ADVANCED COMPUTATIONAL TOOLS FOR COMPUTER-AIDED CONTROL SYSTEM DESIGN (CACSD)

Tutorial Workshop

organized by

Peter Benner Paul Van Dooren

http://www.win.tue.nl/niconet/society

European Control Conference 2003
Cambridge, UK, September 1–4, 2003

Preface

With the ever-increasing complexity of control systems, efficient computational methods for their analysis and design are becoming more and more important. These computational methods need to be based on reliable and robust numerical software provided by well-tested and user-friendly software libraries.

This workshop is intended as a tutorial on recent developments in advanced reliable and efficient computational methods for solving analysis and synthesis problems of modern and robust control. Moreover, the importance of providing corresponding software implementations is demonstrated using the freeware Subroutine Library in Systems and Control Theory (SLICOT) for solving practical control engineering problems within CACSD environments. 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 like the MATLAB Control Toolbox. The SLICOT software library and the related CACSD tools based on SLICOT were developed within the *Numerics in Control Network (NICONET)* funded by the European Community BRITE-EURAM III RTD Thematic Networks Programme. We will present some of the activities within NICONET and introduce SLICOT-based software to be used either within MATLAB and the MATLAB Control Toolbox or the CACSD package Scilab.

Major topics of the course are *basic control software*, *system identification*, *model reduction*, and *robust control design using H_∞ techniques*.

The course will be particularly interesting for advanced graduate students and young researchers in systems and control theory who are engaged in the solution of practical control problems.

Peter Benner and Paul Van Dooren

Table of Contents

Introduction to NICONET and SLICOT

Paul Van Dooren
Department of Mathematical Engineering
Université Catholique de Louvain-la-Neuve
Avenue Georges Lemaitre 4
B-1348 Louvain-la-Neuve (Belgium)

E-mail: vdooren@csam.ucl.ac.be
URL: http://www.auto.ucl.ac.be/~vdooren

Abstract

The aims and scope of the European thematic network NICONET will be presented. The requirements of robust numerical software for solving control engineering problems will be emphasized. Moreover, the contents and structure of the software library SLICOT and the embedding of SLICOT-based CACSD tools in user-friendly environments like MATLAB and Scilab will be discussed.

Overview

- Introduction
- Why numerics
- A bit of history
  - Working Group on Software
  - SLICOT
- SLICOT and NICONET
- Conclusion

Introduction

Systems and control used in real world applications

requires a good balance between :

1. Theory
2. Design methodology
3. Numerical algorithms
4. Software implementation
5. Integration in an application

Computer Aided Control Systems Design tries to address this

Is CACSD the answer ?

There are several useful developments

- software environments
- data structures
- user interaction and GUI's ...

but there are also many problems

- real applications are often ill-posed or large-scale
- simple algorithms often fail in practice
- need for reliable, high performance and robust software
- need for good benchmarks

Why numerics in control ?

Packages like MATLAB have pro's and con's

Pro's : powerful tool because

- flexible for developing new algorithmic ideas
- user-friendly and interactive
- widely used in academia

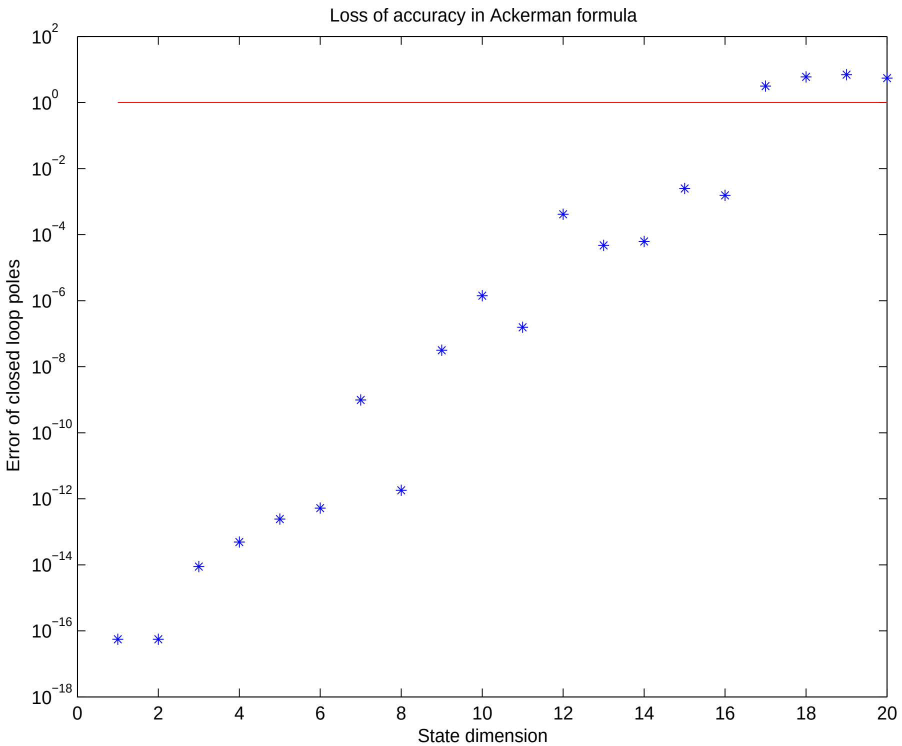
Con's : sometimes poor performance due to

- MATLAB's data structure
  use of dense matrix as main data structure
- structure in control problems
  exploiting structure leads to large overhead
- simple "academic" algorithms
  Control Toolbox is one of the oldest
- often sacrifice efficiency for flexibility

Pole placement example

```
for n=1:20;A=randn(n,n);B=randn(n,1);L=randn(n,1);F=acker(A,B,L);
Lcomp=eig(A-B*F);err(n)=norm(sort(L)-sort(Lcomp));end
```

The closed loop eigenvalues loose all accuracy for $n > 15$



Advantages of Fortran libraries

– can be integrated in CACSD platforms
  rely on robust numerical software (RASP, SLICOT, ...)
– layer of computational routines
  basic mathematical routines (linear algebra, simulation, optimization, ...)
– development of Control library
  choice of robust control algorithms
– reusability of developed software

Position of Control Library

– True independence of CACSD platforms
– Use of high performance linear algebra software
– Better use of low level routines
– Fortran and C allow better exploitation of structure
– Automated Fortran to C conversion

A bit of history

Retrospect

- 70's : Scandinavian Control library, Swiss library AUTLIB
- 80's : SLICE (+NAG), BIMAS(C), LISPACK, SYSLAB, RASP
- 90's : SLICOT (WGS) still active !

WGS and SLICOT

- Benelux initiative involving several universities
- Collaboration with NAG and DLR
- Extension with European universities
- Evolved to NICONET with EU support

SLICOT and NICONET

Subroutine LIbrary for COntrol Theory

- mathematical library for control theoretic computations
- main emphasis on numerical reliability, robustness and efficiency
- selection of robust and reliable algorithms
- rigorous implementation and standardization
- over 200 user-callable routines
- copyrighted software
- ftp downloadable
- chapters and subchapters
- user manual
- benchmarks
- driver routines
- LAPACK-based

Contents of SLICOT

- A : Analysis routines
- B : Benchmarks and test programs
- D : Data analysis
- F : Filtering
- I : Identification
- M : Mathematical
- S : Synthesis
- T : Transformations
- U : Utility

over 200 example programs

over 400 documented routines

over 100 MATLAB/SCILAB m-files

dozens of MATLAB/SCILAB mex-files

NICONET PROJECT (BRITE-EURAM 1998–2002)

- EU/BRITE-EURAM thematic network
  (1998–2002, preparatory phase 1996/97)
- Involved 7 countries, 9 universities, 2 research institutes and 6 companies
- developed benchmarks and maintained SLICOT
- integrated LAPACK in SLICOT
- integrated SLICOT in MATLAB and SCILAB
- maintains and updates the freeware library

Niconet Subtasks

- Task I : Basic numerical tools
- Task II : Model reduction
- Task III : Identification
- Task IV : Robust control
- Task V : Nonlinear systems in robotics

International Society NICONET (founded January 2001)

aim is to

- stimulate research and development of software,
- maintain and publish SLICOT (copyrighted freeware),
- collect and distribute information on new software,
- publish in journals and present at conferences,
- provide commercial licenses (for commercial use).

See http://www.win.tue.nl/niconet/society

Conclusions

- SLICOT offers better numerics
- SLICOT is often much faster
- NICONET integration SLICOT in MATLAB and SCILAB
- NICONET provides benchmarks and test programs
- NICONET also issues a newsletter
- SLICOT freely available for non-commercial use

see also
P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga, *SLICOT - a subroutine library in systems and control theory*, Applied and Computational Control, Signals, and Circuits, **1**(10), 499–539, Birkhäuser, Boston, MA, 1999.

Basic Control Software:

System Analysis, Synthesis, and Matrix Equations

Peter Benner
Institut für Mathematik (MA 4-5)
Technische Universität Berlin
Straße des 17. Juni 136
10623 Berlin (Germany)

E-mail: benner@math.tu-berlin.de
URL: http://www.math.tu-berlin.de/~benner

Abstract

This part covers some basic computational problems underlying many control problems like system analysis (e.g., computing controllability/observability normal forms) or solving linear and quadratic matrix equations, (e.g., Lyapunov, Sylvester, and Riccati equations arising in system stabilization, observer design, or optimization of linear control systems).

Overview

- Linear systems
- System analysis
  - canonical forms
  - minimal realization
- System synthesis
  - linear-quadratic regulator
  - H_2-/H_∞ optimal control ($\rightsquigarrow$ Robust control design using H_∞ methods)
- Matrix equations
  - Sylvester and Lyapunov equations
  - algebraic Riccati equations
- Benchmark collections
- References

Linear Systems

Consider continuous or discrete linear time-invariant (LTI) systems.

State-space representations

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \quad t > 0 \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

or

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k, \quad k = 0, 1, 2, \ldots \\ y_k &= Cx_k + Du_k \end{aligned}$$

Assume

- n state variables, i.e., $x(t) \in \mathbb{R}^n$, $n =$ order of the system;
- m inputs, i.e., $u(t) \in \mathbb{R}^m$, and p outputs, i.e., $y(t) \in \mathbb{R}^p$;
- $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$.

Transfer function representations

$$G(s) = C(sI - A)^{-1}B + D$$

or

$$G(z) = C(zI - A)^{-1}B + D$$

System Analysis

Check controllability/observability/stabilizability/detectability numerically.

E.g., (A, B) (completely) controllable $\Longleftrightarrow$

1. For all $x_0, x_1 \in \mathbb{R}^n$ there exists admissible $\tilde{u}$ and $t_1 > 0$ such that $\tilde{x}(t_1) = x_1$ where $\tilde{x}$ solves
$$\dot{x} = Ax + B\tilde{u}, \quad x(0) = x_0.$$
Not feasible.

2. $\operatorname{rank}\left(\left[\begin{array}{cc} A - \lambda I, & B \end{array}\right]\right) = n$ for all $\lambda \in \mathbb{C}$.

   Feasible, but $\mathcal{O}(n^4)$ in general.

3. $\operatorname{rank}(\mathcal{C}(A, B)) = n$ where $\mathcal{C}(A, B) = \left[\begin{array}{ccccc} B & AB & A^2B & \dots & A^{n-1}B \end{array}\right]$.

   Feasible, but computing $\mathcal{C}(A, B)$ and checking rank is numerically unstable.

Want numerically stable procedure with $\mathcal{O}(n^3)$ complexity.

Staircase Form

There exist $U, V \in \mathbb{R}^{n\times n}$ orthogonal such that

$$\hat{A} \;:=\; U^T A U = \left[\begin{array}{cccccc|c} A_{11} & \dots & \dots & & & A_{1,s-1} & A_{1,s} \\ A_{21} & A_{22} & & & & A_{2,s-1} & A_{2,s} \\ 0 & \ddots & \ddots & & & \vdots & \vdots \\ \vdots & & \ddots & \ddots & & \vdots & \vdots \\ \vdots & \ddots & & & A_{s-1,s-2} & A_{s-1,s-1} & A_{s-1,s} \\ \hline 0 & \dots & \dots & & & 0 & A_{s,s} \end{array}\right],$$

$$\hat{B} \;:=\; U^T B V = \begin{bmatrix} B_1 & B_2 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix},$$

where $A_{i,i-1} = \left[\begin{array}{cc} \Sigma_{i,i-1} & 0 \end{array}\right] \in \mathbb{R}^{n_i \times n_{i-1}}$, $A_{s,s} \in \mathbb{R}^{n_s \times n_s}$, $B_1 \in \mathbb{R}^{n_1 \times n_1}$, and

$$n_1 \geq n_2 \geq \ldots \geq n_{s-1} \geq n_s, \qquad n_{s-1} > 0.$$

Computation via numerically stable procedure based on sequence of singular value decompositions or rank-revealing QR decompositions.

Staircase form of (A, B):

$$\left[\begin{array}{ccccc|c} A_{11} & \dots & \dots & & A_{1,s-1} & A_{1,s} \\ A_{21} & A_{22} & & & A_{2,s-1} & A_{2,s} \\ 0 & \ddots & \ddots & & \vdots & \vdots \\ \vdots & & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & & A_{s-1,s-2} & A_{s-1,s-1} & A_{s-1,s} \\ \hline 0 & \dots & \dots & & 0 & A_{s,s} \end{array}\right], \quad A_{s,s} \in \mathbb{R}^{n_s \times n_s}, \quad \left[\begin{array}{cc} B_1 & B_2 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{array}\right].$$

- LTI system controllable $\iff n_s = 0$ in staircase form of (A, B).
  (controllable subsystem: delete states $n_c + 1, \dots, n$ in staircase form of (A, B) where $n_c := n - n_s$.)
- LTI system observable $\iff n_s = 0$ in staircase form of (A^T, C^T).
  (observable subsystem: delete states $n_o + 1, \dots, n$ in staircase form of (A^T, C^T) where $n_o := n - n_s$.)
- LTI system stabilizable $\iff \lambda(A_{s,s}) \subset \mathbb{C}^-$ in staircase form of (A, B).
- LTI system detectable $\iff \lambda(A_{s,s}) \subset \mathbb{C}^-$ in staircase form of (A^T, C^T).

(For discrete-time systems, replace $\mathbb{C}^-$ by $\{z \in \mathbb{C}\,;\ |z| < 1\}$.)

Minimal Realization

Problem: find $r \leq n$ minimal (McMillan degree) and $A_r \in \mathbb{R}^{r\times r}$, $B_r \in \mathbb{R}^{r\times m}$, $C_r \in \mathbb{R}^{p\times r}$ such that

$$G(s) = C(sI - A)^{-1}B + D = C_r(sI - A_r)^{-1}B_r + D.$$

Then (A_r, B_r, C_r, D) is a minimal realization of the LTI system (A, B, C, D).

Computation via staircase algorithm.

Computation of minimal realization:

1. Apply staircase algorithm to (A, B) and update C:

$$\hat{A} := U_c^T A U_c, \quad \hat{B} := U_c^T B V_c, \quad \hat{C} := C U_c, \quad \hat{D} := D V_c \quad n_c := n - n_s(A, B).$$

   Extract controllable subsystem, i.e. delete rows $n_c + 1, \ldots, n$ of $\hat{A}$, $\hat{B}$, columns $n_c + 1, \ldots, n$ of $\hat{A}, \hat{C}$, and call the reduced controllable system with n_c states (A_c, B_c, C_c, D_c).

2. Apply staircase algorithm to (A_c^T, C_c^T) and update B_c:

$$\begin{aligned} \tilde{A} &:= U_o^T A_c U_o, \quad \tilde{B} := U_o^T B_c, \quad \tilde{C} := V_o^T C_c U_o, \quad \tilde{D} := V_o^T \hat{D} \\ n_o &:= n_c - n_s(A_c^T, C_c^T) = n - n_s(A, B) - n_s(A_c^T, C_c^T). \end{aligned}$$

   Extract observable subsystem, i.e. delete rows $n_o + 1, \ldots, n_c$ of $\tilde{A}$, $\tilde{B}$ and columns $n_o + 1, \ldots, n$ of $\tilde{A}, \tilde{C}$. The reduced controllable system (A_o, B_o, C_o, D_o) with n_o states is controllable and observable and hence minimal. Therefore, $r = n_o$ and $(A_r, B_r, C_r, D_r) = (A_o, B_o, C_o, D_o)$.

Software for System Analysis

SLICOT Fortran 77 Subroutines

AB01MD	extract a controllable subsystem from a single-input system
AB01ND	extract a controllable subsystem from a multi-input system
AB01OD	compute controllability staircase form for multi-input system
TB01PD	compute a minimal, controllable or observable block Hessenberg realization
TB01UD	compute a controllable realization
TB01ZD	compute a controllable realization for single-input systems

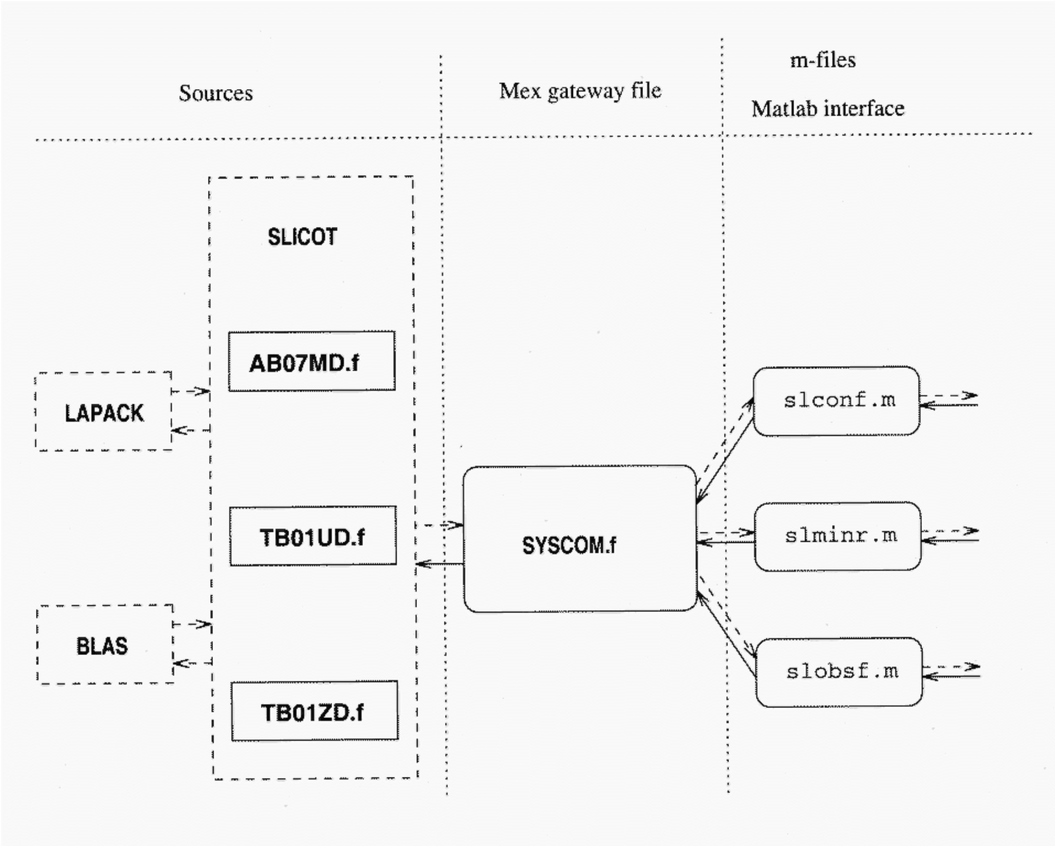
MATLAB Control Toolbox

`ctrbf`	computes the controllability staircase form of an LTI system
`obsvf`	computes the observability staircase form of an LTI system
`minreal`	computes a minimal realization of an LTI system

SLICOT-Based MATLAB Functions

`syscom`	mex file for computing controllability/observability staircase forms and minimal realization based on TB01PD, TB01UD, TB01ZD
`slconf`	computes the controllability staircase form of an LTI system
`slobsf`	computes the observability staircase form of an LTI system
`slminr`	computes a minimal realization of an LTI system

Structure of the SLICOT–Matlab Interfaces



MATLAB-interface for canonical forms and minimal realization

Taken from: V. Mehrmann, V. Sima, A. Varga, and H. Xu, *A* MATLAB *MEX-file environment of SLICOT, SLICOT Working Note 1999-11, August 1999.*
Available from `http://www.win.tue.nl/niconet/NIC2/reports.html` or `ftp://wgs.esat.kuleuven.ac.be/pub/WGS/REPORTS/SLWN1999-11.ps.Z`.

