NICONET Report 1999-8

Model Reduction Routines for SLICOT[1]

Andras Varga[2]

December 1999

[1] This document presents research results of the European Community BRITE-EURAM III Thematic Networks Programme NICONET (contract number BRRT-CT97-5040) and is distributed by the Working Group on Software WGS. *WGS secretariat*: Mrs. Ida Tassens, ESAT - Katholieke Universiteit Leuven, K. Mercierlaan 94, 3001-Leuven-Heverlee, BELGIUM. This report is also available by anonymous ftp from *wgs.esat.kuleuven.ac.be* in the directory *pub/WGS/REPORTS/nic1999-8.ps.Z*

[2] Deutsches Zentrum für Luft und Raumfahrt, Institut für Robotik und Mechatronik, DLR Oberpfaffenhofen, Postfach 1116, D-82230 Wessling, Germany

Contents

1 Introduction **1**

2 Selection and collation of subroutines for standardization **1**

2.1 Overview of balancing related model reduction methods 1

2.2 Algorithms for stable and unstable system 3

2.3 Standardization of software for model reduction 5

2.4 Additional software tools for model reduction 7

3 Integration of subroutines in user-friendly environments **7**

3.1 Integration in MATLAB 8

3.2 Integration in Scilab 9

4 Selection of benchmark problems, testing and performance comparisons **9**

4.1 PS: Power system model – continuous-time. 10

4.2 PSD: Power system model – discrete-time. 12

4.3 TGEN: Nuclear plant turbo-generator model. 12

4.4 PSU: Unstable continuous-time model. 13

4.5 ACT: Actuator model 13

4.6 Randomly generated systems 14

4.7 Further test problems. 15

5 Testing on industrial benchmark problems and comparisons with currently used software **15**

5.1 Industrial benchmark problems 15

5.1.1 ATTAS: Linearized aircraft model 16

5.1.2 CDP: CD-player finite element model 17

5.1.3 GAS: Gasifier model 18

5.2 Comparison with other packages 19

5.2.1 Model reduction tools in ANDECS. 19

5.2.2 Model reduction tools in MATLAB Control Toolbox V4.2. 20

5.2.3 Model reduction tools in MATLAB Robust Control Toolbox V2.0 20

5.2.4 Model reduction tools in MATLAB μ-Analysis and Synthesis V3.0 21

5.2.5 Model reduction tools in MATRIX$_X$ Model Reduction Module 21

5.2.6 Other available model reduction tools 21

6 Summary of achieved results and perspectives **22**

A List of standardized model reduction routines **25**

B List of standardized routines related to model reduction **26**

C MATLAB interface to model reduction routines **28**

D Sample MATLAB *m*-function for B&T model reduction **29**

E Sample Scilab *sci*-function for B&T model reduction **31**

F Benchmark problems for model reduction **32**

F.1 State space matrices of the PS model 32

F.2 State space matrices of the PSD model 33

F.3 State space matrices of the TGEN model 34

F.4 State space matrices of the ACT model 35

G Summary of comparisons of available model reduction tools **36**

H Preamble of AB09MD.F used for automatic documentation **37**

1 Introduction

Model reduction is of fundamental importance in many modeling and control applications. However, reliable and high quality model reduction software tools are scarce. Even the model reduction tools available in commercial packages have strong limitations because using 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 dedicated chapter of SLICOT library for model reduction. The standardization of model reduction tools on basis of the collection of routines available in the RASP-MODRED library [19] was the main objective of Task II.A. This document reports on the main developments of model reduction software for SLICOT within this task.

Three basic model reduction algorithms belonging to the class of methods based on or related to balancing techniques [13, 10, 6] 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. The basics of balancing related model reduction is presented in Section 2. Specific computational methods for reduction of stable systems and associated software available in SLICOT are presented in Section 3. Section 4 discuss the model reduction of unstable system by combining the basic methods either with coprime factorization or with spectral decomposition techniques [9]. Available SLICOT software for reduction of unstable systems is also described here. Section 5 presents new SLICOT routines implemented in conjunction with the model reduction software, while Section 6 presents available model reduction software running in user-friendly environments (MATLAB,Scilab) implemented on basis of SLICOT routines. Performance comparisons shows the superiority of SLICOT based model reduction tools over existing model reduction software.

2 Selection and collation of subroutines for standardization

The basis for developing the model reduction software for SLICOT was the SLICOT Working Note SLWN 2-1998[1], describing the selection of model reduction routines to be standardized. For concrete needs of model reduction algorithms, a selection of related software has been also performed as part of standardization activity within Task I.A.1. In this section we present an overview of implemented model reduction approaches using enhanced accuracy techniques, and of the performed standardization activities.

2.1 Overview of balancing related model reduction methods

Three basic model reduction algorithms belonging to the class of methods based on or related to balancing techniques [13, 10, 6] 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

[1] available at `ftp://wgs.esat.kuleuven.ac.be/pub/WGS/REPORTS/SLWN1998-2.ps.Z`

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 $G_r = C_r(\lambda I - A_r)^{-1}B_r + D_r$. A large class of model reduction methods can be interpreted as performing first a similarity transformation Z yielding

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

and then defining the reduced model (A_r, B_r, C_r, D_r) as the diagonal system $(A_{11},\, B_1,\, C_1,\, D)$. When writing $Z := [\, T \; U \,]$ and $Z^{-1} := [\, L^T \; V^T \,]^T$, then $\Pi = TL$ is a projector on T along L since $LT = I_r$. Thus the reduced system is $(A_r, B_r, C_r, D_r) = (LAT,\, LB,\, CT,\, D)$. Partitioned forms as above can be used to construct a so-called *singular perturbation approximation* (SPA). The matrices of the reduced model in this case are given by

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

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.

Specific requirements for model reduction algorithms are formulated and discussed in [24]. Such requirements are: (1) applicability of methods regardless the original system is minimal or not; (2) emphasis on enhancing the numerical accuracy of computations; (3) relying on numerically reliable procedures.

The first requirement can be fulfilled by computing L and T directly, without determining Z or Z^{-1}. In particular, if the original system is not minimal, then L and T can be chosen to compute an *exact* minimal realization of the original system [18].

The emphasis on improving the accuracy of computations led to so-called algorithms with *enhanced accuracy*. In many model reduction methods, the matrices L and T are determined from two positive semi-definite matrices P and Q, called generically *gramians*. The gramians can be always determined in Cholesky factorized forms $P = S^TS$ and $Q = R^TR$, where S and R are upper-triangular matrices. The computation of L and T can be done by computing the *singular value decomposition* (SVD)

$$SR^T = \left[\begin{array}{cc} U_1 & U_2 \end{array}\right] \mathrm{diag}(\Sigma_1, \Sigma_2) \left[\begin{array}{cc} V_1 & V_2 \end{array}\right]^T$$

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 so-called *square-root* (**SR**) methods determine L and T as [16]

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

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*. 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.

