8

ISSN 1373-6787

February 2002

NICONET Newsletter

Distributed by: **Working Group on Software WGS**

Contents: **Page**

Contact address of the WGS: Mrs. Ida Tassens, Secretary of WGS
Katholieke Universiteit Leuven
Dept. of Electrical Engineering (ESAT-SISTA/COSIC)
Kasteelpark Arenberg 10
3001 Leuven-Heverlee, Belgium
email: ida.tassens@esat.kuleuven.ac.be
phone: + 32 16 32 17 09 and fax: + 32 16 32 19 70

1 Editorial

Welcome to the eighth issue of the NICONET newsletter which informs you about the evolution of the SLICOT library and its integration in user-friendly environments such as `Scilab` and MATLAB, as well as about other NICONET activities related to CACSD software developments.
In the last 6 months several important events happened which are worthwhile to be mentioned. First of all, as mentioned in our previous newsletter, our new EC proposal we submitted in the Growth programme (accompanying measures) in March 2001 hasn't been approved. We have to wait now for new calls in the 6th framework before we can resubmit our proposal. In the mean time we asked the EC to extend our present thematic network project NICONET with 6 months until July 1, 2002, and this extension has been approved. Since September 2001 our international society, also called NICONET, is operational. Any funding received through this society is used for the further development of the SLICOT library and will be needed in periods where no EC funding is available. Simultaneously, this society promotes and supervises the dissemination of the SLICOT software.
Secondly, we want to mention that we had a very successful training course in Bremen at the end of September. A detailed report describing the highlights of this workshop is included in Section 9. Sections 2 to 6 present as usual the new updates of the SLICOT library in subfields of systems and control. Section 7 discusses how SLICOT subspace identification software improves the performance of loadspeakers. In Section 8, the use of SLICOT in Controllabs Products, a small spinoff company in the Netherlands, is discussed. Finally, Section 10 gives more details about the newest additions to the SLICOT library, new reports and forthcoming events.

I hope you enjoy reading this newsletter.

Sabine Van Huffel
NICONET coordinator

2 Basic numerical SLICOT tools for control

2.1 Standard and generalized state space systems and transfer matrix factorizations

This task continued in order to provide basic routines required by other tasks. Therefore, the corresponding routines are mentioned in other sections of this Newsletter issue.

Vasile Sima, Andras Varga and Paul Van Dooren

2.2 Structured matrix computations

The following routines were recently implemented as basic routines for structured matrix computations :

Name	Function
DE01PD	Convolution or deconvolution of two real signals using Hartley transform.
DG01OD	Scrambled discrete Hartley transform of a real signal.
MB02FD	Incomplete Cholesky factor of a positive definite block Toeplitz matrix.
MB02GD	Cholesky factorization of a band symmetric positive definite block Toeplitz matrix.
MB02HD	Cholesky factorization of the matrix T^TT, with T a band block Toeplitz matrix of full rank.
MB02ID	Solution of over- or underdetermined linear systems with a full rank block Toeplitz matrix.
MB02JD	Full QR factorization of a block Toeplitz matrix of full rank.
MB02JX	Low rank QR factorization with column pivoting of a block Toeplitz matrix.
MB02KD	Computation of the product $C = \alpha\,\mathrm{op}(T)B + \beta C$, with T a block Toeplitz matrix.

These Fortran programs will be accessed in the interactive environment MATLAB or `scilab` via the same mex and m.files as for the earlier Fortran codes. The associated test functions and a MATLAB 5.3 demonstration package are available via the NICONET homepage. The revised report SLWN2000-2 describes the functionality of these routines.

[1] D. Kressner and P. Van Dooren, Factorizations and linear system solvers for matrices with Toeplitz structure, SLICOT Working Note 2000-2 (revised), June 2001.

Daniel Kressner, Vasile Sima and Paul Van Dooren

3 SLICOT tools for model reduction

3.1 SLICOT tools for controller reduction

A new user callable routine **AB09JD** for the frequency-weighted Hankel-norm approximation method with invertible proper weights has been implemented, together with three lower level routines: **AB09JV** and **AB09JW** to compute stable projections for left and right frequency weights using a descriptor system formulation, and **AB09JX** for testing eigenvalues or generalized eigenvalues for stability/antistability. Further, 4 new user callable basic routines have been implemented for the special needs of model/controller reduction software: **AG07BD** to perform descriptor system inversion, **AB13DD** to compute the L_∞-norm of a system, **AB08MD** to determine the normal rank of the transfer-function matrix of a state-space system, and **TG01BD** to reduce a descriptor system to orthogonal generalized Hessenberg form. These new basic routines enter in SLICOT as contributions to Basic numerical SLICOT tools for control.

The newly developed model and controller reduction routines are called by user-friendly interfaces defined in 6 mex-files and 8 m-files for MATLAB and `scilab`. The implemented mex-functions complement the previously available mex-function `sysred`. All functions can reduce both stable and unstable as well as continuous- and discrete-time systems or controllers. The frequency-weighted model reduction functions can also be used for unweighted reduction. Besides the model/controller reduction mex-functions, a new, completely general mex-function has been implemented to compute the L_∞-norm of a standard or descriptor system. The implemented new mex-functions for model/controller reduction are:

Name	Function
`bstred`	balanced stochastic truncation based model reduction (based on AB09HD)
`fwered`	frequency-weighted balancing related model reduction (based on AB09ID)
`fwehna`	frequency-weighted Hankel-norm approximation (based on AB09JD)
`conred`	frequency-weighted balancing related controller reduction (based on SB16AD)
`sfored`	coprime factorization based reduction of state feedback controllers (based on SB16BD and SB16CD)
`linorm`	L_∞-norm of a linear time-invariant system (based on AB13DD)

Easy-to-use m-functions have been implemented for MATLAB and `scilab` to provide a convenient interface to the implemented mex-functions. These tools allow to work directly with control objects as defined in the Control Toolbox of MATLAB or in `scilab`. The implemented m-functions for model/controller reduction are:

`bst`	balanced stochastic truncation based model reduction
`fwbred`	frequency-weighted balancing related model reduction
`fwhna`	frequency-weighted Hankel-norm approximation
`fwbconred`	frequency-weighted balancing related controller reduction
`sfconred`	coprime factorization based state feedback controller reduction
`sysredset`	creation of a SYSRED options structure
`sysredget`	extracts the value of a named parameter from a SYSRED options structure
`slinorm`	L-infinity norm of a state-space system

To manage the combinatorial complexity of the multitude of possible user options, a special SYSRED structure has been defined. Two special functions `sysredset` and `sysredget` have been implemented to create/setup this structure and to extract values of option parameters. The m-function `slinorm` is an interface to the `linorm` mex-function.

An intensive testing of all implemented routines has been performed via the newly developed m- and mex-files. For this purpose, both special benchmark problems for controller reduction as well as industrial examples have been used to investigate the capabilities of different approaches to perform controller reduction which preserves closed-loop stability and performance. Four test suites for testing the implemented m- and mex-functions have also been implemented. A controller reduction toolbox has been prepared and made available via the SLICOT library ftp-site.

Andras Varga

3.2 SLICOT tools for model reduction of high order systems

3.2.1 Direct methods for model reduction

During the last 6 months, the work has focused on the adaptation of the newly developed parallel Lyapunov solver (see previous newsletter) to make it consistent with the sequential SLICOT routine performing the same task (`SB03OU`). In particular, the SLICOT routines in charge of the model reduction problem expect to pass the matrices of the Lyapunov equation and receive the solution of it in a different way of that used by the initial parallel solver (which was developed in a more general form). This led to the release of the new parallel `PSB03OU` routine.

The work is now being oriented to the parallelisation of the main routines for reduction of stable systems (`AB09AD, AB09BD, AB09DD`). This requires first the parallelisation of several auxiliary routines used by them: `MB03UD` (routine in charge of computing the Singular Value Decomposition), `TB01ID, TB01WD`, ... The parallel versions of most of these auxiliary routines have already been implemented.

Vicente Hernandez, David Guerrero

3.2.2 Model reduction based on iterative solvers for computing the system Gramians

The integration on parallel computers of the model reduction subroutines based on iterative solvers was finalized. Table 1 lists the user-callable subroutines that have been integrated into PSLICOT.

Besides, 12 lower-level subroutines were also tested and integrated into PSLICOT. All routines have been successfully installed on a cluster of Linux-PCs with 32 nodes.

The benchmark testing was finalized in June 2001. The following benchmark problems have been used:

- For model reduction performance and accuracy tests, examples with prescribed order of the minimal realization and the rank of the Gramians of the systems have been generated.
- A finite-element (FEM) model for a control problem for 1-dimensional heat flow has been used for testing scalability with growing order n of the system.

Routine	Purpose
PAB09AX	computes reduced (or minimal) order balanced models using either the **SR** or the **BFSR** B&T method.
PAB09BX	computes reduced order models. using the **BFSR** or **SR** SPA method.
PAB09CX	computes reduced order models using the optimal HNA method based on **SR** balancing.
PAB09DD	applies the singular perturbation approximation formulae to a general system.
PSB03ODC	solution of coupled stable Lyapunov equations for full-rank factor using sign function method.
PSB03ODD	solution of coupled stable Stein equations for full-rank factor using squared Smith iteration or sign function method.
PSB04MD	solves a stable Sylvester equation using the sign function method.
PMB05RD	computes the sign function of a matrix using the Newton iteration.
PMB03TD	computes the singular value decomposition of a matrix product.
PMB03OX	estimates the rank of a triangular matrix using an incremental condition estimator.