Performance Comparison

- Compute controllability staircase form using MATLAB Control Toolbox function `ctrbf` and SLICOT-based function `slconf` (calling SLICOT Fortran 77 subroutines TB01ZD/TB01UD via mex file `syscom`).
- Use randomly generated matrices A and B for single-input $(m = 1)$ and multi-input $(m > 1)$ systems.
- Accuracy measured by $\|UAU^T - \hat{A}\|$ is comparable.
- Timings (CPU times in sec.) within MATLAB (IEEE double precision arithmetic) on SUN UltraSPARC-IIi/440 MHz workstation.

n	SLICOT	MATLAB
16	0.01	0.02
32	0.01	0.07
64	0.02	0.31
128	0.07	2.58
256	0.51	37.86
512	4.35	563.49

n	m	SLICOT	MATLAB
16	2	0.00	0.03
32	4	0.01	0.01
64	8	0.03	0.05
128	16	0.08	0.28
256	32	0.68	1.83
512	64	5.01	13.63

System Synthesis

The linear-quadratic regulator problem

Minimize

$$\mathcal{J}_c(x_0, u) = \frac{1}{2}\int_0^\infty \left(y^T \tilde{Q} y + 2y^T L u + u^T R u\right) dt$$

where

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \quad x(0) = x_0, \\ y(t) &= Cx(t). \end{aligned}$$

continuous-time LQR problem

discrete-time LQR problem

Minimize

$$\mathcal{J}_d(x^0, u) = \frac{1}{2}\sum_{k=0}^{\infty} \left(y_k^T \tilde{Q} y_k + 2y_k^T L u_k + u_k^T R u_k\right)$$

where

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k, \quad k = 0, 1, \ldots, \\ y_k &= Cx_k, \qquad x_0 = x^0. \end{aligned}$$

Numerical Solution of the LQR Problem

Optimal solution is feedback control $u_*(t) = -F_* x(t)$ where the optimal gain matrix F_* is determined via the solution of an algebraic Riccati equation (ARE).

$$F_* := R^{-1}(B^T X_* + L^T)$$

where X_* is the unique stabilizing solution of the continuous-time algebraic Riccati equation (CARE)

$$0 = C^T \tilde{Q} C + A^T X + XA - (XB + L)R^{-1}(B^T X + L^T).$$

$$F_* := (R + B^T X_* B)^{-1}(B^T X_* A + L^T)$$

where X_* is the unique stabilizing solution of the discrete-time algebraic Riccati equation (DARE)

$$X = C^T \tilde{Q} C + A^T X A - (A^T X B + L)(R + B^T X B)^{-1}(B^T X A + L^T).$$

Remarks:

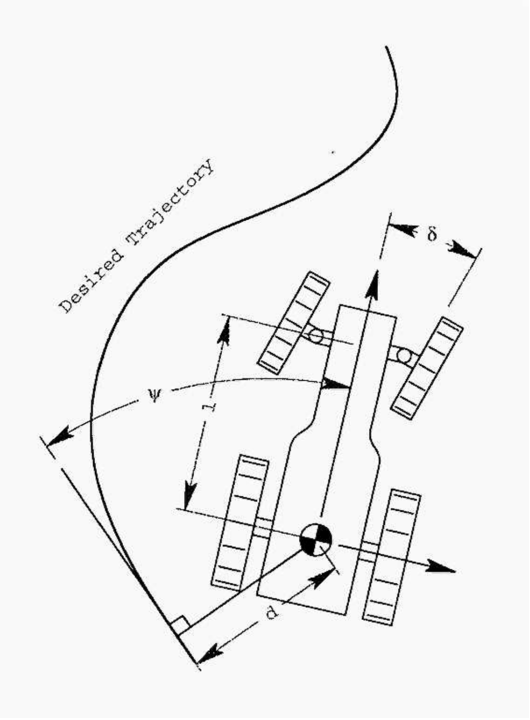
- LQG design consists of LQ regulator plus LQ estimator (Kalman filter); LQ estimator is dual to LQR problem and is solved via the same AREs with different coefficient matrices.
- Optimal gain matrices are returned by MATLAB functions for solving AREs and can be computed by SLICOT subroutine SB02ND.

Design Example

- Control design for GPS-based automatic steering of farm tractor.
  (GPS Lab of Stanford University/University of Bremen)
- SLICOT routine SB02MD (discrete-time algebraic Riccati equation solver) used in LQG control design.
- Resolved computational bottleneck: solve 5 DAREs/sec. — SB02MD needs 0.01 sec. for one DARE!
- This allows for adaptive control strategies.
- See *SLICOT Drives Tractors!*, (P. Benner, H. Faßbender) NICONET Newsletter 2, January 1999, pp. 17–22 and NICONET Report 1999-2.

Figure 1: GPS–equipped tractor.

State-space variables:



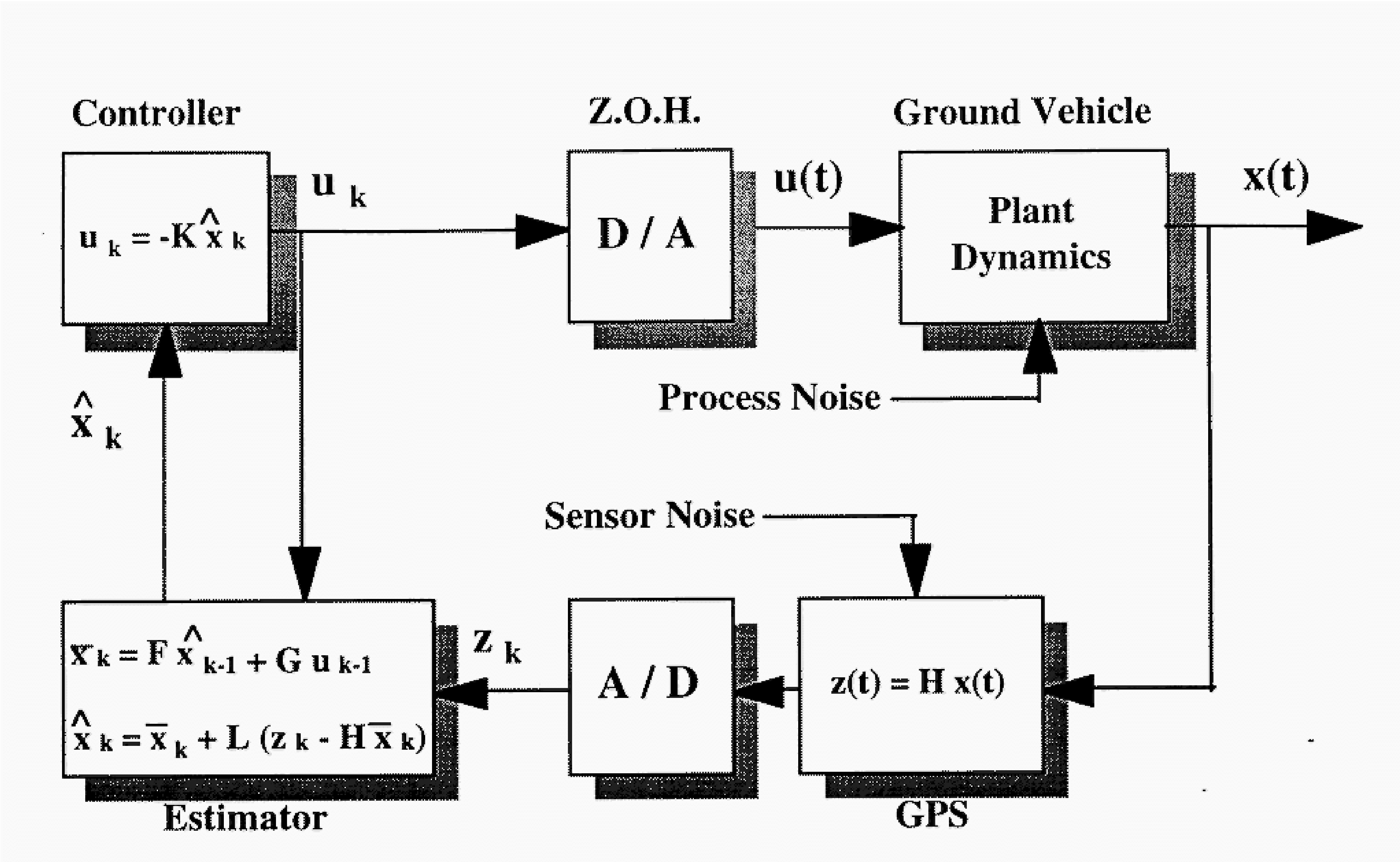
$\Longrightarrow$ state-space model

$$\begin{aligned} \dot{x} &= Ax + Bu, \\ y &= Cx, \end{aligned}$$

with states $x = [\,\psi,\ \dot{\psi},\ \delta,\ \dot{\delta},\ d\,]^T$ and parameters
– V, the forward velocity of the tractor,
– τ_ψ, τ_u identified from experimental data.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & -\frac{1}{\tau_\psi} & \frac{V}{l\tau_\psi} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau_u} & 0 \\ V & 0 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{\tau_u} \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Block diagram of LQG regulator:



Sylvester Equations

continuous-time	or	discrete-time
$AX + XB + C = 0$		$AXB - X + C = 0$

$$A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{m\times m},\ C \in \mathbb{R}^{n\times m} \Longrightarrow \quad X \in \mathbb{R}^{n\times m}$$

Sylvester equation is equivalent to system of linear equations in $\mathbb{R}^{nm}$:
(here consider continuous-time case):

$$\big((I_m \otimes A) + (B^T \otimes I_n)\big)\,\mathrm{vec}(X) = -\mathrm{vec}(C)$$

Kronecker product of $F \in \mathbb{R}^{n\times n}, G \in \mathbb{R}^{m\times m}$: $F \otimes G := \begin{bmatrix} f_{11}G & \dots & f_{1n}G \\ \vdots & \ddots & \vdots \\ f_{n1}G & \dots & f_{nn}G \end{bmatrix}$

Cost for solution via Gaussian elimination/LU factorization for $n = m$ is $O(n^6)$
$\Longrightarrow$ too expensive!

Numerical methods for solving Sylvester equations with cost of $O(n^3)$:

- Bartels-Stewart method (BS) — [*Bartels/Stewart '72*],
- Hessenberg-Schur method (HS) — [*Golub/Nash/Van Loan '79*].

Algorithm:

1. **BS:** Apply QR-algorithm to A and compute Schur decomposition $\tilde{A} = U^T AU$, where $U \in \mathbb{R}^{n\times n}$ is orthogonal and $\tilde{A}$ is (quasi-)upper triangular.
   HS: Compute Hessenberg decomposition of A, i.e., compute orthogonal matrix $U \in \mathbb{R}^{n\times n}$ such that $\tilde{A} = U^T AU$ is in upper Hessenberg form.
2. Apply QR-algorithm to B^T and compute Schur decomposition $\tilde{B} = V^T B^T V$ where $V \in \mathbb{R}^{m\times m}$ is orthogonal and $\tilde{B}$ is (quasi-)upper triangular.
3. $\tilde{C} \leftarrow U^T CV$.
4. Solve reduced equation $\tilde{A}\tilde{X} + \tilde{X}\tilde{B}^T + \tilde{C} = 0$ by back substitution process.
   (In Kronecker product form, this is a system of linear equations with coefficient matrix in (quasi-)upper triangular or Hessenberg form.)
5. $X \leftarrow U\tilde{X}V^T$

Properties of Numerical Algorithms for Solving Sylvester Equations

- Both methods are numerically backward stable if the back substitution process is implemented carefully (see [*Sima '96*] for details).
- The Hessenberg-Schur method is more efficient than the Bartels-Stewart method (estimated 30–70% depending on ratio n/m).
- Discrete Sylvester equations are solved in analogous way.

Lyapunov Equations

Lyapunov equation = symmetric Sylvester equation

continuous-time	or	discrete-time (Stein equations)
$AX + XA^T + C \quad = \quad 0,$		$AXA^T - X + C \quad = \quad 0$

$$A \in \mathbb{R}^{n \times n}, \quad C = C^T \in \mathbb{R}^{n \times n}.$$

Numerical solution: Bartels-Stewart method

1. Apply QR-algorithm to A and compute Schur decomposition $\tilde{A} = U^T AU$ where $U \in \mathbb{R}^{n \times n}$ is orthogonal and $\tilde{A}$ is (quasi-)upper triangular.
2. $\tilde{C} \leftarrow U^T CU$ (symmetric update).
3. Solve reduced equation $\tilde{A}\tilde{X} + \tilde{X}\tilde{A}^T + \tilde{C} = 0$ or $\tilde{A}\tilde{X}\tilde{A}^T - \tilde{X} + \tilde{C} = 0$ by back substitution process.
4. $X \leftarrow U\tilde{X}U^T$ (symmetric update).

Note: special subroutines available for

- computing Cholesky factor Y of stable Lyapunov equations

$$AX + XA^T + C^TC = 0 \qquad\qquad AXA^T - X + C^TC = 0$$

$A \in \mathbb{R}^{n\times n}$ stable, $X = YY^T$, directly (Hammarling's method);

- solving generalized Sylvester equations

$$\begin{array}{rcl} AX - YB & = & C, \\ DX - YE & = & F, \end{array} \qquad A, D \in \mathbb{R}^{n\times n}, B, E \in \mathbb{R}^{m\times m}, C, F \in \mathbb{R}^{n\times m},$$

- solving generalized (discrete, stable) Lyapunov equations

$$A^TXE + E^TXA + C = 0 \qquad\qquad A^TXA - E^TXE + C = 0$$

$$C = C^T \in \mathbb{R}^{n\times n}$$

- computing forward error and condition estimates for all these equations.

Software for Solving Linear Matrix Equations (LMEs)

SLICOT Fortran 77 Subroutines for Sylvester Equations

SB04MD	solve Sylvester equations by Hessenberg-Schur method
SB04ND	solve Sylvester equations if one coefficient is in Schur form
SB04OD	solve generalized Sylvester equations and estimate condition
SB04PD	solve Sylvester equations by Schur method
SB04QD	solve Sylvester equations by Hessenberg-Schur method
SB04RD	solve Sylvester equations if one coefficient is in Schur form

SLICOT Fortran 77 Subroutines for Lyapunov and Stein Equations

SB03MD	solve Lyapunov equations and estimate condition
SB03OD	solve stable Lyapunov equations for Cholesky factor
SB03PD	solve Stein equations and estimate condition
SB03RD	solve Lyapunov equations and estimate condition
SB03TD	solve Lyapunov equations, estimate condition and forward error
SB03UD	solve Stein equations, estimate condition and forward error
SG03AD	solve generalized Lyapunov equations and estimate condition
SG03BD	solve stable generalized Lyapunov equations for Cholesky factor

Matlab Functions for Solving LMEs

Matlab Toolboxes

Control Toolbox	`lyap`	solve Sylvester and Lyapunov equations by Bartels-Stewart method
	`dlyap`	solve Stein equations by Bartels-Stewart method
μ-Analysis and Synthesis Toolbox	`clyap`	solve stable Lyapunov equations for Cholesky factor by Hammarling's method
	`sylv`	solve Sylvester equation using Kronecker product form

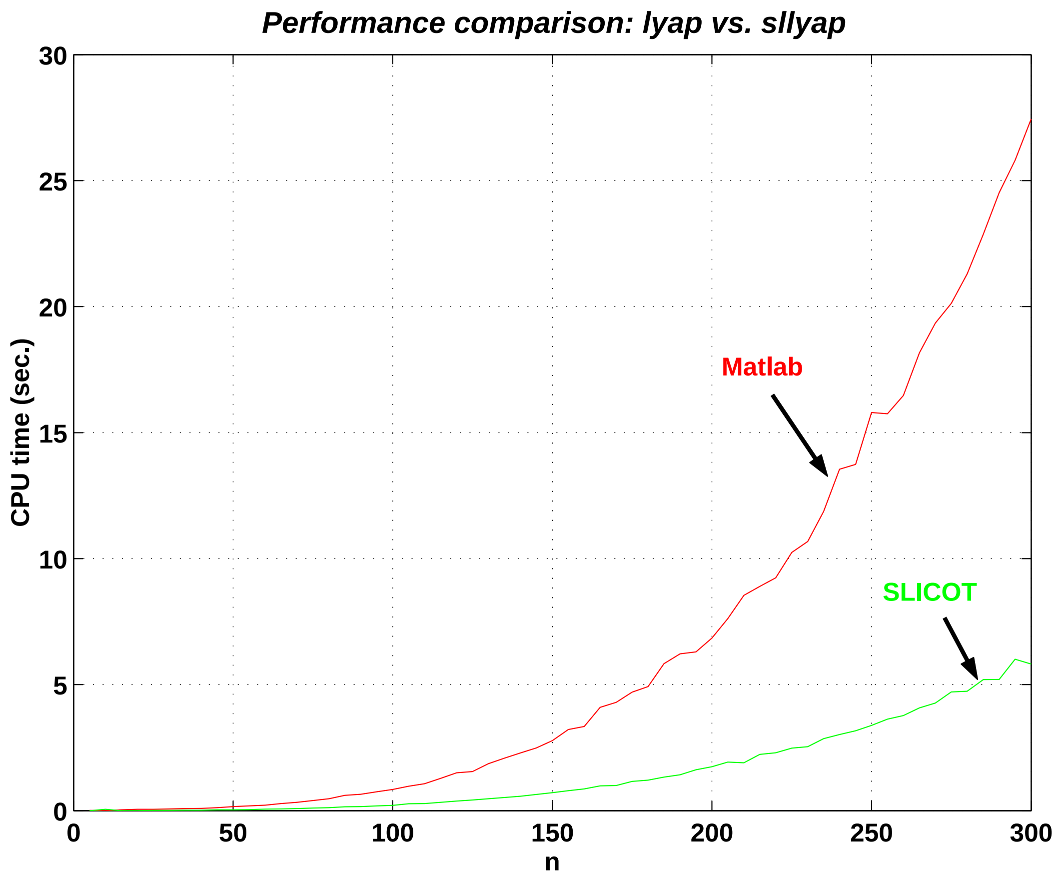
SLICOT-Based MATLAB Functions

`linmeq`	mex file for solving LMEs based on SB03MD, SB03OD, SB04MD, SB04ND, SB04PD, SB04QD, SB04RD
`genleq`	mex file for solving generalized LMEs based on SB04OD, SG03AD, SG03BD
`sllyap`	solve Lyapunov equations
`slstei`	solve Stein equations
`slstly`	solve stable Lyapunov equations for Cholesky factor of solution
`slstst`	solve stable Stein equations for Cholesky factor of solution
`slsylv`	solve continuous-time Sylvester equations
`sldsyl`	solve discrete-time Sylvester equations
`slgely`	solve generalized Lyapunov equations
`slgest`	solve generalized Stein equations
`slgsly`	solve stable generalized Lyapunov equations for Cholesky factor of solution
`slgsst`	solve stable generalized Stein equations for Cholesky factor of solution
`slgesg`	solve generalized Sylvester equations

Performance

- Compare MATLAB Control Toolbox function `lyap` and SLICOT-based function `sllyap` (calling SLICOT Fortran 77 subroutine SB03MD via mex file `linmeq`).
- Accuracy is comparable.
- Timings for randomly generated examples:

```
n = 5:5:300
A = rand(n);
X = rand(n);   X = X + X';
C = -(A'*X + X*A);
```



Algebraic Riccati Equations

continuous-time (CARE)

$$0 = Q + A^T X + XA - XGX$$

or

discrete-time (DARE)

$$X = Q + A^T XA - A^T XB(R + B^T XB)^{-1} B^T XA$$

In control theory, need stabilizing solution $X_* = X_*^T \in \mathbb{R}^{n \times n}$, i.e., the unique solution that makes the closed-loop matrix $A - GX_*$ or $A - B(R + B^T X_* B)^{-1} B^T X_* A$ stable. (Here assume X_* exists.)

$$\lambda(A - GX_*) \subset \mathbb{C}^-$$

or

$$\lambda(A - B(R + B^T X_* B)^{-1} B^T X_* A) \subset \{z \in \mathbb{C}\ ;\ |z| < 1\}$$

Numerical Solution of CAREs

- Consider CARE as system of nonlinear equations $\rightsquigarrow$ Newton's method.
- Use connection to Hamiltonian eigenproblem.

Definition: $H \in \mathbb{R}^{2n \times 2n}$ *Hamiltonian* $\Leftrightarrow HJ = (HJ)^T$ *with* $J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}$.

X stabilizing solution of the CARE	$\Longleftrightarrow$	$H \begin{bmatrix} I_n \\ X \end{bmatrix} = \begin{bmatrix} A & -G \\ -Q & -A^T \end{bmatrix} \begin{bmatrix} I_n \\ X \end{bmatrix} = \begin{bmatrix} I_n \\ X \end{bmatrix} (A - GX),$ $\lambda(A - GX) = \lambda(H) \cap \mathbb{C}^-$

I.e., columns of $\begin{bmatrix} I_n \\ X \end{bmatrix}$ span stable invariant subspace of Hamiltonian matrix H.

Note: here, $\lambda(H) = \{\pm\lambda_j \,|\mathrm{Re}(\lambda_j) < 0\}$.

Methods:

Compute stable H-inv. subspace via (structured, block-) Schur decomposition,

$$T^{-1}HT = \begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix}, \quad \lambda(H_{11}) = \lambda(H) \cap \mathbb{C}^-, \quad T = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix}$$

$$\Longrightarrow \quad X = T_{21}T_{11}^{-1}$$

- QR algorithm (Schur vector method) [*Laub '79*];
- SR algorithm [*Bunse-Gerstner/Mehrmann '86*];
- multishift algorithm [*Ammar/Benner/Mehrmann '93*];
- Jacobi-type algoritms [*Byers '90, Bunse-Gerstner/Faßbender '97*];
- embedding algorithm [*Benner/Mehrmann/Xu '97*];

or spectral projection methods,

- sign function method [*Roberts '71, Byers '87, Gardiner/Laub '86*];
- disk function method [*Malyshev '93, Bai/Demmel/Gu '95, Benner/Byers '95,'97*].

Analogous methods for DAREs: use Newton's method or connection to (generalized) symplectic eigenproblem.

Newton's Method for CAREs

[*Kleinman '68, Mehrmann '91, Lancaster/Rodman '95, Benner/Byers '94/'98, Benner '97*]

1. Find $X_0 = X_0^T$ such that $\lambda\,(A - GX_0) \subset \mathbb{C}^-$.

2. FOR $j = 0,\,1,\,2,\,\ldots$

   2.1 $A_j \leftarrow A - GX_j$.

   2.2 Solve the Lyapunov equation

   $$A_j^T N_j + N_j A_j = -\mathcal{R}(X_j) = -(Q + A^T X_j + X_j A - X_j G X_j).$$

   2.3 $X_{j+1} \leftarrow X_j + t_j N_j$.

   END FOR j

Properties of Newton's Method

Advantages:

- convergence is monotone and quadratic after first step;
- cheap stepsize control (exact line search) is available;
- computes solution to highest possible accuracy.

Disadvantages:

- good starting value needed (often used for iterative refinement only!);
- problems occur when the solution has large norm or eigenvalues of $A - GX_*$ are close to imaginary axis.

Not yet included in SLICOT.

Sign Function Method for CAREs

[*Roberts '71, Byers '87***]**

Assume that $H_0 := H = \begin{bmatrix} A & -G \\ -Q & -A^T \end{bmatrix}$ has no purely imaginary eigenvalues.

1. FOR $j = 0, 1, 2, \ldots$

   1.1 $H_j \leftarrow \gamma H_j$.