In order to avoid ill-conditioned projections, a *balancing-free* (**BF**) approach has been proposed in [15] in which always well-conditioned matrices L and T can be determined. These matrices are computed from orthogonal matrices whose columns span orthogonal bases for the right and left eigenspaces of the product PQ corresponding to the first r largest eigenvalues $\sigma_1^2, \ldots, \sigma_r^2$. 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 which combines the advantages of the **BF** and **SR** approaches has been introduced in [18]. L and T are 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 the QR decompositions $S^T U_1 = XW$ and $R^T V_1 = YZ$, while W and Z are non-singular upper-triangular matrices. The accuracy of the **BFSR** algorithm is usually better than either of **SR** or **BF** approaches.

The SPA formulas can be used directly on a balanced minimal order realization of the original system computed with the **SR** method. A **BFSR** method to compute SPAs has been proposed in [17]. 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.

Provided the Cholesky factors R and S are known, the computation of matrices L and T can be done by using exclusively numerically stable algorithms. Even the computation of the necessary SVD can be done without forming the product SR^T. Thus the effectiveness of the **SR** or **BFSR** techniques depends entirely on the accuracy of the computed Cholesky factors of the gramians. In the following sections we discuss the computation of these factors for several concrete model reduction approaches.

2.2 Algorithms for stable and unstable system

In the *Balance & Truncate* (B&T) method for stable systems [13] P and Q are the controllability and observability gramians satisfying a pair of continuous- or discrete-time Lyapunov equations

$$AP + PA^T + BB^T = 0, \quad A^T Q + QA + C^T C = 0;$$

$$APA^T + BB^T = P, \quad A^T QA + C^T C = Q.$$

These equations can be solved directly for the Cholesky factors of the gramians by using numerically reliable algorithms proposed in [7]. The **BFSR** version of the B&T method is described in [18]. Its **SR** version [16] can be used to compute balanced minimal representations. Such representations are also useful for computing reduced order models by using the SPA formulas [10] or the *Hankel-norm approximation* (HNA) method [6]. A **BFSR** version of the SPA method is described in [17]. Note that the B&T, SPA and HNA methods belong to the family of absolute error methods which try to minimize $\|\Delta_a\|_\infty$, where Δ_a is the absolute error $\Delta_a = G - G_r$. For an r-th order approximation, we have generally

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

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 [6] for details) such that the error bound is one half of the bound for B&T and SPA. This feature is however not available in the implemented SLICOT routine for HNA.

The reduction of unstable systems can be performed by using the methods for stable systems in conjunction with two embedding techniques. 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 $G_r = G_{1r} + G_2$.

Note that for the model reduction at step 2 any of methods available for stable systems can be used.

The second approach is based on computing a stable *rational coprime factorization* (RCF) of G. The following procedure can be used to compute an r-th order approximation G_r of an n-th order (not necessarily stable) system G:

1. Compute a left coprime factorization of the transfer-function matrix G in the form $G = M^{-1}N$, where M, N are stable and proper rational TFMs.
2. Approximate the stable system of order n $[\,N\ M\,]$ with $[\,N_r\ M_r\,]$ of order r.
3. Form the r-th order approximation $G_r = M_r^{-1}N_r$.

The coprime factorization approach used in conjunction with the B&T or BST methods fits in the general projection formulation introduced in Section 2. The gramians necessary to compute the projection are the gramians of the system $[\,N\ M\,]$. The computed matrices L and T by using either the **SR** or **BFSR** methods can be directly applied to the matrices of the original system. The main computational problem is how to compute the RCF to allow a smooth and efficient embedding which prevents computational overheads. Two factorization algorithms proposed recently compute particular RCFs which fulfill these aims: the RCF with prescribed stability degree [20] and the RCF with inner denominator [21]. Both are based on a numerically reliable Schur technique for pole assignment. The state matrix of the resulting factors is already in a real Schur form, thus the method has no overhead if the system is already stable since this reduction is always necessary even for stable systems. Note that the approximations computed for the factors of a coprime factorization with inner denominator by using the SPA method preserve these property also at the level of the reduced factors.

2.3 Standardization of software for model reduction

The basis for standardization of the model reduction routines in SLICOT formed the collection of routines available in the RASP-MODRED library [19], implemented on basis of the linear algebra standard package LAPACK [1]. The underlying algorithms represent the latest developments of various procedures for solving computational problems appearing in the context of model reduction. Most algorithms possess desirable attributes as generality, numerical reliability, enhanced accuracy, and thus are completely satisfactory to serve as bases for robust software implementations. Although most of new SLICOT routines originate of RASP-MODRED software, all routines have been practically rewritten and some of them are even completely new implementations. A special emphasis has been put on an appropriate modularization of routines, such that a set of three low level routines forms practically the basis for all user callable routines. It is worth mentioning that the available model reduction algorithms in SLICOT library are generally superior to those implemented in the model reduction tools of commercial packages [11, 3, 2, 12].

Both the **SR** and **BFSR** versions of the B&T and SPA algorithms are implemented in SLICOT library. The implementation of the HNA method uses the **SR** version of the B&T method to compute a balanced minimal realization of the original system. All implemented routines are applicable to both continuous- and discrete-time systems. Note that implementations provided in commercial software [11, 3, 2, 12] are mostly for continuous-time systems (see Appendix G).

The following routines are available in SLICOT for stable model reduction:

Name	Function
AB09AD	computes reduced (or minimal) order balanced models using either the **SR** or the **BFSR** B&T method
AB09AX	computes reduced (or minimal) order balanced models using either the **SR** or the **BFSR** B&T method (scaled system with state matrix in real Schur form)
AB09BD	computes reduced order models using the **BFSR** SPA method
AB09BX	computes reduced order models using the **BFSR** SPA method (scaled system with state matrix in real Schur form)
AB09CD	computes reduced order models using the optimal HNA method based on **SR** balancing
AB09CX	computes reduced order models using the optimal HNA method based on **SR** balancing (scaled system with state matrix in real Schur form)
AB09DD	computes a reduced order model by using the singular perturbation formulas

Three user callable routines AB09AD, AB09BD and AB09CD implement the three basic algorithms for B&T, SPA and HNA methods, respectively. All these routines perform optionally scaling of the initial 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. Three lower level routines AB09AX, AB09BX and AB09CX perform basically the same reductions as the main user callable routines, but for systems with the state matrix already reduced to the real Schur form and possibly already scaled. These lower level routines are called by the corresponding user-callable routines for reduction of both stable and unstable systems.

SLICOT provides several tools to perform the reduction of unstable system. On basis of new routines to compute left/right RCFs with prescribed stability degree or with inner denominators, or to compute additive spectral decompositions (see next paragraph), 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
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 **BFSR** B&T method in conjunction with left/right coprime factorization methods
AB09GD	computes reduced order models for unstable systems using the **BFSR** SPA method in conjunction with left/right coprime factorization methods
AB09MD	computes reduced order models for unstable systems using the B & T 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

The routines AB09ED, AB09MD and AB09ND implement the spectral separation approach in combination with HNA, B&T, and SPA 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 dominant modal reduction (see also [23]). The coprime factorization based routines AB09FD and AB09GD allows arbitrary combinations of B&T 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. Thus, these routines can be seen as completely general universal tools for order reduction of linear time-invariant systems. This is why they form the basis to implement the interface software to user-friendly environments (see Section 3).

2.4 Additional software tools for model reduction

An important number of user-callable and auxiliary routines have been implemented for the special needs of the model reduction routines. To evaluate the approximation errors for the resulting reduced order models, different transfer matrix norms are necessary to be computed. 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
AB13AX	computes the Hankel norm of a stable system with the state matrix in real Schur form
AB13BD	computes the H_2- or L_2-norm of a transfer-function matrix

A routine to compute the H_∞ norm is presently in standardization.