Table 1: PSLICOT subroutines for absolute error model reduction based on iterative solvers

- For parallel performance, random examples with $n \leq 4000$ were generated.

For industrial case studies, the following problems were selected:

- A control problem for a catalytic tubular reactor model from ABB Corporate Research with state-space dimension $n = 1171$. The numbers of inputs and outputs are $m = 6$ and $p = 4$.

- An FEM model for optimal cooling of steel profiles with state-space dimension $n = 3113$ and 6 inputs and outputs.

Figure 1 shows the performance of some of the implemented routines for some of the benchmark problems and industrial case studies. The left plot shows the Bode diagram comparing the frequency responses (u_1 to y_1) of the catalytic tubular reactor model and the SPA reduced order model computed by PAB09BX. A good match is evident from the plot. The right plot displays execution times for computing reduced-order models using BT for randomly generated data. The result on 1 processor is obtained using the SLICOT routine AB09AD whereas the results on more than 1 processor show the performance of PAB09AX. The efficiency of the parallel method is clearly indicated by the (almost) optimal speed-up that can be concluded from the decreasing execution times.

Rather than providing a MATLAB toolbox, the deliverable here was chosen to be an environment in which users can easily use the parallel algorithms without having to install the routines themselves. We chose to make this possible by providing a mail and a web service for remote model reduction. Here, a user can submit the data related to his model

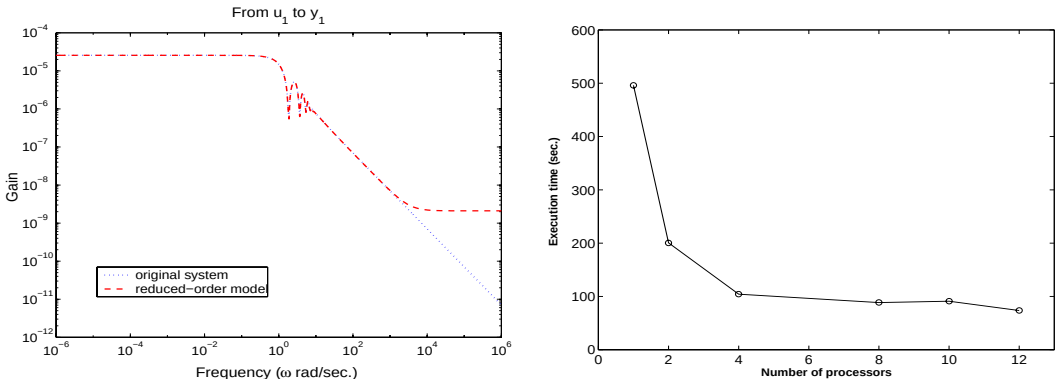

Figure 1: Performance of PSLICOT model reduction routines.

via e-mail[1] or an interactive web form[2] to a remote parallel computer (a Linux cluster). The user only selects the method for model reduction. Then the corresponding PSLICOT routine is called and the reduced order model is computed on the remote system. The data of the reduced order model and some additional information are then sent back to the user via e-mail. The details of this service are described in the SLICOT Working Note 2002-1 where also more computational results involving the benchmark examples and industrial case studies are reported.

Peter Benner

[1] Available at `http://spine.act.uji.es/~plicmr/pmrRemote/pmrMail`
[2] Available at `http://spine.act.uji.es/~plicmr/pmrW3/pmrW3.html`

4 SLICOT tools for subspace identification

4.1 Standard software for nonlinear state space model identification

The effort in this task concentrated on the identification of multivariable nonlinear Wiener systems. Such systems are a concatenation of a linear dynamic block followed by a static nonlinearity. An inventory of existing algorithms focused on schemes that enable the linear part to be represented in state space form. This inventory lead to the selection of a combination of a variant of the subspace identification method and a single layer neural network to model the static nonlinearity.

The standardization of the associated, planned routines for identification of nonlinear, state space systems has been completed, but further polishing appeared as necessary for improving the reliability and speed. Many improvements have been already performed, but this activity will continue in the next months. In parallel, further work will be done for integration of the routines in MATLAB and `scilab`, selection of benchmark problems, and extension of the toolbox.

A list of the routines and a brief description of their functionality is given below. The list includes (part of) the related mathematical and transformation routines which have been developed for identifying Wiener systems.

Name	Function
IB03AD	to compute a set of parameters for approximating a Wiener system in a least-squares sense, using a neural network approach and a Levenberg-Marquardt algorithm.
MB02WD	to solve a system of linear equations $Ax = b$, with A symmetric, positive definite, or, in the implicit form, $f(A, x) = b$, where $y = f(A, x)$ is a symmetric positive definite linear mapping from x to y, using the conjugate gradient algorithm without preconditioning.
MB02XD	to solve a set of systems of linear equations, $A^TAX = B$, or, in the implicit form, $f(A)X = B$, with A^TA or $f(A)$ positive definite, using symmetric Gaussian elimination.
MB02YD	to solve a system of linear equations $Ax = b$, $Dx = 0$, in the least squares sense, with D a diagonal matrix, given a QR factorization with column pivoting of A.
MD03AD	to find the parameters θ for a function $F(x, \theta)$ that give the best approximation for $y = F(x, \theta)$ in a least-squares sense using a Levenberg-Marquardt algorithm based on conjugate gradients for solving linear systems.
MD03BD	to find the parameters θ for a function $F(x, \theta)$ that give the best approximation for $y = F(x, \theta)$ in a least-squares sense using a Levenberg-Marquardt algorithm based on QR factorization with column pivoting.
MD03BX	to compute the QR factorization with column pivoting of an $m \times n$ matrix J ($m \geq n$), that is, $JP = QR$, where Q is a matrix with orthogonal columns, P a permutation matrix, and R an upper trapezoidal matrix with diagonal elements of nonincreasing magnitude, and to apply the transformation Q^T on the error vector e. The 1-norm of the scaled gradient is also returned.

MD03BY	to find a value for the parameter λ such that if x solves the system $Ax = b$, $\lambda^{1/2}Dx = 0$, in the least squares sense, where A is an $m \times n$ matrix, D is an $n \times n$ nonsingular diagonal matrix, and b is an m-vector, and if δ is a positive number, then either $\lambda = 0$ and $(\|Dx\|_2 - \delta) \leq 0.1\delta$, or $\lambda > 0$ and $\mid \|Dx\|_2 - \delta \mid \leq 0.1\delta$. It is assumed that a QR factorization with column pivoting of A is available, that is, $AP = QR$, where P is a permutation matrix, Q has orthogonal columns, and R is an upper triangular matrix with diagonal elements of nonincreasing magnitude.
NF01AD	to compute the output of a Wiener system.
NF01AY	to compute the output of a set of neural networks.
NF01BD	to compute the Jacobian of a Wiener system.
NF01BP	to find a value for the parameter λ such that if x solves the system $Jx = b$, $\lambda^{1/2}Dx = 0$, in the least squares sense, where J is an $m \times n$ matrix, D is an $n \times n$ nonsingular diagonal matrix, and b is an m-vector, and if δ is a positive number, then either $\lambda = 0$ and $(\|Dx\|_2 - \delta) \leq 0.1\delta$, or $\lambda > 0$ and $\mid \|Dx\|_2 - \delta \mid \leq 0.1\delta$. It is assumed that a QR factorization with block column pivoting of J is available, that is, $JP = QR$, where P is a permutation matrix, Q has orthogonal columns, and R is an upper triangular matrix with diagonal elements of nonincreasing magnitude for each block.
NF01BQ	to solve a system of linear equations $Jx = b$, $Dx = 0$, in the least squares sense, with D a diagonal matrix, given a QR factorization with block column pivoting of J.
NF01BR	to solve one of the systems of linear equations $Rx = b$, or $R^Tx = b$, in the least squares sense, where R is an $n \times n$ block upper triangular matrix, with the structure $\left[\begin{array}{cccc\|c} R_1 & 0 & \cdots & 0 & L_1 \\ 0 & R_2 & \cdots & 0 & L_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & R_\ell & L_\ell \\ 0 & 0 & \cdots & 0 & R_{\ell+1} \end{array}\right]$, with the upper triangular submatrices R_k, $k = 1{:}\ \ell + 1$, square, and the first ℓ of the same order. The diagonal elements of each block R_k have nonincreasing magnitude. The matrix R is stored in a compressed form.
NF01BS	to compute the QR factorization with block column pivoting of an $m \times n$ matrix J ($m \geq n$), that is, $JP = QR$, where Q is a matrix with orthogonal columns, P a permutation matrix, and R an upper trapezoidal matrix with diagonal elements of nonincreasing magnitude for each block, and to apply the transformation Q^T on the error vector e. The 1-norm of the scaled gradient is also returned.
NF01BU	to compute the matrix J^TJ+cI, for the Jacobian J given in a compressed form.

