# Contents

# 7

# Model Reduction Software in the SLICOT Library

**Andras Varga**

*ABSTRACT We describe the model reduction software developed recently for the control and systems library SLICOT. Besides a powerful collection of Fortran 77 routines implementing the last algorithmic developments for several well-known balancing related methods, we also describe model reduction tools developed to facilitate the usage of SLICOT routines in user friendly environments like* MATLAB *or* `Scilab`*. Extensive testing of the implemented tools has been done using both special benchmark problems as well as models of several complex industrial plants. Testing results and performance comparisons show the superiority of SLICOT model reduction tools over existing model reduction software.*

## 7.1 Introduction

Model reduction is of fundamental importance in many modeling and control applications. Still, reliable and high quality model reduction software tools are scarce. Even the model reduction tools available in commercial packages have strong limitations because using either inappropriate algorithms or poor software implementations. The lack of good general purpose model reduction software was the motivation to develop with the highest priority a model reduction chapter for the control and systems library SLICOT in the framework of the NICONET project[1] [3].

In this paper we describe the recently developed model reduction software for SLICOT. A powerful collection of user callable Fortran 77 routines has been implemented based on the latest algorithmic developments for three balancing related methods: the *balanced truncation approximation* (BTA) method [14], the *singular perturbation approximation* (SPA) method [11] and the *Hankel-norm approximation* (HNA) method [8]. These methods

[1] `http://www.win.tue.nl/niconet/niconet.html`

belong to the class of additive error methods and rely on guaranteed error bounds. They are primarily intended for the reduction of linear, stable, continuous- or discrete-time systems. However, in combination with additive spectral decomposition or coprime factorization techniques the basic methods can be used to reduce unstable systems too. Most of new model reduction tools and related software included in SLICOT originate from the collection of routines available in the RASP-MODRED library [20].

The new model reduction routines for SLICOT are among the most powerful and numerically most reliable software tools available for model reduction. To facilitate the usage of the new model reduction routines, easy-to-use and flexible interfaces have been developed to integrate them in two popular user friendly computing environments for engineering and scientific applications: the commercial package MATLAB[2] and the free software `Scilab` [5]. The basis for integration was a subset of the new model reduction routines which address the reduction of unstable systems. The same software can be also used without any computational overhead for the reduction of stable systems.

Several benchmark examples have been employed for testing and performance comparisons. Simple well behaving models have been primarily used to check the correct installation of the software. Besides them, several real industrial models exhibiting some challenging features like poor scaling, lack of minimality, presence of unstable modes, have been employed for performance and functional testing. The test results and performance comparisons show the superiority of SLICOT model reduction tools over existing model reduction software.

## 7.2 Development of model reduction subroutines

In this section we present the standardization effort done within the European project NICONET [3] to develop numerically reliable software for model reduction for the SLICOT library. After a short overview of balancing related model reduction methods underlying the implemented algorithms, we describe the modular approach used to implement the model reduction software for SLICOT. Important byproducts of the standardization activity for the model reduction software are many useful routines of more general use implemented in conjunction with or for the special need of model reduction software. These additional routines represent important extensions of existing chapters or constitute even completely new chapters of the SLICOT library, as for example, the routines for transfer-function matrix factorization, spectral decomposition, computation of system norms, or system similarity transformations.

[2] MATLAB is a registered trademark of The MathWorks, Inc.

### *7.2.1 Balancing related model reduction methods*

Three basic model reduction algorithms belonging to the class of methods based on or related to balancing techniques [14, 11, 8] form the basis of model reduction software in SLICOT. These methods are primarily intended for the reduction of linear, stable, continuous- or discrete-time systems. They rely on guaranteed error bounds and have particular features which recommend them for use in specific applications. In what follows we present succinctly the main features of balancing related model reduction.

Consider the $n$-th order original state-space model $G := (A, B, C, D)$ with the *transfer-function matrix* (TFM) $G(\lambda) = C(\lambda I - A)^{-1}B + D$, and let $G_r := (A_r, B_r, C_r, D_r)$ be an $r$-th order approximation of the original model ($r < n$), with the TFM $G_r = C_r(\lambda I - A_r)^{-1}B_r + D_r$. According to the system type, $\lambda$ is either the complex variable $s$ appearing in the Laplace transform in case of a continuous-time system or the variable $z$ appearing in the $Z$-transform in case of a discrete-time system. In our overview we focus on the so-called absolute (or additive) error model reduction method which essentially tries to minimize the absolute approximation error

$$\|G - G_r\|_\infty. \tag{7.1}$$

An important class of model reduction methods, including the popular BTA method, can be interpreted as performing a similarity transformation $Z$ yielding

$$\left[\begin{array}{c|c} Z^{-1}AZ & Z^{-1}B \\ \hline CZ & D \end{array}\right] := \left[\begin{array}{cc|c} A_{11} & A_{12} & B_1 \\ A_{21} & A_{22} & B_2 \\ \hline C_1 & C_2 & D \end{array}\right], \tag{7.2}$$

and then defining the reduced model $G_r$ as the leading diagonal system

$$(A_r, B_r, C_r, D_r) = (A_{11},\, B_1,\, C_1,\, D). \tag{7.3}$$

When writing $Z$ and $Z^{-1}$ in partitioned form

$$Z := [\, T\ U\,], \quad Z^{-1} := \left[\begin{array}{c} L \\ V \end{array}\right],$$

then $LT = I_r$ and $\Pi = TL$ is a projection matrix. Thus the reduced system is given by

$$(A_r, B_r, C_r, D_r) = (LAT,\, LB,\, CT,\, D).$$

The matrices $L$ and $T$ are called *truncation* matrices.

The partitioned system matrices in (7.2) can be used to construct a so-called *singular perturbation approximation* (SPA):

$$\begin{array}{rcl} A_r &=& A_{11} + A_{12}(\gamma I - A_{22})^{-1}A_{21}\ , \\ B_r &=& B_1 + A_{12}(\gamma I - A_{22})^{-1}B_2\ , \\ C_r &=& C_1 + C_2(\gamma I - A_{22})^{-1}A_{21}\ , \\ D_r &=& D + C_2(\gamma I - A_{22})^{-1}B_2\ , \end{array} \tag{7.4}$$

where $\gamma = 0$ for a continuous-time system and $\gamma = 1$ for a discrete-time system. Note that SPA formulas preserve the DC-gains of stable original systems.

Another class of so-called *modal approximation* methods, determine $Z$ such that

$$\left[\begin{array}{c|c} Z^{-1}AZ & Z^{-1}B \\ \hline CZ & D \end{array}\right] := \left[\begin{array}{cc|c} A_{11} & 0 & B_1 \\ 0 & A_{22} & B_2 \\ \hline C_1 & C_2 & D \end{array}\right], \tag{7.5}$$

and $A_{11}$ and $A_{22}$ contain the *dominant* and *non-dominant* modes of the system, respectively. Thus, the reduced model (7.3) contains essentially the dominant dynamics of the system. The modal approach can be easily used in combination with balancing related methods for reduction of unstable systems. In this case, the dominant part includes all unstable dynamics and the model reduction is performed only on the non-dominant stable part.

In selecting numerical algorithms for model reduction, specific requirements for a satisfactory algorithm have been formulated to assess its suitability to serve as basis for robust numerical software implementations: (1) general applicability regardless the original system is minimal or not; (2) emphasis on enhancing the numerical accuracy of computations; (3) relying on numerically reliable procedures; (4) independence of results of state space coordinate scaling. In what follows we discuss these aspects for the numerical algorithms used for the class of balancing related model reduction methods.

The first requirement is very important because, in practice, due to the presence of roundoff errors, it is often impossible to distinguish between a *true* non-minimal and a *nearly* non-minimal system. At algorithmic level, this requirement can be fulfilled by using algorithms which compute $L$ and $T$ directly, without determining $Z$ or $Z^{-1}$. In particular, if the original system is non-minimal, then $L$ and $T$ can be chosen to compute an *exact* minimal realization of the original system [19]. In this way, model reduction can also serve as a numerically sound alternative to solve minimal realization problems.

The emphasis on improving the accuracy of computations led to so-called algorithms with *enhanced accuracy*. In the balancing related model reduction methods, the truncation matrices $L$ and $T$ are usually determined from the controllability and observability gramians $P$ and $Q$, satisfying a pair of continuous-time ($c$) or discrete-time ($d$) Lyapunov equations

$$\left\{\begin{array}{rcl} AP + PA^T + BB^T &=& 0 \\ A^TQ + QA + C^TC &=& 0 \end{array}\right. \quad (c), \quad \left\{\begin{array}{rcl} APA^T + BB^T &=& P \\ A^TQA + C^TC &=& Q \end{array}\right. \quad (d)$$

Since $P$ and $Q$ are positive semi-definite symmetric matrices, they can be expressed in Cholesky factorized forms $P = SS^T$ and $Q = R^TR$ with $S$ and $R$ upper-triangular matrices. The Cholesky factors can be computed directly by solving ($c$) or ($d$) using numerically reliable algorithms pro-

posed by Hammarling [9] to solve non-negative definite Lyapunov equations. Then, the computation of $L$ and $T$ can be done entirely on basis of the Cholesky factors $S$ and $R$, leading to the so-called *square-root* (**SR**) methods for model reduction.

Consider the *singular value decomposition* (SVD)

$$RS = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \mathrm{diag}(\Sigma_1, \Sigma_2) \begin{bmatrix} V_1 & V_2 \end{bmatrix}^T, \tag{7.6}$$

where

$$\Sigma_1 = \mathrm{diag}(\sigma_1, \ldots, \sigma_r), \quad \Sigma_2 = \mathrm{diag}(\sigma_{r+1}, \ldots, \sigma_n),$$

and $\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} \geq \ldots \geq \sigma_n \geq 0$ are *the Hankel singular values* of the system. The **SR** method for the BTA approach of [14] determines $L$ and $T$ as [17]

$$L = \Sigma_1^{-1/2} U_1^T R, \qquad T = S V_1 \Sigma_1^{-1/2}.$$

If $r$ is the order of a minimal realization of $G$ then the gramians corresponding to the resulting realization are diagonal and equal. In this case the minimal realization is called *balanced*. Since the **SR** method can be used to compute balanced minimal representations resulting in a partitioned form like in (7.2), it can also be used for computing reduced order models by using the SPA formulas [11] or as the preliminary step in performing the HNA [8]. The **SR** approach is usually very accurate for well-equilibrated systems. However if the original system is highly unbalanced, potential accuracy losses can be induced in the reduced model if either $L$ or $T$ is ill-conditioned (i.e., nearly losses maximal rank).

In order to avoid computations with ill-conditioned truncation matrices, a *balancing-free* (**BF**) approach has been proposed in [16] in which always well-conditioned matrices $L$ and $T$ can be determined. These matrices are computed from two orthogonal matrices whose columns span the right and left eigenspaces of the product $PQ$ corresponding to the first $r$ largest eigenvalues $\sigma_1^2, \ldots, \sigma_r^2$, respectively. Because of the need to compute explicitly $P$ and $Q$ as well as their product, this approach is usually less accurate for moderately ill-balanced systems than the **SR** approach.

A *balancing-free square-root* (**BFSR**) algorithm for the BTA method, combining the main advantages of the **BF** and **SR** approaches has been introduced in [19]. The truncation matrices $L$ and $T$ can be determined as

$$L = (Y^T X)^{-1} Y^T, \qquad T = X,$$

where $X$ and $Y$ are $n \times r$ matrices with orthogonal columns computed from two QR decompositions

$$SV_1 = XW, \qquad R^T U_1 = YZ,$$

with $W$ and $Z$ non-singular upper-triangular matrices. The accuracy of the **BFSR** algorithm is usually better than either of **SR** or **BF** approaches.

A **BFSR** method for the SPA approach has been proposed in [18]. The matrices $L$ and $T$ are computed such that the system $(LAT, LB, CT, D)$ is minimal and the product of corresponding gramians has a block-diagonal structure which allows the application of the SPA formulas.

