see this, check the Bode plot at the frequencies near the imaginary part of the removed pole).

$(s_k I - \Lambda)^{-1} Y^* \mathbf{b}_d = 0$ and $\mathbf{w}^* E X = \mathbf{c}_d^* (s_k E - A)^{-1} E X = \mathbf{c}_d^* X (s_k I - \Lambda) = 0$. Consequently, $(I - XY^*E)\mathbf{v} = \mathbf{v}$, and since $(I - EXY^*)^2 = (I - EXY^*)$, $\mathbf{v}$ also satisfies $(s_k E_d - A_d)\mathbf{v} = \mathbf{b}_d$, and, similarly, $\mathbf{w}$ also satisfies $(s_k E_d - A_d)^* \mathbf{w} = \mathbf{c}_d$. It follows that if $\tilde{s}_k = s_k$, then

$$\tilde{s}_{k+1} = \frac{\mathbf{w}^* A_d \mathbf{v}}{\mathbf{w}^* E_d \mathbf{v}} = \frac{\mathbf{w}^* A \mathbf{v}}{\mathbf{w}^* E \mathbf{v}} = s_{k+1}$$

for all $k \geq 0$. □

The important consequence is that the single pole DPA can easily be extended, see Algorithm 3.1, to an algorithm that is able to compute more than one pole, while maintaining constant costs per iteration, except for iterations in which a pole is found. The only change to be made to Algorithm 2.1, is when a dominant pole triplet $(\lambda, \mathbf{x}, \mathbf{y})$ is found: in that case, the algorithm continues with $\mathbf{b}$ and $\mathbf{c}$ replaced by $(I - E\mathbf{x}\mathbf{y}^*)\mathbf{b}$ and $(I - E^*\mathbf{y}\mathbf{x}^*)\mathbf{c}$, respectively.

Theoretically speaking this approach has a number of advantages. The implementation is very straightforward and efficient: search spaces, selection strategies and orthogonalization procedures are not needed, so that the computational costs

per iteration remain constant, even if the number of found poles increases. For every found pole only two skew projections are needed once to compute the new $\mathbf{b}_d$ and $\mathbf{c}_d$, so the costs for deflation are constant. The pseudo code in Algorithm 3.1 can almost literally be used as Matlab code. The special properties of DPA ensure convergence to dominant poles (locally). Furthermore, the deflation of found poles is numerically stable in the sense that even if the corresponding transformed residues are not exactly zero, which is usually the case in finite arithmetic, this will hardly influence the effect of deflation: firstly, all the poles are left unchanged, and secondly, already a decrease of dominance of the found poles to nondominance (because of the projected in- and output vectors $\mathbf{b}_d$ and $\mathbf{c}_d$) will shrink the local convergence neighborhood of these poles significantly, again because of the convergence behavior of DPA [126] (see Chapter 2).

Algorithm 3.1 Dominant Pole Algorithm with deflation (DPAd)

INPUT: System $(E, A, \mathbf{b}, \mathbf{c})$, initial pole estimates $s_0^1, \ldots, s_0^p$, tolerance $\epsilon \ll 1$
OUTPUT: Approximate dominant poles $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$, and corresponding right and left eigenvectors $X = [\mathbf{x}_1, \ldots, \mathbf{x}_p]$ and $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_p]$

Set $k = 0$, $i = 0$, $s_k = s_0^1$
while $i < p$ **do**
  {Continue until p poles have been found}
  Solve $\mathbf{v}_k \in \mathbb{C}^n$ from $(s_k E - A)\mathbf{v}_k = \mathbf{b}$
  Solve $\mathbf{w}_k \in \mathbb{C}^n$ from $(s_k E - A)^* \mathbf{w}_k = \mathbf{c}$
  Compute the new pole estimate

$$s_{k+1} = s_k - \frac{\mathbf{c}^* \mathbf{v}_k}{\mathbf{w}_k^* E \mathbf{v}_k} = \frac{\mathbf{w}_k^* A \mathbf{v}_k}{\mathbf{w}_k^* E \mathbf{v}_k}$$

  if $\|A\mathbf{v}_k - s_{k+1} E \mathbf{v}_k\|_2 < \epsilon$ (with $\|\mathbf{v}_k\|_2 = 1$) **then**
    Set $i = i + 1$
    Set $\lambda_{ii} = s_{k+1}$
    Set $\mathbf{v}_k = \mathbf{v}_k / (\mathbf{w}_k^* E \mathbf{v}_k)$
    Set $X = [X, \mathbf{v}_k]$ and $Y = [Y, \mathbf{w}_k]$
    Deflate: $\mathbf{b} = \mathbf{b} - E\mathbf{v}_k \mathbf{w}_k^* \mathbf{b}$
    Deflate: $\mathbf{c} = \mathbf{c} - E^* \mathbf{w}_k \mathbf{v}_k^* \mathbf{c}$
    Set $s_{k+1} = s_0^i$
  end if
  Set $k = k + 1$
end while

This approach, however, may also suffer from several issues. Firstly, the convergence behavior can be very local and hence may heavily depend on the initial estimates s_0^i. Although in practice one often has rather accurate initial estimates of the poles of interest, this may be problematic if accurate information is not available. It may take many iterations until convergence if the initial estimate is not in the neighborhood of a dominant pole. On the other hand, the computational

complexity of this problem depends on the costs of the LU factorization, which in certain practical examples can be computed very efficiently.

Secondly, since the algorithm is usually applied to sparse descriptor systems, there are also eigenvalues of (A, E) at infinity. Although these eigenvalues are usually not dominant poles, since the corresponding residues are equal to zero, the estimates s_k computed by DPA may be approximations of the pole at infinity: due to rounding errors, the approximations $\mathbf{v}_k$ and $\mathbf{w}_k$ may be spoiled by components in the direction of eigenvectors corresponding to the eigenvalues at infinity. This may be prevented by purification techniques (see [97] and Chapter 7), but this does not fit nicely in the Newton scheme and costs (at least) two additional LU solves per iteration. If the poles at infinity are not dominant, DPA will never converge to such a pole, but convergence to other poles may be hampered nonetheless. Also, the presence of Jordan blocks may cause s_k to go to infinity: if λ has geometric multiplicity smaller than its algebraic multiplicity, then $\mathbf{y}^* E \mathbf{x} = 0$. If s_k happens to be a good estimate of λ, then $\mathbf{w}_k E \mathbf{v}_k \to 0$ and s_{k+1} can become very large (although two-sided Rayleigh quotient iteration may still converge at asymptotically linear rate, cf. [110, remark 2, p.689]). Problems due to the presence of Jordan blocks were, however, not experienced in any practical situation.

Thirdly, sooner or later the algorithm may get trapped in a stable periodic point of period 2, or even higher, of the underlying Newton algorithm. Such a point is in general not a zero (i.e. eigenvalue) of the original problem, but a periodic (fixed) point of the Newton scheme (which is a dynamical system itself). The chance that this happens increases as more poles are found. The difficulty here is that it is in general not possibly to predict the location or even the existence of stable periodic points a priori. The resulting stagnation, i.e. cycling between two complex conjugate values in the case of a period 2 point, however, can easily be detected by inspecting the values of s_k and s_{k+1}. Restarting with a different initial shift then solves the problem temporarily and may be sufficient in practice. In numerical experiments, periodic points were only observed after a large number of dominant poles had been found. To be more precise, for the cases where periodic points were encountered already all the dominant poles had been found. Intuitively one could argue that the remaining transfer function is of low magnitude (or even noise) and hence the chance of getting trapped in stable periodic points is bigger.

The main problem encountered in numerical experiments is slow global convergence. Combined with the characteristic convergence to dominant poles close to the initial shift, this stresses why the initial shifts are of great importance. If none of the shifts is in the neighborhood of the dominant pole, it may even not be found, due to the local character of the Newton process. On the other hand, provided the linear solves or LUs can be done cheaply, the low costs per iteration allow for the computation of poles for many different initial shifts, so that the frequency range of interest can be scanned effectively. In the next section a subspace accelerated version of DPA is described, that improves the global convergence by using search spaces.

3.3.2 Subspace accelerated DPA

A drawback of DPA is that information obtained in the current iteration is discarded at the end of the iteration. The only information that is preserved is contained in the new pole estimate s_{k+1}. The current right and left approximate eigenvectors $\mathbf{v}_k$ and $\mathbf{w}_k$, however, may also contain components in the direction of eigenvectors corresponding to other dominant poles. Instead of discarding these approximate eigenvectors, they are kept in search spaces spanned by the columns of V and W, respectively. This idea is known as subspace acceleration.

A global overview of SADPA is shown in Algorithm 3.2. Starting with a single shift s_1, the first iteration is equivalent to the first iteration of the DPA (step 3-4). The right and left eigenvector approximations $\mathbf{v}_1$ and $\mathbf{w}_1$ are kept in spaces V and W. In the next iteration, these spaces are expanded orthogonally, by using modified Gram-Schmidt (MGS) [64] (see also Algorithm 1.3 in Section 1.5.4), with the approximations $\mathbf{v}_2$ and $\mathbf{w}_2$ corresponding to the new shift s_2 (step 5-6). Hence the spaces grow and will contain better approximations. Note that it is also possible to keep V and W bi-orthogonal, or bi-E-orthogonal, see Section 3.4.

Algorithm 3.2 Subspace Accelerated DPA (SADPA)

INPUT: System $(E, A, \mathbf{b}, \mathbf{c})$, initial pole estimate s_1 and the number of wanted poles p

OUTPUT: Dominant pole triplets $(\lambda_i, \mathbf{x}_i, \mathbf{y}_i)$, $i = 1, \ldots, p$

1: $k = 1$, $p_{found} = 0$, $\Lambda = [\,]$, $X = Y = [\,]$
2: **while** $p_{found} < p$ **do**
3: Solve $\mathbf{v}$ from $(s_k E - A)\mathbf{v} = \mathbf{b}$
4: Solve $\mathbf{w}$ from $(s_k E - A)^* \mathbf{w} = \mathbf{c}$
5: $\mathbf{v} = \mathrm{MGS}(V, \mathbf{v})$, $V = [V, \mathbf{v}/||\mathbf{v}||_2]$
6: $\mathbf{w} = \mathrm{MGS}(W, \mathbf{w})$, $W = [W, \mathbf{w}/||\mathbf{w}||_2]$
7: Compute $S = W^*AV$ and $T = W^*EV$
8: $(\tilde{\Lambda}, \tilde{X}, \tilde{Y}) = \mathrm{Sort}(S, T, W^*\mathbf{b}, V^*\mathbf{c})$ {Algorithm 3.3}
9: Dominant approximate eigentriplet of (A, E) is

$$(\hat{\lambda}_1 = \tilde{\lambda}_1, \hat{\mathbf{x}}_1 = V\tilde{\mathbf{x}}_1/\|V\tilde{\mathbf{x}}_1\|_2, \hat{\mathbf{y}}_1 = W\tilde{\mathbf{y}}_1/\|W\tilde{\mathbf{y}}_1\|_2)$$

10: **if** $||A\hat{\mathbf{x}}_1 - \hat{\lambda}_1 E\hat{\mathbf{x}}_1||_2 < \epsilon$ **then**
11: $(\Lambda, X, Y, V, W, \mathbf{b}, \mathbf{c}) =$
$\mathrm{Deflate}(\hat{\lambda}_1, \hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1, \Lambda, X, Y, V\widetilde{X}_{2:k}, W\widetilde{Y}_{2:k}, E, \mathbf{b}, \mathbf{c})$ {Algorithm 3.4}
12: $p_{found} = p_{found} + 1$
13: Set $\tilde{\lambda}_1 = \tilde{\lambda}_2$, $k = k - 1$
14: **end if**
15: Set $k = k + 1$
16: Set the new pole estimate $s_k = \tilde{\lambda}_1$
17: **end while**

Selection strategy

In iteration k the Petrov-Galerking approach leads (step 7) to the projected eigenproblem

$$\begin{aligned} W^*AV\tilde{\mathbf{x}} &= \tilde{\lambda}W^*EV\tilde{\mathbf{x}}, \\ \tilde{\mathbf{y}}W^*AV &= \tilde{\lambda}\tilde{\mathbf{y}}W^*EV. \end{aligned}$$

Since the interaction matrices $S = W^*AV$ and $T = W^*EV$ are of low dimension $k \ll n$, the eigentriplets $(\tilde{\lambda}_i, \tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i)$ of this reduced problem can be computed using the QZ method (or the QR method in the bi-E-orthogonal case (step 1 of Algorithm 3.3)). This provides k approximate eigentriplets $(\hat{\lambda}_i = \tilde{\lambda}_i, \hat{\mathbf{x}}_i = V\tilde{\mathbf{x}}_i, \hat{\mathbf{y}}_i = W\tilde{\mathbf{y}}_i)$ for (A, E). The most natural thing to do is to choose the triplet $(\hat{\lambda}_j, \hat{\mathbf{x}}_j, \hat{\mathbf{y}}_j)$ with the most dominant pole approximation (step 8-9): compute the corresponding residues $\widehat{R}_i = (\mathbf{c}^*\hat{\mathbf{x}}_i)(\hat{\mathbf{y}}_i^*\mathbf{b})$ of the k pairs and select the pole with the largest $|\widehat{R}_j|/|\mathrm{Re}(\hat{\lambda}_j)|$ (see Algorithm 3.3). The SADPA then continues with the new shift $s_{k+1} = \hat{\lambda}_j$ (step 16).

The residues $\widehat{R}_i$ can be computed without computing the approximate eigenvectors explicitly (step 5 of Algorithm 3.3): if the $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{y}}_i$ are scaled so that $\tilde{\mathbf{y}}_i^*T\tilde{\mathbf{x}}_i = 1$ $(= \hat{\mathbf{y}}_i^*E\hat{\mathbf{x}}_i)$, then it follows that the $\widehat{R}_i$ can be computed as $\widehat{R}_i = ((\mathbf{c}^*V)\tilde{\mathbf{x}}_i)(\tilde{\mathbf{y}}_i^*(W^*\mathbf{b}))$ $(= (\mathbf{c}^*\hat{\mathbf{x}}_i)(\hat{\mathbf{y}}_i^*\mathbf{b}))$.

Instead of $\hat{\mathbf{y}}_i^*E\hat{\mathbf{x}}_i = 1$ one can also use the scaling $\|\hat{\mathbf{y}}_i\|_2 = \|\hat{\mathbf{x}}_i\|_2 = 1$ when computing approximate residues. In that case the product of the angles $\angle(\hat{\mathbf{x}}_i, \mathbf{c})$ and $\angle(\hat{\mathbf{y}}_i, \mathbf{b})$ is used in the computation of the approximate residues (see also Chapter 2), which numerically may be more robust. See Section 3.7 and the Addendum of Chapter 4 for more details.

Algorithm 3.3 $(\widetilde{\Lambda}, \widetilde{X}, \widetilde{Y}) = \mathrm{Sort}(S, T, \mathbf{b}, \mathbf{c})$

INPUT: $S, T \in \mathbb{C}^{k\times k}$, $\mathbf{b}, \mathbf{c} \in \mathbb{C}^k$

OUTPUT: $\widetilde{\Lambda} \in \mathbb{C}^k$, $\widetilde{X}, \widetilde{Y} \in \mathbb{C}^{k\times k}$ with λ_1 the pole with largest (scaled) residue magnitude and $\tilde{\mathbf{y}}_1$ and $\tilde{\mathbf{x}}_1$ the corresponding right and left eigenvectors.

1: Compute eigentriplets of the pair (S, T):

$$(\tilde{\lambda}_i, \tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i), \quad \tilde{\mathbf{y}}_i^*T\tilde{\mathbf{x}}_i = 1, \quad i = 1, \ldots, k$$

2: $\widetilde{\Lambda} = [\tilde{\lambda}_1, \ldots, \tilde{\lambda}_k]$
3: $\widetilde{X} = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k]$
4: $\widetilde{Y} = [\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_k]$
5: Compute residues $R_i = (\mathbf{c}^*\tilde{\mathbf{x}}_i)(\tilde{\mathbf{y}}_i^*\mathbf{b})$
6: Sort $\widetilde{\Lambda}$, $\widetilde{X}$, $\widetilde{Y}$ in decreasing $|R_i|/|\mathrm{Re}(\tilde{\lambda}_i)|$ order

---

Deflation

Every iteration a convergence test (step 10) is done like in DPAd (Algorithm 3.1): if for the selected eigentriplet $(\hat{\lambda}_1, \hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1)$ the norm of the residual $||A\hat{\mathbf{x}}_1 - \hat{\lambda}_1\hat{\mathbf{x}}_1||_2$ is smaller than some tolerance ϵ, it is converged. In general more than one dominant eigentriplet is wanted and it is desirable to avoid repeated computation of the same eigentriplet. The same deflation technique as used in DPAd can be applied here (step 5-6 and 12-13 of Algorithm 3.4, see also Section 3.3.1), and since SADPA continues with $\mathbf{b}_d$ and $\mathbf{c}_d$, no explicit E-orthogonalization of expansion vectors against found eigenvectors is needed in step 3 and 4. The effect is similar to the usual deflation in Jacobi-Davidson methods (see [51] and also Section 3.6.2): found eigenvectors are hard-locked, i.e. once deflated, they do not participate and do not improve during the rest of the process (contrary to soft-locking, where deflated eigenvectors still participate in the Rayleigh-Ritz (Ritz-Galerkin) procedure and may be improved, at the cost of additional computations and administration, see [80, 81]). In fact, there is cheap explicit deflation without the need for implicit deflation (cf. [51, remark 5, p. 106], where a combination of explicit and implicit deflation is used).

Algorithm 3.4 $(\Lambda, X, Y, \widetilde{V}, \widetilde{W}, \mathbf{b}_d, \mathbf{c}_d) = \text{Deflate}(\lambda, \mathbf{x}, \mathbf{y}, \Lambda, X, Y, V, W, E, \mathbf{b}, \mathbf{c})$

INPUT: $\lambda \in \mathbb{C}$, $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$, $\Lambda \in \mathbb{C}^p$, $X, Y \in \mathbb{C}^{n\times p}$, $V, W \in \mathbb{C}^{n\times k}$, $E \in \mathbb{C}^{n\times n}$, $\mathbf{b}, \mathbf{c} \in \mathbb{C}^n$

OUTPUT: $\Lambda \in \mathbb{C}^q$, $X, Y \in \mathbb{C}^{n\times q}$,$\widetilde{V}, \widetilde{W} \in \mathbb{C}^{n\times k-1}$,$\mathbf{b}_d, \mathbf{c}_d \in \mathbb{C}^n$, where $q = p + 1$ if λ has zero imaginary part and $q = p + 2$ if λ has nonzero imaginary part.

1: $\Lambda = [\Lambda, \lambda]$
2: Set $\mathbf{x} = \mathbf{x}/(\mathbf{y}^* E\mathbf{x})$
3: $X = [X, \mathbf{x}]$
4: $Y = [Y, \mathbf{y}]$
5: Deflate: $\mathbf{b}_d = \mathbf{b} - E\mathbf{x}(\mathbf{y}^*\mathbf{b})$
6: Deflate: $\mathbf{c}_d = \mathbf{c} - E^*\mathbf{y}(\mathbf{x}^*\mathbf{c})$
7: **if** imag$(\lambda) \neq 0$ **then**
8: {Also deflate complex conjugate}
9: $\Lambda = [\Lambda, \bar{\lambda}]$
10: $\mathbf{x} = \bar{\mathbf{x}}$, $X = [X, \mathbf{x}]$
11: $\mathbf{y} = \bar{\mathbf{y}}$, $Y = [Y, \mathbf{y}]$
12: Deflate: $\mathbf{b}_d = \mathbf{b}_d - E\mathbf{x}(\mathbf{y}^*\mathbf{b}_d)$
13: Deflate: $\mathbf{c}_d = \mathbf{c}_d - E^*\mathbf{y}(\mathbf{x}^*\mathbf{c}_d)$
14: **end if**
15: $\widetilde{V} = \widetilde{W} = [\ ]$
16: **for** $j = 1, \ldots, k$ **do**
17: $\widetilde{V} = \text{Expand}(\widetilde{V}, X, Y, E, \mathbf{v}_j)$ {Algorithm 3.5}
18: $\widetilde{W} = \text{Expand}(\widetilde{W}, Y, X, E^*, \mathbf{w}_j)$ {Algorithm 3.5}
19: **end for**

If an eigentriplet has converged (step 11-13), the eigenvectors are deflated from

the search spaces by reorthogonalizing the search spaces against the found eigenvectors. This can be done by using modified Gram-Schmidt (MGS) and by recalling that, if the exact vectors are found, the pencil

$$((I - EXY^*)A(I - XY^*E), \quad (I - EXY^*)E(I - XY^*E))$$

has the same eigentriplets as (A, E), but with the found eigenvalues transformed to zero (Algorithm 3.5, see also [51, 74]). Since in finite arithmetic only *approximations* to exact eigentriplets are available, the computed eigenvalues are transformed to $\eta \approx 0$. The possible numerical consequences of this, however, are limited, since SADPA continues with $\mathbf{b}_d$ and $\mathbf{c}_d$, and as argued in Section 3.3.1, the residues of the found poles are transformed to (approximately) zero.

If a complex pole has converged, its complex conjugate is also a pole and the corresponding complex conjugate right and left eigenvectors can also be deflated. A complex conjugate pair is counted as one pole. The complete deflation procedure is shown in Algorithm 3.4.

After deflation of the found pole(s), SADPA continues with the second most dominant approximate pole (step 13-16).

Algorithm 3.5 $V = \text{Expand}(V, X, Y, E, \mathbf{v})$

INPUT: $V \in \mathbb{C}^{n\times k}$ with $V^*V = I$, $X, Y \in \mathbb{C}^{n\times p}$, $E \in \mathbb{C}^{n\times n}$, $\mathbf{v} \in \mathbb{C}^n$, Y^*EX diagonal, $Y^*EV = 0$

OUTPUT: $V \in \mathbb{C}^{n\times(k+1)}$ with $V^*V = I$ and $\mathbf{v}_{k+1} = \prod_{j=1}^{p}(I - \frac{\mathbf{x}_j\mathbf{y}_j^*E}{\mathbf{y}_j^*E\mathbf{x}_j}) \cdot \mathbf{v}$

1: $\mathbf{v} = \prod_{j=1}^{p}(I - \frac{\mathbf{x}_j\mathbf{y}_j^*E}{\mathbf{y}_j^*E\mathbf{x}_j}) \cdot \mathbf{v}$
2: $\mathbf{v} = \text{MGS}(V, \mathbf{v})$
3: $V = [V, \mathbf{v}/||\mathbf{v}||_2]$

Further improvements and remarks

It may happen that the search spaces V and W become high-dimensional, especially when a large number of dominant poles is wanted. A common way to deal with this is to do an thick restart [51, 132]: if the subspaces V and W reach a certain maximum dimension $k_{max} \ll n$, they are reduced to a dimension $k_{min} < k_{max}$ by keeping the k_{min} most dominant approximate eigentriplets; the process is restarted with the reduced V and W (already converged eigentriplets are not part of the active subspaces V and W). This procedure is repeated until all poles are found.

Furthermore, as more eigentriplets have converged, approximations of new eigentriplets may become poorer or convergence may be hampered, due to rounding errors in the orthogonalization phase and the already converged eigentriplets. It is therefore advised to take a small tolerance $\epsilon \leq 10^{-10}$. Besides that, as the estimate converges to a dominant pole, the right and left eigenvectors computed in step 3 and 4 of Algorithm 3.2 are usually more accurate than the approximations

computed in the selection procedure: if the estimate s_k is close to an eigenvalue λ, then $s_kE - A$ may become ill conditioned, but, as is discussed in [116] and [112, Section 4.3], the solutions $\mathbf{v}_k$ and $\mathbf{w}_k$ have large components in the direction of the corresponding right and left eigenvectors (provided $\mathbf{b}$ and $\mathbf{c}$ have sufficiently large components in those directions). In the deflation phase, it is therefore advised to take the most accurate of both, i.e., the approximate eigenvector with smallest residual. It may also be advantageous to do an additional step of two-sided Rayleigh quotient iteration to improve the eigenvectors (see Section 3.5.2 for more details).

SADPA requires only one initial estimate. If rather accurate initial estimates are available, one can take advantage of this in SADPA by setting the next estimate after deflation to a new initial estimate (step 16 of Algorithm 3.2).

Every iteration, two linear systems are to be solved (step 3 and 4). As was also mentioned in Section 3.2.1, this can efficiently be done by computing one LU-factorization and solving the systems by using L and U, and U^* and L^*, respectively. Because in practice the system matrices A and E are often very sparse and structured, computation of the LU-factorizations can be relatively inexpensive.

Although poles at infinity are usually not dominant, components in the direction of the corresponding eigenvectors may still enter the search spaces (as was observed occasionally). Most likely these components are removed from the search spaces at a restart, but they might enter the search spaces again during the remainder of the process. A way to prevent this is to keep these components in the search spaces at a restart. The selection criterion takes care of selection of the approximations to finite dominant poles. In Chapter 7 more sophisticated purification techniques are studied.

The selection criterion can easily be changed to another of the several existing indices of modal dominance [4, 68, 159]. Furthermore, the strategy can be restricted to considering only poles in a certain frequency range. Also, instead of providing the number of wanted poles, the procedure can be automated even further by providing the desired maximum error $|H(s) - H_k(s)|$ for a certain frequency range: the procedure continues computing new poles until the error bound is reached. Note that such an error bound requires that the transfer function of the complete model can be evaluated efficiently for the frequency range of interest.

A numerical example

For illustrational purposes, SADPA was applied to a transfer function of the New England test system, a model of a power system. This small benchmark system has 66 state variables (for more information, see [91]). The tolerance used was $\epsilon = 10^{-10}$ and no restarts were used. Every iteration, the pole approximation $\hat{\lambda}_j$ with largest $|\widehat{R}_j|/|\mathrm{Re}(\hat{\lambda}_j)|$ was selected. Table 3.1 shows the found dominant poles and the iteration number for which the pole satisfied the stopping criterion. Bodeplots of two modal equivalents are shown in Figure 3.2. The quality of the modal equivalent increases with the number of found poles, as can be observed from the better match of the exact and reduced transfer function. In Section 3.7,

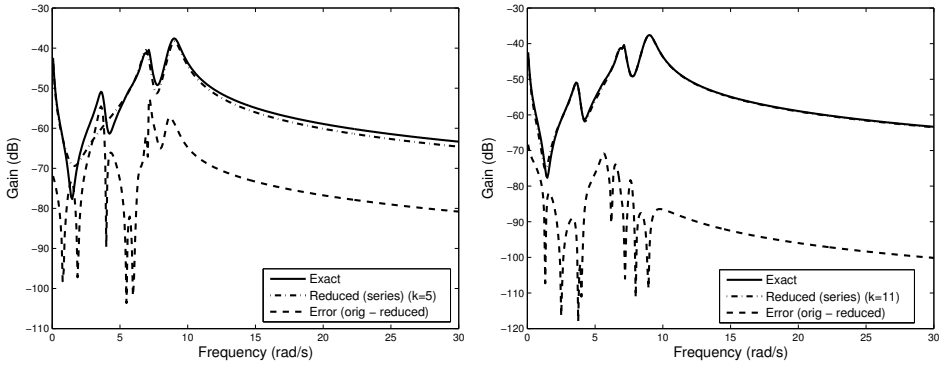


Figure 3.2: Bode plot of 5th order (left) and 11th order (right) modal equivalent, complete model and error for the transfer function of the New England test system (66 states in the complete model).

large-scale numerical examples are considered.

Table 3.1: Results for SADPA applied to the New England test system ($s_1 = 1i$).

#poles	#states	new pole	iteration	Bodeplot
1	2	$-0.4672 \pm 8.9644i$	13	-
2	4	$-0.2968 \pm 6.9562i$	18	-
3	5	-0.0649	21	Figure 3.2 (left)
4	7	$-0.2491 \pm 3.6862i$	25	-
5	9	$-0.1118 \pm 7.0950i$	26	-
6	11	$-0.3704 \pm 8.6111i$	27	Figure 3.2 (right)

3.3.3 Comparison between DPAd and SADPA

The expectation is that the use of search spaces not only improves the global convergence, but also reduces the average number of iterations needed to converge to a pole. This was confirmed by numerical experiments. In Figure 3.3 the average number of iterations and time per pole are shown, as well as the average residue norm of the found poles, for SADPA and DPAd. The test system (BIPS, see Section 3.7 for more details) was a descriptor system of dimension $n = 13,251$. It is clear that SADPA is faster than DPAd and requires less LU-factorizations. Since SADPA selects the most dominant approximation using $|R_i/\mathrm{Re}(\lambda_i)|$ as index for dominance, it converges to more dominant poles (measured using this index). Measured in the index $|R_i|$, DPAd computes more dominant poles on the average for the first 60 poles. However, if SADPA uses this index as selection criterion, it outperforms DPAd also for smaller numbers of poles.

The additional costs for SADPA for keeping the search spaces are earned back by faster global convergence: as a pole has converged and its corresponding eigen-

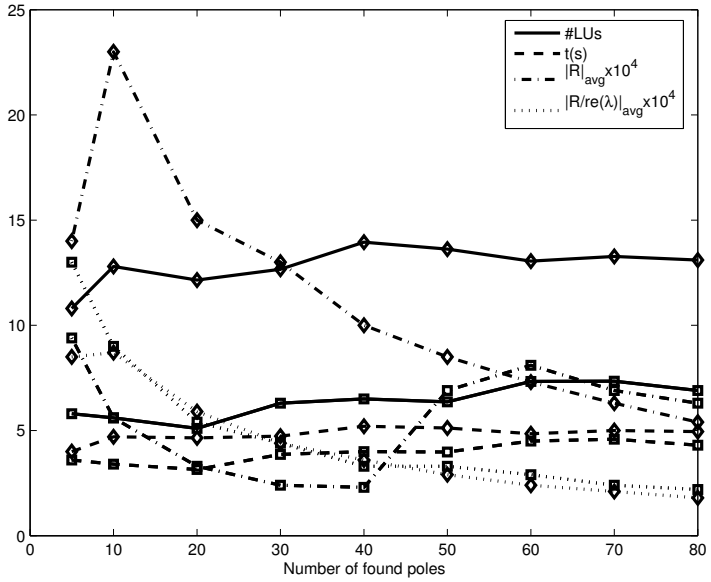


Figure 3.3: Average number of LUs (solid), time (dashed), residue norm $|R|$ (dash-dot) and scaled residue norm $|R|/|\mathrm{Re}(\lambda)|$ (dotted) per pole for SADPA (squares) and DPAd (diamonds). SADPA is faster, requires less LUs and has a larger average scaled residue norm than DPAd.

vectors are deflated from the search spaces, the remaining search spaces contain approximations of eigenvectors corresponding to other dominant poles. Hence, SADPA continues with a rather promising approximation of the next pole, while DPAd has to (re)start from scratch. The additional memory requirements are also limited since SADPA is restarted at dimension $k_{max} = 6$ to search spaces of dimension $k_{min} = 2$.

In practice, SADPA is the method of choice for many problems, because of its robustness, flexibility and fast convergence. For a wide range of problems, varying from small to large-scale systems, $1 \leq k_{min} < k_{\max} \leq 10$ are appropriate settings. The selection criterion can easily be adapted to the preferred measure of dominance. If, on the other hand, a small number of dominant poles in the close vicinity of some accurate initial shifts are wanted, and the LU factorizations can be computed efficiently, then it may be more advantageous to use DPAd, because of its local character.

3.4 SADPA and two-sided Jacobi-Davidson

3.4.1 Two-sided Jacobi-Davidson

If in the k-th iteration of SADPA, the right-hand sides $\mathbf{b}$ and $\mathbf{c}$ are replaced by $E\mathbf{v}_k$ and $E^*\mathbf{w}_k$, where $\mathbf{v}_k$ and $\mathbf{w}_k$ are the right and left approximate eigenvectors corresponding to the selected s_k, then the algorithm becomes subspace accelerated two-sided RQI. In this section, two-sided Jacobi-Davidson [74, 139, 148] is described. As will be shown in Section 3.4.2, all three methods are equivalent if the (correction) equations are solved exactly.

Two-sided Jacobi-Davidson (BiJD) [74, 148] is the Jacobi-Davidson [140] analog of two-sided Rayleigh quotient iteration [107, 110]. Let the columns of $V \in \mathbb{C}^{n\times k}$ and $W \in \mathbb{C}^{n\times k}$ be (orthogonal or bi-orthogonal) bases for the k-dimensional right and left search spaces $\mathcal{V}$ and $\mathcal{W}$. Two questions arise: (1) how can approximate eigentriplets be extracted from the search spaces, and (2) how can the search spaces be expanded in an effective way. To determine right and left approximate eigenvectors $\mathbf{v} = V\mathbf{s}$ and $\mathbf{w} = W\mathbf{t}$, with $\mathbf{s}, \mathbf{t} \in \mathbb{C}^k$, Petrov-Galerkin conditions are imposed on the right residual $\mathbf{r}_v$ and the left residual $\mathbf{r}_w$:

$$\mathbf{r}_v = A\mathbf{v} - \theta E\mathbf{v} \perp W \quad \text{and} \quad \mathbf{r}_w = A^*\mathbf{w} - \bar{\theta}E^*\mathbf{w} \perp V,$$

where $\theta = (\mathbf{w}^*A\mathbf{v})/(\mathbf{w}^*E\mathbf{v})$. It follows that $(\theta, \mathbf{s}, \mathbf{t})$ can be computed as an eigentriplet of the projected eigenproblem

$$W^*AV\mathbf{s} = \theta W^*EV\mathbf{s} \quad \text{and} \quad V^*A^*W\mathbf{t} = \bar{\theta}V^*E^*W\mathbf{t}.$$

Note that this is a problem of small size $k \ll n$ that can be solved using a full space method like the QZ method. When computing dominant poles, the selection strategy used in SADPA (Algorithm 3.3) can be used to select to most dominant approximate triplet.

Given an approximate eigentriplet $(\theta, \mathbf{v}, \mathbf{w})$, two-sided JD computes corrections $\mathbf{f}$ and $\mathbf{g}$ so that

$$A(\mathbf{v}+\mathbf{f}) = \lambda E(\mathbf{v}+\mathbf{f}) \quad \text{and} \quad A^*(\mathbf{w}+\mathbf{g}) = \bar{\lambda}E^*(\mathbf{w}+\mathbf{g}),$$

where λ is unknown. This leads to the right correction equation

$$(A-\lambda E)\mathbf{f} = -\mathbf{r}_v + (\lambda-\theta)\mathbf{v},$$

and the left correction equation

$$(A-\lambda E)^*\mathbf{g} = -\mathbf{r}_w + (\bar{\lambda}-\bar{\theta})\mathbf{w}.$$

Typical for the JD process, these equations are projected so that $E\mathbf{v}$ and $E^*\mathbf{w}$ are mapped to 0, respectively, and so that $\mathbf{r}_v$ and $\mathbf{r}_w$ are kept fixed. Replacing λ by θ, this gives the correction equations

$$\begin{aligned}\left(I - \frac{E\mathbf{v}\mathbf{w}^*}{\mathbf{w}^*E\mathbf{v}}\right)(A-\theta E)\mathbf{f} &= -\mathbf{r}_v,\\ \left(I - \frac{E^*\mathbf{w}\mathbf{v}^*}{\mathbf{v}^*E^*\mathbf{w}}\right)(A-\theta E)^*\mathbf{g} &= -\mathbf{r}_w.\end{aligned}$$

There are two obvious choices for the search spaces: either they are kept bi-E-orthogonal, i.e. $W^*EV = I$, or they are kept orthogonal, i.e. $W^*W = V^*V = I$. If V and W are kept bi-E-orthogonal, which seems natural since the right and left eigenvectors are also bi-E-orthogonal, the correction equations become

$$\left(I - \frac{E\mathbf{v}\mathbf{w}^*}{\mathbf{w}^*E\mathbf{v}}\right)(A - \theta E)\left(I - \frac{\mathbf{v}\mathbf{w}^*E}{\mathbf{w}^*E\mathbf{v}}\right)\mathbf{f} = -\mathbf{r}_v \quad (\mathbf{f} \perp E^*\mathbf{w}), \quad (3.4.1)$$

$$\left(I - \frac{E^*\mathbf{w}\mathbf{v}^*}{\mathbf{v}^*E^*\mathbf{w}}\right)(A - \theta E)^*\left(I - \frac{\mathbf{w}\mathbf{v}^*E^*}{\mathbf{v}^*E^*\mathbf{w}}\right)\mathbf{g} = -\mathbf{r}_w \quad (\mathbf{g} \perp E\mathbf{v}). \quad (3.4.2)$$

If V and W are kept orthogonal, the correction equations become

$$\left(I - \frac{E\mathbf{v}\mathbf{w}^*}{\mathbf{w}^*E\mathbf{v}}\right)(A - \theta E)\left(I - \frac{\mathbf{v}\mathbf{v}^*}{\mathbf{v}^*\mathbf{v}}\right)\mathbf{f} = -\mathbf{r}_v \quad (\mathbf{f} \perp \mathbf{v}), \quad (3.4.3)$$

$$\left(I - \frac{E^*\mathbf{w}\mathbf{v}^*}{\mathbf{v}^*E^*\mathbf{w}}\right)(A - \theta E)^*\left(I - \frac{\mathbf{w}\mathbf{w}^*}{\mathbf{w}^*\mathbf{w}}\right)\mathbf{g} = -\mathbf{r}_w \quad (\mathbf{g} \perp \mathbf{w}). \quad (3.4.4)$$

Keeping V and W bi-E-orthogonal has the advantage that the projected problem reduces to an ordinary eigenproblem, but the disadvantage that the E-inner products are more expensive than standard inner products and possibly less stable or even not well-defined. There are also other options for the projections in the correction equations, see [139]. See Alg. 1.5 for BiJD with orthogonal search spaces.

If the correction equations are solved exactly, then BiJD is equivalent to accelerated two-sided Rayleigh quotient iteration, see [74]. Note again that the projected operators are never formed explicitly. The solution for the right correction equation (3.4.1) with bi-E-orthogonal search spaces, for instance, can be computed as

$$\mathbf{f} = -\mathbf{v} + \mu(A - \theta E)^{-1}E\mathbf{v},$$

with $\mu = \mathbf{w}^*E\mathbf{v}/(\mathbf{w}^*E(A - \theta E)^{-1}E\mathbf{v})$. In practice it is often sufficient and even advantageous to solve the correction equation up to a certain precision with Krylov subspace methods. Inexact variants are discussed in Section 3.6.

3.4.2 Equivalence of SADPA and two-sided JD

While DPA and two-sided Rayleigh quotient iteration have different convergence behavior (locally quadratic versus cubic rate of convergence, and, moreover, DPA has better convergence to dominant poles, see also Chapter 2), their subspace accelerated versions can be shown to be equivalent if the linear systems are solved exactly (see also [74] for the equivalence of two-sided JD and two-sided RQI). This may be somewhat surprising, but with the following two lemmas it can be shown that the search spaces constructed by both algorithms are the same, provided the same initial vectors and selection criterion are used.

Lemma 3.4.1. *Let* $\sigma \neq \tau \in \mathbb{C} \setminus \Lambda(A, E)$, $\mathbf{x} = (\sigma E - A)^{-1}\mathbf{b}$, $\mathbf{y} = (\tau E - A)^{-1}\mathbf{b}$ *and* $\mathbf{z} = (\tau E - A)^{-1}E(\sigma E - A)^{-1}\mathbf{b}$. *Then* $\mathbf{z} \in$ *span*$(\mathbf{x}, \mathbf{y})$.

Proof. Suppose $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are independent. Then for α, β, γ,

$$\alpha\mathbf{x} + \beta\mathbf{y} + \gamma\mathbf{z} = 0 \Longleftrightarrow \alpha = \beta = \gamma = 0.$$

Multiplication by $(\tau E - A) = (\sigma E - A + (\tau - \sigma)E)$ gives

$$\alpha\mathbf{b} + \alpha(\tau - \sigma)E(\sigma E - A)^{-1}\mathbf{b} + \beta\mathbf{b} + \gamma E(\sigma E - A)^{-1}\mathbf{b} = 0,$$

and the contradiction follows for $\alpha = -\gamma/(\tau - \sigma)$ and $\beta = -\alpha$. □

This result can be generalized in the following way.

Lemma 3.4.2. *Let $\sigma_i \in \mathbb{C} \setminus \Lambda(A, E)$ for $i = 0, 1, \ldots, k$, and let*

$$V_k = \mathit{span}((\sigma_0 E - A)^{-1}\mathbf{b}, (\sigma_1 E - A)^{-1}\mathbf{b}, \ldots, (\sigma_{k-1} E - A)^{-1}\mathbf{b}).$$

If $\mathbf{v}_k \in V_k$, then $\mathbf{v}_{k+1} = (\sigma_k E - A)^{-1} E\mathbf{v}_k \in V_{k+1}$.

Proof. Since $\mathbf{v}_k \in V_k$, there is a $\mathbf{u} \in \mathbb{C}^k$ so that $\mathbf{v}_k = V_k\mathbf{u}$. By lemma 3.4.1 it follows that

$$(\sigma_k E - A)^{-1} E(\sigma_i E - A)^{-1}\mathbf{b} \in \mathrm{span}((\sigma_i E - A)^{-1}\mathbf{b}, (\sigma_k E - A)^{-1}\mathbf{b})$$

for $i = 0, 1, \ldots, k-1$. Hence, $\mathbf{v}_{k+1} \in V_{k+1}$. □

In other words, given a search space V_k and a Ritz pair $(s_k, \mathbf{v})$ with $\mathbf{v} \in V_k$, the search space V_k expanded with the new approximation $\mathbf{v}_{k+1} = (s_k E - A)^{-1} E\mathbf{v}$ (RQI) is equal to V_k expanded with $\mathbf{v}_{k+1} = (s_k E - A)^{-1}\mathbf{b}$ (DPA). A similar conclusion can be drawn for the left search space W_k with expansions $\mathbf{w}_{k+1} = (s_k E - A)^{-*} E^*\mathbf{w}$ or $\mathbf{w}_{k+1} = (s_k E - A)^{-*}\mathbf{c}$. Consequently, accelerated DPA and accelerated two-sided RQI are equivalent, provided that if DPA is started with s_0, then two-sided RQI has to be started with $\mathbf{v}_0 = (s_0 E - A)^{-1}\mathbf{b}$ and $\mathbf{w}_0 = (s_0 E - A)^{-*}\mathbf{c}$, and both methods use the same selection criterion.

Theorem 3.4.3. *Let $s_0 \in \mathbb{C}$, $\mathbf{v}_0 = (s_0 E - A)^{-1}\mathbf{b}$ and $\mathbf{w}_0 = (s_0 E - A)^{-*}\mathbf{c}$. If the same selection criterion is used, then SADPA($E, A, \mathbf{b}, \mathbf{c}, s_0$), exact two-sided RQI($E, A, \mathbf{v}_0, \mathbf{w}_0$), and exact BiJD($E, A, \mathbf{v}_0, \mathbf{w}_0$) are equivalent.*

Proof. Use lemma 3.4.2 and an induction argument. For the equivalence of BiJD and two-sided RQI, see [74]. □

Note that it is not only crucial that two-sided RQI starts with the specific $\mathbf{v}_0 = (s_0 E - A)^{-1}\mathbf{b}$ and $\mathbf{w}_0 = (s_0 E - A)^{-*}\mathbf{c}$, but that this is also a natural choice: given an initial shift s_0 it is (locally) the best choice, and, secondly, in general $E\mathbf{b} = 0$ and/or $E^*\mathbf{c} = 0$, so that the choices $\mathbf{v}_0 = (s_0 E - A)^{-1} E\mathbf{b}$ and $\mathbf{w}_0 = (s_0 E - A)^{-*} E^*\mathbf{c}$ are not applicable at all. Strictly speaking, BiJD (and subspace accelerated Rayleigh quotient iteration) start with search spaces of dimension 1, while SADPA starts with empty search spaces.

The advantage of SADPA over BiJD and two-sided RQI is that SADPA can make use of very cheap deflation via $\mathbf{b}_d = (I - E\mathbf{x}\mathbf{y}^*)\mathbf{b}$ and $\mathbf{c}_d = (I - E^*\mathbf{y}\mathbf{x}^*)\mathbf{c}$. Since for BiJD and two-sided RQI the relation with $\mathbf{b}$ and $\mathbf{c}$ is lost (except for the initial vectors), vectors that are added to the search spaces need to be orthogonalized against found eigenvectors. This needs to be done every iteration and becomes more expensive as more eigenvectors are found, while SADPA only has to deflate the found eigenvectors once during the whole process. To conclude, if it is affordable to solve the (correction) equations exactly, then SADPA is the method of choice. Numerical experiments confirm the equivalence of SADPA, BiJD and two-sided RQI, but also show that numerical round off may spoil the processes (see the example in the Section 3.5.2). In Section 3.6 it is shown that inexact BiJD is to be preferred if exact solves of the (correction) equations are not affordable.

The right and left search spaces constructed by SADPA (and accelerated two-sided RQI) are also known as rational Krylov subspaces [131]. In [131, Section 6] it is shown that for certain choices of the shifts and Ritz pairs, the rational Krylov scheme (RKS) is equivalent to the Jacobi-Davidson method. Generalizing, it can be shown in a similar way that SADPA and two-sided RKS are equivalent. See [131] and references therein for more details about the rational Krylov scheme, and [58] for the rational Lanczos process in the context of model order reduction.

3.5 The role of subspace acceleration

If a subspace accelerated process such as SADPA or BiJD is not restarted when a certain maximum dimension k_{max} of the search space is reached, then (in exact arithmetic) it trivially terminates within n iterations. In practice, usually good approximations of the wanted eigentriplet can be extracted from search spaces of dimension $k \ll n$. Together with restarting this makes the accelerated schemes preferable over the standard DPA and two-sided RQI, that are not guaranteed to converge within n iterations, or even do not converge at all in some situations (although this may also not be guaranteed when using restarts). For inexact versions the effects of subspace acceleration may have even bigger impact. Other effects of subspace acceleration are discussed in the following subsections.

3.5.1 Global convergence

Subspace acceleration not only speeds up the convergence, it also improves global convergence. Firstly, in the case of SADPA, it tempers the local character of DPA: since DPA is a Newton scheme, it converges most likely to a pole in the neighborhood of the initial guess. The search spaces may contain approximations of more dominant eigentriplets, that can be extracted using a selection criterion. Note that the use of subspace acceleration also offers flexibility with respect to the selection criterion, and provides some control on the convergence region.

Secondly, the search spaces automatically provide approximations for other dominant eigentriplets, so that after deflation of a found eigentriplet, the search for

the next can be continued with the most promising or dominant approximation. Also, if the initial guess is not close to a dominant pole, search spaces may quickly provide a better approximation. Note that in the case of DPAd a new shift should be supplied by the user to obtain better global convergence (although the same initial shift could be used after deflation without the risk of converging to the same pole).

As is also described in Section 3.3.2, the costs for keeping the (bi)-orthogonal search spaces is earned back by the faster convergence, lower overall computational time and qualitatively more dominant poles.

3.5.2 Local convergence

The situation is different in the final (local) phase of convergence to a pole. Suppose that the current approximation $(\hat{\lambda}, \mathbf{v}, \mathbf{w})$ of $(\lambda, \mathbf{x}, \mathbf{y})$ satisfies $||\mathbf{r}_v||_2 = ||A\mathbf{v} - \hat{\lambda}E\mathbf{v}||_2 = \tau$, with $\epsilon = 10^{-10} < \tau < 10^{-4}$. If the (correction) equations are solved exactly, then from now on it is not likely that much new information will be added to the search spaces, since $(\hat{\lambda}, \mathbf{v}, \mathbf{w}) \rightarrow (\lambda, \mathbf{x}, \mathbf{y})$ (up to $||\mathbf{r}_v||_2 < \epsilon$) within a few iterations. Instead of (bi)orthogonally expanding the search spaces, it is more efficient to improve the approximate eigentriplet using DPA or two-sided RQI. Usually this only requires 1 or 2 iterations, since the initial vectors are very close to the wanted eigenvectors and $\hat{\lambda} \approx \lambda$. For the same reason, there is low risk to converge to a pole $\mu \neq \lambda$, so deflation of already found eigentriplets is not needed in the DPA and RQI iterations.

Another problem that might occur is that round off errors, introduced by deflation, orthogonalization, and extraction of the eigentriplets from the projected problem, prevent the approximate eigentriplet $(\hat{\lambda}, \mathbf{v}, \mathbf{w})$ from satisfying $||\mathbf{r}_v||_2 = ||A\mathbf{v} - \hat{\lambda}E\mathbf{v}||_2 < \epsilon$: there is stagnation at $||\mathbf{r}_v|| = \tau$ with $\epsilon \lesssim \tau$. Also in this case two-sided RQI may help, since round off due to the inversion of $(A - \hat{\lambda}E)$ is in the desired direction [116], [112, Section 4.3] and the process is not disturbed by orthogonalization and deflation.

In practice the tolerance $\tau \ll 1$ for switching to DPA or two-sided RQI should be chosen with care. If τ is too large, the process of finding the most dominant will be more or less randomized and become greedy, resulting in less dominant poles. On the other hand, if τ is too small, the process may be hampered by stagnation or slow convergence. Based on experimental experience, a value $10^{-4} < \tau < 10^{-7}$ is a good choice in practice.

Numerical experiments show that a switch to two-sided RQI is more crucial for SADPA than for BiJD. The example displayed in Figure 3.4 is typical for this phenomenon. The left figure shows the convergence for SADPA and BiJD with tolerance $\epsilon = 10^{-8}$ for computing the 5 most dominant poles of the PEEC example from [28], *without* switching. Up to the 14th iteration the equivalence of both methods is apparent. However, SADPA, without switching, stagnates after iteration 17, while BiJD converges smoothly. This can be explained by the fact that Jacobi-Davidson computes the expansion vectors as corrections to the cur-

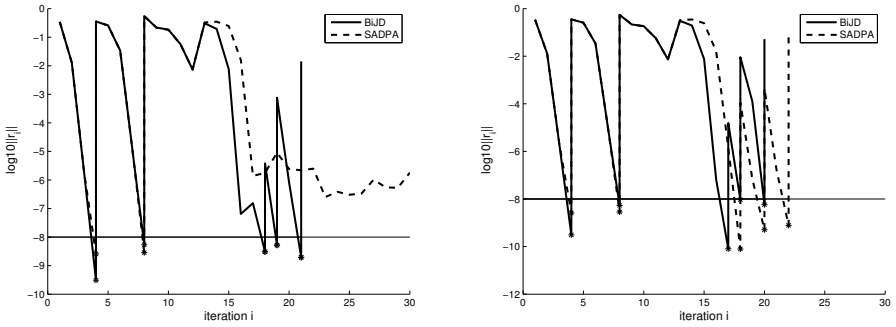


Figure 3.4: Convergence histories for BiJD and SADPA without switch (left) and with switch to two-sided RQI at $||\mathbf{r}_v||_2 = 10^{-6}$. The system is the PEEC example from [28], of order $n = 480$.

rent approximations, while SADPA in the final phase computes expansion vectors that make a (very) small angle with the current search spaces, possibly leading to cancellation during the orthogonal expansion. Also, the fixed right-hand sides in SADPA might hamper convergence to the desired tolerance.

The right history shows the convergence of SADPA and BiJD with switching to two-sided RQI at $||\mathbf{r}_v||_2 = 10^{-6}$. It can be observed that both methods take advantage of the switch to two-sided RQI, both with respect to the number of iterations and the accuracy. Note that SADPA now converges almost as smoothly as BiJD. Both methods suffer from loss of track in iteration 13. If the tolerance is decreased to $\epsilon = 10^{-10}$, both SADPA and BiJD require the switch in order to converge to the desired accuracy. Extensive numerical experiments with many different dynamical systems has led to the insight that these problems specifically arise if there are (many) poles with relatively small real part, compared to the imaginary part. Apparently it is difficult to compute the real part of the pole (and corresponding eigenvectors) accurately, and errors in already found eigentriplets spoil the rest of the process. The PEEC example [28] in Figure 3.4 has a lot of dominant poles λ_i with $\mathrm{Im}(\lambda_i)$ of $O(1)$, while $\mathrm{Re}(\lambda_i)$ varies from $O(10^{-2})$ to $O(10^{-6})$.

If the (correction) equations are solved up to a certain precision, as will be discussed in the next section, then the above is still true and the use of subspace acceleration even helps more to converge to eigentriplets. The method of choice is then inexact BiJD.

3.6 Inexact variants for computing dominant poles

Inexact Rayleigh quotient iteration may have poor or even no convergence if the solutions of the linear systems are too inaccurate, see for instance [154, experiment 2.1]. Because the Jacobi-Davidson method computes the expansion vector as the

solution of a correction equation, it suffers less from this problem [74, 140, 154] and is therefore the method of choice for large-scale eigenvalue problems.

It can be shown, as a generalization of [154, Proposition 2.2], [143, Corollary 4.3], [74, Theorem 4.2] and Theorem 4.2 in Chapter 2, that inexact DPA also converges asymptotically quadratically. However, an inexact version of (subspace accelerated) DPA is expected to have the same, and probably even more severe problems than inexact RQI, as will be explained next. Inexact SADPA solves the equations in step 3 and 4 of Algorithm 3.2 with only moderate accuracy, for instance using a fixed number of iterations of GMRES (solving both equations separately) or BiCG (solving both equations simultaneously, see [74]). Suppose now that already a good approximation $\tilde{\lambda} \approx \lambda$ of the dominant pole is available, but that the approximate right and left eigenvectors are rather inaccurate. Since $\tilde{\lambda}$ will not change very much during the following iterations, also the expansion vectors will remain the same, because the right-hand sides remain constant. This explains why the problems might be even more severe for inexact DPA than for inexact two-sided RQI. Of course, increasing the number of iterations of the iterative solver as $\tilde{\lambda}$ converges might help, as is usual practice for inexact Newton schemes (see for instance [51, 141]), but still the ill-conditioning of the system as $\tilde{\lambda} \to \lambda$ causes difficulties, like in two-sided RQI. Since the projected correction equations in the two-sided Jacobi-Davidson method handle these problems better, BiJD is the preferred method when exact solves are not affordable. In practice, this better convergence behavior compensates the locally linear convergence [74, Theorem 5.3] of BiJD by far.

3.6.1 Preconditioning

Because the domain and image spaces of the operators in the correction equations (3.4.1)–(3.4.4) are not the same, it is unattractive to solve the equations with a Krylov solver. The usual way to fix this problem is to precondition with a suitable preconditioner (see for instance [51, Section 3.3] and [50, Section 6.2.4]). In the bi-E-orthogonal case (equations (3.4.1) and (3.4.2)), an obvious preconditioner for the right correction equation (3.4.1) is, given a preconditioner $K \approx A - \theta E$,

$$\tilde{K} = (I - E\mathbf{v}\mathbf{w}^*)K(I - \mathbf{v}\mathbf{w}^*E) \ : \ (E^*\mathbf{w})^\perp \longrightarrow \mathbf{w}^\perp,$$

with $\mathbf{w}^*E\mathbf{v} = 1$. Typical use in a Krylov solver requires actions of the form: solve $\mathbf{s} \perp (E^*\mathbf{w})$ from

$$(I - E\mathbf{v}\mathbf{w}^*)K(I - \mathbf{v}\mathbf{w}^*E)\mathbf{s} = \mathbf{t}, \quad \mathbf{w}^*\mathbf{t} = 0.$$

To obtain an expression for $\mathbf{s}$ (see also, for instance, [141, Section 3.2] and [50, Section 6.2.4]), first note that $\mathbf{w}^*E\mathbf{s} = 0$ and hence, since K is assumed to be invertible,

$$\mathbf{s} = K^{-1}\mathbf{t} + K^{-1}E\mathbf{v}(\mathbf{w}^*K\mathbf{s}).$$

Again use $\mathbf{w}^*E\mathbf{s} = 0$ to obtain by multiplication by $\mathbf{w}^*E$

$$\mathbf{w}^*K\mathbf{s} = -(\mathbf{w}^*EK^{-1}E\mathbf{v})^{-1}\mathbf{w}^*EK^{-1}\mathbf{t},$$

and with $\mathbf{q}_r = E^*\mathbf{w}$, $\mathbf{y}_r = K^{-1}E\mathbf{v}$ and $h_r = \mathbf{q}_r^*\mathbf{y}_r$ the solution $\mathbf{s}$ becomes, if h_r is nonzero,

$$\mathbf{s} = (I - \mathbf{y}_r h_r^{-1}\mathbf{q}_r^*)K^{-1}\mathbf{t}.$$

The following identities are easily verified by expanding the products (cf. [50, lemma 6.1]):

$$\begin{aligned}(I - \mathbf{y}_r h_r^{-1}\mathbf{q}_r^*)(I - \mathbf{v}\mathbf{w}^*E) &= (I - \mathbf{v}\mathbf{w}^*E)\\ (I - \mathbf{v}\mathbf{w}^*E)(I - \mathbf{y}_r h_r^{-1}\mathbf{q}_r^*) &= (I - \mathbf{y}_r h_r^{-1}\mathbf{q}_r^*)\\ (I - \mathbf{y}_r h_r^{-1}\mathbf{q}_r^*)K^{-1}(I - E\mathbf{v}\mathbf{w}^*) &= (I - \mathbf{y}_r h_r^{-1}\mathbf{q}_r^*)K^{-1}\end{aligned}$$

To conclude, using these identities, the (left) preconditioned right correction equation in the bi-E-orthogonal case becomes

$$(I - \mathbf{y}_r h_r^{-1}\mathbf{q}_r^*)K^{-1}(A - \theta E)(I - \mathbf{y}_r h_r^{-1}\mathbf{q}_r^*)\mathbf{s} = -(I - \mathbf{y}_r h_r^{-1}\mathbf{q}_r^*)K^{-1}\mathbf{r}_v,$$

with $Q^*\mathbf{s} = \mathbf{w}^*E\mathbf{s} = 0$, and clearly the operator maps $(E^*\mathbf{w})^\perp$ onto $(E^*\mathbf{w})^\perp$. In a similar way, the (left) preconditioned left correction equation becomes

$$(I - \mathbf{y}_l h_l^{-1}\mathbf{q}_l^*)K^{-*}(A - \theta E)^*(I - \mathbf{y}_l h_l^{-1}\mathbf{q}_l^*)\mathbf{s} = -(I - \mathbf{y}_l h_l^{-1}\mathbf{q}_l^*)K^{-*}\mathbf{r}_w,$$

with $\mathbf{v}^*E^*\mathbf{s} = 0$ and where $\mathbf{q}_l = E\mathbf{v}$, $\mathbf{y}_l = K^{-*}E^*\mathbf{w}$, and $h_l = \mathbf{q}_l^*\mathbf{y}_l$. Note that in this case K^* was taken as preconditioner of $(A - \theta E)^*$, but that other choices are possible as well.