NF01BV	to compute the matrix J^TJ+cI, for the Jacobian J fully given, for one output variable.
NF01BW	to compute the matrix-vector product $x \leftarrow (J^TJ+cI)x$, where J is given in a compressed form.
NF01BX	to compute $x \leftarrow (A^TA+cI)x$, where A is an $m \times n$ real matrix, and c is a scalar.
NF01BY	to compute the Jacobian of the error function for a neural network (for one output variable).
TB01VD	to convert the linear discrete-time system given as (A, B, C, D), with initial state x_0, into the output normal form, with parameter vector θ. The matrix A is assumed to be stable. The matrices A, B, C, D and the vector x_0 are transformed, so that on exit they correspond to the system defined by θ.
TB01VY	to convert the linear discrete-time system given as its output normal form, with parameter vector θ, into the state-space representation (A, B, C, D), with the initial state x_0.
TF01MY	to compute the output sequence of a linear time-invariant open-loop system given by its discrete-time state-space model (A, B, C, D), where A is an $n \times n$ general matrix (the input and output trajectories are stored differently from SLICOT Library routine TF01MD).

The Levenberg-Marquardt algorithm using block QR factorization with column pivoting is a specialized, structure-exploiting LAPACK-based implementation of the approach used in the MINPACK package, developed at the Argonne National Laboratory, U.S.A. By a suitable reordering of the parameters describing the Wiener system, the Jacobian matrices (in the multi-output case) could be put in a block diagonal form, with an additional block column at the right. This structure is preserved in a QR factorization with column pivoting, if the pivoting is restricted to each block column. This strategy makes sense in the identification context, due to the noise components (usually, the block columns have full rank). The rank deficient case is also covered when solving the associated linear systems. For reliability, an option for finding the ranks by incremental condition estimation is provided.

Several additional lower-level routines have been implemented, but they have not been included in the table above. Also, some mexfiles have been developed. They could be used both for testing purposes, but also for problem solving. The documentation of the use of the developed routines, and their integration into MATLAB and scilab via mexfiles will be documented in a SLICOT Working Note (a preliminary version has already been produced).

Vasile Sima and Michel Verhaegen

5 SLICOT tools for robust control

The NICONET Robust Control Sub-Group has been concentrating on, in the last six months, the development of robust control subroutines, testing and case studies, as well as on development of interface with `scilab`.

The following subroutines were completed and have been included in the SLICOT library:

- **SB10ZD**, implementing the discrete-time $\mathcal{H}_\infty$ loop shaping design procedure for the general case of non-zero D term;
- **SB10AD**, implementing the $\mathcal{H}_\infty$ optimal design using bisection method to decide the minimum γ;
- **SB10FD**, has been further modified for sub-optimal $\mathcal{H}_\infty$ design.

In addition, another subroutine which aims at improving the coding of μ-calculation is currently in development.

For testing of developed subroutines and as part of building up the benchmark/industrial case study collections, several practical, application design examples were studied and completed during the period. In particular, a disc drive servo system robust design has been conducted. In this exercise, the $\mathcal{H}_\infty$ mixed sensitivity optimisation, LSDP design and μ-synthesis and analysis methods have been applied. Satisfactory results were obtained with the SLICOT software in all the designs.

The interface between SLICOT and `scilab` at higher (design) levels is currently being developed. The following SLICOT routines have so far been successfully included in `scilab`:

- $\mathcal{H}_\infty$ optimal design of continuous-time systems;
- $\mathcal{H}_\infty$ optimal design of discrete-time systems;
- Eigenstructure computation and design;
- μ computation.

Together with other partners, the NICONET Robust Control Sub-Group introduced the robust control routines developed in this project and related theories at the 4th Industrial Workshop at Bremen in September, and presented successful design exercises.

Da-Wei Gu

6 SLICOT tools for nonlinear systems in robotics

6.1 Nonlinear Systems Control

During the last 6 months, a High Performance Computing sequential version of the DRESOL package has been implemented. DRESOL package is a collection of subroutines for the numerical integration of differential Riccati equations,

$$\frac{dX}{dt} = A_{21}(t) + A_{22}(t)X - XA_{11}(t) - XA_{12}(t)X + \ appropriate\ ICs,$$

where $A_{11}(t) \in R^{nnq \times nnq}$, $A_{12}(t) \in R^{nnq \times nnp}$, $A_{21}(t) \in R^{nnp \times nnq}$ and $A_{22}(t) \in R^{nnp \times nnp}$. This package is based upon the well-known stiff/nonstiff solver LSODE of Hindmarsh.

The development of this High Performance Computing sequential version of DRESOL involved basically two working lines:

- The intensive use of BLAS and LAPACK subroutines inside DRESOL subroutines. Examples are scaling operations on vectors (`SSCAL`), matrix-matrix multiplication operations (SGEMM), or reduction of a matrix to real Schur form (`SGEES`).
- The introduction of new block-oriented subroutines.

Efficiency and portability have been improved by using standard linear algebra libraries (BLAS and LAPACK).

On the other hand, we have used the KINSOL package for the problem of hydraulic simulation of water networks as a test case. As reported also in the previous newsletter, KINSOL (*Krylov Inexact Newton SOLver*) is a general purpose nonlinear system solver, callable from either C or Fortran programs.

The work is still in a preliminary state, and currently we have succeeded in the simulation of small networks (tens of nodes) without valves or pumps. For simulation of larger networks, preconditioning of the underlying systems is necessary, which will be done in the following months.

Vicente Hernandez, Enrique Arias and Fernando Alvarruiz

7 SLICOT determines models for active noise control

7.1 Introduction

The active noise control problem considers the rejection of a noise signal (sound or vibration noise) in a particular region by actively producing anti-noise. In active noise control, accurate dynamical models of the paths along which the noise travels are of paramount importance. In this note we show that subspace identification techniques can be used to obtain such models. Subspace identification is a numerically robust noniterative method to estimate multivariable state-space models from input and output data [1], [2]. SLICOT provides efficient and numerical reliable implementations of different subspace identification methods [3]. Below, we describe how the subspace identification tools from SLICOT can be used to obtain models for active noise control. The application that we present is a laboratory set-up consisting of a duct with two loudspeakers in which one loudspeaker radiates an undesirable noise and the other loudspeaker has to be controlled to eliminate the noise as much as possible at a certain location. First, we describe the laboratory set-up, then we discuss the use of the SLICOT subspace identification software.

7.2 Description of the experimental setup

The system to be modeled is an acoustical duct, which is used for active noise control experiments. Figure 2 presents a block scheme of the system. At the left end of a duct a loudspeaker is mounted that produces undesired noise. The goal is to drive the secondary loudspeaker mounted just before the other end of the duct such that at the far right end of the duct a region of silence is created. Most control algorithms used in active noise control need a model of the transfer from the secondary loudspeaker to the error microphone, called the secondary path, and from the secondary loudspeaker to the detector microphone, called the acoustical feedback. The transfer function of the secondary path is denoted by $S(z)$ and the transfer function of the acoustical feedback by $F(z)$.

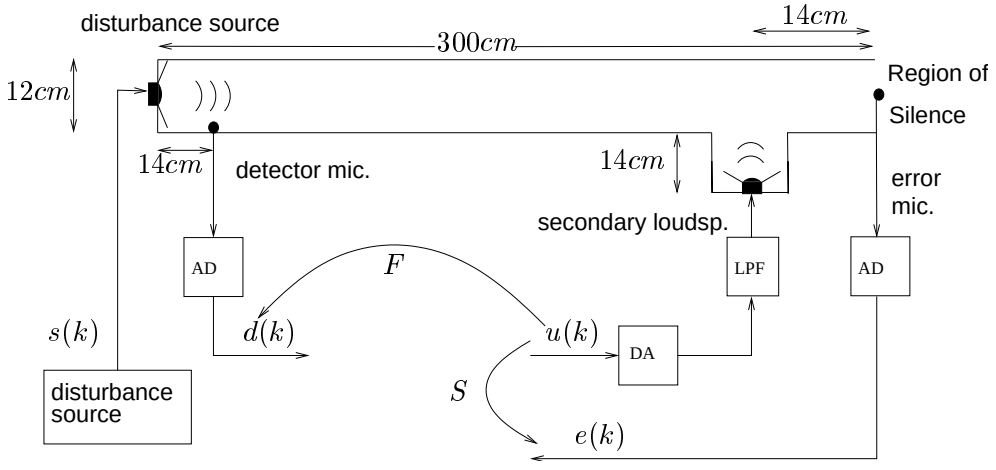

Figure 2: Schematic drawing of the acoustical duct experiment, with S the secondary path, F the acoustical feedback path, s the primary disturbance, d the detector signal, u the control signal, and e the error signal.

When the noise source is switched off, that is, there is no primary disturbance ($s(k) = 0$), we can write

$$\begin{bmatrix} e(k) \\ d(k) \end{bmatrix} = \begin{bmatrix} S(q) \\ F(q) \end{bmatrix} u(k),$$

where q is the shift operator, $u(k)$ the input to the secondary loudspeaker, $e(k)$ the signal measured by the error microphone and $d(k)$ the signal measured by the detector microphone. A state-space description of this system can be written as

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \\ \begin{bmatrix} e(k) \\ d(k) \end{bmatrix} &= \begin{bmatrix} C_e \\ C_d \end{bmatrix} x(k) + \begin{bmatrix} D_e \\ D_d \end{bmatrix} u(k), \end{aligned}$$

with $x(k) \in \mathbb{R}^n$ the state and n the order of the system. The transfer function of the secondary path and of the acoustical feedback path become $S(z) = C_e(zI - A)^{-1}B + D_e$ and $F(z) = C_d(zI - A)^{-1}B + D_d$, respectively. The sampling frequency used for the experiment is 1 kHz. The low pass filters (LPF) in Figure 2 are first order anti-aliasing filters with a cutoff frequency of 300 Hz.