   1.2 $H_{j+1} \leftarrow H_j - \frac{1}{2}(H_j - (JH_j)^{-1})J)$. $\quad (J = \left[\begin{smallmatrix} 0 & I_n \\ -I_n & 0 \end{smallmatrix}\right])$

   END FOR j

2. Solve consistent least-squares problem

$$(H_\infty - I_{2n}) \begin{bmatrix} I \\ X_* \end{bmatrix} = 0 \iff \begin{bmatrix} H_{\infty,12} \\ H_{\infty,22} - I_n \end{bmatrix} X_* = \begin{bmatrix} H_{\infty,11} - I_n \\ H_{\infty,21} \end{bmatrix}$$

   using a QR-factorization with column pivoting.

Properties of the Sign Function Method

- $H_\infty = \lim_{j\to\infty} H_j = \operatorname{sign}(H)$.
  Here: $\operatorname{sign}(H) := T\operatorname{diag}\{\operatorname{sign}(\operatorname{Re}(\lambda_k))\}T^{-1}$ if $H = T(\operatorname{diag}\{\lambda_k\} + N)T^{-1}$ is the Jordan normal form of H.

- γ is scaling parameter for accelerating convergence, e.g.,
  $\gamma = \det(H_j)^{-\frac{1}{2n}}$ [*Byers '87*] or $\gamma = \sqrt{\|H_j^{-1}\|/\|H_j\|}$ [*Higham '86*].

- Advantages:
  - quadratic convergence;
  - structure-preserving, i.e., all H_j are Hamiltonian;
  - easily parallelizable;
  - applicable for medium large problems.

- Disadvantages:
  - numerical problems if H_j highly ill-conditioned;
  - method does not work if eigenvalues are near or on imaginary axis.

The Schur Vector Method for CAREs

[*Laub '79*]

Use QR-algorithm (from LAPACK) to compute stable H-invariant subspace,

$$\begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} W, \quad \lambda\,(W) \subset \mathbb{C}^-.$$

1. Apply QR-algorithm to H and compute Schur decomposition $\tilde{H} = \tilde{U}^T H \tilde{U}$ where $\tilde{U}$ is orthogonal and $\tilde{H}$ is (quasi-)upper triangular.

2. Re-order eigenvalues, i.e., compute orthogonal $\hat{U}$ such that

$$\hat{H} = \hat{U}^T \tilde{H} \hat{U} = \begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix}, \quad \lambda\,(H_{11}) = \lambda\,(H) \cap \mathbb{C}^-.$$

3. Partition $\tilde{U}\hat{U} = \begin{bmatrix} U_1 & U_3 \\ U_2 & U_4 \end{bmatrix}, U_j \in \mathbb{R}^{n \times n}$, and solve linear system $X_* U_1 = -U_2$.

Note: columns of $\begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$ are called Schur vectors of H.

Properties of the Schur Vector Method

Advantages:

- easy to implement using LAPACK kernels only;
- QR-algorithm and re-ordering are numerically backward stable;
- method can be used for medium large problems;
- error and condition estimation available.

Disadvantages:

- problems if eigenvalues are near or on imaginary axis
  (structure is destroyed, numerically computed eigenvalues may be on wrong side of imaginary axis leading to unequal numbers of stable and unstable eigenvalues);
- numerical problems U_1 is ill-conditioned (usually the case if X_* has large norm).

Note: in DARE case need nonsingular A to use Schur vector method; otherwise form *symplectic pencil* $\begin{bmatrix} A & 0 \\ Q & I_n \end{bmatrix} - \lambda \begin{bmatrix} I_n & -BR^{-1}B^T \\ 0 & A^T \end{bmatrix}$ and use QZ algorithm!

The Generalized Schur Vector Method for CAREs

[*Pappas/Laub/Sandell '80, Van Dooren 81***]**

Often, ARE coefficients $\mathcal{A}, \mathcal{G}, \mathcal{Q}$ come from LQR problem:

$$\mathcal{A} = A - BR^{-1}L^T, \quad \mathcal{G} = BR^{-1}B^T, \quad \mathcal{Q} = Q - LR^{-1}L^T.$$

(Coefficients can be formed using SLICOT subroutine SB02MT.)

Numerical problems can be expected if R is ill-conditioned!

Better: use n-dimensional stable deflating subspace $\mathcal{U}$ of extended matrix pencil

$$H - \lambda K = \begin{bmatrix} A & 0 & B \\ Q & A^T & L \\ L^T & B^T & R \end{bmatrix} - \lambda \begin{bmatrix} I_n & 0 & 0 \\ 0 & -I_n & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \mathcal{U} = \mathrm{colspan} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix}.$$

Solution of CARE is then $X_* = U_2 U_1^{-1}$, optimal feedback is $F_* = U_3 U_1^{-1}$.

Implementation: apply QZ-algorithm (LAPACK) with re-ordering to compute generalized Schur form of $H - \lambda K$.

Software for Solving AREs

MATLAB Toolboxes

Control Toolbox	`care`	(generalized) Schur vector method (CAREs)
	`dare`	(generalized) Schur vector method (DAREs)
Robust Control Toolbox	`aresolv`	eigenvector or Schur vector method (CAREs)
	`daresolv`	eigenvector or Schur vector method (DAREs)
μ-Analysis and Synthesis Toolbox	`ric_eig`	eigenvector method (CAREs)
	`ric_schr`	Schur vector method (CAREs)

SLICOT Fortran 77 Subroutines

SB02MD	Schur vector method (invariant subspace method) (CAREs/DAREs)
SB02OD	generalized Schur vector method (deflating subspace method) (CAREs/DAREs)
SB02PD	matrix sign function method with condition and forward error estimates (CAREs)
SB02RD	refined invariant subspace method with scaling, condition and forward error estimates (CAREs/DAREs)

SLICOT-Based Matlab Software for AREs

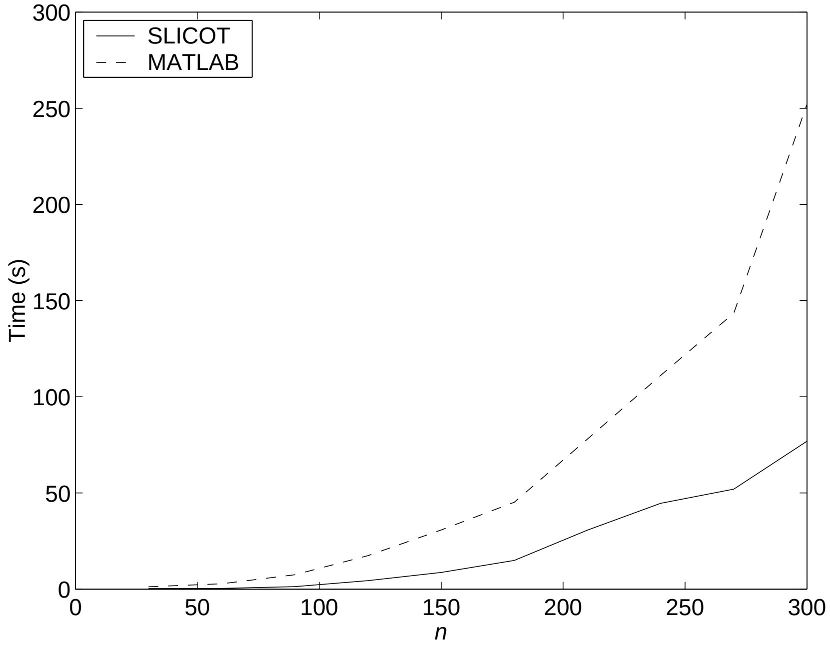
`aresol`	mex file for solving AREs based on SB02MD, SB02OD, SB02MT, SB02ND
`aresolc`	mex file for solving AREs based on SB02RD, SB02OD, SB02MT, SB02ND
`slcares`	solve CARE with Schur vector method
`sldares`	solve DARE with Schur vector method
`sldaregsv`	solve DARE with generalized Schur vector method applied to symplectic pencil
`slcaregs`	solve CARE with generalized Schur vector method applied to extended matrix pencil
`sldaregs`	solve DARE with generalized Schur vector method applied to extended matrix pencil
`slcaresc`	solve CARE with refined Schur vector method and condition estimation
`sldaresc`	solve DARE with refined Schur vector method and condition estimation

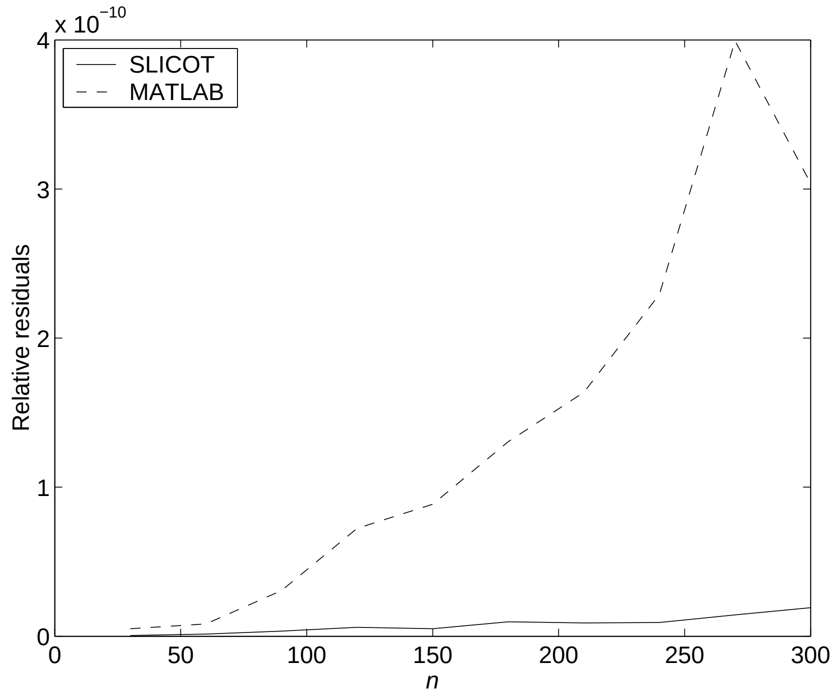
Numerical Experiments

- Compare performance of SLICOT-based MATLAB functions with MATLAB toolbox functions:
  - `care`, `dare` from Control and LMI Toolboxes,
  - `aresolv`, `daresolv` from the Robust Control Toolbox,
  - `ric_eig`, `ric_schr` from the μ-Analysis and Synthesis Toolbox (only CARE solvers).
- Chosen test cases:
  - Random examples with $n = 30 : 30 : 300$ and $m = n/5$.
  - CARE benchmark collection ([*Abels/B. '99*], see SLICOT routine BB01AD): 20 CAREs, partially parameterized $\rightsquigarrow$ 34 data sets,
  - DARE benchmark collection ([*Abels/B. '99*], see SLICOT routine BB02AD): 19 DAREs, partially parameterized $\rightsquigarrow$ 25 data sets.
- Results obtained on PC Pentium 3 (500 MHz) and MATLAB 6.1.

Accuracy and Efficieny for Random CARE Examples

- `slcares` versus MATLAB `care`,
- random CAREs, $n = 30 : 30 : 300$, $m = n/5$.

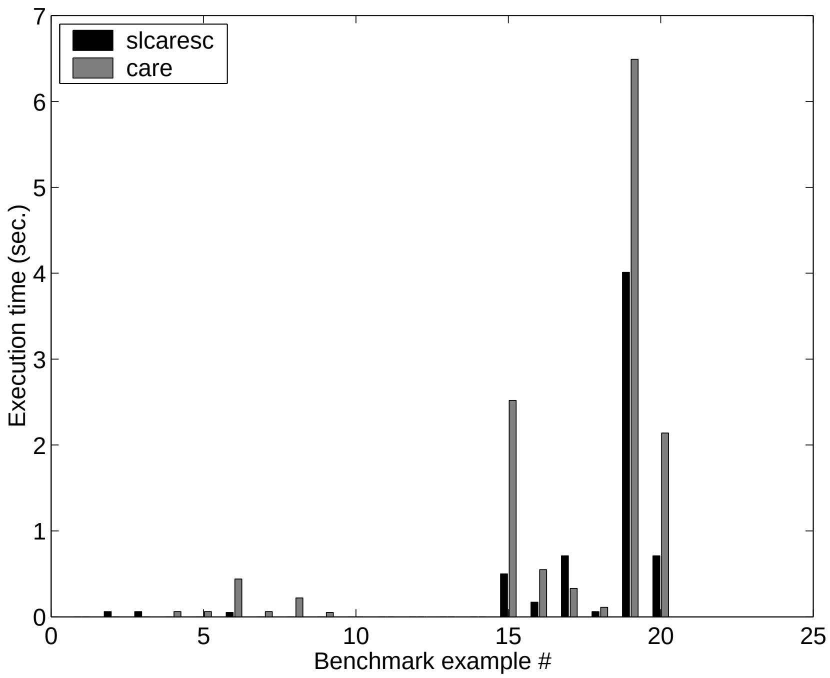


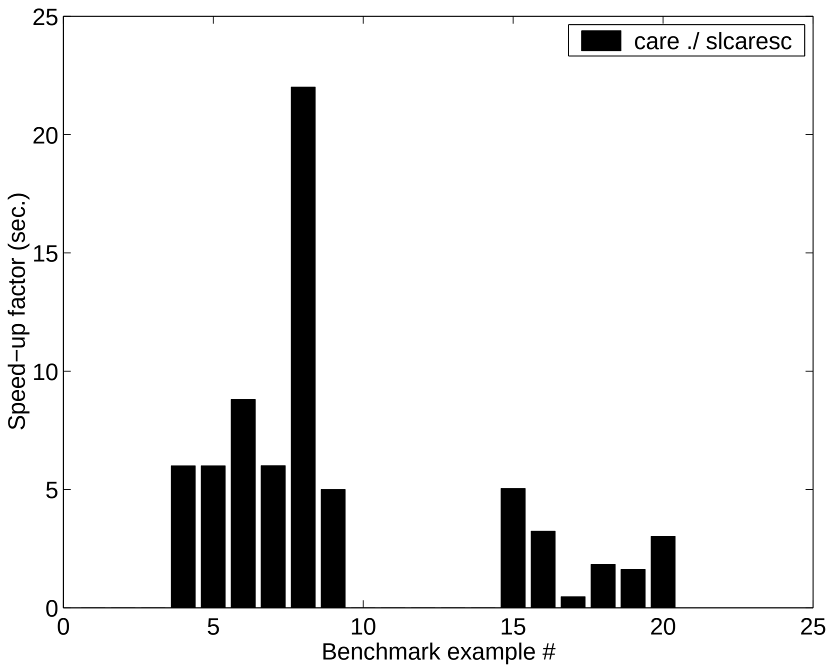


CPU times (left) and relative residuals (right)

Efficiency for CARE Benchmark Examples

- `slcaresc` versus MATLAB `care`,
- CAREs from CARE benchmark collection.

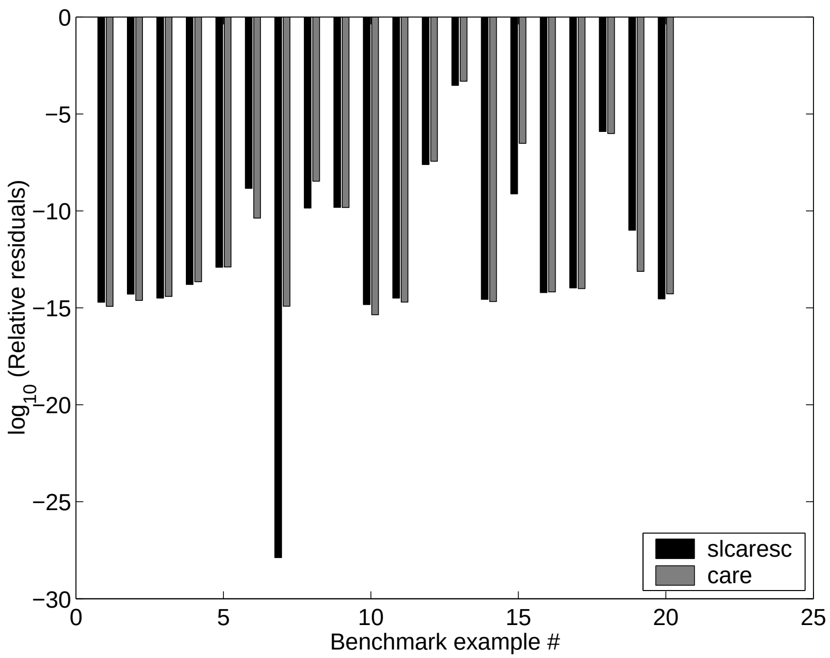


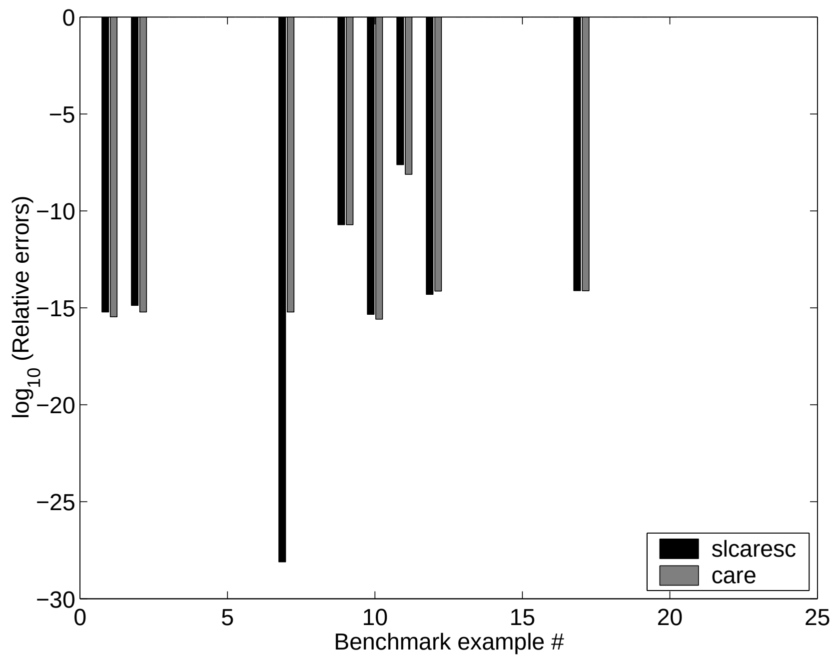


Timing (left) and speed-up factors (right)

Accuracy for CARE Benchmark Examples

- `slcaresc` versus MATLAB `care`,
- CAREs from CARE benchmark collection.





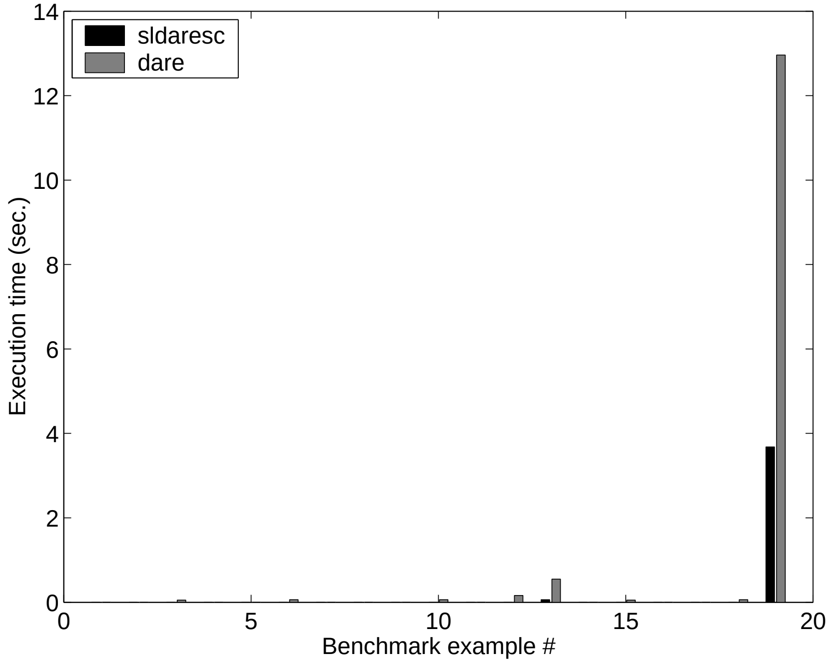
Relative residuals (left) and relative errors (right)

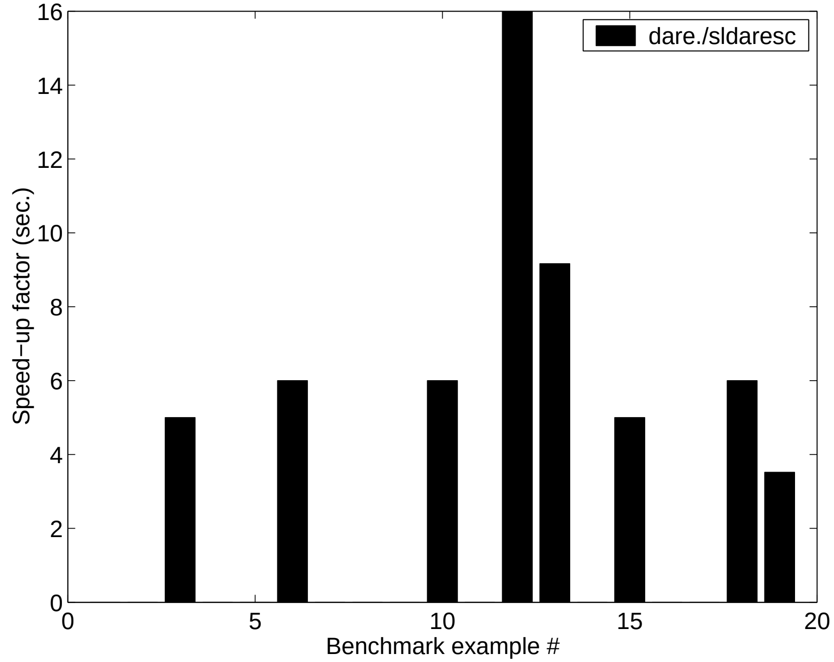
Cumulative Performance of CARE solvers for Benchmark Collection

Perform.	slcaresc	care	ric_eig	ric_schr	aresolv 'eigen'	aresolv 'Schur'
Time (sec.)	6.71	13.84	2.9	15.75	4.07	15.21
Rel. residuals	3.0e-4	4.9e-4	1.3e+3	1.3e+3	1.3e+3	3.2e+3
Rel. errors	8.9e-5	4.4e-5	2.3e-4	2.5e-4	2.3e-4	5.7e-4

Efficiency for DARE Benchmark Examples

- `sldaresc/sldaregs` versus MATLAB `dare`,
- DAREs from DARE benchmark collection.