If the search spaces are kept orthogonal, the (left) preconditioned right correction equation (3.4.3) becomes

$$(I - \mathbf{y}_r h_r^{-1}\mathbf{q}_r^*)K^{-1}(A - \theta E)(I - \mathbf{y}_r h_r^{-1}\mathbf{q}_r^*)\mathbf{s} = -(I - \mathbf{y}_r h_r^{-1}\mathbf{q}_r^*)K^{-1}\mathbf{r}_v,$$

with $\mathbf{v}^*\mathbf{s} = 0$ and where $\mathbf{q}_r = \mathbf{v}$, $\mathbf{y}_r = K^{-1}E\mathbf{v}$, and $h_r = \mathbf{q}_r^*\mathbf{y}_r$. Similarly, the (left) preconditioned left correction equation (3.4.4) becomes

$$(I - \mathbf{y}_l h_l^{-1}\mathbf{q}_l^*)K^{-*}(A - \theta E)^*(I - \mathbf{y}_l h_l^{-1}\mathbf{q}_l^*)\mathbf{s} = -(I - \mathbf{y}_l h_l^{-1}\mathbf{q}_l^*)K^{-*}\mathbf{r}_w,$$

with $\mathbf{w}^*\mathbf{s} = 0$ and where $\mathbf{q}_l = \mathbf{w}$, $\mathbf{y}_l = K^{-*}E^*\mathbf{w}$, and $h_l = \mathbf{q}_l^*\mathbf{y}_l$.

Similar expressions can be derived for right preconditioned corrections equations [50, Section 6.2.4].

Unlike the equivalence of exact SADPA and BiJD in Section 3.4.2, there is no relation between inexact SADPA and BiJD similar to [74, proposition 5.5], where it is shown that BiJD and subspace accelerated two-sided RQI are equivalent if the (correction) equations are solved with m and $m + 1$ steps of BiCG, respectively.

3.6.2 Deflation

If $X \equiv X_k$ and $Y \equiv Y_k$ have as their columns the k found right and left eigenvectors of (A, E) and are normalized so that $Y_k^*EX_k = I$, then the pencil

$$((I - EXY^*)A(I - XY^*E), (I - EXY^*)E(I - XY^*E))$$

has the same eigentriplets as (A, E), but with the eigenvalues corresponding to the found eigenvectors transformed to 0. Following the steps in Section 3.6.1, with deflation, the preconditioned right correction equation for the bi-E-orthogonal case becomes (if H_r is nonsingular)

$$(I - Y_r H_r^{-1} Q_r^*) K^{-1} (A - \theta E)(I - Y_r H_r^{-1} Q_r^*)\mathbf{s} = -(I - Y_r H_r^{-1} Q_r^*)\mathbf{r}_v, \quad (3.6.1)$$

with $Q_r^*\mathbf{s} = 0$, and where $Q_r = [E^*Y, E^*\mathbf{w}]$, $Y_r = K^{-1}[EX, E\mathbf{v}]$ and $H_r = Q_r^* Y_r$. Similar expressions can be derived for the left correction equation and for the correction equations for the orthogonal case.

3.6.3 Computational notes

For each of the preconditioned correction equations, the matrices Q, Y, and H grow as the number of found eigentriplets increases. In the case of the right correction equation for the bi-E-orthogonal case (3.6.1), for example, this can be handled efficiently by storing $Q_r^k = E^*Y$ and $Y_r^k = K^{-1}EX$, so that every iteration only the last column $\mathbf{q}_r = E^*\mathbf{w}$ of Q_r and $\mathbf{y}_r = K^{-1}E\mathbf{v}$ of Y_r need to be computed. If additionally $H_r^k = Q_r^{k*} Y_r^k$ is stored, H_r can every iteration be computed as

$$H_r = \begin{bmatrix} H_r^k & Q_r^{k*}\mathbf{y}_r \\ \mathbf{q}_r^* Y_r^k & \mathbf{q}_r^*\mathbf{y}_r \end{bmatrix}.$$

Also, if the approximate solution $\tilde{\mathbf{s}}$ is computed with a Krylov solver and the initial guess satisfies $Q^*\mathbf{s}_0 = 0$, then $Q^*\tilde{\mathbf{s}} = 0$ as well. Applications with the preconditioned operator reduce to

$$(I - YH^{-1}Q^*)K^{-1}(A - \theta E)\mathbf{s}.$$

3.6.4 Numerical experiment

To illustrate the better performance of inexact BiJD, Figure 3.5 shows the convergence histories of inexact BiJD and SADPA for the BIPS system of Section 3.7. The linear (correction) equations were solved using 10 steps of GMRES, with preconditioner $LU = s_0 E - A$, where $s_0 = 1i$ was the initial shift. Both methods started with $\mathbf{v}_1 = (s_0 E - A)^{-1}\mathbf{b}$ and $\mathbf{w}_1 = (s_0 E - A)^{-*}\mathbf{c}$, and were restarted every $k_{max} = 10$ iterations, keeping the $k_{min} = 2$ most dominant approximates. Inexact BiJD finds the first pole $\lambda_1 \approx -0.03353 \pm 1.079i$ within a few iterations and eventually also finds a second dominant pole $\lambda_2 \approx -0.5208 \pm 2.881i$. Inexact SADPA, on the other hand, was not able to approximate λ_1 and corresponding eigenvectors up to tolerance $\epsilon = 10^{-8}$. The search spaces were kept bi-E-orthogonal. Similar experiments with orthogonal search spaces showed worse performance for both BiJD and SADPA.

Where BiJD converges more or less smoothly, inexact SADPA suffers from stagnation and hampering around $\|\mathbf{r}\|_2 \approx 10^{-5}$: this can be explained by the fact that the projected operators in the correction equations are better conditioned than

the operators in the linear DPA equations, especially if $\theta \to \lambda$. In that case, the condition number of $\theta E - A$ becomes unbounded, while the (effective) condition number of the projected operator remains bounded (see also [74, p. 159]). As a consequence, preconditioned GMRES may compute better approximate solutions of the correction equations. This advantageous behavior of the Jacobi-Davidson method is well known, see [51, 74, 139, 140, 154] for more details and options to optimize the JD process.

Note that most advantage is taken from the projected corrections equations in the final phase of convergence to an eigentriplet. In the initial phase, subspace acceleration and the initial shift help to create promising search spaces, and it is not expected (and not confirmed by experiments) that it makes a big difference whether the BiJD corrections equations or the DPA equations are solved, since both are solved only up to moderate accuracy. To take care of the ill conditioning in the final phase of convergence, it is not only more efficient to use the BiJD corrections equations, but also more natural (correspondence with two-sided Rayleigh quotient iteration) than to use projected variants of the DPA equations.

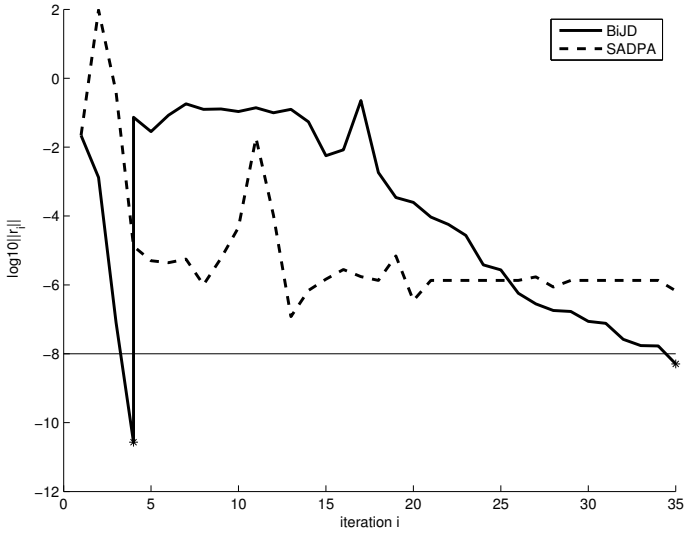


Figure 3.5: Convergence histories for inexact SADPA (dashed) and BiJD (solid), for the BIPS system of order $n = 13,251$. The linear equations were solved using 10 steps of GMRES with $LU = iE - A$ as preconditioner.

3.7 Numerical results

SADPA was tested on a number of systems, for a number of different input and output vectors **b** and **c**. Here the results for the Brazilian Interconnected Power

System (BIPS) are shown (see Section 3.3.2 for the application of SADPA to a small test system). The system data corresponds to a year 1999 planning model, having 2,400 buses, 3,400 lines, a large HVDC link, 123 power plants with detailed dynamic representation, 46 of which have power system stabilizers. The BIPS model is linearized around an operating point having a total load of 46,000 MW, with the North-Northeast generators exporting 1,000 MW to the South-Southeast Region, through the planned 500 kV, series compensated North-South intertie. The Power Oscillation Damping (POD) controllers of the two Thyristor Controlled Series Compensators (TCSC) are disabled, causing the low frequency North-South mode to become poorly damped ($\lambda_{ns} \approx -0.0335 + i1.0787$).

In the experiments, the convergence tolerance used was $\epsilon = 10^{-10}$. The spaces V and W were limited to dimension $n \times 10$ (i.e. a restart, with $k_{min} = 4$, every $k_{max} = 10$ iterations). The results are compared with the results of DPSE on quality of the modal equivalents, computed poles, CPU time and number of factorizations. DPSE uses deflation of complex conjugate pairs. All experiments were carried out in Matlab 7.3 [151] on a SUN Ultra 20 (AMD Opteron 2.8GHz, 2GB RAM).

The state-space realization of the BIPS model has 1664 states. The sparse, unreduced Jacobian has dimension 13251. Like the experiments in [91, 90], the practical implementation operates on the sparse unreduced Jacobian of the system, instead of on the dense state matrix A.

To demonstrate the performance of SADPA, it is applied to six transfer functions of BIPS to compute a number of dominant poles (complex conjugate pairs are counted as one pole). The first four transfer functions relate the rotor shaft speed deviations (W) of major synchronous generators to disturbances applied to the voltage references of their corresponding excitation control systems. Note that only a single shift, $s_1 = 1i$, is used: after the first pole has converged, the next most dominant approximate pole is used as new shift. The results are in Table 3.2 and the Bode plots of the corresponding modal equivalents, complete models and errors for the first four transfer functions are in Figure 3.6 and Figure 3.7. It is clearly observable that the reduced models capture the important dynamics. For completeness, the 15 most dominant poles and corresponding residues of $W_{6405}/Vref_{6405}$, according to the index $|R_i|/|\mathrm{Re}(\lambda_i)|$, are shown in Table 3.3.

It can also be observed in Table 3.2 that the $\|\hat{\mathbf{y}}_i\|_2 = \|\hat{\mathbf{x}}_i\|_2 = 1$ scaling leads to much better results than the $\hat{\mathbf{y}}_i^* E \hat{\mathbf{x}}_i = 1$ scaling. The decrease in the number of LU-factorizations and CPU time is significant. Although not shown here, both approaches find the most dominant poles (in a different order), and the modal equivalents are of comparable quality (except for $W_{6405}/Vref_{6405}$ and $W_{1107}/Vref_{1107}$, where 10 additional poles are needed). When using the scaling $\|\hat{\mathbf{y}}_i\|_2 = \|\hat{\mathbf{x}}_i\|_2 = 1$, the selection is based on the angles $\angle(\hat{\mathbf{x}}_i, \mathbf{c})$ and $\angle(\hat{\mathbf{y}}_i, \mathbf{b})$, which is in accordance with the results in Chapter 2. In the addendum of Chapter 4 a more detailed comparison is made.

In Table 3.3 there are also indications whether SADPA, DPSE and DPSE with deflation (DPSEd) were able to find the most dominant poles when given at most 35 seconds to compute 20 dominant poles. In Table 3.4, SADPA, DPSE and DPSE with deflation (DPSEd) are compared on computational time. SADPA succeeds in

Table 3.2: Results of SADPA for six transfer functions of the Brazilian Interconnected Power System (BIPS), with two types of scaling. Shift $s_1 = 1i$.

Transfer function	#poles	$\hat{\mathbf{y}}_i^* E \hat{\mathbf{x}}_i = 1$		$\|\hat{\mathbf{y}}_i\|_2 = \|\hat{\mathbf{x}}_i\|_2 = 1$	
		#LU	Time (s)	#LU	Time (s)
$W_{5061}/Vref_{5061}$	30	846	155	169	42
$W_{6405}/Vref_{6405}$	45	852	164	222	58
$W_{1155}/Vref_{1155}$	40	2229	462	256	65
$W_{1107}/Vref_{1107}$	40	816	158	217	54
P_{sc}/B_{sc}	50	2047	446	289	78
$Pt_{501}/Pref_{501}$	70	3436	855	317	87

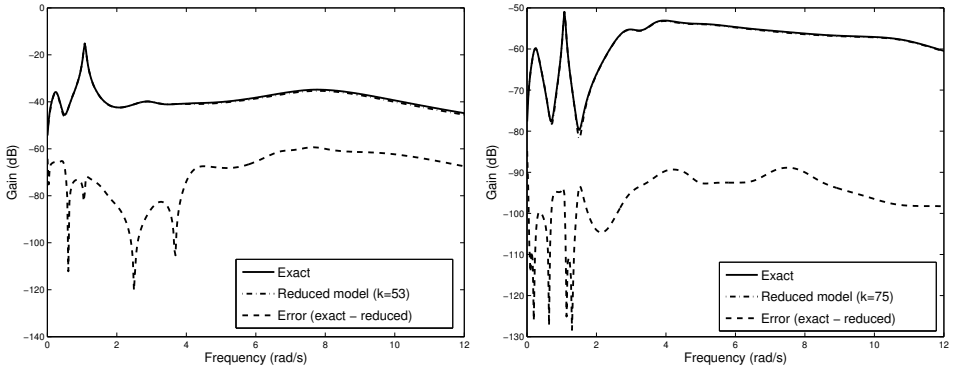


Figure 3.6: Bode plots of modal equivalent, complete model and error for transfer function $W_{5061}/Vref_{5061}$ (left, $k = 53$ states) and $W_{6405}/Vref_{6405}$ (right, $k = 75$ states) of BIPS (1664 states in the complete model).

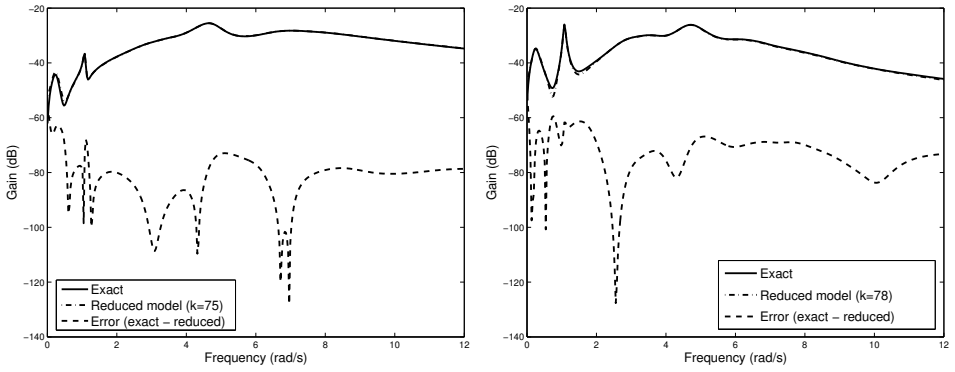


Figure 3.7: Bode plot of modal equivalent, complete model and error for transfer function $W_{1155}/Vref_{1155}$ (left, $k = 75$ states) and $W_{1107}/Vref_{1107}$ (right, $k = 78$ states) of BIPS (1664 states in the complete model).

Table 3.3: The 15 most dominant poles and corresponding residues of $W_{6405}/Vref_{6405}$, sorted in decreasing $|R_i|/|\mathrm{Re}(\lambda_i)|$ order (29-state modal equivalent). Poles found by SADPA, DPSE, and DPSEd are marked with a +.

Num. Mode	Modes	Residues		SADPA found	DPSE found	DPSEd found
		Magnitude	Phase			
1	$-0.0335 \pm 1.0787i$	$9.241 \cdot 10^{-5}$	± 173.0	+	+	+
3	$-0.5208 \pm 2.8814i$	$7.542 \cdot 10^{-4}$	± 139.7	+	+	+
5	$-0.5567 \pm 3.6097i$	$7.475 \cdot 10^{-4}$	± 102.9	+	+	+
7	$-2.9445 \pm 4.8214i$	$3.022 \cdot 10^{-3}$	± 167.7	+	+	+
9	$-0.1151 \pm 0.2397i$	$1.078 \cdot 10^{-4}$	± 176.9	+	+	+
11	$-6.4446 \pm 0.0715i$	$5.162 \cdot 10^{-3}$	± 144.0	+		+
13	-4.9203	$3.386 \cdot 10^{-3}$	0	+		
14	$-7.5118 \pm 0.2321i$	$3.625 \cdot 10^{-3}$	± 128.7	+		
16	$-2.3488 \pm 11.001i$	$1.014 \cdot 10^{-3}$	± 4.167	+	+	+
18	$-10.068 \pm 1.1975i$	$3.841 \cdot 10^{-3}$	± 17.57	+		+
20	$-1.4595 \pm 10.771i$	$5.305 \cdot 10^{-4}$	± 47.01	+	+	+
22	$-20.539 \pm 1.0930i$	$6.224 \cdot 10^{-3}$	± 65.00	+		
24	$-2.8052 \pm 11.551i$	$7.558 \cdot 10^{-4}$	± 8.385	+	+	
26	$-4.0233 \pm 4.2124i$	$1.009 \cdot 10^{-3}$	± 35.78	+	+	
28	$-0.7584 \pm 4.9367i$	$1.780 \cdot 10^{-4}$	± 146.0	+	+	+

Table 3.4: Computational statistics for SADPA ($s_1 = 1i$), the DPSE and the DPSE with deflation (DPSEd) (shifts $1i, 1.5i, \ldots, 10.5i$), when computing 20 dominant poles.

Method	#LU	time (s)	#poles	#repeated poles
SADPA	204	35	20	0
DPSE	160	25	20	3
DPSEd	193	35	17	0

finding both real and complex dominant poles, and, moreover, finds all of the 15 most dominant poles, while DPSE and DPSEd miss 5 of the most dominant poles. DPSE also recomputed already found poles three times. To make the comparison as fair as possible, the restart parameters for SADPA were $k_{min} = 1$ and $k_{max} = 10$, and the $\hat{\mathbf{y}}_i^* E \hat{\mathbf{x}}_i = 1$ scaling was used (SADPA was even faster for $k_{min} = 4$ and $\|\hat{\mathbf{y}}_i\|_2 = \|\hat{\mathbf{x}}_i\|_2 = 1$ scaling).

DPSE has more difficulties to converge as the number of shifts increases: typically, the tolerance is not reached for an increasing number of poles (like in SADPA, this can be (and was in the experiments) fixed by doing an additional step of two-sided Rayleigh quotient iteration). Besides that, also the computational costs increase rapidly as the number of shifts increases, because the interaction matrices grow. To compute 60 poles with DPSE, it has to be started again with different sets of shifts, which has the risk of computing the same poles again, is time consuming and requires human interaction. SADPA, however, finds 60 dominant poles, starting with just one single shift, without any human interaction during the process. Because of the selection strategy, truly dominant poles are computed; the deflation

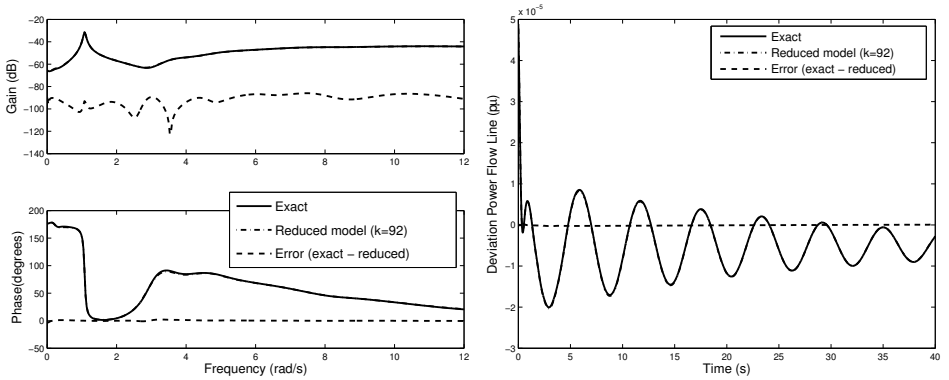


Figure 3.8: Bode plot (left) and step response (right) of modal equivalent, complete model and error for transfer function $P_{sc}(s)/B_{sc}(s)$ of BIPS (92 states in the modal equivalent, 1664 in the complete model).

strategy prevents repeated computation of the same pole. Furthermore, SADPA succeeds in computing real poles, while DPSE has many difficulties in computing real poles: real shifts are needed for DPSE, while SADPA finds the real poles automatically. SADPA is not sensitive to the initial shift: repeated experiments with other shifts give the same results.

The fifth transfer function of BIPS is P_{sc}/B_{sc}, relating the active power deviations flowing through the North-end series capacitor of the planned intertie, to disturbances in the reference value of the TCSC series admittance. This transfer function was used in the basic design of the POD controllers of the two TCSCs, in order to damp the North-South mode [32, 60, 94]. Modal equivalents of the transfer function for damping the North-South mode, whose state-space realization has a direct transmission term $d = 4.88 \cdot 10^{-3}$, are considered in [94]. Figure 3.8 (left) shows the frequency response of the complete model and the reduced model (92 states) together with the error. Figure 3.8 (right) shows the corresponding step response (step 0.01)[2]. The North-South mode (1 rad/s = 0.17 Hz) is well observable in both responses, and the reduced model nicely captures the system oscillations. The reduced model (50 poles, 92 states) was computed by SADPA in 78 seconds (289 factorizations). Occasional corruption of the search spaces by components in the direction of eigenvectors corresponding to eigenvalues at infinity was successfully limited by keeping these components in the search spaces at restarts. A table with the dominant poles and corresponding residues can be found in [94].

The sixth transfer function, $Pt_{501}/Pmec_{501}$, relates the active power deviations of a large hydro-electric plant, located in the Southeast region, to disturbances applied to its speed-governor reference. For this transfer function it is known from numerical experiments that DPSE has difficulties in producing a good modal equivalent: several DPSE attempts are needed to obtain an acceptable modal equivalent.

[2]If $h_k(t)$ is the inverse Laplace transform of $H_k(s)$ (3.2.4), the step response for step $u(t) = c$ of the reduced model is given by $y(t) = \int_0^t h(t)u(t) = c(\sum_{i=1}^k \mathrm{Re}((\frac{R_i}{\lambda_i}(\exp(\lambda_i t) - 1)) + d)$.

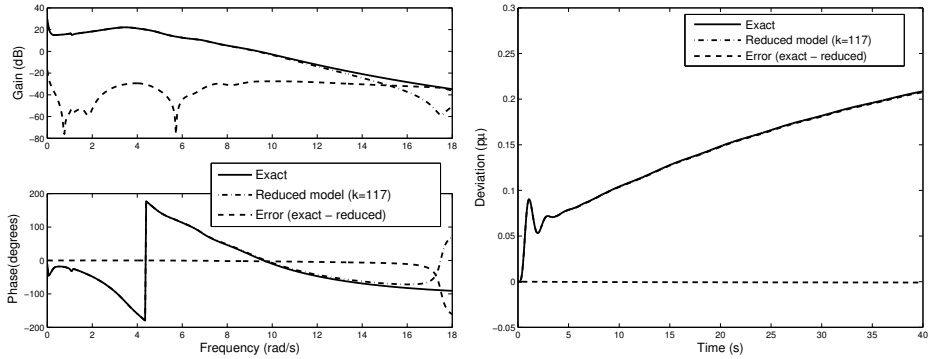


Figure 3.9: Bode plot (left) and step response (right) of modal equivalent, complete model and error for transfer function $Pt_{501}(s)/Pref_{501}(s)$ of BIPS (117 in the modal equivalent, 1664 in the complete model).

SADPA is able to *automatically* produce good results for both the frequency and step response, as can be observed from Figure 3.9. SADPA computed the reduced model (70 poles, 117 states) in 87 seconds (317 factorizations) using just a single initial shift $s_1 = 1i$, in one single run.

It should be noted that all modal equivalents in this section can be reduced even further by neglecting less dominant contributions, or by application of balanced truncation to a state-space realization of the modal equivalent.

3.8 Reflection

SADPA is primarily an algorithm for the computation of dominant poles of large-scale transfer functions. The corresponding left and right eigenvectors, that are computed as well by SADPA, can be used to construct reduced-order models of the original system in the form of modal equivalents. Although these modal equivalents are of practical value, both for other (transient) simulations with and to verify (via the Bode plot) whether all dominant poles are found, there are two points of concern in comparison with Krylov subspace projection based model reduction techniques that immediately come to front: first, in exact SADPA, every iteration exact solves with $s_k E - A$ and $(s_k E - A)^*$ are required (via a single LU-factorization), and second, there is no moment matching property for the modal equivalents produced by SADPA. The first point can be overcome by using inexact solves, as described in Section 3.6, but in this section it is assumed that exact solves are feasible (recall that SADPA is the method of choice in this case). This assumption is reasonable, since Krylov based methods also require at least exact solves with $\sigma_0 E - A$ for one σ_0 (usually via LU-factors). Instead of matching moments, the poles of the modal equivalent are poles of the original system [18] and hence modal approximation preserves stability automatically and is applicable to both stable and unstable systems. Although the exact location (except for the sign of the real part) of poles is not important in all kinds of analysis of dynamical

systems, it is important in for instance vibration analysis [18], where the poles correspond to resonance frequencies of the original system, and damping control of electromechanical oscillations in power systems [90, 91, 93], where stabilizers are designed to damp specific dominant poles.

The goal of this section is not to argue which one of modal approximation and Krylov based methods is the best, but to show the qualities and limitations of both methods, and how one method may help the other to deal with or even overcome its limitations. Krylov based methods are widely described in the literature, see for instance [5, part IV] and references therein. Modal approximation methods, on the other hand, are often mentioned briefly or even not, and if mentioned, focus is usually on the limitations (see for instance [5, Section 9.2]). It may be the case that this typical focus is caused by the availability of many Krylov based schemes such as PRIMA [105] and rational Arnoldi and Lanczos, while practical modal approximation methods, other than computing a *full* eigendecomposition and selecting the dominant eigenvectors [159], were not available. SADPA is novel in the sense that it computes the dominant poles measured in (scaled) residue norm, instead of nearness to the imaginary axis, as is done in some literature [5, p. 283], and that it computes these poles automatically in a single run, together with the left and right eigenvectors, that can be used to construct the modal equivalent. SADPA succeeds in finding the dominant poles even if good initial estimates are not available and does not require a full (block) diagonalization of (A, E). This allows for a comparison between modal approximation by SADPA and Krylov based methods. Theoretical considerations are illustrated by numerical examples.

Of the Krylov subspace projection methods for model reduction, rational interpolation is probably one of the most effective techniques nowadays (see Section 1.5.3, and see [70] and [5] for good introductions). Contrary to the single interpolation point methods PRIMA [105] and PVL [48], rational interpolation uses multiple interpolation points and constructs bases for rational Krylov subspaces instead of ordinary Krylov subspaces. The big advantage of rational interpolation is that moments are matched around several distinct interpolation points, leading to much better and smaller reduced-order models than conventional Krylov based methods [58]. If good interpolation points are known (and the number of moments to match around each point), then the availability of efficient codes for the construction of ((bi)-orthogonal) bases for the rational Krylov subspaces makes rational interpolation an efficient and effective approach. If, on the other hand, there is no or limited knowledge of good interpolation points, rational interpolation becomes less effective. Although there are techniques for dynamically selecting and adding interpolation points, see [58] and [70, Chapter 6], rational interpolation may not succeed in capturing (some of) the dominant behavior, typically visible in missing peaks in the Bode plot.

These problems are inherent to the use of Padé approximations and (rational) Krylov subspaces [58, p. 39]. Padé approximations are exact at the point of interpolation σ, while accuracy is lost away from σ, even more rapidly if σ is close to a pole [14, 30]. The methods used to construct the bases for the rational Krylov subspaces are rational Bi-Lanczos [58] and dual rational Arnoldi [70, Section 4.1.2].

These methods typically tend to approximate the eigenvalues at the outer edge of the spectrum of $(A-\sigma E)^{-1}E$, converging to well-separated eigenvalues at the edge first. Practically speaking this means that the interpolation points should be chosen in such a way that the dominant poles become well separated eigenvalues at the edge of the spectrum of $(A-\sigma E)^{-1}E$. Without knowledge of the location of the dominant poles, probably the best one can do is to logarithmically space the real interpolation points between ω_{min} and ω_{max} to obtain the general frequency response, and to add purely imaginary interpolation points to capture the exact response at frequencies of interest, as proposed in [70, Chapter 6]: real interpolation points σ tend to transform the eigenvalues $|\lambda_i| \approx \sigma$ to the outer edge of the spectrum of $(A-\sigma E)^{-1}E$, leading to reduced models that capture the global response, while purely imaginary interpolation points σ transform eigenvalues $\lambda_i \approx \sigma$ to the outer edge of spectrum, leading to locally good results around frequencies $\omega \approx \text{imag}(\sigma)$.

Based on the properties of the rational Krylov subspace based methods, one may expect that these methods work well for transfer functions with relatively smooth frequency response: as is also confirmed by numerical experience, real interpolation points suffice in many cases to obtain good reduced-order models. As an illustration, consider the Bode plots of two reduced models of $W_{6405}/Vref_{6405}$ (see Section 3.7) in Figure 3.10. The reduced-order model computed by dual Arnoldi with a single interpolation point $\sigma_0 = 0$ is not able to capture the frequency response for the range of interest, even not for a model of order 85. Adding a second point $\sigma_1 = 1$ leads to a better result, but more interpolation points are needed to obtain an exact match. Although not shown here, reduced-order models of similar sizes produced by PRIMA are of worse quality, since PRIMA computes a basis for the Krylov space $\mathcal{K}^k((\sigma_0 E-A)^{-1}E, (\sigma_0 E-A)^{-1}\mathbf{b})$ and neglects the output vector $\mathbf{c} \neq \mathbf{b}$ (and matches only k moments, while the (two-sided) rational k-dimensional models match $2k$ moments, albeit at the cost of $2k$ matrix vector multiplications against k for PRIMA).

The situation becomes different if the frequency response shows a large number of peaks, caused by (relatively) dominant poles with small real part, close to each other, or if dominant poles are not transformed to the outer edge of the spectrum due to the choice of interpolation points, and consequently are not present in the reduced model. SADPA, on the other hand, is expected to be able to handle both smooth and less smooth transfer functions, since it computes the most dominant poles and constructs modal equivalents using the corresponding left and right eigenspaces. Inherent to modal approximation, however, is that the reduced model is accurate for frequencies in the neighborhood of the imaginary parts of the dominant poles, while less accurate for distant frequencies (typically the smooth parts of the response). The following two examples illustrate these issues and also indicate that rational Krylov methods and SADPA can be combined to deal with these problems.

The model of a portable compact disc player, where the control task is to point the lens of the laser to the track of pits of the rotating CD, is also considered in [58, 70] and is available via [28]. Figure 3.11 shows the frequency response for

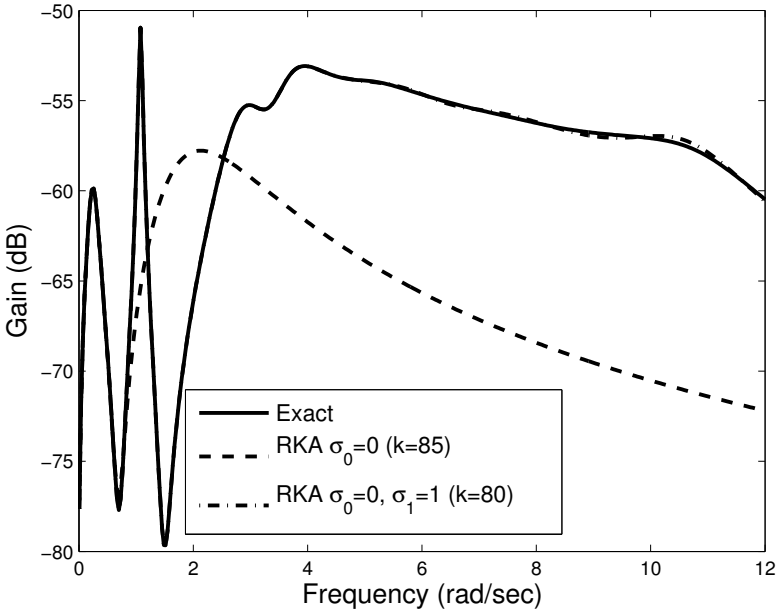


Figure 3.10: Exact frequency response of $W_{6405}/Vref_{6405}$, and responses of a 85-th order RKA model (interpolation point $\sigma_0 = 0$) and a 80-th order RKA model ($\sigma_0 = 0, \sigma_1 = 1$).

the second input and the first output (the 120-th order system has 2 inputs and 2 outputs), together with the responses of 60-th order reduced models computed by SADPA and dual rational Arnoldi (RKA, the implementation of [70, Appendix B] was used). SADPA needed 203 iterations (LU-factorizations) to compute 30 dominant poles and construct the real modal equivalent ($s_0 = 1i$, $k_{min} = 1$, $k_{max} = 10$, $1.4s$ CPU time), while RKA generated a 60-th order real model by matching 10 moments around each of the 12 logarithmically spaced points in the interval $[1, 5 \cdot 10^5]$ within $0.4s$ CPU time (to match p moments, $p/2$ left and right basis vectors are needed). Note that only a single initial estimate $s_0 = 1i$ was needed for SADPA. In the "eye-norm" there is hardly any difference: both frequency responses seem to match exactly. The relative error in Figure 3.12 shows a different picture. For frequency ranges where the response is relatively smooth or only has isolated peaks, the RKA model is orders of magniture more accurate. If there are many peaks close to each other, the SADPA model is more accurate, while the RKA response even has relative errors of order 1 (see near $\omega = 10^4$ rad/s). This example confirms that Krylov based models capture the global dynamics accurately while missing important details, and that modal equivalents are locally accurate, while only moderately accurate for frequencies away from the imaginary parts of dominant poles. It also suggests that the modal equivalent may be improved by adding a small part of the Krylov subspaces to left and right eigenspaces, and that RKA model may be improved by adding some of the left and right eigenvectors to the Krylov subspaces. These and other possibilities will be discussed in more

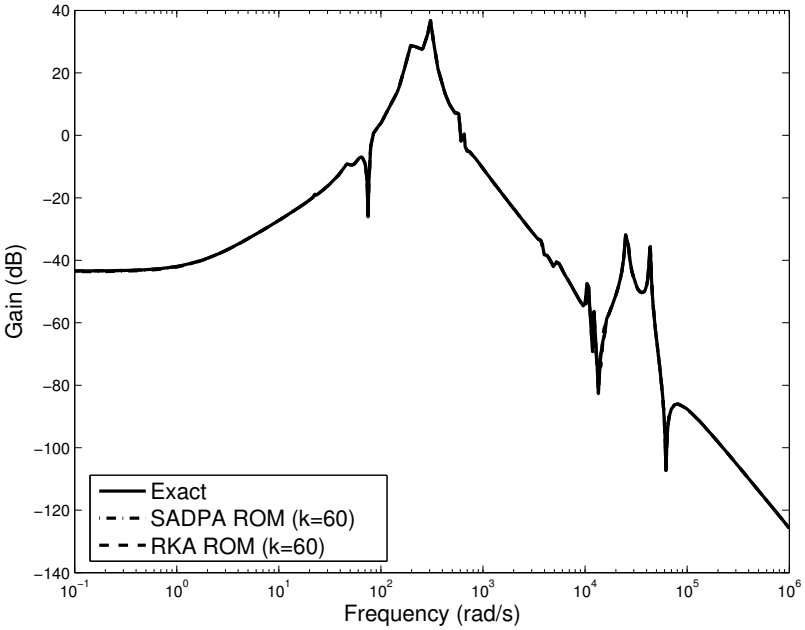


Figure 3.11: Exact frequency response of the CD player model, and respones of 60-th order approximations computed by SADPA (dash-dot) and dual rational Arnoldi (RKA, dash).

detail after the following example.

The PEEC model arises from a partial element equivalent circuit (PEEC) of a patch antenna structure, and the dimension of the matrices A and E is $n = 480$ (see [28, 129, 161] for more details and the model data). The PEEC model is known as a difficult problem, because it has many poles close to each other [73]. The frequency response is shown in Figure 3.13, together with the frequency responses of a 89-th order model computed by SADPA, and a 90-th order real model computed by RKA. SADPA needed 222 iterations (LU-factorizations) to compute 45 dominant poles ($s_0 = 96i$, $k_{min} = 4$, $k_{max} = 10$, $38s$ CPU time), leading to a 89-th order reduced real model. RKA generated a 90-th order real model by matching 2 moments around each of the 90 logarithmically spaced imaginary points in the interval $i[0.1, 10^5]$ within $41s$ CPU time (82 states after removing dependencies in the bases [70, Section 3.3.2]). Compared to the results of the CD player model, three important observations can be made. Firstly, the frequency responses of the SADPA and RKA models differ from the exact response in the "eye-norm" (cf. $\omega > 200$ rad/s for SADPA and RKA, and $\omega \approx 8$ rad/s for RKA). Secondly, the relative errors (Figure 3.14) for SADPA and RKA differ only by 2 orders for smooth parts of the frequency response. Thirdly, the SADPA model is much more accurate in the interval $[1, 10]$ rad/s: in the "eye-norm" the match is exact, and the relative error is much smaller. As was also reported in [73, p. 146], it is difficult to improve the RKA model by using even more interpolation points. SADPA, on the other hand, finds the dominant poles automatically without any

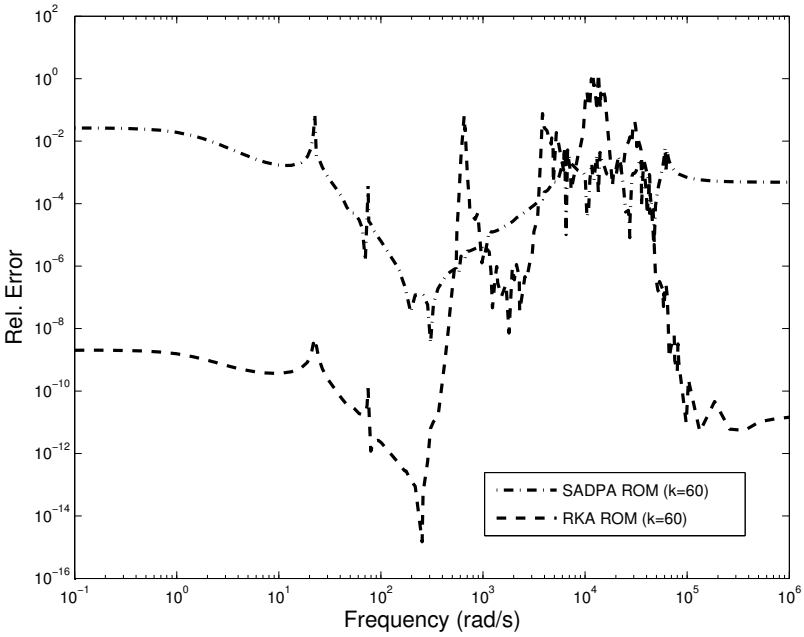


Figure 3.12: Relative error in the frequency response of the CD player model for the 60-th order approximations computed by SADPA (dash-dot) and dual rational Arnoldi (RKA, dash).

human interaction or a priori knowledge of the locations.

A straightforward approach to improve a modal equivalent computed by SADPA, is to (orthogonally) expand the (real) bases Y and X for the left and right dominant eigenspaces with a small number of basis vectors computed by the rational Krylov subspace method. If the columns of W and V form a basis for the left and right rational Krylov spaces, then the extended reduced model can be constructed by using $Z = [Y, W]$ and $Q = [X, V]$ in $(Z^*EQ, Z^*AQ, Z^*\mathbf{b}, Q^*\mathbf{c}, d)$. Note that this new model preserves the exact poles of the modal equivalent (that are exact poles of the original model as well), and also preserves the matched moments due to W and V, since $\text{span}(W) \subset \text{span}(Z)$ and $\text{span}(V) \subset \text{span}(Q)$. In Figure 3.15 the relative error for the original 60-th order SADPA model is shown together with the relative errors for two extended models. The 66-th order model was constructed by expanding the eigenspaces with six basis vectors for the rational Krylov subspaces with interpolation points $\{1, 10, 100, 10^3, 10^4, 10^5\}$. The 72-nd order model was constructed by expanding the eigenspaces with 6 six basis vectors for the rational Krylov subspaces with interpolation points $\{1, 5, 10, 50, 100, 500, 10^3, 5 \cdot 10^3, 10^4, 5 \cdot 10^4, 10^5, 5 \cdot 10^5\}$ (matching two moments around each point in both cases). Note that reduced models based on only these Krylov spaces are of poor quality. The improvement of the modal equivalent is apparent, and similar results can be reported for the PEEC model, although less dramatic, since the modal equivalent already has a small relative error. In practice, usually a naive choice for the interpolation points (logarithmically spaced) and 2

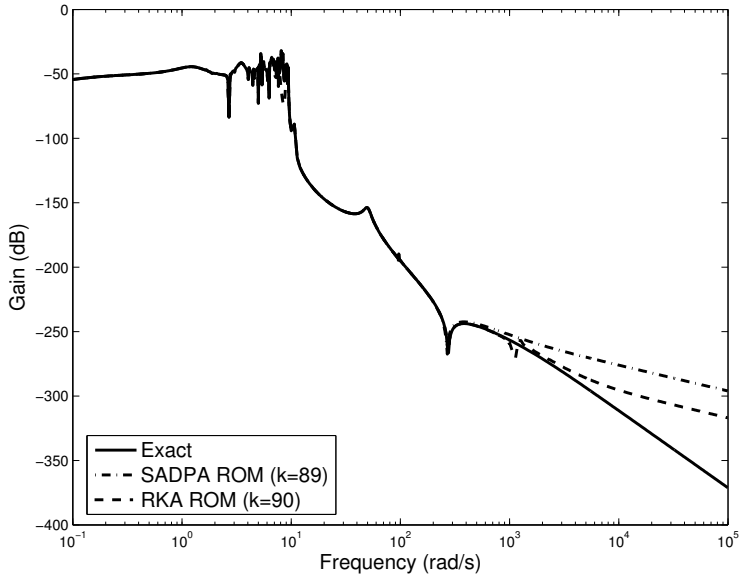


Figure 3.13: Exact frequency response of the PEEC model, and responses of 89-th order model computed by SADPA (dash-dot) and 90-th order model computed by dual rational Arnoldi (RKA, dash).

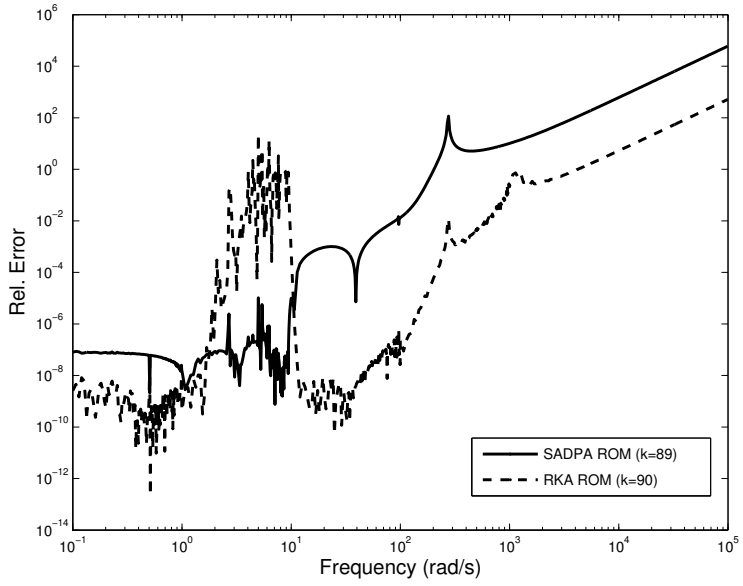


Figure 3.14: Relative error in the frequency response of the PEEC model for the 89-th order model computed by SADPA (solid) and 90-th order model computed by dual rational Arnoldi (RKA, dash)

matched moments per points suffice to improve the modal equivalent significantly: this will improve the global frequency response, while the local details are already covered by the dominant modes found by SADPA.

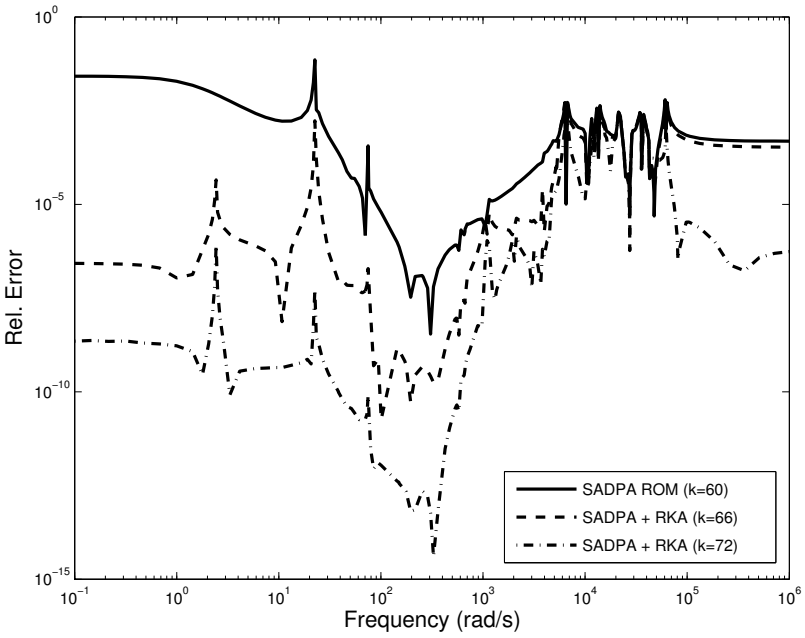


Figure 3.15: Relative error in the frequency response of the CD player model for the 60-th order approximation computed by SADPA (solid), and for 66-th (dash) and 72-th order (dash-dot) extended models, constructed by adding 6 and 12 dual rational Arnoldi basis vectors, respectively.

The other way around, reduced models constructed by rational Krylov methods can be improved by expanding the spaces with left and right eigenvectors corresponding to dominant poles. Again, this does not violate the moment matching properties and also leaves the found poles unchanged. This approach is useful if some isolated peaks are missed by the reduced model: the corresponding frequencies are good estimates for the missing poles and can be used as initial estimates for SADPA. For the CD player and PEEC models this approach is less practical, because most likely a large number of dominant poles has to be added, for which human interaction is needed. Vice versa, the imaginary parts of dominant poles computed by SADPA may be good interpolation points for the rational Krylov methods [70, Chapter 6]; an example of this can be found in Section 6.4.2 of Chapter 6. If exact solves are not feasible, inexact two-sided Jacobi-Davidson (Section 3.6) can be used to compute a few dominant poles. Yet another option is to use rational Krylov spaces as initial search spaces for SADPA.

The idea of improving modal equivalents by expanding the eigenspaces with rational Krylov spaces also sheds new light on the results in Section 3.7. As is also observed in experiments for other large-scale systems, SADPA tends to converge to the most dominant poles rather quickly, but stagnates when the remaining poles are

more or less equally dominant. In other words, SADPA smoothly computes modal equivalents that capture the peaks of the frequency response, but has difficulties in matching the response between peaks. These observations suggest to compute the most dominant poles with SADPA, and to expand the eigenspaces with relatively small rational Krylov spaces for a small number of interpolation points, as described before.

In Figure 3.16 the relative errors in the frequency response of $W_{6405}/Vref_{6405}$ are shown for an 18-th order RKA model (6 moments around each of the points $\{0, 2, 4, 6, 8, 10\}$), a 36-th order SADPA modal equivalent (20 (complex conjugate pairs of) poles), a 37-th order SADPA + RKA model (19 left and right eigenvectors (10 poles) + 18 RKA vectors), and 54-th order SADPA + RKA (36 left and right eigenvectors (20 poles) + 18 RKA vectors) model (cf. Figures 3.6 and 3.10). The two small models are not very accurate, although one can recognize that the SADPA model captures the details, while the RKA model makes a more global match. The combined models have the best of both and lead to acceptable reduced models, of better quality and computed within less time than the large SADPA models in Section 3.7, since the construction of the RKA bases requires only 4 seconds.

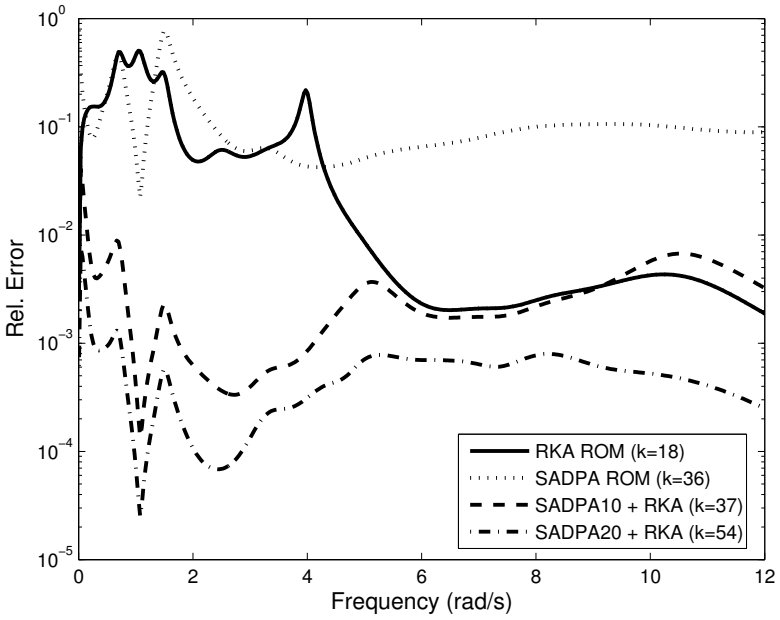


Figure 3.16: Relative errors in the frequency response of $W_{6405}/Vref_{6405}$ for 18-th order RKA model, 36-th order SADPA modal equivalent, 37-th order SADPA(10 poles, 19 states) + RKA(18) model and 54-th order SADPA(20,36) + RKA(18) model.

3.9 Conclusions

The algorithm described in this chapter, SADPA, is a fast method to compute the dominant poles and corresponding eigenvectors of a scalar transfer function. It has several advantages over existing methods. Firstly, it is more robust because it uses a natural selection method to converge to both real and complex dominant poles. Secondly, SADPA needs less iterations to converge by using subspace acceleration. Thirdly, it has less risk of missing a dominant pole, and of computing an already found pole, because of effective and efficient deflation techniques. Fourthly, SADPA is completely automatic: with a single initial estimate it is able to compute as many dominant poles as wanted, without intermediate human interaction.

It was also shown that if the linear (correction) equations are solved exactly, SADPA is equivalent to the two-sided Jacobi-Davidson method. If it is feasible to solve the linear (correction) equations exactly, using for instance LU-factorizations, then SADPA is the method of choice since the costs per iteration are less, and deflation can be done in an efficient and stable way via the input and output vectors (only a one time deflation is needed per found eigentriplet). If, on the other hand, exact solves are not feasible and one has to use iterative methods like GMRES, then two-sided Jacobi-Davidson is the preferred method, because the approximate solutions of the correction equations lead to better expansions of the search spaces and better convergence.

The modal equivalents constructed using the left and right eigenvectors of the dominant poles computed by SADPA, can be of use for further analysis of the original system. It was shown how rational Krylov methods can be used to improve the modal equivalents at low costs. The dominant poles computed by SADPA can be used to determine interpolation points for rational Krylov methods.

Addendum

The first version of SADPA (without the efficient deflation as described in Section 3.3.1) was presented in [123]

> Joost Rommes and Nelson Martins, *Efficient computation of transfer function dominant poles using subspace acceleration*, IEEE Transactions on Power Systems **21** (2006), no. 3, 1218–1226,

but this chapter is rewritten, reorganized and extended with new results, relations to Jacobi-Davidson and rational Krylov, additional numerical experiments, and reflections on rational Krylov based model reduction methods.

Chapter 4

Efficient computation of multivariable transfer function dominant poles

Abstract. This chapter describes a new algorithm to compute the dominant poles of a high-order multi-input multi-output (MIMO) transfer function. The algorithm, called the Subspace Accelerated MIMO Dominant Pole Algorithm (SAMDP), is able to compute the full set of dominant poles efficiently. SAMDP can be used to produce good modal equivalents automatically. The general algorithm is robust, applicable to both square and non-square transfer function matrices, and can easily be tuned to suit different practical system needs.

Key words. small-signal stability, poorly-damped oscillations, power system dynamics, transfer function, system poles, model reduction, dominant pole spectrum, large-scale systems, sparse eigenanalysis, modal equivalents, modal analysis, multivariable systems, transfer function residues

4.1 Introduction

Current model reduction techniques for power system stability analysis and controller design [134, 108, 29, 119, 153] produce good results but are either not applicable or require excessive computational effort to deal with large scale problems. If only a small part of the system pole spectrum is controllable-observable for the transfer function, a low-cost alternative for large-scale systems is modal model reduction. Modal reduction approximates the transfer function by a modal equivalent that is computed from the dominant poles and their corresponding residues. To produce a good modal equivalent, specialized eigensolution methods are needed. An algorithm that automatically and efficiently computes the full set of dominant poles of a scalar transfer function was presented recently [123] (see Chapter 3), but existing methods for multi-input multi-output (MIMO) transfer functions [93] are not capable enough to produce good modal equivalents automatically. A sur-

vey on model reduction methods employing either singular value decompositions or moment matching based methods is found in [6, 5]. Practically all examples provided in [5], a recent and valuable reference of model reduction of multivariable systems, have less than one thousand states. An introduction on modal model reduction on state-space models can be found in [68], while [159] describes a possible enhancement to modal model reduction. However, the authors believe that modal model reduction has been largely neglected (see [5], for example) by the engineering community, mostly due to the lack of reliable eigensolution algorithms.

In this chapter, a new extension of the Subspace Accelarated Dominant Pole Algorithm (SADPA) [123] will be proposed: Subspace Accelerated MIMO Dominant Pole Algorithm (SAMDP). The SADPA is a generalization of the Dominant Pole Algorithm [91], that automatically computes a high quality modal equivalent of a transfer function. The SAMDP can also be seen as a generalization of the MIMO Dominant Pole algorithm [93]. SAMDP computes the dominant poles and corresponding residue matrices one by one by selecting the most dominant approximation every iteration. This approach leads to a faster, more robust and more flexible algorithm. To avoid repeated computation of the same dominant poles, a deflation strategy is used. The SAMDP directly operates on implicit state space systems, also known as descriptor systems, which are very sparse in practical power system applications.

The chapter is organized as follows. Section 4.2 summarizes some well known properties of MIMO transfer functions and formulates the problem of computing the dominant poles of a MIMO transfer function. Section 4.3 describes the new SAMDP algorithm. In Section 4.4, numerical aspects concerning practical implementations of SAMDP are discussed. Extensive numerical results are presented in 4.5. Section 4.6 concludes.

4.2 MIMO transfer functions, sigma plots and dominant poles

For a multi-input multi-output (MIMO) system

$$\left\{ \begin{array}{rcl} \dot{\mathbf{x}}(t) & = & A\mathbf{x}(t) + B\mathbf{u}(t) \\ \mathbf{y}(t) & = & C^*\mathbf{x}(t) + D\mathbf{u}(t), \end{array} \right. \tag{4.2.1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{n\times p}$, $\mathbf{x}(t) \in \mathbb{R}^n$, $\mathbf{u}(t) \in \mathbb{R}^m$, $\mathbf{y}(t) \in \mathbb{R}^p$ and $D \in \mathbb{R}^{p\times m}$, the transfer function $H(s) : \mathbb{C} \longrightarrow \mathbb{C}^{p\times m}$ is defined as

$$H(s) = C^*(sI - A)^{-1}B + D, \tag{4.2.2}$$

where $I \in \mathbb{R}^{n\times n}$ is the identity matrix and $s \in \mathbb{C}$. Without loss of generality, $D = 0$ in the following.

It is well known that the transfer function of a single-input single-output system is defined by a complex number for any frequency. For a MIMO system, the transfer function is a $p \times m$ matrix and hence does not have a unique gain for a given

frequency. The SISO concept of a single transfer function gain must be replaced by a range of gains that have an upper bound for non-square matrices $H(s)$, and both upper and lower bounds for square matrices $H(s)$. Denoting the smallest and largest singular values [64] of $H(i\omega)$ by $\sigma_{\min}(\omega)$ and $\sigma_{\max}(\omega)$, it follows for square $H(s)$ that

$$\sigma_{\min}(\omega) \leq \frac{||H(i\omega)\mathbf{u}(i\omega)||_2}{||\mathbf{u}(i\omega)||_2} \leq \sigma_{\max}(\omega),$$

i.e. for a given frequency ω, the gain of a MIMO transfer function is between the smallest and largest singular value of $H(i\omega)$, which are also called the smallest and largest principal gains [89]. For non-square transfer functions $H(s)$, only the upper bound holds. Plots of the smallest and largest principal gains against frequency, also known as sigma plots, are used in the robust control design and analysis of MIMO systems [89].

Let the eigenvalues (poles) of A and the corresponding right and left eigenvectors be given by the triplets $(\lambda_j, \mathbf{x}_j, \mathbf{y}_j)$, and let the right and left eigenvectors be scaled so that $\mathbf{y}_j^*\mathbf{x}_j = 1$. Note that $\mathbf{y}_j^*\mathbf{x}_k = 0$ for $j \neq k$. The transfer function $H(s)$ can be expressed as a sum of residue matrices $R_j \in \mathbb{C}^{p\times m}$ over first order poles [78]:

$$H(s) = \sum_{j=1}^{n} \frac{R_j}{s - \lambda_j},$$

where the residue matrices R_j are

$$R_j = (C^*\mathbf{x}_j)(\mathbf{y}_j^*B).$$

A pole λ_j that corresponds to a residue R_j with relatively large $\|R_j\|_2/|\mathrm{Re}(\lambda_j)| = \sigma_{\max}(R_j)/|\mathrm{Re}(\lambda_j)|$ is called a dominant pole, i.e. a pole that is well observable and controllable in the transfer function. This can also be observed from the corresponding $\sigma_{\max}$-plot of $H(s)$, where peaks occur at frequencies close to the imaginary parts of the dominant poles of $H(s)$. An approximation of $H(s)$ that consists of $k < n$ terms with $||R_j||_2/|\mathrm{Re}(\lambda_j)|$ above some value, determines the effective transfer function behavior [144] and will be referred to as transfer function modal equivalent:

$$H_k(s) = \sum_{j=1}^{k} \frac{R_j}{s - \lambda_j},$$

Because a residue matrix R_j is the product of a column vector and a row vector, it is of unit rank. Therefore at least $\min(m, p)$ different poles are needed to obtain a modal equivalent with nonzero $\sigma_{\min}(\omega)$ plot [93, 114].