7.3 Subspace identification of acoustic transfer functions using SLICOT tools

Because of the physical distance between the loudspeaker and the microphones and the presence of the AD/DA converters, there is a pure delay in the secondary path and in the acoustical feedback. These delays were determined before estimating the remaining dynamics for $S(z)$ and $F(z)$ by subspace identification. The pure delay in $S(z)$ was found to be 2 samples and in $F(z)$ to be 9 samples. The delays were removed from the corresponding signals, by shifting the signals $e(k)$ and $d(k)$ in time.

To determine $S(z)$ and $F(z)$ (without the delay) the noise source was switched off ($s(k) = 0$) and $u(k)$ was driven by a zero-mean white noise. For each of the signals $u(k)$, $e(k)$, and $d(k)$, two data sets of 4000 samples were collected. The first data set was used to estimate a model of the system; the second data set was used to validate the estimated model. The first data set is called the identification data set and the second data set the validation data set. The MOESP subspace identification method was used to determine a state-space model for the secondary path and the acoustical feedback. The calculations were performed on a 366MHz Pentium II computer with 128Mb memory running Red Hat Linux version 6.2. We used SLICOT release 4.0 [3] based on LAPACK Version 3.0 and the standard reference implementation of BLAS. To show that the use of fast implementations provided by SLICOT reduces the calculation time significantly, a comparison was made with the MATLAB MOESP implementation contained in the SMI toolbox [4]. We used SMI toolbox version 1.0 and MATLAB version 6.0.

Using the pre-processing routine `order` with 100 block rows, a data-compressed matrix was calculated from which A and C can be calculated easily. The pre-processing routine also returned 200 (that is, 100 block rows times 2 outputs) singular values. A large gap between two subsequent singular values indicates the order of the system. Figure 3 shows the first 60 singular values. From the figure, we conclude that a good initial guess of the model order is 16. However, to obtain a better insight in the true model order, models with orders between 10 and 40 were calculated. For each order, the A and C matrices were calculated using the

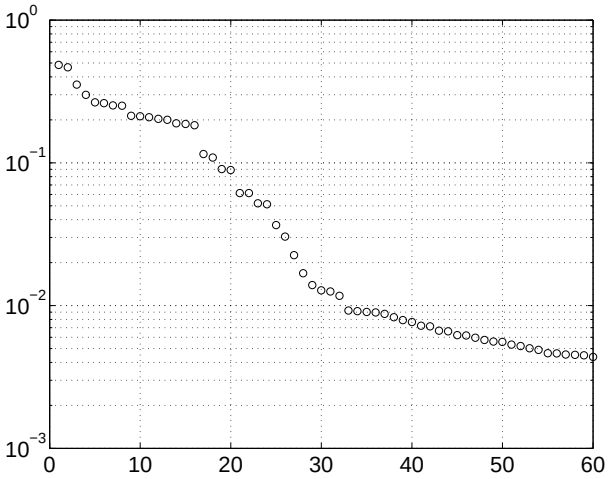

Figure 3: The first 60 singular values returned by the SLICOT pre-processing routine `order`.

routine `sident`, and the B and D matrices were calculated using the routine `findBD`. The quality of the obtained models is expressed by the Variance Accounted For (VAF), which is defined as

$$\text{VAF} = \max\left\{1 - \frac{\text{var}(y_k - \widehat{y}_k)}{\text{var}(y_k)}, 0\right\} \times 100\%,$$

where $\widehat{y}_k$ denotes the output signal of the model, y_k the measured output signal, and var$(\cdot)$ denotes the variance of a quasi-stationary signal. Figure 4 shows the VAF values for models of different orders on the identification and validation data sets.

Using Figure 4 we estimated the order of the system to be 28. The VAF values for the two outputs on the identification data and the validation data are approximately the same, and equal to 99.74% for the secondary path and 99.93% for the acoustical feedback..

Figure 5 contains the output spectra for both the secondary path and the acoustical feedback based on the measured outputs from the validation data; it also contains the spectra of the error between the measured outputs and the outputs of the model (using again the validation data). This figure shows that below 50 Hz the measured outputs only consist of measurement noise, which can be explained by realizing that the loudspeaker is not able to generate sound waves below 50 Hz. It also shows that the anti-resonances in the secondary path at 346 Hz, 408 Hz and 467 Hz are poorly identified. However, on the average the error is about 2.5–3 decades lower than the measured output, which is accurate enough for most active noise control applications.

SLICOT provides fast implementations of the pre-processing routine `order`. The calculation time of this routine is dominated by performing a QR factorization. To reduce calculation time considerably, this factorization can be replaced by a Fast-QR factorization or a Cholesky factorization [5]. Table 3 shows the average calculation time of the pre-processing routine `order` over 20 experiments using the QR, the Fast-QR, and the Cholesky factorization, and of the MATLAB routine `dordpo` from the SMI toolbox. To provide insight into the accuracy of the different implementations, Table 3 also presents the 2-norm of the distance between the singular value vector computed by the SLICOT routine `order` using the default QR algorithm and the singular value vector computed by the other implementations. The differences are

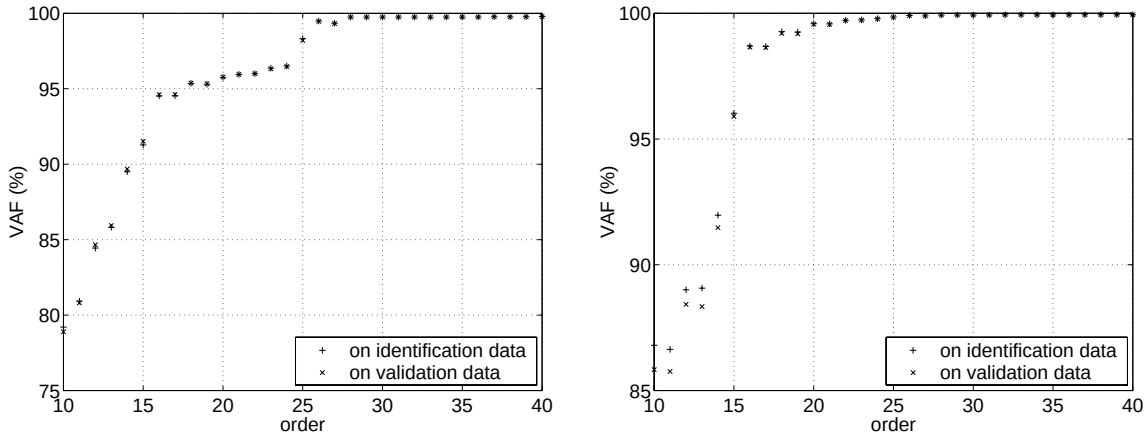

Figure 4: VAF (%) values for different model orders of the output of the secondary path (*left*) and of the acoustical feedback path (*right*).

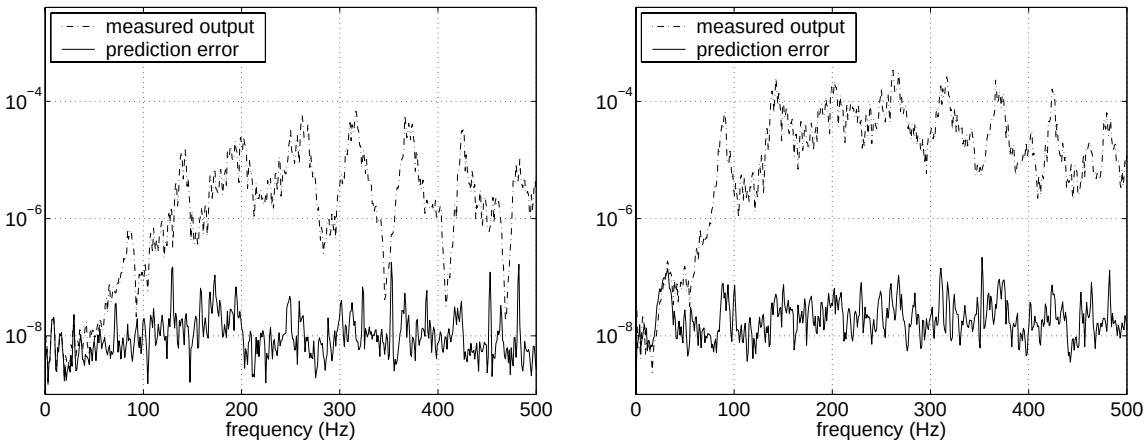

Figure 5: Spectra of the measured output of the secondary path (*left*) and of the acoustical feedback path (*right*). The *dotted* lines represent the measured outputs; the *solid* lines represent the errors between the measured outputs and the outputs of the model.