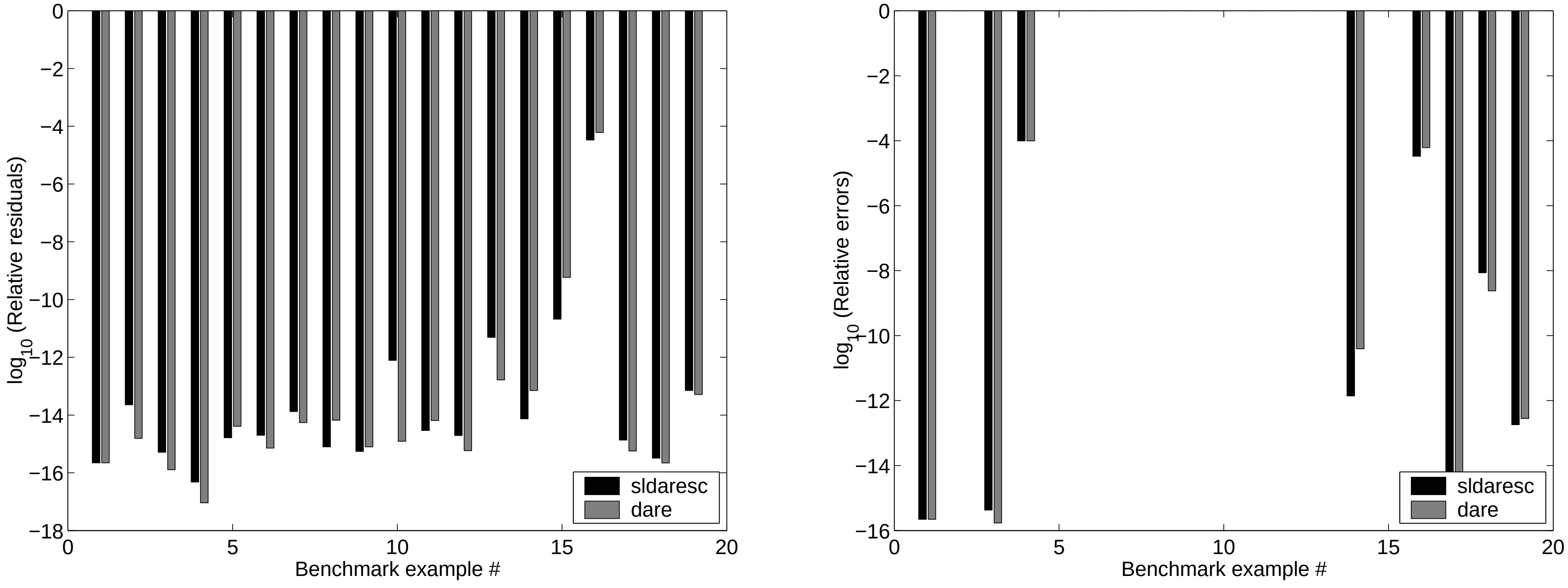




Timing (left) and speed-up factors (right)

Accuracy for DARE Benchmark Examples

- `sldaresc`/`sldaregs` versus MATLAB `dare`,
- DAREs from DARE benchmark collection.



Relative residuals (left) and relative errors (right)

Cumulative Performance of CARE solvers for Benchmark Collection

Performance	sldaresc/ sldaregs	dare	daresolv 'eigen'	daresolv 'Schur'
Time (sec.)	3.74	14.12	12.64	13.38
Rel. residuals	3.3e-5	6.1e-5	6.1e-5	6.1e-5
Rel. errors	3.3e-5	6.1e-5	6.1e-5	6.1e-5

SLICOT Benchmark Collections

Collections of benchmark examples for the testing of numerical methods are available from SLICOT.

Fortran 77

BD01AD	benchmark examples of continuous LTI systems
BD02AD	benchmark examples of discrete LTI systems
BB01AD	CARE benchmark examples
BB02AD	DARE benchmark examples
BB03AD	benchmark examples of (generalized) Lyapunov equations
BB04AD	benchmark examples of (generalized) Stein equations

MATLAB

`ctdsx`	benchmark examples of continuous-time LTI systems
`dtdsx`	benchmark examples of discrete-time LTI systems
`ctlex`	benchmark examples of (generalized) Lyapunov equations
`dtlex`	benchmark examples of (generalized) Stein equations

Concluding Remarks

SLICOT contains many more subroutines for basic control problems, e.g., for computing

- system norms like Hankel norm, H_2-/L_2-norm, H_∞-/L_∞-norm with MATLAB interfaces etc.

  (In particular `slinorm`, is a lot more reliable than MATLAB toolbox functions for H_∞-norm computation!)

- stability radii,
- poles and zeros,
- inverse systems,
- many more!

SLICOT also contains many mathematical subroutines extending the functionality of standard linear algebra software like LAPACK.

Further Reading

[1] G. Ammar, P. Benner, and V. Mehrmann, *A multishift algorithm for the numerical solution of algebraic Riccati equations*, Electr. Trans. Num. Anal., 1 (1993), pp. 33–48.

[2] Z. Bai, J. Demmel, and M. Gu, *An inverse free parallel spectral divide and conquer algorithm for nonsymmetric eigenproblems*, Numer. Math., 76 (1997), pp. 279–308.

[3] R. Bartels and G. Stewart, *Solution of the matrix equation $AX + XB = C$: Algorithm 432*, Comm. ACM, 15 (1972), pp. 820–826.

[4] P. Benner, *Computational methods for linear-quadratic optimization*, Supplemento ai Rendiconti del Circolo Matematico di Palermo, Serie II, No. 58 (1999), pp. 21–56.[1].

[5] P. Benner, *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*, Logos–Verlag, Berlin, Germany, 1997. *Also:* Dissertation, Fakultät für Mathematik, TU Chemnitz–Zwickau, 1997.

[6] P. Benner and R. Byers, *Disk functions and their relationship to the matrix sign function*, in Proc. European Control Conf. ECC 97, Paper 936, BELWARE Information Technology, Waterloo, Belgium, 1997. CD-ROM.

[7] P. Benner and R. Byers, *An exact line search method for solving generalized continuous-time algebraic Riccati equations*, IEEE Trans. Automat. Control, 43 (1998), pp. 101–107.

[1] `http://www.math.uni-bremen.de/zetem/reports/reports-psgz/report9804.ps.gz`

[8] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga. SLICOT - a subroutine library in systems and control theory. In B.N. Datta, editor, *Applied and Computational Control, Signals, and Circuits*, vo. 1, ch. 10, pages 499–539. Birkhäuser, Boston, MA, 1999.

[9] P. Benner, V. Mehrmann, and H. Xu, *A new method for computing the stable invariant subspace of a real Hamiltonian matrix*, J. Comput. Appl. Math., 86 (1997), pp. 17–43.

[10] A. Bunse-Gerstner and H. Fassbender, *A Jacobi-like method for solving algebraic Riccati equations on parallel computers*, IEEE Trans. Automat. Control, 42 (1997), pp. 1071–1084.

[11] A. Bunse-Gerstner and V. Mehrmann, *A symplectic QR-like algorithm for the solution of the real algebraic Riccati equation*, IEEE Trans. Automat. Control, AC-31 (1986), pp. 1104–1113.

[12] R. Byers, *Solving the algebraic Riccati equation with the matrix sign function*, Linear Algebra Appl., 85 (1987), pp. 267–279.

[13] R. Byers, *A Hamiltonian-Jacobi algorithm*, IEEE Trans. Automat. Control, 35 (1990), pp. 566–570.

[14] J. Gardiner and A. Laub, *A generalization of the matrix-sign-function solution for algebraic Riccati equations*, Internat. J. Control, 44 (1986), pp. 823–832.

[15] G. H. Golub, S. Nash, and C. F. Van Loan, *A Hessenberg–Schur method for the problem* $AX + XB = C$, IEEE Trans. Automat. Control, AC-24 (1979), pp. 909–913.

[16] D. Kleinman, *On an iterative technique for Riccati equation computations*, IEEE Trans. Automat. Control, AC-13 (1968), pp. 114–115.

[17] P. Lancaster and L. Rodman, *The Algebraic Riccati Equation*, Oxford University Press, Oxford, 1995.

[18] A. Laub, *A Schur method for solving algebraic Riccati equations*, IEEE Trans. Automat. Control, AC-24 (1979), pp. 913–921.

[19] V. Mehrmann. *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution*. Number 163 in Lecture Notes in Control and Information Sciences. Springer-Verlag, Heidelberg, July 1991.

[20] V. Mehrmann, V. Sima, A. Varga, and H. Xu. A MATLAB MEX-file environment of SLICOT. SLICOT Working Note 1999-11, August 1999. [2]

[21] T. Pappas, A. Laub, and N. Sandell, *On the numerical solution of the discrete-time algebraic Riccati equation*, IEEE Trans. Automat. Control, AC-25 (1980), pp. 631–641.

[22] P.H. Petkov, N.D. Christov, and M.M. Konstantinov. *Computational Methods for Linear Control Systems*. Prentice-Hall, Hertfordshire, UK, 1991.

[23] J. Roberts, *Linear model reduction and solution of the algebraic Riccati equation by use of the sign function*, Internat. J. Control, 32 (1980), pp. 677–687. (Reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department, 1971).

[2] `http://www.win.tue.nl/niconet/NIC2/reports.html`

[24] V. Sima. *Algorithms for Linear-Quadratic Optimization*, volume 200 of *Pure and Applied Mathematics*. Marcel Dekker, Inc., New York, NY, 1996.

[25] P. Van Dooren, *A generalized eigenvalue approach for solving Riccati equations*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 121–135.

System Identification Using Subspace Methods

Vasile Sima
National Institute for Research & Development in Informatics
Bd. Mareşal Averescu, Nr. 8–10
011455 Bucharest 1 (Romania)

E-mail: vsima@iciadmin.ici.ro
URL: http://www.ici.ro/ici/organizare/directory/vsima.html

Abstract

System identification means finding mathematical models of dynamic systems from measured data. This is the first, and basic step for both system analysis and control system design. This lecture mainly addresses linear and Wiener-type discrete-time systems in the multivariable case. The algorithms discussed are based on subspace methods (MOESP and N4SID) for the linear part and a neural network approach for the nonlinear part. Efficient and reliable implementations of these methods are available through the SLICOT-based toolbox SLIDENT.

Overview

- System Identification
- Subspace State-space System Identification
- Basic Subspace Identification Algorithm
- Mathematical Foundations
- Algorithmic Details
- Estimation of a Wiener System
- SLICOT-based Software Tools
- Numerical Results
- Summary and Future Work

System Identification

System identification: finding mathematical models of dynamic systems from measured data.

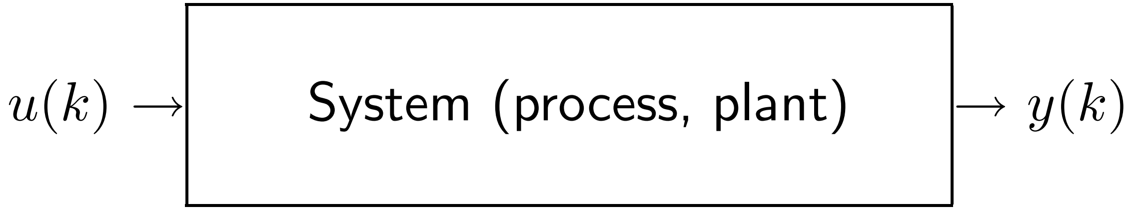


Figure 1: Dynamic system with input vector u and output vector y.

Note: u causally influences y, possibly by intermediate, state variables x.

Discrete-time LTI case:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k, \quad k = 1, 2, \ldots \\ y_k &= Cx_k + Du_k + v_k, \quad u_k := u(k),\ y_k := y(k). \end{aligned}$$

Typically, one uses

$$U = \begin{bmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_t^T \end{bmatrix}, \qquad Y = \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_t^T \end{bmatrix}.$$

Example: 120 MW Power Plant

DAISY Example [96-003], http://www.esat.kuleuven.ac.be/sista/daisy
Number of data samples: $t = 200$. Sampling time: $\approx$ 1200 seconds.

Outputs (3):
steam pressure,
main steam temperature,
reheat steam temperature.

Inputs (5):
gas flow,
turbine valves opening,
super heater spray flow,
gas dampers,
air flow.

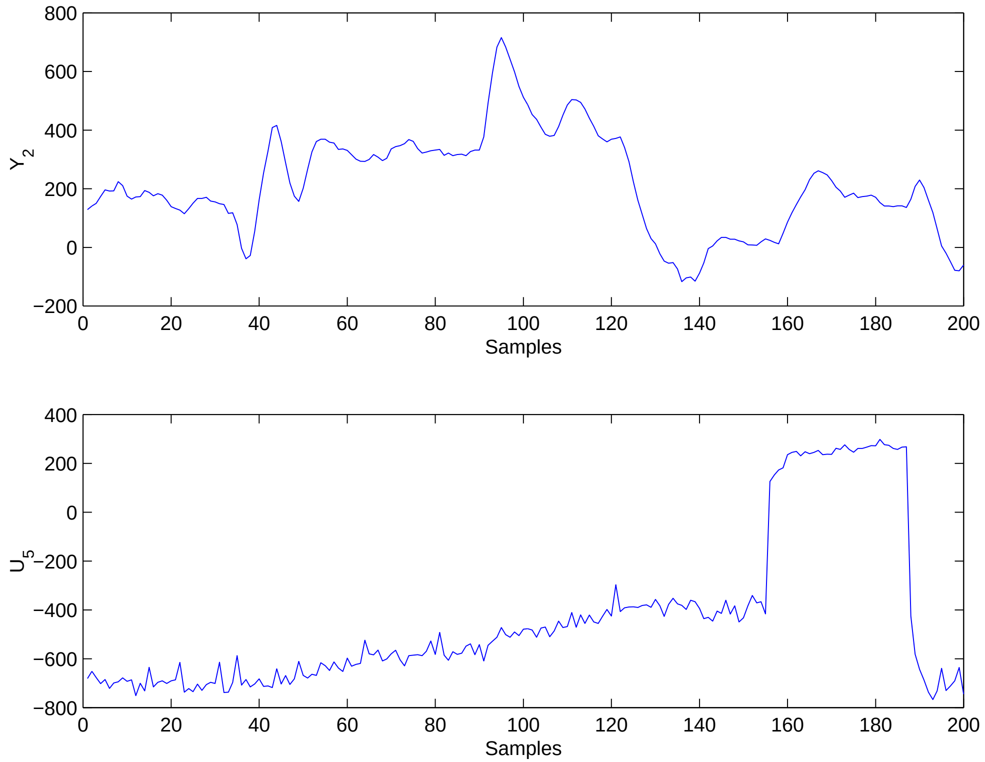


Figure 2: Output y_2, Input u_5.

Remarks: • *noisy data*; • *trends*.

Mathematical Models

Mathematical models are used for:

analysis
simulation
prediction
optimization
monitoring
fault detection
training
control/synthesis ...

Finding models. Approaches:

white-box – first principles;
gray-box – known structure, parameters = ?;
black-box – just data measurements (system identification).

Models/Systems Types:

linear	nonlinear
time-invariant	time-varying
discrete-time	continuous-time
lumped parameters	distributed parameters

Common models: those in the LH side. Very useful around nominal regimes.

Key References

From the 1970's, tremendous development: dozens of books, hundreds of papers, several software tools, much practical experience.

- *time-series analysis*: [*Åström/Eykhoff '71*], [*Eikhoff '74*], [*Box/Jenkins '76*], [*Söderström/Stoica '89*];
- *prediction error methods (PEM)*: [*Ljung '87*];
- *subspace methods*:
  - *realization theory*: [*Ho/Kalman '66*], [*Kung '78*], [*Moore '81*];
  - *stochastic realization*: [*Faurre '76*], [*Van Overschee/De Moor '93*], [*Akaike '75*];
  - *deterministic realization*: [*De Moor '88*], [*Moonen et al. '89*];
  - *combined stochastic/deterministic realization*: [*Larimore '83*], [*Van Overschee/De Moor '94, '96*], [*Verhaegen '93, '94*], [*Verhaegen/Dewilde '92*].

Surveys for subspace methods: [*Viberg '95*], [*Van Overschee/De Moor '96*], [*De Moor/Van Overschee/Favoreel '99*], etc.

Special issues: *Automatica* (Jan. '94, Dec. '95), *Signal Processing* (July '96).

Software Tools (selective):

- published codes, e.g., [*Van Overschee/De Moor '96*];
- commercial codes, e.g., MATLAB Identification Toolbox, [*Ljung '88–'00*];
- (copyrighted) free codes, e.g., SLICOT Library + MATLAB/Scilab Interfaces, [*Benner et al. '99*], [*Sima et al. '00–'03*].

Subspace State-Space System Identification

State space model:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k, \\ y_k &= Cx_k + Du_k + v_k, \end{aligned} \qquad (1)$$

x_k is n-dimensional state vector at k, u_k is m-dimensional input (control) vector, y_k is ℓ-dimensional output vector, $\{w_k\}$, $\{v_k\}$ are zero-mean, stationary ergodic state and output disturbance or noise sequences (uncorrelated with $\{u_k\}$ and initial state of (1)), with

$$\mathcal{E}\left[\begin{bmatrix} w_p \\ v_p \end{bmatrix} \begin{bmatrix} w_q^T & v_q^T \end{bmatrix} \right] = \begin{bmatrix} Q & S \\ S^T & R_v \end{bmatrix} \delta_{pq} \geq 0,$$

and A, B, C, D are real matrices, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{\ell\times n}$, $D \in \mathbb{R}^{\ell\times m}$.

Basic System Identification Problem:

Find n, and system matrices (A, B, C, D), using input and output (I/O) data sequences, $\{u_k\}$ and $\{y_k\}$, $k = 1\colon t$, and an upper bound, s, on n.

Note: Non-uniqueness of (A, B, C, D), up to a similarity transformation.

Control and Prediction

The identified model could be used, e.g., for either controlling a system, or predicting its behavior.

Given the initial state estimate $\widehat{x}_1$, and the input and output trajectories, $\{u_k\}$ and $\{y_k\}$, $k = 1\!:t$, the predicted output can be computed using the formulas

$$
\boxed{
\begin{array}{rcl}
\widehat{y}_k & = & \widehat{C}\widehat{x}_k + \widehat{D}u_k, \\
\widehat{x}_{k+1} & = & \widehat{A}\widehat{x}_k + \widehat{B}u_k + K(y_k - \widehat{y}_k),
\end{array}
}
$$

where the estimated quantities are marked by hat signs (suppressed in the sequel, for convenience).

The Kalman predictor gain matrix K is computed using the model and covariances.

If K not available, its contribution is omitted. In that case, the predicted output might not be very good, if the actual system includes disturbance terms.

Subspace System Identification Summary

Abbreviations

I/O : input/output;
MOESP : Multivariable Output Error state SPace;
N4SID : Numerical algorithm for Subspace State Space System IDentification;
SMI : Subspace Model Identification;
SVD : Singular Value Decomposition;
LS : Least Squares.

SMI advantages:

– no parameterizations needed;
– robust linear algebra tools (QR and SVD);
– only one parameter to be selected, s (a strict upper bound on n).

Classes of SMI techniques:

– State Intersection (SI)—N4SID;
– Output Error—MOESP.

Motivation

State-space system models are basis for:
– modern systems theory;
– advanced industrial applications.

Need for highly reliable and efficient algorithms and associated software for solving system identification problems.

SLIDENT—The new system identification toolbox incorporated in the Fortran 77 **S**ubroutine **L**ibrary **in** **CO**ntrol **T**heory (**SLICOT**)—explicitly addresses these quality requirements.

SLIDENT is freely available (for academic use) from the NICONET Web page http://www.win.tue.nl/niconet

Basic Subspace Identification Algorithm

Non-sequential data processing

1. Construct (explicitly or implicitly)

$$H = \left[\; U_{1,q,N}^T \quad Y_{1,q,N}^T \;\right], \quad N \times (m+\ell)q, \qquad \text{(N4SID)},$$

where $N = t - q + 1$, and $U_{1,q,N}$ and $Y_{1,q,N}$ are block-Hankel matrices, e.g.,

$$U_{1,q,N} = \begin{bmatrix} u_1 & u_2 & u_3 & \cdots & u_N \\ u_2 & u_3 & u_4 & \cdots & u_{N+1} \\ u_3 & u_4 & u_5 & \cdots & u_{N+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_q & u_{q+1} & u_{q+2} & \cdots & u_{N+q-1} \end{bmatrix}.$$

For MOESP with past I/O and N4SID, $q = 2s$; s: the "number of block rows".

2. Use a QR factorization, $H = QR$, for data compression (Q not needed);

3. Compute a SVD of a matrix built from R;
   $n =$ number of "non-zero" singular values.
   E.g., the MOESP approach finds the SVD of $R_{ms+1:(2m+\ell)s,(2m+\ell)s+1:2(m+\ell)s}$, while the N4SID approach first computes an "oblique projection."

4. Find system matrices from the right singular vectors, and other submatrices of the matrix R.

5. Find covariance matrices using the residuals of a least squares problem.

6. Find the Kalman gain by solving a discrete-time algebraic matrix Riccati equation.

Mathematical Foundations

Split up the system (1) into deterministic and stochastic parts, $x_k = x_k^d + x_k^s$, $y_k = y_k^d + y_k^s$,

$$\begin{aligned} x_{k+1}^d &= Ax_k^d + Bu_k, & x_{k+1}^s &= Ax_k^s + w_k, \\ y_k^d &= Cx_k^d + Du_k, & y_k^s &= Cx_k^s + v_k, \end{aligned}$$

and define the "past" and "future" parts

$$\begin{aligned} U_p &= U_{1,s,N}, & U_f &= U_{s+1,2s,N+s}, \\ Y_p &= Y_{1,s,N}, & Y_f &= Y_{s+1,2s,N+s}, \end{aligned}$$

and similarly define the block-Hankel matrices for y_k^s, w_k, and v_k, as Y_*^s, M_*, and N_*, respectively, with $* \in \{p, f\}$.