The problem of concern can now be formulated as: Given a MIMO linear, time invariant, dynamical system (A, B, C, D), compute $k \ll n$ dominant poles λ_j and the corresponding right and left eigenvectors $\mathbf{x}_j$ and $\mathbf{y}_j$.

4.3 Subspace Accelerated MIMO Dominant Pole Algorithm (SAMDP)

The subspace accelerated MIMO dominant pole algorithm (SAMDP) is based on the dominant pole algorithm (DPA) [91], the subspace accelerated DPA (SADPA) [123] (see also Chapter 3) and the MIMO dominant pole algorithm (MDP) [93]. First, a Newton scheme will be derived for computing the dominant poles of a MIMO transfer function. Then, the SAMDP will be formulated as an accelerated Newton scheme, using the same improvements that were used in the robust SADPA algorithm.

All algorithms are described as directly operating on the state-space model. The practical implementations (see Section 4.4.1) operate on the sparse descriptor system model, which is the unreduced Jacobian for the power system stability problem, analyzed in the examples of this chapter (see Section 4.5). Matlab implementations of the algorithms are presented in the appendix.

4.3.1 Newton scheme for computing dominant poles

The dominant poles of a MIMO transfer function $H(s) = C^*(sI - A)^{-1}B$ are those $s \in \mathbb{C}$ for which $\sigma_{\max}(H(s)) \to \infty$. For square transfer functions ($m = p$), there is an equivalent criterion: the dominant poles are those $s \in \mathbb{C}$ for which $\lambda_{\min}(H^{-1}(s)) \to 0$. In the following it will be assumed that $m = p$; for general MIMO transfer functions, see 4.4.3.

The Newton method can be used to find the $s \in \mathbb{C}$ for which the objective function

$$f : \mathbb{C} \longrightarrow \mathbb{C} : s \longmapsto \lambda_{\min}((C^*(sI - A)^{-1}B)^{-1}) \tag{4.3.1}$$

is zero. Let $(\mu(s), \mathbf{u}(s), \mathbf{z}(s))$ be an eigentriplet of $H^{-1}(s) \in \mathbb{C}^{m\times m}$, so that $H^{-1}(s)\mathbf{u}(s) = \mu(s)\mathbf{u}(s)$ and $\mathbf{z}^*(s)H^{-1}(s) = \mu(s)\mathbf{z}^*(s)$, with $\mathbf{z}^*(s)\mathbf{u}(s) = 1$. The derivative of $\mu(s)$ is given by [103]

$$\frac{d\mu}{ds}(s) = \mathbf{z}^*(s)\frac{dH^{-1}}{ds}(s)\mathbf{u}(s), \tag{4.3.2}$$

where

$$\begin{aligned}\frac{dH^{-1}}{ds}(s) &= -H^{-1}(s)\frac{dH}{ds}(s)H^{-1}(s) \\ &= H^{-1}(s)C^*(sI - A)^{-2}BH^{-1}(s).\end{aligned} \tag{4.3.3}$$

Note that it is assumed that $H^{-1}(s)$ has distinct eigenvalues and that the function that selects $\mu_{\min}(s)$ has derivative 1. Substituting (4.3.3) in (4.3.2), it follows that

$$\begin{aligned}\frac{d\mu}{ds}(s) &= \mathbf{z}^*(s)H^{-1}(s)C^*(sI - A)^{-2}BH^{-1}(s)\mathbf{u}(s) \\ &= \mu^2(s)\mathbf{z}^*(s)C^*(sI - A)^{-2}B\mathbf{u}(s).\end{aligned}$$

The Newton scheme then becomes

$$\begin{aligned} s_{k+1} &= s_k - \frac{f(s_k)}{f'(s_k)} \\ &= s_k - \frac{\mu_{\min}}{\mu_{\min}^2 \mathbf{z}^* C^* (s_k I - A)^{-2} B \mathbf{u}} \\ &= s_k - \frac{1}{\mu_{\min}} \frac{1}{\mathbf{z}^* C^* (s_k I - A)^{-2} B \mathbf{u}}, \end{aligned}$$

where $(\mu_{\min}, \mathbf{u}, \mathbf{z}) = (\mu_{\min}(s_k), \mathbf{u}_{\min}(s_k), \mathbf{z}^*_{\min}(s_k))$ is the eigentriplet of $H^{-1}(s_k)$ corresponding to $\lambda_{\min}(H^{-1}(s_k))$. An algorithm, very similar to the DPA algorithm [91], for the computation of a single dominant pole of a MIMO transfer function using the above Newton scheme, is shown in Alg. 4.1. Note that this algorithm is easier to recognize as a Newton scheme than the MDP presented in [93], which is conceptually the same. In the neighborhood of a solution, Alg. 4.1 converges quadratically.

Algorithm 4.1 MIMO Dominant Pole Algorithm (MDP)

INPUT: System (A, B, C), initial pole estimate s_1

OUTPUT: Approximate dominant pole λ and corresponding right and left eigenvectors $\mathbf{x}$ and $\mathbf{y}$.

1: Set $k = 1$
2: **while** not converged **do**
3: Compute eigentriplet $(\mu_{\min}, \mathbf{u}, \mathbf{z})$ of $H^{-1}(s_k)$
4: Solve $\mathbf{v} \in \mathbb{C}^n$ from

$$(s_k I - A)\mathbf{v} = B\mathbf{u}$$

5: Solve $\mathbf{w} \in \mathbb{C}^n$ from

$$(s_k I - A)^* \mathbf{w} = C\mathbf{z}$$

6: Compute the new pole estimate

$$s_{k+1} = s_k - \frac{1}{\mu_{\min}} \frac{1}{\mathbf{w}^* \mathbf{v}}$$

7: The pole $\lambda = s_{k+1}$ with $\mathbf{x} = \mathbf{v}/\|\mathbf{v}\|_2$ and $\mathbf{y} = \mathbf{w}/\|\mathbf{w}\|_2$ has converged if

$$||A\mathbf{x} - s_{k+1}\mathbf{x}||_2 < \epsilon$$

for some $\epsilon \ll 1$
8: Set $k = k + 1$
9: **end while**

4.3.2 SAMDP as an accelerated Newton scheme

The three strategies that are used for SADPA [123], are also used to make SAMDP, a generalization of Alg. 4.1 that is able to compute more than one dominant pole: subspace acceleration, selection of most dominant approximation and deflation. A global overview of the SAMDP is shown in Alg. 4.2. Each of the three strategies is explained in the following paragraphs.

Algorithm 4.2 Subspace Accelerated MDP Algorithm (SAMDP)

INPUT: System (A, B, C), initial pole estimate s_1 and the number of wanted poles p_{max}
OUTPUT: Dominant pole triplets $(\lambda_i, \mathbf{x}_i, \mathbf{y}_i)$, $i = 1, \ldots, p_{max}$

1: $k = 1$, $p_{found} = 0$, $V = W = \Lambda = X = Y = []$
2: **while** $p_{found} < p_{max}$ **do**
3: Compute eigentriplet $(\mu_{\min}, \mathbf{u}, \mathbf{z})$ of $H^{-1}(s_k)$
4: Solve $\mathbf{v} \in \mathbb{C}^n$ from
$$(s_k I - A)\mathbf{v} = B\mathbf{u}$$
5: Solve $\mathbf{w} \in \mathbb{C}^n$ from
$$(s_k I - A)^* \mathbf{w} = C\mathbf{z}$$
6: $V = \text{Expand}(V, X, Y, \mathbf{x})$ {Alg. 4.4}
7: $W = \text{Expand}(W, Y, X, \mathbf{y})$ {Alg. 4.4}
8: Compute $T = W^*V$ and $S = W^*AV$
9: $(\widehat{\Lambda}, \widehat{X}, \widehat{Y}) = \text{Sort}(S, T, V, W, B, C)$ {Alg. 4.3}
10: **if** $||A\hat{\mathbf{x}}_1 - \hat{\lambda}_1\hat{\mathbf{x}}_1||_2 < \epsilon$ **then**
11: $(\Lambda, X, Y, V, W) =$
$\text{Deflate}(\hat{\lambda}_1, \hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1, \Lambda, X, Y, \widehat{X}_{2:k}, \widehat{Y}_{2:k})$ {Alg. 4.5}
12: $p_{found} = p_{found} + 1$
13: Set $\hat{\lambda}_1 = \hat{\lambda}_2$, $k = k - 1$
14: **end if**
15: Set $k = k + 1$
16: Set the new pole estimate $s_{k+1} = \hat{\lambda}_1$
17: **end while**

Subspace acceleration

The approximations $\mathbf{v}$ and $\mathbf{w}$ that are computed in steps 4 and 5 of Alg. 4.1 are kept in orthogonal search spaces V and W, using modified Gram-Schmidt (MGS) [64]. These search spaces grow every iteration and will contain better approximations (see step 6 and 7 of Alg. 4.2).

Selection strategy

Every iteration a new pole estimate s_k must be chosen. There are several strategies (see [123] and Section 4.4.2). Here the most natural choice is to select the triplet $(\hat{\lambda}_j, \hat{\mathbf{x}}_j, \hat{\mathbf{y}}_j)$ with largest residue norm $||\widehat{R}_j||_2/|\mathrm{Re}(\hat{\lambda}_j)|$. SAMDP continues with $s_{k+1} = \hat{\lambda}_j$. See Alg. 4.3.

Algorithm 4.3 $(\widehat{\Lambda}, \widehat{X}, \widehat{Y}) = \mathrm{Sort}(T, G, V, W, B, C)$

INPUT: $S, T \in \mathbb{C}^{k\times k}$, $V, W \in \mathbb{C}^{n\times k}$, $B \in \mathbb{R}^{n\times p}$,$C \in \mathbb{R}^{n\times m}$
OUTPUT: $\widehat{\Lambda} \in \mathbb{C}^n$, $\widehat{X}, \widehat{Y} \in \mathbb{C}^{n\times k}$ with $\hat{\lambda}_1$ the pole with largest residue matrix norm and $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{y}}_1$ the corresponding approximate right and left eigenvectors.

1: Compute eigentriplets of (S, T):

$$(\tilde{\lambda}_i, \tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i), \quad i = 1, \ldots, k$$

2: Compute approximate eigentriplets of A as

$$(\hat{\lambda}_i = \tilde{\lambda}_i, \hat{\mathbf{x}}_i = V\tilde{\mathbf{x}}_i, \hat{\mathbf{y}}_i = W\tilde{\mathbf{y}}_i), \quad i = 1, \ldots, k$$

3: $\widehat{\Lambda} = [\hat{\lambda}_1, \ldots, \hat{\lambda}_k]$
4: $\widehat{X} = [\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_k]$
5: $\widehat{Y} = [\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_k]$
6: Compute residue matrices $R_i = (C^*\hat{\mathbf{x}}_i)(\hat{\mathbf{y}}_i^* B)$
7: Sort $\widehat{\Lambda}$, $\widehat{X}$, $\widehat{Y}$ in decreasing $||R_i||_2/|\mathrm{Re}(\lambda_i)|$ order

Deflation

An eigentriplet $(\hat{\lambda}_j, \hat{\mathbf{x}}_j, \hat{\mathbf{y}}_j)$ has converged if $||A\hat{\mathbf{x}}_j - \hat{\lambda}_j\hat{\mathbf{x}}_j||_2$ is smaller than some tolerance ϵ. If more than one eigentriplet is wanted, repeated computation of already converged eigentriplets must be avoided. This can be achieved by using deflation [132, 111].

If the exact right and left eigenvectors $\mathbf{x}_j$ and $\mathbf{y}_j$ are found, then it can be verified that the matrix

$$\widetilde{A} = \prod_j (I - \frac{\mathbf{x}_j\mathbf{y}_j^*}{\mathbf{y}_j^*\mathbf{x}_j}) \cdot A \cdot \prod_j (I - \frac{\mathbf{x}_j\mathbf{y}_j^*}{\mathbf{y}_j^*\mathbf{x}_j})$$

has the same eigentriplets as A, but with the found eigenvalues transformed to zero.

Using this, the space V needs to be orthogonally expanded with $\prod_j (I - \frac{\mathbf{x}_j\mathbf{y}_j^*}{\mathbf{y}_j^*\mathbf{x}_j}) \cdot \mathbf{v}$ and similarly, the space W needs to orthogonally expanded with $\prod_j (I - \frac{\mathbf{y}_j\mathbf{x}_j^*}{\mathbf{x}_j^*\mathbf{y}_j}) \cdot$

w. These projections are implemented using modified Gram-Schmidt (MGS) (see Alg. 4.4).

Algorithm 4.4 $V = \text{Expand}(V, X, Y, \mathbf{v})$

INPUT: $V \in \mathbb{C}^{n\times k}$ with $V^*V = I$, $X, Y \in \mathbb{C}^{n\times p}$, $\mathbf{v} \in \mathbb{C}^n$, Y^*X diagonal, $Y^*V = 0$

OUTPUT: $V \in \mathbb{C}^{n\times(k+1)}$ with $V^*V = I$ and $\mathbf{v}_{k+1} = \prod_{j=1}^{p}(I - \frac{\mathbf{x}_j\mathbf{y}_j^*}{\mathbf{y}_j^*\mathbf{x}_j}) \cdot \mathbf{v}$

1: $\mathbf{v} = \prod_{j=1}^{p}(I - \frac{\mathbf{x}_j\mathbf{y}_j^*}{\mathbf{y}_j^*\mathbf{x}_j}) \cdot \mathbf{v}$
2: $\mathbf{v} = \text{MGS}(V, \mathbf{v})$
3: $V = [V, \mathbf{v}/||\mathbf{v}||_2]$

If a complex pole has converged, its complex conjugate is also a pole and the corresponding complex conjugate right and left eigenvectors can also be deflated. A complex conjugate pair is counted as one pole. The complete deflation procedure is shown in algorithm 4.5.

Algorithm 4.5 $(\Lambda, X, Y, \widetilde{X}, \widetilde{Y}) = \text{Deflate}(\lambda, \mathbf{x}, \mathbf{y}, \Lambda, X, Y, V, W)$

INPUT: $\lambda \in \mathbb{C}$, $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$, $\Lambda \in \mathbb{C}^p$, $X, Y \in \mathbb{C}^{n\times p}$, $V, W \in \mathbb{C}^{n\times k}$

OUTPUT: $\Lambda \in \mathbb{C}^q$, $X, Y \in \mathbb{C}^{n\times q}$,$\widetilde{X}, \widetilde{Y} \in \mathbb{C}^{n\times k-1}$, where $q = p + 1$ if λ has zero imaginary part and $q = p + 2$ if λ has nonzero imaginary part.

1: $\Lambda = [\Lambda, \lambda]$
2: $X = [X, \mathbf{x}]$
3: $Y = [Y, \mathbf{y}]$
4: **if** $\text{imag}(\lambda) \neq 0$ **then**
5: {Also deflate complex conjugate}
6: $\Lambda = [\Lambda, \bar{\lambda}]$
7: $X = [X, \bar{\mathbf{x}}]$
8: $Y = [Y, \bar{\mathbf{y}}]$
9: **end if**
10: $\widetilde{X} = \widetilde{Y} = []$
11: **for** $j = 1, \ldots, k-1$ **do**
12: $\widetilde{X} = \text{Expand}(\widetilde{X}, X, Y, V_j)$
13: $\widetilde{Y} = \text{Expand}(\widetilde{Y}, Y, X, W_j)$
14: **end for**

4.4 Practical implementations of SAMDP

In this section, aspects concerning practical implementations of SAMDP and the generalization of SAMDP to non-square MIMO transfer functions ($m \neq p$) are discussed.

4.4.1 Sparse descriptor system models

The sparse descriptor system formulation of (4.2.1) becomes

$$\begin{cases} I_d\dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + Bu(t) \\ y(t) &= C^*\mathbf{x}(t) + Du(t), \end{cases}$$

where $A \in \mathbb{R}^{N\times N}$, $B \in \mathbb{R}^{N\times m}$, $C \in \mathbb{R}^{N\times p}$, $\mathbf{x}(t) \in \mathbb{R}^N$, $u(t) \in \mathbb{R}^m$, $y(t) \in \mathbb{R}^p$, $D \in \mathbb{R}^{p\times m}$ and $I_d \in \mathbb{R}^{N\times N}$ is a diagonal matrix with diagonal elements either 0 or 1. The corresponding transfer function $H_d(s) : \mathbb{C} \longrightarrow \mathbb{C}^{p\times m}$ is defined as

$$H_d(s) = C^*(sI_d - A)^{-1}B + D,$$

where $s \in \mathbb{C}$. Without loss of generality, $D = 0$ in the following.

The algorithms presented in this chapter can easily be adapted to handle sparse descriptor systems. The changes essentially boil down to replacing I by I_d on most places and noting that for eigentriplets $(\lambda_j, \mathbf{x}_j, \mathbf{y}_j)$ the relation $\mathbf{y}_i^* I_d \mathbf{x}_j = 0, i \neq j$ holds. For completeness, the changes are given for each algorithm:

- Algorithm 4.1:
  - Replace I by I_d in step 4 and 5.
  - Step 6 becomes
$$s_{k+1} = s_k - \frac{1}{\mu_{\min}} \frac{1}{\mathbf{w}^* I_d \mathbf{v}}.$$
  - The criterion in step 7 becomes
$$||A\mathbf{x} - s_{k+1} I_d \mathbf{x}||_2 < \epsilon.$$
- Algorithm 4.2:
  - Replace I by I_d in step 4 and 5.
  - Replace step 6 and 7 by
$$\begin{aligned} V &= \text{Expand}(V, X, I_d \cdot Y, \mathbf{v}), \\ W &= \text{Expand}(W, Y, I_d \cdot X, \mathbf{w}). \end{aligned}$$
  - In step 8, use $T = W^* I_d V$.
  - The criterion in step 10 becomes
$$||A\hat{\mathbf{x}}_1 - \hat{\lambda}_1 I_d \hat{\mathbf{x}}_1||_2 < \epsilon.$$
- Algorithm 4.5:
  - Replace step 12 and 13 by
$$\begin{aligned} \widetilde{X} &= \text{Expand}(\widetilde{X}, X, I_d \cdot Y, V_j), \\ \widetilde{Y} &= \text{Expand}(\widetilde{Y}, Y, I_d \cdot X, W_j). \end{aligned}$$

All the experiments described in this chapter were done using implementations that operate on the sparse descriptor system model. The algorithm can readily be extended to handle general descriptor system transfer functions $H(s) = C^*(sE - A)^{-1}B + D$, with $E \in \mathbb{R}^{n\times n}$, as well.

4.4.2 Computational optimizations

If a large number of dominant poles is wanted, the search spaces V and W may become very large. By imposing a certain maximum dimension $k_{\max}$ for the search spaces, this can be controlled: when the dimension of V and W reaches $k_{\max}$, they are reduced to dimension $k_{\min} < k_{\max}$ by keeping the $k_{\min}$ most dominant approximate eigentriplets. The process is restarted with the reduced V and W, a concept known as implicit restarting [132, 123]. This procedure is continued until all poles are found.

The systems in step 4 and 5 of Alg. 4.2 can be solved with the same LU-factorization of $(s_k I_d - A)$, by using L and U in step 4 and U^* and L^* in step 5. Because in practice the sparse Jacobian is used, computation of the LU-factorization is inexpensive.

In step 3 of Alg. 4.2, the eigentriplet $(\mu_{\min}, \mathbf{u}, \mathbf{z})$ of $H^{-1}(s)$ must be computed. This triplet can be computed with inverse iteration [132], or, by noting that this eigentriplet corresponds to the eigentriplet $(\theta_{\max}, \mathbf{u}, \mathbf{z})$ of $H(s)$, with $\mu_{\min} = \theta_{\max}^{-1}$, with the power method [132] applied to $H(s)$. Note that there is no need to compute $H(s)$ explicitly. However, if the number of states of the system is large, and the number of inputs/outputs of matrix $H(s)$ is large as well, applying the power or inverse iteration methods at every iteration may be expensive. It may then be more efficient to only compute a new eigentriplet $(\mu_{\min}, \mathbf{u}, \mathbf{z})$ after a dominant pole has been found, or once every restart.

As more eigentriplets have converged, approximations of new eigentriplets may become poorer due to rounding errors in the orthogonalization phase and the already converged eigentriplets. It is therefore advised to take a small tolerance $\epsilon = 10^{-10}$. Besides that, if the residual for the current approximation drops below a certain tolerance $\epsilon_r > \epsilon$, one or more iterations may be saved by using generalized Rayleigh quotient iteration [110] to let the residual drop below ϵ. In practice, a tolerance $\epsilon_r = 10^{-5}$ is safe enough to avoid convergence to less dominant poles. Closely-spaced poles and repeated poles are not a problem for SAMDP, although convergence to defective poles may be only linear (see also [110]).

The SAMDP requires a single initial shift, even if more than one dominant pole is wanted, because the selection strategy automatically provides a new shift once a pole has converged. On the other hand, if one has special knowledge of the transfer function, for instance the approximate location of dominant poles, this information can be used by providing additional shifts to SAMDP. These shifts can then be used to accelerate the process of finding dominant poles. Although the location of the initial shift may influence the order in which the dominant poles are found, the new shifts provided by the selection strategy are helpful in computing poles located away from the initial shift and hence the quality of the modal equivalent is not much influenced by the initial shift.

As is also mentioned in [123], one can easily change the selection strategy to use any of the existing indices of modal dominance [68, 4]. The procedure can be automated even further by providing the desired maximum error $||H(s) - H_k(s)||$ for a suitable norm and frequency range: the procedure continues computing new

poles until the error bound is reached. Note that such an error bound requires that the transfer function of the complete model is known for a range of $s \in \mathbb{C}$ (which is usually the case for sparse descriptor systems).

4.4.3 General MIMO transfer functions ($m \neq p$)

For a general non-square transfer function $H(s) = C^*(sI-A)^{-1}B \in \mathbb{C}^{p\times m}$ ($p \neq m$), the objective function (4.3.1) cannot be used, because the eigendecomposition is only defined for square matrices. However, the singular value decomposition is defined for non-square matrices and hence the objective function becomes

$$f : \mathbb{C} \longrightarrow \mathbb{R} : s \longmapsto \frac{1}{\sigma_{\max}(H(s))}. \tag{4.4.1}$$

Let $(\sigma_{\max}(s), \mathbf{u}(s), \mathbf{z}(s))$ be a singular triplet of $H(s)$, i.e. $H(s)\mathbf{z}(s) = \sigma_{\max}(s)\mathbf{u}(s)$ and $H^*(s)\mathbf{u}(s) = \sigma_{\max}(s)\mathbf{z}(s)$. It follows that $H^*(s)H(s)\mathbf{z}(s) = \sigma^2_{\max}\mathbf{z}(s)$, so the objective function (4.4.1) can also be written as

$$f : \mathbb{C} \longrightarrow \mathbb{R} : s \longmapsto \frac{1}{\lambda_{\max}(H^*(s)H(s))}, \tag{4.4.2}$$

with $\lambda_{\max} = \sigma^2_{\max}$. Because $f(s)$ in (4.4.2) is a function from $\mathbb{C} \longrightarrow \mathbb{R}$, the derivative $df(s)/ds : \mathbb{C} \longrightarrow \mathbb{R}$ is not injective. A complex scalar $z = a + ib \in \mathbb{C}$ can be represented by $[a, b]^T \in \mathbb{R}^2$. The partial derivatives of the objective function (4.4.2) become

$$\begin{aligned}
\frac{\partial f}{\partial a}(s) &= \frac{1}{\lambda^2_{\max}(s)}\sigma_{\max}(s)(\mathbf{w}^* I_d \mathbf{v} + \mathbf{v}^* I_d \mathbf{w}),\\
\frac{\partial f}{\partial b}(s) &= \frac{1}{\lambda^2_{\max}(s)} i\sigma_{\max}(s)(\mathbf{w}^* I_d \mathbf{v} - \mathbf{v}^* I_d \mathbf{w}),
\end{aligned}$$

where

$$\mathbf{v} = (sI - A)^{-1}B\mathbf{z}, \quad \mathbf{w} = (sI - A)^{-*}C\mathbf{u}.$$

The derivative of (4.4.2) then becomes

$$\nabla f = 2\frac{\sigma_{\max}}{\lambda^2_{\max}(s)}[\mathrm{Re}(\mathbf{w}^* I_d \mathbf{v}), -\mathrm{Im}(\mathbf{w}^* I_d \mathbf{v})],$$

where $\mathrm{Re}(a + ib) = a$ and $\mathrm{Im}(a + ib) = b$. The Newton scheme is

$$\begin{aligned}
\begin{bmatrix}\mathrm{Re}(s_{k+1})\\ \mathrm{Im}(s_{k+1})\end{bmatrix} &= \begin{bmatrix}\mathrm{Re}(s_k)\\ \mathrm{Im}(s_k)\end{bmatrix} - (\nabla f(s_k))^\dagger f(s_k)\\
&= \begin{bmatrix}\mathrm{Re}(s_k)\\ \mathrm{Im}(s_k)\end{bmatrix} - [\mathrm{Re}(\mathbf{w}^* I_d \mathbf{v}), -\mathrm{Im}(\mathbf{w}^* I_d \mathbf{v})]^\dagger \frac{\sigma_{\max}}{2},
\end{aligned}$$

where $A^\dagger = A^*(AA^*)^{-1}$ denotes the pseudo-inverse of a matrix $A \in \mathbb{C}^{n\times m}$ with $\mathrm{rank}(A) = n$ ($n \leq m$) [64].

This Newton scheme can be proved to have superlinear convergence locally. Because the SAMDP uses subspace acceleration, which accelerates the search for new directions, and relies on Rayleigh quotient iteration for nearly converged eigentriplets, it is expected that performance for square and non-square systems will be equally good, as is also confirmed by experiments.

4.5 Numerical results

The algorithm was tested on a number of systems, for a number of different input and output matrices B and C. Here the results for the Brazilian Interconnected Power System (BIPS) are shown. The BIPS data corresponds to a year 1999 planning model, having 2,370 buses, 3,401 lines, 123 synchronous machines plus field excitation and speed-governor controls, 46 power system stabilizers, 4 static var compensators, two TCSCs equipped with oscillation damping controllers, and one large HVDC link. Each generator and associated controls is the aggregate model of a whole power plant. The BIPS model is linearized about an operating point having a total load of 46,000 MW, with the North-Northeast generators exporting 1,000 MW to the South-Southeast Region, through the planned 500 kV, series compensated North-South interti.e.

The state-space realization of the BIPS model has 1,664 states and the sparse, unreduced Jacobian has dimension 13,251. The sparse Jacobian structure and the full eigenvalue spectrum, for this 1,664-state BIPS model, are pictured in [93]. Like the experiments in [93, 123], the practical implementation operates on the sparse unreduced Jacobian of the system, instead of on the dense state matrix A.

In the experiments, the convergence tolerance used was $\epsilon = 10^{-10}$. The spaces V and W were limited to size 10 ($k_{min} = 2$, $k_{max} = 10$). New orientation vectors $\mathbf{u}$ and $\mathbf{z}$ corresponding to $\sigma_{\max}$ (see step 3 in Alg. 4.2) were only computed after a pole had converged. Every iteration, the approximation with largest residue norm was selected. All experiments were carried out in Matlab 6.5 [151] on an Intel Centrino Pentium 1.5 GHz with 512 MB RAM.

To demonstrate the performance of SAMDP, it was applied to two square transfer functions and two non-square transfer functions of BIPS to compute a number of dominant poles (complex conjugate pairs are counted as one pole). Table 4.1 shows the statistics of SAMDP for the transfer functions. The eigenvalue spectrum of the 8×8 MIMO modal equivalent, whose sigma plots are given in Figure 4.3, is pictured in Fig. 4.1. The eigenvalue spectrum of the 28×28 MIMO modal equivalent, whose sigma plot is given in Fig. 4.6, is pictured in Fig. 4.2.

SAMDP is able to automatically compute a modal equivalent for the 8×8 transfer function of acceptable size that captures both the $\sigma_{\min}$ and $\sigma_{\max}$ curves rather well, as shown in the sigma plots in Figure s 4.3 and 4.4. Increasing the number of states also reduces the error $||H(i\omega) - H_k(i\omega)||_2$. The 8×8 transfer function is taken from [93], where a fairly low performance modal equivalent, having 39 states, was obtained through repeated MDP runs, in a procedure that required considerable human interaction. The SAMDP automatically computes all the poles

Table 4.1: Results of SAMDP for the 8×8, 28×28, 8×6, and 28×25 transfer functions of the Brazilian Interconnected Power System (BIPS). Shift $s_1 = 0.1i$.

Transfer function	#poles	#states	#LU	Time (s)	Fig.
8×8	120	184	913	750	4.3
8×8	200	294	1380	1500	4.4
28×28	150	248	2823	2400	4.5
28×28	180	291	3010	3030	4.6
8×6	160	240	1285	1200	4.7
28×25	180	287	2991	2980	4.8

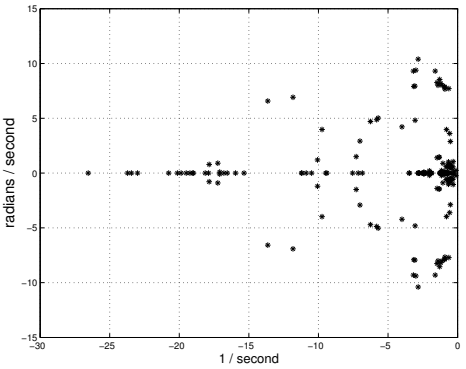


Figure 4.1: Pole spectrum of 184th order modal equivalent of the 8×8 transfer function.

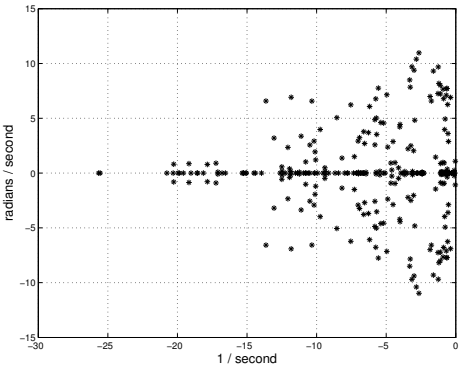


Figure 4.2: Pole spectrum of 291st order modal equivalent of the 28×28 transfer function.

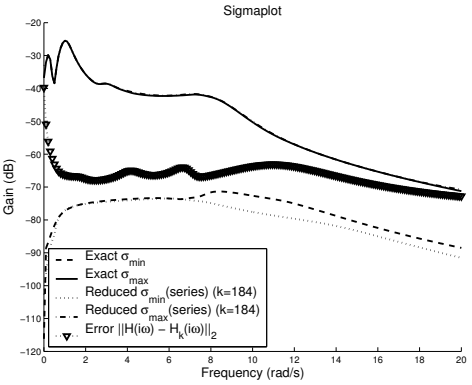


Figure 4.3: Sigma plot of modal equivalent and complete model, and error for the 8×8 Brazilian system transfer function (1,664 states in the complete model, 184 in the reduced model).

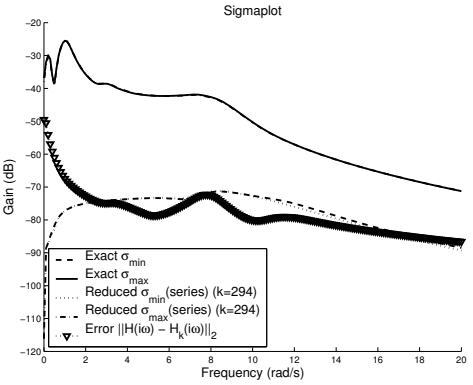


Figure 4.4: Sigma plot of modal equivalent and complete model, and error for the 8×8 Brazilian system transfer function (1,664 states in the complete model, 294 in the reduced model).

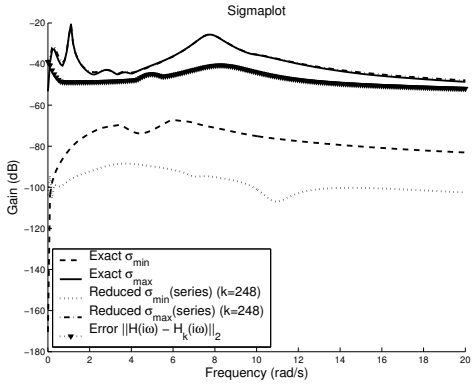


Figure 4.5: Sigma plot of modal equivalent and complete model, and error for the 28×28 Brazilian system transfer function (1,664 states in the complete model, 248 in the reduced model).

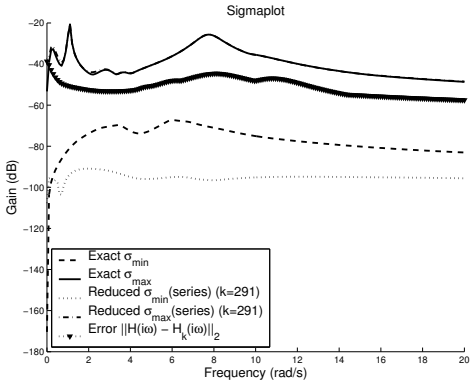


Figure 4.6: Sigma plot of modal equivalent and complete model, and error for the 28×28 Brazilian system transfer function (1,664 states in the complete model, 291 in the reduced model).

in one run. The reader is referred to [93] for more practical details on this 8×8 power system transfer function and a complete list of the 39 dominant poles.

The second example is a 28×28 transfer function of the BIPS model. Here one is in particular interested in a good fitting of the $\sigma_{\max}$ curve over the 10^{-2} Hz to 2 Hz range, as this indicates all major electromechanical modes have been captured, revealing its potential value for applications in the damping analysis and control of power system oscillations. Matrix $B \in \mathbb{R}^{n\times 28}$ is comprised of mechanical power input disturbance vectors for 28 generators, while $C \in \mathbb{R}^{n\times 28}$ is comprised of output row vectors for the rotor speed deviations of the same generators. These 28 generators were selected for being of large size and also located in strategic parts of the BIPS, so that the MIMO function has good observability/controllability of the major system electromechanical modes. From Figure s 4.5 and 4.6 it can be observed that the SAMDP is able to approximate the $\sigma_{\max}$-curve well, while it has more difficulties in approximating the $\sigma_{\min}$ curve: many more poles would be needed for a good fitting of the $\sigma_{\min}$-curve.

Figures 4.7 and 4.8 show the sigma plots for the non-square 8×6 and 28×25 transfer functions, which were obtained by truncating the last columns of B of the 8×8 and 28×28 transfer functions respectively. The results confirm that SAMDP is also applicable to non-square transfer functions, with comparable performance.

It must be noted that all modal equivalents in this section are automatically computed by SAMDP, without human interaction. The relatively small 2-norm of the error function $H(s) - H_k(s)$ indicates the zeros are also preserved, although not explicitly computed by the algorithm. The $\sigma_{\min}$-plots are related to the location of the transmission zeros, as experimentally verified by the authors. A good fitting of the $\sigma_{\min}$-plot over a given frequency range indicates that the zeros located within this range are preserved in the modal equivalent. Numerically, however, the $\sigma_{\min}$-

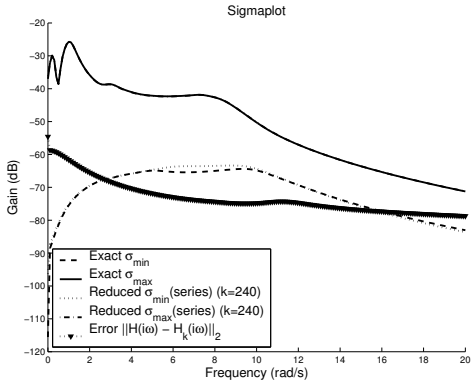


Figure 4.7: Sigma plot of modal equivalent and complete model, and error for the 8×6 Brazilian system transfer function (1,664 states in the complete model, 240 in the reduced model).

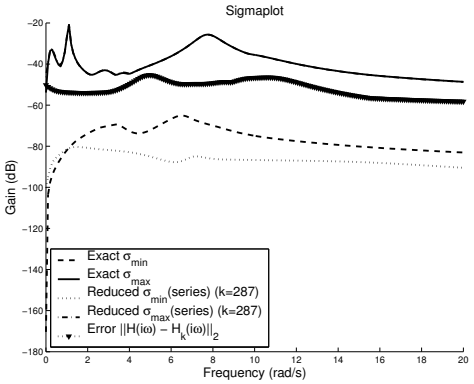


Figure 4.8: Sigma plot of modal equivalent and complete model, and error for the 28×25 Brazilian system transfer function (1,664 states in the complete model, 287 in the reduced model).

curves are difficult to approximate, due the low magnitude of the $\sigma_{\min}$ values.

Transfer function zeros give a measure of the controllability of the system and have also other applications, but are a less known subject for multivariable systems [44, 92, 31]. Dominant zeros for transfer function matrices may be computed by a Newton scheme very similar to Alg. 4.1. Generalizations to non-square transfer functions are under current investigation[1].

Further reduced system models, if needed for advanced control applications, may be obtained by applying the Balanced Model Reduction algorithm [68] to a state-space realization of the modal equivalent, after having used the SAMDP algorithm to produce this modal equivalent for a large scale system.

4.6 Conclusions

The SAMDP algorithm is a fast and robust method to compute dominant poles and corresponding residue matrices of both square and non-square MIMO transfer functions. The algorithm is a variant of the SADPA [123] and has several advantages compared to existing methods: a natural selection method is used to converge to both real and complex dominant poles, subspace acceleration accelerates the algorithm and provides new pole estimates, and deflation techniques prevent the algorithm from (re-)computing already found poles. Finally, SAMDP is completely automatic: with a single shift, it is able to compute as many dominant poles as wanted, without intermediate human interaction.

The chapter results are related to the analysis and control of small signal stability, but the SAMDP algorithm is general and could be effectively applied to problems in other engineering fields that allow sparse descriptor system formula-

[1] See also Chapter 5.

tions. It can easily be adjusted to take advantage of specific properties or knowledge of the system.

Addendum

The more efficient deflation as described for the SISO case (SADPA) in Section 3.3.1 and Section 3.3.2 of Chapter 3, can also be applied here:

Theorem 4.6.1. *Consider the transfer function $H(s) = C^*(sE - A)^{-1}B$. Let X and Y have as their columns the normalized found right and left eigenvectors $\mathbf{x}_i$ and $\mathbf{y}_i$ $(i = 1, \ldots, k)$ of (A, E), respectively, and let Λ be a diagonal matrix with on its diagonal the corresponding eigenvalues, i.e. $\Lambda = diag(\lambda_1, \ldots, \lambda_k)$, $Y^*AX = \Lambda$ and $Y^*EX = I$. The deflated transfer function $H_d(s) = C_d^*(sE - A)^{-1}B_d$, where*

$$B_d = (I - EXY^*)B \quad \text{and} \quad C = (I - E^*YX^*)C,$$

has the same poles λ_i and corresponding residues R_i as $H(s) = C^*(sE-A)^{-1}B$, but with the residues R_i corresponding to the found poles λ_i $(i = 1, \ldots, k)$ transformed to $R_i = 0$.

Proof. This is a straightforward generalization of theorem 3.3.1. □

Secondly, the computation of the approximate residues in Alg. 4.3 can also be simplified (cf. Section 3.3.2): if in step 1 of Alg. 4.3 the $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{y}}_i$ are scaled so that $\tilde{\mathbf{y}}_i^* T \tilde{\mathbf{x}}_i = 1$ $(= \hat{\mathbf{y}}_i^* E \hat{\mathbf{x}}_i)$, then it follows that the $\widehat{R}_i$ can be computed as $\widehat{R}_i = ((C^*V)\tilde{\mathbf{x}}_i)(\tilde{\mathbf{y}}_i^*(W^*B))$ $(= (C^*\hat{\mathbf{x}}_i)(\hat{\mathbf{y}}_i^*B))$. The algorithms in this chapter can readily be adapted to these improvements (see also the SADPA algorithm in Section 3.3.2).

It is, however, not always clear whether the scaling $\hat{\mathbf{y}}_i^* E \hat{\mathbf{x}}_i = 1$ is optimal. If $\hat{\mathbf{y}}_i$ and $\hat{\mathbf{x}}_i$ are good approximations of dominant left and right eigenvectors, then this scaling most likely causes $\hat{\lambda}_i$ to be selected as next shift, leading to fast convergence to the corresponding dominant pole. If, on the other hand, non of the approximate eigentriplets are accurate, this scaling may significantly influence the selection process, possibly more than one desires. In the absence of clearly dominant approximations, this may lead to stagnation, because every iteration a (very) different approximate pole is selected. This stagnation may occur if most of the dominant poles are already found, as was seen in Section 3.8 of Chapter 3, but also if the search spaces do not have (strong) components in the direction of the dominant eigenvectors. A remedy is to consider the approximate residues $\widehat{R}_i = (C^*\hat{\mathbf{x}}_i)(\hat{\mathbf{y}}_i^*B)$ with normalized $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{y}}_i$, i.e. $\|\hat{\mathbf{x}}_i\|_2 = \|\hat{\mathbf{y}}_i\|_2 = 1$. Note that if V and W are orthogonal, then still $\widehat{R}_i = ((C^*V)\tilde{\mathbf{x}}_i)(\tilde{\mathbf{y}}_i^*(W^*B))$ with $\|\tilde{\mathbf{x}}_i\|_2 = \|\tilde{\mathbf{y}}_i\|_2 = 1$. With this scaling, emphasis is more on components in the direction of C and B. Note that the approximate residues can still be scaled by $\mathrm{Re}(\hat{\lambda}_i)$, if this measure of dominance is desired.

A detailed view on some selected iterations of the SAMDP process makes the effect clear. Two SAMDP processes were started for the 8×8 transfer function

Table 4.2: Iterations of SAMDP for the 8×8 transfer function of BIPS, with $\hat{\mathbf{y}}_i^* E \hat{\mathbf{x}}_i = 1$ (process I) and $\|\hat{\mathbf{y}}_i\|_2 = \|\hat{\mathbf{x}}_i\|_2 = 1$ scaling (process II). Convergence is denoted by a plus (+) and an asterisk (*), respectively.

	Process I			Process II	
Iteration	Selected pole	rank in II	$\|\mathbf{r}_i\|_2$	Selected pole	$\|\mathbf{r}_i\|_2$
1	$-0.21 + 0.23i$	1	$O(10^{-2})$	$-0.21 + 0.23i$	$O(10^{-2})$
2	$-0.13 + 0.25i$	1	$O(10^{-3})$	$-0.13 + 0.25i$	$O(10^{-3})$
3	$-0.11 + 0.24i$	1	$O(10^{-4})$	$-0.11 + 0.24i$	$O(10^{-4})$
4+*	$-0.11 + 0.24i$	1	$O(10^{-9})$	$-0.11 + 0.24i$	$O(10^{-9})$
5	$-0.25 + 0.42i$	1	$O(10^{-3})$	$-0.25 + 0.42i$	$O(10^{-3})$
6	$-0.39 + 0.51i$	1	$O(10^{-4})$	$-0.39 + 0.51i$	$O(10^{-4})$
7+*	$-0.35 + 0.56i$	1	$O(10^{-9})$	$-0.35 + 0.56i$	$O(10^{-9})$
8	$-0.52 + 0.43i$	1	$O(10^{-3})$	$-0.52 + 0.43i$	$O(10^{-3})$
9	$-0.65 + 0.94i$	1	$O(10^{-3})$	$-0.65 + 0.94i$	$O(10^{-3})$
10	$-0.37 - 0.32i$	2	$O(10^{-3})$	$-0.37 + 1.00i$	$O(10^{-3})$
11	$-0.64 - 0.64i$	2	$O(10^{-3})$	$-0.31 + 1.04i$	$O(10^{-3})$
12*	$-0.78 - 0.49i$	3	$O(10^{-3})$	$-0.31 + 1.04i$	$O(10^{-9})$

of Section 4.5: process I with the usual scaling ($\hat{\mathbf{y}}_i^* E \hat{\mathbf{x}}_i = 1$), and process II with normalized scaling ($\|\hat{\mathbf{x}}_i\|_2 = \|\hat{\mathbf{y}}_i\|_2 = 1$); the other settings were as in Section 4.5. Table 4.2 shows the selected pole approximations, the ranking of the pole selected by I in the measure of II, and residual norms for both processes, denoting convergence with a plus and an asterisk, respectively. Up to the second found pole the processes are identical: every iteration the same approximation is selected. After the second found pole, however, process I loses track and starts to select approximations more or less randomly (influenced by the 'dominant' scaling $\hat{\mathbf{y}}_i^* E \hat{\mathbf{x}}_i = 1$), while process II smoothly converges to a third dominant pole. Apparently, the normalization also helps the process to keep track. This behavior was observed during the complete process, and in other experiments as well.

The remedy works well for the examples in this chapter: for the 8×8 transfer function, SAMDP, with the deflation as in Thm. 4.6.1 and the described scaling, needed 670 iterations to compute 120 poles (vs. 913 iterations, with $\hat{\mathbf{y}}_i^* \hat{\mathbf{x}}_i$ instead of $\hat{\mathbf{y}}_i^* E \hat{\mathbf{x}}_i$ scaling by accident, cf. Table 4.1), and for the 28×28 transfer function, SAMDP needed 1231 iterations to compute 180 poles (vs. 3010 iterations). For SISO transfer functions the effect was similar. In numerical experiments with various other systems it was confirmed that this remedy is most effective when the transfer function is relatively smooth (or after deflation of most of the dominant poles), but is less crucial for frequency responses with many peaks, such as the PEEC system in Section 3.8. Although the order in which the dominant poles were computed differed in some cases, missing of dominant poles was not observed.

Except for some changes in notation, this chapter has been published (with appendix) as [122]

Joost Rommes and Nelson Martins, *Efficient computation of multivari-*

able transfer function dominant poles using subspace acceleration, IEEE Transactions on Power Systems **21** (2006), no. 4, 1471–1483.

Chapter 5

Efficient computation of large-scale transfer function dominant zeros

Abstract. This chapter describes efficient algorithms for the computation of dominant zeros of large-scale transfer functions. The transfer function zeros are demonstrated to be equal to the poles of a new inverse system, which is valid even for the strictly proper case. This is a new finding, which is important from practical as well as theoretical viewpoints. Hence, the dominant zeros can be computed as the dominant poles of the inverse transfer function by the recent efficient SADPA and SAMDP algorithms. The importance of computing dominant zeros and the performance of the algorithms are illustrated by examples from practical power system models.

Key words. transfer function zeros, small-signal stability, poorly-damped oscillations, power system dynamics, transfer function, transfer function residues, pole-zero map, model reduction, large-scale systems, sparse eigenanalysis, modal equivalents, modal analysis, multivariable systems, inverse systems.

5.1 Introduction

The zeros of a transfer function are important in several applications such as the analysis and design of multivariable feedback control systems. The coordinated design of power system stabilizers, for damping electromechanical oscillations in large interconnected power systems, involves numerous decentralized control loops and greatly benefits from the knowledge of scalar as well as multivariable zeros.

There are several methods for the computation of all zeros of a transfer function, see [135, 100, 164] and references therein. However, all these methods are only applicable to moderately sized systems, while in practice one is often interested only in the most dominant zeros of large-scale transfer functions.

The SADPA (SAMDP) algorithms for the computation of dominant poles of (multivariable) transfer functions that have been developed in [123, 122] (see Chapters 3 and 4, respectively), can be used, when adapted, for several other applications. This chapter is concerned with variants that can be used to compute the dominant zeros of transfer functions of large-scale dynamical systems. In fact, it will be shown that the SADPA (SAMDP) algorithms can be used without adaptation, by using the relationship between the zeros of the transfer function and the poles of the corresponding inverse transfer function.

The outline of the chapter is as follows. Definitions of poles and zeros of general transfer functions are given in Section 5.2, as well as the motivation for finding dominant zeros of scalar and multivariable transfer functions. In Section 5.3, algorithms will be described to compute a dominant zero of a large-scale SISO or MIMO transfer function. It will be shown that the zeros of a transfer function are equal to the poles of the inverse transfer function. Hence, the SADPA [123] and SAMDP [122] algorithms can be used to compute the dominant zeros via the dominant poles of the inverse transfer function, as will be described in Section 5.4. All algorithms are illustrated by large-scale examples and applications from practical power system models in Section 5.5. Section 5.6 concludes.

5.2 Transfer functions, poles, and zeros

Throughout this chapter, the dynamical systems (E, A, B, C, D) are of the form

$$\left\{ \begin{array}{rcl} E\dot{\mathbf{x}}(t) & = & A\mathbf{x}(t) + B\mathbf{u}(t) \\ \mathbf{y}(t) & = & C^*\mathbf{x}(t) + D\mathbf{u}(t), \end{array} \right. \tag{5.2.1}$$

where $A, E \in \mathbb{R}^{n\times n}$, E singular, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{n\times p}$, $\mathbf{x}(t) \in \mathbb{R}^n$, $\mathbf{u}(t) \in \mathbb{R}^m$, $\mathbf{y}(t) \in \mathbb{R}^p$ and $D \in \mathbb{R}^{p\times m}$. The transfer function $H(s) : \mathbb{C} \longrightarrow \mathbb{C}^{p\times m}$ is defined as

$$H(s) = C^*(sE - A)^{-1}B + D, \tag{5.2.2}$$

with $s \in \mathbb{C}$. If $m = p = 1$, the system (5.2.1) is called a single-input single-output (SISO) system, and $\mathbf{b} = B, \mathbf{c} = C \in \mathbb{R}^n$ are vectors and $d = D \in \mathbb{R}$ is a scalar. Otherwise it is a multi-input multi-output (MIMO) system.

The eigenvalues $\lambda_i \in \mathbb{C}$ of the matrix pencil (A, E) are the poles of transfer function (5.2.2). An eigentriplet $(\lambda_i, \mathbf{x}_i, \mathbf{y}_i)$ is composed of an eigenvalue λ_i of (A, E) and the corresponding right and left eigenvectors $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{C}^n$:

$$\begin{array}{rcll} A\mathbf{x}_i & = & \lambda_i E\mathbf{x}_i, & \mathbf{x}_i \neq 0, \\ \mathbf{y}_i^* A & = & \lambda_i \mathbf{y}_i^* E, & \mathbf{y}_i \neq 0. \end{array}$$

Assuming that all eigenvalues are distinct, the right and left eigenvectors corresponding to finite eigenvalues can be scaled so that $\mathbf{y}_i^* E\mathbf{x}_i = 1$. Furthermore, it can be shown that left and right eigenvectors corresponding to distinct eigenvalues

are E-orthogonal: $\mathbf{y}_i^* E \mathbf{x}_j = 0$ for $i \neq j$. The transfer function $H(s)$ can be expressed as a sum of residue matrices $R_i \in \mathbb{C}^{p \times m}$ over the $\tilde{n} \leq n$ finite first-order poles [78]:

$$H(s) = \sum_{i=1}^{\tilde{n}} \frac{R_i}{s - \lambda_i} + R_\infty + D,$$

where the residues R_i are

$$R_i = (C^* \mathbf{x}_i)(\mathbf{y}_i^* B),$$

and R_∞ is the constant contribution of the poles at infinity (often zero).

5.2.1 Dominant poles

Although there are different indices of modal dominance [4, 68, 159], the following will be used in this chapter.

Definition 5.2.1. *A pole λ_i of $H(s)$ with corresponding right and left eigenvectors $\mathbf{x}_i$ and $\mathbf{y}_i$ ($\mathbf{y}_i^* E \mathbf{x}_i = 1$) is called* the *dominant pole if $\widehat{R}_i = ||R_i||_2/|Re(\lambda_i)| = \sigma_{\max}(R_i)/|Re(\lambda_i)| > \widehat{R}_j$ for all $j \neq i$.*

A pole λ_j that corresponds to a residue R_j with large $\sigma_{\max}(R_j)/|\mathrm{Re}(\lambda_j)|$ is called a dominant pole, i.e. a pole that is well observable and controllable in the transfer function. This can also be observed from the corresponding $\sigma_{\max}$-plot [89, 122, 17] of $H(s)$, where peaks occur at frequencies close to the imaginary parts of the dominant poles of $H(s)$ (in the SISO case this plot is called the Bode plot, see also Fig. 5.1). An approximation of $H(s)$ that consists of $k \ll n$ terms with $||R_j||_2/|\mathrm{Re}(\lambda_j)|$ above some value, determines the effective transfer function behavior [144] and will be referred to as transfer function modal equivalent:

$$H_k(s) = \sum_{j=1}^{k} \frac{R_j}{s - \lambda_j}.$$

Because a residue matrix R_j is the product of a column vector and a row vector, it is of unit rank. Therefore at least $\min(m, p)$ different poles (and corresponding residue matrices) are needed to obtain a modal equivalent with nonzero $\sigma_{\min}(\omega)$ plot [93, 114].

5.2.2 Dominant zeros

There are many different approaches to the definition of multivariable system zeros, see for instance [135] for an extensive overview. In the SISO case, $z_0 \in \mathbb{C}$ is called a transmission zero if $H(z_0) = \mathbf{c}^*(z_0 E - A)^{-1}\mathbf{b} + d = 0$. More generally, there is a distinction between transmission zeros and invariant zeros. This chapter will focus on the computation of transmission zeros.

The Rosenbrock system matrix [127] corresponding to (5.2.1) is

$$\Sigma = \begin{bmatrix} sE - A & B \\ -C^* & D \end{bmatrix}, \tag{5.2.3}$$

and is of importance, amongst others, because it can be transformed by basic operations to the (block Schur) form

$$\begin{bmatrix} I & 0 \\ 0 & H(s) \end{bmatrix}, \tag{5.2.4}$$

from which it can be seen that $\text{rank}(\Sigma) = n + \text{rank}(H(s))$. The definition of a transmission zero is given in Def. 5.2.2.

Definition 5.2.2. *A number $z_0 \in \mathbb{C}$ is called a transmission zero if it satisfies*

$$\textit{rank}(H(z_0)) < \max_s \textit{rank}(H(s)).$$

Note that with this definition it is in the MIMO case not sufficient (and not necessary) to have one entry of $H(z_0)$ equal to zero, and not necessary to have $H(z_0) = 0$. The transmission zeros are a subset of the invariant zeros of a system, that are defined as follows.

Definition 5.2.3. *A number $z_0 \in \mathbb{C}$ is called an invariant zero if it satisfies*

$$\textit{rank} \begin{bmatrix} z_0 E - A & B \\ -C^* & D \end{bmatrix} < \textit{rank}(\Sigma).$$

If the system (5.2.1) is minimal, the transmission zeros and invariant zeros coincide, and are simply called zeros.

If Σ, and hence $H(s)$ by (5.2.4), have full column rank ($m \leq p$), the vectors $\mathbf{x}$ and $\mathbf{u}$ corresponding to z_0 that satisfy

$$\begin{bmatrix} z_0 E - A & -B \\ C^* & D \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} = 0, \quad \mathbf{x} \in \mathbb{C}^n, \mathbf{u} \in \mathbb{C}^m,$$

are called the *right zero state* and *state zero input*, respectively [135]. Dually, if Σ has full row rank ($m \geq p$), the vectors $\mathbf{y}$ and $\mathbf{v}$ corresponding to z_0 that satisfy

$$\begin{bmatrix} \mathbf{y}^* & \mathbf{v}^* \end{bmatrix} \begin{bmatrix} z_0 E - A & -B \\ C^* & D \end{bmatrix} = 0, \quad \mathbf{y} \in \mathbb{C}^n, \mathbf{v} \in \mathbb{C}^p,$$

are called the *left zero state* and *state zero input*, respectively [135].

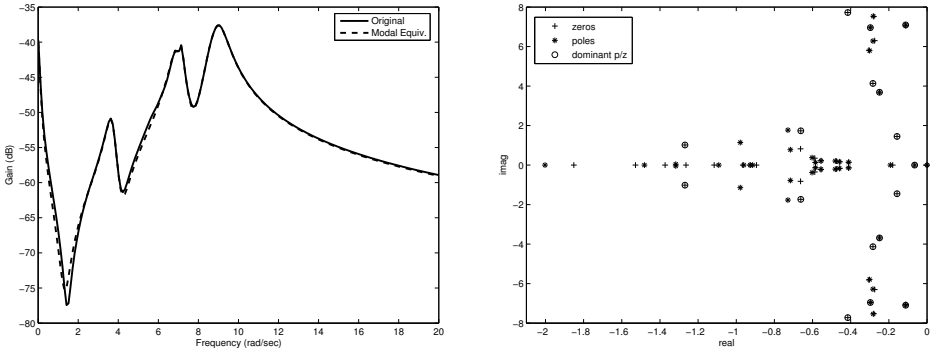
The problem of finding the invariant zeros of a square system can also be formulated as the generalized eigenvalue problem

$$A_z \mathbf{x}_z = \lambda E_z \mathbf{x}_z, \quad \mathbf{x}_z \in \mathbb{C}^{n+m}, \tag{5.2.5}$$

where

$$A_z = \begin{bmatrix} A & B \\ -C^* & -D \end{bmatrix}, \quad E_z = \begin{bmatrix} E & 0 \\ 0 & 0 \end{bmatrix}, \quad \text{and } \mathbf{x}_z = \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix}.$$

There are also definitions of input decoupling zeros and output decoupling zeros, and system zeros [135], but these are beyond the scope of this chapter.



(a) (b)

Figure 5.1: (a) Bode plot of transfer function $W_{36}/Pmec_{36}$ of the New England test system (66 states, see also [91]), together with a 11th order modal equivalent. (b) Part of corresponding pole ($*$) - zero ($+$) plot. The dominant zeros (poles), marked by encircled $+$-s ($*$-s), cause the dips (peaks) in the Bode plot.

Whereas a dominant pole causes a peak in a Bode plot of the transfer function, a dominant zero cause a dip, see also Fig. 5.1. In other words, the dominant zeros are the zeros close to the imaginary axis, that 'stand out' in the pole-zero plot, see also Fig. 5.1. In Section 5.3.3, it will be shown that the zeros are the poles of the inverse system, if it exists. Then, a zero is called dominant if it is a dominant pole of the inverse system (by Definition 5.2.1). It must be stressed that in different applications different definitions of the dominance of a zero can be used. In stabilization studies, for example, the rightmost and lowest damped zeros are the dominant zeros.