Table 3: Average calculation time of the pre-processing routine with 100 block rows over 20 experiments and 2-norm of the distance of the singular value vector returned by `order` using the QR algorithm.

	SLICOT `order` (QR)	SLICOT `order` (Fast-QR)	SLICOT `order` (Cholesky)	MATLAB `dordpo`
Calculation time (sec.)	39.3	2.44	2.32	22.7
2-norm of the difference in the singular values	0	$1.54 \cdot 10^{-14}$	$1.76 \cdot 10^{-14}$	$1.22 \cdot 10^{-15}$

Table 4: Average calculation times of A and C, and of B and D over 20 experiments using SLICOT and MATLAB routines.

	SLICOT `sident/findBD`	MATLAB `dmodpo/dac2bd`
Calculation time A, C:	0.5215	0.0182
Calculation time B, D:	1.4844	3.6958

negligibly small. The differences between the VAF values obtained from a model estimated by the SLICOT routines `sident` and `findBD`, and a model estimated by the MATLAB routines `dmodpo` and `dac2bd` are also negligible.

From the results of Table 3 we conclude that using the Fast-QR and the Cholesky factorization, the calculation time can be reduced significantly without introducing large computational errors in the singular values. The fact that the SLICOT implementation based on the QR factorization takes more calculation time than the MATLAB routine `dordpo` (also based on a standard QR factorization routine) may be due to a suboptimal choice of the work space in the SLICOT implementation by the user and/or the use of a nonoptimized BLAS implementation.

Table 4 provides the average calculation times of A and C, and of B and D over 20 experiments using the SLICOT routines `sident`, `findBD` and using the MATLAB routines `dmodpo`, `dac2bd`. We see that using SLICOT for the calculation of A and C takes more time than using the MATLAB routine; this may be due to a relatively large overhead time in the routine `sident` for allocating memory and performing several checks. However, we also see that using the SLICOT routine the calculation time of B and D can be reduced significantly. Since the calculation time for B and D is much larger than the calculation time for A and C, the SLICOT routines are faster on the whole. Figure 6 illustrates this by comparing the calculation time of B and D by different model orders for both the SLICOT and MATLAB implementations.

7.4 Conclusions

We successfully used the SLICOT tools for subspace identification to estimate from experimental data a model that can be used for active noise control of an acoustical duct. Accurate

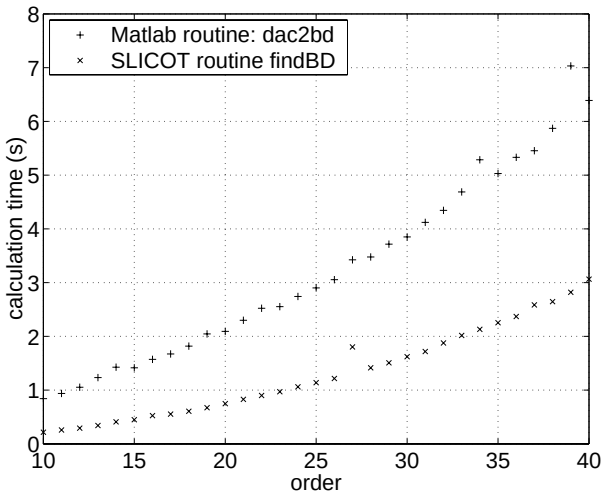

Figure 6: Time needed for the calculation of the matrices B and D for different choices of the system order, using the SLICOT routine `findBD` ($\times$) and the MATLAB routine `dac2bd` ($+$).

estimates of the acoustical transfer functions of the secondary path and of the acoustical feedback path were obtained; the VAF values for the validation data were 99.7% and 99.9%. A comparison was made among different implementations of the QR factorization used in subspace identification. The fast implementations in SLICOT can reduce the calculation time by an order of magnitude compared with the standard MATLAB implementation. The difference in accuracy of the models obtained by these different implementations is negligible.

References

[1] P. Van Overschee and B. De Moor. *Subspace Identification for Linear Systems; Theory, Implementation, Applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1996.

[2] M. Verhaegen. Identification of the deterministic part of MIMO state space models given in innovations form from input-output data. *Automatica* **30**(1), 1994, pp. 61–74.

[3] NICONET International Society. *The SLICOT package*. Secretariat: Mrs. Ida Tassens, Department of Electrical Engineering, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium.

[4] B. Haverkamp and M. Verhaegen. *State Space Model Identification Software for Multivariable Dynamical Systems*. Delft University of Technology, Systems and Control Engineering Group, Delft, The Netherlands, 1997.

[5] V. Sima. Cholesky or QR Factorization for Data Compression in Subspace-based Identification. *Proc. 2nd NICONET Workshop on "Numerical Control Software: SLICOT, a Useful Tool in Industry"*, Dec. 3, 1999, INRIA Rocquencourt, France, pp. 75-80, 1999.

Vincent Verdult and Rufus Fraanje

8 SLICOT: a useful tool in industry?

8.1 Using SLICOT in Controllabs Products

8.1.1 Introduction

Controllab Products BV (CLP) is a spin-off company of the Control Laboratory of the University of Twente. CLP is engaged with two things:

1. CLP is the creator and distributor of 20-sim^{TM}, a software package for modeling and simulation of dynamic systems for Microsoft Windows.

2. Design: CLP designs on demand technical software and offers consultancy for control engineering and mechatronics.

With 20-sim you can simulate the behavior of dynamic systems, such as electric, mechanical and hydraulic systems or any combination of these systems. 20-sim has been developed at the Control Laboratory of the University of Twente, as successor of the famous TUTSIM package.

20-sim fully supports graphical modeling, allowing you to design and analyze dynamic systems in an intuitive and user-friendly way, without compromising power. Some of the key features of 20-sim are:

- rapid system modeling through iconic diagrams, block diagrams, bond graphs and equations
- fully observable, unlimited hierarchical model structure
- active support of top-down, inside-out and bottom-up modeling
- multiple libraries with a large set of domain oriented models
- inspect any library model and change it for your own use
- save models for reuse later
- add graphical elements (lines, arrows, rectangles, text) to your models
- advanced simulation algorithms with high simulation speeds
- advanced debugging facilities
- import and export of data to MATLAB, also during a simulation run

Two important tools in 20-sim are the Linear System Editor and the Controller Design Editor. These tools use functionality of the SLICOT, LAPACK and BLAS libraries.

8.1.2 Linear System Editor

The Linear System Editor is a specialized tool for the design and analysis of linear systems. The editor supports continuous-time and discrete-time SISO systems using various representations. Standardized plots enable you to quickly evaluate system behavior.

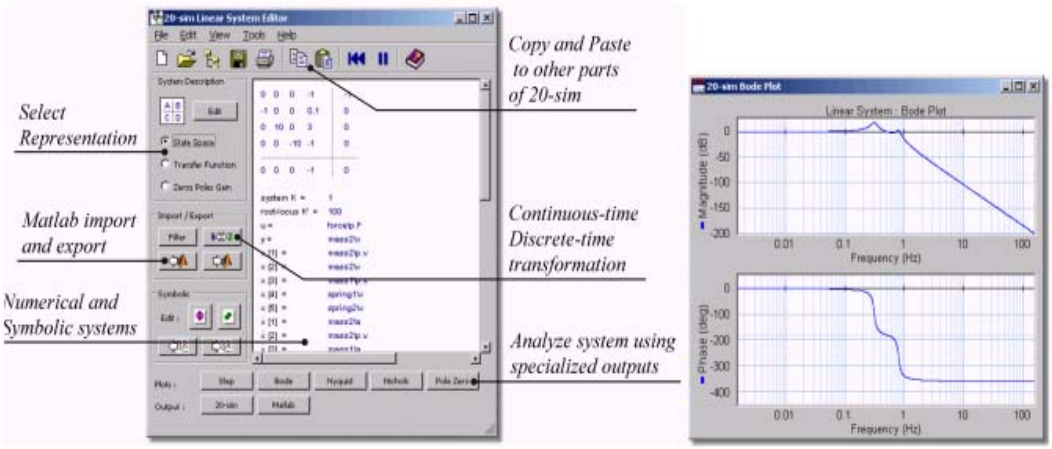

Figure 1: Linear System Editor in 20-sim

A graphical interface allows you to edit a linear system in any desired form: State Space (ABCD), Transfer Function or Zero Pole Gain. Designing a linear system is performed interactively by editing the system and observing its behavior in time and frequency domain. Input can originate from a 20-sim Linear System Model, 20-sim linearization output, the 20-sim Filter Editor, clipboard, MATLAB workspace or from the user. Output can be generated for a 20-sim linear system model, the 20-sim Filter Editor, clipboard and MATLAB workspace.

The linear models can either be numeric (numerical values as parameters) or symbolic (variables and equations as parameters). Symbolic linear models are related to their (non-linear) parent models because they share the same parameter space.

Some important features of the Linear System Editor are:

- Editing as ABCD State Space, Transfer Function or Zeros Poles Gain with automatic transformation between these forms.
- Generate and edit continuous-time and discrete-time models.
- Handles numeric and symbolic models.
- Time and frequency plots: Step Response, Bode Plot, Nyquist Diagram, Nichols Chart and Pole-Zero Plot.