From (1),

$$\begin{aligned}
Y_* &= \Gamma_s X_*^d + H_s^d U_* + Y_*^s, \quad * \in \{p, f\} \\
Y_*^s &= \Gamma_s X_*^s + H_s^s M_* + N_*, \quad \text{with} \\
\Gamma_s &= \begin{bmatrix} C^T & (CA)^T & (CA^2)^T & \cdots & (CA^{s-1})^T \end{bmatrix}^T, \\
X_p^l &= \begin{bmatrix} x_1^l & x_2^l & x_3^l & \cdots & x_N^l \end{bmatrix}, \quad l \in \{d, s\}, \\
X_f^l &= \begin{bmatrix} x_{s+1}^l & x_{s+2}^l & x_{s+3}^l & \cdots & x_{s+N}^l \end{bmatrix},
\end{aligned}$$

H_s^l, $l \in \{d, s\}$, are lower block triangular Toeplitz matrices in Markov parameters

$$H_s^d = \begin{bmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CAB & CB & D & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{s-2}B & CA^{s-3}B & \cdots & \cdots & D \end{bmatrix} =: H_s^d(A, B, C, D),$$

and $H_s^s := H_s^d(A, I, C, 0)$.

Projecting the row space of Y_f above into the orthogonal complement of U_f, $U_f^\perp$, we have asymptotically (for $t \to \infty$)

$$\begin{aligned} Y_f/U_f^\perp &= \Gamma_s X_f^d/U_f^\perp + Y_f^s/U_f^\perp \\ &= \Gamma_s X_f/U_f^\perp + H_s^s M_f + N_f, \end{aligned}$$

since $U_f/U_f^\perp = 0$ and the noise is uncorrelated with the inputs, where $X_f = X_f^d + X_f^s$.

Weighting to the left and right with W_1 and W_2, chosen so that

- $\text{rank}(W_1\Gamma_s) = \text{rank}(\Gamma_s)$;
- $\text{rank}(X_f) = \text{rank}(X_f/U_f^\perp W_2)$;
- $M_f W_2 = 0, \quad N_f W_2 = 0,$

it follows,

$$\mathcal{O}_s := W_1 Y_f/U_f^\perp W_2 = W_1 \Gamma_s X_f/U_f^\perp W_2.$$

From SVD of $\mathcal{O}_s$,

$$\mathcal{O}_s = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} S_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix},$$

$$\Longrightarrow \boxed{n = \text{rank}(\mathcal{O}_s); \quad W_1\Gamma_s = U_1 S_1^{1/2}, \quad \widetilde{X}_s := X_f/U_f^{\perp} W_2 = S_1^{1/2} V_1^T \ .}$$

MOESP and N4SID use $W_1 = I_{\ell s}$ and

$$W_2 = \begin{cases} (W_p/U_f^{\perp})^{\dagger}(W_p/U_f^{\perp}), & \text{for MOESP}, \\ (W_p/U_f^{\perp})^{\dagger} W_p, & \text{for N4SID}, \end{cases}$$

where

$$W_p = \begin{bmatrix} U_p \\ Y_p \end{bmatrix}.$$

Main Subspace Identification Theorem

Assuming that:

1. $\{u_k\}$ *is uncorrelated with* $\{w_k\}$ *and* $\{v_k\}$*;*
2. $\{u_k\}$ *is* persistently exciting of order $2s$, *i.e., rank*$(U_{1,2s,N}U_{1,2s,N}^T) = 2ms$*;*
3. $N \to \infty$*;*
4. $\{w_k\}$ *and* $\{v_k\}$ *are not identically 0;*
5. $W_2 = (W_p/U_f^\perp)^\dagger W_p$ *(N4SID);*

then

1. *The system order equals the number of nonzero singular values of* $\mathcal{O}_s$.
2. $\mathcal{O}_s$ *is the* oblique projection *of future outputs into the past inputs and outputs along the future inputs,* $\mathcal{O}_s = W_1 Y_f /_{U_f} W_p$*, and it can be factored as* $\mathcal{O}_s = W_1 \Gamma_s \widetilde{X}_s$*, where* Γ_s *is the* extended observability matrix*, and* $\widetilde{X}_s$ *is a* Kalman filter estimated state sequence of X_f.
3. Γ_s *and* $\widetilde{X}_s$ *can be recovered from* $\Gamma_s = W_1^{-1} U_1 S_1^{1/2}$ *and* $\widetilde{X}_s = S_1^{1/2} V_1^T$*, respectively.*

Computation of System Matrices

Use *shift invariance property* of Γ_s:

$$C = \Gamma_s(1{:}\ell, :), \qquad A = \underline{\Gamma_s}^{\dagger}\overline{\Gamma_s},$$

where $\overline{\Gamma_s}$ and $\underline{\Gamma_s} = \Gamma_{s-1}$ denote Γ_s without the first and last ℓ rows, respectively.

In principle, system matrices could be found from the LS problem

$$\begin{bmatrix} \widetilde{X}_{s+1} \\ Y_{s+1,s+1,N+s} \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \widetilde{X}_{s} \\ U_{s+1,s+1,N+s} \end{bmatrix} + \begin{bmatrix} \rho_w \\ \rho_v \end{bmatrix},$$

(generally giving biased estimates) and the covariance matrices from

$$\begin{bmatrix} Q & S \\ S^T & R_v \end{bmatrix} \approx \frac{1}{N} \begin{bmatrix} \rho_w \\ \rho_v \end{bmatrix} \begin{bmatrix} \rho_w^T & \rho_v^T \end{bmatrix}.$$

Algorithmic Details

Sequential data processing: Consider several batches, (U_1, Y_1), (U_2, Y_2), . . . ,

$$U = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}, \qquad Y = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}, \ldots.$$

Let H_i be the block-Hankel matrix for (U_i, Y_i).

QR Factorization

1. Compute standard QR, $H_1 = Q_1 R_1$. Set $i = 2$.
2. Update R_1 using a specialized QR

$$\begin{bmatrix} R_1 \\ H_i \end{bmatrix} = QR, \quad \begin{bmatrix} \widetilde{R}_1 \\ 0 \end{bmatrix} = R.$$

   ($2(m+\ell)s$ Householder transformations of the same order are used.)

3. Repeat 2 for each additional data batch i.

Cholesky factorization

1. Build the inter-correlation matrix, $G = H^T H$, exploiting the block-Hankel structure.
2. Factor $G = R^T R$, assuming $G > 0$.

Details

$$G = \begin{bmatrix} G_{uu} & G_{uy} \\ G_{uy}^T & G_{yy} \end{bmatrix},$$

G_{vw} consists of $2s \times 2s$ submatrices of orders $m \times m$ (for $v = u, w = u$), $m \times \ell$ (for $v = u, w = y$), and $\ell \times \ell$ (for $v = y, w = y$).
Let $G_{vw}^{i,j}$—the (i,j)-th submatrix of G_{vw}. The first block-row of G_{uu} is given by

$$\boxed{G_{uu}^{1,j} = \widehat{G}_{uu}^{1,j} + u_1 u_j^T + u_2 u_{j+1}^T + \cdots + u_N u_{j+N-1}^T,}$$

$j = 1\colon 2s$, $\widehat{G}_{uu}^{i,j}$ is either a zero matrix, if the first (or single) data batch is computed, or the currently computed $G_{uu}^{i,j}$, otherwise.

Exploiting the block-Hankel structure,

$$G_{uu}^{i+1,j+1} = \widehat{G}_{uu}^{i+1,j+1} - \widehat{G}_{uu}^{i,j} + G_{uu}^{i,j} + u_{i+N}u_{j+N}^T - u_i u_j^T,$$

$j = 1\colon 2s-1$, and $i = 1\colon j$; only upper triangular part evaluated for $i = j$.

Compute G_{yy} and G_{uy} similarly.
For G_{uy}, the first block-row and block-column are fully computed by an "expensive" formula, while the other blocks follow from updating formulas.

Fast QR factorization is also included, based on displacement rank techniques. The generators of $H^T H$ are computed and then used to obtain R.

If Cholesky, or fast QR factorization algorithm fails, the QR factorization is automatically used, for non-sequential processing.

Computation of System Matrices (Details)

Other computational steps: analyzed for exploiting any existing structure.

Determination of weighted "oblique projection" $\mathcal{O}$:

Partition $R = \left[\begin{array}{cccc} U_p & U_f & Y_p & Y_f \end{array} \right]$, with ms, ms, ℓs, and ℓs columns (p - "past", f - "future"). **Note:** Notation differs from that used before.

Let $W_p = \left[\begin{array}{cc} U_p & Y_p \end{array} \right]$, and

$$r_1 = W_p - U_f X_1, \qquad r_2 = Y_f - U_f X_2,$$

the residuals of the two LS problems giving $\mathcal{O}$,

$$\min \|U_f X - W_p\|_2, \qquad \min \|U_f X - Y_f\|_2.$$

Then, with MOESP weightings, $\mathcal{O}^M = r_2^T Q_1 Q_1^T$, with Q_1 denoting the first rank(r_1) columns of the matrix Q in the QR factorization of r_1.

No least squares problems should be actually solved. Both problems: the same U_f, consisting of two $ms \times ms$ submatrices, the second - upper triangular.

Fast algorithm for B and D

A structure-exploiting QR factorization algorithm for computing B and D is available. Essentially, this algorithm solves the problem

$$\begin{bmatrix} Q_{1s} & \cdots & Q_{12} & Q_{11} \\ 0 & \cdots & Q_{13} & Q_{12} \\ 0 & \cdots & Q_{14} & Q_{13} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & Q_{1s} \end{bmatrix} \begin{bmatrix} \Gamma_- & 0 \\ 0 & I_\ell \end{bmatrix} \begin{bmatrix} B \\ D \end{bmatrix} = \begin{bmatrix} K_1 \\ K_2 \\ K_3 \\ \vdots \\ K_s \end{bmatrix},$$

where $\Gamma_- \in \mathbb{R}^{(\ell s-\ell)\times n}$, $Q_{1i} \in \mathbb{R}^{(\ell s-n)\times \ell}$, and $K_i \in \mathbb{R}^{(\ell s-n)\times m}$, $i = 1\colon s$.
The first matrix is fast triangularized, and B and D are then found in two steps.

- The matrix $[Q_{ij}]$ is a block permutation of the matrix appearing in literature.
- LS solution is obtained only if the second LHS matrix is square and nonsingular.
- For true LS solutions—algorithm based on Kronecker products with a matrix having half the size of the corresponding original N4SID matrix.
- Computation of K_j might be ill-conditioned.

Simulation-based algorithm for B and D

A *simulation-based* algorithm is also included for the computation of B and D. Specifically, denoting

$$X = \left[\ (\text{vec}(D^T))^T \quad (\text{vec}(B))^T \quad x_0^T \ \right]^T,$$

then X is the LS of $SX = \text{vec}(Y)$, with

$$S = \left[\ \text{diag}(U) \quad y^{11} \quad \cdots \quad y^{n1} \quad y^{12} \quad \cdots \quad y^{nm} \quad P\Gamma \ \right],$$

where $\text{diag}(U) \in \mathbb{R}^{lt \times lm}$ has ℓ-by-ℓ blocks, Γ is given by

$$\Gamma = \left[\ C^T \quad (CA)^T \quad (CA^2)^T \quad \cdots \quad (CA^{t-1})^T \ \right]^T,$$

P is a permutation matrix that groups together the rows of Γ depending on the same row c_j of C, for $j = 1\colon \ell$, and y^{ij}, $j = 1\!: m$, $i = 1\!: n$, are computed using the following model,

$$\begin{aligned} x^{ij}(k+1) &= Ax^{ij}(k) + e_i u_j(k), \quad x^{ij}(1) = 0, \\ y^{ij}(k) &= Cx^{ij}(k). \end{aligned}$$

The structure of the other block-columns of S is exploited.

The calculations are simpler if D and/or x_0 are not needed.

Recommended algorithm: Kronecker product-based algorithm.

Estimation of a Wiener System

Discrete-time Wiener system: linear part + static nonlinearity

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \\ z(k) &= Cx(k) + Du(k), \\ y(k) &= f\left(z(k)\right) + v(k), \end{aligned}$$

where $x(k)$, $u(k)$, and $y(k)$ defined, $z(k)$ — output of the linear part, and $f(\cdot)$ nonlinear vector function, $f(\cdot) : \mathbb{R}^\ell \rightarrow \mathbb{R}^\ell$.

The linear part, found by subspace techniques, is then parameterized using the output normal form, to reduce the number of its parameters to l (including the initial state vector, $x(1)$), $l := n(\ell + m + 1) + \ell m$.

Nonlinear part is modeled by a set of ℓ single layer neural networks,

$$f_r\,(z(k)) = \hat{f}_r\,(z(k)) + \epsilon_r(k)\,, \qquad r = 1, \ldots, \ell\,,$$

$$\hat{f}_r\,(z(k)) := \sum_{i=1}^{\nu}\left(\alpha(r,i)\phi\left(\sum_{j=1}^{\ell}\beta(r,i,j)z_j(k) + b(r,i)\right)\right) + b(r,\nu+1)\,, \qquad (2)$$

where $z_r(k)$—the r-th entry of $z(k) := z_k$, $\epsilon(k)$—approximation error, ν—number of neurons, and $\alpha(r,i)$, $\beta(r,i,j)$, $b(r,i)$ and $b(r,\nu+1)$—real numbers to be estimated.

The estimation problem formulated as a structured nonlinear least squares (NLS) problem, solved in three steps.

Conceptual algorithm

Step 1: identify linear part assuming $f(\cdot)$ identity (subspace approach).

Step 2: find initial values of weights for $\hat{f}_r$ in (2). (Hyperbolic tangent used as ϕ.) All α, β, b stacked in θ,

$$\theta = \left(\theta_1^T \,|\, \theta_2^T \,|\, \cdots \,|\, \theta_\ell^T \right)^T \in \mathbb{R}^{\ell((\ell+2)\nu+1)},$$

Solve the NLS problem

$$\min_\theta \sum_{k=1}^{N} \left\| \begin{bmatrix} y_1(k) - \hat{y}_1(k) \\ \vdots \\ y_\ell(k) - \hat{y}_\ell(k) \end{bmatrix} \right\|^2, \qquad (3)$$

with $\hat{y}_r(k) := \hat{f}_r(\hat{z}_k)$, $\hat{z}_k$—estimated output of linear part.
Note: (3) $\equiv \ell$ independent NLS problems, solved separately.

Conceptual algorithm: continued

Step 3: optimize parameters of linear + nonlinear parts, starting with values corresponding to the results of Steps 1 and 2.

Linear part parameters added at the end of θ $\rightarrow$ Jacobian matrix of optimization problem is block diagonal + a right block column:

$$J = \left[\begin{array}{cccc|c} J_1 & 0 & \cdots & 0 & L_1 \\ 0 & J_2 & \cdots & 0 & L_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & J_\ell & L_\ell \end{array} \right], \quad J_{\mathsf{C}} = \begin{bmatrix} J_1 & L_1 \\ J_2 & L_2 \\ \vdots & \vdots \\ J_\ell & L_\ell \end{bmatrix},$$

where $J_r \in \mathbb{R}^{N\times((\ell+2)\nu+1)}$ and $L_r \in \mathbb{R}^{N\times l}$ are full matrices, corresponding to the nonlinear and linear part, respectively, $r = 1\colon \ell$.

The submatrices J_r, $r = 1\colon \ell$, are computed analytically, and the block-matrix $\left[\, L_1^T \;\cdots\; L_\ell^T \,\right]^T$ is computed by a forward-difference approximation. The Jacobian J is stored in a compressed form, J_{C}.

The full nonlinear least squares problem is written as (3), with θ replaced by Θ, $\Theta \in \mathbb{R}^c$, $c := \ell((\ell+2)\nu+1)+l$, and it is no longer separable. This problem, as well as the ℓ separate problems in (3), are solved by a Levenberg-Marquardt algorithm.

Specialized implementations of the Levenberg-Marquardt (LM) algorithm:

a. standard implementation: Cholesky factorization for solving symmetric positive definite linear systems;
b. standard implementation: conjugate gradients (CG) algorithm (idem);
c. MINPACK-like, but LAPACK-based, structure-exploiting QR factorization.

SLICOT-based Software Tools

SLIDENT abilities

- Flexibility of usage $\Leftarrow$ options:
  - deterministic and stochastic identification;
  - MOESP, N4SID, MOESP+N4SID;
  - standard or fast techniques for data compression;
  - multiple data batches processing;
  - *non-sequential*, and *sequential data processing*;
  - fully documented drivers and computational routines (on-line, `html`).
- Efficiency:
  - structure-exploiting algorithms;
  - fast algorithms for data compression (exploit block-Hankel structure);
  - LAPACK-based.
- Reliability:
  - condition numbers returned.

SLICOT system identification routines

IB01AD	preprocesses the I/O data and estimates n (driver).
IB01BD	estimates (A, C, B, D), covariances and K (driver).
IB01CD	estimates x_0 and/or B, D, given (A, B, C, D), or (A, C), and I/O trajectories (driver).
IB01MD	computes R from I/O data.
IB01MY	computes R using fast QR.
IB01ND	finds the SVD, using R.
IB01OD	finds n, using the SVD.
IB01OY	asks for user's confirmation of n.
IB01PD	estimates system and covariance matrices.
IB01PX	computes B and D using Kronecker products.
IB01PY	computes B and D using a structure exploiting algorithm.
IB01QD	estimates x_0 and B, D, given (A, C) and the I/O trajectories.
IB01RD	estimates x_0, given (A, B, C, D) and the I/O trajectories.
IB03AD	estimates the parameters of a Wiener system, using a standard Levenberg-Marquardt algorithm (Cholesky or CG-based).
IB03BD	idem, using a MINPACK-like Levenberg-Marquardt algorithm.

SLIDENT: MEX-file interfaces to Matlab

`order`	preprocesses the I/O data for estimating system matrices and n.
`sident`	computes (A, B, C, D), Kalman gain K, and covariances, given n and part of R, using MOESP, N4SID, or combination.
`findBD`	estimates x_0 and/or B and D, given A, C, possibly B, D, and the I/O trajectories.
`widentc`	estimates the parameters of a Wiener system, using a standard Levenberg-Marquardt algorithm (Cholesky or CG-based).
`wident`	idem, using a MINPACK-like Levenberg-Marquardt algorithm.

```
[R,n(,sval)(,rcnd)] = order(meth,alg,jobd,batch,conct,...
                            s,Y(,U,tol,printw,ldwork,R));

[(A,C)(,B(,D))(,K(,Q,Ry,S))(,rcnd)] = sident(meth,job,s,n,...
                            l,R(,tol,t,A,C,printw));

[(x0)(,B(,D))(,V)(,rcnd)] = findBD(jobx0,comuse(,job),A(,B),...
                            C(,D),Y(,U,tol,printw,ldwork));
```

SLIDENT: M-file interfaces

`findR`	preprocesses the I/O data and estimates n.
`findABCD`	finds system matrices and Kalman gain, given n and part of R, using MOESP, N4SID, or MOESP+N4SID.
`findAC`	finds A and C, given n and part of R, using MOESP or N4SID.
`findBDK`	finds B and D and Kalman gain, given n, A, C, and part of R, using MOESP, N4SID, or MOESP+N4SID.
`inistate`	estimates x_0, given system matrices, and a set of I/O data.
`findx0BD`	estimates x_0 and/or B and D, given A, C, and a set of I/O data.
`slmoesp`	SLICOT MOESP (method-oriented).
`sln4sid`	SLICOT N4SID (method-oriented).
`slmoen4`	MOESP + N4SID (method-oriented).
`slmoesm`	MOESP + simulation (method-oriented).

Calling sequences for computational M-files

```
[R,n(,sval,rcnd)]        = findR(s,Y(,U,meth,alg,jobd,tol,printw));
[sys(,K,Q,Ry,S,rcnd)] = findABCD(s,n,l,R(,meth,nsmpl,tol,printw));
[A,C(,rcnd)]             = findAC(s,n,l,R(,meth,tol,printw));
[B(,D,K,Q,Ry,S,rcnd)] = findBDK(s,n,l,R,A,C(,meth,job,nsmpl,tol,
                                     printw));
[x0(,V,rcnd)]            = inistate(sys,Y(,U,tol,printw));
[x0,B,D(,V,rcnd)]        = findx0BD(A,C,Y(,U,withx0,withd,tol,printw));
```

Shorter calls

```
[sys,rcnd] = findABCD(s,n,l,R);
[B,D,rcnd] = findBDK(s,n,l,R,A,C);
x0          = inistate(A,C,Y);
[B,D]       = findx0BD(A,C,Y,U,0);
```

Calling sequences for method-oriented files

```
[sys(,K,rcnd,R)]    = slmoesp(s,Y(,U,n,alg,tol,printw));
[sys(,K,rcnd,R)]    = sln4sid(s,Y(,U,n,alg,tol,printw));
[sys(,K,rcnd,R)]    = slmoen4(s,Y(,U,n,alg,tol,printw));
[sys(,K,rcnd,x0,R)] = slmoesm(s,Y(,U,n,alg,tol,printw));
```

If `n = 0`, or `n = []`, or `n` is omitted, the user is prompted to provide its value, after inspecting the singular values, shown as a bar plot.

If `n < 0`, `n` is determined automatically, according to `tol(2)`.

Shorter calls, e.g.:

```
[sys,K] = slmoesp(s,Y);
[sys,K] = slmoesp(s,Y,[],n);
 sys    = slmoesp(s,Y,U);
```

The first two calls estimate the matrices A, C, and K of a stochastic system (with no inputs), for an order `n` found automatically, or specified, respectively.