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

Name	Function
TB01KD	computes the terms G_1 and G_2 of an additive spectral decomposition of a transfer-function matrix G 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 auxiliary routines for the needs of model reduction is given in Appendix B.

3 Integration of subroutines 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 environment 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/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 use them in `Scilab`. It is to be expected that generally MATLAB *mex*-functions could serve a starting points for other similar environments (e.g., Matrix_X).

3.1 Integration in MATLAB

One important consideration implementing *mex*-functions is to keep their total size as small as possible. Since the standardized model reduction programs in SLICOT share many routines from BLAS, LAPACK and even from 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 interface to the model reduction routines AB09MD, AB09ND, AB09ED, AB09FD, AB09GD for reduction of stable/unstable linear systems using the B&T, SPA and HNA methods in conjunction with stable coprime factorization and stable/unstable spectral decomposition. The MATLAB help function of this function is listed in Appendix C.

To provide a convenient interface to work with control objects defined in the Control Toolbox, several easy-to-use interface functions have been additionally implemented explicitly addressing some of supported features. The following table contains the list of implemented *m*-functions. A sample *m*-function, `bta.m`, is listed in Appendix D.

Table 1: MATLAB *m*-functions for model reduction

Name	Function
`bta`	for balancing-free square-root B & T method in combination with stable/unstable additive spectral decomposition
`btabal`	for square-root B & T method in combination with stable/unstable additive spectral decomposition
`bta_cf`	for balancing-free square-root B & T method in combination with stable left/right coprime factorizations
`btabal_cf`	for square-root B & T method in combination with stable left/right coprime factorizations
`spa`	for balancing-free square-root SPA in combination with stable/unstable additive spectral decomposition
`spabal`	for square-root SPA in combination with stable/unstable additive spectral decomposition
`spa_cf`	for balancing-free square-root SPA in combination with stable left/right coprime factorizations
`spabal_cf`	for square-root Singular Perturbation Approximation in combination with stable left/right coprime factorizations
`hna`	for HNA in combination with stable/unstable additive spectral decomposition

3.2 Integration in Scilab

The Scilab interface has been realized in such a way to be mostly compatible with the MATLAB interface. In particular, the names of the mex-files and m-files are the same. The source code of the mex files can be used nearly as it is. The only significant difference is that we have chosen, for portability reasons, not to make use of the Fortran 90 "ALLOCATE" function. This choice implies that the needed variables are allocated into Scilab internal stack. To be more specific, here is the modified sequence in the source code for `sysred.f`:

```
C
C Allocate variable dimension local arrays.
C !Fortran 90/95
C       ALLOCATE ( A( LDA, MAX( 1, N ) ), B( LDB, MAX( 1, M ) ),
C      $           C( LDC, MAX( 1, N ) ), D( LDD, MAX( 1, M ) ),
C      $           DWORK( LDWORK ), HSV( MAX( 1, N ) ),
C      $           IWORK( MAX( 1, 2*N, M ) ) )
C
C Copy inputs from MATLAB workspace to locally allocated arrays
C !Fortran 77
C
      if( .not.createvar( rhs+1, 'd', LDA, MAX(1,N), ptrA ) ) return
      if( .not.createvar( rhs+2, 'd', LDB, MAX(1,M), ptrB ) ) return
      if( .not.createvar( rhs+3, 'd', LDC, MAX(1,N), ptrC ) ) return
      if( .not.createvar( rhs+4, 'd', LDD, MAX(1,M), ptrD ) ) return
      if( .not.createvar( rhs+5, 'd', 1, LDWORK, ptrDWORK ) ) return
      if( .not.createvar( rhs+6, 'd', 1, MAX(1,N), ptrHSV ) ) return
      if( .not.createvar( rhs+7, 'i', 1, MAX(1,2*N,M), ptrIWORK ) ) return
```

Then the variables are referred to by their position into the Scilab internal stack, e.g., `A` should be replaced by `stk(ptrA)` and the integer array `IWORK` by `stk(ptrIWORK)`.

These minor modifications allow the use of a standard Fortran 77 compiler such as `f2c` or `g77`. The MATLAB m-files also require some modification to follow the Scilab syntax. As an illustration, in Appendix E is the Scilab code `bta.sci` corresponding to the MATLAB m-file `bta.m` given in Appendix D. For compatibility purposes, the calling sequences are the same both in MATLAB and Scilab, and the help files are very similar. The only difference is that Scilab help files are formatted ASCII texts which can be automatically generated from the corresponding m-files.

4 Selection of benchmark problems, testing and performance comparisons

Extensive testing has been performed of the implemented software using several benchmark problems. In what follows we describe examples used to test the model reduction routines for stable and unstable systems.

4.1 PS: Power system model – continuous-time.

This is a continuous-time linearized state space model of a two-area interconnected power system [5]. The model has the form

$$\begin{aligned} \dot{x} &= Ax + B_1 u + B_2 w \\ y &= Cx \end{aligned}$$

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 F. 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 B&T, SPA and HNA methods produced reduced order models of order 5, PS_1, PS_2 and PS_3, respectively, which approximately preserve the dominant poles of the original system

Poles of PS	Poles of PS_1	Poles of PS_2	Poles of 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 B&T and SPA methods producing even non-minimum phase zeros:

Zeros of PS	Zeros of PS_1	Zeros of PS_2	Zeros of PS_3
∞	1.4438	1.4438	-9.2069
∞	∞	0	∞
	∞	∞	∞
	∞		

There is little difference in step responses for different input-output channels and also the Nyquist plots show good agreements. In Figures 1 and 2 the corresponding plots for the u_1–y_1 chanel are presented. Each of the computed 5-th order approximate models is suitable to perform controller synthesis.

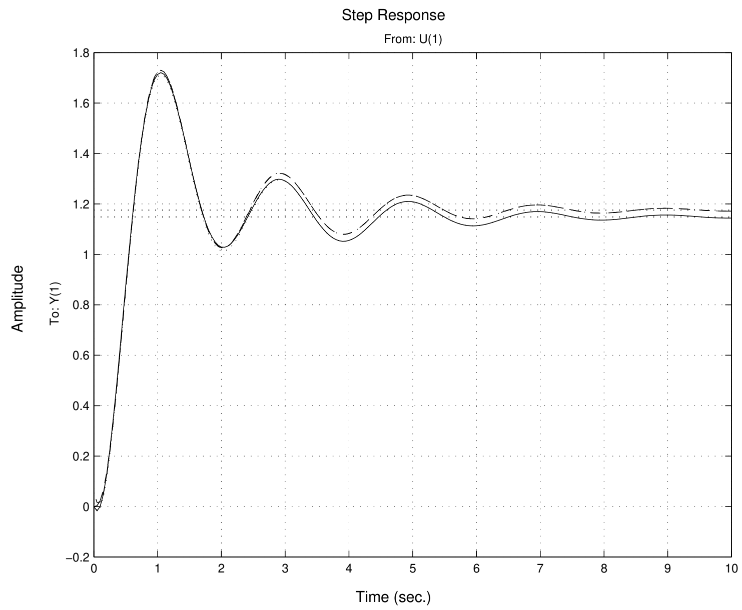

Figure 1: Comparison of step responses of element $g_{11}(s)$ of PS.