Plot Options

- Multi-view: show various plots of same system.
- Multi-system: show one plot of various systems.
- Inspect numerical values.
- Zoom in / zoom out.
- Copy to Clipboard and Print.

System Parameters

- Phase Margin.
- Gain Margin.
- Rise Time.
- Steady State Value.
- Overshoot.
- Modulus Margin.

Linear System Representation Transformations
The transformation between the different linear system representations is performed using the SLICOT and LAPACK routines.

From Transfer Function to ABCD system:

AB08ND	[1]	create a System Pencil: $\lambda B - A$
TD04AD	[3]	

From ABCD system to Transfer Function:

TB04AD	[2]

From ABCD to Zeros Poles Gain:

dgees	[6]	find the eigenvalues of the A, and the poles of the system
AB08ND	[1]	create a System Pencil: $\lambda B - A$
dgegv	[7]	find the zeros using the System Pencil.

From Zeros Poles Gain to Transfer Function:
No SLICOT routines are necessary to perform this task. Creating a polynomial of a given set of zeros is straightforward. E.g., $(s - z_1)(s - z_2) \cdots (s - z_n)$. Using these three functions

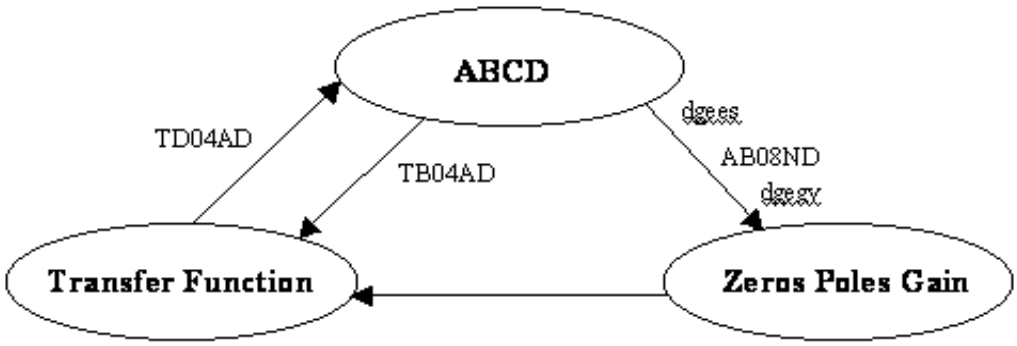

Figure 2: transformation between linear system representations

makes it possible to inspect the linear system in any of the three possible representations. In

order to create the frequency plots for the linear system the SLICOT routine TD05AD [4] is used.

Discrete-time ↔ continuous-time systems conversion

In 20-sim it is possible to create a discrete-time system from a continuous-time system and vice-versa with the following transformations:

- Euler
- Backward Difference
- Tustin (bilinear transformation)

The Euler and Backward Difference methods are straightforward by applying the transformation functions:

$$z = sh + 1,$$

for the Euler method and

$$z = 1/(1 - sh),$$

for the Backward Difference method. For the Tustin transformation the SLICOT routine AB04MD [5] is used.

8.1.3 Linearization

Any 20-sim model can be linearized to generate a linear continuous-time or discrete-time state-space model. Linearization can be performed at any point of a simulation run. The result is a linear model that is presented in the Linear System Editor. The linear models can either be numeric (numerical values as parameters) or symbolic (variables and equations as parameters). Symbolic linear models are related to their (non-linear) parent models because they share the same parameter space.

8.1.4 Controller Design Editor

The Controller Design Editor is a specialized tool for the design of feedback control systems. A feedback structure of subsystems is presented with a linear plant, controller, measurement and prefilter. All the subsystems can be edited and inspected in much the same way as in the Linear System Editor. Also the open-loop and closed-loop systems and the sensitivities are available.

Immediate Results

Changes in one of the subsystems directly update all open plots and dialogs. Adapting for instance the controller gain, immediately changes poles and zeros of the closed-loop system and the overall step response. The integration within 20-sim and linear system exchange with MATLAB makes this editor a powerful tool for designing feedback control systems.

Some important additional features of the controller design editor are:

- Additional ports allow the connection of disturbances and feedforward signals to your controlled system.
- Template models make it easy to reuse your design (e.g., to replace the linear plant with a non-linear iconic-diagram implementation).

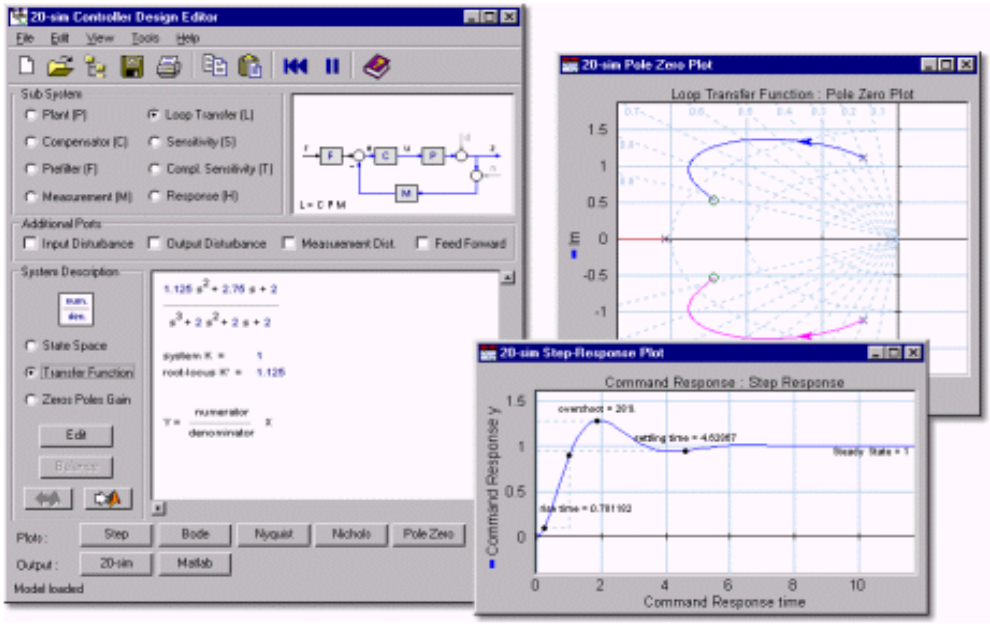

Figure 3: Controller Design Editor

8.1.5 Conclusions and Future Work

We think that 20-sim is a powerful tool which enables the user to create models of physical systems possibly in combination with a controller to obtain better system behavior. Models can be linearized which result in a Linear System Model which can be inspected and edited in the Linear System Editor or Controller Design Editor. In these editors some SLICOT routines are used which make it a powerful and user-friendly tool and an important enhancement of 20-sim.

We are planning to extent the Controller Design Editor with automatic controller design. For this purpose the SLICOT library for Identification and Control can be used.

Although the SLICOT library functions are stable and fast, implementing them in a C or C++ environment like 20-sim is not straightforward. We used the utility f2c to convert the Fortran files of SLICOT, LAPACK and BLAS to C. In this way it is almost impossible to use more than just a small subset of files necessary for a particular problem. Also the interface of the library routines is sometimes difficult, for example some of the functions take no less than 29 arguments (AB08ND).

Contact

A fully working viewer version of 20-sim can be downloaded from the 20-sim web-site: `www.20sim.com`. Also on this web-site more information can be found how to contact Controllab Products BV.

References

Library Routines used
SLICOT

[1]	AB08ND	System zeros and Kronecker structure of system pencil
[2]	TB04AD	Transfer matrix of a state-space representation
[3]	TD04AD	Minimal state-space representation for a proper transfer matrix
[4]	TD05AD	Evaluation of a transfer function for a specified frequency
[5]	AB04MD	Discrete-time $\leftrightarrow$ continuous-time conversion by bilinear transformation

LAPACK

[6]	dgees	computes for an $n \times n$ real nonsymmetric matrix A, the eigenvalues, the real Schur form T, and, optionally, the matrix of Schur vectors Z.
[7]	dgegv	computes for a pair of $n \times n$ real nonsymmetric matrices A and B, the generalized eigenvalues, and optionally, the left and/or right generalized eigenvectors.

Frank N.J. Groen, Controllab Products B.V.

9 Highlights of the SLICOT training course in Bremen, Germany

The workshop and training course

ADVANCED COMPUTATIONAL TOOLS FOR COMPUTER-AIDED CONTROL SYSTEMS DESIGN

took place September 27–29, 2001, at University of Bremen, Germany

This workshop and training course was intended as a tutorial on the use of the freeware Subroutine Library in Systems and Control Theory (SLICOT) for solving practical control engineering problems within computer-aided control systems design (CACSD) environments. The exercises solved by the participants during the hands-on training sessions demonstrated that SLICOT-based software usually has improved reliability and efficiency as well as extended functionality compared to the computational methods implemented in other CACSD software packages.

The workshop was organized by Angelika Bunse-Gerstner and Peter Benner from the Zentrum für Technomathematik/AG Numerik of the University of Bremen, Germany, in cooperation with the Numerics in Control Network (NICONET) funded by the European Community BRITE-EURAM III Thematic Networks Programme.