Parameter `R` returns the processed upper triangular factor R of the block-Hankel-block matrix H, built from the input-output data.
It can be used for fast identification of systems of various orders, using, e.g., the following commands:

```
[sys,K,rcnd,R] = sln4sid(s,Y,U,n0,alg);
for n = n0+1 : min( n0+nf, s-1 )
   [sys,K,rcnd] = sln4sid(s,Y,U,n,R);
   ...
end
```

Inside the loop, the data for `Y` and `U` are not used (only size(`Y`) is needed), but `R` replaces `alg`. The systems of orders (`n0+1:min(n0+nf,s-1)`) should be used inside the loop.
`rcnd(1)` and `rcnd(2)` set to 1 when `sln4sid` is called with `R` instead of `alg`.

Numerical Results

Data sets used

The data sets used (except for Appl. 22), are available on the DAISY site

`http://www.esat.kuleuven.ac.be/sista/daisy`

for increasing accessibility and reproducibility (see Table 1).

Table 1: *Summary description of applications*

#	Application	t	m	ℓ	s	n
1	Ethane-ethylene distillation column	4×90	5	3	5	4
2	Glass furnace	1247	3	6	10	5
3	120 MW power plant	200	5	3	10	8
4	Industrial evaporator	6305	3	3	10	4
5	Simulation data for a pH neutralization process	2001	2	1	15	6
6	Industrial dryer	867	3	3	15	10

Table 1: *continued*

#	Application	t	m	ℓ	s	n
7	Liquid-saturated steam heat exchanger	4000	1	1	15	5
8	Test setup of an industrial winding process	2500	5	2	15	6
9	Continuous stirred tank reactor	7500	1	2	15	5
10	Model of a steam generator	9600	4	4	15	9
11	Ball-and-beam	1000	1	1	20	2
12	Laboratory setup for a hair dryer	1000	1	1	15	4
13	CD-player arm	2048	2	2	15	8
14	Wing flutter	1024	1	1	20	6
15	Flexible robot arm	1024	1	1	20	4
16	Steel subframe flexible structure	8523	2	28	21	20
17	Cutaneous potential of a pregnant woman	2500	0	8	21	14
18	Western basin of Lake Erie	4×57	5	2	5	4
19	Heat flow through a two layer wall	1680	2	1	20	3
20	Heating system	801	1	1	15	7
21	1 hour Internet traffic at Berkeley Laboratory	99999	0	1	8	2
22	Glass tubes	1401	2	2	20	8

Linear systems identification results

Numerical results: on a Sun 4 SPARC Ultra-2 computer, using OS 5.6, Sun WorkShop Compiler FORTRAN 77 5.0 and MATLAB 5.3.0.10183 (R11).

On an IBM PC computer, 500 MHz, 128 Mb memory, with Digital or Compaq Visual Fortran, version $>$ V5.0, and/or with MATLAB 6.5, the results are similar.

The simplest calls have been used for standard calculations, e.g.,

```
[sys,K,rcnd] = slsolver(s,y,u,n,alg);
```

where `solver` is `moesp`, `n4sid`, `moen4`, or `moesm`. The notation `moesp`, `n4sid`, `moen4` and `moesm` with indices 1, 2, or 3, indicate the algorithm used in SLICOT implementation: fast Cholesky, fast QR, and standard QR, respectively.

Alternative MATLAB codes for comparison:
MOESP (corresponds to `slmoesm`) and N4SID.
SLIDENT function `slmoesp`: refined version of an older MOESP code (OMOESP).

Relative output errors computed with
`err = norm(y - ye,1)/norm(y,1);` % `ye` $:= \widehat{y}$.

Relative output errors using QR or Cholesky factorization (selection)

#	Relative output errors			
	slmoesp	sln4sid N4SID	OMOESP	slmoesm MOESP
2	6.03e-01	6.21e-01	6.42e-01	4.96e-01
4	5.50e-01	5.53e-01	5.67e-01	4.89e-01
11	3.61e-01	2.78e-01	2.21e+01	2.54e-01
12	3.30e-02	2.16e-02	7.33e-02	1.50e-02
13	1.13e+02	3.45e-01	8.06e+04	1.76e-01
14	2.24e+02	2.94e-01	6.49e+10	2.37e-01
15	1.14e-01	4.51e-02	7.38e+04	3.60e-02
19	4.23e-01	1.38e-01	4.20e-01	1.38e-01
22	5.39e-01	6.09e-01	1.02e+01	4.86e-01

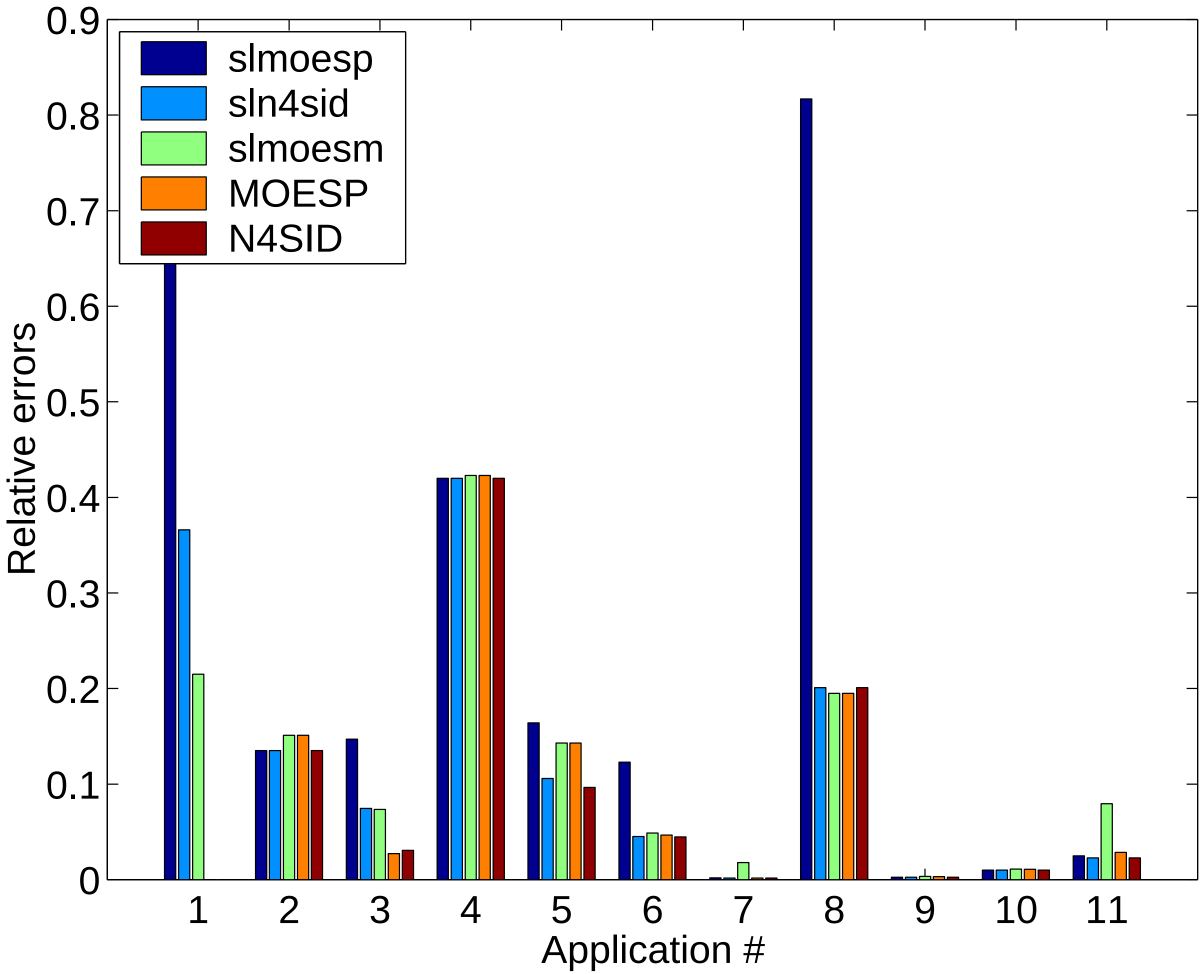
Remarks: MOESP could not solve the identification problem for Application 16 ("Out of memory" error message), and N4SID did not finish after 16 hours of execution on the Sun machine.

CPU time (sec. on a Sun) for computing the system matrices using SLICOT Cholesky factorization and Matlab codes (selection)

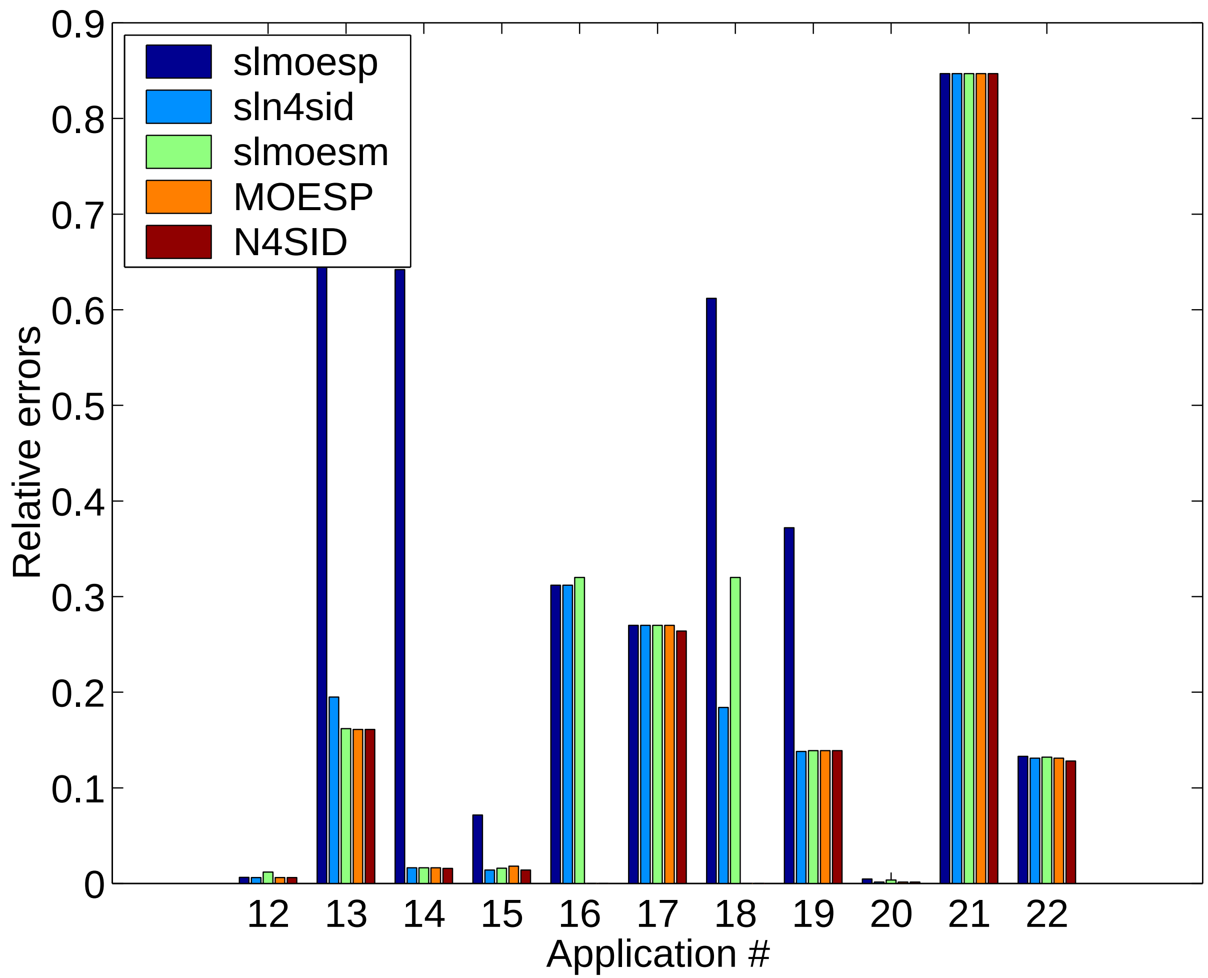
#	Time				
	slmoesp	sln4sid	OMOESP	MOESP	N4SID
2	0.27	0.36	3.46	4.10	3.48
4	0.32	0.37	8.23	10.29	8.13
11	0.03	0.04	0.69	0.48	0.62
12	0.01	0.03	0.38	0.33	0.41
13	0.10	0.16	2.83	3.37	2.80
14	0.37	0.37	0.71	0.55	0.68
15	0.03	0.04	0.72	0.51	0.69
19	0.08	0.13	2.13	1.75	2.10
22	0.04	0.06	0.76	1.32	0.92

Speed-up factors:

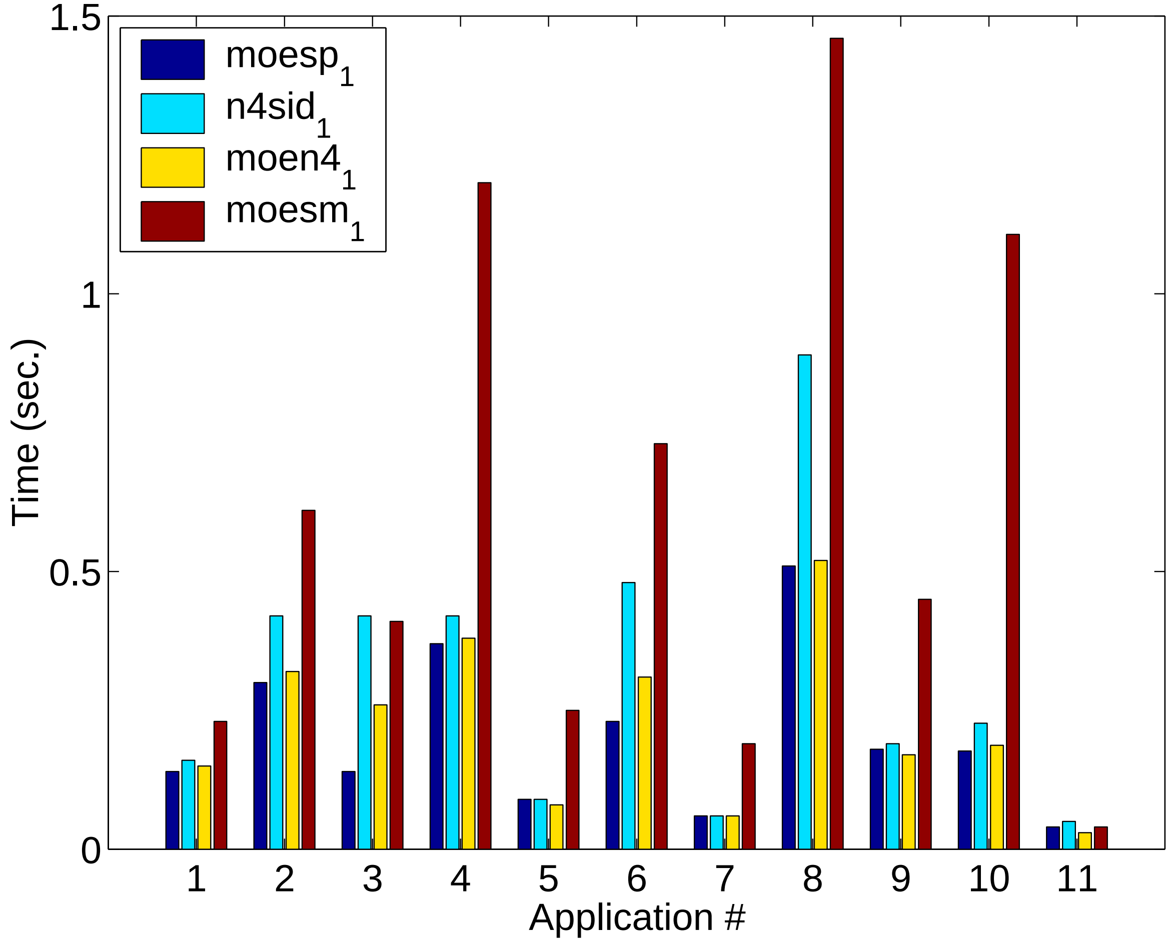
- 10 to 20 comparing to SLICOT QR factorization algorithm;
- 15 to 40 (and even over 200) comparing to MATLAB codes.



Relative output errors, Applications 1–11.

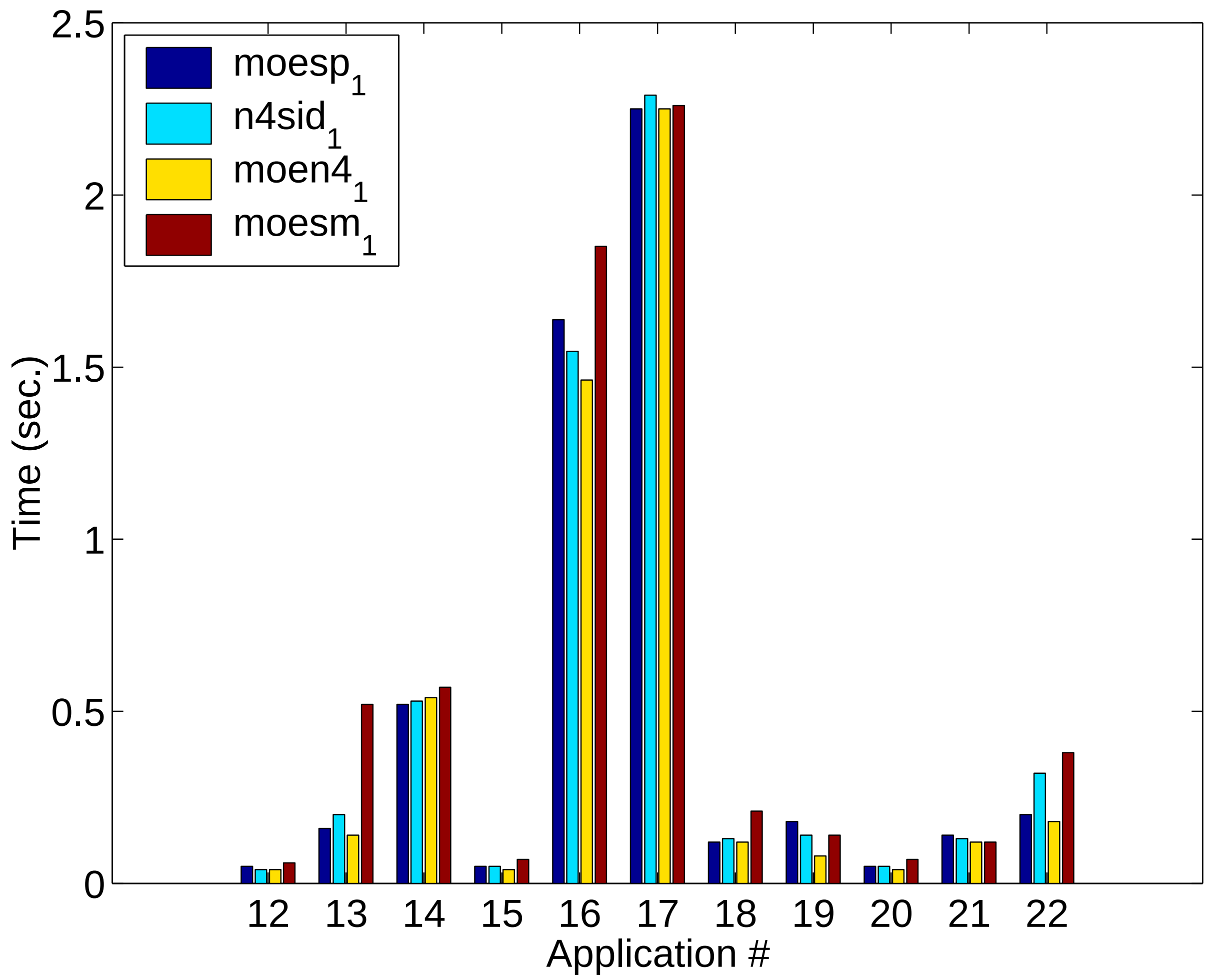


Relative output errors, Applications 12–22.



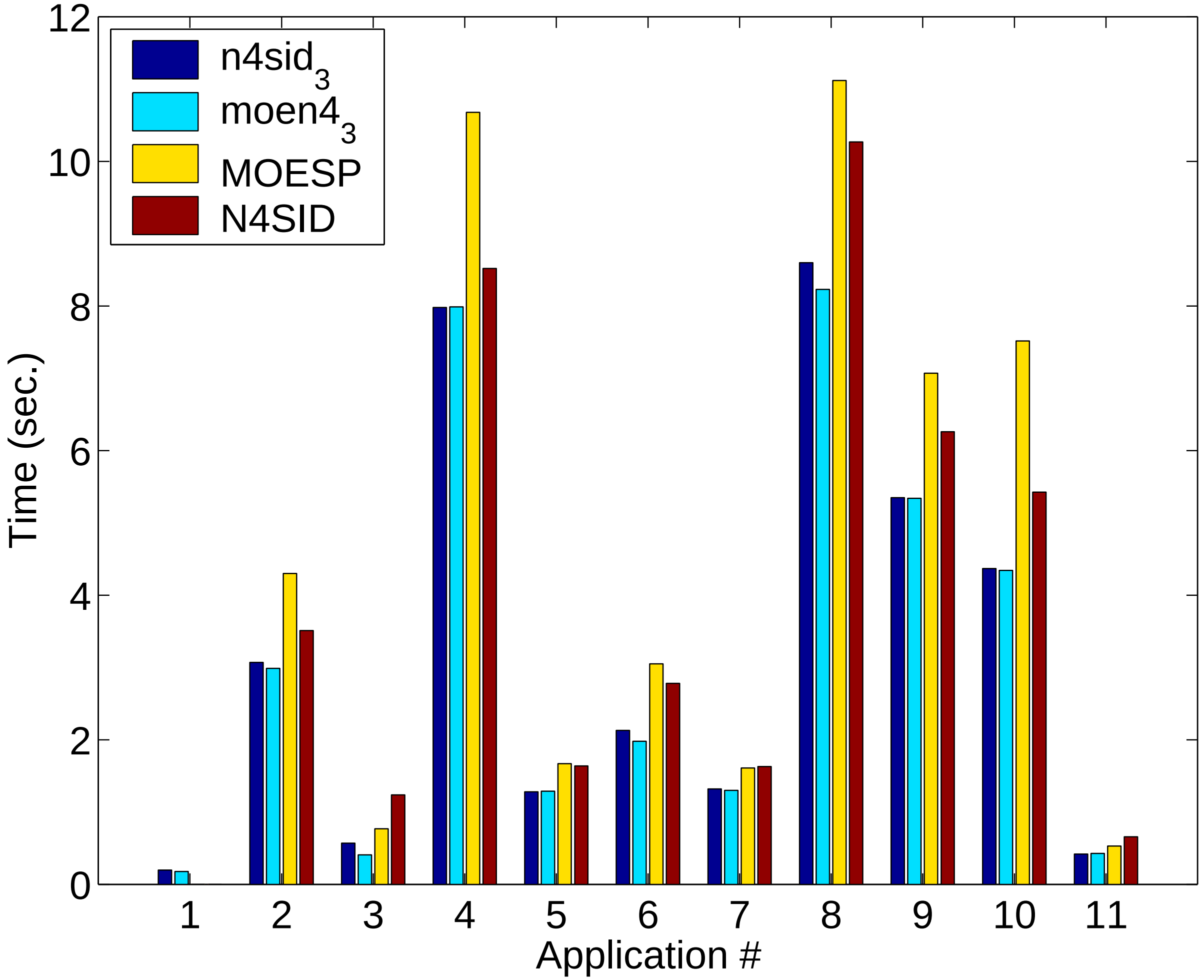
Timings for fast Cholesky algorithm, Applications 1–11.