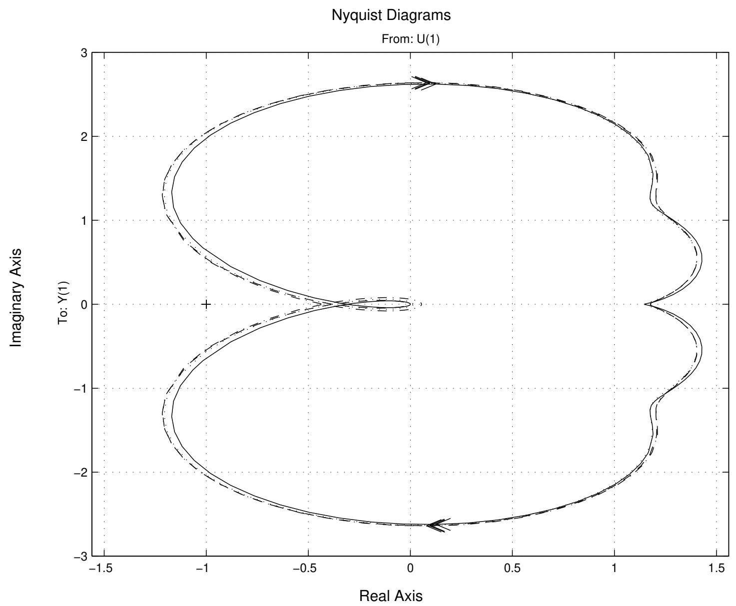

Figure 2: Comparison of frequency responses for element $g_{11}(s)$ of PS.

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 F. Three 5-th order approximations have been computed using the B&T, SPA and HNA methods. Figure 3 shows the good agreement both in time domain and frequency-domain of the original and reduced models.

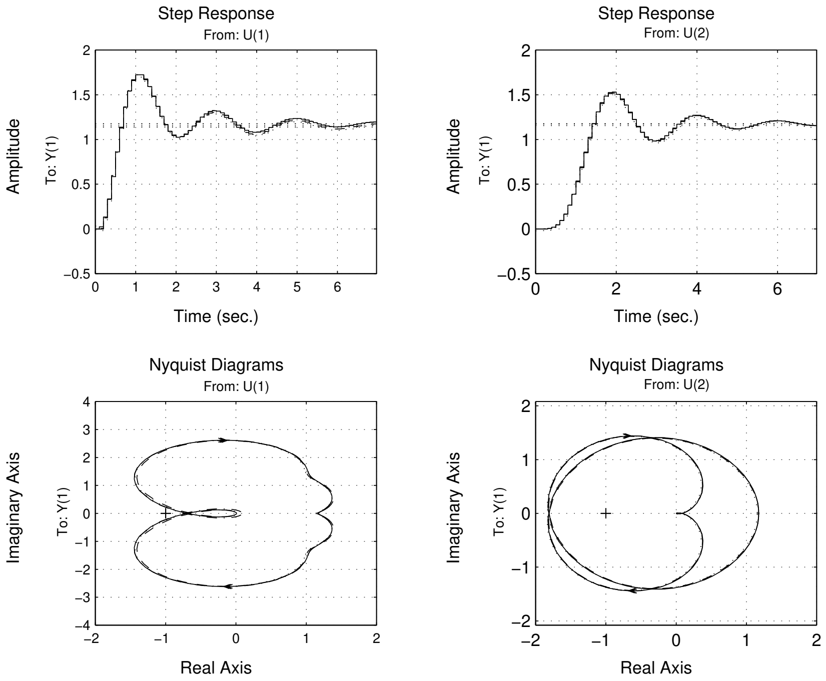

Figure 3: Comparison of time and frequency responses for elements $g_{11}(s)$ and $g_{12}(s)$ of PSD.

4.3 TGEN: Nuclear plant turbo-generator model.

This is a 10-th order linearized model of a 1072 MVA nuclear powered turbo-generator [8]. The system is stable and minimal phase. The Hankel-singular values of the system are

$$\{455.9850, 76.5248, 68.5691, 10.4294, 7.2374, 0.2704, 0.1115, 0.0022, 0.0019, 0.0014\},$$

thus a 5-th order approximation seems to be appropriate. Figure 4 compares the three different methods (B&T, SPA, HNA) on basis of element $g_{11}(s)$ of the transfer function matrix. It is easy to see the good low frequency approximation property of SPA method and the good high-frequency approximation property of B&T and HNA methods. Note that all three methods produce non-minimum phase approximations.

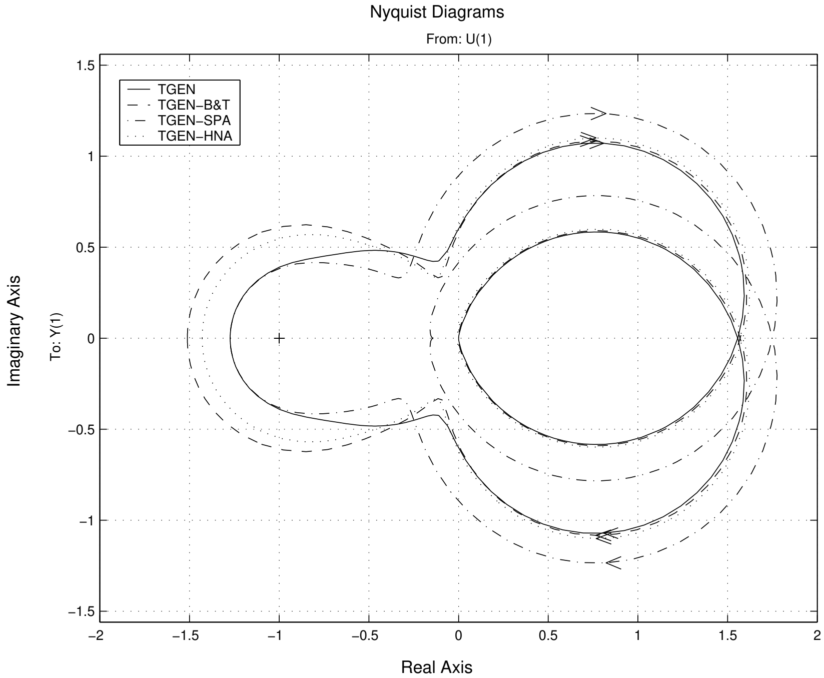

Figure 4: Comparison of frequency responses for element $g_{11}(s)$ of TGEN.

4.4 PSU: Unstable continuous-time model.

This model is only for numerical test and resulted by replacing in the PS model A by $A + \alpha I$, for different values of α leading to unstable systems. For $\alpha = 1$ we applied the B&T 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 5 shows good agreements of the Nyquist plots for the transfer function of 1–1 input/output chanel for all five methods. Similar results are obtained for the SPA and HNA methods used in combination with factorization/decomposition techniques.

4.5 ACT: 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 F. Due to its extremely poor scaling originated from the usage of International System (SI) units, it is expected that this model will rise numerical difficulties to many category of programs thus leading often to wrong numerical results.