The workshop was attended by 39 participants from Belgium, Bulgaria, France, Germany, Great Britain, The Netherlands, Romania, Spain, Sweden, and Switzerland. The workshop provided a stimulating atmosphere for discussions on how to use SLICOT-based software in industrial applications. Some posters during the poster session on Thursday, September 27, showed the performance of the SLICOT software for use in MATLAB and `scilab` applied to real-world problems.

Major topics of the course were

- basic control software
- system identification
- model reduction
- robust control design using H-infinity techniques

The program of the workshop was as follows:

Thursday, September 27, 2001

09.00–09.15	S. Van Huffel, A. Bunse-Gerstner: Welcome addresses
09.15–09.45	S. Van Huffel: Introduction to NICONET and SLICOT
09.45–10.30	S. Hammarling: The backbone of reliable numerical software: BLAS and LAPACK
10.30–11.00	Coffee break
11.00–12.00	H. Stahl: The MATLAB Control Toolbox
12.00–13.30	Lunch
13.30–14.15	P. Van Dooren, P. Benner: Basic control software
15.00–16.00	Coffee & Posters
16.00–19.00	V. Sima: Training Part I

Friday, September 28, 2001

09.00–10.30 V. Verdult: System identification with SLICOT
10.30–11.00 Coffee break
11.00–12.30 A. Varga: SLICOT model reduction tools
12.30–14.00 Lunch
14.00–15.00 J. de Cuyper: Applications of SLICOT in industry
15.00–15.30 Coffee break
15.30–18.30 V. Sima: Training II
19:30–23:00 Banquet

Saturday, September 29, 2001

09.00–10.30 P. Petkov, D.-W. Gu: Robust control design using SLICOT
10.30–11.00 Coffee break
11.00–13.00 V. Sima: Training III
13.00–14.00 Lunch

During the poster session participants presented their own projects and discussed current problems or open questions. The following posters were presented:

- Peter Benner, Enrique Quintana-Ortí, Gregorio Quintana-Ortí: *Model reduction of large-scale dense systems*
- Younès Chahlaoui, Antoine Vandendorpe, Paul Van Dooren: *Model reduction of large-scale dynamical systems via Multipoint Padé*
- Francois Delebecque, Serge Steer: *scilab-SLICOT*
- Erik Elmroth, Pedher Johansson, Bo Kågström, Daniel Kressner, Volker Mehrmann: *SLICOT Web Computing*
- Heike Faßbender, Peter Benner: *SLICOT drives tractors!*
- Sven Hammarling: *LAPACK and ScaLAPACK: The underlying linear algebra for SLICOT*
- Inès Ferrer-Mallorquí: *Interval-based tools for robust control*
- Ivan Markovsky, Sabine Van Huffel: *Element-Wise Weighted Total Least Squares*
- Ivan Markovsky, Sabine Van Huffel, Bart De Moor: *Multi-Model System Parameter Estimation*
- Horst Schulte: *Motion control for servo-pneumatic actuators using Takagi-Sugeno fuzzy models*
- Diana Maria Sima, Vasile Sima, Sabine Van Huffel: *Recent Developments in SLICOT Library for System Identification*

For further information on the workshop see

`http://www.math.uni-bremen.de/zetem/workshops/cacsd`

Peter Benner

10 NICONET information corner

This section informs the reader on how to access the SLICOT library, the main product of the NICONET project, and how to retrieve its routines and documentation. Recent updates of the library are also described. In addition, information is provided on the newest NICONET reports, available via the NICONET website or ftp site, as well as information about upcoming workshops/conferences organized by NICONET or with a strong NICONET representation.

Additional information about the NICONET Thematic Network can be found from the NICONET homepage World Wide Web URL

`http://www.win.tue.nl/wgs/niconet.html`

10.1 Electronic Access to the Library

The SLICOT routines can be downloaded from the WGS ftp site,

`ftp://wgs.esat.kuleuven.ac.be`

(directory `pub/WGS/SLICOT/` and its subdirectories) in compressed (gzipped) tar files. On line `.html` documentation files are also provided there. It is possible to browse through the documentation on the WGS homepage at the World Wide Web URL

`http://www.win.tue.nl/wgs/`

after linking from there to the SLICOT web page and clicking on the `FTP site` link in the freeware SLICOT section. The SLICOT index is operational there. Each functional "module" can be copied to the user's current directory, by clicking on an appropriate location in the `.html` image. A "module" is a compressed (gzipped) tar file, which includes the following files: source code for the main routine and its example program, example data, execution results, the associated `.html` file, as well as the source code for the called SLICOT routines.

The entire library is contained in a file, called `slicot.tar.gz`, in the SLICOT root directory `/pub/WGS/SLICOT/`. The following Unix commands should be used for decompressing this file:

```
gzip -d slicot.tar
tar xvf slicot.tar
```

The created subdirectories and their contents is summarized below:

`slicot`	contains the files `libindex.html`, `make.inc`, `makefile`, and the following subdirectories:
`benchmark_data`	contains benchmark data files for Fortran benchmark routines (`.dat`);
`doc`	contains SLICOT documentation files for routines (`.html`);
`examples`	contains SLICOT example programs, data, and results (`.f`, `.dat`, `.res`), and `makefile`, for compiling, linking and executing these programs;
`src`	contains SLICOT source files for routines (`.f`), and `makefile`, for compiling all routines and creating an object library;
`SLTools`	contains MATLAB `.m` files and data `.mat` files;
`SLmex`	contains Fortran source codes for mexfiles (`.f`).

Another, similarly organized file, called `slicotPC.zip`, is available in the SLICOT root directory; it contains the MS-DOS version of the source codes of the SLICOT Library, and can be used on Windows 9x/2000 or NT platforms. Included are several source makefiles.

After downloading and decompressing the appropriate SLICOT archive, the user can then browse through the documentation on his local machine, starting from the index file `libindex.html` from `slicot` subdirectory.

10.2 SLICOT Library updates in the period July 2001—December 2001

There have been one major SLICOT Library update during the period July 2001—December 2001: on October 3. Details are given in the files `Release.Notes` and `Release.History`, located in the directory `pub/WGS/SLICOT/` of the ftp site.

Few changes have been made in the routines `AB09IY`, `IB01PD`, `IB01BD`, `IB01PD`, `IB01PX`, `IB01PY`, `SB16AD`, and `SB16BD`, for removing some bugs. Also, five m/mexfiles (`slinorm.m`, `aresol.f`, `aresolc.f`, `findBD.f`, and `find_models.m`) have been updated. Details are given in the file `Release.Notes`.

A new routine, and associated mexfile and example program, have been added to compute the matrices of a positive feedback controller in the Discrete-Time Loop Shaping Design Procedure.

In addition, the file plicmr.tar.gz has been added to the SLICOT root directory. It contains 3 model reduction routines, 7 associated routines (solvers for coupled stable Lyapunov/Stein equations, stable Sylvester equations, etc.), and 12 lower-level routines, all for parallel computers. Details are given in the incorporated .html documentation, accessible from the file index.html, included in the archive.

Over 20 new user-callable and computational routines for basic control problems, and identification of Wiener systems, have been implemented and will be posted soon on the SLICOT ftp site. They include *Identification Routines*, *Mathematical Routines*, and *Transformation Routines*, performing the following main computational tasks:

- compute a set of parameters for approximating a Wiener system in a least-squares sense, using a neural network approach and a Levenberg-Marquardt algorithm.
- solve a system of linear equations $Ax = b$, with A symmetric, positive definite, or, in the implicit form, $f(A, x) = b$, where $y = f(A, x)$ is a symmetric positive definite linear mapping from x to y, using the conjugate gradient algorithm without preconditioning.
- solve a set of systems of linear equations, $A^TAX = B$, or, in the implicit form, $f(A)X = B$, with A^TA or $f(A)$ positive definite, using symmetric Gaussian elimination.
- solve a system of linear equations $Ax = b$, $Dx = 0$, in the least squares sense, with D a diagonal matrix, given a QR factorization with column pivoting of A.
- find the parameters θ for a function $F(x, \theta)$ that give the best approximation for $y = F(x, \theta)$ in a least-squares sense using a Levenberg-Marquardt algorithm based on conjugate gradients for solving linear systems.
- find the parameters θ for a function $F(x, \theta)$ that give the best approximation for $y = F(x, \theta)$ in a least-squares sense using a Levenberg-Marquardt algorithm based on QR factorization with column pivoting.

- compute the QR factorization with column pivoting of an $m \times n$ matrix J ($m \geq n$), that is, $JP = QR$, where Q is a matrix with orthogonal columns, P a permutation matrix, and R an upper trapezoidal matrix with diagonal elements of nonincreasing magnitude, and apply the transformation Q^T on the error vector e; the 1-norm of the scaled gradient is also returned.

- find a value for the parameter λ such that if x solves the system $Ax = b$, $\lambda^{1/2}Dx = 0$, in the least squares sense, where A is an $m \times n$ matrix, D is an $n \times n$ nonsingular diagonal matrix, and b is an m-vector, and if δ is a positive number, then either $\lambda = 0$ and $(\|Dx\|_2 - \delta) \leq 0.1\delta$, or $\lambda > 0$ and $|\, \|Dx\|_2 - \delta \,| \leq 0.1\delta$. It is assumed that a QR factorization with column pivoting of A is available, that is, $AP = QR$, where P is a permutation matrix, Q has orthogonal columns, and R is an upper triangular matrix with diagonal elements of nonincreasing magnitude.