Concerning the numerical reliability aspect, all computational steps for the **SR** or **BFSR** methods can be performed by using exclusively numerically stable algorithms. In the case of the HNA method for a continuous-time system, straightforward matrix expressions [8] are used to determine a system whose stable part represents the optimal Hankel-norm approximate model $G_r$ of the original $G$ (see section 7.2.2 for more details on the modal separation technique). Since the formulas for the discrete-time case are much more involved, we preferred to use an approach based on discrete-to-continuous bilinear transformations, which allows the usage of continuous-time formulas for the discrete-time case as well.

The independence of scaling is a very important aspect for the overall applicability of the BTA, SPA or HNA methods to possibly badly scaled models. In theory, the Hankel singular values are input-output system invariants, and therefore are not affected by coordinate transformations. In particular, they are not affected by state coordinate scaling with diagonal matrices. However, the numerical accuracy of computed Hankel singular values can be tremendously affected by scaling since their computation involves two critical numerical computations which are not independent of the scaling: (1) the solution of Lyapunov equations with possibly badly scaled system matrices; and (2) the SVD determination from the product of two matrices resulted from previous step having possibly quite different ranges of elements. It is therefore clear that the accuracy of singular values and of all subsequent computations can be substantially improved by a proper scaling of the original system by trying to reduce the 1-norm of the scaled system matrix

$$\mathcal{S} = \begin{bmatrix} Z^{-1}AZ & Z^{-1}B \\ CZ & 0 \end{bmatrix},$$

with an appropriate positive definite diagonal scaling matrix $Z$. Such an optional preliminary scaling is part of each model reduction routine implemented for SLICOT.

For all three methods BTA, SPA and HNA the absolute approximation error $G - G_r$ for an $r$-th order approximation satisfies

$$\|G - G_r\|_\infty \leq 2 \sum_{k=r+1}^{n} \sigma_k. \tag{7.7}$$

Note that the actual error may be considerably less than the above error bound, so that this formula can be seen generally as a guide to choose the appropriate order of the reduced system. In case of optimal HNA method,

the optimum $G_r$ achieves

$$\inf \|G - G_r\|_H = \sigma_{r+1}$$

and even a feedtrough matrix $D_r$ can be chosen (see [8] for details) such that the error bound in (7.7) is half of the bound for BTA and SPA. This feature is however not available in the implemented SLICOT routine for HNA.

### *7.2.2 Reduction of unstable systems*

The reduction of unstable systems can be performed by using the methods for stable systems in conjunction with two embedding techniques. One important requirement for these techniques is that they must not have computational overhead when applied to stable systems. By fulfilling this requirement, a single routine can be used in principle for reducing both stable and unstable systems.

The first approach consists in reducing only the stable projection of $G$ and then including the unstable projection unmodified in the resulting reduced model. The following is a simple procedure for this computation:

1. Decompose additively $G$ as $G = G_1 + G_2$, such that $G_1$ has only stable poles and $G_2$ has only unstable poles.
2. Determine $G_{1r}$, a reduced order approximation of the stable part $G_1$.
3. Assemble the reduced model $G_r$ as $G_r = G_{1r} + G_2$.

For the model reduction at step 2 any of methods available for stable systems can be used. The stable-unstable spectral separation at step 1 is done by performing a system similarity transformation as in (7.5), where $A_{11}$ contains the stable eigenvalues and $A_{22}$ contains the unstable eigenvalues of $A$. This separation is performed in two steps. First the state matrix $A$ is reduced to an ordered real Schur form with the diagonal $1 \times 1$ or $2 \times 2$ blocks ordered according to the desired stable-unstable eigenvalue splitting. This reduction is performed by using only orthogonal transformations. Then the (1,2)-block of the resulting partitioned Schur matrix is zeroed by performing an additional non-orthogonal similarity transformation. This transformation involves the solution of a Sylvester equation with the coefficient matrices in real Schur form. Note that for a stable system this step is not performed and thus no computational overhead occurs since the reduction to the Schur form is always necessary when employing Hammarling's method [9] to compute the Cholesky factors of gramians.

The second approach is based on reducing the factors of a stable rational coprime factorization of $G$. For example, consider the left coprime factorization $G = M^{-1}N$, where $M$ and $N$ are stable rational TFMs. Note that the factors are left coprime if there exist stable rational $X$ and $Y$ such

that $MX + NY = I$. The following procedure can be used to compute an $r$-th order approximation $G_r$ of an arbitrary $n$-th order system G by approximating the factors of its left coprime factorization:

1. Compute a left coprime factorization of $G$ as $G = M^{-1}N$, with $M$ and $N$ stable TFMs.

2. Compute for the stable system of order $n$ with the TFM $[\, N \; M \,]$ an approximation $[\, N_r \; M_r \,]$ of order $r$.

3. Form the $r$-th order approximation $G_r = M_r^{-1}N_r$.

The usefulness of this approach relies on the assumption that the McMillan degree of $G = M^{-1}N$ is generally that of $[\, N \; M \,]$ and similarly for $G_r = M_r^{-1}N_r$ and $[\, N_r \; M_r \,]$. In this way the reduction of the McMillan degree of $[\, N \; M \,]$ through approximation by $[\, N_r \; M_r \,]$ is equivalent to reduction of the McMillan degree of $G$ by $G_r$, which is our objective. The factorization methods proposed in [21, 22] fulfill the above assumptions, and thus can be employed to implement the above procedure. Note that the approximations computed for the factors of a coprime factorization with inner denominator by using the SPA method preserve this property for the reduced factors [22].

The coprime factorization approach used in conjunction with the BTA method fits well in the general projection formulation introduced in previous subsection. The gramians necessary to compute the projection are the gramians of the extended system with the TFM $[\, N \; M \,]$. The resulting truncation matrices $L$ and $T$ determined by using either the **SR** or **BFSR** methods can be directly applied to the matrices of the original system.

The requirement for an efficient embedding which prevents computational overheads for stable systems can be also fulfilled by using the factorization algorithms of [21, 22]. These algorithms are based on numerically reliable Schur techniques for pole assignment and can be used to compute coprime factorizations with prescribed stability degree of the factors [21] or with inner denominators [22]. In both cases, the resulting state matrix is common to both factors and is in a real Schur form. Thus the model reduction method has no overhead if the original system $G$ is already stable, because in this case $N = G$ and $M = I$ and the reduction of state matrix to the real Schur form is still necessary to compute the gramians.

### *7.2.3 Implementation of software for model reduction*

The basis for implementation of the model reduction routines in SLICOT formed the collection of routines available in the RASP-MODRED library [20], implemented on the basis of the standard linear algebra package LAPACK [1]. All new SLICOT routines originating from the RASP-MODRED

library have been practically rewritten. Several routines represent completely new implementations. A special emphasis has been put on an appropriate modularization of the routines in the model reduction chapter of SLICOT. For this purpose, a computational kernel formed of three basic routines is shared by all higher level user callable routines.

Both the **SR** and **BFSR** versions of the BTA and SPA algorithms are implemented in SLICOT library. The implementation of the HNA method uses the **SR** BTA method to compute a balanced minimal realization of the original system. The following table contains the complete list of model reduction routines available in SLICOT for stable model reduction:

| Name | Function |
|---|---|
| AB09AD | computes reduced (or minimal) order balanced models using either the **SR** or the **BFSR** BTA method |
| AB09BD | computes reduced order models using the **SR** or **BFSR** SPA method |
| AB09CD | computes reduced order models using the optimal HNA method based on **SR** balancing |
| AB09DD | computes reduced order models using the singular perturbation formulas (7.4) |
| AB09AX | computes reduced (or minimal) order balanced models using either the **SR** or the **BFSR** BTA method for a system with state matrix in real Schur form |
| AB09BX | computes reduced order models using the **SR** or **BFSR** SPA method for a system with state matrix in real Schur form |
| AB09CX | computes reduced order models using the optimal HNA method based on **SR** balancing for a system with state matrix in real Schur form |

Three user callable routines AB09AD, AB09BD and AB09CD implement the three basic algorithms for BTA, SPA and HNA methods, respectively. All these routines perform optionally the scaling of the original system. Each of routines handles both continuous-time as well as discrete-time systems. For implementing the discrete-time HNA method, bilinear continuous-to-discrete transformation techniques have been employed. AB09AX, AB09BX and AB09CX are lower level supporting routines which perform basically the same reductions as the corresponding main user callable routines, but for systems with the state matrix already reduced to the real Schur form and possibly already scaled. These routines form the computational kernel of the whole model reduction software in SLICOT being called by the user-callable routines for reduction of both stable and unstable systems.

SLICOT also provides tools to perform the reduction of unstable systems. On the basis of newly implemented routines to compute left/right

coprime factorizations with prescribed stability degree or with inner denominators, or to compute additive spectral decompositions, several user callable routines have been implemented for reduction of unstable systems. A modular implementation allowed flexible combinations between various factorization/decomposition and model reduction methods for stable systems. The following routines are available to perform model reduction of unstable systems:

| Name | Function |
|---|---|
| AB09MD | computes reduced order models for unstable systems using the BTA method in conjunction with additive stable/unstable spectral decomposition |
| AB09ND | computes reduced order models for unstable systems using the SPA method in conjunction with stable/unstable additive spectral decomposition |
| AB09ED | computes reduced order models for unstable systems using the optimal HNA method in conjunction with additive stable/unstable spectral decomposition |
| AB09FD | computes reduced order models for unstable systems using the **SR** or **BFSR** BTA method in conjunction with left/right coprime factorization methods |
| AB09GD | computes reduced order models for unstable systems using the **SR** or **BFSR** SPA method in conjunction with left/right coprime factorization methods |

The routines AB09MD, AB09ND and AB09ED implement the spectral separation approach in combination with BTA, SPA and HNA methods, respectively. They provide an additional flexibility by allowing to specify an arbitrary stability boundary inside the standard stability regions (continuous or discrete). The dominant part of the system having poles only in the "unstable" region is retained in the reduced model, and only the "stable" part is approximated. This leads to an effective combination of balancing methods with the modal reduction approach (see also [24]). The coprime factorization based routines AB09FD and AB09GD allows arbitrary combinations of BTA and SPA methods, respectively, with four types of coprime factorizations.

It is important to emphasize that the model reduction routines for unstable systems can be applied with practically no efficiency loss to reduce stable systems too. Since these routines can be seen as completely general tools for order reduction of linear time-invariant systems, they form the basis to implement the interface software to user-friendly environments (see Section 7.3).

An important number of new routines to perform system similarity transformations, to compute system norms, special factorizations or decompositions, have been implemented for the special needs of the model reduction

routines. Two user callable transformation routines implement frequently used similarity transformations on system matrices:

| Name | Function |
|---|---|
| TB01WD | performs an orthogonal similarity transformation to reduce the system state matrix to a real Schur form |
| TB01LD | performs an orthogonal similarity transformation to reduce the system state matrix to an ordered real Schur form |

The following routines have been implemented for computing system norms:

| Name | Function |
|---|---|
| AB13AD | computes the Hankel norm and the Hankel singular values of the stable projection of a transfer-function matrix |
| AB13BD | computes the $H_2$- or $L_2$-norm of a transfer-function matrix |
| AB13CD[3] | computes the $H_\infty$-norm of a continuous-time transfer-function matrix |

Several factorization and decomposition routines of TFMs have been implemented for the special needs of model reduction of unstable systems:

| Name | Function |
|---|---|
| TB01KD | computes an additive spectral decomposition of a transfer-function matrix with respect to a specified region of the complex plane |
| SB08CD | computes the state-space representations of the factors of a left coprime factorization with inner denominator |
| SB08DD | computes the state-space representations of the factors of a right coprime factorization with inner denominator |
| SB08ED | computes the state-space representations of the factors of a left coprime factorization with prescribed stability degree |
| SB08FD | computes the state-space representations of the factors of a right coprime factorization with prescribed stability degree |
| SB08GD | computes the state-space representation corresponding to a left coprime factorization |
| SB08HD | computes the state-space representation corresponding to a right coprime factorization |

A complete list of implemented model reduction and auxiliary routines is given in the NICONET Report NIC1999-8.[4] A typical user interface is presented in Appendix 7.A for the user callable Fortran subroutine `AB09MD`.