5.2.3 Practical need for the computation of dominant transfer function zeros

Theoretical considerations

Consider the block diagram of a typical control system depicted in Fig. 5.2. Let $n_G(s)$, $d_G(s)$, $n_K(s)$ and $d_K(s)$ be polynomials of s defining scalar proper rational functions

$$G(s) = \frac{n_G(s)}{d_G(s)} \quad \text{and} \quad K(s) = \frac{n_K(s)}{d_K(s)},$$

where $G(s)$ represents the system dynamics to be controlled and $K(s)$ is the designed controller. It is easily verified that the closed-loop transfer function $H(s)$ is

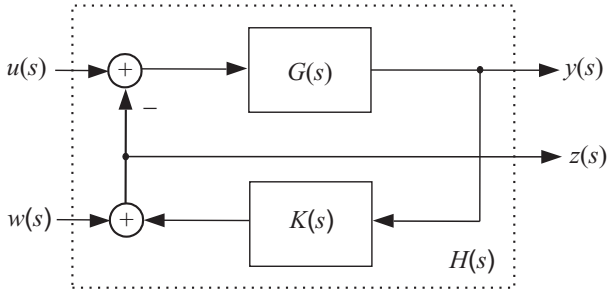


Figure 5.2: Closed-loop system.

a (2×2) transfer matrix:

$$\begin{bmatrix} y(s) \\ z(s) \end{bmatrix} = \underbrace{\begin{bmatrix} H_{yu}(s) & H_{yw}(s) \\ H_{zu}(s) & H_{zw}(s) \end{bmatrix}}_{H(s)} \begin{bmatrix} u(s) \\ w(s) \end{bmatrix},$$

with

$$\begin{aligned} H &= \begin{bmatrix} G & -G \\ GK & 1 \end{bmatrix} \tfrac{1}{1+GK} \\ &= \begin{bmatrix} n_G d_K & -n_G d_K \\ n_G n_K & d_G d_K \end{bmatrix} \tfrac{1}{d_G d_K + n_G n_K}. \end{aligned} \tag{5.2.6}$$

The focus here is on the main diagonal elements of the numerator matrix of H shown in (5.2.6). While the transfer function $H_{yu}(s)$ represents the actual closed-loop control channel, the input-output pair (w, z) defines a synthetic disturbance channel $H_{zw}(s)$ used here only for analysis purposes. It is clear from (5.2.6) that the set of zeros of H_{yu} and H_{zw} can be split in two subsets:

$$\begin{aligned} \{\text{zeros of } H_{yu}\} &= \{\text{zeros of } G\} \cup \{\text{poles of } K\}, \\ \{\text{zeros of } H_{zw}\} &= \{\text{poles of } G\} \cup \{\text{poles of } K\}. \end{aligned} \tag{5.2.7}$$

As the controller K is determined by the designer and is, in general, of small order, the subset of closed-loop zeros corresponding to the poles of K are already known and not relevant to the current analysis. The remaining closed-loop zeros are either open-loop zeros (for the control channel (H_{yu})) or open-loop poles (for the disturbance channel (H_{zw})) of possibly large-scale transfer functions.

The case where the signals u, y, w and z are vectors of compatible dimensions is also of interest. In this case, the MIMO closed-loop transfer functions for the control and disturbance channels are, respectively:

$$\begin{aligned} H_{yu} &= (I + GK)^{-1} G = G(I + KG)^{-1}, \text{ and} \\ H_{zw} &= (I + KG)^{-1}. \end{aligned} \tag{5.2.8}$$

Considering a right and a left coprime factorization for G and K, respectively, as

$$G = N_G D_G^{-1} \text{ and } K = D_K^{-1} N_K,$$

(N_G, D_G, N_K, D_K polynomial matrices, with D_G, D_K nonsingular) and substituting them in (5.2.8) yields

$$\begin{aligned} H_{zw} &= (I + D_K^{-1} N_K N_G D_G^{-1})^{-1} \\ &= D_G (D_K D_G + N_K N_G)^{-1} D_K \end{aligned} \tag{5.2.9}$$

that pre-multiplied by $N_G D_G^{-1}$ becomes

$$H_{yu} = N_G (D_K D_G + N_K N_G)^{-1} D_K. \tag{5.2.10}$$

Since D_G and N_G are numerators in (5.2.9) and (5.2.10), respectively, one can draw similar conclusions regarding closed-loop zeros for MIMO systems (even for non-square G and K).

Practical applications

a. The first practical application comes from the fact that the closed-loop transfer functions H_{zw} and H_{yu} have some zeros on the two extreme points of relevant root locus branches. In other words, some branches of interest start at the zeros of H_{zw} (null loop gain) and end at the zeros of H_{yu} (infinite loop gain). Hence, an efficient algorithm for the computation of dominant transfer function zeros, applied to adequately chosen closed-loop channels, is useful to the stability analysis of large-scale control systems.

b. There are at least two other interesting uses of the property (5.2.7) in the dominant zeros context. Some important features of the dominant open- and closed-loop poles can be catched from a single frequency response diagram, making the control analysis easier. For instance, given a stabilizing controller K, the Bode plot of the synthetic disturbance channel H_{zw} provides good estimates for the frequency and damping ratio of the open- (dips) and closed-loop (peaks) poles.

c. Dominant transmission zeros properties are applied to identify the best candidate locations for additional stabilizers to most effectively damp electromechanical modes in large power systems, as described in the closure of [92] and in [24].

d. Dominant zeros of driving point impedances are computed and effectively shifted, by use of eigenvalue sensitivity coefficients, in the harmonic distortion analysis of industrial power systems [160].

5.3 Computing transfer function zeros

The algorithms for the computation of zeros in this chapter are in fact Newton schemes and have many similarities with the dominant pole algorithms DPA [91],

SADPA [123] and SAMDP [122]. Algorithms that compute a single zero are shown in Section 5.3.1 for the SISO case and in Section 5.3.2 for the MIMO case. In Section 5.3.3 it is shown that the zeros are poles of the inverse transfer function. Consequently, the SADPA and SAMDP algorithms can be used to compute the dominant zeros via the poles of the inverse transfer function, as will be described in Section 5.4.

Earlier work in this direction was done in [92]. Full space methods based on the QZ algorithm [101, 64] to compute *all* zeros of a system are described in [44, 100]. These methods are of complexity $O((n + \max(m, p))^3)$ and hence only applicable to moderately sized systems. The methods described in this chapter are intended for very large systems, where one is specifically interested in the dominant zeros.

5.3.1 SISO Dominant Zero Algorithm (DZA)

Newton's method can be used to compute the zeros $z_i \in \mathbb{C}$ of the objective function

$$f : \mathbb{C} \longrightarrow \mathbb{C} : s \longmapsto \mathbf{c}^*(sE - A)^{-1}\mathbf{b} + d. \tag{5.3.1}$$

The derivative of the inverse of a square matrix valued function $Y(s)$ is given by

$$\frac{d[Y^{-1}(s)]}{ds} = -Y^{-1}(s)Y'(s)Y^{-1}(s),$$

and it follows that the derivative of the objective function (5.3.1) is

$$f'(s) = -\mathbf{c}^*(sE - A)^{-1}E(sE - A)^{-1}\mathbf{b}. \tag{5.3.2}$$

Using (5.3.1) and (5.3.2), the Newton scheme for the computation of a zero becomes

$$\begin{aligned} s_{k+1} &= s_k - \frac{f(s_k)}{f'(s_k)} \\ &= s_k + \frac{\mathbf{c}^*(s_kE - A)^{-1}\mathbf{b} + d}{\mathbf{c}^*(s_kE - A)^{-1}E(s_kE - A)^{-1}\mathbf{b}} \\ &= s_k + \frac{\mathbf{c}^*\mathbf{v} + d}{\mathbf{w}^*E\mathbf{v}}, \end{aligned}$$

where

$$\mathbf{v} = (s_kE - A)^{-1}\mathbf{b}, \text{ and } \mathbf{w} = (s_kE - A)^{-*}\mathbf{c}.$$

An implementation of this scheme is shown in Alg. 5.1. In the neighborhood of a solution the algorithm converges quadratically. A different, more difficult formulation that works with Σ (5.2.3) can be found in [92].

The DZA scheme only differs a minus sign from the Newton scheme for finding dominant poles (DPA, see [91, 20, 21]). For DPA, the objective function is $1/f(s)$

Algorithm 5.1 Dominant Zero Algorithm (DZA)

INPUT: System $(E, A, \mathbf{b}, \mathbf{c}, d)$, initial zero estimate $s_1 \in \mathbb{C}$, tolerance $\epsilon \ll 1$
OUTPUT: Approximate zero λ and corresponding right and left zero states $\mathbf{x}$ and $\mathbf{y}$

1: Set $k = 1$
2: **while** not converged **do**
3: Solve $\mathbf{v} \in \mathbb{C}^n$ from $(s_k E - A)\mathbf{v} = \mathbf{b}$
4: Solve $\mathbf{w} \in \mathbb{C}^n$ from $(s_k E - A)^* \mathbf{w} = \mathbf{c}$
5: Compute the new zero estimate

$$s_{k+1} = s_k + \frac{\mathbf{c}^* \mathbf{v} + d}{\mathbf{w}^* E \mathbf{v}}$$

6: The zero $\lambda = s_{k+1}$ with $\mathbf{x} = \mathbf{v}$ and $\mathbf{y} = \mathbf{w}$ has converged if $||\mathbf{r}_k||_2 < \epsilon$, where

$$\mathbf{r}_k = \begin{bmatrix} s_{k+1}E - A & -\mathbf{b} \\ \mathbf{c}^* & d \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix}$$

7: Set $k = k + 1$
8: **end while**

and the scheme becomes (see also [123])

$$\begin{aligned} s_{k+1}^{dpa} &= s_k^{dpa} - \frac{\mathbf{c}^*(s_k^{dpa}E - A)^{-1}\mathbf{b} + d}{\mathbf{c}^*(s_k^{dpa}E - A)^{-1}E(s_k^{dpa}E - A)^{-1}\mathbf{b}} \\ &= \frac{\mathbf{w}^* A \mathbf{v}}{\mathbf{w}^* E \mathbf{v}} \\ &\equiv \rho(A, E, \mathbf{v}, \mathbf{w}), \end{aligned}$$

where $\rho(A, E, \mathbf{v}, \mathbf{w})$ is called the two-sided Rayleigh quotient [110]. Similarly, it follows that in DZA, the new shift s_{k+1} can be written as a two-sided Rayleigh

quotient $\rho(A_z, E_z, \mathbf{v}_z, \mathbf{w}_z)$:

$$\begin{aligned} s_{k+1}^{dza} &= s_k^{dza} + \frac{\mathbf{c}^*\mathbf{v} + d}{\mathbf{w}^* E \mathbf{v}} \\ &= \frac{\mathbf{w}^*(A + (s_k^{dza}E - A))\mathbf{v}}{\mathbf{w}^* E \mathbf{v}} + \frac{\mathbf{c}^*\mathbf{v} + d}{\mathbf{w}^* E \mathbf{v}} \\ &= \frac{\mathbf{w}^*(A\mathbf{v} + \mathbf{b}) + \mathbf{c}^*\mathbf{v} + d}{\mathbf{w}^* E \mathbf{v}} \\ &= \frac{\begin{bmatrix}\mathbf{w}^* & 1\end{bmatrix}\begin{bmatrix}A & \mathbf{b}\\ \mathbf{c}^* & d\end{bmatrix}\begin{bmatrix}\mathbf{v}\\ 1\end{bmatrix}}{\begin{bmatrix}\mathbf{w}^* & 1\end{bmatrix}\begin{bmatrix}E & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\mathbf{v}\\ 1\end{bmatrix}} \\ &= \frac{\mathbf{w}_z^* A_z \mathbf{v}_z}{\mathbf{w}_z^* E_z \mathbf{v}_z} \\ &\equiv \rho(A_z, E_z, \mathbf{v}_z, \mathbf{w}_z), \end{aligned}$$

with

$$A_z = \begin{bmatrix}A & \mathbf{b}\\ \mathbf{c}^* & d\end{bmatrix}, \quad E_z = \begin{bmatrix}E & 0\\ 0 & 0\end{bmatrix}, \quad \mathbf{v}_z = \begin{bmatrix}\mathbf{v}\\ 1\end{bmatrix}, \quad \mathbf{w}_z = \begin{bmatrix}\mathbf{w}\\ 1\end{bmatrix}.$$

Like the poles can also be computed with two-sided Rayleigh quotient iteration (two-sided RQI) [110] applied to (A, E), the zeros can be computed by applying two-sided RQI to (A_z, E_z) as well. However, as is discussed in [126] (see Chapter 2), DPA (DZA) tends to converge to dominant poles (zeros), while two-sided RQI tends to converge to poles (zeros) closest to the initial shift.

5.3.2 MIMO Dominant Zero Algorithm (MDZA)

The objective function for finding the zeros of a general MIMO transfer function (5.2.2) is

$$f : \mathbb{C} \longrightarrow \mathbb{R} : s \longmapsto \sigma_{\min}(H(s)). \tag{5.3.3}$$

Let $(\sigma_{\min}(s), \mathbf{u}(s), \mathbf{z}(s))$ be the minimal singular triplet of $H(s)$, i.e. $H(s)\mathbf{z}(s) = \sigma_{\min}(s)\mathbf{u}(s)$ and $H^*(s)\mathbf{u}(s) = \sigma_{\min}(s)\mathbf{z}(s)$. It follows that $H^*(s)H(s)\mathbf{z}(s) = \sigma_{\min}^2\mathbf{z}(s)$. It is more convenient to work with the following objective function (equivalent to (5.3.3)):

$$f : \mathbb{C} \longrightarrow \mathbb{R} : s \longmapsto \lambda_{\min}(H^*(s)H(s)), \tag{5.3.4}$$

with $\lambda_{\min} = \sigma_{\min}^2$. Because $f(s)$ in (5.3.4) is a function from $\mathbb{C} \longrightarrow \mathbb{R}$, in general the derivative $df(s)/ds$ is not injective. A complex scalar $z = a + ib \in \mathbb{C}$ can be represented by $[a, b]^T \in \mathbb{R}^2$. The partial derivatives of the objective function (5.3.4) become

$$\begin{aligned} \frac{\partial f}{\partial a}(s) &= -\sigma_{\min}(s)(\mathbf{w}^* E \mathbf{v} + (\mathbf{w}^* E \mathbf{v})^*), \\ \frac{\partial f}{\partial b}(s) &= -i\sigma_{\min}(s)(\mathbf{w}^* E \mathbf{v} - (\mathbf{w}^* E \mathbf{v})^*), \end{aligned}$$

where

$$\mathbf{v} = (sE - A)^{-1}B\mathbf{z}, \quad \mathbf{w} = (sE - A)^{-*}C\mathbf{u}.$$

The derivative of (5.3.4) then becomes

$$\nabla f = -2\sigma_{\min}(s)[\mathrm{Re}(\mathbf{w}^*E\mathbf{v}), -\mathrm{Im}(\mathbf{w}^*E\mathbf{v})],$$

where $\mathrm{Re}(a + ib) = a$ and $\mathrm{Im}(a + ib) = b$. The Newton scheme is

$$\begin{aligned}\begin{bmatrix}\mathrm{Re}(s_{k+1})\\ \mathrm{Im}(s_{k+1})\end{bmatrix} &= \begin{bmatrix}\mathrm{Re}(s_k)\\ \mathrm{Im}(s_k)\end{bmatrix} - (\nabla f(s_k))^{\dagger} f(s_k)\\ &= \begin{bmatrix}\mathrm{Re}(s_k)\\ \mathrm{Im}(s_k)\end{bmatrix}\\ &+ [\mathrm{Re}(\mathbf{w}^*E\mathbf{v}), -\mathrm{Im}(\mathbf{w}^*E\mathbf{v})]^{\dagger}\frac{\sigma_{\min}}{2},\end{aligned}$$

where $Y^{\dagger} = Y^*(YY^*)^{-1}$ denotes the pseudo-inverse of a matrix $Y \in \mathbb{C}^{n\times m}$ with $\mathrm{rank}(Y) = n$ $(n \leq m)$ [64].

This generalization of the Newton method is also known as the normal flow method [163] and can be shown, under certain conditions, to converge (super) linearly. The algorithm for systems with full column rank transfer functions $H(s)$ $(p \geq m)$ is shown in Alg. 5.2. If $p = m$, both right and left zero states are returned. If $p > m$, the right zero states are returned. If $p < m$, the left zero states can be computed by calling the algorithm with the dual system $(E^*, A^*, -C, -B, D^*)$.

For SISO systems, the algorithm simplifies to Alg. 5.1. For square MIMO systems, it is advised to follow an approach similar to the MDPA algorithm for square systems in [122]: if the objective function

$$f : \mathbb{C} \longrightarrow \mathbb{C} : s \longmapsto \lambda_{\min}(H(s))$$

is taken, the scheme simplifies to a standard Newton scheme, where no pseudo-inverses are needed and convergence is asymptotically quadratic.

5.3.3 Zeros as the poles of the inverse transfer function

In this section realizations of inverse systems are given. An inverse system is a system whose transfer function is the inverse of the transfer function of the original system. The zeros of the original transfer function become poles of the inverse transfer function. This fact can be used to compute the zeros via the poles of the inverse transfer function, using the SADPA [123] and SAMDP [122] algorithms, as is shown in Section 5.4. In principle, a modal equivalent of the inverse transfer function can be build from the dominant poles and corresponding residues of the inverse system.

SISO transfer functions

It is well known that for SISO systems, the poles of the high gain output feedback closed-loop system approach the zeros of the open-loop system $\Sigma = (E, A, \mathbf{b}, \mathbf{c}, d)$.

Algorithm 5.2 MIMO Transmission Zero Algorithm (MDZA)

INPUT: System (E, A, B, C, D) with $p \geq m$, initial pole estimate s_1, tolerance $\epsilon \ll 1$

OUTPUT: Approximate zero λ and corresponding right and/or left state zero directions, inputs $\mathbf{v}, \mathbf{u}$ and $\mathbf{w}, \mathbf{v}$.

1: Set $k = 1$
2: **while** not converged **do**
3: Compute singular triplet $(\sigma_{\min}, \mathbf{z}, \mathbf{u})$ of $H(s_k)$
4: Solve $\mathbf{v} \in \mathbb{C}^n$ from $(s_k E - A)\mathbf{v} = B\mathbf{z}$
5: Solve $\mathbf{w} \in \mathbb{C}^n$ from $(s_k E - A)^* \mathbf{w} = C\mathbf{u}$
6: Compute the new pole estimate

$$\begin{bmatrix} \mathrm{Re}(s_{k+1}) \\ \mathrm{Im}(s_{k+1}) \end{bmatrix} = \begin{bmatrix} \mathrm{Re}(s_k) \\ \mathrm{Im}(s_k) \end{bmatrix} + [\mathrm{Re}(\mathbf{w}^* E \mathbf{v}), -\mathrm{Im}(\mathbf{w}^* E \mathbf{v})]^\dagger \frac{\sigma_{\min}}{2},$$

7: The zero $\lambda = s_{k+1}$ with $\mathbf{x} = \mathbf{v}$ and $\mathbf{y} = \mathbf{w}$ has converged if $||\mathbf{r}_k||_2 < \epsilon$, where

$$\mathbf{r}_k = \begin{bmatrix} s_{k+1}E - A & -B \\ C^* & D \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{z} \end{bmatrix}$$

8: Set $k = k + 1$
9: **end while**

This result is used in some algorithms for the computation of transmission zeros [39, 92]. In the following it will be shown how, given a system $\Sigma = (E, A, \mathbf{b}, \mathbf{c}, d)$, an inverse system $\Sigma_z = (E_z, A_z, \mathbf{b}_z, \mathbf{c}_z, d_z)$ can be defined, so that the transfer function $H_z(s)$ of Σ_z is the inverse of the transfer function $H(s)$ of Σ. The important consequence of these results is that the dominant zeros become the dominant poles of the inverse transfer function, so that existing algorithms such as DPA and SADPA can be used to compute the dominant zeros via the dominant poles of the inverse system.

The following theorem defines a system $\Sigma_z = (E_z, A_z, \mathbf{b}_z, \mathbf{c}_z, 0)$, with transfer function $H_z(s)$, such that $H_z(s)H(s) = H(s)H_z(s) = 1$, where $H(s)$ is the transfer function of $\Sigma = (A, \mathbf{b}, \mathbf{c}, 0)$. Existing literature that mentions such a system (for $d = 0$) is not known to the authors.

Theorem 5.3.1. *Let*

$$y(s) = H(s)u(s) = \left[\mathbf{c}^*(sE - A)^{-1}\mathbf{b}\right] u(s)$$

be the transfer function of the descriptor realization $\Sigma = (E, A, \mathbf{b}, \mathbf{c}, d = 0)$*, and let*

$$y_z(s) = H_z(s)u_z(s) = \left[\mathbf{c}_z^*(sE_z - A_z)^{-1}\mathbf{b}_z\right] u_z(s)$$

be the transfer function of the augmented descriptor realization

$$\Sigma_z = (E_z, A_z, \mathbf{b}_z, \mathbf{c}_z, d_z = 0),$$

where

$$A_z = \begin{bmatrix} A & \mathbf{b} \\ -\mathbf{c}^* & 0 \end{bmatrix}, \quad E_z = \begin{bmatrix} E & 0 \\ 0 & 0 \end{bmatrix},$$
$$\mathbf{b}_z = \begin{bmatrix} \mathbf{b} \\ 1 \end{bmatrix}, \qquad \mathbf{c}_z = \begin{bmatrix} \mathbf{c} \\ 1 \end{bmatrix}, \qquad d_z = 0.$$

Then $H_z(s) = H^{-1}(s)$.

Proof. Let $u \equiv u(s)$, $y \equiv y(s) = \mathbf{c}^*\mathbf{x}(s)$, $\mathbf{x} \equiv \mathbf{x}(s) = (sE - A)^{-1}\mathbf{b}u(s)$, $u_z \equiv u_z(s)$, $y_z \equiv y_z(s) = \mathbf{c}_z^*\mathbf{x}_z(s)$, $\mathbf{x}_z \equiv \mathbf{x}_z(s) = (sE_z - A_z)^{-1}\mathbf{b}_z u_z(s)$, and the Rosenbrock system matrix [127] equation for Σ

$$\begin{bmatrix} sE - A & -\mathbf{b} \\ \mathbf{c}^* & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ u \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}.$$

By adding $(\mathbf{b}y - \mathbf{b}y)$ to the left side of the equation $(sE - A)\mathbf{x} - \mathbf{b}u = 0$ and defining the error $(u - y)$ as a new algebraic variable, one obtains

$$\left[\begin{array}{cc:c} sE - A & -\mathbf{b} & -\mathbf{b} \\ \mathbf{c}^* & 0 & -1 \\ \hdashline \mathbf{c}^* & 1 & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{x} \\ u - y \\ \hdashline y \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ \hdashline u \end{array}\right]$$

or

$$\begin{bmatrix} sE_z - A_z & -\mathbf{b}_z \\ \mathbf{c}_z^* & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_z \\ u_z \end{bmatrix} = \begin{bmatrix} 0 \\ y_z \end{bmatrix},$$

which is the Rosenbrock system matrix equation for Σ_z. Hence $u_z(s) = y(s)$, $y_z(s) = u(s)$ and $H_z(s) = H^{-1}(s)$. □

A similar result can be obtained for the case $d \neq 0$, see for example [78, Ex. 2.2-20] and [79, 137], which is described below in the form of a theorem.

Theorem 5.3.2. *Let*

$$y(s) = H(s)u(s) = \left[\mathbf{c}^*(sE - A)^{-1}\mathbf{b} + d\right] u(s)$$

be the transfer function of the descriptor realization $\Sigma = (E, A, \mathbf{b}, \mathbf{c}, d \neq 0)$*, and let*

$$y_z(s) = H_z(s)u_z(s) = \left[\mathbf{c}_z^*(sE_z - A_z)^{-1}\mathbf{b}_z + d_z\right] u_z(s)$$

be the transfer function of the descriptor realization $\Sigma_z = (E_z, A_z, \mathbf{b}_z, \mathbf{c}_z, d_z \neq 0)$*, where*

$$\begin{aligned} A_z &= A - d^{-1}\mathbf{b}\mathbf{c}^*, \qquad E_z = E, \\ \mathbf{b}_z &= d^{-1}\mathbf{b}, \quad \mathbf{c}_z = -d^{-1}\mathbf{c}, \quad d_z = d^{-1}. \end{aligned}$$

Then $H_z(s) = H^{-1}(s)$*.*

Proof. Let $u \equiv u(s)$, $y \equiv y(s) = \mathbf{c}^*\mathbf{x}(s) + du(s)$, $\mathbf{x} \equiv \mathbf{x}(s) = (sE - A)^{-1}\mathbf{b}u(s)$, $u_z \equiv u_z(s)$, $y_z \equiv y_z(s) = \mathbf{c}_z^*\mathbf{x}_z(s) + d_z u_z(s)$, $\mathbf{x}_z \equiv \mathbf{x}_z(s) = (sE_z - A_z)^{-1}\mathbf{b}_z u_z(s)$, and the Rosenbrock system matrix equation for Σ

$$\begin{bmatrix} sE - A & -\mathbf{b} \\ \mathbf{c}^* & d \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ u \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}. \tag{5.3.5}$$

The expression $u = -d^{-1}\mathbf{c}^*\mathbf{x} + d^{-1}y$ is obtained from the second equation of (5.3.5), which is substituted into the first equation of (5.3.5) to yield

$$\begin{bmatrix} sE - (A - d^{-1}\mathbf{b}\mathbf{c}^*) & -d^{-1}\mathbf{b} \\ -d^{-1}\mathbf{c}^* & d^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_z \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ u \end{bmatrix},$$

or

$$\begin{bmatrix} sE_z - A_z & -\mathbf{b}_z \\ \mathbf{c}_z^* & d_z \end{bmatrix} \begin{bmatrix} \mathbf{x}_z \\ u_z \end{bmatrix} = \begin{bmatrix} 0 \\ y_z \end{bmatrix},$$

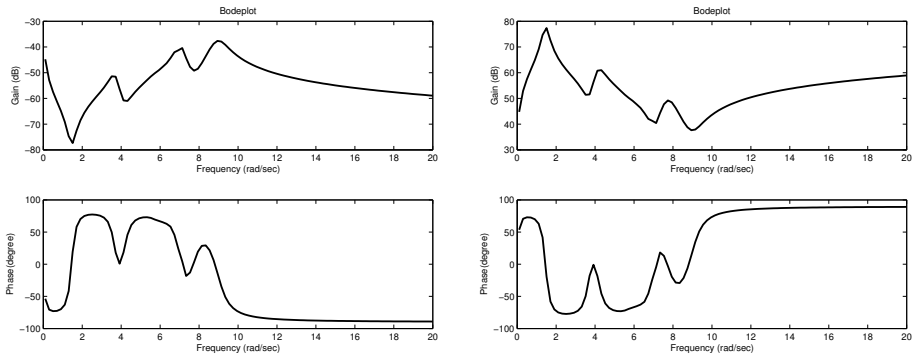
which is the Rosenbrock system matrix equation for Σ_z. Hence $u_z(s) = y(s)$, $y_z(s) = u(s)$ and $H_z(s) = H^{-1}(s)$. □

In fact, DZA applied to $(E, A, \mathbf{b}, \mathbf{c}, d)$ and DPA applied to $(E_z, A_z, \mathbf{b}_z, \mathbf{c}_z, d_z)$ are equivalent processes and hence produce the same results, provided the initial shift is the same:

Theorem 5.3.3. *DZA(E, A,* $\mathbf{b}$*,* $\mathbf{c}$*, d, s*$_0$*) and DPA(*E_z*,* A_z*,* $\mathbf{b}_z$*,* $\mathbf{c}_z$*,* d_z*, s*$_0$*) are equivalent.*

Proof. First consider the case $d = d_z = 0$ (cf. Theorem 5.3.1 for notation). It can be verified, with $\mathbf{v} = (s_k E - A)^{-1}\mathbf{b}$ and $\mathbf{w} = (s_k E - A)^{-*}\mathbf{c}$, that

$$\mathbf{v}_z = \frac{1}{\mathbf{c}^*\mathbf{v}} \begin{bmatrix} \mathbf{v} \\ 1 - \mathbf{c}^*\mathbf{v} \end{bmatrix} \quad \text{and} \quad \mathbf{w}_z = \frac{1}{\mathbf{b}^*\mathbf{w}} \begin{bmatrix} -\mathbf{w} \\ 1 + \mathbf{b}^*\mathbf{w} \end{bmatrix}$$



(a) Bode plot of $H(s)$. (b) Bode plot of $H_z(s)$.

Figure 5.3: Bode plots of the New England test system (66 states, see also Fig. 5.1): (a) strictly proper transfer function $H(s) = W_{36}(s)/Pmec_{36}(s)$; (b) improper transfer function $H_z(s) = H^{-1}(s)$. The dominant zeros of $H(s)$ (dips) are the dominant poles of $H_z(s)$ (peaks).

are solutions of

$$(s_k E_z - A_z)\mathbf{v}_z = \mathbf{b}_z \quad \text{and} \quad (s_k E_z - A_z)^* \mathbf{w}_z = \mathbf{c}_z.$$

Since $(\mathbf{b}^*\mathbf{w})^* = \mathbf{c}^*\mathbf{v}$, it follows that if $s_k^{dza} = s_k^{dpa}$, then

$$s_{k+1}^{dpa} = s_k^{dpa} - \frac{\mathbf{c}_z^*\mathbf{v}_z}{\mathbf{w}_z^* E_z \mathbf{v}_z} = s_k^{dza} + \frac{\mathbf{c}^*\mathbf{v}}{\mathbf{w}^* E \mathbf{v}} = s_{k+1}^{dza}.$$

Because $s_0^{dza} = s_0^{dpa}$ it follows that $s_k^{dza} = s_k^{dpa}$ for all $k \geq 0$.

A similar argument can be used for the case $d \neq 0$ (cf. Theorem 5.3.2), by noting that

$$\mathbf{v}_z = \frac{1}{\mathbf{c}^*\mathbf{v} + d}\mathbf{v} \quad \text{and} \quad \mathbf{w}_z = \frac{-1}{\mathbf{b}^*\mathbf{w} + d}\mathbf{w}$$

are solutions of

$$(sE_z - A_z)\mathbf{v}_z = \mathbf{b}_z \quad \text{and} \quad (sE_z - A_z)^* \mathbf{w}_z = \mathbf{c}_z.$$

□

This equivalence strengthens the motivation in Section 5.4 to use the SADPA and SAMDP algorithms for the computation of the dominant zeros.

The fact that $H_z(s) = H^{-1}(s)$ can also be observed from the symmetric Bode plots in Fig. 5.3, where it is clearly visible that the dominant zeros of $H(s)$ (dips) become the dominant poles of $H_z(s)$ (peaks). A dominant zero of $H(s)$ can now also be defined as a dominant pole of $H^{-1}(s)$.

Definition 5.3.4. *A zero z_i of an invertible transfer function $H(s)$ is called dominant if it is a dominant pole of $H^{-1}(s)$ (by Definition 5.2.1).*

MIMO transfer functions

Also for a square MIMO system $\Sigma = (E, A, B, C, D)$ with transfer function $H(s)$, a system Σ_z can be defined so that $H_z(s)$ is the inverse of $H(s)$. The result for systems with $D = 0$ is:

Theorem 5.3.5. *Let $H(s) = C^*(sE - A)^{-1}B$ be the transfer function of $\Sigma = (E, A, B, C, D = 0)$, and let $H_z(s) = C_z^*(sE_z - A_z)^{-1}B_z$ be the transfer function of $\Sigma_z = (E_z, A_z, B_z, C_z, D_z = 0)$, where*

$$A_z = \begin{bmatrix} A & B \\ -C^* & 0 \end{bmatrix}, \quad E_z = \begin{bmatrix} E & 0 \\ 0 & 0 \end{bmatrix},$$
$$B_z = \begin{bmatrix} B \\ I_m \end{bmatrix}, \quad C_z = \begin{bmatrix} C \\ I_m \end{bmatrix}, \quad D_z = 0,$$

and $I_m \in \mathbb{R}^{m\times m}$ is the $m \times m$ identity matrix. Then $H_z(s) = H^{-1}(s)$.

Proof. Manipulation using the Schur complement of A in A_z and basic linear algebra operations show that $H(s)H_z(s) = I_m$. □

For nonsingular D, a generalization of Theorem 5.3.2 can be obtained, see also [137]:

Theorem 5.3.6. *Let $H(s) = C^*(sE - A)^{-1}B + D$ be the transfer function of $\Sigma = (E, A, B, C, D$ non-singular$)$, and let $H_z(s) = C_z^*(sE_z - A_z)^{-1}B_z + D_z$ be the transfer function of $\Sigma_z = (E_z, A_z, B_z, C_z, D_z)$, where*

$$A_z = A - BD^{-1}C^*, \quad E_z = E, \quad B_z = BD^{-1}, \quad C_z = -D^{-1}C, \quad D_z = D^{-1}.$$

Then $H_z(s) = H^{-1}(s)$.

Proof. The Sherman-Morrison-Woodbury formula [64, p. 50] gives

$$\begin{aligned}&(sE - (A - BD^{-1}C^*)^{-1} \\ &= \tilde{E}(I_n - BD^{-1}(I_m + C^*\tilde{E}BD^{-1})^{-1}C^*\tilde{E}),\end{aligned}$$

where $\tilde{E} = (sE - A)^{-1}$, and using basic matrix-vector operations it follows that $H(s)H_z(s) = I_m$. □

The important consequence of these results is that the MIMO dominant pole algorithm SAMDP can be used to compute the dominant zeros, via the dominant poles of the inverse system. In fact, MDZA(E, A, B, C, D, s_0) and MDPA(E_z, A_z, B_z, C_z, D_z, s_0) are equivalent processes (cf. Theorem 5.3.3). Similar results may be obtained for non-square MIMO systems, using pseudo-inverses (see for instance [164]).

5.4 Computing more than one zero

It is not attractive to run (M)DZA for a number of different initial shifts to compute more than one dominant zero, since there is a big chance of recomputing an already found zero. Deflation, see for instance [132], is a well known technique to avoid this problem. For the computation of many dominant poles, deflation combined with subspace acceleration and a selection strategy has lead to the efficient Subspace Accelerated DPA (SADPA) [123] and SAMDP [122].

Although there are efficient methods for the computation of few eigenvalues of large-scale generalized eigenvalue problems of the form (5.2.5), such as the Jacobi-Davidson QZ method [51], these methods are less suitable for the computation of dominant zeros, since they are designed to compute a few eigenvalues near a specific target.

The dominant zeros of $H(s)$ can be computed as the dominant poles of $H_z(s)$, by applying SADPA (or SAMDP) to the inverse systems as defined in Section 5.3.3. This is the most flexible approach, since it not only allows for the computation of the dominant zeros in the sense of Definition 5.3.4, but can also easily be adapted to compute the RHP zeros or lowest damped zeros (by using a different selection criterion, see [123, 122] for more details). The realizations of the inverse systems can be constructed efficiently following the results in Section 5.3.3, and because the structure of A_z and E_z can be exploited in the solves with $s_k E_z - A_z$, the increase in computational time will be very limited.

The use of subspace acceleration has also the advantage of avoiding convergence to (extraneous) zeros at infinity, since the selection criterion takes care of selecting the most dominant estimates every iteration. Convergence to zeros at infinity occasionally (and eventually) happens for the single zero DZA algorithm. Deflation by restriction ([111], see also chapter 3) can be used to compute more than one zero with the single zero DZA algorithm, with the advantage of not keeping a search space. However, also this variant eventually converges to zeros at infinity, and furthermore, since it is a Newton scheme, it may get stuck in stable period 2 cycle of the Newton scheme itself. Subspace acceleration also prevents this and therefore the most attractive approach is to compute the zeros via the poles of the inverse system.

Another advantage of using SADPA/SAMDP for the computation of the dominant zeros is that a single algorithm can be used for both the computation of the dominant poles and zeros, so that only one implementation is needed.

5.5 Numerical experiments and applications

There are several applications where it is needed to compute some specific zeros. In the following subsections a number of SISO and MIMO transfer functions of the Brazilian Interconnect Power System (BIPS) are considered. See [123, 122] for more information on BIPS. The practical implementations operate on the sparse descriptor system model. All experiments were carried out in Matlab 7.2 on a SUN

Table 5.1: The 10 most dominant zeros and corresponding scaled residue matrix norms of the 2×2 transfer function of BIPS, sorted in decreasing $|R_i|/|\mathrm{Re}(\lambda_i)|$. Zeros found by SAMDP and/or SAMDZ are marked with 'yes'.

Modes		Residues	SAMDP	SAMDZA
Real	Imaginary	$\|R_i\|/\mathrm{Re}(\lambda_i)$	$s_0 = 10i$	$s_0 = 10i$
$-3.138 \cdot 10^{-2}$	± 4.356	1.010	yes	
$+4.484 \cdot 10^{-2}$	± 1.154	$1.761 \cdot 10^{-1}$	yes	
$-3.059 \cdot 10^{-1}$	± 3.667	$6.238 \cdot 10^{-2}$	yes	
$-6.203 \cdot 10^{-1}$	± 7.402	$3.709 \cdot 10^{-2}$	yes	yes
$-1.391 \cdot 10^{-1}$	± 0.2675	$3.305 \cdot 10^{-2}$	yes	
$-4.451 \cdot 10^{-1}$	± 3.015	$2.413 \cdot 10^{-2}$	yes	
-2.456		$2.031 \cdot 10^{-2}$		
-2.455		$1.711 \cdot 10^{-2}$		
$-7.478 \cdot 10^{-1}$	± 5.993	$1.269 \cdot 10^{-3}$		
-2.403		$8.678 \cdot 10^{-3}$		

Ultra 20 (AMD Opteron 2.8GHz), using codes based on the algorithms presented in [123, 122].

5.5.1 A comparison

The SADPA [123] and SAMDP [122] algorithms can be used to compute the dominant poles of the inverse system. To illustrate the importance of the selection criterion, SAMDP is compared to SAMDZ, subspace accelerated MDZA, with a (naive) selection criterion: select the estimates closest to the initial estimate s_0. As an example, consider the 2×2 transfer function of BIPS, whose inputs are the voltage references at the excitation systems for generators 1107 and 5061, and whose outputs are the rotor speed deviations for the same two generators (see also [123]). Of interest here are the non-minimum phase zero (in the right half plane), and a number of most dominant zeros. The non-minimum phase (complex conjugate) zero $z \approx 4.48 \cdot 10^{-2} \pm 1.15i$ is exactly the unstable (complex conjugate) pole associated with the North-South mode in the absence of power system stabilizers at the two generating stations (see Section 5.2.3 for the theoretical explanation). SAMDP with $s_0 = 10i$ not only finds the RHP zero, but also computes many of the other dominant zeros: in Table 5.1, the 10 most dominant zeros are listed, and the zeros computed by SAMDP and SAMDZ, respectively, are marked with 'yes'. Both methods were allowed to compute 30 (pairs of) zeros. SAMDP needed 50 seconds (227 LU-factorizations), while SAMDZ needed 45 seconds (141 LU-factorizations). Clearly, SAMDP applied to the inverse system is better than SAMDZ. While SAMDP finds dominant zeros, SAMDZ finds zeros in the neighborhood of the initial shift. Providing a different initial shift does not help SAMDZ, while SAMDP still finds the dominant zeros.

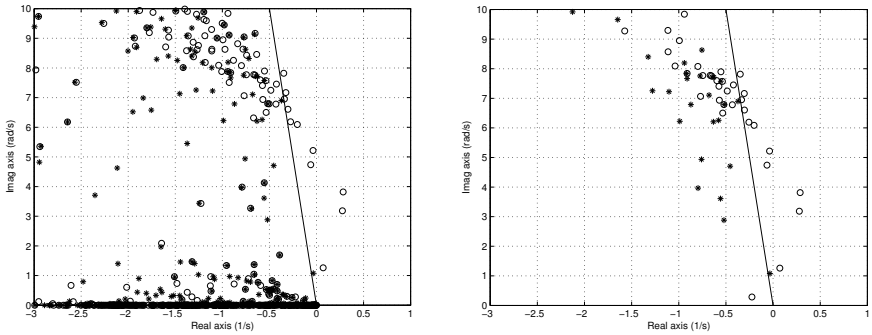


Figure 5.4: Pole-zero map of the 46×46 transfer matrix for disturbance channels (left), and poles and zeros found by SAMDP (right). Circles denote the transfer function zeros and stars denote the system poles. Complex zeros (and poles) appear in complex conjugate pairs, but here only the upper half of the complex plane is shown. Left half plane poles and zeros to the right of the solid line have damping ratio below 5%.

5.5.2 Computing dominant poles and zeros by SAMDP

Fig. 5.4 shows the pole-zero map for the 46×46 matrix transfer function $VPSS(s)/VPSS_d(s)$, where vector $VPSS_d$ contains input disturbance channels to each of the 46 PSSs in the system. The poles and zeros, and corresponding left and right eigenvectors, were obtained by use of the Matlab QR and QZ eigenroutines (`eig`, `schur`) in 146 seconds and 408 seconds, respectively. The zeros for this disturbance channel transfer function, as explained in Section 5.2.3, are identical to the system poles in the absence of PSSs. Note that the BIPS would be unstable at this operating point in the absence of PSSs: there are 3 unstable poles and 8 other poles (see the corresponding circles in Fig. 5.4 (left)) whose damping ratios are below 5%.

The SAMDP algorithm was applied to the inverse system in order to find rightmost dominant zeros for this 46×46 transfer matrix. The results are shown in Fig. 5.4 (right), where it is seen that the algorithm was able to capture the most relevant zeros. The 35 (complex conjugate) zeros, and corresponding left and right eigenvectors, were computed within 95 seconds (initial shift $s_0 = 1i$, 364 factorizations). Also the rightmost dominant poles computed by SAMDP (applied to the original system) are shown ($s_0 = 5i$, 35 (complex conjugate) triplets computed within 100 seconds and 408 factorizations). In order to select the rightmost zeros and poles, the selection criterion used in SAMDP was adapted to select the approximations with smallest damping ratio. Note that the CPU time required to compute the dominant poles and zeros is less than the time needed to compute all eigentriplets of the full state-space matrices (via QZ). For larger systems it is not feasible to compute all eigentriplets, while SAMDP can be applied to very large systems to compute the dominant poles and zeros (and corresponding left and right eigenvectors).

The results not only confirm that the dominant zeros can be computed via the dominant poles of the inverse system, but also show the flexibility of the SADPA/SAMDP algorithms when using different selection criteria.

5.6 Conclusions

This chapter described some control system applications where there is need for the computation of scalar as well as multivariable tranfer function zeros. It was shown that dominant zeros can be computed as the dominant poles of the inverse transfer function, by the SADPA and SAMDP algorithms. A new inverse system formulation, which is valid even for strictly proper transfer functions, is described in the chapter. This is a new finding, which is important from practical as well as theoretical viewpoints.

Various uses for transfer function zeros are described, and numerical results are given for multivariable system models of realistic power systems, showing the practical value and flexibility of the algorithms presented in this chapter.

Addendum

This chapter is also available as [125]

> Joost Rommes, Nelson Martins, and Paulo C. Pellanda, *Efficient computation of large-scale transfer function dominant zeros*, Preprint 1358, Utrecht University, 2006.

and has been submitted for publication.

Chapter 6

Efficient computation of transfer function dominant poles of large second-order dynamical systems

Abstract. This chapter presents a new algorithm for the computation of dominant poles of transfer functions of large-scale second-order dynamical systems: the Quadratic Dominant Pole Algorithm (QDPA). The algorithm works directly with the system matrices of the original system, so no linearization is needed. To improve global convergence, QDPA uses subspace acceleration, and deflation of found dominant poles is implemented in a very efficient way. The dominant poles and corresponding eigenvectors can be used to construct structure-preserving modal approximations, but also to improve reduced-order models computed by Krylov subspace methods, as is illustrated by numerical results. Generalizations to MIMO systems, higher-order systems and the computation of dominant zeros are also described.

Key words. poles, transfer function, quadratic eigenvalue problem, second-order dynamical systems, small-signal stability, transfer function residues, model reduction, large-scale systems, sparse eigenanalysis, modal equivalents, modal analysis, second-order Arnoldi

6.1 Introduction

A transfer function usually reflects the engineer's interest in studying a given part of a large dynamical system, and often only has a small number of dominant poles compared to the number of state variables. The dominant behavior of the system can be captured by projecting the state-space on the modes corresponding to the dominant poles. This type of model reduction is known as modal approximation, see for instance [159]. The computation of the dominant poles and modes requires specialized eigenvalue methods.

In [91] Newton's method is used to compute a dominant pole of single-input single-output (SISO) transfer function: the Dominant Pole Algorithm (DPA). In two recent publications this algorithm was improved and extended to a robust and efficient method for the computation of the dominant poles and corresponding eigenvectors of large scale SISO [123] (see Chapter 3) and multi-input multi-output (MIMO) transfer functions [122] (see Chapter 4).

In this chapter an efficient algorithm, Quadratic DPA (QDPA), for the computation of dominant poles of second-order transfer functions is presented. It is shown how DPA can be generalized to second and higher order polynomial transfer functions. Furthermore, it is described how subspace acceleration and efficient deflation can be added to obtain a more effective algorithm similar to SADPA. All algorithms presented work directly with the state-space matrices of the higher order system, i.e. no linearization is needed. Moreover, any modal equivalents that are constructed by projecting the state-space matrices on the dominant left and right eigenspaces, preserve the structure of the original system. The modal equivalents are compared to reduced-order models computed by (second-order) Arnoldi methods [12, 11] and it is shown how these models can be improved by using the dominant poles computed by QDPA.

The chapter is organized as follows. Section 6.2 gives definitions and properties of transfer functions and dominant poles, and describes the basic Quadratic DPA algorithm (QDPA). In Section 6.3 subspace acceleration and deflation are added. Numerical results are presented in Section 6.4. Generalizations to higher order polynomial transfer functions and MIMO systems, and computation of dominant zeros, are described in Section 6.5. Section 6.6 concludes.

6.2 Transfer functions and dominant poles

In this chapter, the second-order dynamical systems $(M, C, K, \mathbf{b}, \mathbf{c}, d)$ are of the form

$$\begin{cases} M\ddot{\mathbf{x}}(t) + C\dot{\mathbf{x}}(t) + K\mathbf{x}(t) &= \mathbf{b}u(t) \\ y(t) &= \mathbf{c}^*\mathbf{x}(t) + du(t), \end{cases} \tag{6.2.1}$$

where $M, C, K \in \mathbb{R}^{n\times n}$, $\mathbf{b}, \mathbf{c}, \mathbf{x}(t) \in \mathbb{R}^n$, $u(t), y(t), d \in \mathbb{R}$. In typical applications such as structural system analysis, M is the mass matrix, K is the stiffness matrix and C is the damping matrix. The vectors $\mathbf{b}$ and $\mathbf{c}$ are called the input and output vector, respectively. The transfer function $H : \mathbb{C} \longrightarrow \mathbb{C}$ of (6.2.1) is defined as

$$H(s) = \mathbf{c}^*(s^2M + sC + K)^{-1}\mathbf{b} + d. \tag{6.2.2}$$

The poles of transfer function (6.2.2) are a subset of the eigenvalues $\lambda_i \in \mathbb{C}$ of the quadratic eigenvalue problem

$$(\lambda^2 M + \lambda C + K)\mathbf{x} = 0, \qquad \mathbf{x} \neq 0.$$

Most material on quadratic eigenvalue problems in this section can be found in more detail in [10, 152]. An eigentriplet $(\lambda_i, \mathbf{x}_i, \mathbf{y}_i)$ is composed of an eigenvalue

λ_i and corresponding right and left eigenvectors $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{C}^n$ (identified by their components in $\mathbf{b}$ and $\mathbf{c}$):

$$\begin{cases} (\lambda_i^2 M + \lambda_i C + K)\mathbf{x}_i &= 0, \qquad \mathbf{x}_i \neq 0, \quad (i = 1, \ldots, 2n), \\ \mathbf{y}_i^*(\lambda_i^2 M + \lambda_i C + K) &= 0, \qquad \mathbf{y}_i \neq 0, \quad (i = 1, \ldots, 2n). \end{cases} \tag{6.2.3}$$

The quadratic eigenvalue problem (6.2.3) has $2n$ eigenvalues with at most $2n$ right (and left) eigenvectors. It is clear that $2n$ eigenvectors do not form an independent set in an n-dimensional space. Still, however, it is possible to obtain a partial fraction representation of the transfer function similar to the first order case. The approach followed here is also described in [152, Section 3.5]. If K is nonsingular, the quadratic eigenvalue problem (6.2.3) can be linearized to the generalized eigenproblem

$$A\phi_i = \lambda_i B\phi_i, \quad \psi_i^* A = \lambda_i \psi_i^* B, \quad \phi_i, \psi_i \neq 0, \quad (i = 1, \ldots, 2n),$$

where

$$A = \begin{bmatrix} 0 & -K \\ -K & -C \end{bmatrix}, \text{ and } B = \begin{bmatrix} -K & 0 \\ 0 & M \end{bmatrix}, \tag{6.2.4}$$

and

$$\phi_i = \begin{bmatrix} \mathbf{x}_i \\ \lambda \mathbf{x}_i \end{bmatrix}, \text{ and } \psi_i = \begin{bmatrix} \mathbf{y}_i \\ \bar{\lambda}_i \mathbf{y}_i. \end{bmatrix}$$

The corresponding linearized dynamical system is

$$\begin{cases} B\dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + \mathbf{b}_l u(t) \\ y(t) &= \mathbf{c}_l^* \mathbf{x}(t) + du(t), \end{cases} \tag{6.2.5}$$

where $\mathbf{b}_l = [0, \mathbf{b}^T]^T$ and $\mathbf{c}_l = [\mathbf{c}^T, 0]^T$. Note that also other linearizations are possible, but that this choice preserves symmetry, if M, C, and K are symmetric, and is convenient for other reasons as well, as will become clear in the following. Denoting the matrix polynomial in (6.2.3) by $Q(\lambda) = \lambda^2 M + \lambda C + K$, it can be verified that for nonsingular K one has

$$\begin{bmatrix} Q(\lambda) & 0 \\ 0 & I \end{bmatrix} = E(\lambda)(A - \lambda B)F(\lambda), \tag{6.2.6}$$

where

$$E(\lambda) = \begin{bmatrix} (C + \lambda M)K^{-1} & -I \\ -K^{-1} & 0 \end{bmatrix}, \text{ and } F(\lambda) = \begin{bmatrix} I & 0 \\ \lambda I & I \end{bmatrix},$$

i.e. the linear matrix $A - \lambda B$ is indeed a linearization of $Q(\lambda)$ [152, Section 3.5]. Some matrix manipulation of (6.2.6) gives

$$\begin{aligned} Q(\lambda)^{-1} &= \begin{bmatrix} I & 0 \end{bmatrix} F(\lambda)^{-1}(A - \lambda B)^{-1} E(\lambda)^{-1} \begin{bmatrix} I \\ 0 \end{bmatrix} \\ &= -\begin{bmatrix} I & 0 \end{bmatrix} (A - \lambda B)^{-1} \begin{bmatrix} 0 \\ I \end{bmatrix}. \end{aligned} \tag{6.2.7}$$

Now assume that M and K are nonsingular and that all eigenvalues are semisimple (i.e., for every eigenvalue the algebraic multiplicity is equal to the geometric multiplicity). Let $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_{2n})$ be a diagonal matrix with eigenvalues of (6.2.3), and let $X = [\mathbf{x}_1, \ldots, \mathbf{x}_{2n}]$ and $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_{2n}]$ have as their columns the corresponding right and left eigenvectors, respectively. Because the eigenvalues of $Q(\lambda)$ are semisimple, the eigenvalues of (A, B) are semisimple as well and (A, B) is diagonalizable, and the corresponding matrices with right and left eigenvectors are

$$\Phi = \begin{bmatrix} X \\ X\Lambda \end{bmatrix}, \text{ and } \Psi = \begin{bmatrix} Y \\ Y\Lambda^* \end{bmatrix}.$$

If Φ and Ψ are normalized so that $\Psi^* A \Phi = \Lambda$ and $\Psi^* B \Phi = I$, and if s is not an eigenvalue of (A, B) (or (6.2.3)), then

$$(A - sB)^{-1} = \Phi(\Lambda - sI)^{-1}\Psi^*,$$

and by (6.2.7) it follows that[1]

$$Q(s)^{-1} = X(sI - \Lambda)^{-1}\Lambda Y^*.$$

Finally, the partial fraction representation becomes

$$H(s) = \mathbf{c}^* X(sI - \Lambda)^{-1}\Lambda Y^* \mathbf{b} = \sum_{i=1}^{2n} \frac{R_i}{s - \lambda_i},$$

where the residues are given by

$$R_i = (\mathbf{c}^* \mathbf{x}_i)(\mathbf{y}_i^* \mathbf{b})\lambda_i. \tag{6.2.8}$$

Note that the $\mathbf{x}_i$ and $\mathbf{y}_i$ are scaled so that

$$\begin{bmatrix} \mathbf{y}_i^* & \lambda_i \mathbf{y}_i^* \end{bmatrix} \begin{bmatrix} -K & 0 \\ 0 & M \end{bmatrix} \begin{bmatrix} \mathbf{x}_i \\ \lambda_i \mathbf{x}_i \end{bmatrix} = -\mathbf{y}_i^* K \mathbf{x}_i + \lambda_i^2 \mathbf{y}_i^* M \mathbf{x}_i = 1. \tag{6.2.9}$$

Although there are different indices of modal dominance [4, 68, 123, 159], the following will be used in this chapter.

Definition 6.2.1. *A pole λ_i of $H(s)$ with corresponding right and left eigenvectors $\mathbf{x}_i$ and $\mathbf{y}_i$ ($-\mathbf{y}_i^* K \mathbf{x}_i + \lambda_i^2 \mathbf{y}_i^* M \mathbf{x}_i = 1$) is called* the *dominant pole if*

$$\widehat{R}_i \equiv \frac{|R_i|}{Re(\lambda_i)} > \widehat{R}_j$$

for all $j \neq i$.

More generally, a pole λ_i is called dominant if $|\widehat{R}_i|$ is not very small compared to $|\widehat{R}_j|$, for all $j \neq i$. This can also be seen in the corresponding Bode-plot, which is a plot of $|H(i\omega)|$ against $\omega \in \mathbb{R}$: peaks occur at frequencies ω close to the imaginary

[1] Note that Λ is missing in equation (3.11) in [152, Section 3.5].

parts of the dominant poles of $H(s)$. An approximation of $H(s)$ that consists of $k < 2n$ terms with $|R_j|$ above some value, determines the effective transfer function behavior [144] and is also known as transfer function modal equivalent:

$$H_k(s) = \sum_{j=1}^{k} \frac{R_j}{s - \lambda_j} + d.$$

Modal equivalents and reduced-order models that are constructed by projecting the state-space matrices on the dominant left and right eigenvectors, preserve the structure of the original system: if X and Y are $n \times k$ matrices with $k \ll n$ right and left eigenvectors corresponding to dominant poles, then the reduced-order model

$$\left\{ \begin{array}{lcl} M_r \ddot{\mathbf{x}}_r(t) + C_r \dot{\mathbf{x}}_r(t) + K_r \mathbf{x}_r(t) & = & \mathbf{b}_r u(t) \\ y_r(t) & = & \mathbf{c}_r^* \mathbf{x}_r(t) + du(t), \end{array} \right. \tag{6.2.10}$$

with $M_r = Y^*MX, C_r = Y^*CX, K_r = Y^*KX \in \mathbb{C}^{k \times k}$ and $\mathbf{b}_r = Y^*\mathbf{b}, \mathbf{c}_r = X^*\mathbf{c} \in \mathbb{C}^k$ is still a second-order system. In practice, it is advisable to make a real reduced model in the following way: for every complex pole triplet $(\lambda, \mathbf{x}, \mathbf{y})$, construct real bases for the right and left eigenspaces via $[\mathrm{Re}(\mathbf{x}), \mathrm{Im}(\mathbf{x})]$ and $[\mathrm{Re}(\mathbf{y}), \mathrm{Im}(\mathbf{y})]$, respectively. Let the columns of X_r and Y_r be such bases, respectively. Because the complex conjugate eigenvectors are also in this space, the real bases formed by the columns of X_r and Y_r for the eigenspaces are still (at most) k dimensional. The real reduced model can be formed by using X_r and Y_r in (6.2.10). Preserving the second-order structure is a desirable property from the modeling viewpoint, but also from the viewpoint of generating realizable reduced-order models (see also [55]). Note that although the k dominant poles are also poles of the original transfer function, the k-th order modal equivalent will have k other poles as well, since it is a second-order system.

The dominant poles are specific (complex) eigenvalues of the QEP (6.2.3) and usually form a small subset of the spectrum, so that rather accurate modal equivalents are possible for $k \ll n$. The dominant poles can be located anywhere in the spectrum, depending on the components of the corresponding eigenvectors in $\mathbf{b}$ and $\mathbf{c}$ [126] (see Chapter 2). Since the dominance of a pole is independent of d, $d = 0$ in the following without loss of generality.

6.2.1 The Quadratic Dominant Pole Algorithm (QDPA)

The poles of transfer function (6.2.2) are the $\lambda \in \mathbb{C}$ for which $\lim_{s \to \lambda} |H(s)| = \infty$. Consider now the function $G : \mathbb{C} \longrightarrow \mathbb{C}$

$$G(s) = \frac{1}{H(s)}.$$

For a pole λ of $H(s)$, $\lim_{s \to \lambda} G(s) = 0$. In other words, the poles are the roots of $G(s)$ and a good candidate to find these roots is Newton's method. This idea is the basis of the Dominant Pole Algorithm (DPA) [91] for first order transfer functions,

but can be generalized to transfer functions of any order (and to MIMO systems, see [93, 122]).