Note: Times for Appl. 10 are divided by 10.



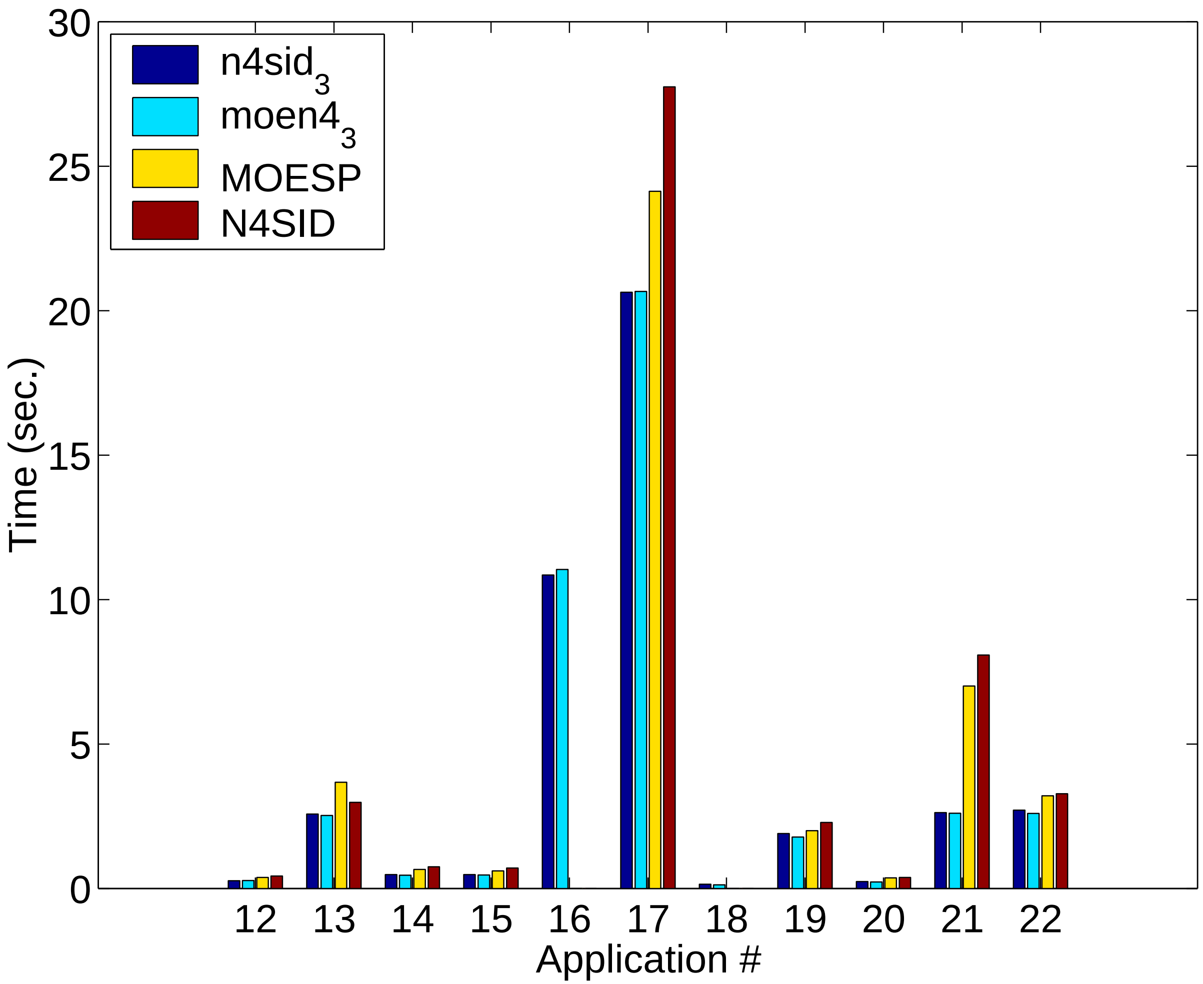
Timings for fast Cholesky algorithm, Applications 12–22.

Note: Times for Appl. 16 are divided by 100.



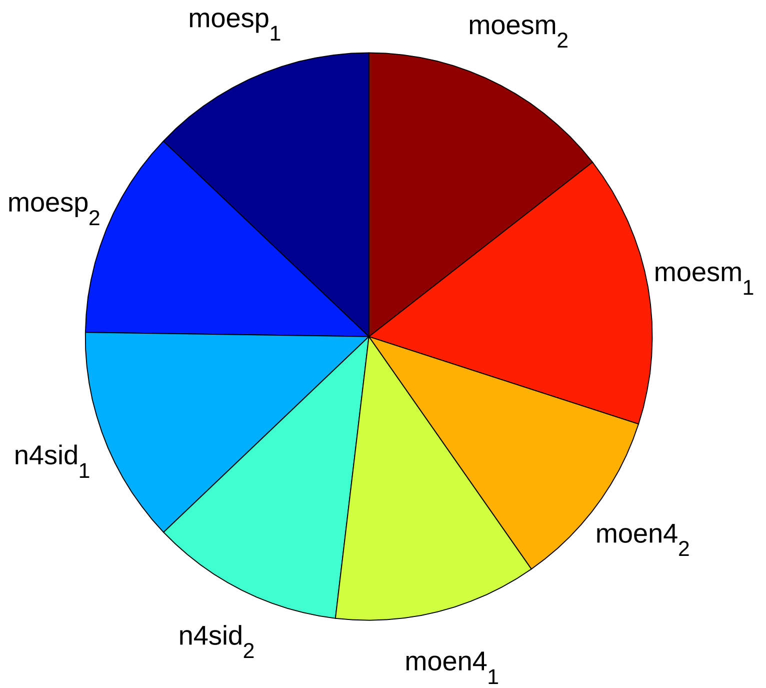
Timings for QR algorithm, Applications 1–11.

Note: Times for Appl. 10 are divided by 10.

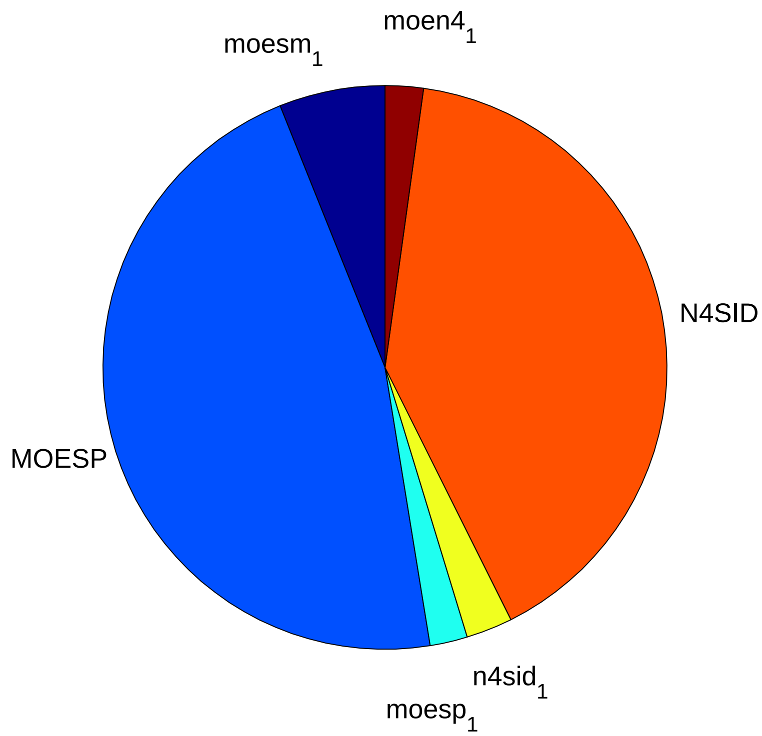


Timings for QR algorithm, Applications 12–22.

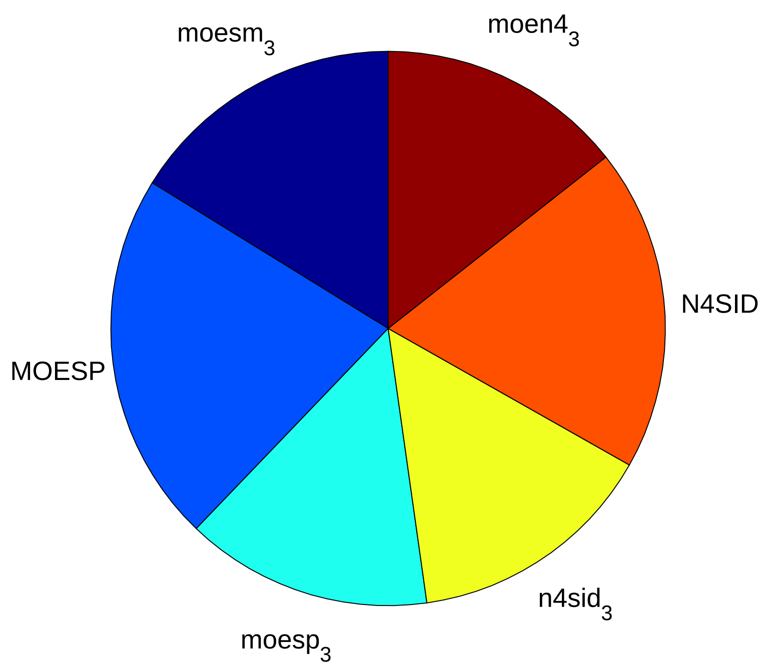
Note: Times for Appl. 16 are divided by 100.



Timing comparison (cumulative): fast Cholesky versus fast QR factorization algorithms.



Timing comparison (cumulative): fast Cholesky versus QR factorization algorithms.



Timing comparison (cumulative): SLIDENT QR versus MATLAB QR factorization algorithms.

Example: 120 MW power plant

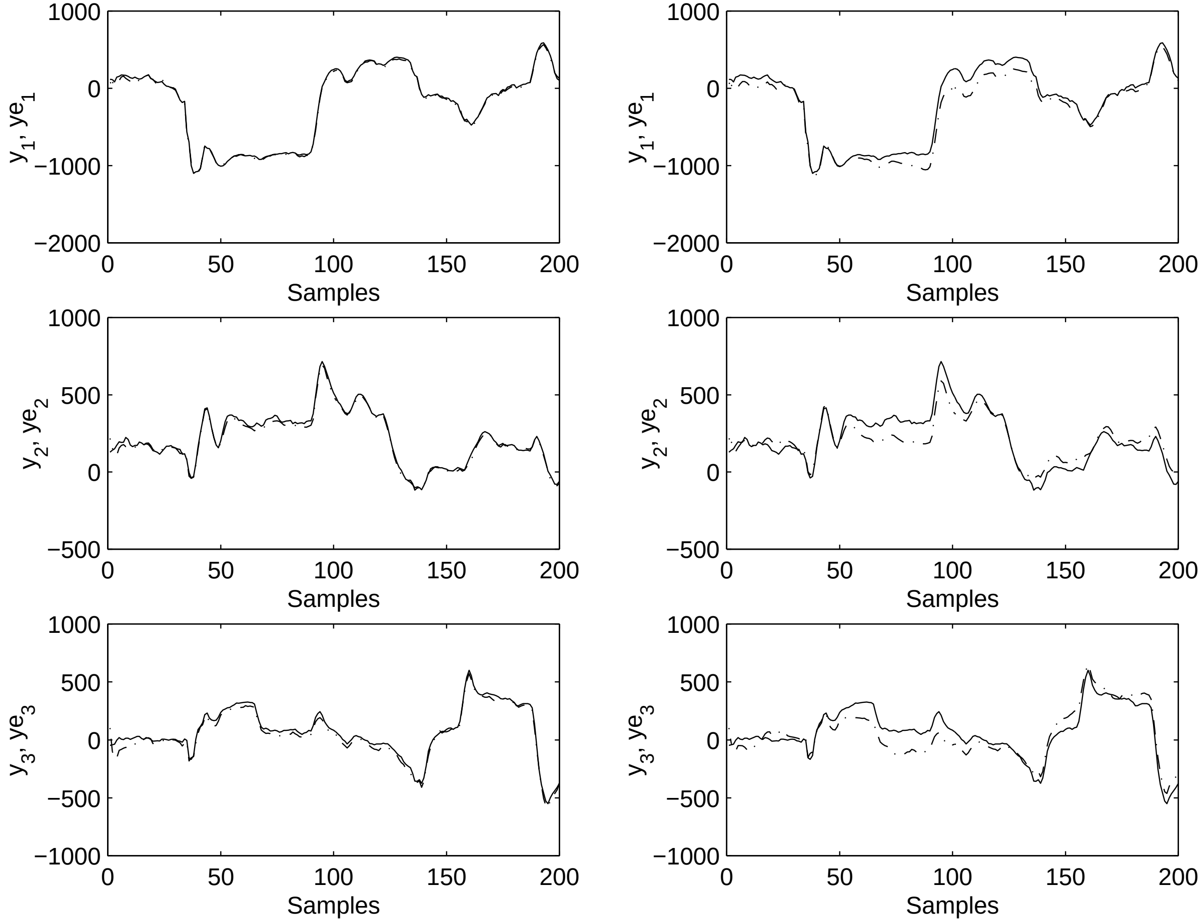


Figure 3: Output (solid) and estimated output (dash-dotted) trajectories for the Application 3, with (left), or without (right) Kalman predictor.

Wiener systems identification results

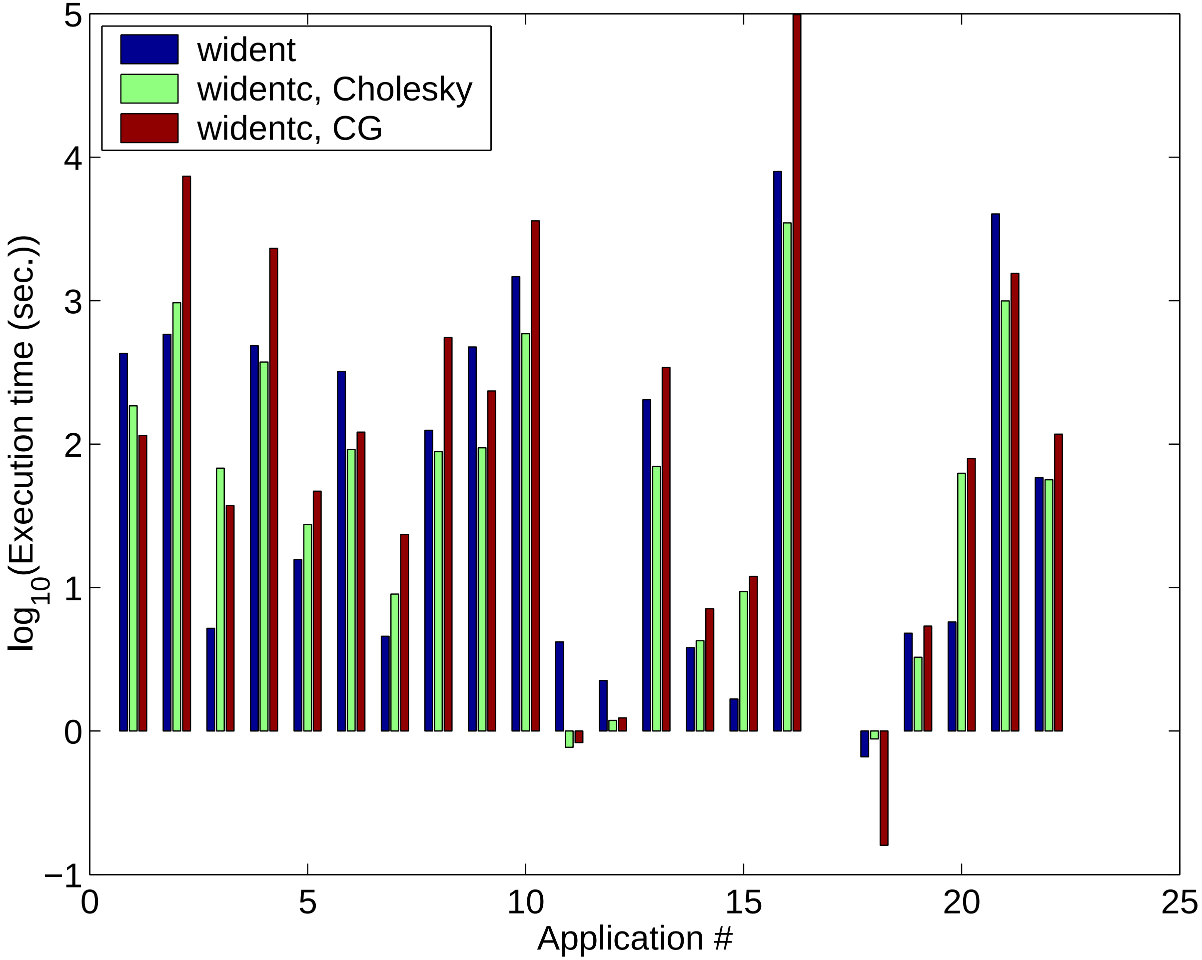


Figure 4: Decimal logarithms of the execution times in seconds (on the PC machine) for solving the Wiener system identification problem.

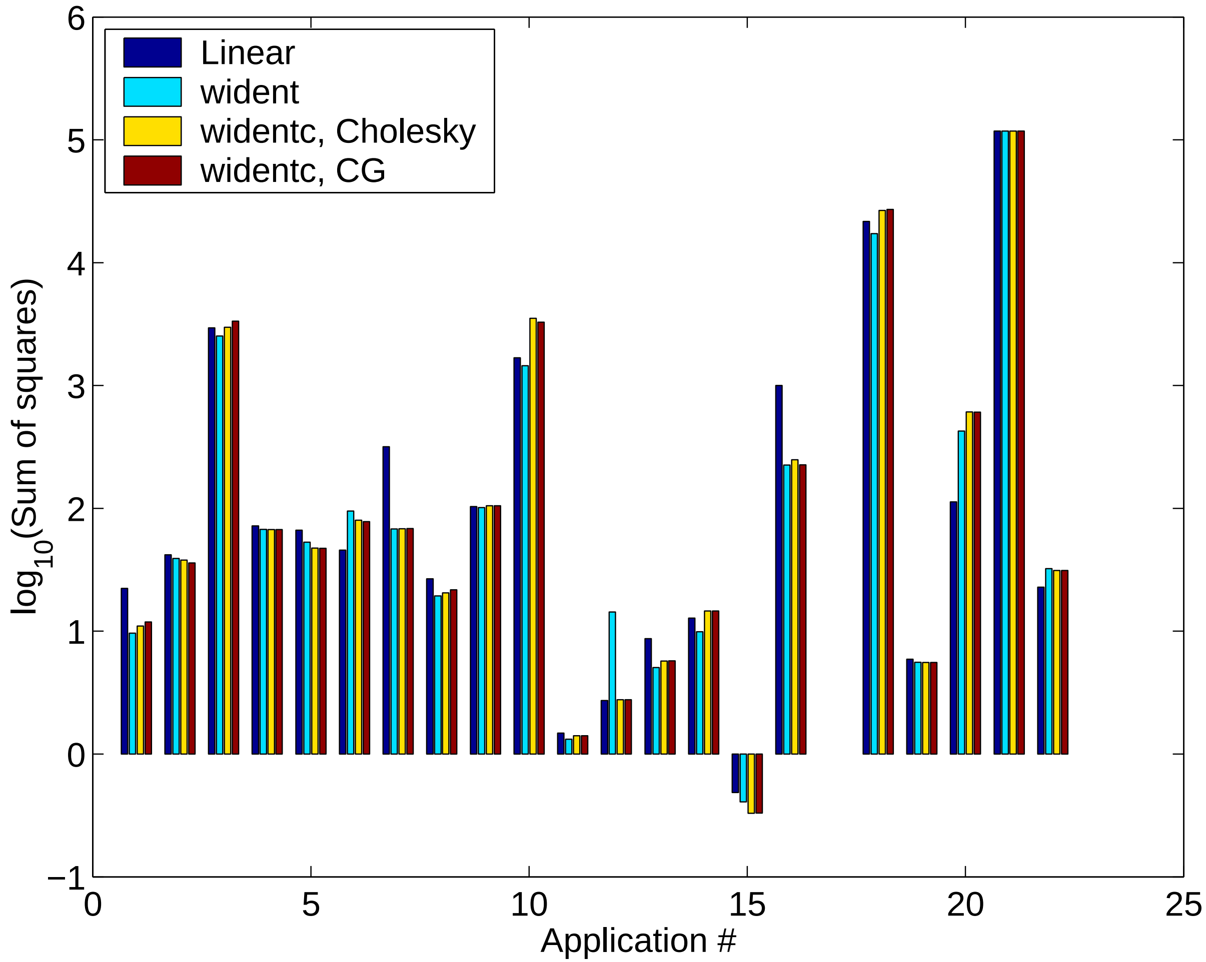


Figure 5: Decimal logarithms of the sums of squares of the prediction errors for solving the linear and Wiener system identification problem.