[3] implemented by P. Petkov from the Technical University of Sofia

[4] available at `ftp://wgs.esat.kuleuven.ac.be/pub/WGS/REPORTS/NIC1999-8.ps.Z`

## 7.3 Integration in user-friendly environments

One of the main objectives of the NICONET project is to provide, additionally to standardized Fortran codes, high quality software embedded into user-friendly environments for *computer aided control system design* (CACSD). Two target environments have been envisaged: the popular commercial numerical computational environment MATLAB and the public domain MATLAB-like environment `Scilab`. Both allows to easily add external functions implemented in general purpose programming languages like C or Fortran. In case of MATLAB, the external functions are called *mex*-functions and have to be programmed according to precise programming standards. In `Scilab`, external functions can be similarly implemented and only several minor modifications were necessary to the MATLAB *mex*-functions to adapt them to `Scilab`. It is expected that generally MATLAB *mex*-functions could also serve a starting points for implementing external function interfaces to other similar CACSD environments (e.g., MATRIX$_X$).

### *7.3.1 Integration in* MATLAB

One important aspect implementing *mex*-functions was to keep their total size as small as possible. Since the standardized model reduction programs in SLICOT share many routines from BLAS, LAPACK and SLICOT, it was decided to implement a single function covering all model reduction functionality provided in SLICOT. The *mex*-function for model reduction is called `sysred` and provides a flexible interface to practically all functional features provided by the model reduction routines AB09MD, AB09ND, AB09ED, AB09FD, AB09GD for reduction of stable/unstable linear systems using the BTA, SPA and HNA methods in conjunction with stable coprime factorization and stable/unstable spectral decomposition. The MATLAB help function of this *mex*-function is listed in Appendix 7.B.

The *mex*-function `sysred` has been implemented in Fortran 90. However, the only used feature of Fortran 90 is the use of `ALLOCATE` and related statements for internal workspace management (i.e., allocating or deleting workspace for arrays). The parts of the codes using Fortran 90 features are clearly delimited by comments and can be easily converted to Fortran 77 by using calls to special MATLAB storage allocation routines.

To provide a convenient interface to work with control objects defined in the Control Toolbox, several easy-to-use higher level model reduction functions have been additionally implemented explicitly addressing some of available features. The Table 7.1 contains the list of implemented *m*-functions. A sample *m*-function, `bta.m`, is listed in Appendix 7.C.

**TABLE 7.1. Model reduction $m$-functions**

| Name | Function |
|---|---|
| `bta` | **BFSR** BTA combined with additive spectral decomposition |
| `btabal` | **SR** BTA combined with additive spectral decomposition |
| `bta_cf` | **BFSR** BTA combined with stable coprime factorizations |
| `btabal_cf` | **SR** BTA combined with stable coprime factorizations |
| `spa` | **BFSR** SPA combined with additive spectral decomposition |
| `spabal` | **SR** SPA combined with additive spectral decomposition |
| `spa_cf` | **BFSR** SPA combined with stable coprime factorizations |
| `spabal_cf` | **SR** SPA combined with stable coprime factorizations |
| `hna` | **SR** HNA combined with additive spectral decomposition |

### *7.3.2 Integration in `Scilab`*

The `Scilab` interface[5] is essentially the same as that for MATLAB. In particular, the names of the *mex*-file and $m$-files are the same. The source code of the *mex*-files for MATLAB served as basis for the `Scilab` interface. For an increased portability, it was decided do not use the Fortran 90 `ALLOCATE` statement for workspace management. Instead, the needed workspace for each variable is allocated on the internal stack of `Scilab` using the `CREATEVAR` routine of `Scilab`. For example, a variable $A$ is referred in the `Scilab` interface program as `stk(ptrA)`, where `ptrA` is the pointer determined by `CREATEVAR` when allocating storage for $A$. These minor modifications allow the use of a freely available Fortran 77 compiler like `g77` and facilitate also an automatic translation to C using `f2c`.

The MATLAB $m$-files also required some modifications to cope with the syntax of the `Scilab` command language. As an illustration, in Appendix 7.D is the `Scilab` code `bta.sci` corresponding to the MATLAB $m$-file `bta.m` given in Appendix 7.C. Observe that the calling sequences are the same both in MATLAB and `Scilab`, and even the help files are very similar. The only difference is that `Scilab` help files are formatted ASCII texts which are automatically generated from the corresponding $m$-files.

## 7.4 Testing and performance comparisons on benchmark problems

Extensive testing of the implemented software has been performed using several benchmark problems. In what follows we describe examples used to

[5]implemented by F. Delebecque from INRIA-Rocquencourt

test the model reduction routines for stable and unstable systems.

### *7.4.1 PS: Power system model – continuous-time.*

This is a continuous-time linearized state space model of a two-area interconnected power system [7]. The model has the form

$$\begin{aligned} \dot{x} &= Ax + B_1u + B_2w \\ y &= Cx \end{aligned}$$

where $x \in \mathbb{R}^7$ is the state vector, $u \in \mathbb{R}^2$ is the command input vector, $w \in \mathbb{R}^2$ is the disturbance input vector and $y \in \mathbb{R}^3$ is the measurable output vector. The matrices of this model are given in Appendix 7.E. Note that the partial model $(A, B_1, C)$ has been used as test example by several SLICOT test programs.

The PS model is stable, minimal and has the Hankel-singular values

$$\{\, 3.9137,\ 3.5944,\ 2.5277,\ 1.0888,\ 0.6526,\ 0.0276,\ 0.0275 \,\}.$$

Taking into account the gap between the 5-th and 6-th singular values, a 5-th order model seems to be appropriate for a lower order approximation. The BTA, SPA and HNA methods produced reduced order models of order 5, $\mathrm{PS}_1$, $\mathrm{PS}_2$ and $\mathrm{PS}_3$, respectively, which approximately preserve the dominant poles of the original system

| Poles of PS | Poles of $\mathrm{PS}_1$ | Poles of $\mathrm{PS}_2$ | Poles of $\mathrm{PS}_3$ |
|---|---|---|---|
| $-0.5181 + 3.1259i$ | $-0.5053 + 3.1206i$ | $-0.5186 + 3.1228i$ | $-0.5118 + 3.1221i$ |
| $-0.5181 - 3.1259i$ | $-0.5053 - 3.1206i$ | $-0.5186 - 3.1228i$ | $-0.5118 - 3.1221i$ |
| $-1.3550 + 2.1866i$ | $-1.2923 + 2.1162i$ | $-1.3598 + 2.1692i$ | $-1.3250 + 2.1430i$ |
| $-1.3550 - 2.1866i$ | $-1.2923 - 2.1162i$ | $-1.3598 - 2.1692i$ | $-1.3250 - 2.1430i$ |
| $-1.6916$ | $-1.4233$ | $-1.6578$ | $-1.5351$ |
| $-13.1438$ | | | |
| $-13.1617$ | | | |

However, the three methods approximate differently the zeros of the original system, the BTA and SPA methods producing even non-minimum phase zeros:

| Zeros of PS | Zeros of $\mathrm{PS}_1$ | Zeros of $\mathrm{PS}_2$ | Zeros of $\mathrm{PS}_3$ |
|---|---|---|---|
| $\infty$ | 1.4438 | 1.4438 | $-9.2069$ |
| $\infty$ | $\infty$ | 0 | $\infty$ |
| | $\infty$ | $\infty$ | $\infty$ |
| | $\infty$ | | |

There is little difference in step responses for different input-output channels and also the Nyquist plots show good agreements. In Figure 7.1 the Nyquist plots for the $u_1$–$y_1$ channel are presented. Each of the computed 5-th order approximate models is suitable to perform controller synthesis.

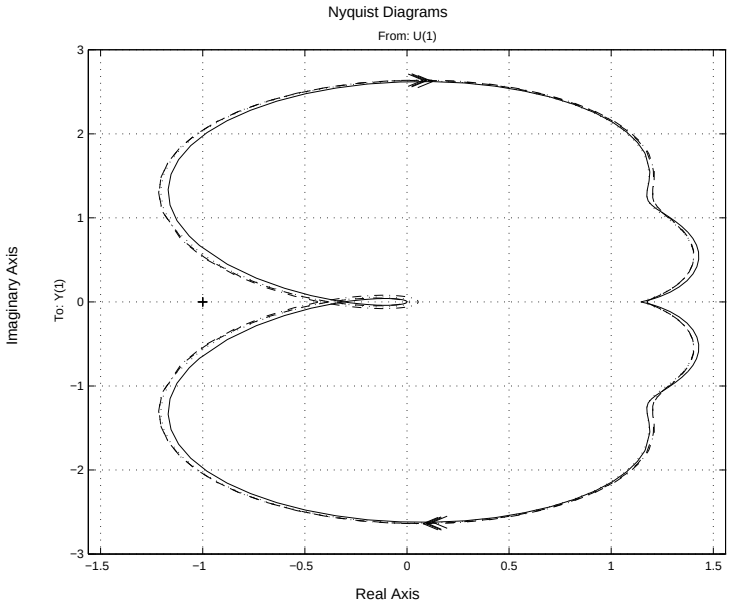


**FIGURE 7.1. Frequency responses for element $g_{11}(s)$ of PS.**

### *7.4.2 PSD: Power system model – discrete-time.*

This is the PS model discretized with a sampling period of $T = 0.1$ sec. The matrices of sampled-data system matrices are given in Appendix 7.E. Three 5-th order approximations have been computed using the BTA, SPA and HNA methods. Figure 7.2 shows the good agreement both in time domain and frequency-domain of the original and reduced models.

### *7.4.3 TGEN: Nuclear plant turbo-generator model.*

This is a 10-th order linearized model of a 1072 MVA nuclear powered turbo-generator [10]. The system is stable and minimal phase. The Hankel-singular values of the system are

$$\{\, 455.98,\ 76.52,\ 68.57,\ 10.429,\ 7.24,\ 0.27,\ 0.11,\ 0.0022,\ 0.0019,\ 0.0014 \,\},$$

thus a 5-th order approximation seems to be appropriate. Figure 7.3 compares the results obtained with all three methods: BTA (dashed line), SPA (dashdot line), and HNA (dotted line), on basis of element $g_{11}(s)$ of the TFM. It is easy to see the good low frequency approximation property of SPA method and the good high-frequency approximation property of BTA and HNA methods. Note that all three methods produce non-minimum phase approximations.