The computed Hankel singular values by `sysred` are:

$$\left\{ \begin{array}{lllll} 1.38934e+007 & 6.10346e+006 & 1.04050e+006 & 1.03985e+006 & 8.26634e+005 \end{array} \right\}$$

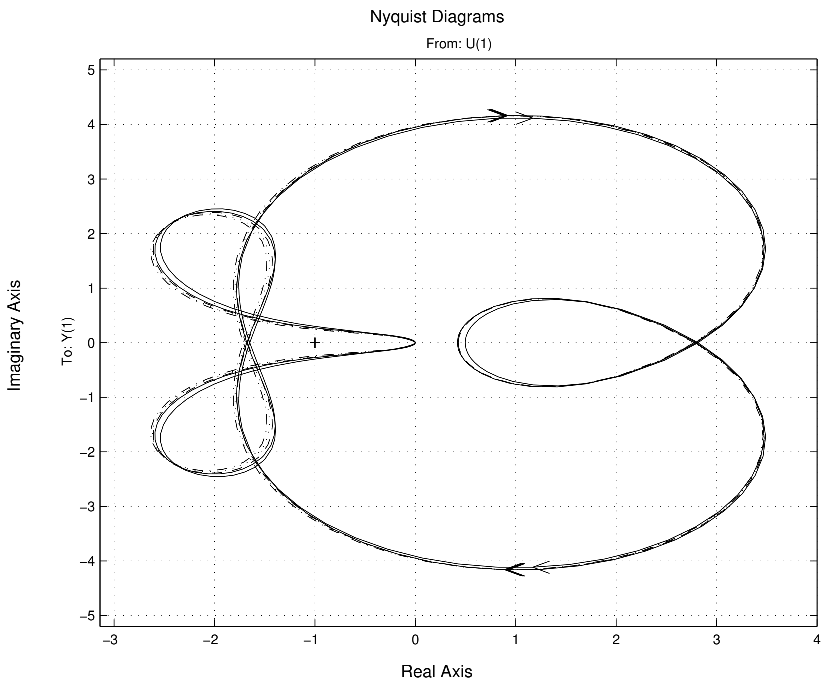

Figure 5: Comparison of frequency responses for element $g_{11}(s)$ of PSU.

Note that the Hankel singular values computed by the MATLAB function `balreal` without scaling are completely wrong

$$\left\{ \; 3.33570e+008 \quad 1.92108e+008 \quad 9.92157e+005 \quad 1.47773e+004 \quad 1.91871e+003 \; \right\}.$$

After a preliminary scaling with `ssbal`, the Hankel singular values computed by `balreal` agree to 15 decimal figures with those computed by `sysred`.

4.6 Randomly generated systems

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 B&T method the efficiency of the mex-function `sysred`, the m-functions `sqrmr` from HTOOLS and `balreal` from the Control Toolbox [11]. 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 2 have been obtained on a Pentium II 400 Hz Personal Computer running under Windows NT 4.0. Themex-function `sysred` has been produced using Digital Visual Fortran V 5.1.

This table illustrate not only the numerical robustness of structure exploiting numerical algorithms, but also the advantage in the speed of executions (up to one order of magnitude) allowing to solve relatively large order dense problems on a desktop PC.

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 2: Timing results for `sysred`, `sqrmr` and `balreal`.

4.7 Further test problems.

Recently the model reduction *mex*-function `sysred` has been employed as basic computational tool for exact and approximate order reduction of linear parametric uncertain systems described by *linear fractional transformations* (LFTs). An LFT model arises for instance 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 typically the resulting orders of ad-hoc built LFTs, even for relatively simple parametric uncertain models, are high, order reduction (exact or approximate) is an important aspect of LFT-modelling. Sequential 1-D minimal realization techniques can be successfully employed to achieve substantial order reduction. Since most of LFT descriptions are basically similar 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 rational parametric linear state space of a generic aircraft model. The order of initial LFT models were up to 300 and reduction with `sysred` led to low order exact and approximate LFT models. The high accuracy of approximations was assessed by using Monte-Carlo analysis. Note that the features required for this order reduction (discrete-time, non-minimal, unstable) are not available in the commercial CACSD software (see Appendix G).

5 Testing on industrial benchmark problems and comparisons with currently used software

5.1 Industrial benchmark problems

Several industrial problems have been used to test the SLICOT software via the *mex*-function `sysred`.

5.1.1 ATTAS: Linearized aircraft model

This model describes the linearized rigid body dynamics of the DLR Advanced Technology Testing Aircraft System (ATTAS) during the landing approach. The nonlinear model of ATTAS used for linearization has been obtained using the object oriented modelling tool Dymola [4]. Besides flight dynamics, this model includes actuators and sensors dynamics, as well as engine dynamics. Several low pass filters to eliminate structure induced 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 obtained by using model reduction techniques. A 15-th order approximation has been computed using model reduction followed by minimal realization which fits almost exactly the original 51 order model both in time as well as in frequency domain. Figure 6 shows a very good agreement obtained between the frequency responses of the original and reduced model for element g_{22} of the corresponding transfer function matrix.

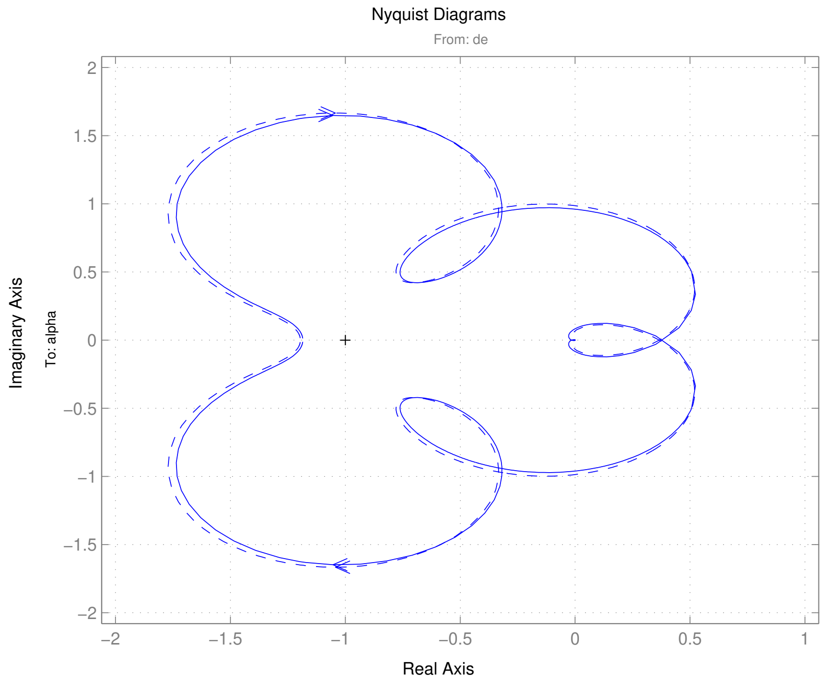

Figure 6: Comparison of frequency responses for element $g_{22}(s)$ of ATTAS.

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.

5.1.2 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. Only to test our software, three 10-th order models have been determined using the B&T, SPA and HNA methods. Figure 7 compares the performance of different computed approximations on basis of Bode plots. All methods approximate satisfactorily the central peek at 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 B&T methods appears to provide a good 10-th order approximation.