The derivative of $G(s)$ with respect to s is given by

$$G'(s) = -\frac{H'(s)}{H^2(s)}. \tag{6.2.11}$$

The derivative of $H(s) = \mathbf{c}^*(s^2M + sC + K)^{-1}\mathbf{b}$ with respect to s is

$$H'(s) = -\mathbf{c}^*(s^2M + sC + K)^{-1}(2sM + C)(s^2M + sC + K)^{-1}\mathbf{b}. \tag{6.2.12}$$

Equations (6.2.11) and (6.2.12) lead to the following Newton scheme:

$$\begin{aligned} s_{k+1} &= s_k - \frac{G(s_k)}{G'(s_k)} \\ &= s_k + \frac{1}{H(s_k)}\frac{H^2(s_k)}{H'(s_k)} \\ &= s_k - \frac{\mathbf{c}^*(s_k^2M + s_kC + K)^{-1}\mathbf{b}}{\mathbf{c}^*(s_k^2M + s_kC + K)^{-1}(2s_kM + C)(s_k^2M + s_kC + K)^{-1}\mathbf{b}} \\ &= s_k - \frac{\mathbf{c}^*\mathbf{v}}{\mathbf{w}^*(2s_kM + C)\mathbf{v}}. \end{aligned} \tag{6.2.13}$$

where $\mathbf{v} = (s_k^2M+s_kC+K)^{-1}\mathbf{b}$ and $\mathbf{w} = (s_k^2M+s_kC+K)^{-*}\mathbf{c}$. An implementation of this Newton scheme, called QDPA, is represented in Alg. 6.1.

Algorithm 6.1 Quadratic Dominant Pole Algorithm (QDPA)

INPUT: System $(M, C, K, \mathbf{b}, \mathbf{c})$, initial pole estimate s_0, tolerance $\epsilon \ll 1$

OUTPUT: Approximate dominant pole λ and corresponding right and left eigenvectors $\mathbf{x}$ and $\mathbf{y}$

1: Set $k = 0$

2: **while** not converged **do**

3: Solve $\mathbf{v}_k \in \mathbb{C}^n$ from $(s_k^2M + s_kC + K)\mathbf{v}_k = \mathbf{b}$

4: Solve $\mathbf{w}_k \in \mathbb{C}^n$ from $(s_k^2M + s_kC + K)^*\mathbf{w}_k = \mathbf{c}$

5: Compute the new pole estimate

$$s_{k+1} = s_k - \frac{\mathbf{c}^*\mathbf{v}_k}{\mathbf{w}_k^*(2s_kM + C)\mathbf{v}_k}$$

6: The pole $\lambda = s_{k+1}$ with $\mathbf{x} = \mathbf{v}_k/\|\mathbf{v}_k\|_2$ and $\mathbf{y} = \mathbf{w}_k/\|\mathbf{w}_k\|_2$ has converged if

$$\|(s_{k+1}^2M + s_{k+1}C + K)\mathbf{x}\|_2 < \epsilon$$

7: Set $k = k + 1$

8: **end while**

The two linear systems that need to be solved in step 3 and 4 of Alg. 6.1 can be efficiently solved using one LU-factorization $LU = s_k^2 M + s_k C + K$, by noting that $U^* L^* = (s_k^2 M + s_k C + K)^*$. If an exact LU-factorization is not available, one has to use inexact Newton schemes, such as Jacobi-Davidson style methods [139, 140, 142].

Note that QDPA operates on the original $n \times n$ state-space matrices and that no linearization is needed. This is an advantage, since applying standard DPA to the linearized system (6.2.5) requires factorizations of $2n \times 2n$ matrices (although the $2n \times 2n$ linear equations can also be solved using only factorization of $n \times n$ matrices, see also Section 6.3.2).

It is well known that when computing Rayleigh quotients corresponding to $\mathbf{v}$ and $\mathbf{w}$, this leads to a second-order equation and hence to *two* Ritz values [10]. Selection of the best Ritz value can be difficult [75]. However, in QDPA the choice is automatically made via the Newton update (cf. step 5).

The matrix $(s_{k+1}^2 M + s_{k+1} C + K)$ in step 6 can be reused in step 3 and 4 of the next iteration, where it is needed anyway. In step 5, however, it is in general (depending on the sparsity of M and C), more efficient to compute $(2s_k M + C)\mathbf{v}_k$ as $(2s_k)(M\mathbf{v}_k) + C\mathbf{v}_k$.

In [126] it is shown that DPA tends to converge to more dominant poles than two-sided Rayleigh quotient iteration. The fixed right-hand sides in DPA are crucial for this behavior, and it is expected that QDPA has the same desirable convergence. In [65, 66] DPA was applied to compute dominant poles of high-order rational (polynomial) matrices $Y(s)$ (nodal admittance matrices for electrical networks), which together with the derivative $Y'(s)$ have straightforward construction laws. Other Newton based schemes (of which QDPA in fact is a specialization) for the computation of eigenvalues are presented, for instance, in [130]. In [152, Section 6] and [99] large overviews of existing methods are given. The methods presented in this work focus on the computation of dominant poles.

6.3 Subspace acceleration, selection and deflation

QDPA can be extended with subspace acceleration and a selection strategy to improve global convergence to the most dominant poles, and deflation to avoid recomputation of already found poles. Although the ingredients are the same as for SADPA [123] and SAMDP [122], they are repeated here because especially the deflation procedure is more complicated for the quadratic eigenvalue problem [95].

6.3.1 Subspace acceleration and selection

Instead of discarding the intermediate approximations $\mathbf{v}_k$ and $\mathbf{w}_k$ of the right and left eigenvectors in step 3 and 4, they are kept in (bi)orthogonal search spaces V and W. Following the Petrov-Galerkin approach, the projected quadratic eigenvalue

problem becomes

$$(\tilde{\lambda}^2\widetilde{M} + \tilde{\lambda}\widetilde{C} + \widetilde{K})\tilde{\mathbf{x}} = 0, \text{ and } \tilde{\mathbf{y}}^*(\tilde{\lambda}^2\widetilde{M} + \tilde{\lambda}\widetilde{C} + \widetilde{K}) = 0, \tag{6.3.1}$$

where the $k \times k$ matrices $\widetilde{M}$, $\widetilde{C}$, $\widetilde{K}$ are given by

$$\widetilde{M} = W^*MV, \quad \widetilde{C} = W^*CV, \text{ and } \widetilde{K} = W^*KV.$$

In the k-th iteration this projected quadratic eigenvalue problem is of small size $k \ll n$ and hence can be solved via linearization and the QZ method. With the eigentriplets $(\tilde{\lambda}_i, \tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i)$ of (6.3.1), approximate eigentriplets of the original problem can be constructed as

$$(\hat{\lambda}_i = \tilde{\lambda}_i, \hat{\mathbf{x}}_i = V\tilde{\mathbf{x}}_i, \hat{\mathbf{y}}_i = W\tilde{\mathbf{y}}_i), \quad (i = 1, \ldots, k). \tag{6.3.2}$$

From these approximate triplets the most dominant on is selected. To determine the most dominant one, first the approximate left and right eigenvectors need to be normalized so that (cf. (6.2.9))

$$-\hat{\mathbf{y}}_i^*K\hat{\mathbf{x}}_i + \hat{\lambda}_i^2\hat{\mathbf{y}}_i^*M\hat{\mathbf{x}}_i = 1.$$

The approximate residues $\widehat{R}_i$ are given by

$$\widehat{R}_i = (\mathbf{c}^*\hat{\mathbf{x}}_i)(\hat{\mathbf{y}}_i^*\mathbf{b})\hat{\lambda}_i,$$

and the approximate triplets are sorted in decreasing $|\widehat{R}_i|/|\mathrm{Re}(\hat{\lambda}_i)|$ order. The shift in the next iteration becomes $s_{k+1} = \hat{\lambda}_1$.

Note that it is not necessary to compute the approximate eigentriplets (6.3.2) explicitly, since the approximate residues $\widehat{R}_i$ can be computed as

$$\widehat{R}_i = ((\mathbf{c}^*V)\tilde{\mathbf{x}}_i)(\tilde{\mathbf{y}}_i^*(W^*\mathbf{b}))\tilde{\lambda}_i \quad (= (\mathbf{c}^*\hat{\mathbf{x}}_i)(\hat{\mathbf{y}}_i^*\mathbf{b})\hat{\lambda}_i),$$

provided the $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{y}}_i$ are scaled so that

$$1 = -\tilde{\mathbf{y}}_i^*\widehat{K}\tilde{\mathbf{x}}_i + \tilde{\lambda}_i^2\tilde{\mathbf{y}}_i^*\widehat{M}\tilde{\mathbf{x}}_i \quad (= -\hat{\mathbf{y}}_i^*K\hat{\mathbf{x}}_i + \hat{\lambda}_i^2\hat{\mathbf{y}}_i^*M\hat{\mathbf{x}}_i).$$

Numerically, however, it is often more robust to normalize the approximate eigenvectors so that

$$\hat{\mathbf{x}}_i^*\hat{\mathbf{x}}_i = \hat{\mathbf{y}}_i^*\hat{\mathbf{y}}_i = 1,$$

since then the angles $\angle(\hat{\mathbf{x}}_i, \mathbf{c})$ and $\angle(\hat{\mathbf{y}}_i, \mathbf{b})$ are exploited [72, 126]. These scalings can still be performed on the vectors $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{y}}_i$.

The use of subspace acceleration has several advantages. Firstly, it may improve global convergence since the most dominant approximation is selected every iteration, and secondly, after convergence of a pole triplet already good approximations of other dominant pole triplets may be available. The price one has to pay are the costs for keeping the search spaces V and W orthogonal.

6.3.2 Deflation

In contrast to ordinary and generalized eigenproblems, the at most $2n$ eigenvectors of a quadratic eigenproblem obviously are not independent. Hence deflation by restriction cannot be applied directly (without linearization). Instead, an approach inspired by the deflation described in [95] is used here. The idea is to implement deflation via the linearized eigenproblem, since the eigenvectors of the linearized eigenproblem are independent (assuming all eigenvalues are nondegenerate). This can be organized in two non-equivalent ways, as will be discussed next.

In the following, suppose that the $(n \times k)$ matrices X and Y have as their columns the found right and left eigenvectors $\mathbf{x}_i$ and $\mathbf{y}_i$ $(i = 1, \ldots, k)$ of the QEP, respectively, and let Λ be a diagonal $(k \times k)$ matrix with the corresponding eigenvalues on its diagonal. The corresponding eigenvector matrices for the linearized problem (A, B) are

$$\Phi = \begin{bmatrix} X \\ X\Lambda \end{bmatrix}, \text{ and } \Psi = \begin{bmatrix} Y \\ Y\Lambda^* \end{bmatrix}.$$

Furthermore, let the eigenvectors in X and Y be normalized so that $\Psi^* A \Phi = \Lambda$ and $\Psi^* B \Phi = I$. Then it follows that the pencil

$$((I - B\Phi\Psi^*)A(I - \Phi\Psi^* B), (I - B\Phi\Psi^*)B(I - \Phi\Psi^* B))$$

has the same eigenvalues and eigenvectors as (A, B), but with the found eigenvalues transformed to zero.

Approach I: Orthogonalizing against found eigenvectors

A way to implement deflation for the expansion vectors $\mathbf{v}_k$ and $\mathbf{w}_k$ of the search spaces V and W of quadratic eigenproblem is:

1. Construct approximate eigenvectors for (A, B) via

$$\phi_k = \begin{bmatrix} \mathbf{v}_k \\ \sigma \mathbf{v}_k \end{bmatrix}, \quad \psi_k = \begin{bmatrix} \mathbf{w}_k \\ \bar{\sigma} \mathbf{w}_k \end{bmatrix},$$

   where σ is the corresponding approximate eigenvalue, computed via the Newton update (cf. (6.2.13)).

2. Project these approximations via

$$\begin{bmatrix} \phi^1 \\ \phi^2 \end{bmatrix} = (I - \Phi\Psi^* B)\phi_k, \text{ and } \begin{bmatrix} \psi^1 \\ \psi^2 \end{bmatrix} = (I - \Psi\Phi^* B^*)\psi_k.$$

3. Expand V and W (bi)orthogonally with ϕ^1 and ψ^1, respectively.

Approach II: Deflation by restriction of linearized input and output vectors

A second way is to take advantage of the following efficient deflation for first-order transfer functions. It can be verified ($\Psi^* B \Phi = I$) that with the deflated input and output vectors of the linearized problem

$$\mathbf{b}_d = \begin{bmatrix} \mathbf{b}_d^1 \\ \mathbf{b}_d^2 \end{bmatrix} = (I - B\Phi\Psi^*) \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}, \text{ and } \mathbf{c}_d = \begin{bmatrix} \mathbf{c}_d^1 \\ \mathbf{c}_d^2 \end{bmatrix} = (I - B^*\Psi\Phi^*) \begin{bmatrix} \mathbf{c} \\ 0 \end{bmatrix},$$

the residues of the deflated found poles are transformed to zero. Secondly, a straightforward manipulation shows that

$$(s_k B - A)^{-1} \mathbf{b}_d \perp B^*\Psi, \text{ and } (s_k B - A)^{-*} \mathbf{c}_d \perp B\Phi,$$

so that in principle no explicit B-orthogonalizations of expansion vectors against found eigenvectors are needed. This property makes deflation very cheap, since per found pole only once two projections are needed to compute $\mathbf{b}_d$ and $\mathbf{c}_d$.

The linear systems for DPA applied to the linearized system (6.2.4) are

$$(s_k B - A)\mathbf{v}_g = \mathbf{b}_d, \text{ and } (s_k B - A)^* \mathbf{w}_g = \mathbf{c}_d,$$

with A and B as in (6.2.4). These systems can be solved without forming A and B: with $\mathbf{v}_g^2$ and $\mathbf{w}_g^2$ solutions of

$$(s_k^2 M + s_k C + K)\mathbf{v}_g^2 = s_k \mathbf{b}_d^2 + \mathbf{b}_d^1, \text{ and } (s_k^2 M + s_k C + K)^* \mathbf{w}_g^2 = \bar{s}_k \mathbf{c}_d^2 + \mathbf{c}_d^1,$$

it follows that for $s_k \neq 0$

$$\mathbf{v}_g = \begin{bmatrix} (-K^{-1}\mathbf{b}_d^1 + \mathbf{v}_g^2)/s_k \\ \mathbf{v}_g^2 \end{bmatrix}, \text{ and } \mathbf{w}_g = \begin{bmatrix} (-K^{-*}\mathbf{c}_d^1 + \mathbf{w}_g^2)/\bar{s}_k \\ \mathbf{w}_g^2 \end{bmatrix}.$$

The search spaces V and W are expanded with $\mathbf{v}_g^1 = (-K^{-1}\mathbf{b}_d^1 + \mathbf{v}_g^2)/s_k$ and $\mathbf{w}_g^1 = (-K^{-*}\mathbf{c}_d^1 + \mathbf{w}_g^2)/\bar{s}_k$, respectively. Due to rounding errors, however, components in the direction of already found eigenvectors may enter the search spaces again. To avoid convergence to found eigentriplets it is therefore needed to compute the approximate residues via the deflated input and output vectors $\mathbf{b}_d$ and $\mathbf{c}_d$, so that in fact deflation takes place implicitly. If $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are approximate normalized right and left eigenvectors corresponding to $\hat{\lambda}$ of the QEP, then the approximate linearized residue is computed as

$$\widehat{R} = (\mathbf{c}_d^* \begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\lambda}\hat{\mathbf{x}} \end{bmatrix})(\begin{bmatrix} \hat{\mathbf{y}}^* & \hat{\lambda}\hat{\mathbf{y}}^* \end{bmatrix} \mathbf{b}_d).$$

Because the residues of already found poles are transformed to zero, there is practically no risk of selecting approximations of already found poles, so that expansion of the search spaces with already found components is limited.

The two approaches described in this section are not equivalent, i.e. given the same shift s_k, they do not compute the same expansion vectors, nor is it possible to

decide which is the best on pure theoretical grounds. An advantage of approach I is that the original Newton equations are solved, but on the other hand the Newton update needs to be computed in step 1, and moreover, orthogonalization must be carried out every iteration. The big advantage of approach II is that deflation needs to be carried out only once per found pole (for $\mathbf{b}$ and $\mathbf{c}$), while it is required that K is nonsingular, since one solve with K and K^* per found pole is needed to compute $\mathbf{b}_d$ and $\mathbf{c}_d$. Numerical experiments, reported in Section 6.4, show that approach II is more efficient and effective in practice.

6.3.3 Improving local convergence

As soon as an approximate pole triplet is accurate enough, for instance when $\|(s_{k+1}^2 M + s_{k+1}C + K)\mathbf{v}_k\|_2 < \epsilon_r$ with $\epsilon < \epsilon_r < 10^{-4}$, then convergence can be accelerated by using one or two iterations of quadratic two-sided Rayleigh Quotient Iteration, described in Alg. 6.2 (see [139] for a Jacobi-Davidson equivalent). The equation for the Rayleigh quotients, given left and right eigenvector approximations $\mathbf{w}_{k+1}$ and $\mathbf{v}_{k+1}$, is

$$(\tilde{\lambda}^2 \mathbf{w}_{k+1}^* M \mathbf{v}_{k+1} + \tilde{\lambda}\mathbf{w}_{k+1}^* C \mathbf{v}_{k+1} + \mathbf{w}_{k+1}^* K \mathbf{v}_{k+1})\tilde{\mathbf{x}} = 0,$$

which is a 1×1 QEP, and hence provides *two* Ritz values, so a selection must be made [75]. The best choice here would be to select the Ritz value closest to s_{k+1}. For the goal of improving an already rather accurate eigentriplet, however, it is very likely that the approximate eigenvalue is already of the desired accuracy, while the corresponding eigenvector approximations need to be improved. To avoid the selection procedure, the new eigenvalue is computed via the Newton update (cf. step 5 in Alg. 6.1). Essentially, the QRQI presented in Alg. 6.2 is the same as QDPA, but the right-hand sides are updated every iteration.

6.4 Numerical results

In this section, subspace accelerated QDPA is applied to two large-scale examples. The constructed modal equivalents are compared to reduced-order models computed by a second-order Arnoldi method [11], and it is shown how the dominant poles computed by QDPA can be used to improve such reduced-order models. All experiments reported here are executed in Matlab 7.3 on a SUN Ultra 20 (AMD Opteron 2.8GHz, 2GB RAM). During the selection procedure of the most dominant eigentriplet, the approximate eigenvectors are scaled so that (cf. Section 6.3.1)

$$\hat{\mathbf{x}}_i^* \hat{\mathbf{x}}_i + |\hat{\lambda}_i|^2 \hat{\mathbf{x}}_i^* \hat{\mathbf{x}}_i = \hat{\mathbf{y}}_i^* \hat{\mathbf{y}}_i + |\hat{\lambda}_i|^2 \hat{\mathbf{y}}_i^* \hat{\mathbf{y}}_i = 1.$$

6.4.1 The Butterfly gyro

The Butterfly gyro, developed at the Imego Institute with Saab Bofors Dynamics AB, is a vibrating micro-mechanical gyro that is used in inertial navigation

Algorithm 6.2 Quadratic Rayleigh Quotient Iteration (QRQI)

INPUT: System $(M, C, K, \mathbf{v}_0, \mathbf{w}_0)$, initial estimate s_0, tolerance $\epsilon \ll 1$
OUTPUT: Approximate eigenvalue λ and corresponding right and left eigenvectors $\mathbf{x}$ and $\mathbf{y}$

1: Set $k = 0$
2: **while** not converged **do**
3: Solve $\mathbf{v}_{k+1} \in \mathbb{C}^n$ from $(s_k^2 M + s_k C + K)\mathbf{v}_{k+1} = (2s_k M + C)\mathbf{v}_k$
4: Solve $\mathbf{w}_{k+1} \in \mathbb{C}^n$ from $(s_k^2 M + s_k C + K)^* \mathbf{w}_{k+1} = (2s_k M + C)^* \mathbf{w}_k$
5: Compute the new eigenvalue estimate

$$s_{k+1} = s_k - \frac{\mathbf{w}_k^*(2s_k M + C)^* \mathbf{v}_{k+1}}{\mathbf{w}_{k+1}^*(2s_k M + C)\mathbf{v}_{k+1}}$$

6: Set $\mathbf{v}_{k+1} = \mathbf{v}_{k+1}/\|\mathbf{v}_{k+1}\|_2$ and $\mathbf{w}_{k+1} = \mathbf{w}_{k+1}/\|\mathbf{w}_{k+1}\|_2$
7: The eigenvalue $\lambda = s_{k+1}$ with $\mathbf{x} = \mathbf{v}_{k+1}$ and $\mathbf{y} = \mathbf{w}_{k+1}$ has converged if

$$\|(s_{k+1}^2 M + s_{k+1} C + K)\mathbf{v}_{k+1}\|_2 < \epsilon$$

8: Set $k = k + 1$
9: **end while**

applications, for instance for the detection of (unwanted) forces in (unwanted) directions (see [2, 88] for more details). For future improvements of the Butterfly gyro, efficiency and accuracy of gyro simulations is of great importance. Model order reduction not only reduces the simulation times significantly, it also provides a state-space equivalent formulation of the original finite element representation, which is helpful in developing and testing signal processing algorithms for the gyro.

The system matrices of the second-order system $(M, C, K, \mathbf{b}, \mathbf{c})$ can be found in the Oberwolfach benchmark collection [2]. The full system has 17361 degrees of freedom, one input and 12 outputs. For the experiments here $\mathbf{b}$ was the input vector B, $\mathbf{c}$ was taken to be the first column of the 17361×12 selector (output) matrix L, and $C = \beta K$ with $\beta = 10^{-7}$ (equal to the settings in [88]).

The QDPA algorithm with subspace acceleration was used to compute 5, 10, 20, 30, and 35 dominant poles, using both of the deflation strategies described in Section 6.3.2. Since the frequency range of interest is from 10^4 Hz to 10^6 Hz, the initial shift was $s_0 = 2\pi i(1.5 \cdot 10^5)$. The process was restarted with search spaces containing the $k_{\min} = 4$ most dominant approximations at dimension $k_{\max} = 10$. Note that all linear system solves were computed directly, that is using the backslash operator in Matlab, and not using LU-factorizations of $Q(s_k) = s_k^2 M + s_k C + K$, since this appeared to be significantly faster (and requiring less memory). The running times shown in Table 6.1 show that deflation approach II (implicitly solving the linearized system and deflating $\mathbf{b}$ and $\mathbf{c}$, see Section 6.3.2) is more efficient than deflation approach I (explicit orthogonalization against found eigenvectors). The

Table 6.1: Computational times for computing dominant poles and left and right eigenvectors of the Butterfly Gyro ($n = 17361$). Deflation type I refers to explicit orthogonalization against found eigenvectors, type II refers to using implicit linearization and deflated input and output vectors. Every iteration a solve with $Q(s_k) = s_k^2 M + s_k C + K$ and $(Q(s_k))^*$ is needed.

	Deflation I		Deflation II	
#poles	Time (s)	#iterations	Time (s)	#iterations
5	147	28	111	16
10	541	111	228	33
20	1157	235	576	91
30	1947	389	1100	191
35	2560	508	1345	233

frequency response Bode plots, based on the modal equivalents computed using the second deflation strategy, are shown in Figure 6.1. Since qualitatively there is no difference between the two deflation strategies (not shown here), deflation strategy II is the method of choice.

Typically, the 20 most dominant poles, causing the highest peaks in the Bode plot, were found relatively quickly compared to the total running time for computing 35 dominant pole triplets. As more dominant poles are found, only less distinguishing, more or less equally dominant poles remain and QDPA has difficulties in finding these less dominant poles. Comparing the 20th, 30th and 35th order solutions (note that the *real orthogonal* bases for the left and right eigenspaces were equal to the number of poles found for these models, see also Section 6.2), it can also be observed that qualitatively the improvement is smaller. For practical purposes, especially a good match in the $10^4 - 10^5$ Hz frequency range is of importance.

In [88] a 40th order reduced model of the Butterfly gyro was presented. This model was created by computing an orthonormal basis for the Krylov space $\mathcal{K}^{40}(K^{-1}M, K^{-1}\mathbf{b})$ (note that C was neglected and in fact PRIMA [105] was applied, see [128] for more details). If the columns of X form an orthonormal basis for this Krylov space, then the reduced system becomes

$$(X^*MX, X^*CX, X^*KX, X^*\mathbf{b}, X^*\mathbf{c}).$$

In Figure 6.2 the Bode plot of this 40th order model is plotted together with the 30th and 35th order QDPA models. In the eye norm it appears as if the 35th and 40th order model are qualitatively the same, except near 10^6 Hz, where the 35th order QDPA model is more accurate. Although the latter is confirmed by the relative error plots in Figure 6.3, it can also be observed that the 40th order model is more accurate in the lower frequency range. Both the 35th order QDPA and 40th order PRIMA model, however, are of acceptable accuracy. The accuracy of the QDPA model is almost constant over the range 10^4 Hz to 10^6 Hz, while the accuracy of the PRIMA model decreases after 10^5 Hz. This can be explained by the fact that the expansion point for PRIMA was $\sigma = 0$, so that the reduced model is more accurate in the lower frequency range. The QDPA models, on the other hand, are accurate in the neighborhood of frequencies near the imaginary parts of

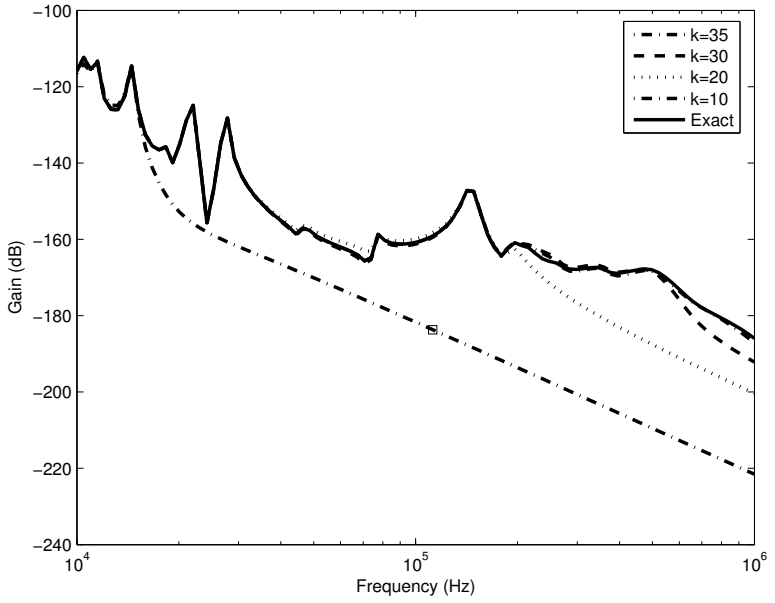


Figure 6.1: Exact transfer function (solid) and 35th (dash-dot), 30th (dash), 20th (dot), and 10th (dash-dot-square) order modal equivalents (based on eigenspaces of 35, 30, 20 and 10 dominant poles, respectively).

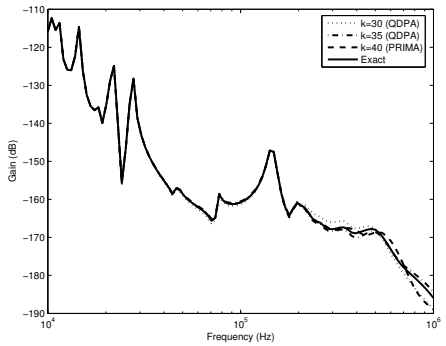


Figure 6.2: Exact response (solid), 40th order PRIMA model (dash), and 35th (dash-dot) and 30th (dot) order modal equivalents.

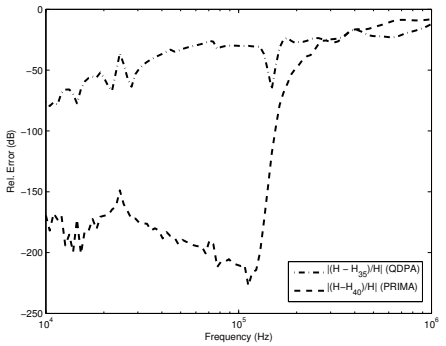


Figure 6.3: Relative error for 40th order PRIMA model (dash) and 35th order QDPA modal equivalent (dash-dot).

the dominant poles.

Although the PRIMA model can be computed in considerably less time (about $20s$), it will not always produce more accurate reduced models. If the imaginary parts of the dominant poles vary largely in magnitude and hence the Bode plot shows peaks over a larger frequency range, then a single expansion point may not be sufficient to produce an acceptable reduced model (rational Krylov methods may be used to handle this, see [58] and the next example), while subspace accelerated QDPA finds the dominant poles automatically.

6.4.2 The breathing sphere

The breathing sphere, taken from [82], is a three-dimensional vibrating body and has its origin in sound radiation analysis (acoustics). A finite element discretization leads to a second-order model of order $n = 17611$, with transfer function

$$H(s) = \mathbf{c}^*(sM^2 + sC + K)\mathbf{b}.$$

The input vector $\mathbf{b}$ contains the normal-velocities of each element, the state $\mathbf{x}$ represents the acoustic pressure on the corresponding elements, and the output vector $\mathbf{c}$ selects state variables. In this experiment all elements of $\mathbf{c}$ are zero, except for $\mathbf{c}_{173} = 1$[2]. The mass matrix M is symmetric.

The QDPA algorithm with subspace acceleration was used to compute 10, 20, 40, and 60 dominant poles, using deflation strategy II described in Section 6.3.2. The initial shift was $s_0 = 2i$. The process was restarted with search spaces containing the $k_{\min} = 4$ most dominant approximations at dimension $k_{\max} = 10$. All linear system solves were computed directly using the backslash operator in Matlab. The running times are shown in Table 6.2. The frequency response

[2]Communicated by J. Lampe [82].

Bode plots of the modal equivalents of order $k = 2 \times 40$ and $k = 2 \times 60$ are shown in Figure 6.4, together with a 40th order rational second-order Krylov model (Rational Krylov via Arnoldi (RKA)). The latter model was constructed as $(W^*MV, W^*CV, W^*KV, W^*\mathbf{b}, V^*\mathbf{c})$, where the columns of V and W are orthonormal bases for the second-order rational Krylov subspaces

$$\sum_{j=1}^{4} \mathcal{G}^{10}(-\widetilde{K}_j^{-1}\widetilde{C}_j, -\widetilde{K}_j^{-1}M, \widetilde{K}_j^{-1}\mathbf{b}),$$

and

$$\sum_{j=1}^{4} \mathcal{G}^{10}(-\widetilde{K}_j^{-*}\widetilde{C}_j^{*}, -\widetilde{K}_j^{-*}M^{*}, \widetilde{K}_j^{-*}\mathbf{c}),$$

respectively, with $\widetilde{K}_j = \sigma_j^2 M + \sigma_j C + K$, $\widetilde{C}_j = 2\sigma_j M + C$, and interpolation points $\sigma_1 = 0.1$, $\sigma_2 = 0.5$, $\sigma_3 = 1$, $\sigma_4 = 5$. A second-order Krylov subspace is defined as

$$\mathcal{G}^{k+1}(A, B, \mathbf{v}_0) = \text{span}(\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_k),$$

with

$$\begin{aligned} \mathbf{v}_1 &= A\mathbf{v}_1 \\ \mathbf{v}_i &= A\mathbf{v}_{i-1} + B\mathbf{v}_{i-2}, \quad (i = 2, \ldots, k). \end{aligned}$$

See [11] for the complete second-order Arnoldi algorithm (SOAR) and more details.

The 40th order SOAR model captures the response for low frequencies, but fails to match the peaks around 0.6 rad/s and higher frequencies. The QDPA modal equivalents, on the other hand, are more accurate in the details. The CPU time to produce the SOAR model was only 506 seconds, but it is difficult to choose the interpolation points in such a way that the model captures more detail. QDPA computes the dominant poles automatically and consequently, the modal equivalents capture the details (the peaks) in the frequency response.

These observations lead to the idea of filling in the missing details of the rational SOAR model by expanding the Krylov bases V and W with the right and left eigenvectors X and Y of dominant poles computed by QDPA: $\widetilde{V} = [V, X]$ and $\widetilde{W} = [W, Y]$, respectively. The columns of $\widetilde{V}$ and $\widetilde{W}$ are kept orthogonal, and the new reduced-order model is constructed as $(\widetilde{W}^*M\widetilde{V}, \widetilde{W}^*C\widetilde{V}, \widetilde{W}^*K\widetilde{V}, \widetilde{W}^*\mathbf{b}, \widetilde{V}^*\mathbf{c})$. Note that the number of matched moments (thanks to SOAR [11]) is the same for this hybrid model, and also the dominant poles computed by QDPA are poles of this model. Figure 6.5 shows the frequency responses of a 10th order modal equivalent computed by QDPA with $s_0 = 0.6i$ (25 iterations, 691 seconds), the 40th order SOAR RKA model (506 seconds) and the hybrid model, and the exact response. The 10th order modal equivalent captures the peaks near $\omega = 0.6$ rad/s that the SOAR model misses. The hybrid model combines the best of the two models, as can also be observed in the error plot in Fig. 6.6: the relative error near $\omega = 0.6$ rad/s drops from $O(1)$ to $O(10^{-2})$. Adding more dominant poles leads to improvements in other frequency ranges as well.

Table 6.2: Computational times for computing dominant poles and left and right eigenvectors of the breathing sphere ($n = 17611$). Every iteration a solve with $Q(s_k) = s_k^2 M + s_k C + K$ and $(Q(s_k))^*$ is needed.

#poles	Time (s)	#iterations
10	2800	108
20	4280	156
40	13901	546
60	29880	1223

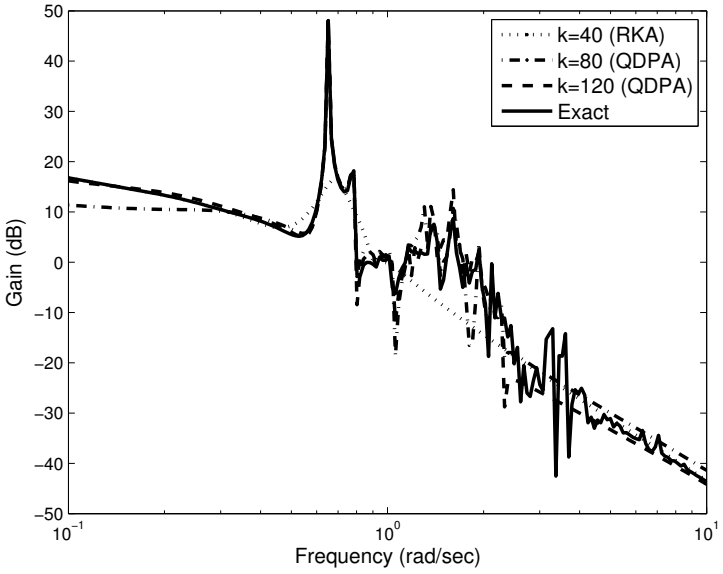


Figure 6.4: Exact transfer function (solid), 40th order SOAR RKA model (dot), and 80th (dash-dot) and 120th (dash) order modal equivalents.

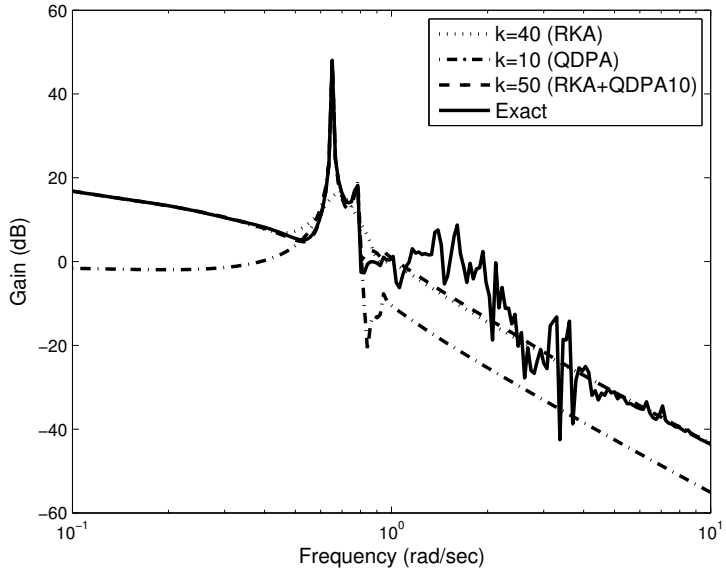


Figure 6.5: Exact transfer function (solid), 40th order SOAR RKA model (dot), 10th (dash-dot) order modal equivalent, and 50th order hybrid RKA+QDPA (dash).

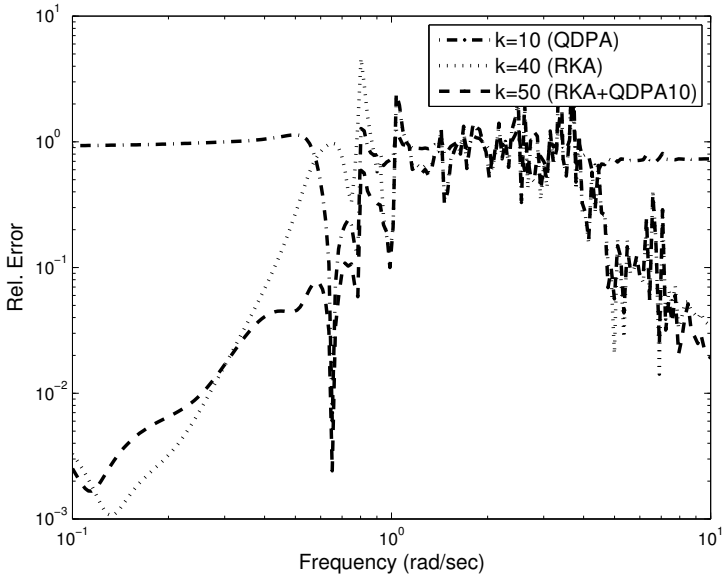


Figure 6.6: Relative errors for 10th order modal equivalent (dash-dot), 40th order SOAR RKA model (dot), and 50th order hybrid RKA+QDPA (dash).

The other way around, the imaginary parts of the dominant poles computed by QDPA can also be used as interpolation points for the rational SOAR models, to get better detail in the corresponding frequency range. Given a dominant pole $\lambda = \alpha + \beta i$, one can use either $\sigma = \beta$ (real) or $\sigma = \beta i$ as interpolation point. According to [70, Chapter 6] real shifts have a more global effect, while purely imaginary shifts focus more on detail, and hence this is advised when trying to capture the peaks. Figure 6.7 shows the frequency response of a 70th order SOAR model that was computed using interpolation points $\sigma_1 = 0.65i$, $\sigma_2 = 0.78i$, $\sigma_3 = 0.93i$, and $\sigma_4 = 0.1$. The imaginary shifts σ_1, σ_2, and σ_3 correspond to the imaginary parts of the dominant poles that cause peaks between $\omega = 0.6$ rad/s and $\omega = 1$ rad/s. These poles were selected from 5 poles computed by QDPA with $s_0 = 0.6i$ (691 seconds, the same run as in the previous paragraph). For each interpolation point a 10-dimensional dual second-order Krylov subspace was computed using SOAR (1373 seconds), and because real bases were used in the projection, the dimension of the reduced-order model is $k = 3 \times 20 + 10 = 70$. This reduced model is more accurate than the previous reduced-order models, up to $\omega = 1$ rad/s. Although the dominant poles near the imaginary shifts are present in this reduced-order model, this is in general not guaranteed. Nevertheless, this approach appears to be more robust than adding dominant poles (and corresponding states) to the reduced-order model like in the previous paragraph, albeit computationally more expensive since complex interpolation points are needed. Combined with strategies that determine the dimensions of the rational Krylov spaces dynamically, as described in [70, Chapter 6] and [58], more efficient schemes can be designed.

6.5 Higher order systems, zeros, and MIMO systems

6.5.1 Higher order polynomial systems

The algorithms and techniques developed in the previous sections can be applied to higher order polynomial transfer functions. Consider the p-th order polynomial transfer function

$$H(s) = \mathbf{c}^*(s^p A_p + s^{p-1}A_{p-1} + \cdots + sA_1 + A_0)^{-1}\mathbf{b},$$

where $A_i \in \mathbb{R}^{n\times n}$ and $\mathbf{b}, \mathbf{c} \in \mathbb{R}^n$. Then again Newton's method can be applied to compute the poles of $H(s)$ via the zeros of $1/H(s)$, leading to

$$s_{k+1} = s_k - \frac{\mathbf{c}^*(s_k^p A_p + s_k^{p-1}A_{p-1} + \cdots + s_k A_1 + A_0)^{-1}\mathbf{b}}{\mathbf{w}^*(ps_k^{p-1}A_p + (p-1)s_k^{p-2}A_{p-1} + \cdots + s_k A_2 + A_1)\mathbf{v}},$$

where $\mathbf{v} = (s_k^p A_p + s_k^{p-1}A_{p-1} + \cdots + s_k A_1 + A_0)^{-1}\mathbf{b}$ and $\mathbf{w} = (s_k^p A_p + s_k^{p-1}A_{p-1} + \cdots + s_k A_1 + A_0)^{-*}\mathbf{c}$. Since the poles are eigenvalues of the polynomial eigenproblem (PEP)

$$\begin{array}{rcll} (\lambda_i^p A_p + \lambda_i^{p-1}A_{p-1} + \cdots + \lambda_i A_1 + A_0)\mathbf{x}_i & = & 0, & \mathbf{x}_i \neq 0, \\ \mathbf{y}_i^*(\lambda_i^p A_p + \lambda_i^{p-1}A_{p-1} + \cdots + \lambda_i A_1 + A_0) & = & 0, & \mathbf{y}_i \neq 0, \end{array} \tag{6.5.1}$$

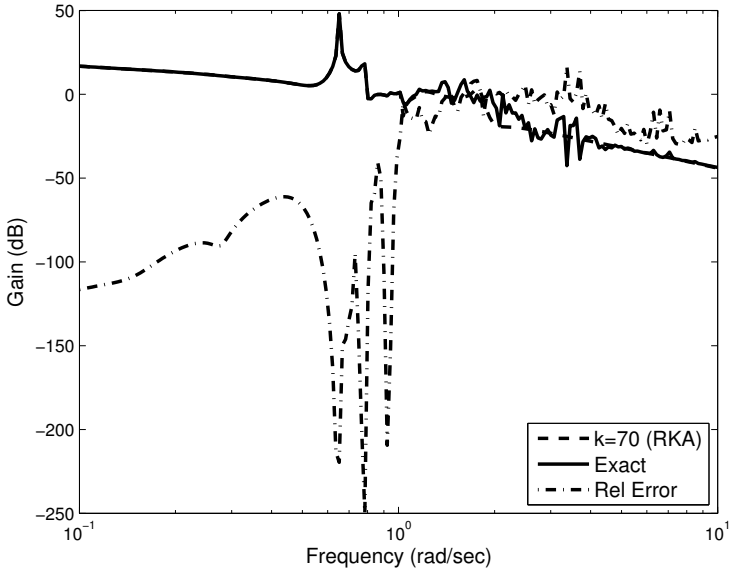


Figure 6.7: Exact transfer function (solid), 70th order SOAR RKA model (dash) using interpolation points based on dominant poles, and relative error (dash-dot).

for $i = 1, \ldots, np$, a stopping criterion is

$$||(s_{k+1}^p A_p + s_{k+1}^{p-1} A_{p-1} + \cdots + s_{k+1} A_1 + A_0)\mathbf{v}||_2 < \epsilon, \quad \epsilon \ll 1.$$

The vectors $\mathbf{x} = \mathbf{v}$ and $\mathbf{y} = \mathbf{w}$ are the corresponding approximate right and left eigenvectors of the PEP.

A corresponding linearized system can be of the form

$$\begin{cases} B\dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + \mathbf{b}_l u(t) \\ y(t) &= \mathbf{c}_l^* \mathbf{x}(t) + du(t), \end{cases}$$

where

$$A = \begin{bmatrix} 0 & N_0 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & N_1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{p-2} \\ -A_0 & -A_1 & \cdots & \cdots & -A_{p-2} & -A_{p-1} \end{bmatrix},$$

$B = \mathrm{diag}(N_0, N_1 \ldots, N_{p-2}, A_p)$, the N_i $(i = 0, \ldots, p-2)$ are nonsingular $n \times n$ matrices , $\mathbf{b}_l = [0, \ldots, 0, \mathbf{b}^T]^T \in \mathbb{R}^{np \times 1}$ and $\mathbf{c}_l = [\mathbf{c}^T, 0, \ldots, 0]^T \in \mathbb{R}^{np \times 1}$. Note that with the choice $N_0 = A_0$ (if A_0 is nonsingular), one obtains for $p = 2$ the QEP as described in Section 6.2. Another possible choice is $N_i = I$ for all $i = 0, \ldots, p-2$.

The right and left eigenvectors of the linearized eigenproblem

$$A\phi_i = \lambda_i B\phi_i, \quad \psi_i^* A = \lambda_i \psi_i^* B, \quad \phi_i, \psi_i \neq 0,$$

can be verified to be

$$\phi_i = \begin{bmatrix} \mathbf{x}_i \\ \lambda_i \mathbf{x}_i \\ \lambda_i^2 \mathbf{x}_i \\ \vdots \\ \lambda_i^{p-2} \mathbf{x}_i \\ \lambda_i^{p-1} \mathbf{x}_i \end{bmatrix}, \text{ and } \psi_i = \begin{bmatrix} -N_0^{-*}(\lambda_i^{p-2} A_0)^* \mathbf{y}_i \\ -N_1^{-*}(\lambda_i^{p-3} A_0 + \lambda_i^{p-2} A_1)^* \mathbf{y}_i \\ \vdots \\ \vdots \\ -N_{p-2}^{-*}(A_0 + \lambda_i A_1 + \ldots + \lambda_i^{p-2} A_{p-2})^* \mathbf{y}_i \\ \bar{\lambda}_i^{p-1} \mathbf{y}_i \end{bmatrix}.$$

Denoting the matrix polynomial of the PEP (6.5.1) by $Q(\lambda) = s^p A_p + s^{p-1} A_{p-1} + \cdots + sA_1 + A_0$, it can be verified that

$$\begin{bmatrix} Q(\lambda) & 0 \\ 0 & I \end{bmatrix} = E(\lambda)(A - \lambda B)F(\lambda),$$

where

$$E(\lambda) = \begin{bmatrix} E_{11} & E_{12} & \cdots & \cdots & E_{1,(p-1)} & I \\ -N_0^{-1} & 0 & 0 & \cdots & \cdots & 0 \\ 0 & -N_1^{-1} & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -N_{p-3}^{-1} & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & -N_{p-2}^{-1} & 0 \end{bmatrix},$$

and

$$F(\lambda) = \begin{bmatrix} I & 0 & \cdots & \cdots & \cdots & 0 \\ \lambda I & I & 0 & \cdots & \cdots & 0 \\ \lambda^2 I & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \lambda^{p-1} & \cdots & \cdots & \lambda^2 I & \lambda I & I \end{bmatrix},$$

whit

$$\begin{aligned} E_{11} &= (A_1 + \lambda A_2 + \lambda^2 A_3 + \cdots + \lambda^{p-1} A_p) N_0^{-1}, \\ E_{12} &= (A_2 + \lambda A_3 + \cdots + \lambda^{p-2} A_p) N_0^{-1}, \\ \vdots \quad &\vdots \qquad \vdots \\ E_{1,(p-1)} &= (A_p + \lambda A_{p-1}) N_{p-2}^{-1}. \end{aligned}$$

Indeed, the linear matrix $A - \lambda B$ is a linearization of $Q(\lambda)$.

Using a similar approach to that in Section 6.2, it follows that

$$Q(s)^{-1} = X(sI - \Lambda)^{-1}\Lambda^{p-1}Y^*.$$

The partial fraction representation becomes

$$H(s) = \mathbf{c}^*X(sI - \Lambda)^{-1}\Lambda^{p-1}Y^*\mathbf{b} = \sum_{i=1}^{pn} \frac{R_i}{s - \lambda_i},$$

where the residues are given by (cf. (6.2.8))

$$R_i = (\mathbf{c}^*\mathbf{x}_i)(\mathbf{y}_i^*\mathbf{b})\lambda_i^{p-1}.$$

Note that the $\mathbf{x}_i$ and $\mathbf{y}_i$ are scaled so that

$$\psi_i^* B \phi_i = 1.$$

The techniques for subspace acceleration, selection and deflation, and local convergence improvement, described in Section 6.3, can be generalized to polynomial transfer functions. The modal equivalents and reduced-order models preserve the structure of the original systems.

6.5.2 Transfer function zeros

In [125] it is shown how the (SA)DPA algorithms can be used for the computation of dominant zeros via the dominant poles of the inverse transfer function. In the single input single output (SISO) case, $z_0 \in \mathbb{C}$ is called a transmission zero [135] if $H(z_0) = 0$, where

$$H(s) = \mathbf{c}^*(s^pA_p + s^{p-1}A_{p-1} + \cdots + sA_1 + A_0)^{-1}\mathbf{b} + d.$$

The following two theorems are straightforward generalizations of Theorem 3.1 and Theorem 3.2 in [125] (Chapter 5) and show how inverse systems $\Sigma^z = (A_p^z, A_{p-1}^z, \ldots, A_0^z, \mathbf{b}^z, \mathbf{c}^z, d^z)$ can be defined so that $H^z(s)$ is the inverse of $H(s)$, for $d = 0$ and $d \neq 0$, respectively.

Theorem 6.5.1. *Let*

$$y(s) = H(s)u(s) = \left[\mathbf{c}^*(s^pA_p + s^{p-1}A_{p-1} + \cdots + sA_1 + A_0)^{-1}\mathbf{b}\right] u(s)$$

be the transfer function of the descriptor realization

$$\Sigma = (A_p, A_{p-1}, \ldots, A_0, \mathbf{b}, \mathbf{c}, d = 0),$$

and let

$$y^z(s) = H^z(s)u^z(s) = \left[\mathbf{c}^{z*}(s^pA_p^z + s^{p-1}A_{p-1}^z + \cdots + sA_1^z + A_0^z)^{-1}\mathbf{b}^z\right] u^z(s)$$

be the transfer function of the augmented descriptor realization

$$\Sigma^z = (A_p^z, A_{p-1}^z, \ldots, A_0^z, \mathbf{b}^z, \mathbf{c}^z, d^z = 0),$$

where

$$A_0^z = \begin{bmatrix} A_0 & \mathbf{b} \\ -\mathbf{c}^* & 0 \end{bmatrix}, \quad A_i^z = \begin{bmatrix} A_i & 0 \\ 0 & 0 \end{bmatrix} \quad (i = 1, \dots p),$$
$$\mathbf{b}^z = \begin{bmatrix} \mathbf{b} \\ 1 \end{bmatrix}, \qquad \mathbf{c}^z = \begin{bmatrix} \mathbf{c} \\ 1 \end{bmatrix}, \qquad d^z = 0.$$

Then $H^z(s) = H^{-1}(s)$.

Proof. This proof is a generalization of the proof of [125, Thm. 3.1]. Let $u \equiv u(s)$, $y \equiv y(s) = \mathbf{c}^*\mathbf{x}(s)$, $\mathbf{x} \equiv \mathbf{x}(s) = (s^pA_p + s^{p-1}A_{p-1} + \cdots + sA_1 + A_0)^{-1}\mathbf{b}u(s)$, $u^z \equiv u^z(s)$, $y^z \equiv y^z(s) = \mathbf{c}^{z*}\mathbf{x}^z(s)$, $\mathbf{x}^z \equiv \mathbf{x}^z(s) = (s^pA_p^z + s^{p-1}A_{p-1}^z + \cdots + sA_1^z + A_0^z)^{-1}\mathbf{b}^z u^z(s)$, and the Rosenbrock system matrix [127] equation for Σ

$$\begin{bmatrix} s^pA_p + s^{p-1}A_{p-1} + \cdots + sA_1 + A_0 & -\mathbf{b} \\ \mathbf{c}^* & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ u \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}.$$

By adding $(\mathbf{b}y - \mathbf{b}y)$ to the left side of the first equation and defining the error $(u - y)$ as a new algebraic variable, one obtains

$$\left[\begin{array}{cc:c} s^pA_p + s^{p-1}A_{p-1} + \cdots + sA_1 + A_0 & -\mathbf{b} & -\mathbf{b} \\ \mathbf{c}^* & 0 & -1 \\ \hdashline \mathbf{c}^* & 1 & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{x} \\ u - y \\ \hdashline y \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ \hdashline u \end{array}\right]$$

or

$$\begin{bmatrix} s^pA_p^z + s^{p-1}A_{p-1}^z + \cdots + sA_1^z + A_0^z & -\mathbf{b}^z \\ \mathbf{c}^{z*} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}^z \\ u^z \end{bmatrix} = \begin{bmatrix} 0 \\ y^z \end{bmatrix},$$

which is the Rosenbrock system matrix [135] equation for Σ^z. Hence $u^z(s) = y(s)$, $y^z(s) = u(s)$ and $H^z(s) = H^{-1}(s)$. □

A similar result can be obtained for the case $d \neq 0$, see for example for the first order case [78, Ex. 2.2-20] and [79, 137].

Theorem 6.5.2. *Let*

$$y(s) = H(s)u(s) = \left[\mathbf{c}^*(s^pA_p + s^{p-1}A_{p-1} + \cdots + sA_1 + A_0)^{-1}\mathbf{b} + d\right] u(s)$$

be the transfer function of the descriptor realization

$$\Sigma = (A_p, A_{p-1}, \dots, A_0, \mathbf{b}, \mathbf{c}, d \neq 0),$$

and let

$$y^z(s) = H^z(s)u^z(s) = \left[\mathbf{c}^{z*}(s^pA_p^z + s^{p-1}A_{p-1}^z + \cdots + sA_1^z + A_0^z)^{-1}\mathbf{b}^z + d^z\right] u^z(s)$$

be the transfer function of the descriptor realization

$$\Sigma_z = (A_p^z, A_{p-1}^z, \dots, A_0^z, \mathbf{b}^z, \mathbf{c}^z, d^z \neq 0),$$

where

$$A_0^z = A_0 - d^{-1}\mathbf{b}\mathbf{c}^*, \qquad A_i^z = A_i \qquad (i = 1, \dots, p),$$
$$\mathbf{b}^z = d^{-1}\mathbf{b}, \quad \mathbf{c}^z = -d^{-1}\mathbf{c}, \quad d^z = d^{-1}.$$

Then $H^z(s) = H^{-1}(s)$.

Proof. This proof is a generalization of the proof of [125, Thm. 3.1]. Let $u \equiv u(s)$, $y \equiv y(s) = \mathbf{c}^*\mathbf{x}(s) + du(s)$, $\mathbf{x} \equiv \mathbf{x}(s) = (s^pA_p + s^{p-1}A_{p-1} + \cdots + sA_1 + A_0)^{-1}\mathbf{b}u(s)$, $u^z \equiv u^z(s)$, $y^z \equiv y^z(s) = \mathbf{c}^{z*}\mathbf{x}^z(s) + d^zu^z(s)$, $\mathbf{x}^z \equiv \mathbf{x}^z(s) = (s^pA_p^z + s^{p-1}A_{p-1}^z + \cdots + sA_1^z + A_0^z)^{-1}\mathbf{b}_z u_z(s)$, and the Rosenbrock system matrix equation for Σ

$$\begin{bmatrix} s^pA_p + s^{p-1}A_{p-1} + \cdots + sA_1 + A_0 & -\mathbf{b} \\ \mathbf{c}^* & d \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ u \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}. \tag{6.5.2}$$

The expression $u = -d^{-1}\mathbf{c}^*\mathbf{x} + d^{-1}y$ is obtained from the second equation of (6.5.2), which is substituted into the first equation of (6.5.2) to yield

$$\begin{bmatrix} s^pA_p + s^{p-1}A_{p-1} + \cdots + sA_1 + (A_0 - d^{-1}\mathbf{b}\mathbf{c}^*) & -d^{-1}\mathbf{b} \\ -d^{-1}\mathbf{c}^* & d^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_z \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ u \end{bmatrix},$$

or

$$\begin{bmatrix} s^pA_p^z + s^{p-1}A_{p-1}^z + \cdots + sA_1^z + A_0^z & -\mathbf{b}_z \\ \mathbf{c}^{z*} & d^z \end{bmatrix} \begin{bmatrix} \mathbf{x}^z \\ u^z \end{bmatrix} = \begin{bmatrix} 0 \\ y^z \end{bmatrix},$$

which is the Rosenbrock system matrix equation for Σ^z. Hence $u^z(s) = y(s)$, $y^z(s) = u(s)$ and $H^z(s) = H^{-1}(s)$. □

Consequently, the QDPA algorithm with subspace acceleration can be used to compute the dominant zeros of higher order polynomial transfer functions via the dominant poles of the inverse transfer function. In practice, a single implementation of the QDPA algorithm can be used to compute both dominant poles and dominant zeros. For more details on the computation of dominant zeros, see [125].

6.5.3 MIMO transfer functions

For a multi-input multi-output (MIMO) system

$$\begin{cases} A_p\mathbf{x}^{(p)}(t) + A_{p-1}\mathbf{x}^{(p-1)}(t) + \cdots + A_1\dot{\mathbf{x}}(t) + A_0\mathbf{x}(t) &= B\mathbf{u}(t) \\ \mathbf{y}(t) &= C^*\mathbf{x}(t) + D\mathbf{u}(t), \end{cases}$$

where $A_i \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{n\times q}$, $\mathbf{x}(t) \in \mathbb{R}^n$, $\mathbf{u}(t) \in \mathbb{R}^m$, $\mathbf{y}(t) \in \mathbb{R}^q$ and $D \in \mathbb{R}^{q\times m}$, the transfer function $H(s) : \mathbb{C} \longrightarrow \mathbb{C}^{q\times m}$ is defined as

$$H(s) = C^*(s^pA_p + s^{p-1}A_{p-1} + \cdots + sA_1 + A_0)^{-1}B + D. \tag{6.5.3}$$

The dominant poles of (6.5.3) are those $s \in \mathbb{C}$ for which $\sigma_{\max}(H(s)) \to \infty$. For square transfer functions ($m = q$), there is an equivalent criterion: the dominant poles are those $s \in \mathbb{C}$ for which $\lambda_{\min}(H^{-1}(s)) \to 0$. This leads, for square transfer functions, to the following Newton scheme (cf. (6.2.13)):

$$s_{k+1} = s_k - \frac{1}{\mu_{\min}} \frac{1}{\mathbf{v}^*C^*(Q(s_k))^{-1}Q'(s_k)(Q(s_k))^{-1}B\mathbf{u}},$$

where $(\mu_{\min}, \mathbf{u}, \mathbf{v})$ is the eigentriplet of $H^{-1}(s_k)$ corresponding to $\lambda_{\min}(H^{-1}(s_k))$, and

$$Q(\lambda) = \lambda^pA_p + \lambda^{p-1}A_{p-1} + \cdots + \lambda A_1 + A_0.$$

An algorithm for computing the dominant poles of a MIMO transfer function can readily be derived from Alg. 6.1. The reader is referred to [93] for the initial MIMO DPA algorithm and to [122] for an algorithm similar to SADPA, generalizations to non-square MIMO systems and more details.