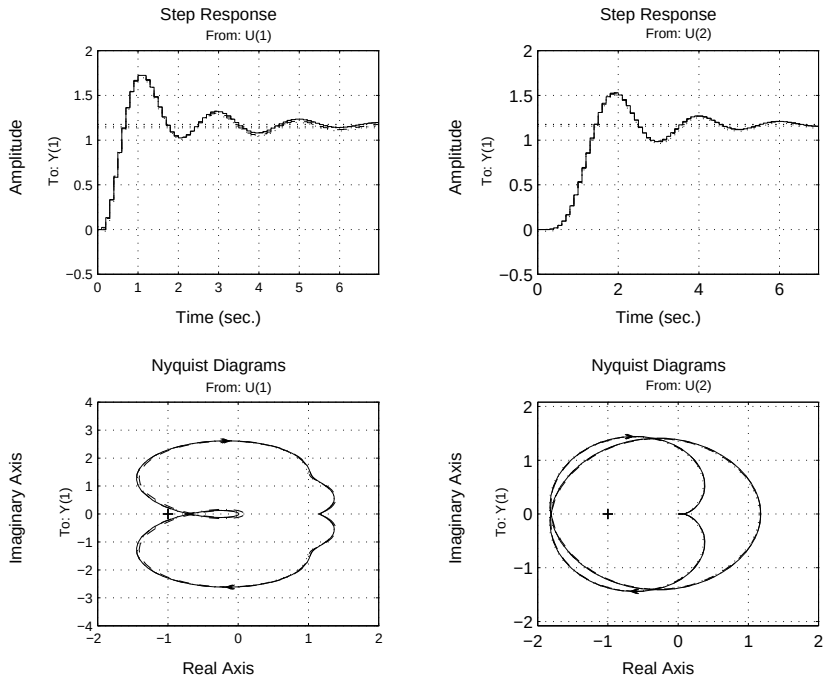


**FIGURE 7.2. Time and frequency responses for elements $g_{11}(s)$ and $g_{12}(s)$ of PSD.**

### *7.4.4 PSU: Unstable continuous-time model.*

This model served only for numerical test and resulted by replacing in the PS model $A$ by $A+\alpha I$, for different values of $\alpha$ leading to unstable systems. For $\alpha = 1$ we applied the BTA method to the PSU model in combination with additive spectral decomposition and four types of coprime factorizations: stable letft/rigth coprime factorizations and letft/rigth right coprime factorizations with inner denominators. Figure 7.4 shows good agreements of the Nyquist plots for the transfer function of $u_1$–$y_1$ input/output channel for all five methods. Similar results have been obtained with the SPA and HNA methods used in combination with factorization/decomposition techniques.

### *7.4.5 ACT: Badly scaled actuator model*

This 5-th order single-input model resulted from the physical modelling of a hydraulic actuator for helicopter active vibration damping. The state space representation for this model is given in Appendix 7.E. Due to its extremely poor scaling originated from the usage of International System (SI) units, it is expected that this model will pose numerical difficulties to

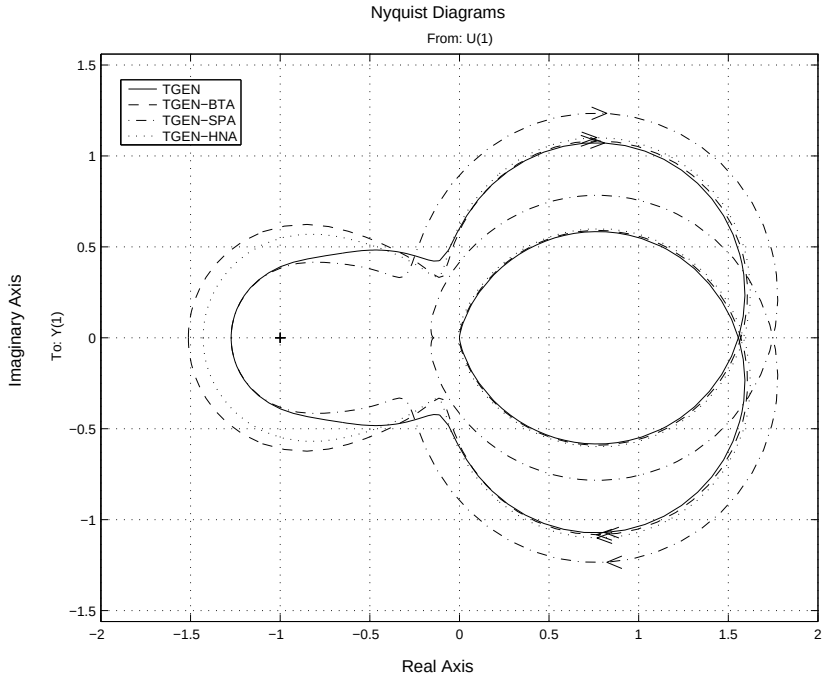


**FIGURE 7.3. Frequency responses for element $g_{11}(s)$ of TGEN.**

many category of programs thus leading often to wrong numerical results.

The computed Hankel singular values with `sysred` are:

$$\{\, 1.38934 \cdot 10^7,\ 6.10346 \cdot 10^6,\ 1.04050 \cdot 10^6,\ 1.03985 \cdot 10^6,\ 8.26634 \cdot 10^5 \,\}$$

However, the Hankel singular values computed with the MATLAB function `balreal` are completely wrong:

$$\{\, 3.3357 \cdot 10^8,\ 1.92108 \cdot 10^8,\ 9.92157 \cdot 10^5,\ 1.47773 \cdot 10^4,\ 1.91871 \cdot 10^3 \,\}.$$

### *7.4.6 Uncertainty models*

The newly developed model reduction tools are well-suited to be employed as basic computational tools for exact and approximate order reduction of parametric uncertainty models described by *linear fractional transformations* (LFTs). An LFT model arises, for example, by expressing the state space matrices $A(p)$, $B(p)$, $C(p)$, $D(p)$ of a symbolically linearized system depending rationally on parameters in a vector $p$, as a diagonal structured feedback around a constant linear system. LFT-models are basically multi-dimensional (m-D) systems and no general algorithms are known to compute m-D minimal realizations or approximations. Since the resulting

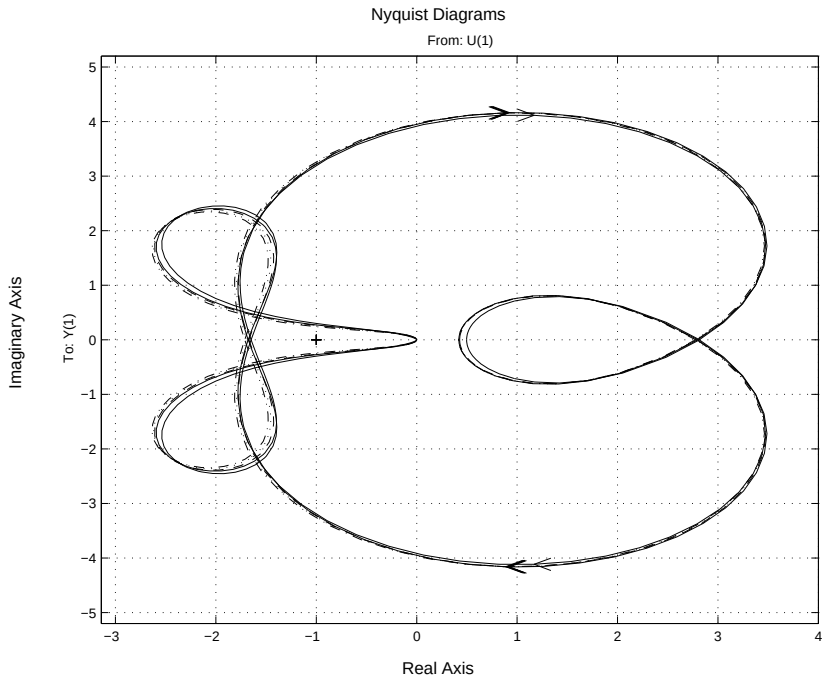


**FIGURE 7.4. Frequency responses for element $g_{11}(s)$ of PSU.**

orders of ad-hoc built LFTs are typically high even for relatively simple parametric models, order reduction (exact or approximate) is an important aspect of LFTs based modeling. Sequential 1-D minimal realization techniques can be successfully employed to achieve substantial order reduction. Since most of LFT descriptions are basically assimilated to discrete-time systems, model reduction techniques able to handle non-minimal 1-D discrete-time systems can be employed not only to perform exact reductions but also to compute lower order approximations. In [26], several LFT-models have been generated starting from a parametric linear state space model of a civil aircraft. The order of initial LFT models were up to 300 and reductions employing the *mex*-function `sysred` led to manageable low order exact and approximate LFT models. The high accuracy of approximations was assessed by using Monte-Carlo analysis. Note that the typical model features required for such an order reduction (i.e, discrete-time, non-minimal, unstable) are not available in the commercial CACSD software (see also Table 7.3).

### *7.4.7 Timing results*

Randomly generated systems have been used to compare the speed of methods with carefully implemented MATLAB *m*-functions from the HTOOLS Toolbox [25]. In the following table, we present timing results for randomly generated stable systems of orders up to 512 comparing for the square-root BTA method the efficiency of the *mex*-function `sysred`, the *m*-functions `sqrmr` from HTOOLS and `balreal` from the Control Toolbox [12]. Note that for dimensions above $n = 32$, `balreal` systematically exited with the message `"System must be reachable"`, which is evidently a nonsense. The results in Table 7.2 have been obtained on a Pentium II 400 MHz Personal Computer running under Windows NT 4.0. The *mex*-function `sysred` has been produced using Digital/Compaq Visual Fortran V 5.1.

| Order | Times [sec] | | |
|---|---|---|---|
| | `sysred` | `sqrmr` | `balread` |
| 16 | 0.003 | 0.17 | 0.04 |
| 32 | 0.01 | 0.5 | 0.17 |
| 64 | 0.11 | 2.14 | * |
| 128 | 0.78 | 10.55 | * |
| 256 | 6.12 | 63.75 | * |
| 512 | 76.23 | 478.69 | * |

**TABLE 7.2. Timing results for `sysred`, `sqrmr` and `balreal`.**

A speed-up of order 10 or higher can be observed when using the Fortran based implementation `sysred` instead of the pure MATLAB implementations in `sqrmr` and `balreal`. The above table also illustrates the increased numerical robustness of structure exploiting algorithms used both in `sysred` and `sqrmr`, in comparison with the less accurate and numerically fragile method implemented in the MATLAB function `balreal`.

## 7.5 Testing on industrial benchmark problems

### *7.5.1 ATTAS: Linearized aircraft model*

This model describes the linearized rigid body dynamics of the Advanced Technology Testing Aircraft System (ATTAS) of DLR[6] during the landing approach. The nonlinear model of ATTAS used for linearization has been obtained using the object oriented modelling tool Dymola [6]. Besides flight dynamics, this model includes actuators and sensors dynamics, as well as engine dynamics. Several low pass filters to eliminate structure induced

[6]German Aerospace Center

dynamics in outputs are also included. The total order of the model is 51. The linearized model has an unstable spiral mode. Moreover, because of presence of position states, there are three pure integrators in the model and an additional one for the heading angle. There are 6 control inputs and 3 wind disturbance inputs, and 9 measurement outputs. This model serves basically for the evaluation of linear handling criteria in a multi-model based robust autopilot design.

To speed-up the evaluation of different handling quality criteria, lower order design models have been determined by using the BTA model reduction approach. A 15-th order approximation has been computed using model reduction followed by minimal realization. The reduced order model fits almost exactly the original 51-th order model both in time as well as in frequency domain. Figure 7.5 shows very good agreements between the frequency responses of the original and reduced model for elements $g_{22}$ of the corresponding TFMs.

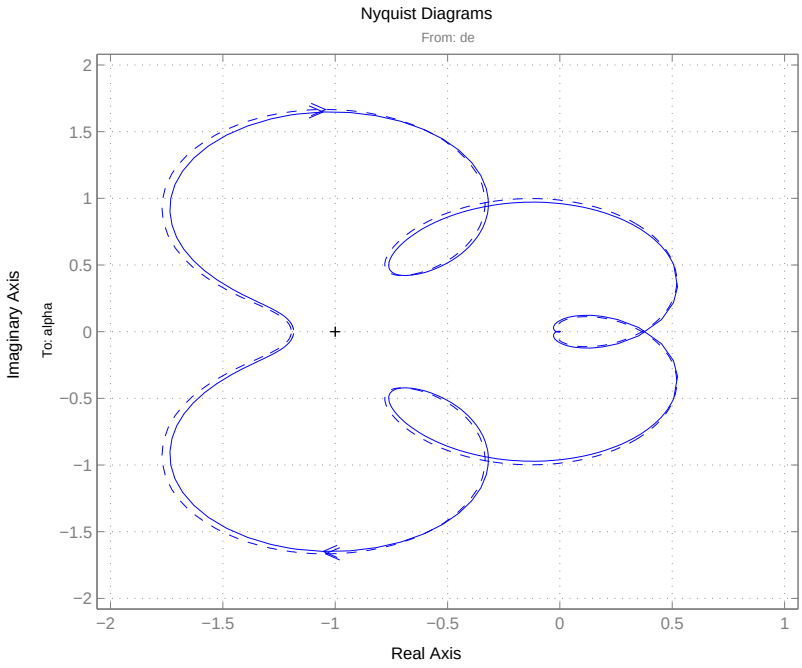


**FIGURE 7.5. Frequency responses for element $g_{22}(s)$ of ATTAS.**

For longitudinal flight, a minimal order stable model has been derived by combining model reduction and minimal realization techniques. The reduced longitudinal ATTAS model has 7 states, 4 inputs and 4 outputs. For lateral flight, a minimal order model has been computed having 10 states, 2 inputs and 5 outputs. Both these models approximate practically

exactly the corresponding parts of the dynamics of the original 51 order model. Note that handling this model raises several difficulties for currently available model reduction software such as the presence of unstable modes or of redundant dynamics (non-minimal model). For instance, this model is intractable with standard model reduction tools available in the Control Toolbox of MATLAB.