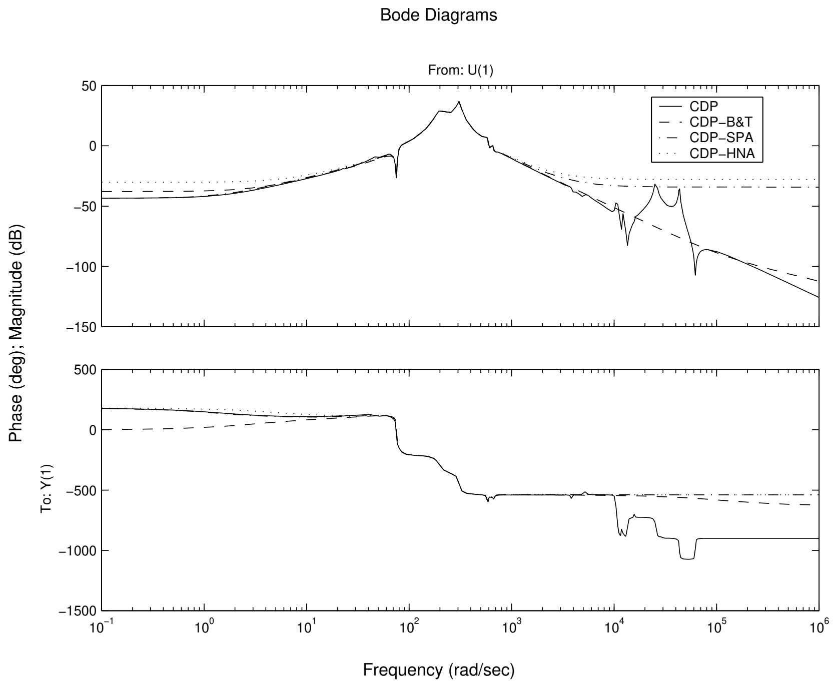

Figure 7: Comparison of frequency responses for CDP.

5.1.3 GAS: Gasifier model

A detailed nonlinear gasifier model has been developed by GEC ALSTHOM, in October 1997 as a benchmark problem for simulation and robust control. The model includes all significant effects; e.g., 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 fuel gas quality. Linearized models at 0%, 50% and 100% load are available to support a multi-model based robust controller design. Some analysis results on the 100% load models are discussed in [14]. Numerical difficulties with respect to using MATLAB model reduction tools, but also of the symbolic manipulation tools in Mathematica, have been reported. The apparent cause of difficulties is the poor scaling of the model. This can be seen by comparing the step response for element $g_{11}(s)$ for the original and scaled system at 0% load in Figure 8.

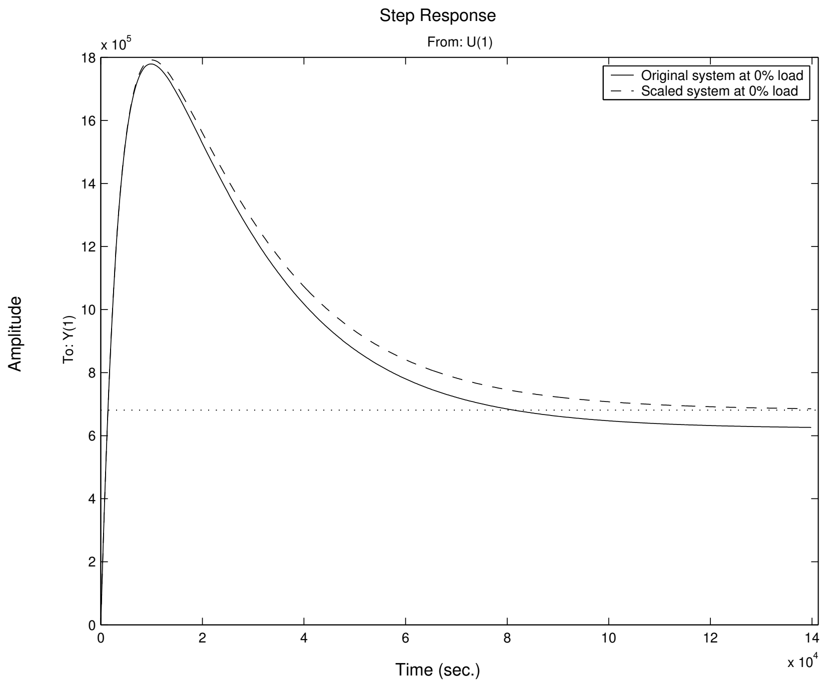

Figure 8: Comparison of step responses for original and scaled GAS.

The GAS model has order 25 and is non-minimal. The norm of state matrices for the three models ranges between is about $7.64 \cdot 10^8 - -1.03 \cdot 10^9$, but after scaling with the SLICOT routine TB01ID, all norms can be reduced below 100. However, the preliminary is not obligatory for using `sysred`, being an implicit feature for this *mex*-function. Still, for simulations we used the scaled models to avoid numerical difficulties with MATLAB plotting functions and to make comparison more reliable. The three models are non-minimal. For example, the last 10 Hankel singular values of the 100% load model are

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

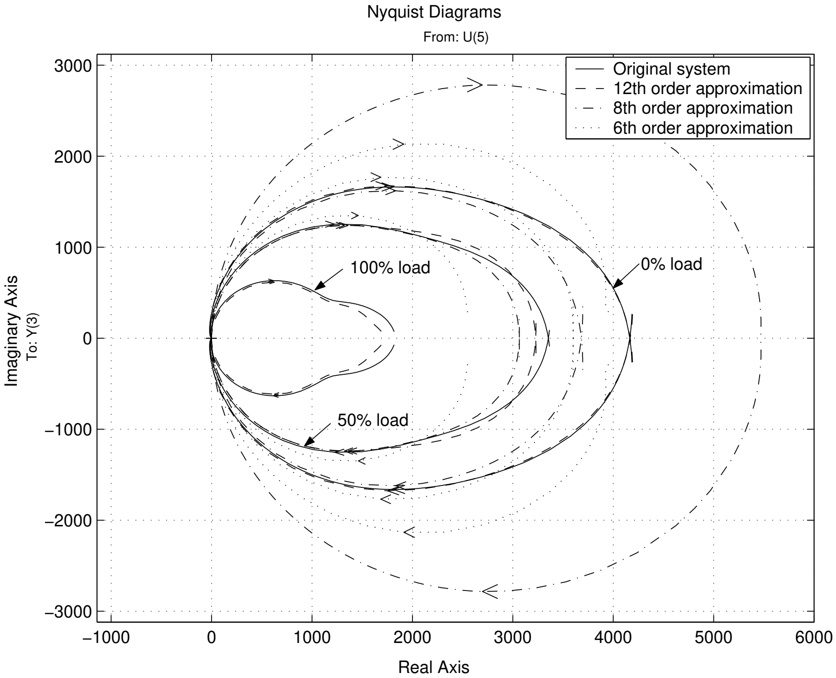

Figure 9: Comparison of frequency responses for $g_{35}(s)$ elements of GAS models.

and the Hankel-norm (the largest singular value) 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 9.