6.6 Conclusions

The Quadratic Dominant Pole Algorithm (QDPA) presented in this chapter is an efficient and effective method for the computation of dominant poles of transfer functions of large-scale second-order dynamical systems. Subspace acceleration improves the global convergence, while the inexpensive deflation strategy makes QDPA able to compute more than one dominant pole automatically, without the risk of computing already found poles again. Another advantage of QDPA is that no linearization of the system is needed, since QDPA operates on the original system matrices.

The dominant poles and corresponding left and right eigenvectors can be used to construct structure-preserving modal equivalents. The dominant eigenspaces can be combined with (second-order) Krylov subspaces (SOAR) to produce reduced-order models of better quality than computed by both methods independently. Furthermore, interpolation points for rational second-order Krylov methods can be based on the (imaginary part) of the dominant poles. Numerical experiments confirmed that accurate reduced-order models can be computed this way.

QDPA can be generalized to MIMO systems and higher-order systems, and can be used for the computation of dominant zeros as well.

Addendum

This chapter is also available as [124]

> Joost Rommes and Nelson Martins, *Efficient computation of transfer function dominant poles of large second-order dynamical systems*, Preprint 1360, Utrecht University, 2007,

and has been submitted for publication.

Chapter 7

Arnoldi and Jacobi-Davidson methods for generalized eigenvalue problems $A\mathbf{x} = \lambda B\mathbf{x}$ with singular B

Abstract. In many physical situations, a few specific eigenvalues of a large sparse generalized eigenvalue problem $A\mathbf{x} = \lambda B\mathbf{x}$ are needed. If exact linear solves with $A - \sigma B$ are available, implicitly restarted Arnoldi with purification is a common approach for problems where B is positive semidefinite. In this chapter, a new approach based on implicitly restarted Arnoldi will be presented that avoids most of the problems due to the singularity of B. Secondly, if exact solves are not available, Jacobi-Davidson QZ will be presented as a robust method to compute a few specific eigenvalues. Results are illustrated by numerical experiments.

Key words. sparse generalized eigenvalue problems, purification, semi-inner product, implicitly restarted Arnoldi, Jacobi-Davidson, preconditioning

7.1 Introduction

Large sparse generalized eigenvalue problems of the form

$$A\mathbf{x} = \lambda B\mathbf{x}, \qquad \mathbf{x} \neq 0, \tag{7.1.1}$$

with $A, B \in \mathbb{R}^{n\times n}$, $\mathbf{x} \in \mathbb{C}^n$ and $\lambda \in \mathbb{C}$, arise in physical situations like stability analysis of the discretized Navier-Stokes equations. Typically, the matrix A is nonsymmetric and of full rank, and B is singular. The pencil (A, B) is regular, that is, $A-\gamma B$ is singular only for a finite number of $\gamma \in \mathbb{C}$. Because B is singular, (7.1.1) can have eigenvalues at infinity, which are of no physical relevance, but may lead to numerical difficulties. In practice, one is often interested in the few left- or

rightmost *finite* eigenvalues that determine the stability, and hence one wants to avoid approximations to eigenvalues at infinity. This chapter is concerned with the computation of a few left- or rightmost eigenvalues of large generalized eigenvalue problems.

One way to compute a few eigenvalues of (7.1.1) close to $\sigma \in \mathbb{C}$ is to apply Arnoldi's method to the shift-and-invert transformation $S = (A - \sigma B)^{-1}B$:

$$S\mathbf{x} = \tilde{\lambda}\mathbf{x}, \qquad \mathbf{x} \neq 0. \tag{7.1.2}$$

An eigenpair $(\lambda, \mathbf{x})$ of (7.1.1) corresponds to an eigenpair $(\tilde{\lambda} = (\lambda - \sigma)^{-1}, \mathbf{x})$ of (7.1.2). Hence, the infinite eigenvalues of (7.1.1) correspond to eigenvalues $\tilde{\lambda} = 0$ of (7.1.2). Arnoldi's method may compute approximations $\tilde{\theta}$ to $\tilde{\lambda} = 0$. These approximations are known as *spurious* eigenvalues and after back transformation via $\theta = \tilde{\theta}^{-1} + \sigma$, they may be hard to distinguish from wanted eigenvalues, which typically reside in the exterior of the spectrum. This problem has been addressed for the symmetric nondefective problem [46, 104] and for the defective problem [46]. The ideas presented there are extended to the nonsymmetric defective case in [97], where the implicitly restarted Arnoldi method [85, 146] is implemented with a B semi-inner product and purification. Purification is a technique to remove unwanted components from Arnoldi vectors and approximate eigenvectors, and will be explained in more detail in Section 7.3. A new strategy will be presented that, by exploiting the structure of (7.1.1), reduces the corruption by unwanted components significantly.

The scheme based on the Arnoldi method fails to be applicable if the linear system solves with $A - \sigma B$, e.g. via the LU-factorization of $A - \sigma B$, are inaccurate or not computable within reasonable time. The Jacobi-Davidson QZ method [51] has the advantage that it computes with the matrices A and B directly and that in principle no inverses or exact solves are needed; furthermore, it poses no restrictions on the matrices A and B, so it is also applicable if B is not symmetric positive semidefinite or if both A and B are singular. In Section 7.4, it is shown that the Jacobi-Davidson method with harmonic Petrov values has some favorable properties with respect to purification. If, additionally, a preconditioner is available in the form of an LU-factorization, the correction equation can be solved efficiently and purification is obtained automatically.

Throughout this chapter, it will be assumed that the leftmost finite eigenvalues are wanted. This is a natural assumption in practical situations where the stability of steady states for a number of different parameter values is to be determined, see for instance [33, 69]: not only the leftmost eigenvalue is of interest, but also the eigenvalue(s) close to the leftmost that may become the leftmost for different parameter values. The theory extends readily to problems where the rightmost finite eigenvalues are wanted.

The outline of the chapter is as follows. Some properties of generalized eigenvalue problems are described in Section 7.2. In Section 7.3, the Arnoldi method with purification is explained and the new scheme is presented, illustrated by numerical examples. The approach based on the JDQZ method is described in Section 7.4. Section 7.5 concludes.

7.2 Some properties of generalized eigenvalue problems

Central point of the discussion is the generalized eigenproblem

$$A\mathbf{x} = \lambda B\mathbf{x}, \qquad \mathbf{x} \neq 0,$$

with $A, B \in \mathbb{R}^{n\times n}$, $\mathbf{x} \in \mathbb{C}^n$ and $\lambda \in \mathbb{C}$. Only regular matrix pencils will be considered, i.e. pencils (A, B) for which $A - \gamma B$ is singular only for a finite number of $\gamma \in \mathbb{C}$. Note that B is allowed to be singular. The corresponding ordinary eigenproblem is

$$S\mathbf{x} = \tilde{\lambda}\mathbf{x}, \qquad \mathbf{x} \neq 0,$$

with $S = (A - \sigma B)^{-1}B$ for a σ such that $A - \sigma B$ is non singular. A generalized eigenpair $(\lambda, \mathbf{x})$ corresponds to an ordinary eigenpair $(\tilde{\lambda} = (\lambda - \sigma)^{-1}, \mathbf{x})$ of (7.1.2). The generalized eigenvalues can be computed via the relation $\lambda = \tilde{\lambda}^{-1} + \sigma$.

The eigenspace corresponding to the infinite eigenvalues is the null space $\mathcal{N}(S)$ of S:

$$V_\infty = \mathcal{N}(S) = \mathcal{N}(B) = \{\mathbf{x} \in \mathbb{R}^n \mid B\mathbf{x} = 0\}.$$

The eigenvectors corresponding to the finite eigenvalues span a real invariant subspace of S and form a subspace of the range of S^{j_s}, $\mathcal{R}(S^{j_s})$:

$$V_{finite} \subseteq \mathcal{R}(S^{j_s}) = \{\mathbf{x} \in \mathbb{R}^n \mid ((A - \sigma B)^{-1}B)^{j_s}\mathbf{y} = \mathbf{x}, \mathbf{y} \in \mathbb{R}^n\}, \tag{7.2.1}$$

where j_s is the size of the largest Jordan block corresponding to the zero eigenvalue of S. The generalized null space $\mathcal{G}(S)$ of S is defined as the complement in $\mathcal{N}(S^{j_s})$ of $\mathcal{N}(S)$

$$\mathcal{G}(S) = \mathcal{N}(S^{j_s})\backslash\mathcal{N}(S),$$

It follows that $\mathbb{R}^n = \mathcal{R}(S^{j_s}) + \mathcal{G}(S) + \mathcal{N}(S)$. Note that (7.2.1) becomes an equality if for all finite eigenvalues the algebraic multiplicity is equal to the geometric multiplicity.

It is important to keep in mind that eigenvectors $\mathbf{x}$ corresponding to finite eigenvalues do not necessarily satisfy $\mathbf{x} \perp V_\infty$. In other words, in general it does not hold that $\mathcal{R}(S) \perp \mathcal{N}(S)$. So restricting the search space to $\mathcal{R}(B)$, to avoid approximations to infinite eigenvalues, is not effective. Only if A is block upper triangular and B is block diagonal it is effective, but then the problem can also easily be reduced to a smaller problem by considering the non-zero diagonal blocks.

This chapter is concerned with block structured generalized eigenvalue problems of the form

$$\begin{bmatrix} K & C \\ C^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \lambda \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix}, \tag{7.2.2}$$

with $n = m + k$, $C \in \mathbb{R}^{m\times k}$ of full rank, stiffness matrix $K \in \mathbb{R}^{m\times m}$, mass matrix $M = M^T \in \mathbb{R}^{m\times m}$, velocity $\mathbf{u} \in \mathbb{C}^m$ and pressure $\mathbf{p} \in \mathbb{C}^k$, that arise in the

linearized stability analysis of steady state solutions of the Navier-Stokes equations [33]. The corresponding ordinary eigenproblem is

$$\begin{bmatrix} S_1 & 0 \\ S_2 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \tilde{\lambda} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix}, \qquad S_1 \in \mathbb{R}^{m\times m}, S_2 \in \mathbb{R}^{k\times m},$$

and, as is also noted in [97], this leads to the reduced problem

$$S_1\mathbf{u} = \tilde{\lambda}\mathbf{u}, \qquad S_1 \in \mathbb{R}^{m\times m}. \tag{7.2.3}$$

If $(\tilde{\lambda}, \mathbf{u})$ is an exact eigenpair of S_1, then for nonzero $\tilde{\lambda}$, $(\tilde{\lambda}, [\mathbf{u}^*, \mathbf{p}^*]^*)$ with $\mathbf{p} = \tilde{\lambda}^{-1} S_2\mathbf{u}$ is an exact eigenpair of S. It can be shown [97, Section 2.3] that $\dim(\mathcal{N}(S_1)) = k$, $\dim(\mathcal{R}(S_1)) = m - k$ and

$$\mathbf{u} \in \mathcal{N}(S_1) \quad \Leftrightarrow \quad \begin{bmatrix} \mathbf{u} \\ 0 \end{bmatrix} \in \mathcal{G}.$$

Hence, by reducing the problem to (7.2.3), the geometric multiplicity of the k eigenvalues $\tilde{\lambda} = 0$ is reduced from 2 to 1.

7.3 Arnoldi methods with purification

In Section 7.3.1, the implicitly restarted B-orthogonal Arnoldi method will be described. In Section 7.3.2, it will be shown how this method can be improved by exploiting the specific structure of the generalized eigenproblem. A new strategy, based on this known but previously not used fact, for the computation of a few leftmost eigenvalues will be presented in Section 7.3.3, followed by numerical examples in Section 7.3.4.

7.3.1 Implicitly restarted B-orthogonal Arnoldi methods

B-orthogonal Arnoldi

The B-orthogonal Arnoldi method is the standard Arnoldi method with the usual inner product $(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$ replaced by the semi-inner product $(\mathbf{x}, \mathbf{y})_B = \mathbf{x}^T B\mathbf{y}$. The B-orthogonal Arnoldi method constructs a B-orthonormal basis $\mathbf{v}_1, \ldots, \mathbf{v}_{k+1}$ for the Krylov subspace

$$\mathcal{K}^{k+1}(S, \mathbf{v}_1) = \operatorname{span}(\mathbf{v}_1, S\mathbf{v}_1, \ldots, S^k\mathbf{v}_1),$$

where $S = (A - \sigma B)^{-1}B$. The basis vectors are related by

$$SV_k = V_k H_k + h_{k+1,k}\mathbf{v}_{k+1}\mathbf{e}_k^T = V_{k+1}\underline{H}_k, \quad V_{k+1}^T B V_{k+1} = I, \tag{7.3.1}$$

where $V_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k] \in \mathbb{R}^{n\times k}$ and $H_k \in \mathbb{R}^{k\times k}$ and $\underline{H}_k = [H_k^T, h_{k+1,k}\mathbf{e}_k]^T \in \mathbb{R}^{(k+1)\times k}$ are upper Hessenberg[1]. Relation (7.3.1) characterizes a k-step Arnoldi

[1] Barred identifiers $\underline{H}_k$ are elements of $\mathbb{R}^{(k+1)\times k}$, whereas $H_k \in \mathbb{R}^{k\times k}$.

factorization. Like in the standard Arnoldi method, approximate eigenpairs $(\theta_i, V_k\mathbf{y}_i)$, called Ritz pairs, can be computed from eigenpairs $(\theta_i, \mathbf{y}_i)$ of H_k.

The usual criterion for convergence of a Ritz pair $(\theta, V_k\mathbf{y})$ with $H_k\mathbf{y} = \theta\mathbf{y}$ is derived from the relation

$$SV_k\mathbf{y} = V_kH_k\mathbf{y} + h_{k+1,k}\mathbf{v}_{k+1}\mathbf{e}_k^T\mathbf{y} = \theta V_k\mathbf{y} + h_{k+1,k}\mathbf{v}_{k+1}\mathbf{e}_k^T\mathbf{y}.$$

If $\|h_{k+1,k}\mathbf{v}_{k+1}\mathbf{e}_k^T\mathbf{y}\|$ is smaller than a given tolerance τ, the Ritz pair $(\theta, V_k\mathbf{y})$ is said to be converged. It follows that if the $\mathbf{v}_i$ are orthonormalized in the 2-norm, it suffices to inspect $|h_{k+1,k}\mathbf{e}_k^T\mathbf{y}|$. Since for B-orthogonal Arnoldi the B-inner product is used, the convergence criterion becomes

$$|h_{k+1,k}\mathbf{e}_k^T\mathbf{y}| \cdot \|\mathbf{v}_{k+1}\|_2 < \tau. \tag{7.3.2}$$

In [97] and [104], the use of the semi-inner product is motivated by the fact that the B inner product is not affected by components of $\mathbf{v}_i$ in the null space of B, $\mathcal{N}(B)$, and hence H_k is independent of components in $\mathcal{N}(S)$. Note, however, that H_k can be corrupted by components in the generalized null space $\mathcal{G}(S)$. Moreover, *because* the B-inner product is not affected by components in the null space of B, there is no reason to assume that components of v_i in the null space of B will not grow; for $\mathbf{z} \in \mathcal{N}(B)$, $\mathbf{x} \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$, one has $\|\mathbf{x}\|_B = \|\mathbf{x} + \alpha\mathbf{z}\|_B$. As a consequence, the Ritz vector $V_k\mathbf{y}$ will be spoiled with error components in $\mathcal{N} + \mathcal{G}$ (see also [97, Sect. 4.1] and [104, Sect. 2.3]).

The presence of components in $\mathcal{N} + \mathcal{G}$ in the Arnoldi basis may not only cause spurious eigenvalues and inaccurate Ritz vectors, it may also hamper convergence to the wanted eigenvalues. Purification techniques aim at eliminating the components in $\mathcal{N} + \mathcal{G}$ from the Arnoldi vectors, with the following three goals:

- removal of spurious eigenpair approximations;
- improvement of wanted eigenpair approximations by removing $\mathcal{N} + \mathcal{G}$ components from the Ritz vectors;
- increase of the speed of convergence.

Following [97], the notion of purification can be used in several ways, but the idea boils down to eliminating components in $\mathcal{N} + \mathcal{G}$ of a vector $\mathbf{x}$ by applying S^{j_s} to it, either explicitly or implicitly. In exact arithmetic, the effect is that $S^{j_s}\mathbf{x} \in \mathcal{R}(S^{j_s})$, i.e. $S^{j_s}\mathbf{x}$ is in the wanted eigenspace. See [47] and [104] for the first occurences of the term purification.

Implicitly restarted B-orthogonal Arnoldi with purification

In [97, Sect. 3.2], an implicitly restarted Arnoldi method with B-inner product is proposed. The method, see Alg. 7.1, reduces the corruption of H_k by components in $\mathcal{N}$ and $\mathcal{G}$ significantly (after the implicit restart), and only requires one additional purification step of the Ritz vectors. The result of the implicit restart in step 4,

Algorithm 7.1 Implicitly restarted B-orthogonal Arnoldi with purification

1: Choose an initial vector $\mathbf{v}_1 \leftarrow S^2\mathbf{v}_1$
2: Do $k+1$ steps of B-orthogonal Arnoldi to compute V_{k+2} and $\underline{H}_{k+1}$
3: Compute the QR-factorization $\underline{H}_{k+1} = \underline{Q}_{k+1}R_{k+1}$
4: Implicitly restart: $W_{k+1} = V_{k+2}\underline{Q}_{k+1}, \underline{G}_k = R_{k+1}\underline{Q}_k$
5: Compute eigenpairs $(\theta_i, \mathbf{y}_i)$ of the upper $k \times k$ part of $\underline{G}_k$
6: Purify the Ritz vectors: $\mathbf{x}_i = S(W_k\mathbf{y}_i) = W_{k+1}\underline{G}_k\mathbf{y}_i$
7: The eigen approximations for the generalized problem are $(1/\theta_i + \sigma, \mathbf{x}_i)$

is that the Arnoldi vectors in W_{k+1} and the upper Hessenberg $\underline{G}_k$ are the same as the ones that would have been computed with starting vector $S\mathbf{v}_1/\|S\mathbf{v}_1\|_B$. In other words, the implicit restart removes the $\mathcal{N}$ part from V_{k+2} and the $\mathcal{G}$ part from $\underline{H}_{k+1}$, and it maps the $\mathcal{G}$ part from V_{k+2} to the $\mathcal{N}$ part of W_{k+1}. Note that because of the B-inner product, $\underline{H}_{k+1}$ and $\underline{G}_k$ are free of contributions of components in $\mathcal{N}$. The second purification, in step 6, removes the $\mathcal{N}$ part from the Ritz vector (the $\mathcal{G}$ part was already removed by the implicit restart). The method can still fail due to corruption of $\underline{H}_{k+1}$ by rounding errors, but this can be detected by inspecting $\|R_k^{-1}\|_2$ [97, Thm. 4]: if $\|R_k^{-1}\|_2$ is large and growing for successive values of k, spurious Ritz values may be computed. Secondly, purification of the Ritz vector $W_k\mathbf{y}_i$ may fail if the corresponding Ritz value θ_i is small, i.e. $\theta_i \sim \epsilon\|S\|$.

7.3.2 Exploiting the structure of $A\mathbf{x} = \lambda B\mathbf{x}$

In [97, p. 670], [84, p. 8] and [34, p. 1313] it is concluded that the reduced problem

$$S_1\mathbf{u} = \tilde{\lambda}\mathbf{u}, \qquad S_1 \in \mathbb{R}^{m\times m},$$

see also (7.2.3), is only of theoretical interest, because S_1 and S_2 depend on blocks in A^{-1} which are unlikely to be known. However, matrix vector multiplications with S_1, the only operation with S_1 that is required by the Arnoldi algorithm, and with S_2, can easily be performed by making use of the available multiplication with S. Note that in practical situations also S is not available explicitly and that matrix vector multiplications with S are for instance implemented using the LU-factorization of A.

Theorem 7.3.1. *Let $S \in \mathbb{R}^{n\times n}$ have the block structure*

$$\begin{bmatrix} S_1 & 0 \\ S_2 & 0 \end{bmatrix},$$

with $S_1 \in \mathbb{R}^{m\times m}, S_2 \in \mathbb{R}^{k\times m}$, and let $P = [I_m, 0]^T \in \mathbb{R}^{n\times m}$, $Q = [0, I_k]^T \in \mathbb{R}^{n\times k}$ with $I_m \in \mathbb{R}^{m\times m}$ an identity matrix of dimension m. Then for $\mathbf{x} \in \mathbb{C}^m$,

$$\begin{aligned} S_1\mathbf{x} &= P^T SP\mathbf{x}, \\ S_2\mathbf{x} &= Q^T SP\mathbf{x}. \end{aligned}$$

Proof. The results follow immediately from the identities $S_1 = P^T SP$ and $S_2 = Q^T SP$. □

The operations with P and Q in Theorem 7.3.1 can be performed very efficiently and hence with virtually no additional costs the Arnoldi method can be applied to S_1. This leads to Alg. 7.2, a modification of Alg. 7.1. The Arnoldi basis vectors have length $m < n$, which reduces the costs of orthogonalization (although usually the costs of operations with S are dominant). In step 1, only a single explicit purification of the initial vector is needed. Furthermore, the B-inner product, that was used for its purifying property, and the purification in step 6, are no longer needed, because the implicit restart removes all corruption by components in $\mathcal{N}$ from V_{k+2} and H_{k+1}. On the other hand, to recover the eigenvectors of the original problem, an additional multiplication with S_2 is needed.

Algorithm 7.2 Implicitly restarted Arnoldi for S_1

1: Choose an initial vector $\mathbf{v}_1 \leftarrow S_1\mathbf{v}_1 \in \mathbb{R}^m$
2: Do $k+1$ steps of Arnoldi with S_1 to compute V_{k+2} and $\underline{H}_{k+1}$
3: Compute the QR-factorization $\underline{H}_{k+1} = \underline{Q}_{k+1}R_{k+1}$
4: Implicitly restart: $W_{k+1} = V_{k+2}\underline{Q}_{k+1}, \underline{G}_k = R_{k+1}\underline{Q}_k$
5: Compute eigenpairs $(\theta_i, \mathbf{y}_i)$ of the upper $k \times k$ part of $\underline{G}_k$
6: Compute $\mathbf{p}_i = \theta_i^{-1} S_2\mathbf{x}_i$ with $\mathbf{x}_i = W_k\mathbf{y}_i$
7: The eigen approximations for the generalized problem are $(1/\theta_i + \sigma, [\mathbf{x}_i^*, \mathbf{p}_i^*]^*)$

Improved rounding error analysis

The most important consequence of Theorem 7.3.1, however, is that the results of the error analysis in [97, Section 5] improve considerably. Following the notation and assumptions there, let $P_{\mathcal{R}_1}$ and $P_{\mathcal{N}_1}$ be normalized projectors that map a vector into $\mathcal{R}_1 = \mathcal{R}(S_1)$ and $\mathcal{N}_1 = \mathcal{N}(S_1)$ respectively, so $\mathbf{x} \in \mathbb{C}^m$ can be decomposed uniquely as $\mathbf{x} = P_{\mathcal{R}_1}\mathbf{x} + P_{\mathcal{N}_1}\mathbf{x}$. Note that $P_{\mathcal{N}_1}S_1 = 0$. The computed Arnoldi vectors satisfy

$$\begin{aligned} h_{j+1,j}\mathbf{v}_{j+1} &= S\mathbf{v}_j - \sum_{i=1}^{j} h_{ij}\mathbf{v}_i + \psi_j, \\ h_{ij} &= \mathbf{v}_i^T S\mathbf{v}_j + \delta_{ij}, \\ \mathbf{v}_i^T\mathbf{v}_j &= \begin{cases} 1+\gamma_{ij}, & j = i \\ \gamma_{ij} & j = 1, \ldots, i+1, j \neq i. \end{cases} \end{aligned}$$

In block form, the round-off errors $\|\Psi_{k+1}\|_2$, $\|\Gamma_{k+1}\|_2$ and $\|\Delta_k\|_2$ for the k-step Arnoldi factorization are given by the following relations:

$$V_{k+1}\underline{H}_k = SV_k + \Psi_{k+1}, \tag{7.3.3}$$

$$V_{k+1}^T V_{k+1} = I + \Gamma_{k+1}, \tag{7.3.4}$$

$$\underline{H}_k = V_{k+1}^T SV_k + \Delta_k. \tag{7.3.5}$$

Result 7.3.2. *The $\mathcal{N}_1$ component in $\mathbf{v}_j$ may increase as j increases.*

Proof. Repeating the proof in [97, Section 4.1] leads to

$$\begin{aligned} h_{j,j+1}P_{\mathcal{N}_1}\mathbf{v}_{j+1} &= P_{\mathcal{N}_1}S_1\mathbf{v}_j - \sum_{i=1}^{j} h_{ij}P_{\mathcal{N}_1}\mathbf{v}_i + P_{\mathcal{N}_1}\psi_j \\ &= -\sum_{i=1}^{j} h_{ij}P_{\mathcal{N}_1}\mathbf{v}_i + P_{\mathcal{N}_1}\psi_j. \end{aligned}$$

There is no reason to assume that $\|P_{\mathcal{N}_1}\mathbf{v}_{j+1}\|$ does not increase. □

The improvement over the result in [97, Section 4.1] is that for Arnoldi applied to S, there may be an increase of both components in $\mathcal{N}$ and $\mathcal{G}$, while here there only may be a smaller increase of components in $\mathcal{N}_1$.

The following result shows the improved effect of the implicit purification via $\mathbf{x}_j = S(V_k\mathbf{z}_j) = V_{k+1}\underline{H}_k\mathbf{z}_j$, where $\mathbf{z}_j$ is an eigenvector of H_k. Although this purification is not needed in Alg. 7.2, as will become clear in result 7.3.4 and Theorem 7.3.5, it is included here, however, to show that the relative contributions of the $\mathcal{N}_1$ components are smaller than in the results in [97, Section 4.2].

Result 7.3.3. *The purification operation $\mathbf{x}_j = V_{k+1}\underline{H}_k\mathbf{z}_j$ produces an approximate eigenvector with no $\mathcal{N}_1$ component. This step may fail if $|\theta_j^{-1}| \gg \epsilon_M^{-1}$, where ϵ_M is the machine precision number.*

Proof. From the proof in [97, Section 4.2], it follows that the purified $\mathbf{x}_j$ computed by $\mathbf{x}_j = V_{k+1}\underline{H}_k\mathbf{z}_j$ with $\|\mathbf{z}_j\|_2 = 1$ satisfies

$$\begin{aligned} P_{\mathcal{N}_1}\mathbf{x}_j &= P_{\mathcal{N}_1}S_1V_k\mathbf{z}_j + P_{\mathcal{N}_1}\xi_j \\ &= P_{\mathcal{N}_1}\xi_j, \end{aligned} \tag{7.3.6}$$

with

$$\|\xi_j\|_2 \leq 3k^{3/2}\|V_{k+1}\|_F\|A^{-1}\|_2\epsilon_M + \|\Psi_{k+1}\|_2 + O(\epsilon_M^2).$$

If $\|\mathbf{z}_j\|_2 = 1$ (note that $H_k\mathbf{z}_j = \theta_j\mathbf{z}_j$), then $\|V_k\mathbf{z}_j\|_2 \simeq 1$ and $\|\mathbf{x}_j\|_2 \simeq \theta_j$ and hence relative contributions of the $\mathcal{N}_1$ components in $\mathbf{x}_j$ are obtained by dividing (7.3.6) by θ_j. If θ_j is small, these relative contributions become large and purification may fail. If $|\theta_j^{-1}|\|\xi_j\|_2 \ll 1$, then the $\mathcal{N}_1$ component in $\mathbf{x}_j$ is removed. □

Result 7.3.4. *One implicit restart of Arnoldi produces a $\underline{G}_k$ that is not corrupted by $\mathcal{N}_1$ components, and a W_{k+1} that has no $\mathcal{N}_1$ component. This step may fail if $\|R_{k+1}^{-1}\|_2 \gg \epsilon_M^{-1}$.*

Proof. Repeating the proof in [97, Section 4.4] leads to

$$\|P_{\mathcal{N}_1}W_{k+1}\|_2 \leq \|P_{\mathcal{N}_1}\Xi_{k+1}\|_2 + O(\epsilon_M^2),$$

with

$$\begin{aligned}\|\Xi_{k+1}\|_2 &\leq (k+2)^{3/2}\|V_{k+2}\|_F\epsilon_M \\ &\quad +(\omega\|V_{k+1}\|_2\|A^{-1}\|_2\epsilon_M + \|\Psi_{k+2}\|_2)\|R_{k+1}^{-1}\|_2 + O(\epsilon_M^2),\end{aligned}$$

and $\omega = O(1)$. If $\|R_{k+1}^{-1}\|_2$ is small, W_{k+1} has no significant components in $\mathcal{N}_1$. If $\|R_{k+1}^{-1}\|_2$ is large, then $\|\Xi_k\|_2 \gg \epsilon_M$ and W_{k+1} can have components in $\mathcal{N}_1$. Consequently, G_k is corrupted by the components in $\mathcal{N}_1$ and may cause spurious Ritz values. □

Compared to the results in [97, Section 4.4], the corruption in W_{k+1} and G_k is decreased.

The following theorem shows that, as a Ritz pair $(\tilde{\lambda}, \mathbf{x})$ with not too small $\tilde{\lambda}$ converges to an eigenpair of S_1, it is purified automatically. This is consistent with results 7.3.3 and 7.3.4, and also explains why implicit purification of converged Ritz pairs (step 6 in Alg. 7.1) is not needed.

Theorem 7.3.5. *Let $(\tilde{\lambda}, \mathbf{x})$ be a converged Ritz pair of S_1, with $\mathbf{r} = S_1\mathbf{x} - \tilde{\lambda}\mathbf{x}$ and $\|\mathbf{r}\|_2 < \epsilon$. Then $\|P_{\mathcal{N}_1}\mathbf{x}\|_2 \leq \epsilon/\tilde{\lambda}$.*

Proof. Write $\tilde{\lambda}P_{\mathcal{N}_1}\mathbf{x} = P_{\mathcal{N}_1}S_1\mathbf{x} - P_{\mathcal{N}_1}\mathbf{r}$ and note that $P_{\mathcal{N}_1}S_1\mathbf{x} = 0$. □

Although failure of IRA if $\|R_k^{-1}\|_2$ is large is still possible, the results above show that the (growth of the) corruption by $\mathcal{N}_1$ components is reduced. The rounding errors made during the orthogonalization phase are also reduced, because the standard inner product is used instead of the B-inner product, and hence no additional multiplications with B are needed.

Numerical example

To illustrate the new results, the growth of $\|\Psi_{k+1}\|_2$ (see (7.3.3)) for S and S_1 is compared. Figure 7.1 shows $\|\Psi_{k+1}\|_2$ at every Arnoldi iteration for the example matrix pencil taken from [97, Sect. 3.3]. For S, the B-orthogonal Arnoldi method is used, while for S_1 Arnoldi with the usual inner product is used. For both cases, the initial vector $\mathbf{v}_1$, with all entries equal to one, was purified using $\mathbf{v}_1 \leftarrow S^2\mathbf{v}_1$ and $\mathbf{v}_1 \leftarrow S_1\mathbf{v}_1$ respectively. It is clear that the growth of $\|\Psi_{k+1}\|_2$ is much smaller for S_1. The growth of $\|\Psi_{k+1}\|_2$ for S can be explained as follows: let $\mathbf{w}_k$ be the new Arnoldi vector in iteration k, just after orthogonalization against V_k, but before normalization. The B-inner product neglects any components in $\mathcal{N}(B)$, but these components are normalized with the same factor $h_{k+1,k} = \|\mathbf{w}_k\|_B$. If $h_{k+1,k} < 1$, then these components increase in 2-norm. This may lead to an increase of $\|\Psi_{k+1}\|_2$. Typical values of $h_{k+1,k}$ in this example were of order $O(10^{-4})$. An explanation for the apparent stagnation of the growth of $\|\Psi_{k+1}\|_2$ at some iterations may be: the new Arnoldi vector is computed as $S\mathbf{v}_k$ and $S_1\mathbf{v}_k$ respectively, which is in fact an explicit purification of $\mathbf{v}_k$. Combined with a not too small $h_{k+1,k}$, this will cause only a limited increase.

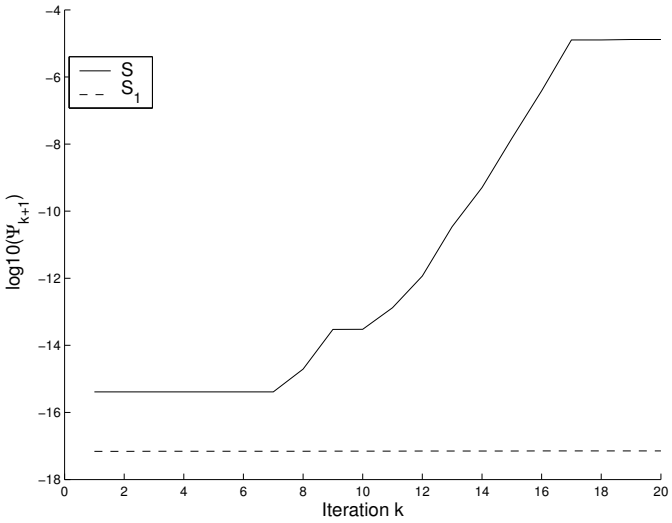


Figure 7.1: The size of $\|\Psi_{k+1}\|_2$ for B-orthogonal Arnoldi applied to $S = (A - 60B)^{-1}B$, and Arnoldi applied to S_1.

A large $\|\Psi_{k+1}\|_2$ may not only prevent the implicit restart with zero shift from purifying the factorization, it also reduces the effect of the implicit purification via $\mathbf{x}_j = V_{k+1}\underline{H}_k\mathbf{z}_j$, as can be deduced from results 7.3.3 and 7.3.4 and their equivalents in [97]. With this in mind, the choice for Arnoldi with S_1 is obvious.

7.3.3 A new strategy

Implicitly restarted Arnoldi with deflation

It is not clear from [97] how the idea of the implicit restart with shift $\sigma_0 = 0$ (Alg. 7.1) is incorporated with the implicitly restarted Arnoldi method with deflation [85, 146]. The IRA method starts with a k-step Arnoldi factorization $SV_k = V_kH_k + h_{k+1,k}\mathbf{v}_{k+1}\mathbf{e}_k^T$. Then, until convergence, the following steps are iterated:

1. Compute the Ritz values θ_i, i.e. the eigenvalues of H_k and split them in a set of wanted Ritz values $\{\theta_1 \dots \theta_j\}$ and unwanted Ritz values $\{\sigma_1 \dots \sigma_p\}$, with $k = j + p$.
2. Apply p QR-steps to H_k with shifts σ_i to remove the unwanted Ritz values.
3. Extend the j-step Arnoldi factorization to a k-step Arnoldi factorization.

Like in Alg. 7.2, the idea now is to implicitly restart with $\sigma_0 = 0$ just before the computation of the Ritz values in step (1), i.e. just after the extension of the Arnoldi factorization. Any detected spurious Ritz values can be removed by

including these as shifts for the implicit restarts. The algorithm is summarized in Alg. 7.3. For details about the implementation of implicit shifts, deflation and the locking procedure, the reader is referred to [85, 146, 147].

Algorithm 7.3 Implicitly restarted Arnoldi for S_1 with purification and deflation

1: Choose an initial vector $\mathbf{v}_1 \leftarrow S_1\mathbf{v}_1$
2: Do $k+1$ steps of Arnoldi to compute V_{k+2} and $\underline{H}_{k+1}$
3: **while** not all converged **do**
4: Purify by applying one restart with $\sigma = 0$: $[V_{k+1}, \underline{H}_k] = \text{purify}(V_{k+2}, \underline{H}_{k+1})$
5: Compute $\lambda(H_k)$ and lock converged wanted Ritz values
6: Select p shifts $\sigma_1, \ldots, \sigma_p$
7: Apply p implicit shifts to compute the $(k-p)$ step Arnoldi factorization $S_1V_{k-p} = V_{k-p+1}\underline{H}_{k-p}$
8: Extend $S_1V_{k-p} = V_{k-p+1}\underline{H}_{k-p}$ to $S_1V_{k+1} = V_{k+2}\underline{H}_{k+1}$
9: **end while**

Exploiting transformations to improve selection and convergence

Besides the shift-and-invert transformation $T_{SI}(A, B, \sigma) = (A - \sigma B)^{-1}$, the generalized Cayley transformation

$$T_C(A, B, \alpha_1, \alpha_2) = (A - \alpha_1 B)^{-1}(A - \alpha_2 B) = B + (\alpha_1 - \alpha_2)T_{SI}, \quad \alpha_1, \alpha_2 \in \mathbb{R} \tag{7.3.7}$$

with $\alpha_1 < \alpha_2$ and $\alpha_1 \neq \lambda_i, i = 1, \ldots, n$, can be used for problems of the form (7.2.2), see [33, 34, 84]. The eigenvalues μ_i of T_C are related to the eigenvalues of (A, B) by the relation $\mu_i = (\lambda_i - \alpha_1)^{-1}(\lambda_i - \alpha_2)$ and the infinite eigenvalues are transformed to 1. Eigenvalues close to α_1 are mapped to eigenvalues far from the unit circle, while eigenvalues close to α_2 are mapped to eigenvalues with small magnitude. The property that is of most use is that eigenvalues with $\text{Re}(\lambda_i) < (\alpha_1 + \alpha_2)/2$ are mapped outside the unit circle, while eigenvalues with $\text{Re}(\lambda_i) > (\alpha_1 + \alpha_2)/2$ are mapped inside the unit circle. The modified Cayley transformation is defined by

$$T_M(A, B, \alpha_1, \alpha_2, \alpha_3) = \begin{bmatrix} K - \alpha_1 M & C \\ C^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} K - \alpha_2 M & \alpha_3 C \\ \alpha_3 C^T & 0 \end{bmatrix}, \tag{7.3.8}$$

and has the same properties as the generalized Cayley transform, except that the infinite eigenvalues are transformed to α_3 [33, 34, 84].

In [84], an algorithm is described where Cayley-transformations are combined with shift-and-invert Arnoldi. The algorithm is based on the hybrid algorithm presented in [33, Section 2.3] and consists of two phases. In the first phase, an r-step Arnoldi factorization is computed using B-orthogonal Arnoldi with purification. The corresponding r Ritz values are used to determine the parameters $\alpha_1, \alpha_2 \in \mathbb{R}$ of the (modified) Cayley transform T_C. In the second phase, implicitly restarted B-orthogonal Arnoldi with purification is applied to T_C to compute the wanted eigenvalues. The parameters α_1, α_2 are updated during the restarts.

The Ritz values that are computed in phase 1 may not have converged (moreover, in [84] there is *no* convergence testing for the Ritz pairs of phase 1, to avoid accepting wrong eigenvalues), and there may be spurious Ritz values as well. Also in the second phase spurious Ritz values may be computed, that make the determination of α_1, α_2 more difficult. In [84] a selection strategy is used to deal with spurious Ritz values. The approach presented here makes such a strategy unnecessary and also reduces the number of different Cayley-transformations needed.

Assume that the $k = 2$ leftmost eigenvalues of (7.2.2) are wanted (complex conjugate pairs counted as one eigenvalue), including any eigenvalues with negative real part. The algorithm is readily adjustable for any number of wanted eigenvalues.

Simply computing the leftmost eigenvalues of $S = T_{SI}(A, 0) = A^{-1}B$ is not advisable for several reasons. Firstly, even if there are eigenvalues with negative real part, the process will most likely be disturbed by spurious Ritz values, as has been explained in the previous sections. Secondly, the leftmost eigenvalues of S do not necessarily correspond to the leftmost eigenvalues of (A, B). The extremal eigenvalues $\tilde{\lambda}_i$ of S, that correspond to the eigenvalues $\lambda_i = 1/\tilde{\lambda}_i$ of (A, B), however, can be computed safely, efficiently and accurately with IRA. These eigenvalues, sorted in increasing real part order, that also not necessarily are the leftmost eigenvalues of (A, B), can be used to compute α_1, α_2 for the modified Cayley transform:

- If $\mathrm{Im}(\lambda_1) = 0$, then $\alpha_1 = \lambda_1 + \frac{\mathrm{Re}(\lambda_2) - \lambda_1}{2}$.
- If $\mathrm{Im}(\lambda_1) \neq 0$, then $\alpha_1 = \lambda_1$.
- In both cases, $\alpha_2 = 2 \times \mathrm{Re}(\lambda_2) - \alpha_1$.

With these choices for α_1, α_2, eigenvalues with $\mathrm{Re}(\lambda_i) < \mathrm{Re}(\lambda_2)$ correspond to eigenvalues $\tilde{\lambda}_i$ of $S_M = T_M(A, B, \alpha_1, \alpha_2, 0)$ with $|\tilde{\lambda}_i| > 1$, while eigenvalues with $\mathrm{Re}(\lambda_i) > \mathrm{Re}(\lambda_2)$ are transformed inside the unit circle. Hence also any missed eigenvalues between λ_1 and λ_2 correspond to eigenvalues μ_i of S_M with $|\mu_i| > 1$. The eigenvalues of S_M with largest magnitude can again be computed by IRA and because the infinite eigenvalues are transformed to $\alpha_3 = 0$, there is virtually no danger that spurious Ritz values will be selected as wanted eigenvalue. As soon as eigenvalues inside the unit circle are computed, it can be safely concluded that the leftmost eigenvalues (including the eigenvalues with negative real part) are found. The strategy is shown in Alg. 7.4.

The strategy consists of two phases: in phase 1 (step 1-2), the largest eigenvalues (in magnitude) of S are computed. Phase 2 (step 3-6) checks for any missed eigenvalues using the Cayley transformation. In step 1, a larger number r will increase the chance of computing the leftmost eigenvalue already in this phase. In step 4, one could also take $\alpha_1 = 0$, but because any missed eigenvalues are expected to be close to λ_1, this is not preferred. Additional verification of any missed eigenvalues can be done by choosing new α_1, α_2 based on the eigenvalues found in step 7 to compute the largest eigenvalues of the new S_M, or by using techniques described in [98].

Algorithm 7.4 Strategy for computing the 2 left most eigenvalues of (A, B)

PHASE 1

1: Compute the $r \geq 2$ largest eigenvalues $\tilde{\lambda}_i$ of S_1 with Alg. 7.3
2: Order $\lambda_i = 1/\tilde{\lambda}_i$, $i = 1, \ldots, r$ by increasing real part

PHASE 2

3: Determine α_1 and α_2, and $\alpha_3 = 0$
4: $S_M = T_M(A, B, \alpha_1, \alpha_2, \alpha_3)$
5: Compute the largest 2 eigenvalues μ_i of S_{M1} with Alg. 7.3
6: The eigenvalues of (A, B) are $\lambda_i = \frac{\alpha_1\mu_i - \alpha_2}{\mu_i - 1}$

The difference with existing approaches is that in the determination of α_1, α_2 rather accurate eigenvalue approximations are used, with as advantages that fewer updates of S_M are needed and that the risk of missing eigenvalues is reduced. Furthermore, the possible disturbance by spurious Ritz values is reduced by first computing only the largest eigenvalues of S_1. Note that with the choice $\alpha_3 = 0$, S_M can be reduced to S_{M1} in the same way as S to S_1, as described in Section 7.3.2. If $(\mu, \mathbf{u})$ is an exact eigenpair of S_{M1}, then for nonzero μ, $(\mu, [\mathbf{u}^*, \mathbf{p}^*]^*)$ with $\mathbf{p} = \mu^{-1} S_{M2}\mathbf{u}$ is an exact eigenpair of S_M. If $(\mu, [\mathbf{u}^*, \mathbf{p}^*]^*)$ is an eigenpair of the modified eigenvalue problem (7.3.8), then $(\frac{\alpha_1\mu_i - \alpha_2}{\mu_i - 1}, [\mathbf{u}^*, \mathbf{q}^*]^*)$, with $\mathbf{q} = (\mu - \alpha_3)/(\mu - 1)\mathbf{p}$, is an eigenpair of the original generalized eigenvalue problem (7.2.2), provided $\mu \notin \{1, \alpha_3\}$ (see [61, 84]).

It may seem that there is no advantage in using $S_M = T_M(\alpha_1, \alpha_2, 0)$ instead of S, since in exact arithmetic, due to shift-invariance of Krylov subspaces, Arnoldi for S and S_M produces the same eigenvalue estimates of (A, B) [98, Lemma 2.5]. However, when using Arnoldi for S, the spurious Ritz values may be hard to distinguish from wanted leftmost Ritz values, as both may be close to zero, while when using Arnoldi for S_M, the spurious Ritz values (near zero) are clearly separated from the wanted Ritz values (magnitude larger than 1).

7.3.4 Realistic examples

The strategy in Alg. 7.4 is applied to two large-scale examples. The first example is the stability analysis of the flow over a backward facing step, a well known benchmark problem from fluid dynamics [69]. The second examples is the flow in a driven cavity [43, Section 7.1.3]. When referring to (finite) eigenvalues λ_i, it is assumed that the λ_i are sorted in increasing real part order, i.e. λ_1 is the leftmost eigenvalue. For more information about the bifurcation analysis of such nonlinear systems, see [35].

The method `eigs` of Matlab 6.5, which is a wrapper around ARPACK [86], is used in all experiments. The stopping criterion is $\tau = 10^{-6}$ and the size of the Arnoldi factorization is $k = 20$.

Table 7.1: Statistics for Alg. 7.4 for the flow over a backward facing step with Reynolds number $Re = 800$ (Section 7.3.4): number of restarts, time, found eigenvalues and residuals after each phase.

	reduced		unreduced	
	phase 1	phase 2	phase 1	phase 2
#restarts	3	2	3	3
time (s)	118	95	120	97
eigenvalues	$\lambda_1, \lambda_{2,3}$	$\lambda_1, \lambda_{2,3}$	$\lambda_1, \lambda_{2,3}$	$\lambda_1, \lambda_{2,3}$
$\max_i \|A\mathbf{x}_i - \lambda_i B\mathbf{x}_i\|$	$1 \cdot 10^{-12}$	$1 \cdot 10^{-12}$	$9 \cdot 10^{-11}$	$9 \cdot 10^{-11}$

Table 7.2: Statistics for Alg. 7.4 for the driven cavity with Reynolds number $Re = 500$ (Section 7.3.4): number of restarts, time, found eigenvalues and residuals after each phase.

	$r = 2$		$r = 5$	
	phase 1	phase 2	phase 1	phase 2
#restarts	1	4	6	4
time (s)	33	112	106	112
eigenvalues	λ_1, λ_4	$\lambda_1, \lambda_{2,3}$	λ_1, λ_{4-7}	$\lambda_1, \lambda_{2,3}$
$\max_i \|A\mathbf{x}_i - \lambda_i B\mathbf{x}_i\|$	$1 \cdot 10^{-16}$	$1 \cdot 10^{-14}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-14}$

Flow over a backward facing step

The matrices A and B with $n = m + p = 21,730 + 7,872 = 29,602$, were obtained using the package IFISS [138]. The Reynolds number was $Re = 800$ (see [43, p. 315]). Table 7.1 shows statistics for Alg. 7.4 with $r = 2$, both for S and the reduced problem S_1. The leftmost eigenvalues $\lambda_1 = 6.04 \cdot 10^{-2}$ and $\lambda_{2,3} = 7.97 \cdot 10^{-2} \pm i1.92 \cdot 10^{-2}$ were already found in the first phase of the algorithm: the validation in phase 2 did not result in new eigenvalues. Although the running times for both the reduced and the unreduced problem are equal, as expected, the residuals are better for the reduced problem. The claim in [69], that the steady state flow at a Reynolds number $Re = 800$ is stable, is confirmed by the results.

Flow in a driven cavity

The matrices A and B with $n = m + p = 8,450 + 1,089 = 9,539$, for Reynolds number $Re = 500$, were obtained using the package IFISS [138]. Table 7.2 shows statistics for Alg. 7.4 with $r = 2$ and $r = 5$, for the reduced problem S_1. The eigenvalues $\lambda_1 = 3.21 \cdot 10^{-2}$ and $\lambda_4 = 1.01 \cdot 10^{-1}$ were found in the first phase of the algorithm. The validation in phase 2 identified the missed eigenvalue pair $\lambda_{2,3} = 6.20 \cdot 10^{-2} \pm i4.61 \cdot 10^{-1}$. Increasing r does not help finding the missed eigenvalue in phase 1, while it increases the running time.

7.4 Jacobi-Davidson methods, preconditioning and purification

If the linear system solves with $A - \sigma B$ are inaccurate or not computable within reasonable time, the strategy based on the implicitly restarted Arnoldi method is no longer applicable, although an inexact variant could be considered [96]. Here a scheme based on the Jacobi-Davidson QZ method [51] is proposed, that does not require exact solves with $(A - \sigma B)$.

The Jacobi-Davidson method [140] combines two principles to compute eigenpairs of eigenvalue problems $A\mathbf{x} = \lambda\mathbf{x}$. The first principle is to apply a Ritz-Galerkin approach with respect to a subspace spanned by $\mathbf{v}_1, \ldots, \mathbf{v}_k$, the search space. The second principle is the computation of a correction orthogonal the current eigenvector approximation. The Jacobi-Davidson method for generalized eigenvalue problems will be briefly explained in Sections 7.4.1 and 7.4.2. For a more detailed description, the reader is referred to [51, 139, 140].

In Section 7.4.3, it will be shown that when an exact preconditioner is used to solve the correction equation, purification is obtained automatically. In Section 7.4.4, this fact will be combined with other properties of Jacobi-Davidson to obtain an efficient method for the computation of a few selected eigenvalues.

7.4.1 The Jacobi-Davidson method for generalized eigenproblems

Given the generalized eigenvalue problem

$$A\mathbf{x} = \lambda B\mathbf{x}, \qquad \mathbf{x} \neq 0,$$

with $A, B \in \mathbb{R}^{n\times n}$, the Jacobi-Davidson method applies a Petrov-Galerkin condition to compute approximate eigenpairs. If the search space is spanned by $\mathbf{v}_1, \ldots, \mathbf{v}_k$, with $V_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$ orthogonal, and the test space is spanned by $\mathbf{w}_1, \ldots, \mathbf{w}_k$, with $W_k = [\mathbf{w}_1, \ldots, \mathbf{w}_k]$ orthogonal, the Petrov-Galerkin condition becomes

$$AV_k\mathbf{s} - \theta BV_k\mathbf{s} \perp \{\mathbf{w}_1, \ldots, \mathbf{w}_k\}.$$

This leads to the reduced $k \times k$ system

$$W_k^* AV_k\mathbf{s} = \theta W_k^* BV_k\mathbf{s},$$

which can be solved using full space methods like QZ to compute eigenpair approximations $(\theta_i, \mathbf{q}_i = V_k\mathbf{s}_i)$ of (7.4.1).

Given such an eigenpair approximation $(\theta_i, \mathbf{q}_i)$, the question is how to expand the search and test space to improve the approximation. With the corresponding residual vector given by

$$\mathbf{r}_i = (A\mathbf{q}_i - \theta_i B\mathbf{q}_i),$$

the Jacobi-Davidson method computes a correction $\mathbf{t} \perp \mathbf{q}_i$ from the Jacobi-Davidson correction equation

$$(I - \mathbf{z}_i\mathbf{z}_i^*)(A - \theta_i B)(I - \mathbf{q}_i\mathbf{q}_i^*)\mathbf{t} = -\mathbf{r}_i, \tag{7.4.1}$$

where the test vector $\mathbf{z}_i = \mu A\mathbf{q}_i + \nu B\mathbf{q}_i$ for a suitable pair $\mu, \nu \in \mathbb{C}$. The search space is expanded with $\mathbf{t}$ and the test space is expanded with $\mu A\mathbf{t} + \nu B\mathbf{t}$. A Ritz pair is accepted if $\|\mathbf{r}_i\|_2 = \|(A\mathbf{q}_i - \theta_i B\mathbf{q}_i)\|_2$ is smaller than a given tolerance.

7.4.2 Jacobi-Davidson QZ

In [51, 139], the Jacobi-Davidson method is extended with deflation. The Jacobi-Davidson QZ (JDQZ) method computes a partial generalized Schur form of the pencil (A, B). Let the current approximate partial generalized Schur form be given by

$$AQ_k = Z_k S_k, \qquad BQ_k = Z_k T_k,$$

with Q_k, Z_k $n \times k$ matrices and S_k, T_k upper triangular $k \times k$ matrices. The problem of finding the next Schur triple $(\mathbf{q}_{k+1}, \mathbf{z}_{k+1}, (\alpha_{k+1}, \beta_{k+1}))$ with $\theta_{k+1} = \alpha_{k+1}/\beta_{k+1}$ can be rewritten as a deflated generalized eigenvalue problem

$$Q_k^*\mathbf{q}_{k+1} = 0, \qquad (I - Z_kZ_k^*)(\beta_{k+1}A - \alpha_{k+1}B)(I - Q_kQ_k^*)\mathbf{q}_{k+1} = 0, \tag{7.4.2}$$

which can be solved by the Jacobi-Davidson method. With the search space represented by the orthogonal matrix V and the test space by the orthogonal matrix W, so that $V^*Q_k = W^*Z_k = 0$, the reduced system matrices become

$$\begin{aligned} M_A &\equiv W^*(I - Z_kZ_k^*)A(I - Q_kQ_k^*) = W^*AV, \\ M_B &\equiv W^*(I - Z_kZ_k^*)B(I - Q_kQ_k^*) = W^*BV. \end{aligned}$$

The generalized Schur decomposition of (M_A, M_B) is computed using QZ:

$$Z_M^*M_AQ_M = S_A, \qquad Z_M^*BQ_M = S_B.$$

The generalized Schur form is ordered with respect to the target τ, and an approximate Petrov triple for (7.4.2) is obtained as

$$(\tilde{\mathbf{q}}, \tilde{\mathbf{z}}, (\tilde{\alpha}, \tilde{\beta})) = (VQ_M\mathbf{e}_1, WZ_M\mathbf{e}_1, (s_{A,11}, s_{B,11})).$$

Given a Petrov triple $(\tilde{q}, \tilde{z}, (\tilde{\alpha}, \tilde{\beta}))$ for the deflated problem, the corresponding generalized deflated correction equation becomes

$$(I - \tilde{\mathbf{z}}\tilde{\mathbf{z}}^*)(I - Z_kZ_k^*)(\tilde{\beta}A - \tilde{\alpha}B)(I - Q_kQ_k^*)(I - \tilde{\mathbf{q}}\tilde{\mathbf{q}}^*)\mathbf{t} = -\tilde{\mathbf{r}}_i, \tag{7.4.3}$$

where the residual $\tilde{\mathbf{r}}$ is

$$\tilde{\mathbf{r}} = (I - Z_kZ_k^*)(\tilde{\beta}A - \tilde{\alpha}B)(I - Q_kQ_k^*)\tilde{\mathbf{q}},$$

and $Q_k^*\mathbf{t} = Z_k^*\tilde{\mathbf{z}} = Q_k^*\tilde{\mathbf{q}} = 0$, $\tilde{\mathbf{q}}^*\mathbf{t} = 0$, $\|t\|_2 = 1$. The search space is expanded with the orthogonal complement of t, and the test space is orthogonally expanded with $(I - Z_kZ_k^*)(\mu A + \nu B)(I - Q_kQ_k^*)\mathbf{t}$.

If the correction equation is solved exactly, the Jacobi-Davidson method converges asymptotically quadratically. In fact, the method can be shown to be a Newton scheme. Solving the correction equation exactly may be too expensive in practice and therefore Krylov subspace methods with preconditioning are used to solve the correction equation approximately. With a preconditioner $K \approx A - \tau B$, the correction equation (7.4.3) can be preconditioned by

$$(I - \tilde{\mathbf{z}}\tilde{\mathbf{z}}^*)(I - Z_kZ_k^*)K(I - Q_kQ_k^*)(I - \tilde{\mathbf{q}}\tilde{\mathbf{q}}^*).$$

With $Q_k := [Q_k, \tilde{\mathbf{q}}]$, $Z_k := [Z_k, \tilde{\mathbf{z}}]$, $Y_k = K^{-1}Z_k$ and $H_k = Q_k^*Z_k$, the left preconditioned correction equation becomes

$$(I - Y_kH_k^{-1}Q_k^*)K^{-1}(\beta A - \alpha B)(I - Y_kH_k^{-1}Q_k^*)\mathbf{t} = -\mathbf{r}, \tag{7.4.4}$$

where $\mathbf{r} = (I - Y_kH_k^{-1}Q_k^*)K^{-1}\tilde{\mathbf{r}}$. In the appendix (Section 7.6) it is described how for certain types of ILU preconditioners the process of solving the correction equation (7.4.3) can be implemented efficiently.

7.4.3 Purification

Jacobi-Davidson style methods select a new Petrov pair according to some criterion, for instance the leftmost Petrov pair, at every iteration. In the absence of infinite eigenvalues, selecting the leftmost Petrov pair will usually result in convergence to the leftmost eigenvalue, assuming that the initial search space contains components in that direction. In the presence of infinite eigenvalues however, this will no longer be a smart strategy: Petrov values will go to infinity, without a proper mechanism to identify them as infinite eigenvalue approximations.

If the search space is restricted to $\mathcal{R}(S^j)$, approximations to infinite eigenvalues can be avoided. Projection of the search space vectors onto $\mathcal{R}(S^j)$ is not attractive because an orthogonal basis for $\mathcal{R}(S^j)$ is not cheaply available. The following lemmas are needed for proving Theorem 7.4.6, which states that if an exact preconditioner[2] is used for the correction equation and if the initial search space $V_0 \subset \mathcal{R}(S^j)$, then, in exact arithmetic, no spurious eigenvalues are computed during the Jacobi-Davidson process.

Lemma 7.4.1. *Let* $\mathbf{q} = ((A - \sigma B)^{-1}B)^j\mathbf{x} \in \mathcal{R}(S^j)$ *and* $K = A - \tau_0 B$. *Then* $\mathbf{r} = (\beta A - \alpha B)\mathbf{q} \in \mathcal{R}(BS^{j-1})$.

Proof. The result follows from some linear algebra:

$$\begin{aligned}(\beta A - \alpha B)\mathbf{q} &= \beta((A - \sigma B) + (\sigma - \alpha/\beta)B)\mathbf{q} \\ &= \beta((\sigma - \alpha/\beta)B\mathbf{q} + (A - \sigma B)((A - \sigma B)^{-1}B)^j)\mathbf{x} \\ &= \beta B((\sigma - \alpha/\beta)\mathbf{q} + ((A - \sigma B)^{-1}B)^{j-1})\mathbf{x} \in \mathcal{R}(BS^{j-1}),\end{aligned}$$

[2] An exact preconditioner is a preconditioner $K = A - \tau_0 B$ for which linear systems of the form $Kx = y$ can be solved exactly, for instance by using an exact LU-factorization $LU = K$.

where in the last step $\mathcal{R}(BS^j) \subseteq \mathcal{R}(BS^{j-1})$ is used. □

Lemma 7.4.2. *Let* $\mathbf{y} = BS^{j-1}\mathbf{x} \in \mathcal{R}(BS^{j-1})$ *and* $K = A - \tau_0 B$. *Then* $K^{-1}\mathbf{y} \in \mathcal{R}(S^j)$.