### *7.5.2 CDP: CD-player finite element model*

This is a 120-th order single-input single-output system which describes the dynamics between the lens actuator and radial arm position of a portable compact disc player discussed in [27]. Due to physical constraints on the size of the systems's controller, a reduced model with order $r \leq 15$ is desired. For testing purposes, three 10-th order models have been determined using the BTA, SPA and HNA methods. Figure 7.6 compares the performance of computed approximations on basis of their Bode plots (BTA – dashed line, SPA – dashdot line, HNA – dotted line):

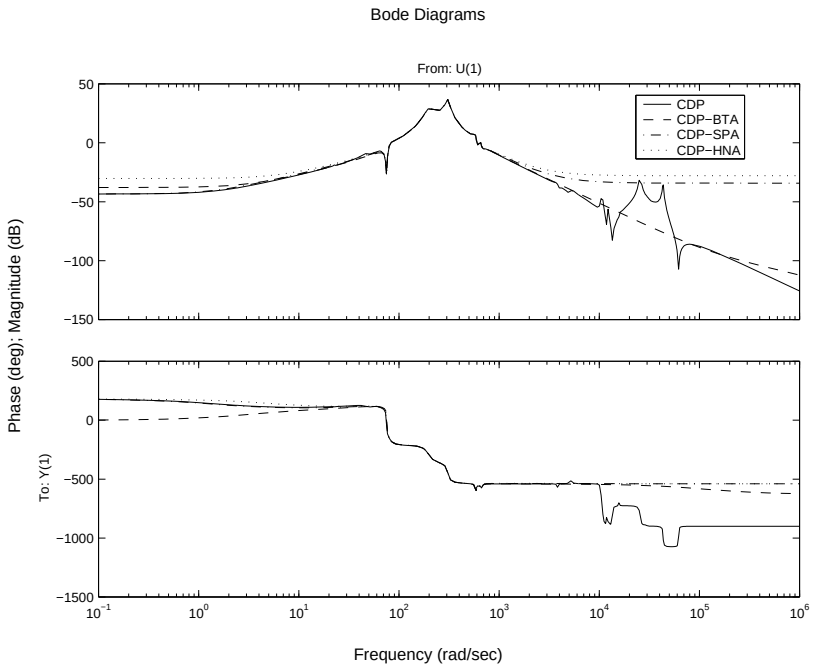


**FIGURE 7.6. Frequency responses for CDP.**

All methods approximate satisfactorily the central peek at a frequency about 120 Hz, but have different approximation properties at low and high frequencies. Both SPA and HNA approximations seems to be inappropriate,

although the stationary error for the SPA method is zero. However, the BTA methods appears to provide an acceptable 10-th order approximation.

### *7.5.3 GAS: Gasifier model*

GEC ALSTHOM developed a detailed nonlinear gasifier model, to serve as a benchmark problem for simulation and robust control. The model includes all significant effects as drying of coal and limestone, pyrolysis and volatilisation of coal, the gasification process itself and elutriation of fines. This model has been validated using measured time histories from the British Coal CTDD experimental test facility and it was shown that the model predicts the main trends in the fuel gas quality. Linearized models at 0%, 50% and 100% loads have been generated to be used for a multi-model based robust controller design. Numerical difficulties have been reported in [15] for the 100% load model when using the model reduction tools in MATLAB but also in the symbolic manipulation package *Mathematica*[7]. The cause of difficulties is the poor scaling of the model. This can be seen by comparing the step responses for element $g_{11}(s)$ for the original and scaled system at 0% load in Figure 7.7.

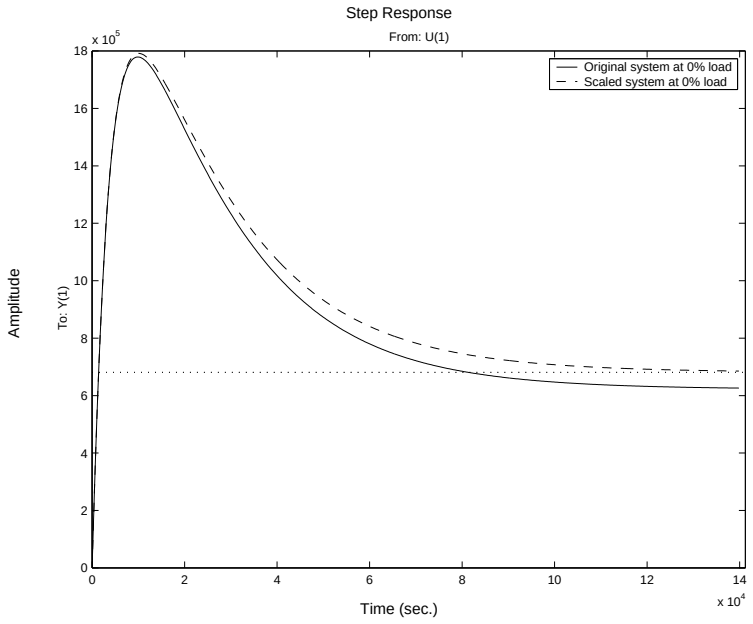


**FIGURE 7.7. Step responses for original and scaled GAS models.**

[7] *Mathematica* is a trademark of Wolfram Research, Inc.

The GAS model has order 25 and is non-minimal. The norms of state matrices for the three models are about $10^9$, but after scaling with the SLICOT routine TB01ID, all norms can be reduced below 100. Such a preliminary scaling is not necessary for using `sysred`, being an implicit feature of this *mex*-function. Still, for simulations we used the scaled models to avoid numerical difficulties with MATLAB plotting functions and to make the comparison more reliable. All three models are non-minimal. For example, the last 10 Hankel singular values of the 100% load model are

$$\sigma_{16-25} = \{0.64046, 1.0852 \cdot 10^{-4}, 0, 0, 0, 0, 0, 0, 0, 0, \}$$

and the Hankel-norm for this model is $3.4078 \cdot 10^5$. The same qualitative results are true for the other two models. The computed three 16 order reduced models can be practically not distinguished from the original models on basis of time or frequency responses. Several lower order approximations have been also computed of orders 6, 8 and 12. The 12 order models represent very good approximation of the original models and can serve as basis for designing a unique robust controller ensuring satisfactory performance for all three models. A comparison on basis of elements $g_{35}(s)$ of the corresponding transfer-function matrices is shown in Figure 7.8.

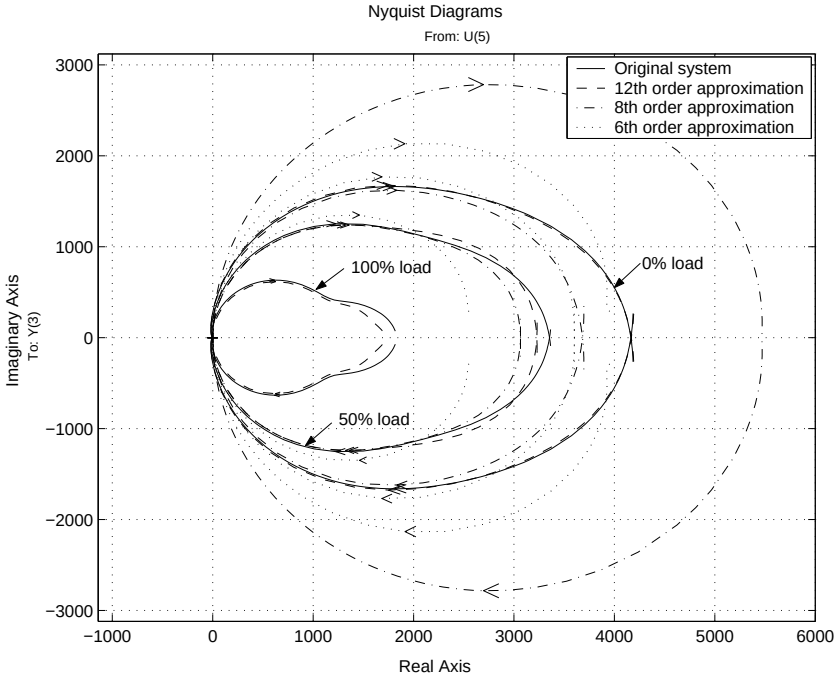


**FIGURE 7.8. Frequency responses for $g_{35}(s)$ elements of GAS models.**

## 7.6 Comparison of available model reduction tools

We present shortly the model reductions tools available in other control packages and compare them with the model reduction tools provided in SLICOT. A summary of capabilities of the reviewed software is presented in the Table 7.3. All reported test results for SLICOT have been obtained via the *mex*-function `sysred`.

**TABLE 7.3. Summary of comparison of model reduction tools.**

| Software | *RASP* | *SLICOT* | `Scilab` | *Control Toolbox* | *Robust Toolbox* | *$\mu$-Toolbox* | *H2Tools* | *MatrixX* | *WOR–Toolbox* |
|---|---|---|---|---|---|---|---|---|---|
| Provided features | | | | | | | | | |
| continuous-time | + | + | + | + | + | + | + | + | + |
| discrete-time | + | + | + | + | – | – | + | – | – |
| unstable | + | + | + | – | + | – | – | – | + |
| non-minimal | + | + | + | – | + | + | + | + | + |
| Methods | | | | | | | | | |
| balancing | + | + | + | + | + | + | + | + | + |
| balancing-free (BF) | + | + | + | – | + | – | + | + | – |
| square-root (SR) | + | + | + | – | – | + | + | – | + |
| BF-SR | + | + | + | – | – | – | + | – | – |
| Problem classes | | | | | | | | | |
| additive error | + | + | + | + | + | + | + | + | + |
| relative error | + | + | – | – | + | + | + | + | – |
| frequency weighted | + | + | – | – | – | + | – | + | + |
| controller reduction | – | + | – | – | – | – | – | + | + |

### *7.6.1 RASP*

RASP is the control system design library of DLR[8]. The model reduction tools available in RASP belong to the Fortran subroutines package RASP-MODRED [23]. Since the standardized SLICOT model reduction software originates mainly from equivalent RASP-MODRED routines, there is no significant difference between the provided functionalities in RASP-MODRED and SLICOT for the class of additive error methods. The stan-

[8]German Aerospace Center

dardized SLICOT implementations are slightly more efficient and possibly also numerically more robust than the original codes from RASP-MODRED, because of extensive error checking and more robust implementation of several supporting mathematical routines. RASP-MODRED contains additionally routines for relative error model reduction based on the balanced stochastic truncation approach as well as some supporting software to allow frequency weighted model reduction. These latter routines served as starting point to implement recently a new set of SLICOT routines covering the main aspects of the controller reduction problem. The new routines for the relative error method, frequency-weighted model reduction and controller reduction have been recently included in SLICOT.