5.2 Comparison with other packages

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

5.2.1 Model reduction tools in ANDECS.

The facilities provided by the model reduction tools of the control system design environment of DLR ANDECS are based on the Fortran routines available in the RASP-MODRED package [22]. Since the SLICOT model reduction relies basically on RASP-MODRED routines, the only difference between the SLICOT and RASP-MODRED model reduction tools is that the SLICOT routines are implemented according to the SLICOT standard. The SLICOT implementations are slightly more efficient and possibly numerically more robust than the original codes from RASP-MODRED, because of better implementation of several supporting routines, as those to solve non-negative Lyapunov equations directly for the Cholesky factors.

5.2.2 Model reduction tools in MATLAB Control Toolbox V4.2.

The available function `balreal` performs the balancing by computing first the Gramians by solving Lyapunov equations and then computing the Cholesky factors of the Gramians. Although less accurate than the square-root method, the current version of `balreal` behaves numerically better than that one from the previous release of the Control Toolbox. Still there are several problems with this function. First, the original system must be minimal. This is a serious limitation since, many systems are almost or numerically non-minimal, which leads to the failure of this function. This has as consequence, that random stable system up to order 25-30 can not be reduced, although they are certainly minimal. Second, by performing always balancing, additional accuracy loss can occur in 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 ??). Finally, practically there is no support for reduction of unstable systems, which once again restricts the applicability of this function to stable models.

5.2.3 Model reduction tools in MATLAB Robust Control Toolbox V2.0

There are several model reduction tools in the Robust Control Toolbox [3] which cover similar model reduction problems as `sysred`. The B&T 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 order systems as well as to unstable system. 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 additive error. In particular, phase information is better approximated, thus approximations of minimum-phase models lead often to minimum-phase reduced models. This functionality is not covered by SLICOT and is intended to be added in the second part of the project.

5.2.4 Model reduction tools in MATLAB μ-Analysis and Synthesis V3.0

There are several model reduction tools in the μ-Analysis and Synthesis Toolbox [2] which cover similar model reduction problems as `sysred`. The square-root B&T 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 reduction of discrete-time systems bilinear transformation techniques are recommended to be used. `sysbal` and `hankmr` can be applied also to non-minimal order systems. Even without a preliminary scaling, `sysbal` was able to compute up to 5 digits accurate 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 two functions for relative error model reduction via balanced stochastic truncation, frequency weighted model reduction and normalized coprime factorization based model reduction.

5.2.5 Model reduction tools in MATRIX$_X$ Model Reduction Module

The model reduction functions in the MATRIX$_X$ Model Reduction Module [12] 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 hints to overcome some of them.

5.2.6 Other available model reduction tools

The model reduction tools in the HTOOLS (H_∞ Tools) Toolbox for MATLAB [25], has been implemented having as basis the numerical algorithms with enhanced accuracy described in [22]. 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 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 time as equivalent Fortran implementations. This aspect is however is general for all model reduction tools implemented exclusively in Matlab.

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 several 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 open-loop and closed-loop, weighted and unweighted configurations. The WOR Toolbox calls several low level functions of the μ-Toolbox.

In the previous version of Scilab no complete 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 last version of Scilab integrates the same functionalies as those available for MATLAB (see section 3).

6 Summary of achieved 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-/discrtete-time, stable/unstable linear time-invariant systems. The final model reduction package consists of 12 user-callable routines. All this routines are thoroughly documented. The documentation is automatically generated from the preamble of each routine (see Appendix H 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 a test program example, test data and the corresponding test results.

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 with Scilab. Additional 9 *m*-functions (see Section 3) have been implemented which exploit the advanced object oriented facilities available both in MATLAB Control Toolbox as well as in Scilab to manipulate control objects. The *mex*-function and *m*-functions are completely documented according to MATLAB/Scilab standards.

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. This software will complement the H_2/H_∞ software currently developed for SLICOT.

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, Second Edition.* SIAM, Philadelphia, 1995.

[2] G. Balas, J. Doyle, K. Glover, A. Packard, and R. Smith. *μ-Analysis and Synthesis Toolbox 3.0.4.* The MathWorks Inc., Natick, MA, 1998.

[3] R. Y. Chiang and M. G. Safonov. *Robust Control Toolbox 2.0.6.* The MathWorks Inc., Natick, MA, 1997.

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

[5] 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.

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

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

[8] 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.

[9] Y. Liu and B. D. O. Anderson. Controller reduction via stable factorization and balancing. *Int. J. Control*, 44:507–531, 1986.

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

[11] MATLAB. *Control System Toolbox 4.2.* The MathWorks Inc., Natick, MA, 1998.

[12] $\mathrm{MATRIX_X}$. *Xmath Model Reduction Module.* ISI, Santa Clara, CA, January 1998.

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

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

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

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

[17] A. Varga. Balancing-free square-root algorithm for computing singular perturbation approximations. In *Proc. of 30th IEEE CDC, Brighton, UK*, pages 1062–1065, 1991.

[18] A. Varga. Efficient minimal realization procedure based on balancing. In A. El Moudni, P. Borne, and S. G. Tzafestas, editors, *Prepr. of IMACS Symp. on Modelling and Control of Technological Systems*, volume 2, pages 42–47, 1991.

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

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

[21] A. Varga. A Schur method for computing coprime factorizations with inner denominators and applications in model reduction. In *Proc. 1993 American Control Conference, San Francisco, CA*, pages 2130–2131, 1993.

[22] A. Varga. Numerical methods and software tools for model reduction. In I. Troch and F. Breitenecker, editors, *Proc. of 1st MATHMOD Conf., Viena*, volume 2, pages 226–230, 1994.

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

[24] A. Varga and K. H. Fasol. A new square-root balancing-free stochastic truncation model reduction algorithm. In *Prepr. of 12th IFAC World Congress, Sydney, Australia*, volume 7, pages 153–156, 1993.

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

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

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

A List of standardized model reduction routines

Routines for reduction of stable systems

Name	Function
AB09AD	computes reduced (or minimal) order balanced models using either the **SR** or the **BFSR** B&T method
AB09AX	computes reduced (or minimal) order balanced models using either the **SR** or the **BFSR** B&T method (scaled system with state matrix in real Schur form)
AB09BD	computes reduced order models using the **BFSR** SPA method
AB09BX	computes reduced order models using the **BFSR** SPA method (scaled system with state matrix in real Schur form)
AB09CD	computes reduced order models using the optimal HNA method based on **SR** balancing
AB09CX	computes reduced order models using the optimal HNA method based on **SR** balancing (scaled system with state matrix in real Schur form)
AB09DD	computes a reduced order model by using the singular perturbation formulas

Routines for reduction of unstable systems

Name	Function
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 **BFSR** B&T method in conjunction with left/right coprime factorization methods
AB09GD	computes reduced order models for unstable systems using the **BFSR** SPA method in conjunction with left/right coprime factorization methods
AB09MD	computes reduced order models for unstable systems using the B & T 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

B List of standardized routines related to model reduction

Transformation routines

Name	Function
TB01KD	computes the terms G_1 and G_2 of an additive spectral decomposition of a transfer-function matrix G with respect to a specified region of the complex plane
TB01LD	performs an orthogonal similarity trasformation to reduce the system state matrix to an ordered real Schur form
TB01WD	performs an orthogonal similarity trasformation to reduce the system state matrix to a real Schur form

