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

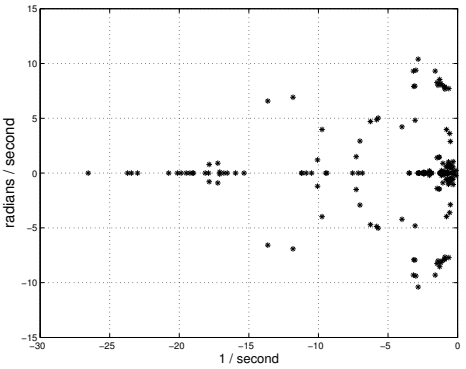

Figure 4.1: Pole spectrum of 184th order modal equivalent of the 8×8 transfer function.

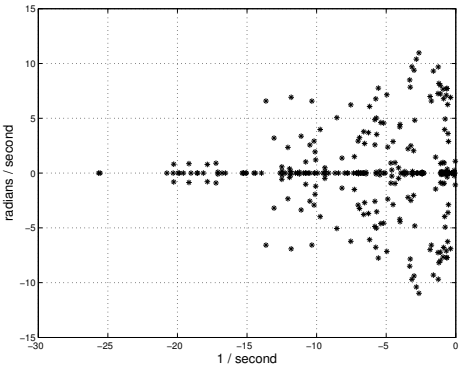

Figure 4.2: Pole spectrum of 291st order modal equivalent of the 28×28 transfer function.

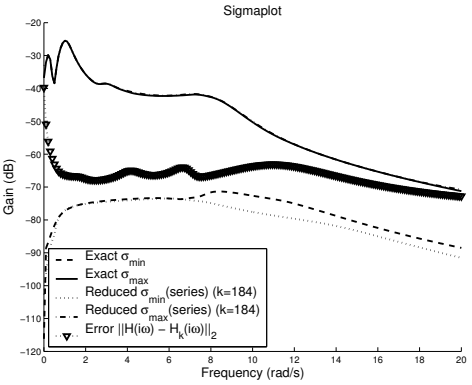

Figure 4.3: Sigma plot of modal equivalent and complete model, and error for the 8×8 Brazilian system transfer function (1,664 states in the complete model, 184 in the reduced model).

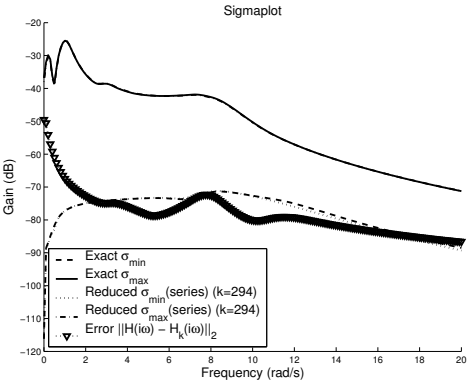

Figure 4.4: Sigma plot of modal equivalent and complete model, and error for the 8×8 Brazilian system transfer function (1,664 states in the complete model, 294 in the reduced model).

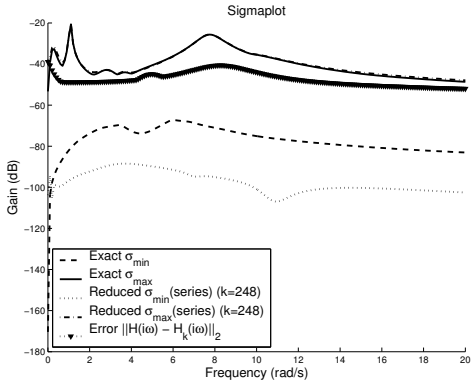

Figure 4.5: Sigma plot of modal equivalent and complete model, and error for the 28×28 Brazilian system transfer function (1,664 states in the complete model, 248 in the reduced model).

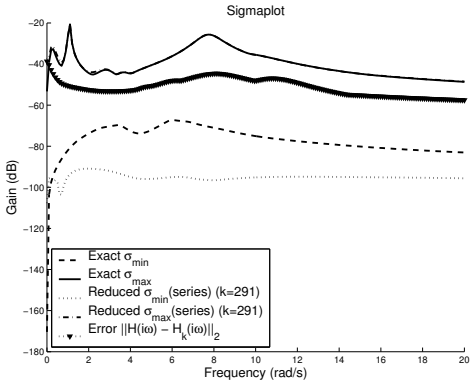

Figure 4.6: Sigma plot of modal equivalent and complete model, and error for the 28×28 Brazilian system transfer function (1,664 states in the complete model, 291 in the reduced model).

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 $\sigma_{\min}$-

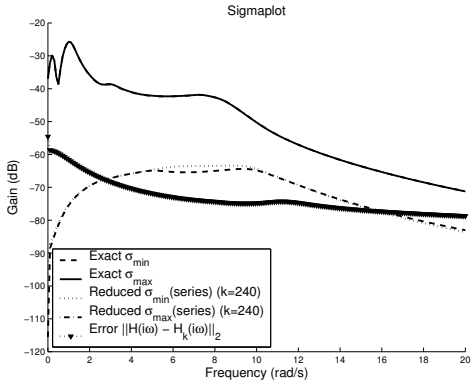

Figure 4.7: Sigma plot of modal equivalent and complete model, and error for the 8×6 Brazilian system transfer function (1,664 states in the complete model, 240 in the reduced model).

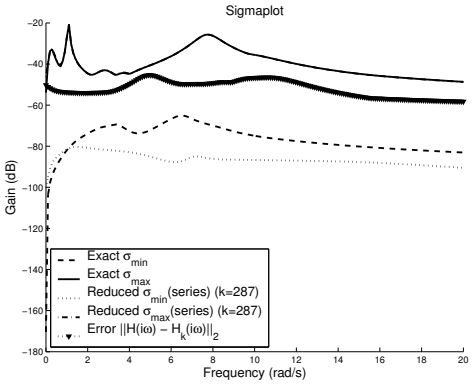

Figure 4.8: Sigma plot of modal equivalent and complete model, and error for the 28×25 Brazilian system transfer function (1,664 states in the complete model, 287 in the reduced model).

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.