### *7.6.2* `Scilab`

In the previous versions of `Scilab` no dedicated model reduction functions were available. Still model reduction was possible using several supporting functions to compute gramians, Hankel-singular values, factors of the projection associated with the small eigenvalues of the product of gramians, and computation of projected systems. The current version of `Scilab` relies on the new SLICOT routines and thus provides the same model reduction functionality as that available for MATLAB (see section 7.3).

### *7.6.3* MATLAB *Control Toolbox*

The function `balreal`, available in Version 5.0, performs balancing by computing first the gramians as solutions to appropriate Lyapunov equations and then determines the Cholesky factors of the gramians. There are several problems with this function. First, to use `balreal`, the original system must be minimal. This is a serious limitation since, many systems are numerically almost non-minimal, and this leads very often to the failure of this function. For instance, this is the cause of failures reported in section 7.4.7 for random stable systems with orders larger than 25-30, which are certainly minimal. At the algorithmic level, the problem is the consequence of computing first the gramians and using them to compute the Cholesky factors. Without exploiting the problem structure (i.e., without computing directly the Cholesky factors) this approach is numerically unreliable because, due to roundoff, the computed gramians could become numerically indefinite for nearly uncontrollable or unobservable systems. This leads then automatically to the failure of Cholesky factorization function. Second, by performing always balancing, additional accuracy loss can occur in the case of poorly scaled systems. Further, since no scaling is performed by this function, for badly scaled problems the results can be very inaccurate (see also Section 7.4.5). Finally, there is no support in the toolbox for reduction of unstable systems.

### *7.6.4* MATLAB *Robust Control Toolbox*

There are several model reduction tools in the Robust Control Toolbox, Version 2.0 [4], which cover similar model reduction problems as `sysred`. The BTA method is implemented in the functions `obalreal`, `balmr` and `schmr` and the optimal HNA method is implemented in `ohklmr`. Only continuous-time systems can be reduced and for reduction of discrete-time systems bilinear transformation techniques are recommended to be used. `obalreal` is practically the same implementation as `balreal` from the Control Toolbox, thus has the same limitations. `balmr`, `schmr` and `ohklmr` can be applied also to non-minimal systems as well as to unstable systems. Without preliminary scaling, all this routines fail to compute accurate Hankel singular values for the ACT model. They also fail on unstable models with eigenvalues on the imaginary axis like ATTAS. On a Pentium II 400 MHz PC, `schmr` needed 17.2 sec to compute a 10-th order approximation for the 120-th order CDP model. In comparison, `sysred` performed the same computation in 0.27 sec.

Besides additive error methods, there are two functions for relative error model reduction via balanced stochastic truncation. These functions are better suited to approximate uniformly the frequency responses than the additive error methods. In particular, phase information is better approximated, thus approximations of minimum-phase models lead often to minimum-phase reduced models. This functionality is covered by the newest additions to SLICOT and will be presented elsewhere.

### *7.6.5* MATLAB *$\mu$-Analysis and Synthesis Toolbox*

There are several model reduction tools in the $\mu$-Analysis and Synthesis Toolbox, Version 3.0 [2], which cover similar model reduction problems as `sysred`. The square-root BTA method is implemented in the functions `sysbal` and the optimal HNA method is implemented in `hankmr`. Only continuous-time stable systems can be reduced and for the reduction of discrete-time systems bilinear transformation techniques are recommended to be used. `sysbal` and `hankmr` can be applied also to non-minimal systems. Even without a preliminary scaling, `sysbal` was able to compute up to 5 digits accuracy the Hankel singular values for the ACT model. On a Pentium II 400 MHz PC, `schmr` needed 2.8 sec to compute a 10-th order approximation for the 120-th order CDP model.

Besides additive error methods, there are functions for relative error model reduction via balanced stochastic truncation, frequency weighted model reduction and normalized coprime factorization based model reduction.

### *7.6.6* HTOOLS

The model reduction tools in the HTOOLS ($H_\infty$–Tools) Toolbox for MATLAB [25] have been implemented having as basis the numerical algorithms with enhanced accuracy described in [23]. Both continuous- and discrete-time models can be reduced and a suite of square-root and balancing-free square-root algorithms are implemented for both additive as well as for stochastic balancing based relative methods. The structure exploiting careful implementations of all functions ensured practically the same numerical performances as those of robust Fortran implementations available in RASP-MODRED and now also in SLICOT. For instance, positive Lyapunov solvers are implemented using the Hammarling's algorithms in both continuous- as well as discrete-time variants. Still, the very detailed structure exploiting implementations have the consequence of much larger execution times as those of equivalent Fortran implementations. This aspect is common to all model reduction tools implemented exclusively in MATLAB.

### *7.6.7* MATRIX$_X$

The model reduction functions in the MATRIX$_X$ Model Reduction Module [13] are very similar to those available in the Robust Control Toolbox of MATLAB. No functions are provided to handle directly discrete-time or unstable systems. Besides additive model reduction methods, also relative error and frequency weighted methods are implemented. A comprehensive documentation is provided with the package, which clearly presents the restrictions of the module and offers valuable hints to overcome some of them.

### *7.6.8* WOR–*Toolbox*

A collection of model reduction functions for frequency *weighted order reduction* forms the WOR–Toolbox for MATLAB. This toolbox has been implemented in connection with the Ph.D. thesis of Wortelboer [27] and provides functions implementing several non-standard methods for model and controller reduction. Functions are available for $H_2$-norm reduction, balanced modal reduction, reduction of unstable systems using normalized coprime factorization. Functions for interactive order reduction in a user-defined configuration are also provided. This comprises both open-loop and closed-loop, as well as frequency weighted and unweighted configurations. The WOR–Toolbox calls some low level functions of the $\mu$-Toolbox.

## 7.7 Summary of results and perspectives

The model reduction tools of SLICOT consists of a functionally reach collection of standardized, comprehensively documented and fully tested Fortran routines implementing rigorously selected methods for order reduction of continuous-/discrete-time, stable/unstable linear time-invariant systems. The final model reduction package consists of 9 user-callable routines and 3 main supporting routines. All these routines are thoroughly documented. The documentation is automatically generated from the comments in the preamble of each routine (see Appendix 7.A for `AB09MD`). The documentation is available in *html*-format and can be viewed with standard browsers like Windows Internet Explorer or Netscape. The documentation also includes for each user callable routine a test program example, test data and the corresponding test results. The documentation of all library routines can be accessed on-line via the ftp-site of NICONET.[9]

Besides standardized Fortran routines, the SLICOT model reduction tools include interface software to two popular user-friendly CACSD environments: MATLAB and `Scilab`. A special *mex*-function `sysred` has been implemented as Fortran interface to MATLAB to provide access to all facilities available in the SLICOT routines. This *mex*-function also served to prepare the interface software for `Scilab`. Additionally, 9 easy-to-use *m*-functions (see Section 7.3.1) have been implemented. They fully exploit the advanced object oriented facilities available both in MATLAB Control Toolbox as well as in `Scilab` to manipulate control objects. Standard help facilities for the *mex*-function and *m*-functions are available both for MATLAB and `Scilab`.

Two main directions are envisaged to continue the efforts to develop reliable numerical model reduction software for SLICOT. The first direction focuses on the reduction of very high order systems using special implementations exploiting parallel architecture machines. The second direction continues the efforts to develop model reduction software for relative error methods and frequency weighted problems, with the main objective to have a powerful collection of tools for controller reduction. A powerful set of routines together with MATLAB interfaces has been recently implemented based on a preliminary selection of routines[10] and is already available on the SLICOT ftp-site. The controller reduction software complements the $H_2/H_\infty$ software developed recently for SLICOT[11].

[9] `ftp://wgs.esat.kuleuven.ac.be/pub/WGS/SLICOT/libindex.html`
[10] see `ftp://wgs.esat.kuleuven.ac.be/pub/WGS/REPORTS/SLWN1999-18.ps.Z`
[11] see `ftp://wgs.esat.kuleuven.ac.be/pub/WGS/REPORTS/SLWN1999-12.ps.Z`