- compute the output of a Wiener system.

- compute the output of a set of neural networks.

- compute the Jacobian of a Wiener system.

- find a value for the parameter λ such that if x solves the system $Jx = b$, $\lambda^{1/2}Dx = 0$, in the least squares sense, where J is an $m \times n$ matrix, D is an $n \times n$ nonsingular diagonal matrix, and b is an m-vector, and if δ is a positive number, then either $\lambda = 0$ and $(\|Dx\|_2 - \delta) \leq 0.1\delta$, or $\lambda > 0$ and $|\, \|Dx\|_2 - \delta \,| \leq 0.1\delta$. It is assumed that a QR factorization with block column pivoting of J is available, that is, $JP = QR$, where P is a permutation matrix, Q has orthogonal columns, and R is an upper triangular matrix with diagonal elements of nonincreasing magnitude for each block.

- solve a system of linear equations $Jx = b$, $Dx = 0$, in the least squares sense, with D a diagonal matrix, given a QR factorization with block column pivoting of J.

- solve one of the systems of linear equations $Rx = b$, or $R^T x = b$, in the least squares sense, where R is an $n \times n$ block upper triangular matrix, with the structure

$$\left[\begin{array}{cccc|c} R_1 & 0 & \cdots & 0 & L_1 \\ 0 & R_2 & \cdots & 0 & L_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & R_\ell & L_\ell \\ 0 & 0 & \cdots & 0 & R_{\ell+1} \end{array}\right],$$

  with the upper triangular submatrices R_k, $k = 1\colon \ell + 1$, square, and the first ℓ of the same order. The diagonal elements of each block R_k have nonincreasing magnitude. The matrix R is stored in a compressed form.

- compute the QR factorization with block column pivoting of an $m \times n$ matrix J ($m \geq n$), that is, $JP = QR$, where Q is a matrix with orthogonal columns, P a permutation matrix, and R an upper trapezoidal matrix with diagonal elements of nonincreasing magnitude for each block, and apply the transformation Q^T on the error vector e; the 1-norm of the scaled gradient is also returned.

- compute the matrix $J^TJ + cI$, for the Jacobian J given in a compressed form.
- compute the matrix $J^TJ + cI$, for the Jacobian J fully given, for one output variable.
- compute the matrix-vector product $x \leftarrow (J^TJ + cI)x$, where J is given in a compressed form.
- compute $x \leftarrow (A^TA + cI)x$, where A is an $m \times n$ real matrix, and c is a scalar.
- compute the Jacobian of the error function for a neural network (for one output variable).
- convert the linear discrete-time system given as (A, B, C, D), with initial state x_0, into the output normal form, with parameter vector θ. The matrix A is assumed to be stable. The matrices A, B, C, D and the vector x_0 are transformed, so that on exit they correspond to the system defined by θ.
- convert the linear discrete-time system given as its output normal form, with parameter vector θ, into the state-space representation (A, B, C, D), with the initial state x_0.
- compute the output sequence of a linear time-invariant open-loop system given by its discrete-time state-space model (A, B, C, D), where A is an $n \times n$ general matrix (the input and output trajectories are stored differently from SLICOT Library routine `TF01MD`).

10.3 New NICONET Reports

Recent NICONET reports (available after June 2001), that can be downloaded as compressed postscript files from the World Wide Web URL

`http://www.win.tue.nl/wgs/reports.html`

or from the WGS ftp site,

`ftp://wgs.esat.kuleuven.ac.be`

(directory `pub/WGS/REPORTS/`), are the following:

- Isak Jonsson and Bo Kågström. *Recursive Blocked Algorithms for Solving Triangular Matrix Equations—Part I: One-Sided and Coupled Sylvester-Type Equations* (file `SLWN2001-4.ps.Z`, revised August 2001).

  Triangular matrix equations appear naturally in estimating the condition numbers of matrix equations and different eigenspace computations, including block-diagonalization of matrices and matrix pairs and computation of functions of matrices. To solve a triangular matrix equation is also a major step in the classical Bartels-Stewart method. We present recursive blocked algorithms for solving one-sided triangular matrix equations, including the continuous-time Sylvester and Lyapunov equations, and a generalized coupled Sylvester equation. The main parts of the computations are performed as level 3 general matrix multiply and add (GEMM) operations. Recursion leads to an automatic variable blocking that has the potential of matching the memory hierarchies of today's HPC systems. Different implementation issues are discussed, including when to end

the recursion, the design of optimized superscalar kernels for solving leaf-node triangular matrix equations efficiently, and how parallelism is utilized in our implementations. Uniprocessor and SMP parallel performance results of our recursive blocked algorithms and corresponding routines in the state-of-the-art libraries LAPACK and SLICOT are presented. The performance improvements of our recursive algorithms are remarkable, including 10-folded speedups compared to standard algorithms.

- Isak Jonsson and Bo Kågström. *Recursive Blocked Algorithms for Solving Triangular Matrix Equations—Part II: Two-sided and Generalized Sylvester and Lyapunov Equations* (file `SLWN2001-5.ps.Z`, September 2001).

  We continue our study on high-performance algorithms for solving triangular matrix equations. They appear naturally in different condition estimation problems for matrix equations and various eigenspace computations, and as reduced systems in standard algorithms. Building on our successful recursive approach applied to one-sided matrix equations (Part I), we now present recursive blocked algorithms for two-sided matrix equations, which include matrix product terms such as AXB^T. Examples are the discrete-time standard and generalized Sylvester and Lyapunov equations. The means for high-performance are the recursive variable blocking, which has the potential of matching the memory hierarchies of today's high-performance computing systems, and level 3 computations which mainly are performed as GEMM operations. Different implementation issues are discussed, focusing on similarities and differences between one-sided and two-sided matrix equations. We present uniprocessor and SMP parallel performance results of recursive blocked algorithms and routines in the state-of-the-art SLICOT library. The performance improvements of our recursive algorithms are remarkable, including 10-folded speedups or more, compared to standard algorithms.

- Chris Denruyter. *Solving Sylvester equations for Model Reduction: SLICOT vs.* MATLAB (file `SLWN2001-6.ps.Z`, October 2001).

  In this report, we compare two Sylvester equation solvers: the MATLAB function lyap and the SLICOT function slsylv. An algorithm designed for model reduction and based on the resolution of a Sylvester equation is presented. In this context, timing results show the superiority of the SLICOT based m.file slsylv.

- P.Hr. Petkov, D.W. Gu and MM. Konstantinov. *Robust control of a disk drive servo system* (file `SLWN2001-7.ps.Z`, December 2001).

  In this expository paper we show the application of some of the SLICOT routines in the robust control analysis and design of a disk drive servo system. An uncertainty model of the system plant is first derived which contains eleven uncertain parameters including four resonance frequencies, four damping coefficients and three rigid body model parameters. Three controllers for the uncertain system are designed using, respectively, the techniques of H_∞ mixed sensitivity design, H_∞ loop shaping design procedure (LSDP) and μ-synthesis method. With these controllers the closed-loop system achieves robust stability and in the cases of H_∞ and μ-controllers the closed loop system practically achieves robust performance. A detailed comparison of the frequency domain and time domain characteristics of the closed-loop system with the three controllers is conducted. Further, model reduction routines have been applied to find a reasonably low order controller based on the μ-synthesis design. This reduced order controller maintains the

robust stability and robust performance of the closed-loop system. Simulations of the nonlinear sampled-data servo system with the low order controller have been included as well, which confirms the practical applicability of the controller obtained.

- Peter Benner, Enrique S. Quintana-Orti, Gregorio Quintana-Orti, and Rafael Mayo. *Enhanced Services for Remote Model Reduction of Large-Scale Dense Linear Systems* (file `SLWN2002-1.ps.Z`, January 2002).

  This paper describes enhanced services for remote model reduction of large-scale, dense linear time-invariant systems. Specifically, we describe a mail service and a web service for model reduction on a cluster of Intel Pentium-II architectures using absolute and relative error methods. Experimental results show the appeal and accessibility provided by these services.

Previous NICONET/WGS reports are also posted at the same address.

10.4 Forthcoming Conferences

Forthcoming Conferences related to the NICONET areas of interest, where NICONET partners submitted proposals for NICONET/SLICOT-related talks and papers, and disseminated or will disseminate information and promote SLICOT, include the following:

- CG50-GG70 meeting, a conference commemorating 50 years of conjugate gradients and celebrating Gene Golub's 70th birthday, ETH Zurich, February 18-20, 2002.
- GAMM Annual meeting, Augsburg, Germany, March 25-28, 2002.
- IFAC symposium on parameter estimation, Barcelona, Spain, March 2002.
- 15th Householder symposium on Numerical Linear Algebra, Peebles Hotel Hydro, Scotland, June 17–21, 2002.
- MTNS, "Mathematical Theory of networks and Systems" meeting 2002, University of Notre Dame, South Bend, Indiana, USA, August 12-16, 2002, see `http://www.nd.edu/~mtns/`
- Joint "IEEE Conference on Control Applications" and "IEEE Conference on Computer Aided Control Systems Design", September 17-20, 2002, Scottish Exhibition & Conference Centre, Glasgow, Scotland.

Vasile Sima