Proof. With basic linear algebra, one finds

$$\begin{aligned} K^{-1}\mathbf{y} &= (A-\tau_0 B)^{-1}\mathbf{y} \\ &= (A-\tau_0 B)^{-1}BS^{j-1}\mathbf{x} \\ &= (A-\sigma B)^{-1}(I+(\tau_0-\sigma)B(A-\tau_0 B)^{-1})BS^{j-1}\mathbf{x} \in \mathcal{R}(S^j). \end{aligned}$$

□

Lemma 7.4.3. *Let* $\mathbf{r} \in \mathcal{R}(S^j)$, $K = A - \tau_0 B$, $AQ_k = Z_k S_A$, $BQ_k = Z_k S_B$, $\mathbf{q} \in \mathcal{R}(S^j)$, $\mathbf{z} = \nu A\mathbf{q} + \mu B\mathbf{q}$, $Y_k = K^{-1}[Z_k, \mathbf{z}]$ *and* $H_k = [Q_k, \mathbf{q}]^* Z_k$. *Then* $(I - Y_k H_k^{-1}[Q_k, \mathbf{q}]^*)\mathbf{r} \in \mathcal{R}(S)$.

Proof. First note that $\mathcal{R}(Z_k) = \mathcal{R}(AQ_k) = \mathcal{R}(BQ_k)$. It follows from Lemma 7.4.1 that $\mathbf{z} \in \mathcal{R}(BS^{j-1})$ and hence $\mathcal{R}(K^{-1}[Z_k, \mathbf{z}]) \subseteq \mathcal{R}(S^j)$. Consequently, $(I - Y_k H_k^{-1}[Q_k, \mathbf{q}]^*)\mathbf{r} \in \mathcal{R}(S^j)$. □

Lemma 7.4.4. *Let* $\mathbf{r} \in \mathcal{R}(S^j)$, $K = A - \tau_0 B$, $AQ_k = Z_k S_A$, $BQ_k = Z_k S_B$, $\mathbf{q} \in \mathcal{R}(S^j)$, $\mathbf{z} = \nu A\mathbf{q} + \mu B\mathbf{q}$, $Y_k = K^{-1}[Z_k, \mathbf{z}]$ *and* $H_k = [Q_k, \mathbf{q}]^* Z_k$. *Then*

$$\mathcal{K}^j((I - Y_k H_k^{-1} Q_k^*)K^{-1}(\beta A - \alpha B)(I - Y_k H_k^{-1} Q_k^*), \mathbf{r}) \subseteq \mathcal{R}(S^j).$$

Proof. The result follows from applying subsequently Lemma 7.4.3, 7.4.1, 7.4.2 and again 7.4.3. □

This lemma not only enables one to use a Krylov solver for the correction equation, it also has consequences for purification in Jacobi-Davidson.

Lemma 7.4.5. *If the initial search space* $V_0 \subset \mathcal{R}(S^j)$ *and the Jacobi-Davidson correction equation is solved using an exact preconditioner, then all subsequent search spaces* $V_k \subset \mathcal{R}(S^j)$.

Proof. The result follows from Lemma 7.4.4. □

Theorem 7.4.6. *If the initial search space* $V_0 \subset \mathcal{R}(S^j)$ *and the Jacobi-Davidson correction equation is solved using an exact preconditioner, then in exact arithmetic no spurious eigenpairs are computed during the Jacobi-Davidson process.*

Proof. The reduced system is $(M_A, M_B) = (W^*AV, W^*BV)$ with test space $W = \nu AV + \mu BV$. Applying Lemma 7.4.1 to W gives $W \subset \mathcal{R}(BS^{j-1})$ and no spurious eigenvalues are computed. From Lemma 7.4.5 it follows that the Petrov vectors $\mathbf{q}_i = V_k \mathbf{s}_i$ satisfy $\mathbf{q}_i \in \mathcal{R}(S^j)$. □

The last theorem says that, in exact arithmetic, if the Jacobi-Davidson method with exact preconditioning starts with $V_0 \subset \mathcal{R}(S^j)$, then $V_k \subset \mathcal{R}(S^j)$ and $W \subset \mathcal{R}(BS^{j-1})$ and no spurious eigenpairs are computed. In other words, with exact preconditioning the search space is purified automatically. The effect is even enforced because usually more than one iteration of the Krylov solver is needed.

However, in finite arithmetic components in $\mathcal{N}+\mathcal{G}$ may still arise due to rounding errors, and if an exact preconditioner is not available, Theorem 7.4.6 is also not applicable. Fortunately, there is a result similar to Theorem 7.3.5. Let $P_{\mathcal{R}}$, $P_{\mathcal{N}}$ and $P_{\mathcal{G}}$ be normalized projectors that map a vector into $\mathcal{R} = \mathcal{R}(A^{-1}B)$, $\mathcal{N} = \mathcal{N}(A^{-1}B)$ and $\mathcal{G} = \mathcal{G}(A^{-1}B)$ respectively, so $\mathbf{x} \in \mathbb{C}^m$ can be decomposed uniquely as $\mathbf{x} = P_{\mathcal{R}}\mathbf{x} + P_{\mathcal{N}}\mathbf{x} + P_{\mathcal{G}}\mathbf{x}$. The following theorem shows that a converged Petrov pair $(\lambda, \mathbf{x})$ is purified automatically, provided $|\lambda|$, $\|A^{-1}\|_2$ and $\|B\|_2$ are not too large.

Theorem 7.4.7. *Let $(\lambda, \mathbf{x})$ be a converged Petrov pair of (A, B), with $\mathbf{r} = A\mathbf{x} - \lambda B\mathbf{x}$ and $\|\mathbf{r}\|_2 < \epsilon$. Then*

$$\begin{aligned} \|P_{\mathcal{N}}\mathbf{x}\|_2 &\leq \epsilon\|A^{-1}\|_2(1 + |\lambda|\|A^{-1}\|_2\|B\|_2), \\ \|P_{\mathcal{G}}\mathbf{x}\|_2 &\leq \epsilon\|A^{-1}\|_2. \end{aligned}$$

Proof. Use $\mathbf{x} = A^{-1}(\mathbf{r} + \lambda B\mathbf{x})$, $P_{\mathcal{N}}(A^{-1}B) = (A^{-1}B)P_{\mathcal{G}}$ and $P_{\mathcal{G}}(A^{-1})B = 0$. □

7.4.4 Harmonic Ritz-values, exact targets and purification

Numerical experiments show that the JDQZ process with harmonic Petrov values and a target equal to an eigenvalue does not converge to this eigenvalue, but to eigenvalues closest to the target. This observation can be understood from a theoretical point of view, as will be explained next, and may be of use in avoiding convergence to eigenvalues at infinity.

First consider the Jacobi-Davidson process for the ordinary eigenproblem

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

In [140] it is shown that the harmonic Ritz values of A are equal to the eigenvalues of the $k \times k$ matrix

$$\tilde{H}_k = (W_k^*V_k)^{-1}W_k^*AV_k = (W_k^*V_k)^{-1}$$

with $W_k = AV_k$ and $W_k^*W_k = I$. Note that $\tilde{H}_k^{-1} = W_k^*V_k = W_k^*A^{-1}W_k$, the projection of A^{-1} with respect to an orthonormal basis W_k. In practice it is not necessary to invert $\tilde{H}_k^{-1}$, because the harmonic Ritz values of A are the reciprocals of the eigenvalues of $\tilde{H}_k = W_k^*V_k$. Hence, no problems are encountered if $W_k^*V_k$ is singular, which may happen if A has an eigenvalue at zero.

Theorem 7.4.8. *Let $A \in \mathbb{R}^{n\times n}$ be a normal matrix and $\tau \in \mathbb{R}$. If τ exactly equals an eigenvalue of A, then, in exact arithmetic, the Jacobi-Davidson process with harmonic Ritz values and target τ will not converge to the eigenvalue $\lambda = \tau$.*

Proof. Without loss of generality, let $\tau = \lambda = 0$: if $\tau = \lambda \neq 0$, the proof follows for $A - \tau I$. Denote the null space of A by $\mathcal{N}$ and the range of A by $\mathcal{R}$. The eigenspace corresponding to the eigenvalue $\lambda = 0$ is (a subset of) the null space $\mathcal{N}$. However, because of the normality of A, the space spanned by the columns of $W_k = AV_k$ does not contain any elements of the eigenspace of $\lambda = 0$. Because the eigenspaces of A and A^{-1} are the same, the proof would be complete if A^{-1} would exist. However, in this case there is an eigenvalue $\lambda = 0$ and hence A^{-1} does not exist.

If $\delta \in \mathbb{C} \backslash \Lambda(A)$, then $(A + \delta I)^{-1}$ exists and the eigenspaces of A, $A + \delta I$, and $(A + \delta I)^{-1}$ are the same, but the eigenvalues are λ, $\lambda + \delta$, and $(\lambda + \delta)^{-1}$ respectively. With the iteration vectors w_k still generated by Av_k, and hence W_k containing no components of the null space of A and the eigenspace V_δ of $A + \delta I$, it follows that the eigenvalue δ^{-1} will not be contained in the set of eigenvalues of $W_k^*(A + \delta I)^{-1} W_k$. □

In finite arithmetic, W_k can still contain components of the (generalized) null space of A, which may hamper convergence to the desired eigenvalues or even cause convergence to the undesired, perturbed eigenvalue. If the starting vector is in the null space of A, Jacobi-Davidson with harmonic Ritz values will break down.

The proof for the Jacobi-Davidson QZ process for generalized eigenproblems is similar. In [141] these observations are used to derive a selection strategy for Ritz pairs.

7.4.5 A strategy with JDQZ

The strategy in Alg. 7.5 is conceptually the same as Alg. 7.4, with JDQZ instead of IRA. By considering the pencil (B, A) instead of (A, B), the infinite eigenvalues are transformed to zero. The extremal eigenvalues of (B, A) can be computed safely, efficiently and accurately by JDQZ with harmonic Petrov values and target $\tau = 0$ (see Theorem 7.4.8). These eigenvalues can be used to determine α_1, α_2 and $\alpha_3 = 0$ for the modified Cayley transform (see also Section 7.3.3), here formulated as the generalized eigenvalue problem $\mathcal{A}(\alpha_2, \alpha_3)\mathbf{x} = \mu \mathcal{B}(\alpha_1)\mathbf{x}$ with

$$\mathcal{A}(\alpha_2, \alpha_3) = \begin{bmatrix} K - \alpha_2 M & \alpha_3 C \\ \alpha_3 C^T & 0 \end{bmatrix}, \quad \mathcal{B}(\alpha_1) = \begin{bmatrix} K - \alpha_1 M & C \\ C^T & 0 \end{bmatrix}.$$

Eigenpairs that are found in phase 1 (step 1 - 3) can be deflated from the problem in phase 2 (step 4 - 6).

7.4.6 Realistic examples

The strategy in Alg. 7.5 is applied to the test problems of Section 7.3.4. To make a fair comparison with the IRA strategy in Alg. 7.4, two situations were considered:

- The correction equation is not solved exactly, but with 20 steps of (unrestarted) GMRES [133] with preconditioner A.

Algorithm 7.5 Strategy for computing the 2 left most eigenvalues of (A, B)

PHASE 1

1: Choose a suitable preconditioner for the correction equation
2: Compute the $r \geq 2$ largest eigenvalues $\tilde{\lambda}_i$ of (B, A) with JDQZ ($\tau = 0$)
3: Order $\lambda_i = 1/\tilde{\lambda}_i$, $i = 1, \ldots, r$ by increasing real part

PHASE 2

4: Determine α_1 and α_2, and $\alpha_3 = 0$
5: Compute the largest 2 eigenvalues μ_i of $(\mathcal{A}(\alpha_2, \alpha_3), \mathcal{B}(\alpha_1))$ with JDQZ
6: The eigenvalues of (A, B) are $\lambda_i = \frac{\alpha_1 \mu_i - \alpha_2}{\mu_i - 1}$

- The correction equation is solved exactly and an initial search space of size $j_{\min}$ is computed with Arnoldi.

In both situations, the initial vector $\mathbf{v}_1$ had all entries one and was not purified. The search and test space dimensions are limited by $j_{\min} = 15$ and $j_{\max} = 20$, and the residual tolerance was 10^{-6} (see [51] for more details about the several parameters and sophisticated stopping criteria). In situation 1, solves with preconditioner A are needed and hence one could argue that in that case also the IRA strategy in Alg. 7.4 could be used. The goal here however is to show that even if the correction equation is not solved exactly and the initial search space is not constructed with Arnoldi, JDQZ is able to compute the leftmost eigenvalues. In this way, situation 1 resembles the situation where indeed solves with A are not possible and the IRA strategy is not applicable. The quality of the preconditioner influences the speed of convergence of the GMRES process, but this chapter is not concerned with designing a good preconditioner (see [43, Chapter 8] and references therein for preconditioners of related systems). For the experiments, a variant of the JDQZ algorithm, that keeps the search and test spaces real, is used (RJDQZ [158]).

Flow over a backward facing step

Table 7.3 shows statistics for Alg. 7.5 with $r = 2$, for both exact and inexact solution of the correction equation. The leftmost eigenvalues $\lambda_1 = 6.04 \cdot 10^{-2}$ and $\lambda_{2,3} = 7.97 \cdot 10^{-2} \pm i1.92 \cdot 10^{-2}$ were already found in the first phase of the algorithm: the validation in phase 2 did not result in new eigenvalues. The differences in residual norms can be explained by the asymptotically quadratical convergence of the exact variant. Concerning the higher computing times for the inexact variant, one should keep in mind that for stability analysis the quality of the solution (no missed eigenvalues) is the most important. Furthermore, it may be expected that the times can be decreased by using a more effective preconditioning, but this goes beyond the scope of this chapter. The exact variant is faster than implicitly restarted Arnoldi (cf. Table 7.2).

Table 7.3: Statistics for Alg. 7.5 for the backward facing step with Reynolds number $Re = 800$ (Section 7.4.6): number of iterations, restarts, time, found eigenvalues and residuals after each phase. No restarts were needed.

	Inexact		Exact	
	phase 1	phase 2	phase 1	phase 2
#iterations	18	9	4	2
time (s)	1400	1000	85	80
eigenvalues	$\lambda_1, \lambda_{2,3}$	λ_1	$\lambda_1, \lambda_{2,3}$	$\lambda_1, \lambda_{2,3}$
$\max_i \|A\mathbf{x}_i - \lambda_i B\mathbf{x}_i\|$	$1 \cdot 10^{-11}$	$1 \cdot 10^{-11}$	$1 \cdot 10^{-15}$	$1 \cdot 10^{-15}$

Table 7.4: Statistics for Alg. 7.4 for the driven cavity with Reynolds number $Re = 500$ (Section 7.4.6): number of restarts, time, found eigenvalues and residuals after each phase. No restarts were needed.

	Inexact		Exact	
	phase 1	phase 2	phase 1	phase 2
#iterations	13	12	3	3
time (s)	330	340	21	58
eigenvalues	λ_1, λ_4	λ_1,$\lambda_{2,3}$	λ_1, λ_4	λ_1,$\lambda_{2,3}$
$\max_i \|A\mathbf{x}_i - \lambda_i B\mathbf{x}_i\|$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-15}$	$1 \cdot 10^{-15}$

Flow in a driven cavity

Table 7.4 shows statistics for Alg. 7.5 with $r = 2$, for both exact and inexact solution of the correction equation. The eigenvalues $\lambda_1 = 3.21 \cdot 10^{-2}$ and $\lambda_4 = 1.01 \cdot 10^{-1}$ were found in the first phase of the algorithm. The validation in phase 2 found the missed eigenvalue $\lambda_{2,3} = 6.20 \cdot 10^{-2} \pm i4.61 \cdot 10^{-1}$. The exact variant is faster than implicitly restarted Arnoldi (cf. Table 7.2).

7.5 Conclusions

The strategy based on implicitly restarted Arnoldi is a reliable and fast method to compute the leftmost eigenvalues of large-scale eigenproblems, if the solves needed for the shift-and-invert and Cayley transformations can be done efficiently and exactly. By exploiting the structure of the generalized eigenvalue problem and by choosing suitable parameters for the modified Cayley transformation, the troubles caused by infinite eigenvalues are circumvented and no purification is needed.

If the solves that are needed for the transformations cannot be done exactly, the Jacobi-Davidson QZ method is a good alternative. Following the same strategy, JDQZ is able the compute the leftmost eigenvalues, without corruption due to infinite eigenvalues. If solves can be done exactly, it is faster than implicitly restarted Arnoldi. Jacobi-Davidson puts no requirements on the matrix pencil: it

can handle both regular and singular pencils.

7.6 Appendix: Efficient preconditioning

For efficiency reasons, the preconditioner for the correction equation must be available in such a way that solves with K are relatively cheap. In practice, K is usually available as (incomplete) LU factorization of $A-\tau_0 B$ for some target τ_0. For some types of preconditioners, the process of solving correction equation (7.4.4) can be implemented rather efficiently.

7.6.1 The case $K = LU = A - \tau_0 B$

If the factorization $LU = A - \tau_0 B$ is available, then it is a good candidate for a preconditioner $K = LU$ of (7.4.3). Straightforward application of a Krylov solver to (7.4.4) requires one matrix-vector operation with A, one matrix-vector operation with B, two scalar-vector operations, one vector-vector addition, one solve with K, and one projection with $(I - Y_k H_k^{-1} Q_k^*)$ per iteration. The following lemma shows that the operation with A can be saved.

Lemma 7.6.1. *Let $A, B \in \mathbb{C}^{n\times n}$, $\alpha, \beta, \tau_0 \in \mathbb{C}$, $K = LU$ be the LU-factorization of $A - \tau_0 B$ and $\mathbf{t} \in \mathbb{C}^n$. Then $\mathbf{x} \in \mathbb{C}^n$ can be computed from*

$$\mathbf{x} = K^{-1}(\beta A - \alpha B)\mathbf{t}$$

at the expense of one matrix-vector operation with B, one solve with K, one vector-vector addition and two scalar-vector operations.

Proof. If $\beta = 0$, the result follows immediately. If $\beta \neq 0$, straightforward linear algebra gives

$$\begin{aligned} \mathbf{x} &= K^{-1}(\beta A - \alpha B)\mathbf{t} \\ &= \beta K^{-1}(A - \tau_0 B + (\tau_0 - \alpha/\beta)B)\mathbf{t} \\ &= \beta(I + (\tau_0 - \alpha/\beta)K^{-1}B)\mathbf{t}, \end{aligned}$$

where in the last step $K = LU = A - \tau_0 B$ is used. The result follows from counting the operations in the last equality. □

Lemma 7.6.1 says that one can save one matrix-vector operation per iteration of the Krylov solver. Because there are several iterations of the Krylov solver per Jacobi-Davidson iteration, this may lead to a substantial reduction in computational costs. The reduction also depends on the sparsity of A. Note that for JDQR for the ordinary eigenproblem ($B = I$), no matrix-vector operations are needed during iterations of the linear solver.

7.6.2 The case $K = \mathbf{ILU}(A - \tau_0 B)$

If an exact LU-factorization is not available, then in some cases still a computational gain can be obtained. Suppose that the incomplete factorization can be written as

$$K = (L_A - \tau_0 L_B + D)D^{-1}(U_A - \tau_0 U_B + D), \tag{7.6.1}$$

where $A - \tau_0 B = L_A - \tau_0 L_B + \mathrm{diag}(A - \tau_0 B) + U_A - \tau_0 U_B$, and L_A, L_B and U_A, U_B are the strictly lower and strictly upper triangular parts of A and B, respectively. Incomplete LU factorizations and modifications can be written in this form. Then Eisenstat's trick [42, 157] can be applied to save one multiplication with A. In practice, $ILU(0)$-factorizations are a popular choice as preconditioner for the Jacobi-Davidson correction equation, and hence Eisenstat's trick can be applied if two-sided preconditioning is used. With K as in (7.6.1), the system is first (diagonally) preconditioned with D:

$$\tilde{A} - \tau_0 \tilde{B} = D^{-1/2}(A - \tau_0 B)D^{-1/2}$$

It then follows that

$$\tilde{K} = (L_{\tilde{A}} - \tau_0 L_{\tilde{B}} + I)(U_{\tilde{A}} - \tau_0 U_{\tilde{B}} + I) \equiv \tilde{K}_l \tilde{K}_r.$$

With $D_{\tilde{A}\tilde{B}} = \mathrm{diag}(\tilde{A} - \tau_0 \tilde{B})$, an application of $\tilde{K}_l^{-1}(\tilde{A} - \theta\tilde{B})\tilde{K}_r^{-1}$ to a vector $\mathbf{z}$ can be rewritten as

$$\begin{aligned} & (L_{\tilde{A}} - \tau_0 L_{\tilde{B}} + I)^{-1}(\tilde{A} - \theta\tilde{B})(U_{\tilde{A}} - \tau_0 U_{\tilde{B}} + I)^{-1}\mathbf{z} \\ = \;& \tilde{K}_l^{-1}(L_{\tilde{A}} - \tau_0 L_{\tilde{B}} + I + D_{\tilde{A}\tilde{B}} + (\tau_0 - \theta)B - 2I + U_{\tilde{A}} - \tau_0 U_{\tilde{B}} + I)\tilde{K}_r^{-1}\mathbf{z} \\ = \;& \mathbf{t} + (L_{\tilde{A}} - \tau_0 L_{\tilde{B}} + I)^{-1}(\mathbf{z} + ((\tau_0 - \theta)B + D_{\tilde{A}\tilde{B}} - 2I)\mathbf{t}) \end{aligned}$$

where $\mathbf{t} = (U_{\tilde{A}} - \tau_0 U_{\tilde{B}} + I)^{-1}\mathbf{z}$. In this way a multiplication with $\tilde{A}$ is saved.

In order to apply Eisenstat's trick to the deflated Jacobi-Davidson correction, twosided preconditioning is needed. Denote the deflated preconditioner by

$$K = ((I - ZZ^*)K_l)(K_r(I - QQ^*)) \equiv \tilde{K}_l \tilde{K}_r,$$

where $K_l K_r$ is still of the form (7.6.1). The twosided preconditioned deflated operator is

$$\tilde{K}_l^{\dagger}(I - ZZ^*)(\beta A - \alpha B)(I - QQ^*)\tilde{K}_r^{\dagger}, \tag{7.6.2}$$

where $P^{\dagger}\mathbf{s}$ must be read as "solve $\mathbf{t}$ from $P\mathbf{t} = \mathbf{s}$". Straightforward application of the operator (7.6.2) consists of the following operations:

1. Solve $\tilde{\mathbf{t}}$ with $Q^*\tilde{\mathbf{t}} = 0$ from $(I - QQ^*)K_r(I - QQ^*)\tilde{\mathbf{t}} = \mathbf{s}$ for $\mathbf{s}$ with $Q^*K_r^{-1}\mathbf{s} = 0$,
2. Compute $\mathbf{p} = (I - ZZ^*)(\beta A - \alpha B)\tilde{\mathbf{t}}$,
3. Solve $\mathbf{t}$ with $Q^*K_r^{-1}\mathbf{t} = 0$ from $(I - ZZ^*)K_l\mathbf{t} = \mathbf{p}$, for $\mathbf{p}$ with $Z^*\mathbf{p} = 0$.

The requirement $Q^*K_r^{-1}\mathbf{t} = 0$ in step 3 is needed to guarantee that the domain and image space of (7.6.2) are equal[3]. Following the approach in [141], these operations can be simplified. Using the relation $Q^*\tilde{\mathbf{t}} = Q^*K_r^{-1}\mathbf{s} = 0$ in step 1, $\tilde{\mathbf{t}}$ can be computed as $\tilde{\mathbf{t}} = K_r^{-1}\mathbf{s}$ and the projection with $(I - QQ^*)$ on the right in step 2 can be saved. Step 3 can be worked out similarly: $\mathbf{t} = \tilde{\mathbf{p}} - K_l^{-1}ZH_l^{-1}Q^*K_r^{-1}\tilde{\mathbf{p}}$ with $\tilde{\mathbf{p}} = K_l^{-1}\mathbf{p}$ and $H_l = Q^*K_r^{-1}K_l^{-1}Z$. Note that the solution $\mathbf{t}$ also satisfies the equalities

$$Q^*K_r^{-1}\mathbf{t} = 0, \quad (I - ZZ^*)K_l\mathbf{t} = (I - ZZ^*)\mathbf{p},$$

for arbitrary $\mathbf{p}$. This saves the projection with $(I - ZZ^*)$ on the left in step 2. These observations lead to the following efficient scheme for the application of the twosided preconditioned deflated operator (7.6.2):

1. Compute $\tilde{\mathbf{p}} = K_l^{-1}(\beta A - \alpha B)K_r^{-1}\tilde{\mathbf{t}}$,
2. Compute $\mathbf{t} = \tilde{\mathbf{p}} - K_l^{-1}ZH_l^{-1}Q^*K_r^{-1}\tilde{\mathbf{p}}$.

Step 1 now can be computed using Eisenstat's trick (if the preconditioner is of the form (7.6.1)). Because the resulting vector $\mathbf{t}$ is orthogonal to $K_r^{-*}Q$, no intermediate projections are needed for the Krylov subspace method, provided the initial vector is orthogonal to $K_r^{-*}Q$. The right-hand side must be preconditioned as well:

$$\tilde{\mathbf{r}}_i = (I - K_l^{-1}ZH_l^{-1}Q^*K_r^{-1})K_l^{-1}\mathbf{r}_i.$$

The approximate solution $\mathbf{t}$ of the correction equation (the new expansion vector) can be obtained from the solution $\mathbf{y}$ of the twosided preconditioned deflated correction equation

$$P_l(K_l^{-1}(\beta A - \alpha B)K_r^{-1})P_l\mathbf{y} = -\tilde{\mathbf{r}}_i,$$

with $P_l = I - K_l^{-1}ZH_l^{-1}Q^*K_r^{-1}$ as follows:

$$\mathbf{t} = K_r^{-1}\mathbf{y},$$

where explicit projection with $(I - QQ^*)$ is not needed because $Q^*K_r^{-1}\mathbf{y} = 0$.

Addendum

Except for some changes in notation and the appendix, this chapter is also available as [121]

> Joost Rommes, *Arnoldi and Jacobi-Davidson methods for generalized eigenvalue problems $A\mathbf{x} = \lambda B\mathbf{x}$ with singular B*, Preprint 1339, Utrecht University, 2005 (revised 2007),

and has been accepted for publication in Mathematics of Computation.

[3]Thanks to Gerard Sleijpen for observing this choice, which saves one projection per iteration.

Chapter 8

Computing a partial generalized real Schur form using the Jacobi-Davidson method

Abstract. In this chapter, a new variant of the Jacobi-Davidson method is presented that is specifically designed for *real unsymmetric* matrix pencils. Whenever a pencil has a complex conjugate pair of eigenvalues, the method computes the two dimensional real invariant subspace spanned by the two corresponding complex conjugate eigenvectors. This is beneficial for memory costs and in many cases it also accelerates the convergence of the JD method. Both real and complex formulations of the correction equation are considered. In numerical experiments, the RJDQZ variant is compared with the original JDQZ method.

Key words. partial Schur form, real Schur form, ordered Schur form, Jacobi-Davidson, large scale eigenvalue problem

8.1 Introduction

Real unsymmetric matrices or real unsymmetric matrix pencils may have complex eigenvalues and corresponding eigenvectors. Therefore, the (partial generalized) Schur form may consist of complex matrices. In some situations, (e.g., in a continuation context [22]) it is more desirable to compute a real (partial generalized) Schur form. This decomposition consists for a matrix of an orthogonal real matrix and block upper triangular matrix, which has scalars or two by two blocks on the diagonal. The eigenvalues of such a two by two block correspond to two complex conjugate eigenvalues of the matrix (pencil) itself. Advantages of the real Schur form are that it requires less storage, since for every complex conjugate pair of eigenvalues only two real vectors need to be stored instead of two complex vectors,

and that complex conjugate pairs of eigenvalues always appear together.

In this chapter, a variant of the JDQZ method [51] is considered for the computation of a partial generalized real Schur form of a matrix pencil. The original JDQZ method [51] does not use the fact that the pencil is real: (1) it does not exploit the fact that eigenvalues are real or appear in complex conjugate pairs and (2) it needs complex arithmetic, even when only real eigenvalues appear. This is in contrast with other iterative eigenvalue solvers such as the Arnoldi method.

Algorithm 4.1 proposed in [25] computes a partial Schur form in a naive way. This algorithm consists of an outer iteration in which the partial Schur form is expanded by a scalar block whenever the inner iteration, which may consist of the Jacobi-Davidson method applied to a deflated matrix, returns a real eigenvalue, and with a two by two block if the inner iteration returns an eigenvalue with nonzero imaginary part. Thus the Algorithm 4.1 in [25] solves problem (1), but it does not solve problem (2): the inner iteration does not use the fact that the matrix is real.

The variant of the Jacobi-Davidson method, called RJD, that is presented in this chapter, takes advantage of the fact that the pencil is real: it only needs real arithmetic to compute real eigenvalues and computes complex conjugate pairs of eigenvalues by approximating the corresponding two dimensional invariant subspace, by keeping the search and test spaces real. The goal of this chapter is to show that RJD solves problems (1) and (2), and, furthermore, that it converges faster and is more efficient than standard JD, most likely due to the larger search space.

In [26], it is stated that a real implementation of the JD method is used for a comparison with the Riccati method, but implementation details and a comparison with the original JD method are not presented. Also in [51] (Remark 1), the authors hint at a real version of the JD method, but do not pursue this matter further.

The structure of this chapter is as follows. In Section 2, the JDQZ method [51] is discussed. The RJDQZ method is presented in Section 3, and in Section 4, different ways to solve the correction equation in the RJDQZ method are discussed and compared. A comparison of the computational and memory costs between the original JDQZ method and the RJDQZ method is found in Section 5, and a numerical comparison in Section 6. Section 7 concludes.

8.2 A Jacobi-Davidson style QZ method

In this section the JDQZ method [51] for the construction of a partial generalized Schur form of an unsymmetric matrix pencil is discussed.

8.2.1 The Jacobi-Davidson Method for the Generalized Eigenvalue Problem

The Jacobi-Davidson (JD) method [140] iteratively computes approximations to eigenvalues, and their corresponding eigenvectors, that are close to some specified

target τ, of the generalized unsymmetric eigenvalue problem

$$A\mathbf{x} = \lambda B\mathbf{x}, \tag{8.2.1}$$

where A and B are in general unsymmetric $n \times n$ matrices. In each iteration, a search subspace colspan(V) and a test subspace colspan(W) are constructed. V and W are complex $n \times j$ matrices with $j \ll n$ and $V^*V = W^*W = I$. In the first part of an iteration, an approximation to an eigenvector of the generalized eigenvalues problem (8.2.1) is obtained from the projected eigenvalue problem

$$W^*AV\mathbf{u} = \lambda W^*BV\mathbf{u}. \tag{8.2.2}$$

This is a small eigenvalue problem of size $j \times j$, so that a full space method like the QZ method can be used to compute all the eigenvalues and eigenvectors.

Suppose $(\tilde{\lambda}, \mathbf{u})$ is the eigenpair of the projected eigenvalue problem (8.2.2), of which the eigenvalue $\tilde{\lambda}$ is closest to τ. An approximation $(\tilde{\lambda}, \tilde{\mathbf{x}})$ to an eigenpair of the full sized eigenvalue problem (8.2.1) can be constructed by computing $\tilde{\mathbf{x}} = V\mathbf{u}$. The residual vector $\mathbf{r}$ of the approximate eigenpair $(\tilde{\lambda}, \tilde{\mathbf{x}})$ is defined by

$$\mathbf{r} := A\tilde{\mathbf{x}} - \tilde{\lambda}B\tilde{\mathbf{x}}.$$

The second part in a JD iteration is the expansion of the search and test space. The search space V is expanded by an approximate solution $\mathbf{x}$ of the linear equation

$$(I - \tilde{\mathbf{z}}\tilde{\mathbf{z}}^*)(A - \tilde{\lambda}B)(I - \tilde{\mathbf{x}}\tilde{\mathbf{x}}^*)\mathbf{t} = -\mathbf{r}. \tag{8.2.3}$$

This equation is called the Jacobi-Davidson correction equation. Here $\tilde{\mathbf{z}}$ is the vector $\tilde{\mathbf{z}} = (\kappa_0 A + \kappa_1 B)\tilde{\mathbf{x}}$. The test space W is expanded with the vector $\mathbf{w} = (\kappa_0 A + \kappa_1 B)\mathbf{t}$. This procedure is repeated until $||\mathbf{r}||$ is small enough.

There are several possible choices for the complex numbers κ_0 and κ_1. In [51] it is shown that an effective choice for interior eigenvalues is $\kappa_0 = (1 + |\tau|^2)^{-1/2}$ and $\kappa_1 = -\tau(1 + |\tau|^2)^{-1/2}$. This corresponds to the harmonic Petrov value approach, see [51] for more choices.

8.2.2 The JDQZ Method

The JDQZ method is a Jacobi-Davidson style method that is designed to compute an approximation to a partial generalized Schur form of the matrix pair (A, B)

$$AQ_k = Z_k S_k, \quad BQ_k = Z_k T_k, \tag{8.2.4}$$

where Q_k and Z_k are $n \times k$ matrices with orthonormal columns, and S_k and T_k are $k \times k$ upper triangular matrices. Eigenvalues of the pair (S_k, T_k) are also eigenvalues of the pair (A, B).

The first column of Q_k is an eigenvector of the pair (A, B), and can thus be computed with the JD method. Suppose that a partial Schur form (8.2.4) is computed

already. It can be shown [51] that the next Schur vector $\mathbf{q}_{k+1}$ and corresponding eigenvalue λ_{k+1} satisfy $Q_k^*\tilde{\mathbf{q}}_{k+1} = 0$ and

$$(I - Z_k Z_k^*)(A - \lambda_{k+1}B)(I - Q_k Q_k^*)\mathbf{q}_{k+1} = 0. \tag{8.2.5}$$

This deflated generalized eigenvalue problem can be solved with the JD method. The projected generalized eigenvalue problem that has to be solved can be written as

$$W^*(I - Z_k Z_k^*)A(I - Q_k Q_k^*)V\mathbf{u} = \lambda W^*(I - Z_k Z_k^*)B(I - Q_k Q_k^*)V\mathbf{u}, \tag{8.2.6}$$

with $V^*Q_k = W^*Z_k = 0$, where V and W are again search and test spaces, respectively.

Let $(\tilde{\lambda}, \tilde{\mathbf{u}})$ denote an eigenpair of the projected eigenvalue problem (8.2.6). Again an approximation $(\tilde{\lambda}, \tilde{\mathbf{q}})$ to the eigenpair $(\lambda_{k+1}, \mathbf{q}_{k+1})$ of the eigenvalue problem (8.2.5) can be constructed by computing $\tilde{\mathbf{q}} = V\tilde{\mathbf{u}}$. In order to expand the search space V, an approximate solution $\mathbf{t}$ of the correction equation

$$(I - \tilde{\mathbf{z}}\tilde{\mathbf{z}}^*)(I - Z_k Z_k^*)(A - \tilde{\lambda}B)(I - Q_k Q_k^*)(I - \tilde{\mathbf{q}}\tilde{\mathbf{q}}^*)\mathbf{t} = -\mathbf{r}, \tag{8.2.7}$$

is computed, where $\tilde{\mathbf{z}} = (\kappa_0 A + \kappa_1 B)\tilde{\mathbf{q}}$, and $\mathbf{r} = (I - Z_k Z_k^*)(A - \tilde{\lambda}B)(I - Q_k Q_k^*)\tilde{\mathbf{q}}$. The test space W is expanded with the vector $w = (\kappa_0 A + \kappa_1 B)\mathbf{t}$.

Observe that complex arithmetic needs to be used to compute approximations to solutions of equation (8.2.7) whenever $\tilde{\lambda}$ has a nonzero imaginary part, or whenever the matrices Q_k and Z_k contain entries with nonzero imaginary part.

When the JD correction equation (8.2.7) is solved exactly in each step of the JDQZ method, then it can be shown that the method converges asymptotically quadratically to an eigenvector. Solving the correction equation exactly, however, can be expensive. It may be better for the overall performance of the JDQZ method to solve the correction equation (8.2.7) only approximately. See [51] for an effective stopping criterion and also for the use of a preconditioner.

If $B = I$ the generalized eigenvalue problem (8.2.1) reduces to the standard eigenvalue problem $A\mathbf{x} = \lambda\mathbf{x}$, that can be solved by the JDQR method [51], a simplification of the JDQZ method. For the standard eigenvalue problem the eigenvalues of the projected eigenvalue problem (8.2.6) are called (harmonic) Ritz values, instead of (harmonic) Petrov values [51].

8.3 A new JDQZ Method for Real Matrix Pencils

In this section a Jacobi-Davidson style method that is specifically designed for *real unsymmetric* matrix pencils, is discussed.

A partial generalized real Schur form of the real matrix pencil (A, B) is a decomposition of the following form

$$AQ_k = Z_k S_k, \quad BQ_k = Z_k T_k,$$

where now Q_k and Z_k are *real* matrices with orthonormal columns, and S_k and T_k are *real* block upper triangular matrices with scalar or two by two diagonal blocks. The eigenvalues of the two by two diagonal blocks correspond to complex conjugate pairs of eigenvalues of the pencil (A, B).

8.3.1 The JDQRd and JDQZd Algorithms

In [25] an adaptation of the JDQR algorithm for the standard eigenvalue problem is proposed (in Algorithm 4.1) to compute a partial real Schur form. This algorithm is referred to as JDQRd in this chapter.

In order to explain the difference between the JDQR and the JDQRd method, it is convenient to look at the JDQR/JDQZ method as a 3-nested loop. The outer loop (level 1) accumulates the Schur vectors and in this loop the matrices Q_k, Z_k, S_k and T_k, which are needed for the partial Schur form (8.2.4), are constructed. The columns of Q_k and Z_k are computed as left and right eigenvectors of the deflated generalized eigenvalue problem (8.2.5). This problem is solved using the JD method, of which the iterations constitute the level 2-loop. The innermost loop (level 3) is the iterative solution of the correction equation (8.2.7).

The original JDQR/JDQZ methods use complex arithmetic in all three of the levels. The difference between the JDQR and JDQRd method is that the JDQRd method constructs a real partial Schur form by accumulating real vectors in the outermost loop. Thus level 1 in the JDQRd method uses real arithmetic, while levels 2 and 3 use complex arithmetic. This means that the JDQRd algorithm proceeds just as the JDQR method, except that when a complex eigenvalue λ_k, with corresponding eigenvector $\mathbf{q}_k$, is computed, then the partial Schur form is kept real by augmenting it with a real basis of the space spanned by $\mathbf{q}_k$ and $\bar{\mathbf{q}}_k$. In JDQRd, like in JDQR, the JD search space V in level 2 does not have to be real, and, therefore, the Ritz values do not have to appear in complex conjugate pairs. This causes difficulties for the identification of complex pairs of eigenvalues of the original eigenvalue problem, see e.g. Chapter 8 in [132], and it also introduces additional rounding errors when an computed approximate eigenvalue with small imaginary part is replaced by its real part. In Section 8.6.3, also the JDQZd method is considered, which is the QZ version of the JDQRd method.

In the next section, the RJDQZ method is introduced. This method uses real arithmetic in in both levels 1 and 2 and uses real arithmetic in level 3 when possible.

8.3.2 The RJDQZ Algorithm

A more thorough and robust way to compute a partial generalized real Schur form is to keep the JD search and test space in level 2 real. Suppose one has already a real search space V and a real test space W. Then one has to compute the eigenvalues of the projected generalized eigenvalue problem (8.2.2):

$$W^T A V \mathbf{u} = \lambda W^T B V \mathbf{u}.$$

Since the projected matrices W^TAV and W^TBV are real, the eigenvalues are either real or form a complex conjugate pair, and since the projected eigenvalue problem is small, all eigenvalues can be computed accurately and cheaply. From these eigenvalues one eigenvalue (or complex conjugate pair) is selected with a given selection criterion. Denote the selected Petrov value by $\tilde{\lambda}$ and the corresponding eigenvector by $\tilde{\mathbf{u}}$.

In the actual RJDQZ algorithm, instead of an eigenvalue decomposition of the projected eigenvalue problem (8.2.2), a sorted real generalized Schur form is computed. How this form is computed in an efficient and stable way can be found in [9, 23, 25] for the standard eigenvalue problem and in [76, 77, 41] for the generalized eigenvalue problem. As mentioned above, it is in the construction of the sorted real generalized Schur form where it is decided (up to machine precision) whether an eigenvalue is real or appears in a complex conjugate pair.

For a Petrov pair $(\tilde{\lambda}, \tilde{\mathbf{q}} = V\tilde{\mathbf{u}})$, the JDQZ correction equation (8.2.7) is of the following form:

$$(I - ZZ^*)(A - \theta B)(I - QQ^*)\mathbf{t} = -\mathbf{r}, \tag{8.3.1}$$

with $Q = [Q_k, \tilde{\mathbf{q}}]$ and $Z = [Z_k, \tilde{\mathbf{z}}]$. If the selected Petrov value $\tilde{\lambda}$ is real then the matrix and the right hand side in the correction equation (8.3.1) are both real, and an approximate solution can be computed using real arithmetic. Hence the search and test space can be expanded with a real vector.

If the selected Petrov value λ has nonzero imaginary part, then the JD correction equation will have to be solved using complex arithmetic, and one will obtain a complex approximate solution $\mathbf{t}$. In order to keep the search space real, it is expanded with the two dimensional real space $U = \text{span}\{\text{Re}(\mathbf{t}), \text{Im}(\mathbf{t})\}$, which contains the vector $\mathbf{t}$.

Remark 1: It is irrelevant whether a Petrov value $\tilde{\lambda}$ or $\tilde{\lambda}^*$ is selected, since in both cases the search space is expanded with the same space U. This also solves a problem from which the original JD method suffers when the target value is real (see Section 8.6.2 for more details).

Remark 2: Note that if the target value τ has a nonzero imaginary part then κ_1 (see discussion after equation (8.2.3)) will not be real in the harmonic Petrov approach. Thus in the case of a non-real target value combined with the harmonic Petrov approach, the proposed method loses most of its advantages, although keeping the search space real by expanding it with a two dimensional real space, when appropriate, might still accelerate the convergence of the JD method. Note that in this case also other iterative eigenvalue solvers such as the shift and invert Arnoldi method will need complex arithmetic [113].

Considering a complex target for the RJDQZ method may seem to contradict the goal of the method, but the method only aims to exploit the realness of the pencil. The test pencil MHD416 (see Section 8.6.4) is an example of a real pencil for which a complex target is wanted.

8.4 Formulating and solving the correction equation

If a real Petrov pair is selected, the correction equation can be solved using real arithmetic. If a complex Petrov pair is selected, there are three ways to formulate the correction equation for the real variant of Jacobi-Davidson QZ: (1) the correction equation can be formulated as a complex equation (the usual way), (2) the complex correction equation can be made real by defining two coupled equations for the real and imaginary part, or (3) a generalized real Sylvester equation can be formulated for the correction of the approximate two dimensional invariant subspace. In the following it will be shown that these three formulations are equivalent and that approach (3) is the preferred approach from a conceptual point of view, while approach (1) is more efficient in practice.

8.4.1 Complex correction equation

This is the usual formulation of the correction equation as in (8.3.1), solved using complex arithmetic. To keep the JD search space V real, it is expanded with the two dimensional real space $\mathrm{span}(\mathrm{Re}(\mathbf{t}), \mathrm{Im}(\mathbf{t}))$.

8.4.2 Real variant of complex correction equation

Let $\theta = \nu + i\omega$, $\mathbf{t} = \mathbf{u} + i\mathbf{v}$ and $\mathbf{r} = \mathbf{r}_1 + i\mathbf{r}_2$. Then it follows that

$$(A - \theta B)\mathbf{t} = -\mathbf{r} \Longleftrightarrow \begin{bmatrix} A - \nu B & \omega B \\ -\omega B & A - \nu B \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}.$$

The matrices with already converged right and left Schur vectors Q_k and Z_k are real. Let $\mathbf{q} = \mathbf{q}_1 + i\mathbf{q}_2$ with $\mathbf{q}_1^*\mathbf{q}_1 + \mathbf{q}_2^*\mathbf{q}_2 = 1$. Then

$$\begin{aligned} (I - \mathbf{q}\mathbf{q}^*)t &= (I - (\mathbf{q}_1\mathbf{q}_1^* + \mathbf{q}_2\mathbf{q}_2^*))\mathbf{u} + (\mathbf{q}_2\mathbf{q}_1^* - \mathbf{q}_1\mathbf{q}_2^*)\mathbf{v} \\ &+ i((I - (\mathbf{q}_1\mathbf{q}_1^* + \mathbf{q}_2\mathbf{q}_2^*))\mathbf{v} - (\mathbf{q}_2\mathbf{q}_1^* - \mathbf{q}_1\mathbf{q}_2^*)\mathbf{u}). \end{aligned}$$

The equivalent real block formulation becomes

$$(I - \mathbf{q}\mathbf{q}^*)\mathbf{t} \Longleftrightarrow P_q \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \equiv \begin{bmatrix} I - (\mathbf{q}_1\mathbf{q}_1^* + \mathbf{q}_2\mathbf{q}_2^*) & \mathbf{q}_2\mathbf{q}_1^* - \mathbf{q}_1\mathbf{q}_2^* \\ -(\mathbf{q}_2\mathbf{q}_1^* - \mathbf{q}_1\mathbf{q}_2^*) & I - (\mathbf{q}_1\mathbf{q}_1^* + \mathbf{q}_2\mathbf{q}_2^*) \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}.$$

By using $\mathbf{q}_1^*\mathbf{q}_1 + \mathbf{q}_2^*\mathbf{q}_2 = 1$ and some basic linear algebra, it can be shown that P_q is indeed a projector. In a similar way, P_z for $\mathbf{z} = \mathbf{z}_1 + i\mathbf{z}_2$ and $\mathbf{z}_1^*\mathbf{z}_1 + \mathbf{z}_2^*\mathbf{z}_2 = 1$ can be defined as

$$P_z \equiv \begin{bmatrix} I - (\mathbf{z}_1\mathbf{z}_1^* + \mathbf{z}_2\mathbf{z}_2^*) & \mathbf{z}_2\mathbf{z}_1^* - \mathbf{z}_1\mathbf{z}_2^* \\ -(\mathbf{z}_2\mathbf{z}_1^* - \mathbf{z}_1\mathbf{z}_2^*) & I - (\mathbf{z}_1\mathbf{z}_1^* + \mathbf{z}_2\mathbf{z}_2^*) \end{bmatrix}.$$

The real equivalent of the correction equation (8.3.1) becomes

$$P_z\mathcal{Z}_k \begin{bmatrix} A - \nu B & \omega B \\ -\omega B & A - \nu B \end{bmatrix} \mathcal{Q}_k P_q \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}, \tag{8.4.1}$$

where

$$\mathcal{Z}_k = \begin{bmatrix} I - Z_k Z_k^* & 0 \\ 0 & I - Z_k Z_k^* \end{bmatrix} \quad \text{and} \quad \mathcal{Q}_k = \begin{bmatrix} I - Q_k Q_k^* & 0 \\ 0 & I - Q_k Q_k^* \end{bmatrix}.$$

8.4.3 Real generalized Sylvester equation

An advantage of the RJDQZ algorithm is that approximations to complex conjugate pairs appear in conjugate pairs. The corresponding residual for the approximate two dimensional invariant subspace $\begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix}$ is

$$\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 \end{bmatrix} = A \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix} - B \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix} \begin{bmatrix} \nu & \omega \\ -\omega & \nu \end{bmatrix}.$$

The correction equation for $\begin{bmatrix} \mathbf{u} & \mathbf{v} \end{bmatrix}$ becomes a real generalized Sylvester equation

$$(I - ZZ^*)(A(I - QQ^*) \begin{bmatrix} \mathbf{u} & \mathbf{v} \end{bmatrix} - B(I - QQ^*) \begin{bmatrix} \mathbf{u} & \mathbf{v} \end{bmatrix} \begin{bmatrix} \nu & \omega \\ -\omega & \nu \end{bmatrix}) = - \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 \end{bmatrix}$$

The equivalent block formulation becomes

$$P_z \mathcal{Z}_k \begin{bmatrix} A - \nu B & \omega B \\ -\omega B & A - \nu B \end{bmatrix} \mathcal{Q}_k P_q \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}, \tag{8.4.2}$$

which is the same as the one obtained in (8.4.1).

An alternative formulation of the correction equation can be obtained if one considers a sorted real generalized Schur form instead of an eigenvalue decomposition (see also [51]). The selected approximate Schur quartet for the deflated problem is $(\begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 \end{bmatrix}, S, T)$, with $S, T \in \mathbb{R}^{2\times 2}$, T upper triangular, $\begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix} \perp Q_k$ and $\begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 \end{bmatrix} \perp Z_k$. The residual is computed as

$$\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 \end{bmatrix} = (I - Z_k Z_k^*)(A \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix} S^{-1} - B \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix} T^{-1}),$$

and the correction equation becomes

$$P_z \mathcal{Z}_k \begin{bmatrix} s_{11}^{-1} A - t_{11}^{-1} B & s_{21}^{-1} A \\ s_{12}^{-1} A - t_{12}^{-1} B & s_{22}^{-1} A - t_{22}^{-1} B \end{bmatrix} \mathcal{Q}_k P_q \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}. \tag{8.4.3}$$

8.4.4 Approximate solution of the correction equation

The real and imaginary part of the exact solution of (8.3.1) and the exact solutions of (8.4.1) and (8.4.2), and (8.4.3) span the same two dimensional real subspace. In practice however, the correction equation is only solved approximately using an iterative linear solver like GMRES. The rate of convergence of linear solvers depends, among others, on the condition number of the operator, the distribution of the eigenvalues, and the quality of the preconditioner. The following proposition states that the eigenvalues of the complex matrix $A - \tilde{\lambda} B$ in equation (8.3.1) are also eigenvalues of the equivalent real block matrix in equations (8.4.1) and (8.4.2), together with their complex conjugates and furthermore that the condition numbers of the matrices are the same.

Proposition 8.4.1. *Let $A, B \in \mathbb{R}^{n\times n}$, $\tilde{\lambda} = \nu + i\omega \in \mathbb{C}$ and $(A-\theta B)\mathbf{x}_j = \lambda_j \mathbf{x}_j, j = 1,\ldots,n$ with $\mathbf{x}_j \in \mathbb{C}^n$ and $\lambda_j \in \mathbb{C}$. Then the eigenpairs of*

$$C = \begin{bmatrix} A-\nu B & \omega B \\ -\omega B & A-\nu B \end{bmatrix} \in \mathbb{R}^{2n\times 2n} \tag{8.4.4}$$

are $(\lambda_j, \left[\mathbf{x}_j^T, (-i\mathbf{x}_j)^T\right]^T)$, $(\bar{\lambda_j}, \left[\mathbf{x}_j^T, (-i\mathbf{x}_j)^T\right]^*)$ for $j = 1,\ldots,n$.
Furthermore Cond(C)=Cond(A − θB).

Proof. For an eigenpair $(\lambda, \mathbf{x})$ of $A - \theta B$ it holds that

$$(A-\nu B)\mathbf{x} - \omega B i\mathbf{x} = \lambda\mathbf{x}, \qquad -(A-\nu B)i\mathbf{x} - \omega B\mathbf{x} = -\lambda i\mathbf{x}.$$

Using this it easily follows that

$$\begin{bmatrix} A-\nu B & \omega B \\ -\omega B & A-\nu B \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ -i\mathbf{x} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{x} \\ -i\mathbf{x} \end{bmatrix}.$$

The first part of the proposition follows by noting that because matrix (8.4.4) is a real matrix, its eigenpairs appear in complex conjugate pairs. The equality of the condition numbers for C and $A - \theta B$ follows from the fact that for every $\mathbf{z} = \mathbf{v} + i\mathbf{w} \in \mathbb{C}^n$ it holds that

$$\frac{\mathbf{z}^*(A-\theta B)^*(A-\theta B)\mathbf{z}}{\mathbf{z}^*\mathbf{z}} = \frac{\boldsymbol{v}^T C^T C \boldsymbol{v}}{\boldsymbol{v}^T\boldsymbol{v}},$$

where $\boldsymbol{v} = (\mathbf{v}^T\ \mathbf{w}^T)^T$. □

Using similar arguments, this relation can be extended to the operator in equation (8.4.3). From proposition 8.4.1 it follows that no big differences in convergence are to be expected if the approximate solution is computed with a linear solver. This is also confirmed by numerical experiments.

If a preconditioner $K \approx A - \tau B$ is available for a target $\tau \in \mathbb{R}$, it can be used for the block systems as well:

$$\tilde{K} = \begin{bmatrix} K & 0 \\ 0 & K \end{bmatrix}.$$

Using proposition 8.4.1, the condition numbers of $K^{-1}(A-\theta B)$ and $\tilde{K}^{-1}C$ are the same. So the use of a preconditioner also is not expected to cause big differences in speed convergence between the three approaches.

Numerical experiments showed that solving the complex correction equation was the most efficient with respect to the convergence of both the Krylov solver and the JD algorithm. Part of the explanation of why this is the case can be obtained by looking at the computational complexity of the different versions of the correction equations, which is the topic of the next section.

Table 8.1: Costs of operator application for the three approaches. BLAS terms are used: MV(A) is $A\mathbf{x}$, AXPY = $\alpha\mathbf{x} + \mathbf{y}$. The solve of $\mathbf{y}$ from $LU\mathbf{y} = \mathbf{x}$, given LU-factors, is denoted by LU.

Approach	MV(A)	MV(B)	AXPY	MV(Z)	LU
complex (8.3.1)	2	2	4	4	2
real (8.4.1)	2	2	4	4	2
Sylvester (8.4.3)	2	2	4	4	2

8.4.5 Complexity and practical notes

Table 8.1 shows the costs of an application of the operator in equations (8.3.1), (8.4.1) and (8.4.3). Operations are counted in purely real operations: if $\mathbf{x} \in \mathbb{C}^n$, then an MV ($A\mathbf{x}$) costs two real MVs. Furthermore, operations are counted for $n \times n$ matrices and $n \times 1$ vectors, because in a practical situation the $2n \times 2n$ systems are never constructed explicitly.

Operator applications cost the same for all three approaches[1]. The approach in (8.4.3) is the most elegant approach because no complex arithmetic is involved at all for the RJDQZ algorithm. If the correction equation is solved exactly using the LU decomposition, it is more efficient to solve the complex correction equation: the solve of complex linear system of order n costs half the solve of a real linear system of order $2n$. If an iterative solver like GMRES is used, there is convergence within atmost n iterations for the complex correction equations, while the real variants requires at most $2n$ iterations. Therefore, in the RJDQR/RJDQZ implementations used in Sections 8.6.2 and 8.6.3, the complex correction equation is solved. At first sight this may seem to contradict the goal of RJDQZ. Note, however, that the goal of the RJDQZ method is not to avoid all complex arithmetic. The goal of RJDQZ is to exploit the realness of the pencil. This means that the methods aims to approximate real eigenvalues with real Petrov values and complex conjugate pairs of eigenvalues with complex conjugate pairs of Petrov pairs. In approximating real eigenvalues as much real arithmetic as possible is used, and in approximating complex conjugate pairs of eigenvalues complex arithmetic is used where it is needed.

8.5 RJDQZ versus JDQZ

For a precise account of the computational and memory costs of the JDQR and JDQZ methods, see [51]. In this section only the main differences in the costs between the JDQZ(d) and the RJDQZ method are mentioned.

[1] It must be noted that straightforward application of the operator in (8.4.3) costs an additional 3 SCALs ($\alpha\mathbf{x}$). Clever implementation saves these 3 SCALs.

8.5.1 Differences in Memory Costs

The orthonormal bases for the search and test space in the JDQZ method are expanded with one complex vector in each iteration. For the RJDQZ, the bases of the search and test space are expanded with one real vector if the selected Petrov value is real or with two real vectors if the selected Petrov value appears in a complex conjugate pair. This means that, although the dimension of the search and test space for the RJDQZ method can grow twice as fast as for the JDQZ method, the storage requirements are the same at most, and probably less. The numerical experiments give evidence that the larger subspace in the RJDQZ method is beneficial for the convergence.

8.5.2 Differences in Computational Costs

- The correction equation: When in the RJDQZ method a real Petrov value is selected, the correction equation can be solved in real arithmetic. In the original JDQZ method, the chance that a real Petrov value is selected, even in the process of approximating a real eigenvalue of the original real eigenvalue problem, is negligible. This is caused by the fact that all the projected eigenvalue problems in the JDQZ method are in principle complex, and due to rounding errors, all the Petrov values will be complex, also the Petrov values that are approximating real eigenvalues of the original real eigenvalue problem. These latter Petrov values will of course have small imaginary parts, but will be complex nonetheless. This means that when in a RJDQZ iteration a real Petrov value is selected, approximately halve the number of (real) matrix-vector products is needed (depending on how many iterations are used for the approximate solution of the correction equation), when compared to a JDQZ iteration. When in a RJDQZ iteration a Petrov value is selected that appears in a complex conjugate pair, then the JDQZ and the RJDQZ method need the same work for the approximate solution of the correction equation. Note that the RJDQZ method requires two implementations of the solver for the correction equation: a real and a complex version.

- The projected eigenproblem: In the RJDQZ method the real Schur forms of real projected eigenproblems are computed, but these may be twice as large as the complex projected eigenproblems that appear in the JDQZ method. Assume that computing a Schur form costs $\mathcal{O}(k^3)$ operations [64] and that an operation in complex arithmetic costs in average four operations in real arithmetic. Then it is easily deduced that computing the Schur form of a real eigenvalue problem costs about twice as much as computing the Schur form of a complex eigenvalue problem that is twice as small.

- Orthogonalization: The other point where the RJDQZ method may be computationally more expensive than the original JDQZ, is the orthogonalization procedures. One has to compare these two cases:

  – Orthogonalize a complex vector $\mathbf{x}$ against k other complex vectors. This

requires $4k$ real inner products for the projection plus 2 real inner products for the scaling.