## 7.A Sample user interface in Fortran

```
      SUBROUTINE AB09MD( DICO, JOB, EQUIL, ORDSEL, N, M, P, NR,
     $                   ALPHA, A, LDA, B, LDB, C, LDC, NS, HSV,
     $                   TOL, IWORK, DWORK, LDWORK, IWARN, INFO )
C
C  RELEASE 4.0, WGS COPYRIGHT 1999.
C
C  PURPOSE
C
C  To compute a reduced order model (Ar,Br,Cr) for an original
C  state-space representation (A,B,C) by using either the
C  square-root or the balancing-free square-root
C  Balance & Truncate (B & T) model reduction method for
C  the ALPHA-stable part of the system.
C
C  ARGUMENTS
C
C  Mode Parameters
C
C  DICO    CHARACTER*1
C          Specifies the type of the original system as follows:
C          = 'C':  continuous-time system;
C          = 'D':  discrete-time system.
C
C  JOB     CHARACTER*1
C          Specifies the model reduction approach to be used
C          as follows:
C          = 'B':  use the square-root Balance & Truncate method;
C          = 'N':  use the balancing-free square-root
C                  Balance & Truncate method.
C
C  EQUIL   CHARACTER*1
C          Specifies whether the user wishes to preliminarily
C          equilibrate the triplet (A,B,C) as follows:
C          = 'S':  perform equilibration (scaling);
C          = 'N':  do not perform equilibration.
C
C  ORDSEL  CHARACTER*1
C          Specifies the order selection method as follows:
C          = 'F':  the resulting order NR is fixed;
C          = 'A':  the resulting order NR is automatically
C                  determined on basis of the given tolerance TOL.
C
C  Input/Output Parameters
C
C  N       (input) INTEGER
C          The order of the original state-space representation,
C          i.e. the order of the matrix A.  N >= 0.
```

```
C
C  M       (input) INTEGER
C          The number of system inputs.  M >= 0.
C
C  P       (input) INTEGER
C          The number of system outputs.   P >= 0.
C
C  NR      (input/output) INTEGER
C          On entry, with ORDSEL = 'F', NR is the desired order of
C          the resulting reduced order system.
C          On exit, if INFO = 0, NR is the order of the resulting
C          reduced order model. For a system with NU ALPHA-unstable
C          eigenvalues and NS ALPHA-stable eigenvalues (NU+NS=N),
C          NR is set as follows: if ORDSEL = 'F', NR is equal to
C          NU+MIN(MAX(0,NR-NU),NMIN), where NR is the desired order
C          on entry, and NMIN is the order of a minimal realization
C          of ALPHA-stable part of the given system;
C          NMIN is determined as the number of Hankel singular
C          values greater than NS*EPS*HNORM(As,Bs,Cs), where EPS
C          is the machine precision (see LAPACK Library Routine
C          DLAMCH) and HNORM(As,Bs,Cs) is the Hankel norm of the
C          ALPHA-stable part of the given system computed in HSV(1);
C          if ORDSEL = 'A', NR is the sum of NU and the number of
C          Hankel singular values greater than
C          MAX(TOL,NS*EPS*HNORM(As,Bs,Cs)).
C
C  ALPHA   (input) DOUBLE PRECISION.
C          Specifies the ALPHA-stability boundary for the
C          eigenvalues of the state dynamics matrix A.
C          For a continuous-time system (DICO = 'C'), ALPHA =< 0
C          is the boundary value for the real parts of eigenvalues,
C          while for a discrete-time system (DICO = 'D'),
C          1 >= ALPHA >= 0 represents the boundary value for the
C          moduli of eigenvalues.
C
C  A       (input/output) DOUBLE PRECISION array, dimension (LDA,N)
C          On entry, the leading N-by-N part of this array must
C          contain the state dynamics matrix A.
C          On exit, if INFO = 0, the leading NR-by-NR part of this
C          array contains the state dynamics matrix Ar of the
C          reduced order system. The resulting A has a block
C          diagonal form with two blocks.
C          For a system with NU ALPHA-unstable eigenvalues and
C          NS ALPHA-stable eigenvalues (NU+NS=N), the leading
C          NU-by-NU block contains the unreduced part of A
C          corresponding to ALPHA-unstable eigenvalues in an
C          upper real Schur form.
C          The trailing (NR+NS-N)-by-(NR+NS-N) block contains
C          the reduced part of A corresponding to ALPHA-stable
```

```
C          eigenvalues.
C
C  LDA     INTEGER
C          The leading dimension of array A.  LDA >= MAX(1,N).
C
C  B       (input/output) DOUBLE PRECISION array, dimension (LDB,M)
C          On entry, the leading N-by-M part of this array must
C          contain the original input/state matrix B.
C          On exit, if INFO = 0, the leading NR-by-M part
C          of this array contains the input/state matrix Br of
C          the reduced order system.
C
C  LDB     INTEGER
C          The leading dimension of array B.  LDB >= MAX(1,N).
C
C  C      (input/output) DOUBLE PRECISION array, dimension (LDC,N)
C          On entry, the leading P-by-N part of this array must
C          contain the original state/output matrix C.
C          On exit, if INFO = 0, the leading P-by-NR part
C          of this array contains the state/output matrix Cr of
C          the reduced order system.
C
C  LDC  INTEGER
C          The leading dimension of array C. LDC >= MAX(1,P).
C
C  NS      INTEGER
C          (output) The dimension of the ALPHA-stable subsystem.
C
C  HSV     (output) DOUBLE PRECISION array, dimension (N)
C          If INFO = 0, the leading NS elements of HSV contains
C          the Hankel singular values of the ALPHA-stable part of
C          the original system ordered decreasingly.
C          HSV(1) is the Hankel norm of the ALPHA-stable
C          subsystem.
C
C  Tolerances
C
C  TOL     DOUBLE PRECISION
C          If ORDSEL = 'A', TOL contains the tolerance for
C          determining the order of reduced system.
C          For model reduction, the recommended value is
C          TOL = c*HNORM(As,Bs,Cs), where c is a constant in the
C          interval [0.00001,0.001], and HNORM(As,Bs,Cs) is the
C          Hankel-norm of the ALPHA-stable part of the given system
C          (computed in HSV(1)).
C          If TOL <= 0 on entry, the used default value is
C          TOL = NS*EPS*HNORM(As,Bs,Cs), where NS is the number of
C          ALPHA-stable eigenvalues of A and EPS is the
C          machine precision (see LAPACK Library Routine DLAMCH).
```

```
C          This value is appropriate to compute a minimal
C          realization of the ALPHA-stable part.
C          If ORDSEL = 'F', the value of TOL is ignored.
C
C  Workspace
C
C  IWORK   INTEGER array, dimension (LIWORK)
C          LIWORK = 0, if JOB = 'B';
C          LIWORK = N, if JOB = 'N'.
C
C  DWORK   DOUBLE PRECISION array, dimension (LDWORK)
C          On exit, if INFO = 0, DWORK(1) returns the optimal value
C          of LDWORK.
C
C  LDWORK  INTEGER
C          The length of the array DWORK.
C          LDWORK >= MAX(1,N*(2*N+MAX(N,M,P)+5) + N*(N+1)/2).
C          For optimum performance LDWORK should be larger.
C
C  Warning Indicator
C
C  IWARN   INTEGER
C          = 0: no warning;
C          = 1: with ORDSEL = 'F' the selected order NR is greater
C               than NSMIN, the sum of order of the ALPHA-unstable
C               part and the order of a minimal realization of the
C               ALPHA-stable part of the given system. In this
C               case, the resulting NR is set equal to NSMIN.
C          = 2  with ORDSEL = 'F' the selected order NR is less
C               than the order of the ALPHA-unstable part of the
C               given system. In this case NR is set equal to the
C               order of the ALPHA-unstable part.
C
C  Error Indicator
C
C  INFO    INTEGER
C          = 0: successful exit;
C          < 0: if INFO = -i, the i-th argument had an illegal
C               value;
C          = 1: the computation of the ordered real Schur form of A
C               failed;
C          = 2: the separation of the ALPHA-stable/unstable diagonal
C               blocks failed because of very close eigenvalues;
C          = 3: the computation of Hankel singular values failed.
C
C  METHOD
C
C  Let be the following linear system
C
```

```
C       d[x(t)] = Ax(t) + Bu(t)
C       y(t)    = Cx(t)                                (1)
C
C  where d[x(t)] is dx(t)/dt for a continuous-time system and x(t+1)
C  for a discrete-time system. The subroutine AB09MD determines for
C  the given system (1), the matrices of a reduced order system
C
C       d[z(t)] = Ar*z(t) + Br*u(t)
C       yr(t)   = Cr*z(t)                              (2)
C
C  such that
C
C  HSV(NR+NS-N) <= INFNORM(G-Gr) <= 2*[HSV(NR+NS-N+1)+...+HSV(NS)],
C
C  where G and Gr are transfer-function matrices of the systems
C  (A,B,C) and (Ar,Br,Cr), respectively, and INFNORM(G) is the
C  infinity-norm of G.
C
C  The following procedure is used to reduce a given G:
C
C  1) Decompose additively G as
C
C       G = G1 + G2
C
C     such that G1 = (As,Bs,Cs) has only ALPHA-stable poles and
C     G2 = (Au,Bu,Cu) has only ALPHA-unstable poles.
C
C  2) Determine G1r, a reduced order approximation of the
C     ALPHA-stable part G1.
C
C  3) Assemble the reduced model Gr as
C
C        Gr = G1r + G2.
C
C  To reduce the ALPHA-stable part G1, if JOB = 'B' the square-root
C  Balance & Truncate method of [1] is used and for a ALPHA-stable
C  continuous-time system (DICO = 'C'), the resulting reduced model
C  is balanced. For ALPHA-stable systems, setting TOL < 0, the
C  routine can be used to compute balanced minimal state-space
C  realizations. If JOB = 'N' the square-root balancing-free version
C  of the Balance & Truncate  method is used [2] to reduce the
C  ALPHA-stable part G1.
C
C  REFERENCES
C
C  [1] Tombs M.S. and Postlethwaite I.
C      Truncated balanced realization of stable, non-minimal
C      state-space systems.
C      Int. J. Control, Vol. 46, pp. 1319-1330, 1987.
```

```
C
C  [2] Varga A.
C      Efficient minimal realization procedure based on balancing.
C      Proc. of IMACS/IFAC Symp. MCTS, Lille, France, May 1991,
C      Eds. A. El Moudni, P. Borne, S. G. Tzafestas,
C      Vol. 2, pp. 42-46.
C
C  NUMERICAL ASPECTS
C
C  The implemented methods rely on accuracy enhancing square-root
C  or balancing-free square-root techniques.
C                                      3
C  The algorithms require less than 30N  floating point
C  operations.
C
C  .. Scalar Arguments ..
      CHARACTER         DICO, EQUIL, JOB, ORDSEL
      INTEGER           INFO, IWARN, LDA, LDB, LDC, LDWORK, M, N,
     $                  NR, NS, P
      DOUBLE PRECISION  ALPHA, TOL
C  .. Array Arguments ..
      INTEGER           IWORK(*)
      DOUBLE PRECISION  A( LDA, * ), B( LDB, * ), C( LDC, * ),
     $                  DWORK( * ), HSV(*)
C
```

## 7.B MATLAB *mex*-function interface

```
%SYSRED  MEX-function based on SLICOT model reduction routines.
%   [Ar,Br,Cr,Dr,HSV] = SYSRED(METH,A,B,C,D,TOL,DISCR,ORDER,ALPHA)
%
%   SYSRED returns for an original continuous- or discrete-time
%   state-space system (A,B,C,D) a reduced order state space
%   system (Ar,Br,Cr,Dr) and the Hankel singular values HSV of
%   the ALPHA-stable part. The order of the reduced model is
%   determined either by the number of Hankel-singular values
%   greater than TOL or by the desired order ORDER.
%
%   Description of other input parameters:
%   METH  - method flag with decimal form c*10+m, where:
%             m specifies the basic model reduction method;
%             c specifies the comprime factorization approach to be
%               used in conjunction with the method specified by m.
%           Allowed values for m:
%             m = 1 : for Balance & Truncate method with balancing
%             m = 2 : for Balance & Truncate method (no balancing)
%             m = 3 : Singular Perturbation Approximation with
%                     balancing
%             m = 4 : Singular Perturbation Approximation
%                    (no balancing)
%             m = 5 : Optimal Hankel-Norm Approximation
%           Allowed values for c (only for m = 1..4):
%             c = 0 : no coprime factorization is used (default)
%             c = 1 : RCF with inner denominator
%             c = 2 : LCF with inner denominator
%             c = 3 : RCF with ALPHA stability degree
%             c = 4 : LCF with ALPHA stability degree
%   TOL   - (optional) tolerance vector for determining the order of
%           reduced system of form TOL = [tol1, tol2, tol3], where:
%           tol1 specifies the tolerance for model reduction;
%                default: tol1 = epsilon_machine*Hankel_norm(A,B,C)
%           tol2 specifies the tolerance for minimal realization in
%                case of m = 3, 4 or 5
%                default: tol2 = epsilon_machine*Hankel_norm(A,B,C)
%           tol3 specifies the controllability/observability
%                tolerance for computing coprime factorizations,
%                as follows:
%                controllability tolerance in the case c = 1 or 3
%                default: epsilon_machine*max(norm(A),norm(B))
%                observability tolerance in the case c = 2 or 4
%                default: epsilon_machine*max(norm(A),norm(C))
%   ORDER - (optional) desired order of reduced system
%             default: ORDER = -1 (order determined automatically)
%   DISCR - (optional) type of system
%              = 0 : continuous-time (default)
```

```
%              = 1 : discrete-time
%   ALPHA - (optional) stability boundary for the eigenvalues of A
%             default: sqrt(epsilon_machine)  for continuous-time
%                      1-sqrt(epsilon_machine)  for discrete-time
```

## 7.C Sample MATLAB *m*-function interface

```
function [sysr,hsv] = bta(sys,tol,ord,alpha)
%BTA  Balance & Truncate approximation without balancing.
%     [SYSR,HSV] = BTA(SYS,TOL,ORD,ALPHA)  calculates for the
%     transfer function
%                                      -1
%             G(lambda) =  C(lambdaI-A) B + D
%
%     of an original system SYS = (A,B,C,D), an approximate
%     transfer function
%                                        -1
%             Gr(lambda) =  Cr(lambdaI-Ar) Br + Dr
%
%     of a reduced order system SYSR = (Ar,Br,Cr,Dr) using the
%     balancing-free square-root Balance & Truncate approximation
%     method on the ALPHA-stable part of SYS
%     (see Method with 'type bta').
%
%     TOL is the tolerance for model reduction.
%
%     ORD specifies the desired order of the reduced system SYSR.
%
%     ALPHA is the stability boundary for the eigenvalues of A.
%     For a continuous-time system ALPHA <= 0 is the boundary value
%     for the real parts of eigenvalues, while for a discrete-time
%     system, 1 >= ALPHA >= 0 represents the boundary value for the
%     moduli of eigenvalues.
%
%     HSV contains the decreasingly ordered Hankel singular values
%     of the ALPHA-stable part of SYS.
%
%     The order NR of the reduced system SYSR is determined as
%     follows: let NU be the order of the ALPHA-unstable part of
%     SYS and let NSMIN be the order of a minimal realization of
%     the ALPHA-stable part. Then
%     (1) if TOL > 0 and  ORD < 0, then NR = NU + min(NRS,NSMIN),
%         where NRS is the number of Hankel singular values
%         greater than TOL;
%     (2) if ORD >= 0, then NR = NU+MIN(MAX(0,ORD-NU),NSMIN).
%
%     SYSR = BTA(SYS) calculates for a stabilizable and detectable
```

```
%     system SYS a minimal state-space realization SYSR.

%     Method:
%     The following approach is used to reduce a given G:
%
%     1) Decompose additively G as
%
%          G = G1 + G2
%
%        such that G1 = (As,Bs,Cs,D) has only ALPHA-stable poles
%        and G2 = (Au,Bu,Cu,0) has only ALPHA-unstable poles.
%
%     2) Determine G1r, a reduced order approximation of the
%        ALPHA-stable part G1 using the balancing-free
%        square-root Balance & Truncate Approximation method.
%
%     3) Assemble the reduced model Gr as
%
%           Gr = G1r + G2.
%
%     Interface M-function to the SLICOT-based MEX-function SYSRED.
%     A. Varga 04-05-1998.

if ~isa(sys,'lti')
   error('The input system SYS must be an LTI object')
end

ni = nargin; discr = sys.ts > 0;
if ni < 4
   alpha = -sqrt(eps);
   if discr
      alpha = 1 + alpha;
   end
end
if ni < 3
   ord = -1;
end
if ni < 2
   tol = 0;
end

[a,b,c,d]=ssdata(sys);
[ar,br,cr,dr,hsv]=sysred(2,a,b,c,d,tol,discr,ord,alpha);
sysr = ss(ar,br,cr,dr,sys);

% end bta
```