Mathematical routines

Name	Function
MB03QD	reorders the eigenvalues of a real Schur matrix according to several reordering criteria
MB03UD	computes the singular value decomposition of a square upper-triangular matrix

Analysis routines

Name	Function
AB13AD	computes the Hankel norm and the Hankel singular values of the stable projection of a transfer-function matrix
AB13AX	computes the Hankel norm of a stable system with the state matrix in real Schur form
AB13BD	computes the H_2- or L_2-norm of a transfer-function matrix

Factorization routines

Name	Function
SB08CD	computes the state-space representations of the factors of a LCFID of a TFM
SB08DD	computes the state-space representations of the factors of a RCFID of a TFM
SB08ED	computes the state-space representations of the factors of a LCF with prescribed stability degree
SB08FD	computes the state-space representations of the factors of a RCF with prescribed stability degree
SB08GD	computes the state-space representation of the TFM corresponding to a LCF
SB08HD	computes the state-space representation of the TFM corresponding to a RCF

Lower level/auxiliary routines

Name	Function
MA02AD	transposes all or a part of a matrix
MA02BD	reverses the order of rows and/or columns of a matrix
MA02CD	pertransposes a diagonal band of matrix
MB01SD	scales a matrix by rows or columns
MB03QX	computes the eigenvalues of a matrix in real Schur form
MB03QY	computes the eigenvalues of a 2 by 2 block of matrix in real Schur form and reduces it to the standard LAPACK form
SB01BY	solves an N by N pole placement problem for N = 1 or N = 2
SB01FY	computes the inner denominator of a right-coprime factorization of a system of order N, where N is either 1 or 2
MB04OX	performs a rank-one update of a Cholesky factorization

C MATLAB interface to model reduction routines

```
%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 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 case c = 1 or 3
%                  default: epsilon_machine*max(norm(A),norm(B))
%                  observability tolerance in 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
%                      1.0-sqrt(epsilon_machine)  for discrete-time
```

D Sample MATLAB *m*-function for B&T model reduction

```
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 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
%           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
```

E Sample Scilab *sci*-function for B&T model reduction

```
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
```

F Benchmark problems for model reduction

F.1 State space matrices of the PS model

Name	PS
Description	Power system model
Reference	[5]
Type	State space model, continuous-time
# states	7
# control inputs	2
# disturbance inputs	2
# outputs	3

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

$$B_1 = \begin{bmatrix} 0 & 0 \\ 12.5 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 12.5 \\ 0 & 0 \end{bmatrix}, \quad B_2 = \begin{bmatrix} -4.92 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -4.92 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad D = 0\,.$$

F.2 State space matrices of the PSD model

Name	PSD
Description	Sampled-data power system model ($T = 0.1$ sec)
Reference	[5]
Type	State space model, discrete-time
# states	7
# control inputs	2
# disturbance inputs	2
# outputs	3

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

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

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

F.3 State space matrices of the TGEN model

Name	TGEN
Description	Nuclear-powered turbo-generator
Reference	[8]
Type	State space model, continuous-time
# states	10
# control inputs	2
# disturbance inputs	0
# outputs	2

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.11323 & -0.98109 & -11.847 & -11.847 & -63.08 & -34.339 & -34.339 & -27.645 & 0 \\ 324.121 & -1.1755 & -29.101 & 0.12722 & 2.83448 & -967.73 & -678.14 & -678.14 & 0 & -129.29 \\ -127.3 & 0.46167 & 11.4294 & -1.0379 & 13.1237 & 380.079 & 266.341 & 266.341 & 0 & 1054.85 \\ -186.05 & 0.67475 & 16.7045 & 0.86092 & -17.068 & 555.502 & 389.268 & 389.268 & 0 & -874.92 \\ 341.917 & 1.09173 & 1052.75 & 756.465 & 756.465 & -29.774 & 0.16507 & 3.27626 & 0 & 0 \\ -30.748 & -0.09817 & -94.674 & -68.029 & -68.029 & 2.67753 & -2.6558 & 4.88497 & 0 & 0 \\ -302.36 & -0.96543 & -930.96 & -668.95 & -668.95 & 26.3292 & 2.42028 & -9.5603 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.66667 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1.66667 & 0 \\ 0 & 10 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.49134 & 0 & -0.63203 & 0 & 0 & -0.20743 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad D = 0\,.$$

F.4 State space matrices of the ACT model

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

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -1580000 & -1257 & 0 & 0 & 0 \\ 3.541e{+}14 & 0 & -1434 & 0 & -5.33e{+}11 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -18630 & -1.482 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 110.3 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0.008333 \end{bmatrix}$$

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

G Summary of comparisons of available model reduction tools

Software	RASP/ANDECS	SLICOT/Scilab	Control Toolbox	Robust Toolbox	μ-Toolbox	HTOOLS	MatrixX	WOR-Toolbox
Provided features								
continuous-time	+	+	+	+	+	+	+	+
discrete-time	+	+	+	−	−	+	−	−
unstable	+	+	−	+	−	−	−	+
non-minimal	+	+	−	+	+	+	+	+
Method								
balancing	+	+	+	+	+	+	+	+
balancing-free (BF)	+	+	−	+	−	+	+	−
square-root (SR)	+	+	−	−	+	+	−	+
BF-SR	+	+	−	−	−	+	−	−
Problem classes								
additive error	+	+	+	+	+	+	+	+
relative error	+	−	−	+	+	+	+	−
frequency weighted	+	−	−	−	+	−	+	+

H Preamble of AB09MD.F used for automatic documentation

```
      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 3.0, 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 square-root
C     or the balancing-free square-root Balance & Truncate (B & T)
C     model reduction method for 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 determined
C                     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, i.e.
C             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; NMIN is
C             determined as the number of Hankel singular values greater
C             than NS*EPS*HNORM(As,Bs,Cs), where EPS is the machine
C             precision (see LAPACK Library Routine DLAMCH) and
C             HNORM(As,Bs,Cs) is the Hankel norm of the ALPHA-stable
C             part of the given system (computed in HSV(1));
C             if ORDSEL = 'A', NR is the sum of NU and the number of Hankel
C             singular values greater than MAX(TOL,NS*EPS*HNORM(As,Bs,Cs)).
C
C     ALPHA   (input) DOUBLE PRECISION.
C             Specifies the ALPHA-stability boundary for the eigenvalues of
C             the state dynamics matrix A. For a continuous-time system
C             (DICO = 'C'), ALPHA =< 0 is the boundary value for the real
C             parts of eigenvalues, while for a discrete-time system
C             (DICO = 'D'), 1 >= ALPHA >= 0 represents the boundary value
C             for the 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.
C             The resulting A has a block 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 the
C             Hankel singular values of the ALPHA-stable part of the
C             original system ordered decreasingly.
C             HSV(1) is the Hankel norm of the ALPHA-stable 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 realization
C             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 case,
C                   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 routine
C      can be used to compute balanced minimal state-space realizations.
C      If JOB = 'N' the square-root balancing-free version of the
C      Balance & Truncate  method is used [2] to reduce the ALPHA-stable
C      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 or
C      balancing-free square-root techniques.
C                                            3
C      The algorithms require less than 30N  floating point 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
```