– Orthogonalize two real vectors $\mathbf{a}$ and $\mathbf{b}$ against $2k$ other real vectors. This requires $2k$ inner products for projecting the first vector $\mathbf{a}$ plus one inner product for scaling. For the next vector $\mathbf{b}$, $2k+1$ inner products are needed for the projection plus one for the scaling. This adds up to $4k+3$ inner products.

In the worst case, one iteration of the RJDQZ method will cost about the work of one inner product and one (low dimensional) Schur decomposition more than an iteration of the original version of the JDQZ method. If the initial approximation is a real vector, then the cost of the extra inner product in the RJDQZ method is eliminated, and the orthogonalization process in the RJDQZ method will cost at most as much as in the JDQZ method.

8.6 Numerical Comparison

In this section RJDQZ method is compared with the JDQZ and the JDQZd method, using numerical experiments.

8.6.1 Hardware and software

The experiments were performed on a Sunblade 100 workstation using Matlab 6. The Matlab code that was used for the RJDQZ method is given in the appendix. The cpu-time results are included. Note that these cpu-time results do not always give very accurate information, but they at least give an impression of how the methods perform in comparison to each other. If not mentioned otherwise, the target value in each experiment equals 0, and the tolerance is set to 10^{-9}.

For all the results presented in this section, the correction equation is solved approximately using at most 10 iterations of the Bi-CGSTAB method [155] or the GMRES method [133]. No restart strategy is used in the JD part of the algorithm, in order to focus on the differences in the performance of the methods caused by the different strategies for expanding the search and test space.

8.6.2 Results for the Real JDQR Method

The ideas presented in this chapter are easily incorporated in a QR version of the algorithm for the standard eigenvalue problem. In this section, numerical results for the QR version are reported.

The first test matrix is CRY10000 from the NEP collection [1]. This is a large real unsymmetric standard eigenvalue problem with dimension 10000 by 10000. The target value $\tau = 7$ is selected (this corresponds to the rightmost eigenvalues). Six eigenvalues closest to the target are computed. The convergence tolerance is set to 10^{-8}. The preconditioner that is used for the correction equations is the

Table 8.2: Results for QR methods.

	Bi-CGSTAB			GMRES		
CRY10000	JDQR	JDQRd	RJDQR	JDQR	JDQRd	RJDQR
iterations	41		37	71		52
dim. search sp.	36		38	66		63
mat.vec.	1432		751	1532		733
Cpu (secs)	246		116	386		158
AF23560	JDQR	JDQRd	RJDQR	JDQR	JDQRd	RJDQR
iterations	77	70	49	46	37	32
dim. search sp.	71	65	78	40	32	42
mat.vec.	3002	2708	1603	1622	1244	782
Cpu (secs)	4071	4255	2825	2303	1686	1107
CC100	JDQR	JDQRd	RJDQR	JDQR	JDQRd	RJDQR
iterations	37	29	32	57	47	45
dim. search sp.	32	25	39	52	43	57
mat.vec.	1214	878	883	1226	1006	657
Cpu (secs)	5	3	2	17	10	3

incomplete LU (ILU) factorization of the matrix $A - \tau I$ (where A is the CRY10000 matrix) with drop tolerance 10^{-3}. In Table 8.2 the number of iterations, the number of matrix-vector products, and the dimension of the final search space is given. For the JDQR method, the dimension of the search space should be multiplied by two to obtain the number of *real* vectors that are needed to store the basis of the search space. Note that the search space dimension is smaller than the number of iterations for the JDQR method (and also for the JDQRd in later experiments) because every time a Ritz value has converged, its eigenvector is deflated from the search space.

The computed eigenvalues are all real. The selected Ritz values in the intermediate JD iterations need not be, and in fact were not, all real. This explains why both the RJDQR and JDQR constructed a search space of approximately the same dimension, while the RJDQR method required fewer iterations to build this search space. Note that if only real eigenvalues are computed, the JDQR and JDQRd method coincide. Therefore the results for the JDQRd method are not given for the CRY10000 matrix.

The second test problem is the matrix AF23560 [1]. The order of the test matrix is 23560. The preconditioner that is used for the correction equations is the ILU factorization of the AF23560 matrix with drop tolerance 10^{-3}. Seven eigenvalues with largest real part are computed. Three of the computed eigenvalues are real and the other four are formed by two complex conjugate pairs. Observe that again the RJDQR method needs fewer iterations but builds a subspace that is larger than the JDQR method. Note that, although the dimension of the search space for the RJDQR method is larger, it needs far less storage than the JDQR method to store the basis, since it only stores real vectors.

The test matrix CC100 is constructed for the purpose of comparing the JDQR and the RJDQR method in the case that only complex conjugate pairs of eigenvalues are computed. The matrix has order 100, has the numbers -1 to -100 on the

```
-1    1    0    0    0    0    0    0    0    0
-1   -2    1    0    0    0    0    0    0    0
 0    0   -3    1    0    0    0    0    0    0
 0    0   -1   -4    1    0    0    0    0    0
 0    0    0    0   -5    1    0    0    0    0
 0    0    0    0   -1   -6    0    0    0    0
 0    0    0    0    0    0   -7    0    0    0
 0    0    0    0    0    0    0   -8    0    0
 0    0    0    0    0    0    0    0   -9    0
 0    0    0    0    0    0    0    0    0  -10
```

Figure 8.1: The upper left 10 by 10 block of the matrix CC100.

diagonal, and has some off-diagonal elements only in the upper left corner. The upper left 10 by 10 corner is given below in Fig. 8.1. The six eigenvalues with largest real part are computed. No preconditioner is used for the correction equations.

If only complex pairs of eigenvalues are computed, the RJDQR method may need fewer matrix-vector products than the JDQR method. There can be two different reasons: the intermediate selected Ritz values can be real so that real arithmetic can be used in intermediate RJDQR iterations, and secondly, in case the selected Ritz value is not real, then the dimension of the search space grows faster in the RJDQR method, which may result in a faster convergence and thus fewer matrix-vector products.

Ritz value selection

Two complex conjugate eigenvalues have equal distance to a real target τ. In the JDQR method, the approximating Ritz values, however, do not necessarily have the same distance to the target. Therefore it may happen that in one iteration the Ritz value with positive imaginary part is selected and in the next iteration the Ritz value with negative imaginary part. This may hamper the convergence of the method, as illustrated in Fig. 8.2. The dotted line in the left graph shows the convergence of the JDQR method to an eigenvalue with nonzero imaginary part. In the right graph the imaginary part of the selected Ritz value is plotted versus the iteration number. When the sign of the maginary part of the selected Ritz value changes the residual becomes larger. In order to prevent the sign of the imaginary part of the selected Ritz value from changing, the JDQR method can be adapted such that only Ritz values with non-negative imaginary part are selected. The convergence of this adapted method, depicted with dashed lines, is faster. The RJDQR method (solid line) selects complex conjugate Ritz pairs instead and has the fastest convergence, most likely due to the larger search space.

8.6.3 Results for the Real JDQZ Method

The first test for the QZ methods is the generalized eigenvalue problem BFW782 [1]. The matrix A is non-symmetric and B is symmetric positive definite. Six eigenvalues with largest real part are computed. The preconditioner that is used for the correction equations is the ILU factorization of A with drop tolerance 10^{-3}. In Table 8.3 the number of iterations, the number of matrix-vector products, and

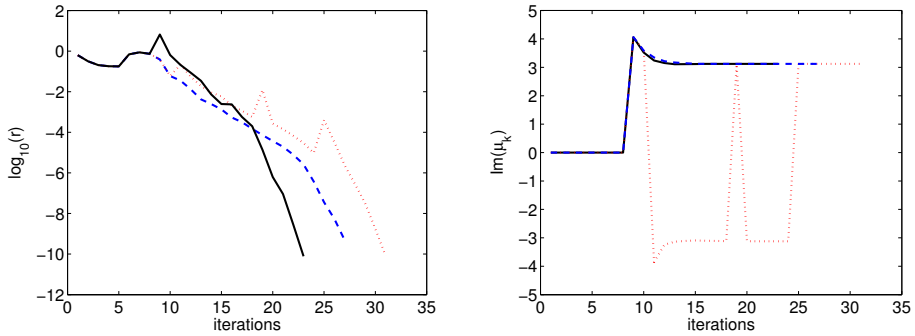


Figure 8.2: Results for matrix CC100. Left graph: the residual for the JDQR, adapted JDQR and RJDQR method versus the number of iterations for the convergence to an eigenvalue with nonzero imaginary part. Right graph: the imaginary part versus the number of iterations. Dotted lines: JDQR, dashed lines: adapted JDQR, solid lines: RJDQR.

the dimension of the final search space is given.

The computed eigenvalues are all real. Observe that the JDQZ and the RJDQZ method both need exactly the same number of iterations and build a search space of the same dimension. From the intermediate iterations (not shown), it can be concluded that the two methods perform exactly the same steps, the only difference being that the RJDQZ method performs the steps in real arithmetic and the JDQZ method performs the steps using complex arithmetic. This explains why the JDQZ method needs twice as many matrix-vector products as the RJDQZ method.

The generalized eigenvalue problem QG6468 arises in the stability analysis of steady states in the finite difference approximation of the QG model described in [40]. The eigenvalues and corresponding eigenvectors of interest are the ones with largest real parts. The matrix A is non-symmetric and B is symmetric positive semi definite. The ten eigenvalues with largest real part are computed. The preconditioner that is used for the correction equations is the ILU factorization of A with drop tolerance 10^{-7}.

For this matrix pencil, two of the computed eigenvalues are real, and the other computed eigenvalues form four complex conjugate pairs. One sees that the RJDQZ method needs fewer iterations, but builds a larger search space than the JDQZ method. The storage requirements for the RJDQZ method are, however, still less than for the JDQZ method.

For the problem ODEP400 [1], the eigenvalues and corresponding eigenvectors with largest real part are wanted. The matrix A is non-symmetric and B is symmetric positive semi definite. The twelve eigenvalues with largest real part are computed. These eigenalues appear in six complex conjugate pairs. The preconditioner that is used for the correction equations is the LU factorization of A.

For this matrix pencil, a similar conclusion as for the CC100 matrix can be drawn. The computed eigenvalues are all complex, but still the RJDQZ method

Table 8.3: Results for QZ methods.

	Bi-CGSTAB			GMRES		
BFW782	JDQZ	JDQZd	RJDQZ	JDQZ	JDQZd	RJDQZ
iterations	23		23	28		28
dim. search sp.	18		18	23		23
mat.vec.	558		279	456		228
Cpu (secs)	13		8	13		8
QG6468	JDQZ	JDQZd	RJDQZ	JDQZ	JDQZd	RJDQZ
iterations	78	59	47	97	67	53
dim. search sp.	69	51	76	88	59	87
mat.vec.	3176	2378	1744	2072	1412	1009
Cpu (secs)	1154	800	548	1090	640	412
ODEP400	JDQZ	JDQZd	RJDQZ	JDQZ	JDQZd	RJDQZ
iterations	33	32	22	31	26	21
dim. search sp.	22	22	24	20	16	29
mat.vec.	1202	1180	570	462	386	243
Cpu (secs)	13	14	4	8	6	3

needs fewer iterations and fewer matrix-vector products than the JDQZ method.

8.6.4 Non-real target

As mentioned in Remark 2, if the target value τ is non-real, the harmonic Petrov approach loses most of its advantages in the RJDQR and RJDQZ methods: when the search space V is expanded with the real vector x, in the harmonic Petrov approach, the test space should be expanded with the vector $w = (\kappa_0 A + \kappa_1 B)x$, where $\kappa_0 = (1+|\tau|^2)^{-1/2}$ and $\kappa_1 = -\tau(1+|\tau|^2)^{-1/2}$. In the case that τ is non-real, κ_1 will also be non-real, and thus the test space has to be expanded with a non-real vector. This spoils the realness of the projected eigenvalue problem and thus the idea behind the RJD methods breaks down.

One could of course abandon the harmonic approach, and choose κ_0 and κ_1 to be real. In this section two such alternatives to the harmonic approach are studied using numerical experiments. For the experiments, the pencil MHD416 [1] is used as an example. The six eigenvalues closest to the target value $\tau = 0.7i$ are computed. As preconditioner the LU factorization of $A-0.7iB$ is used. The results of the experiments in Table 8.4 show that the RJDQZ method is more effective than the JDQZd method.

The results for the JDQZ method are not shown in Table 8.4. The reason is the following. In order to obtain the same set of eigenvalues and eigenvectors as computed using the JDQZd and RJDQZ method, it would be required to run the JDQZ method twice, first with target $\tau = 0.7i$ and then with target $\tau = -0.7i$, since the JDQZ method does not automatically compute complex conjugate eigenvalues and vectors in pairs. This would result in an unfair comparison. One could of course add the complex conjugate of a computed eigenvalue and eigenvector when it is computed to the constructed Schur form, but this would in fact constitute a version of the JDQZd method.

Table 8.4: Results for JDQZd and RJDQZ for the MHD416 pencil.

	Bi-CGSTAB		GMRES	
Harm. Petr.	JDQZd	RJDQZ	JDQZd	RJDQZ
iterations	42		50	
dim. search sp.	32		40	
mat.vec.	1724		1080	
$\kappa_0 = 1$, $\kappa_1 = 0$	JDQZd	RJDQZ	JDQZd	RJDQZ
iterations	50	28	50	41
dim. search sp.	40	46	40	72
mat.vec.	2060	1136	1080	882
$\kappa_0 = 0$, $\kappa_1 = 1$	JDQZd	RJDQZ	JDQZd	RJDQZ
iterations	45	27	55	40
dim. search sp.	35	44	45	70
mat.vec.	1850	1094	1190	860

8.7 Conclusions

In this chapter, an adapted version of the Jacobi-Davidson method is presented, intended for real unsymmetric matrices or pencils.

In all the presented numerical experiments, the RJDQR and RJDQZ variants needed fewer iterations, fewer matrix-vector products and less storage than the original JD methods. The difference is most pronounced for the cases where only real eigenvalues are computed. The better performance of the RJD methods can be attributed to two reasons: first, the method uses real arithmetic where possible, which results in fewer matrix-vector products, and second, the dimension of the search space may grow twice as fast, while it does not use more storage. The limited additional costs for orthogonalization are compensated by far by the faster convergence due to the larger search space.

8.8 Appendix: Matlab-style code for RJDQZ method

In Algorithms 8.1-8.3, the Matlab-style code is presented for the RJDQZ method. The code for the approximate solution of the correction equation is not given. Note that the correction equation can be solved using real arithmetic if the approximate eigenvalue (α, β) is real. Otherwise it must be solved using complex arithmetic.

The routine "realqzsort" computes the ordered generalized real Schur form of the pencil (M_A, M_B), with respect to the target value τ. The logical output variable "complex" should be true if the harmonic Petrov value closest to the target has a nonzero imaginary part, and false if not. The routine "mgs" is a modified Gram-Schmidt routine as described in the appendix of [51].

Addendum

Except for some changes in notation, this chapter is published as [158]

> Tycho van Noorden and Joost Rommes, *Computing a partial generalized real Schur form using the Jacobi-Davidson method*, Numerical Linear Algebra with Applications **14** (2007), no. 3, 197–215.

Algorithm 8.1 The RJDQZ method

function $[Q, Z, R_A, R_B]$ =RJDQZ$(A, B, K, \tau, v_0, \epsilon, k_{max}, j_{min}, j_{max})$
$Q = [\,]$; $Z = [\,]$; $R_A = [\,]$; $Y = [\,]$; $H = [\,]$;
$V = [\,]$; $V_A = [\,]$; $V_B = [\,]$; $W = [\,]$; $M_A = [\,]$; $M_B = [\,]$;
$\gamma = \sqrt{1 + |\tau|^2}$; $\alpha_0 = \tau/\gamma$; $\beta_0 = 1/\gamma$; $k = 0$; $j = 0$;
while $k < k_{max}$
    if $j == 0$, then $v = v_0$, else solve correction equation for v
    Expand (Alg. 8.2)
    $[U_L, U_R, S_A, S_B, \text{complex}] = \text{realqzsort}(\tau, M_A, M_B)$;
    $j = j + 1$; found= 1;
    while found
        if complex
            $\alpha = S_A(1:2, 1:2)$; $\beta = S_B(1:2, 1:2)$;
            $q = VU_R(:, 1:2)$; $z = WU_L(:, 1:2)$;
            $r_A = V_A U_R(:, 1:2)$; $[r_A, s_A] = \text{mgs}(Z, r_A)$;
            $r_B = V_B U_R(:, 1:2)$; $[r_B, s_B] = \text{mgs}(Z, r_B)$;
            $r = r_A/\alpha - r_B/\beta$;
            found= $(||r|| < \epsilon)$ and $(j > 1 | k == k_{max} - 1)$;
            if $||r|| > \epsilon$
                $[U_L, U_R, S_A, S_B] = \text{qz}(\alpha, \beta)$;
                $\alpha = S_A(1,1)$; $\beta = S_B(1,1)$;
                $q = qU_R(:, 1)$; $z = zU_L(:, 1)$;
                $r_A = r_A U_R(:, 1)$; $[r_A, s_A] = \text{mgs}(Z, r_A)$;
                $r_B = r_B U_R(:, 1)$; $[r_B, s_B] = \text{mgs}(Z, r_B)$;
                $r = r_A/\alpha - r_B/\beta$;
            end
            $y = K^{-1}z$; $\tilde{\mathbf{x}} = [Q, q]$; $\tilde{Y} = [Y, y]$; $\tilde{\mathbf{z}} = [Z, z]$;
            $\tilde{H} = [H, Q^*y; q^*Y, q^*y]$;
        else
            $\alpha = S_A(1,1)$; $\beta = S_B(1,1)$; $q = VU_R(:, 1)$;
            $z = WU_L(:, 1)$; $y = K^{-1}z$;
            $r = V_A U_R(:, 1)$; $[r, s_A] = \text{mgs}(Z, r)$;
            $r_B = V_B U_R(:, 1)$; $[r_B, s_B] = \text{mgs}(Z, r_B)$;
            $r = r/\alpha - r_B/\beta$;
            found= $(||r|| < \epsilon)$ and $(j > 1 | k == k_{max} - 1)$;
            $\tilde{\mathbf{x}} = [Q, q]$; $\tilde{Y} = [Y, y]$; $\tilde{\mathbf{z}} = [Z, z]$;
            $\tilde{H} = [H, Q^*y; q^*Y, q^*y]$;
        end
        Found and restart (Alg. 8.3)
    end
end

Algorithm 8.2 RJDQZ: Expand

if isreal(v)
  $v = \text{mgs}(V, v)$; $v = v/||v||$; $v_A = Av$; $v_B = Bv$;
  $w = (\beta_0 v_A - \alpha_0 v_B)$; $w = \text{mgs}(Z, w)$;
  $w = \text{mgs}(W, w)$; $w = w/||w||$;
  $M_A = [M_A, W^* v_A; w^* V_A, w^* v_A]$;
  $M_B = [M_B, W^* v_B; w^* V_B, w^* v_B]$;
  $V = [V, v]$; $V_A = [V_A, v_A]$; $V_B = [V_B, v_B]$; $W = [W, w]$;
else
  $\tilde{v} = [\text{real}(v), \text{imag}(v)]$;
  for $i = 1 : 2$
    $v = \tilde{v}(:, i)$; $v = \text{mgs}(V, v)$; $v = v/||v||$; $v_A = Av$; $v_B = Bv$;
    $w = (\beta_0 v_A - \alpha_0 v_B)$; $w = \text{mgs}(Z, w)$;
    $w = \text{mgs}(W, w)$; $w = w/||w||$;
    $M_A = [M_A, W^* v_A; w^* V_A, w^* v_A]$;
    $M_B = [M_B, W^* v_B; w^* V_B, w^* v_B]$;
    $V = [V, v]$; $V_A = [V_A, v_A]$; $V_B = [V_B, v_B]$; $W = [W, w]$;
  end
end

Algorithm 8.3 RJDQZ: Found and restart

if found
  $Q = \tilde{\mathbf{x}}$; $Z = \tilde{\mathbf{z}}$;
  $R_A = [R_A, s_A; \text{zeros}(1 + \text{complex}, k), \alpha]$;
  $R_B = [R_B, s_B; \text{zeros}(1 + \text{complex}, k), \beta]$;
  $k = k + 1 + \text{complex}$; if $k \geq k_{max}$, break; end
  $Y = \tilde{Y}$; $H = \tilde{H}$;
  $J = [2 + \text{complex} : j]$; $j = j - 1 - \text{complex}$;
  $V = VU_R(:, J)$; $V_A = V_A U_R(:, J)$; $V_B = V_B U_R(:, J)$;
  $W = WU_L(:, J)$; $S_A = S_A(J, J)$; $S_B = S_B(J, J)$;
  $M_A = S_A$; $M_B = S_B$; $U_R = I$; $U_L = I$;
  $[U_L, U_R, S_A, S_B, \text{complex}] = \text{realqzsort}(\tau, M_A, M_B)$;
elseif $j \geq j_{max}$
  $j = j_{min}$; $J = [1 : j]$;
  $V = VU_R(:, J)$; $V_A = V_A U_R(:, J)$; $V_B = V_B U_R(:, J)$;
  $W = WU_L(:, J)$; $S_A = S_A(J, J)$; $S_B = S_B(J, J)$;
  $M_A = S_A$; $M_B = S_B$; $U_R = I$; $U_L = I$;
end

Bibliography

[1] *The Matrix Market*, http://math.nist.gov/MatrixMarket.

[2] *Oberwolfach Model Reduction Benchmark Collection*, http://www.imtek.uni-freiburg.de/simulation/benchmark/.

[3] P. A. Absil, R. Sepulchre, P. Van Dooren, and R. Mahony, *Cubically convergent iterations for invariant subspace computation*, SIAM J. Matrix Anal. Appl. **26** (2004), no. 1, 70–96.

[4] L. A. Aguirre, *Quantitative Measure of Modal Dominance for Continuous Systems*, Proc. of the 32nd Conference on Decision and Control, December 1993, pp. 2405–2410.

[5] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, SIAM, 2005.

[6] A. C. Antoulas and D. C. Sorensen, *Approximation of large-scale dynamical systems: an overview*, Int. J. Appl. Math. Comput. Sci. **11** (2001), no. 5, 1093–1121.

[7] W. E. Arnoldi, *The principle of minimized iteration in the solution of the matrix eigenproblem*, Quart. Appl. Math. **9** (1951), no. 1, 17–29.

[8] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst (eds.), *Templates for the Solution of Algebraic Eigenvalue Problems: a Practical Guide*, SIAM, 2000.

[9] Z. Bai and J. W. Demmel, *On swapping diagonal blocks in real Schur form*, Lin. Alg. Appl. **186** (1993), 73–95.

[10] Z. Bai, G. L. G. Sleijpen, and H. A. van der Vorst, *Quadratic eigenvalue problems*, Templates for the Solution of Algebraic Eigenvalue Problems: a Practical Guide (Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds.), SIAM, 2000, pp. 281–289.

[11] Z. Bai and Y. Su, *Dimension reduction of large-scale second-order dynamical systems via a second-order arnoldi method*, SIAM J. Sc. Comp. **26** (2005), no. 5, 1692–1709.

[12] ———, *SOAR: a second-order Arnoldi method for the solution of the quadratic eigenvalue problem*, SIAM J. Matrix Anal. Appl. **26** (2005), no. 3, 640–659.

[13] Z. Bai and Q. Ye, *Error estimation of the Padé approximation of transfer functions via the Lanczos process*, Electronic Transactions on Numerical Analysis **7** (1998), no. 1, 1–17.

[14] G. A. Baker Jr., *Essentials of Padé approximations*, Academic Press, 1975.

[15] S. Batterson and J. Smillie, *The dynamics of Rayleigh quotient iteration*, SIAM J. Num. Anal. **26** (1989), no. 3, 624–636.

[16] C. Beattie and D. W. Fox, *Localization criteria and containment for Rayleigh quotient iteration*, SIAM J. Matrix Anal. Appl. **10** (1989), no. 1, 80–93.

[17] P. R. Bélanger, *Control Engineering: A Modern Approach*, Oxford University Press, 1994.

[18] P. Benner, *Numerical linear algebra for model reduction in control and simulation*, GAMM Mitteilungen **29** (2006), no. 2, 275–296.

[19] P. Benner, V. Mehrmann, and D. Sorensen (eds.), *Dimension Reduction of Large-Scale Systems*, Lecture Notes in Computational Science and Engineering, vol. 45, Springer, 2005.

[20] L. H. Bezerra, *Written discussion to [91]*, IEEE Trans. Power Syst. **11** (1996), no. 1, 168.

[21] ———, *Eigenvalue computation via Newton methods*, TEMA Tend. Mat. Apl. Comput. **5** (2004), no. 1, 37–47, (in portuguese).

[22] D. Bindel, J. W. Demmel, and M. J. Friedman, *Continuation of invariant subspaces for large and sparse bifurcations problems*, online proceedings of SIAM Conference on Applied Linear Algebra, Williamsburg, 2003, http://www.siam.org/meetings/la03/proceedings/.

[23] A. Bojanczyk and P. Van Dooren, *Reordering diagonal blocks in real Schur form*, Linear Algebra for Large-Scale and Real-Time Applications (M. Moonen, G. H. Golub, and B. De Moor, eds.), Kluwer Academic Publishers, 1993.

[24] A. L. B. Bonfim, G. N. Taranto, and D. M. Falcão, *Simultaneous tuning of power system damping controllers using genetic algorithms*, IEEE Trans. Power Syst. **15** (2000), no. 1, 163–169.

[25] J. H. Brandts, *Matlab code for sorting real Schur forms*, Num. Lin. Alg. Appl. **9** (2002), no. 3, 249–261.

[26] ———, *The Riccati algorithm for eigenvalues and invariant subspaces of matrices with inexpensive action*, Lin. Alg. Appl. **358** (2003), 335–365, Special issue on accurate solution of eigenvalue problems (Hagen, 2000).

[27] J. L. Casti, *Dynamical systems and their applications: Linear theory*, 1 ed., Mathematics in Science and Engineering, vol. 135, Academic Press, 1977.

[28] Y. Chahlaoui and P. van Dooren, *A collection of Benchmark examples for model reduction of linear time invariant dynamical systems*, SLICOT Working Note 2002-2, 2002.

[29] D. Chaniotis and M. A. Pai, *Model reduction in power systems using Krylov subspace methods*, IEEE Trans. Power Syst. **20** (2005), no. 2, 888–894.

[30] E. Chiprout and M. S. Nakhla, *Asymptotic Waveform Evaluation and Moment Matching for Interconnect Analysis*, Kluwer Academic Publishers, 1994.

[31] J. H. Chow and J. J. Sanchez-Gasca, *Power system damping controller design using multiple input signals*, IEEE Control Syst. Mag. **20** (2000), no. 4, 82–90.

[32] CIGRE TF 38.02.16, *Impact of the interactions among power system controls*, Tech. Report 166, CIGRE, July 2000.

[33] K. A. Cliffe, T. J. Garratt, and A. Spence, *Eigenvalues of the discretized Navier-Stokes equation with application to the detection of Hopf bifurcations*, Adv. Math. Comp. **1** (1993), 337–356.

[34] ———, *Eigenvalues of block matrices arising from problems in fluid mechanics*, SIAM J. Matrix Anal. Appl. **15** (1994), no. 4, 1310–1318.

[35] K. A. Cliffe, A. Spence, and S. J. Tavener, *The numerical analysis of bifurcation problems with application to fluid mechanics*, Acta Numerica **9** (2000), 39–131.

[36] J. W. Daniel, W. B. Gragg, L. Kaufmann, and G. W. Stewart, *Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization*, Math. Comp. **30** (1976), no. 136, 772–795.

[37] E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real symmetric matrices*, J. Comp. Phys. **17** (1975), no. 1, 87–94.

[38] E. Davison, *A method for simplifying linear dynamic systems*, IEEE Trans. Aut. Control **11** (1966), no. 1, 93–101.

[39] E. J. Davison and S. H. Wang, *Properties and calculation of transmission zeros of linear multivariable systems*, Automatica **10** (1974), no. 6, 643–658.

[40] H. A. Dijkstra and C. A. Katsman, *Temporal variability of the wind-driven quasi-geostrophic double gyre ocean circulation: basic bifurcation diagrams*, Geophysical and Astrophysical Fluid Dynamics **85** (1997), 195–232.

[41] P. Van Dooren, *A generalized eigenvalue approach for solving Riccati equations*, SIAM J. Sci. Stat. Comput. **2** (1981), no. 2, 121–135.

[42] S. C. Eisenstat, *Efficient implementation of a class of preconditioned conjugate gradient methods*, SIAM J. Sci. Statist. Comput **2** (1981), no. 1, 1–4.

[43] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers*, first ed., Oxford University Press, 2005.

[44] A. Emami-Naeni and P. Van Dooren, *Computation of zeros of linear multivariable systems*, Automatica **18** (1982), no. 4, 415–430.

[45] D. F. Enns, *Model reduction with balanced realizations: An error bound and a frequency weighted generalization*, Proceedings of 23rd Conference on Decision and Control, 1984, pp. 127–132.

[46] T. Ericsson, *A generalised eigenvalue problem and the Lanczos algorithm*, Large Scale Eigenvalue Problems (J. Cullum and R.A. Willoughby, eds.), Elservier Science Publishers BV, 1986, pp. 95–119.

[47] T. Ericsson and A. Ruhe, *The spectral transformation lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems*, Math. Comp. **35** (1980), 1251–1268.

[48] P. Feldmann and R. W. Freund, *Efficient linear circuit analysis by Padé approximation via the Lanczos process*, IEEE Trans. CAD **14** (1995), 639–649.

[49] _____, *Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm*, roceedings of the 32nd Design Automation Conference, June 1995, pp. 474–479.

[50] D. R. Fokkema, *Subspace Methods for Linear, Nonlinear, and Eigen Problems*, Ph.D. thesis, Utrecht University, 1996.

[51] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst, *Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils*, SIAM J. Sc. Comp. **20** (1998), no. 1, 94–125.

[52] J. G. F. Francis, *The QR transformation: a unitary analogue to the LR transformation, parts i and ii*, Computer Journal **4** (1961), 265–272, 332–345.

[53] R. W. Freund, *Krylov-subspace methods for reduced-order modeling in circuit simulation*, J. Comp.Appl. Math. **123** (2000), no. 1-2, 395–421.

[54] _____, *Model reduction methods based on Krylov subspaces*, Acta Numerica **12** (2003), 267–319.

[55] _____, *SPRIM: Structure-preserving reduced-order interconnect macromodeling*, Technical Digest of the 2004 IEEE/ACM International Conference on CAD, 2004, pp. 80–87.

[56] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comp. **14** (1993), no. 1, 137–158.

[57] K. Gallivan, E. Grimme, and P. Van Dooren, *Asymptotic waveform evaluation via a lanczos method*, Appl. Math. Lett. **7** (1994), 75–80.

[58] ———, *A rational Lanczos algorithm for model reduction*, Numerical Algorithms **12** (1996), no. 1, 33–63.

[59] K. Gallivan, P. Van Dooren, and A. Vandendorpe, *Model reduction of MIMO systems via tangential interpolation*, SIAM J. Matrix Anal. Appl. **26** (2004), no. 2, 328–349.

[60] C. Gama, L. Ängquist, G. Ingeström, and M. Noroozian, *Commissioning and operative experience of TCSC for damping power oscillation in the brazilian north-south interconnection*, Proc. CIGRÉ Session No. 38, August 2000, Paper 14-104.

[61] T. J. Garratt, *The numerical detection of Hopf bifurcations in large systems arising in fluid mechanics*, Ph.D. thesis, University of Bath, 1991.

[62] K. Glover, *All optimal Hankel-norm approximations of linear multivariable systems and their L^∞-error bounds*, Int. J. Control **39** (1984), 1115–1193.

[63] G. H. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM J. Num. Anal. **2** (1965), no. 2, 205–224.

[64] G. H. Golub and C. F. van Loan, *Matrix Computations*, third ed., John Hopkins University Press, 1996.

[65] S. Gomes and N. Martins, *Computing multiple dominant poles for the modal analysis of s-domain models*, In preparation.

[66] S. Gomes Jr, N. Martins, S. L. Varricchio, and C. Portela, *Modal analysis of electromagnetic transients in ac networks having long transmission lines*, IEEE Trans. Power Delivery **30** (2005), no. 4, 2623–2630.

[67] W. B. Gragg and A. Lindquist, *On the partial realization problem*, Lin. Alg. Appl. **50** (1983), 277–319.

[68] M. Green and D. J. N. Limebeer, *Linear Robust Control*, Prentice-Hall, 1995.

[69] P. M. Gresho, D. K. Gartling, J. R. Torczynski, K. A. Cliffe, K. H. Winters, T. J. Garratt, A. Spence, and J. W. Goodrich, *Is the steady viscous incompressible two-dimensional flow over a backward-facing step at Re=800 stable?*, Int. J. Num. Meth.Fluids **17** (1993), 501–541.

[70] E. J. Grimme, *Krylov projection methods for model reduction*, Ph.D. thesis, University of Illinois, 1997.

[71] S. Gugercin and A. C. Antoulas, *Model reduction of large-scale systems by least squares*, Lin. Alg. Appl. **415** (2006), no. 2–3, 290–321.

[72] A. M. A. Hamdan and A. H. Nayfeh, *Measures of modal controllability and observability for first- and second-order linear systsems*, J. Guid. Contr. Dyn. **12** (1989), no. 3, 421–428.

[73] P. J. Heres, *Robust and efficient Krylov subspace methods for model order reduction*, Ph.D. thesis, Eindhoven University of Technology, 2005.

[74] M. E. Hochstenbach and G. L. G. Sleijpen, *Two-sided and alternating Jacobi-Davidson*, Lin. Alg. Appl. **358** (2003), no. 1-3, 145–172.

[75] M. E. Hochstenbach and H. A. van der Vorst, *Alternatives to the Rayleigh quotient for the quadratic eigenvalue problem*, SIAM J. Sc. Comp. **25** (2003), no. 2, 591–603.

[76] B. Kågström, *A direct method for reordering eigenvalues in the generalized real Schur form of a regular matrix pair* (A, B), Linear Algebra for Large-Scale and Real-Time Applications (M. Moonen, G. H. Golub, and B. De Moor, eds.), Kluwer Academic Publishers, 1993.

[77] B. Kågström and P. Poromaa, *Computing eigenspaces with specified eigenvalues of a regular matrix pair* (A, B) *and condition estimation: Theory, algorithms and software*, Tech. Report UMINF 94.04, Institute for Information Processing, University of Umeå, Umeå, Sweden, 1994, LAPACK Working Note 87.

[78] T. Kailath, *Linear Systems*, Prentice-Hall, 1980.

[79] I. Kaufman, *On poles and zeros of linear systems*, IEEE Trans. Circ. Theory **CT-20** (1973), no. 2, 93–101.

[80] A. V. Knyazev, *Preconditioned eigensolvers*, Templates for the Solution of Algebraic Eigenvalue Problems: a Practical Guide (Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds.), SIAM, 2000, pp. 337–368.

[81] _____, *Hard and soft locking in iterative methods for symmetric eigenvalue problems*, Copper Mountain Conference, 2004, http://math.cudenver.edu/ aknyazev/research/conf/cm04.htm.

[82] J. Lampe and H. Voss, *Second order Arnoldi reduction: application to some engineering problems*, Report 93, Hamburg University of Technology, 2006.

[83] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Natl. Bur. Stand. **45** (1950), no. 4, 225–280.

[84] R. B. Lehoucq and J. A. Scott, *Implicitly restarted Arnoldi methods and eigenvalues of the discretized Navier Stokes equations*, Tech. Report SAND97-2712J, Sandia National Laboratory, 1997.

[85] R. B. Lehoucq and D. C. Sorensen, *Deflation techniques within an implicitly restarted Arnoldi iteration*, SIAM J. Matrix Anal. Appl. **17** (1996), 789–821.

[86] R. H. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK SOFTWARE*, http://www.caam.rice.edu/software/ARPACK/.

[87] J.-R. Li and J. White, *Low rank solution of Lyapunov equations*, SIAM J. Matrix Anal. Appl. **24** (2002), no. 1, 260–280.

[88] J. Lienemann, D. Billger, E. B. Rudnyi, A. Greiner, and J. G. Korvink, *MEMS compact modeling meets model order reduction: examples of the application of Arnoldi method to microsystem devices*, Technical Proceedings of the 2004 Nanotechnology Conference and Trade Show, 2004.

[89] J. M. Maciejowski, *Multivariable Feedback Design*, Addison-Wesley, 1989.

[90] N. Martins, *The dominant pole spectrum eigensolver*, IEEE Trans. Power Syst. **12** (1997), no. 1, 245–254.

[91] N. Martins, L. T. G. Lima, and H. J. C. P. Pinto, *Computing dominant poles of power system transfer functions*, IEEE Trans. Power Syst. **11** (1996), no. 1, 162–170.

[92] N. Martins, H. J. C. P. Pinto, and L. T. G. Lima, *Efficient methods for finding transfer function zeros of power systems*, IEEE Trans. Power Syst. **7** (1992), no. 3, 1350–1361.

[93] N. Martins and P. E. M. Quintão, *Computing dominant poles of power system multivariable transfer functions*, IEEE Trans. Power Syst. **18** (2003), no. 1, 152–159.

[94] N. Martins, F. G. Silva, and P. C. Pellanda, *Utilizing transfer function modal equivalents of low-order for the design of power oscillation damping controllers in large power systems*, IEEE/PES General Meeting, June 2005, Enlarged version submitted to IEEE Trans. Power Syst., pp. 2642–2648.

[95] K. Meerbergen, *Locking and restarting quadratic eigenvalue solvers*, SIAM J. Sc. Comp. **22** (2000), no. 5, 1814–1839.

[96] K. Meerbergen and D. Roose, *The restarted Arnoldi method applied to iterative linear system solvers for the computation of rightmost eigenvalues*, SIAM J. Matrix Anal. Appl. **18** (1997), no. 1, 1–20.

[97] K. Meerbergen and A. Spence, *Implicitly restarted Arnoldi with purification for the shift-invert transformation*, Math. Comp. **66** (1997), no. 218, 667–689.

[98] K. Meerbergen, A. Spence, and D. Roose, *Shift-invert and Cayley transforms for detection of rightmost eigenvalues of nonsymmetric matrices*, BIT **34** (1994), no. 3, 409–423.

[99] V. Mehrmann and H. Voss, *Nonlinear eigenvalue problems: a challenge for modern eigenvalue methods*, Mitt. Gesell. Ang. Math. Mech. **27** (2005), 121–151.

[100] P. Misra, P. Van Dooren, and A. Varga, *Computation of structural invariants of generalized state-space systems*, Automatica **30** (1994), no. 12, 1921–1936.

[101] C. B. Moler and G. W. Stewart, *An algorithm for generalized matrix eigenvalue problems*, SIAM J. Num. Anal. **10** (1973), 241–256.

[102] B. C. Moore, *Principal component analysis in linear systems: Controllability, observability and model reduction*, IEEE Trans. Aut. Control **26** (1981), no. 1, 17–32.

[103] D. V. Murthy and R. T. Haftka, *Derivatives of eigenvalues and eigenvectors of a general complex matrix*, Int. J. Num. Meth. Eng. **26** (1988), 293–311.

[104] B. Nour-Omid, B. N. Parlett, T. Ericsson, and P. S. Jensen, *How to implement the spectral transformation*, Math. Comp. **48** (1987), no. 178, 663–673.

[105] A. Odabasioglu and M. Celik, *PRIMA: Passive Reduced-order Interconnect Macromodeling Algorithm*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **17** (1998), no. 8, 645–654.

[106] A. M. Ostrowski, *On the convergence of the Rayleigh quotient iteration for the computation of the characteristic roots and vectors. I*, Arch. Rational Mech. Anal. **1** (1958), 233–241.

[107] _____, *On the convergence of the Rayleigh quotient iteration for the computation of the characteristic roots and vectors. III*, Arch. Rational Mech. Anal. **3** (1959), 325–340.

[108] B. C. Pal, A. H. Coonick, and B. J. Cory, *Robust damping of inter-area oscillations in power systems with superconducting magnetic energy storage devices*, Proc. IEE Generation, Transmission and Distribution **146** (1999), 633–639.

[109] R. D. Pantazis and D. B. Szyld, *Regions of convergence of the rayleigh quotient iteration method*, Num. Lin. Alg. Appl. **2** (1995), no. 3, 251–269.

[110] B. N. Parlett, *The Rayleigh quotient iteration and some generalizations for nonnormal matrices*, Math. Comp. **28** (1974), no. 127, 679–693.

[111] _____, *The Symmetric Eigenvalue Problem*, Prentice Hall, 1980.

[112] _____, *The Symmetric Eigenvalue Problem*, Classics in applied mathematics, SIAM, 1998.

[113] B. N. Parlett and Y. Saad, *Complex shift and invert strategies for real matrices*, Lin. Alg. Appl. **88/89** (1987), 575–595.

[114] R. V. Patel and N. Munro, *Multivariable System Theory and Design*, Pergamon, 1982.

[115] T. Penzl, *Algorithms for model reduction of large dynamical systems*, Lin. Alg. Appl. **415** (2006), no. 2-3, 322–343.

[116] G. Peters and J. H. Wilkinson, *Inverse iteration, ill-conditioned equations and Newton's method*, SIAM Review **21** (1979), no. 3, 339–360.

[117] L. T. Pillage and R. A. Rohrer, *Asymptotic Waveform Evaluation for timing analysis*, IEEE Trans. Comp. Aid. Des. Int. Circ. Syst. **9** (1990), no. 4, 352–366.

[118] J. W. Polderman and J. C. Willems, *Introduction to mathematical systems theory*, Texts in Applied Mathematics, vol. 26, Springer, 1998.

[119] A. Ramirez, A. Semlyen, and R. Iravani, *Order reduction of the dynamic model of a linear weakly periodic system-Part I: General methodology*, IEEE Trans. Power Syst. **19** (2004), no. 2, 857–865.

[120] J. Rommes, *Modal approximation and computation of dominant poles*, Model Order Reduction: Theory, Research Aspects and Applications (H. A. van der Vorst and W. H. A. Schilders, eds.), Mathematics in Industry, Springer, To Appear.

[121] ———, *Efficient preconditioning for Jacobi-Davidson methods and its use in purification for generalized eigenvalue problems*, Preprint 1339, Utrecht University, 2005, Revised 2007, Accepted for publication in Mathematics of Computation.

[122] J. Rommes and N. Martins, *Efficient computation of multivariable transfer function dominant poles using subspace acceleration*, IEEE Trans. Power Syst. **21** (2006), no. 4, 1471–1483.

[123] ———, *Efficient computation of transfer function dominant poles using subspace acceleration*, IEEE Trans. Power Syst. **21** (2006), no. 3, 1218–1226.

[124] ———, *Efficient computation of transfer function dominant poles of large second-order dynamical systems*, Preprint 1360, Utrecht University, 2007.

[125] J. Rommes, N. Martins, and P. C. Pellanda, *Efficient computation of large scale transfer function dominant zeros*, Preprint 1358, Utrecht University, 2006, Submitted.

[126] J. Rommes and G. L. G. Sleijpen, *Convergence of the dominant pole algorithm and Rayleigh quotient iteration*, Preprint 1356, Utrecht University, 2006, Submitted.

[127] H. H. Rosenbrock, *State-space and Multivariable Theory*, John Wiley and Sons, Inc., 1970.

[128] E. B. Rudnyi, J. Lienemann, A. Greiner, and J. G. Korvink, *mor4ansys: Generating compact models directly from ANSYS models*, Technical Proceedings of the 2004 Nanotechnology Conference and Trade Show, 2004.

[129] A. E. Ruehli, *Equivalent circuit models for three dimensional multi-conductor systems*, IEEE Trans. Microwave Theory Tech. **22** (1974), 216–221.

[130] A. Ruhe, *Algorithms for the nonlinear eigenvalue problem*, SIAM J. Num. Anal. **10** (1973), no. 4, 674–689.

[131] ———, *Rational Krylov: A practical algorithm for large sparse nonsymmetric matrix pencils*, SIAM J. Sc. Comp. **19** (1998), no. 5, 1535–1551.

[132] Y. Saad, *Numerical Methods for Large Eigenvalue Problems: Theory and Algorithms*, Manchester University Press, 1992.

[133] Y. Saad and M. H. Schultz, *GMRES: A generalized minimial residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. **7** (1986), no. 3, 856–869.

[134] J. J. Sanchez-Gasca and J. H. Chow, *Power system reduction to simplify the design of damping controllers for interarea oscillations*, IEEE Trans. Power Syst. **11** (1996), no. 3, 1342–1349.

[135] C. B. Schrader and M. K. Sain, *Research on system zeros: a survey*, Int. J. Control **50** (1989), no. 4, 1407–1433.

[136] L. M. Silveira, M. Kamon, and J. White, *Efficient reduced-order modeling of frequency-dependent coupling inductances associated with 3-D interconnect structures*, IEEE Trans. Comp. Hybr. Man. Tech., Part B:Adv. Pack. **19** (1996), no. 2, 283–288.

[137] L. M. Silverman, *Inversion of multivariable linear systems*, IEEE Trans. Aut. Control **AC-14** (1969), no. 3, 270–276.

[138] D. Silvester, H. Elman, and A. Ramage, *Incompressible Flow and Iterative Solver Software*, http://www.maths.manchester.ac.uk/ djs/ifiss/.

[139] G. L. G. Sleijpen, J. G. L. Booten, D. R. Fokkema, and H. A. van der Vorst, *Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems*, BIT **36** (1996), no. 3, 595–633.

[140] G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl. **17** (1996), no. 2, 401–425.

[141] G. L. G. Sleijpen, H. A. van der Vorst, and E. Meijerink, *Efficient expansion of subspaces in the Jacobi-Davidson method*, Electronic Transactions on Numerical Analysis **7** (1998), 75–89.

[142] G. L. G. Sleijpen, H. A. van der Vorst, and M. van Gijzen, *Quadratic eigenproblems are no problem*, SIAM News **29** (1996), no. 7, 8–9.

[143] P. Smit and M. H. C. Paardekooper, *The effects of inexact solvers in algorithms for symmetric eigenvalue problems*, Lin. Alg. Appl. **287** (1999), no. 1-3, 337–357.

[144] J. R. Smith, J. F. Hauer, D. J. Trudnowski, F. Fatehi, and C. S. Woods, *Transfer function identification in power system application*, IEEE Trans. Power Syst. **8** (1993), no. 3, 1282–1290.

[145] E. D. Sontag, *Mathematical control theory:deterministic finite dimensional systems*, Texts in Applied Mathematics, vol. 6, Springer, 1998.

[146] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl. **13** (1992), 357–385.

[147] ———, *Deflation for implicitly restarted Arnoldi methods*, Tech. Report TR98-12, Rice University, 1998.

[148] A. Stathopoulos, *A case for a biorthogonal Jacobi-Davidson method: Restarting and correction equation*, SIAM J. Matrix Anal. Appl. **24** (2002), no. 1, 238–259.

[149] T. Stykel, *Analysis and Numerical Solution of Generalized Lyapunov Equations*, Ph.D. thesis, Technischen Universität Berlin, 2002.

[150] D. Szyld, *Criteria for combining inverse and Rayleigh quotient iteration*, SIAM J. Num. Anal. **25** (1988), no. 6, 1369–1375.

[151] The Mathworks, Inc., *Matlab*.

[152] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Review **43** (2001), no. 2, 235–286.

[153] G. Troullinos, J. Dorsey, H. Wong, and J. Myers, *Reducing the order of very large power system models*, IEEE Trans. Power Syst. **3** (1988), no. 1, 127–133.

[154] J. van den Eshof, *The convergence of Jacobi-Davidson iteration for Hermitian eigenproblems*, Num. Lin. Alg. Appl. **9** (2002), 163–179.

[155] H. A. van der Vorst, *Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. **13** (1992), no. 2, 631–644.

[156] ———, *Computational Methods for Large Eigenvalue Problems*, Handbook of Numerical Analysis (P.G. Ciarlet and J.L. Lions, ed.), vol. VIII, North-Holland (Elsevier), 2001, pp. 3–179.

[157] ———, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, 2003.

[158] T. L. van Noorden and J. Rommes, *Computing a partial generalized real Schur form using the Jacobi-Davidson method*, Num. Lin. Alg. Appl. **14** (2007), no. 3, 197–215.

[159] A. Varga, *Enhanced modal approach for model reduction*, Math. Mod. Syst. (1995), no. 1, 91–105.

[160] S. L. Varricchio, N. Martins, and L. T. G. Lima, *A Newton-Raphson method based on eigenvalue sensitivities to improve harmonic voltage performance*, IEEE Trans. Power Delivery **18** (2003), no. 1, 334–342.

[161] M. E. Verbeek, *Partial element equivalent circuit (PEEC) models for on-chip passives and interconnects*, Int. J. Num. Mod.: Electronic Networks, Devices and Fields **17** (2004), no. 1, 61–84.

[162] E. L. Wachspress, *Iterative solution of the Lyapunov matrix equation*, Appl. Math. Lett. **107** (1988), no. 1, 87–90.

[163] H. F. Walker and L. T. Watson, *Least-change secant update methods for under-determined systems*, SIAM J. Numer. Math. **27** (1990), 1227–1262.

[164] K. Zhou and J. Doyle, *Essentials of Robust Control*, Prentice Hall, 1997.

Index

Samenvatting

Mobiele telefoons, laptops, digitale camera's en draagbare mp3-spelers zijn niet meer weg te denken uit de huidige samenleving, net zo min als het verlangen naar kleinere en snellere uitvoeringen ervan met steeds meer mogelijkheden. Gebouwen, bruggen en vliegtuigen worden juist steeds groter. Aan de productie van deze apparaten en constructies gaat vaak een proces van ontwerp, simulatie en analyse vooraf. Simulatie met wiskundige modellen van het eigenlijke systeem kan inzicht geven in het gedrag van het systeem en heeft het voordeel dat het goedkoop en veilig is. De resultaten van de simulatie kunnen gebruikt worden om te controleren of het ontwerp voldoet aan de specificaties en om het ontwerp aan te passen waar nodig.

Als gevolg van de groeiende vraag naar steeds kleinere componenten met meer functionaliteit, zoals computer chips, en steeds grotere constructies zoals vliegtuigen, zijn er grotere, ingewikkelde wiskundige modellen nodig om te kunnen voldoen aan de eisen aan nauwkeurigheid en detail. Alhoewel computers steeds sneller worden, is directe simulatie van de modellen vaak niet mogelijk omdat de complexiteit van de algoritmes kwadratisch of zelfs kubisch groeit met het aantal vrijheidsgraden (bijvoorbeeld het aantal transistoren op een chip) in het rekenmodel. Noodzakelijk zijn daarom modellen van sterk gereduceerde orde, die het gedrag van het oorspronkelijke model benaderen en de dominante eigenschappen behouden. Een voorbeeld van een dominante eigenschap is de resonantiefrequentie: voor een constructie als een brug is dit de frequentie waarop de brug de voorkeur heeft te trillen, voor een elektrische component als een versterker is dit de frequentie van signalen die versterkt worden.

Men kan model orde reductie zien als een toepassing van Ockhams scheermes. William of Ockham, een middeleeuwse filosoof, stelde dat er in de verklaring van een verschijnsel zo weinig mogelijk aannamen gemaakt moeten worden, en dat de voorkeur uitgaat naar de eenvoudigste verklaring. In model orde reductie wordt Ockhams scheermes gebruikt om de niet-dominante eigenschappen "af te scheren", om zo tot een kleiner (eenvoudiger) model te komen. Zoals beschreven in dit proefschrift, kan men hetzelfde doel bereiken door juist de dominante eigenschappen te berekenen.

In dit proefschrift worden methoden gepresenteerd die dominante eigenschappen van grootschalige systemen berekenen. In alle gevallen betreft het hier grote eigenwaardeproblemen waarbij slechts enkele specifieke eigenwaarden van belang

zijn. In stabiliteitsanalyse van dynamische systemen zijn de meest rechtse eigenwaarden de dominante eigenwaarden. Voor een algemeen lineair dynamisch systeem zijn de dominante eigenwaarden de polen die het meest bijdragen aan de bijbehorende overdrachtsfunctie.

In Hoofdstuk 2 wordt het convergentiegedrag van het Dominante Polen Algorithme (DPA) onderzocht en vergeleken met het convergentiegedrag van Rayleigh Quotient Iteratie (RQI). Deze algoritmes kunnen beschouwd worden als Newton-methoden voor de berekening van nulpunten (in dit geval eigenwaarden) van niet-lineaire vergelijkingen. Beide algoritmes werken iedere iteratie met een nieuwe schatting voor de eigenwaarde, maar waar RQI iedere iteratie het rechterlid van de lineare vergelijkingen bijwerkt, houdt DPA deze constant. Juist het constant houden van het rechterlid zorgt ervoor dat DPA meer dan RQI de neiging heeft om te convergeren naar eigenwaarden waarvan de bijbehorende eigenvectoren een kleine hoek met het rechterlid maken. Dit constant houden zorgt er wel voor dat DPA kwadratische convergentie heeft in de nabijheid van een oplossing, tegen kubische voor RQI, maar in de praktijk kost dit slechts één of twee extra iteraties. Vanwege het typische convergentiegedrag is DPA een zeer effectieve methode voor het uitrekenen van dominante polen van grootschalige overdrachtsfuncties.

In Hoofdstuk 3 wordt DPA uitgebreid met deelruimte versnelling en deflatie om de globale convergentie te verbeteren en meerdere polen te kunnen berekenen, zonder al eerder gevonden polen opnieuw te berekenen. Deflatie kan efficiënt gedaan worden vanwege het feit dat DPA werkt met vaste rechterleden. Onder zekere voorwaarden is deelruimte versneld DPA equivalent met tweezijdig Jacobi-Davidson, en als het niet haalbaar is om de lineaire systemen exact op te lossen, dan kan niet-exact tweezijdig Jacobi-Davidson gebruikt worden om de dominante polen te berekenen. De dominante polen en bijbehorende eigenvectoren kunnen ook gebruikt worden om modellen van gereduceerde orde te construeren en om Krylov-deelruimtemethoden te verbeteren.

In Hoofdstuk 4–6 worden varianten van DPA gepresenteerd. In Hoofdstuk 4 wordt DPA uitgebreid tot een algoritme voor de berekening van dominante polen van overdrachtsfuncties met meerdere inputs en outputs. Hoofdstuk 5 laat zien hoe DPA gebruikt kan worden om de dominante nulpunten van overdrachtsfuncties te berekenen. Een variant voor de berekening van dominante polen van overdrachtsfuncties van tweede orde dynamische systemen wordt beschreven in Hoofdstuk 6.

Hoofdstuk 7 behandelt de problemen die kunnen onstaan bij het berekenen van eigenwaarden van matrixparen met een singuliere matrix. Als gevolg van de singulariteit van een van de matrices zijn er oneindige eigenwaarden en deze hebben geen fysische betekenis. De berekening van de interessante grootste eigenwaarden wordt bemoeilijkt door de aanwezigheid van deze oneindige eigenwaarden. Standaard Arnoldi- en Jacobi-Davidson-methoden voor dit soort gegeneraliseerde eigenwaardeproblemen kunnen benaderingen voor oneindige eigenwaarden interpreteren als benaderingen van eindige eigenwaarden. In dit hoofdstuk worden diverse strategieën en technieken gepresenteerd die deze problemen voorkomen.

In Hoofstuk 8 wordt een variant van Jacobi-Davidson QZ besproken die werkt met reële zoekruimtes, wat leidt tot lagere kosten en snellere convergentie.

Acknowledgments

First of all I would like to thank my promotor Henk van der Vorst for his support during my research. He gave me much freedom and was always available for advise, inspiration, and fruitful ideas. His comments and suggestions, which he always brings across positively, helped me to improve both the content and presentation of my work. He made me aware of useful references and conferences, and gave me permission to implement some of my results at Philips Research Laboratories.

I am thankful to Gerard Sleijpen for the many constructive discussions that we had. He pointed me to the work of Nelson Martins and Karl Meerbergen, that inspired me in my research. He was always willing to review and discuss my manuscripts and his suggestions led to improvements, new insights, and motivation for additional research.

I thank Nelson Martins for our work together on the dominant pole algorithm and its variants. Although our communication consisted of many e-mails, one phone call, and one meeting in Manchester in March 2007, our cooperation was efficient and effective. His engineering expertise helped a lot to get clear what is really needed in practical applications and to improve the algorithms.

In the first two years of my research I had many helpful discussions with Tycho van Noorden, and I enjoyed our visit to the Copper Mountain conference and the work on our joint paper. I am also thankful to Jan ter Maten, Jan van Gerwen and Theo Beelen of NXP Semiconductors (formerly Philips Research Laboratories), for discussions on numerical issues in electric circuit simulation, the opportunity to implement some of my results, and the motivation and new insights the latter gave.

I would like to thank the members of the reading committee, prof.dr. Odo Diekmann, prof.dr. Paul van Dooren, prof.dr. Wil Schilders, dr. Gerard Sleijpen, and prof.dr. Arthur Veldman. Suggestions and comments of several anonymous referees helped me to improve the chapters of this thesis that have been appeared as separate papers.

Thanks to all colleagues at the Mathematical Institute and to my former roommates Johnny, Martin, Oliver and Jeroen, and to the members of the numerical analysis group in particular.

I thank my family and my friends for being there. Finally, I thank Eveline for her love and warmth.

Curriculum Vitae

Joost Rommes was born on May 29, 1979 in Utrecht, The Netherlands, where he also grew up. From 1991 to 1997 he attended the Christelijk Gymnasium in Utrecht. In 1997 he started to study Computational Science at Utrecht University. Under supervision of prof.dr. Henk A. van der Vorst and dr. E. Jan W. ter Maten he did an internship at Philips Research Laboratories, which resulted in his master's thesis 'Jacobi-Davidson methods and preconditioning with applications in pole-zero analysis' for which he graduated (cum laude) in June 2002. With the master's thesis 'Syntax macros: attribute redefinitions', under supervision of prof.dr. S. D. Swierstra, he graduated (cum laude) in Computer Science as well in August 2003.

During his studies he worked one day a week as a scientific programmer at the Freudenthal Institute from 1998 to 1999, and from 2000 to 2003 he worked two days a week as a programmer for Ordina.

In September 2003 he started as a PhD student at Utrecht University, under supervision of prof.dr. Henk A. van der Vorst. He visited and gave presentations at conferences, workshops and schools in Copper Mountain, Leuven, Eindhoven, Leiden, Amsterdam, Monopoli and Manchester. He also taught several undergraduate mathematics tutorials at Utrecht University. In 2005 and 2006 he combined his research activities temporarily with a job as programmer at Philips Research Laboratories, where he implemented some of the results of his research. His research on methods for eigenvalue problems and model order reduction has resulted in this thesis.