## 7.D Sample `Scilab` *sci*-function interface

```
function [sysr,hsv] = bta(sys,tol,ord,alpha)

[no,ni]=argn(0);
if ~typeof(sys)=='state-space'
   error('The input system SYS must be a state-space system')
end

discr = bool2s( sys('dt')=='d' );
if ni < 4
   alpha = -sqrt(%eps);
   if discr
      alpha = 1 + alpha;
   end
end
if ni < 3
   ord = -1;
end
if ni < 2
   tol = 0;
end

[a,b,c,d]=abcd(sys);
[ar,br,cr,dr,hsv]=sysred(2,a,b,c,d,tol,discr,ord,alpha);
sysr = syslin( sys('dt'),ar,br,cr,dr);

// end bta
```

## 7.E State space models for benchmark problems

| | |
|---|---|
| Name | PS |
| Description | Power system model |
| Reference | [7] |
| Type | State space model, continuous-time |
| # states | 7 |
| # control inputs | 2 |
| # disturbance inputs | 2 |
| # outputs | 3 |

$$A = \begin{bmatrix} -0.04165 & 0 & 4.92 & -4.92 & 0 & 0 & 0 \\ -5.21 & -12.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.33 & -3.33 & 0 & 0 & 0 & 0 \\ 0.545 & 0 & 0 & 0 & -0.545 & 0 & 0 \\ 0 & 0 & 0 & 4.92 & -0.04165 & 0 & 4.92 \\ 0 & 0 & 0 & 0 & -5.21 & -12.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3.33 & -3.33 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} 0 & 0 \\ 12.5 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 12.5 \\ 0 & 0 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} -4.92 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -4.92 \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \qquad D = 0\,.$$

| | |
|---|---|
| Name | PSD |
| Description | Sampled-data power system model ($T = 0.1$ sec) |
| Reference | [7] |
| Type | State space model, discrete-time |
| # states | 7 |
| # control inputs | 2 |
| # disturbance inputs | 2 |
| # outputs | 3 |

$$A = \begin{bmatrix} 0.97277 & 0.049697 & 0.41432 & -0.48531 & 0.013281 & 0.00013525 & 0.002015 \\ -0.29382 & 0.2793 & -0.077754 & 0.087312 & -0.0017394 & -1.2118e{-5} & -0.00021161 \\ -0.052626 & 0.15561 & 0.7078 & 0.0097701 & -0.00014322 & -6.829e{-7} & -1.3998e{-5} \\ 0.053759 & 0.001022 & 0.011948 & 0.97337 & -0.053759 & -0.001022 & -0.011948 \\ 0.013281 & 0.00013525 & 0.002015 & 0.48531 & 0.97277 & 0.049697 & 0.41432 \\ -0.0017394 & -1.2118e{-5} & -0.00021161 & -0.087312 & -0.29382 & 0.2793 & -0.077754 \\ -0.00014322 & -6.829e{-7} & -1.3998e{-5} & -0.0097701 & -0.052626 & 0.15561 & 0.7078 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} 0.023476 & 3.5533e{-5} \\ 0.71091 & -2.6922e{-6} \\ 0.12681 & -1.2878e{-7} \\ 0.00034405 & -0.00034405 \\ 3.5533e{-5} & 0.023476 \\ -2.6922e{-6} & 0.71091 \\ -1.2878e{-7} & 0.12681 \end{bmatrix}, \; B_2 = \begin{bmatrix} -0.4875 & -0.0021858 \\ 0.087539 & 0.00022642 \\ 0.0097849 & 1.481e{-5} \\ -0.013314 & 0.013314 \\ -0.0021858 & -0.4875 \\ 0.00022642 & 0.087539 \\ 1.481e{-5} & 0.0097849 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad D = 0 \, .$$

| | |
|---|---|
| Name | TGEN |
| Description | Nuclear-powered turbo-generator |
| Reference | [10] |
| Type | State space model, continuous-time |
| # states | 10 |
| # control inputs | 2 |
| # disturbance inputs | 0 |
| # outputs | 2 |

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -0.11323 & -0.98109 & -11.847 & -11.847 & -63.08 \\ 324.121 & -1.1755 & -29.101 & 0.12722 & 2.83448 & -967.73 \\ -127.3 & 0.46167 & 11.4294 & -1.0379 & 13.1237 & 380.079 \\ -186.05 & 0.67475 & 16.7045 & 0.86092 & -17.068 & 555.502 \\ 341.917 & 1.09173 & 1052.75 & 756.465 & 756.465 & -29.774 \\ -30.748 & -0.09817 & -94.674 & -68.029 & -68.029 & 2.67753 \\ -302.36 & -0.96543 & -930.96 & -668.95 & -668.95 & 26.3292 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ -34.339 & -34.339 & -27.645 & 0 \\ -678.14 & -678.14 & 0 & -129.29 \\ 266.341 & 266.341 & 0 & 1054.85 \\ 389.268 & 389.268 & 0 & -874.92 \\ 0.16507 & 3.27626 & 0 & 0 \\ -2.6558 & 4.88497 & 0 & 0 \\ 2.42028 & -9.5603 & 0 & 0 \\ 0 & 0 & -1.66667 & 0 \\ 0 & 0 & 0 & -10 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1.66667 & 0 \\ 0 & 10 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.49134 & 0 & -0.63203 & 0 & 0 & -0.20743 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad D = 0 \, .$$

| | |
|---|---|
| Name | ACT |
| Description | Hydraulic actuator model |
| Reference | |
| Type | State space model, continuous-time |
| # states | 5 |
| # control inputs | 2 |
| # disturbance inputs | 0 |
| # outputs | 5 |

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -1580000 & -1257 & 0 & 0 & 0 \\ 3.541e{+}14 & 0 & -1434 & 0 & -5.33e{+}11 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -18630 & -1.482 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 110.3 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0.008333 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0.6664 & 0 & -6.2e{-}13 & 0 & 0 \\ 0 & 0 & -0.001 & 1896000 & 150.8 \end{bmatrix}, \quad D = 0\,.$$

## References

[1] E. Anderson, Z. Bai, J. Bishop, J. Demmel, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK User's Guide, Third Edition*. SIAM, Philadelphia, 1999.

[2] G. Balas, J. Doyle, K. Glover, A. Packard, and R. Smith. *$\mu$-Analysis and Synthesis Toolbox*. The MathWorks Inc., Natick, MA, 2000.

[3] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga. SLICOT – a subroutine library in systems and control theory. In B. N. Datta (Ed.), *Applied and Computational Control, Signals and Circuits*, vol. 1, pp. 499–539, Birkhäuser, 1999.

[4] R. Y. Chiang and M. G. Safonov. *Robust Control Toolbox*. The MathWorks Inc., Natick, MA, 1999.

[5] C. Gomez (Ed.). *Engineering and Scientific Computing with Scilab*. Birkhauser, Boston, 1999.

[6] H. Elmquist. Object-oriented modeling and automatic formula manipulation in Dymola. Scandinavian Simulation Society SIMS'93, Konsberg, Norway, June 1993.

[7] C. E. Fosha and O. I. Elgerd. The megawatt-frequency control problem: a new approach via optimal control theory. *IEEE Trans. on Power Apparatus and Systems*, 89:563–571, 1970.

[8] K. Glover. All optimal Hankel-norm approximations of linear multivariable systems and their $L^\infty$-error bounds. *Int. J. Control*, 39:1115–1193, 1974.

[9] S. J. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA J. Numer. Anal.*, 2:303–323, 1982.

[10] D. J. N. Limebeer and J. M. Maciejowski. Two tutorial examples of multivariable control system design. Technical Report CUED/F-CAMS/TR-229, Cambridge University, 1982.

[11] Y. Liu and B. D. O. Anderson. Singular perturbation approximation of balanced systems. *Int. J. Control*, 50:1379–1405, 1989.

[12] *Control System Toolbox* The MathWorks Inc., Natick, MA, 2000.

[13] MATRIX$_X$. *Xmath Model Reduction Module*. ISI, Santa Clara, CA, January 1998.

[14] B. C. Moore. Principal component analysis in linear system: controllability, observability and model reduction. *IEEE Trans. Autom. Control*, AC-26:17–32, 1981.

[15] N. Munro. Control system analysis and design using Mathematica. *Proc. CDC'98, Tampa, FL*, pp. 3681–3685, 1998.

[16] M. G. Safonov and R. Y. Chiang. A Schur method for balanced-truncation model reduction. *IEEE Trans. Autom. Control*, 34:729–733, 1989.

[17] M. S. Tombs and I. Postlethwaite. Truncated balanced realization of a stable non-minimal state-space system. *Int. J. Control*, 46:1319–1330, 1987.

[18] A. Varga. Balancing-free square-root algorithm for computing singular perturbation approximations. *Proc. CDC'91, Brighton, UK*, pp. 1062–1065, 1991.

[19] A. Varga. Efficient minimal realization procedure based on balancing. In A. El Moudni, P. Borne, and S. G. Tzafestas, (Eds.), *Prepr. of IMACS Symp. on Modelling and Control of Technological Systems*, vol. 2, pp. 42–47, 1991.

[20] A. Varga. *RASP Model Order Reduction Programs*. Technical Report, University of Bochum and DLR-Oberpfaffenhofen, TR R88-92, August 1992.

[21] A. Varga. Coprime factors model reduction based on accuracy enhancing techniques. *Systems Analysis Modelling and Simulation*, 11:303–311, 1993.

[22] A. Varga. A Schur method for computing coprime factorizations with inner denominators and applications in model reduction. *Proc. ACC'93, San Francisco, CA*, pp. 2130–2131, 1993.

[23] A. Varga. Numerical methods and software tools for model reduction. In I. Troch and F. Breitenecker, (Eds.), *Proc. of 1st MATHMOD Conf., Viena*, vol. 2, pp. 226–230, 1994.

[24] A. Varga. Enhanced modal approach for model reduction. *Mathematical Modelling of Systems*, 1:91–105, 1995.

[25] A. Varga and V. Ionescu. HTOOLS - A Toolbox for solving $H_\infty$ and $H_2$ synthesis problems. *Proc. of IFAC/IMACS Symp. on Computer Aided Design of Control Systems, Swansea, UK*, pp. 508–511, 1991.

[26] A. Varga and G. Looye. Symbolic and numerical software tools for LFT-based low order uncertainty modeling. *Proc. CACSD'99 Symposium, Kohala Coast, Hawaii*, 1999.

[27] P. Wortelboer. *Frequency-weighted Balanced Reduction of Closed-loop Mechanical Servo-systems: Theory and Tools*. PhD thesis, Technical University Delft, 1994.

# Index