NLS optimization problem for Application 16, Steel subframe flexible structure: 8523 samples, 2 inputs, and 28 outputs (too large for standard workstations).

Simplified problem solved:

- first half of I/O data used for estimation (all for validation),
- first 7 outputs only modeled,
- system order $n = 20$,
- 12 neurons for each output.

Corresponding optimization problem:
- 977 variables,
- $7 \times \lfloor 8523/2 \rfloor = 29827$ nonlinear error functions.

	QR	Cholesky	CG
Execution times (sec.):	7956.51	3481.84	98595.72
Sum of Squares:	155	179	155
Error norms, all samples:	225	249	226

Hence, faster Cholesky code was less accurate.

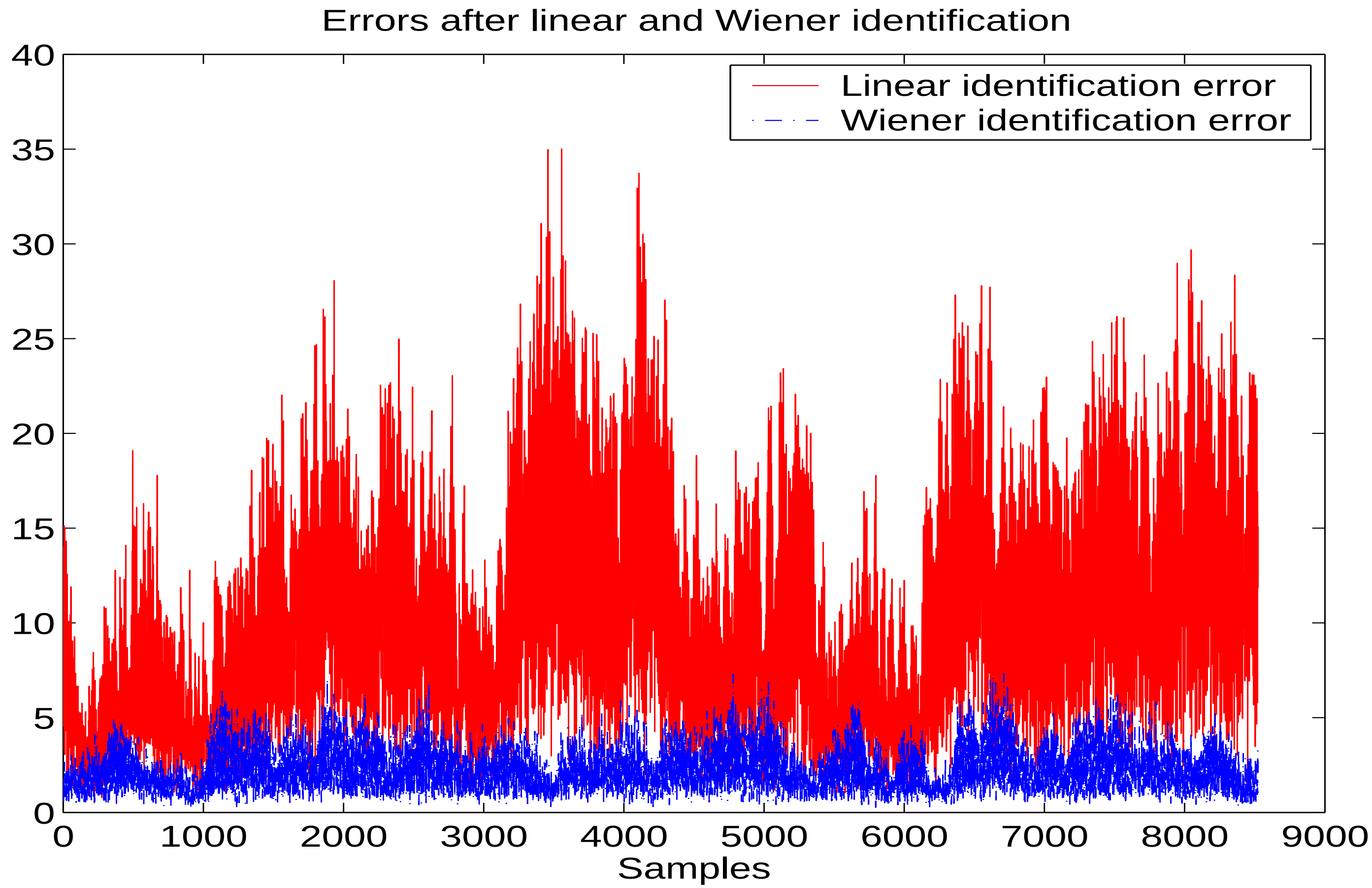


Figure 6: Prediction error norms for Application 16 for linear and Wiener system identification ($t = 8523$, $N = t/2$, $c = 977$, the first 7 outputs only).

Wiener model • significantly reduces prediction error; • has a smoothing effect.

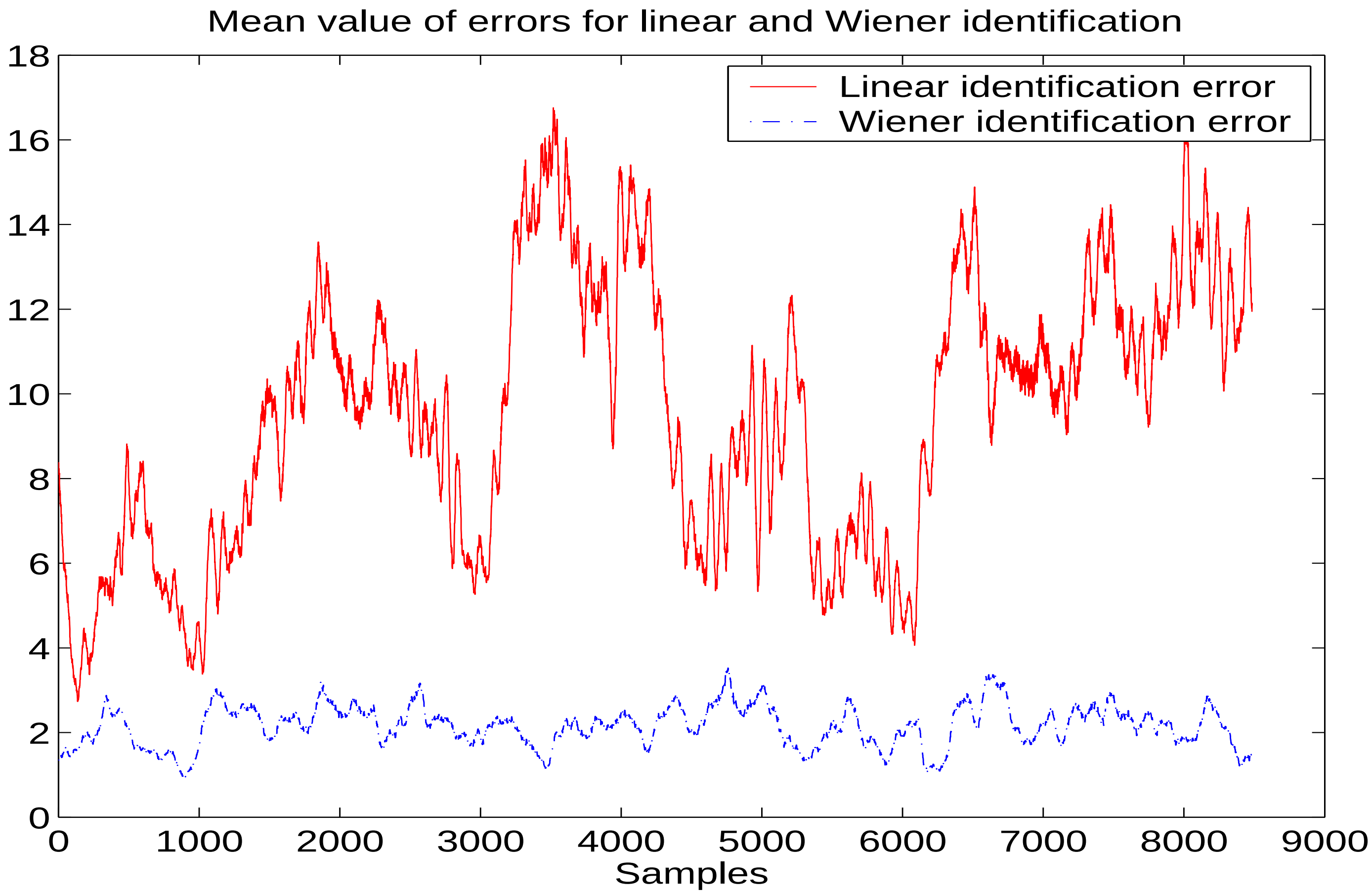


Figure 7: Mean values of errors (on a moving window with 40 samples) for linear and Wiener identification for Application 16 ($N = t/2$, the first 7 outputs only).

Summary

- System identification has important applications.
- Impressive advances in the last 3 decades.
- Algorithmic and numerical details on subspace-based techniques for system identification have been described and compared.
- The techniques are implemented in the new system identification toolbox for the SLICOT Library.
- The results show that the fast algorithmic variants included in the toolbox can frequently be used, and they are significantly more efficient than the standard QR factorization and the existing MATLAB codes.
- SLICOT codes are reliable and able to solve large identification problems.

Future Work

- Further improving the performance and reliability of the SLICOT codes.
- Developing new algorithms or their variations.
- Extensions for other problem classes, e.g., nonlinear systems (bilinear, Hammerstein, etc.).

Further Reading

[1] H. Akaike, *Markovian representation of stochastic processes by canonical variables*, SIAM J. Control, 13 (1975), pp. 162–173.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide: Third Edition*, Software · Environments · Tools, SIAM, Philadelphia, 1999.

[3] K. J. Åström and P. Eykhoff, *System identification: A survey*, Automatica, 7 (1971), pp. 123–167.

[4] G. E. Box and G. M. Jenkins, *Time Series Analysis, Forecasting and Control*, Holden-Day Series in Time Series Analysis and Digital Processing, Holden-Day, Oakland, CA, revised ed., 1976.

[5] B. De Moor, P. Van Overschee, and W. Favoreel, *Numerical algorithms for subspace state-space system identification: An overview*, in Applied and Computational Control, Signals, and Circuits, B. N. Datta, ed., vol. 1, chapter 6, Birkhäuser, 1999, pp. 247–311.

[6] P. Eykhoff, *System Identification*, Wiley, London, 1974.

[7] P. Faurre, *Stochastic realization algorithms*, in System Identification: Advances and Case Studies, R. Mehra and D. Lainiotis, eds., Academic Press, 1976.

[8] B. L. Ho and R. E. Kalman, *Efficient construction of linear state variable models from input/output functions*, Regelungstechnik, 14 (1966), pp. 545–548.

[9] S. Y. Kung, *A new identification method and model reduction algorithm via singular value decomposition*, in Proceedings of the 12th Asilomar Conference on Circuits, Systems and Computation, Asilomar, CA, USA, 1978, pp. 705–714.

[10] W. Larimore, *System identification, reduced order filtering and modeling via canonical variate analysis*, in Proceedings of the American Control Conference, San Francisco, CA, USA, 1983, pp. 445–451.

[11] L. Ljung, *System Identification: Theory for the User*, Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

[12] L. Ljung, *System Identification Toolbox For Use with MATLAB. User's Guide, Version 5*, The MathWorks, Inc, 3 Apple Hill Drive, Natick, MA 01760–2098, 2000.

[13] N. Mastronardi, D. Kressner, V. Sima, P. Van Dooren, and S. Van Huffel, *A fast algorithm for subspace state-space system identification via exploitation of the displacement structure*, J. Comput. Appl. Math., 132 (2001), pp. 71–81.

[14] M. Moonen, B. De Moor, L. Vandenberghe, and J. Vandewalle, *On- and off-line identification of linear state space models*, Int. J. Control, 49 (1989), pp. 219–232.

[15] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *User's guide for MINPACK-1*, Report ANL-80-74, Applied Math. Division, Argonne National Laboratory, Argonne, Illinois, 1980.

[16] R. Schneider, A. Riedel, V. Verdult, M. Verhaegen, and V. Sima, *SLICOT system identification toolbox for nonlinear Wiener systems*, SLICOT Working Note 2002-6, Katholieke Universiteit Leuven (ESAT/SISTA), Leuven, Belgium, June 2002. Available from `ftp://wgs.esat.kuleuven.ac.be/pub/WGS/REPORTS/`, file `SLWN2002-6.ps.Z`, 26 pages.

[17] V. Sima, *Algorithms and LAPACK-based software for subspace identification*, in Proceedings of The 1996 IEEE International Symposium on Computer-Aided Control System Design, September 15–18, 1996, Ritz-Carlton, Dearborn, Michigan, U.S.A., 1996a, pp. 182–187.

[18] V. Sima, *Subspace-based algorithms for multivariable system identification*, Studies in Informatics and Control, 5 (1996b), pp. 335–344.

[19] V. Sima, *Cholesky or QR factorization for data compression in subspace-based identification ?*, in Proceedings of the Second NICONET Workshop on "Numerical Control Software: SLICOT, a Useful Tool in Industry", December 3, 1999, INRIA Rocquencourt, France, 1999, pp. 75–80.

[20] V. Sima, *SLICOT linear systems identification toolbox*, SLICOT Working Note 2000-4, Katholieke Universiteit Leuven (ESAT/SISTA), Leuven, Belgium, July 2000. Available from `ftp://wgs.esat.kuleuven.ac.be/pub/WGS/REPORTS/`, file `SLWN2000-4.ps.Z`, 30 pages.

[21] V. Sima, *Comparison of subspace-based system identification software tools*, in Proceedings of the Third NICONET Workshop on "Numerical Software in Control Engineering", January 19, 2001, Hotel de Lauzelle, Louvain-la-Neuve, Belgium, 2001, pp. 97–104.

[22] V. Sima, *Identification of a steel subframe flexible structure using SLICOT system identification toolbox*, in CD-ROM Proceedings of The 11th Mediterranean Conference

on Control and Automation MED'03, June 18–20 2003, Rhodes, Greece, 2003. Invited session IV01, "Computational Toolboxes in Control Design", Paper IV01-07, 6 pages.

[23] V. Sima, D. M. Sima, and S. Van Huffel, *SLICOT system identification software and applications*, in Proceedings of the 2002 IEEE International Conference on Control Applications and IEEE International Symposium on Computer Aided Control System Design, CCA/CACSD 2002, September 18–20, 2002, Scottish Exhibition and Conference Centre, Glasgow, Scotland, U.K., Omnipress, 2002, pp. 45–50.

[24] V. Sima and S. Van Huffel, *SLICOT subspace identification toolbox*, in Proceedings CD of the UKACC International Conference on Control 2000, University of Cambridge, United Kingdom, 4-7 September, 2000, 2000a. 6 pages.

[25] V. Sima and S. Van Huffel, *Efficient numerical algorithms and software for subspace-based system identification*, in Proceedings of the 2000 IEEE International Conference on Control Applications and IEEE International Symposium on Computer-Aided Control Systems Design, September 25–27, 2000, Anchorage Hilton, Anchorage, Alaska, U.S.A., Omnipress, 2000b, pp. 1–6.

[26] V. Sima and S. Van Huffel, *Performance investigation of SLICOT system identification toolbox*, in Proceedings of the European Control Conference, ECC 2001, 4–7 September, 2001, Seminário de Vilar, Porto, Portugal, 2001, pp. 3586–3591.

[27] T. Söderström and P. Stoica, *System Identification*, Prentice-Hall International Series in Systems and Control Engineering, Prentice-Hall, London, 1989.

[28] P. Van Overschee and B. De Moor, *Subspace algorithms for the stochastic identification problem*, Automatica, 29 (1993), pp. 649–660.

[29] P. Van Overschee and B. De Moor, *N4SID: Two subspace algorithms for the identification of combined deterministic-stochastic systems*, Automatica, 30 (1994), pp. 75–93.

[30] P. Van Overschee and B. De Moor, *Subspace Identification for Linear Systems : Theory – Implementation – Applications*, Kluwer Academic Publishers, Boston/London/Dordrecht, 1996.

[31] M. Verhaegen, *Subspace model identification. Part 3: Analysis of the ordinary output-error state-space model identification algorithm*, Int. J. Control, 58 (1993), pp. 555–586.

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

[33] M. Verhaegen, *Identification of the temperature-product quality relationship in a multi-component distillation column*, Chemical Engineering Communications, 163 (1998), pp. 111–132.

[34] M. Verhaegen and P. Dewilde, *Subspace model identification. Part 1: The output-error state-space model identification class of algorithms*, Int. J. Control, 56 (1992), pp. 1187–1210.

[35] M. Viberg, *Subspace-based methods for the identification of linear time-invariant systems*, Automatica, 31 (1995), pp. 1835–1852.

Model and Controller Reduction

Andras Varga
German Aerospace Center
DLR - Oberpfaffenhofen
Institute of Robotics and System Dynamics
D-82234 Wessling (Germany)

E-mail: `Andras.Varga@dlr.de`
URL: `http://www.robotic.dlr.de/~varga/`

Abstract

Model reduction has become a standard tool in various control system analysis and design applications, ranging from simulation of highorder systems to simplification of large order plant models for efficient evaluation of design criteria in multidisciplinary optimization-based controller tuning. The talk focuses on methods underlying the numerical software for model and controller reduction available in SLICOT. We discuss absolute error model reduction methods such as the balanced truncation, singular perturbation approximation, and Hankel norm approximation, their frequency-weighted counterparts, as well as relative error methods based on balanced stochastic truncation. Designing low order controllers for practical applications involving high order plants is a challenging problem where model reduction techniques often play an important role. To perform controller reduction, special techniques capable to address closed-loop stability and performance preservation aspects are required. We discuss the newest algorithmic developments for controller reduction for which robust numerical software is available in SLICOT.

Outline

- applications of model reduction
- problem formulation
- classification
- historical perspective
- basic concepts
- model reduction methods
- controller reduction approaches
- software for model and controller reduction
- model reduction examples

Applications of model reduction

- Reduction of large order models
  - simulation of systems arising from discretization of partial differential equations
  - low order controller design
  - real time filter implementation
  - complementary to system identification
- Reduction of large order controllers
  - real time controller implementation
  - complementary to controller synthesis methods

Model reduction problem

Given the original system of order n

$$\left\{ \begin{array}{rcl} \dot{x} & = & Ax + Bu \\ y & = & Cx + Du \end{array} \right. \quad \Leftrightarrow \quad G(s) = C(sI - A)^{-1}B + D$$

compute a reduced system of order $r < n$

$$\left\{ \begin{array}{rcl} \dot{x}_r & = & A_r x_r + B_r u \\ y_r & = & C_r x_r + D_r u \end{array} \right. \quad \Leftrightarrow \quad G_r(s) = C_r(sI - A_r)^{-1}B_r + D_r$$

such that G_r approximates G as good as possible.

Classification of model reduction methods

- absolute error methods

$$\|G - G_r\| = \min$$

- relative error methods

$$\|G^{-1}(G - G_r)\| = \min$$

- frequency-weighted methods

$$\|W_o(G - G_r)W_i\| = \min$$

- special methods for controller reduction

Historical perspective

- modal approach: Davison (1966)
  – retain dominant modes of original system

- balanced truncation: Moore (1981)
  – a priori error bound for given order (Enns, 1984)

- optimal Hankel-norm approximation: Glover (1984)
  – exact solution; relevant to H_∞ -norm reduction

- frequency-weighted model reduction:
  – balanced truncation: Enns (1984)
  – Hankel-norm approximation: Latham & Anderson (1985)

- relative error methods: stochastic balanced truncation
  Desai & Pal (1984), Green (1988), Wang & Safonov (1990)

- controller reduction: Liu & Anderson (1989-90)
  – special methods (e.g., coprime factorization based)

- numerical methods:
  – square-root (SR) method: Tombs & Postlethwaite (1987)
  – balancing-free (BF) method: Wang & Safonov (1990)
  – SR & BF methods: Varga (1991,1992)
  – frequency-weighted SR & BF: Varga & Anderson (2001)
  – controller reduction: Varga & Anderson (2002,2003), Varga (2003)
  – large-scale systems: Van Dooren (1995), Penzl (1998), ...

Gramians

- P - controllability Gramian, Q - observability Gramian
  – for a stable, continuous-time system satisfy the Lyapunov equations

$$\begin{aligned} AP + PA^T + BB^T &= 0 \\ A^TQ + QA + C^TC &= 0 \end{aligned}$$

  – for a stable, discrete-time system satisfy the Stein equations

$$\begin{aligned} APA^T + BB^T &= P \\ A^TQA + C^TC &= Q \end{aligned}$$

- Properties: $P > 0 \Leftrightarrow (A, B)$ controllable
  $Q > 0 \Leftrightarrow (A, C)$ observable

Hankel singular values (HSV)

- $P = SS^T$ – controllability Gramian
  $Q = R^T R$ – observability Gramian

$$\sigma_i = \lambda_i^{1/2}(PQ) = \lambda_i^{1/2}(S^T QS) = \sigma_i(RS)$$

- Properties:
  – independent of the used (A, B, C, D) realization
  – $\#$(nonzero HSV) $=$ order of a minimal realization
  – small HSV $\Rightarrow$ system almost not minimal

- Balanced realization: $P = Q = \Sigma = \mathrm{diag}(\sigma_1, ..., \sigma_n)$

System balancing

- Let Z be a transformation matrix such that for the transformed system $(Z^{-1}AZ, Z^{-1}B, CZ, D)$ the transformed Gramians are equal

$$Z^T Q Z = Z^{-1} P Z^{-T} = \Sigma = \mathrm{diag}(\Sigma_1, \Sigma_2)$$

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

- Assuming $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_r \gg \sigma_{r+1} \geq \cdots \geq \sigma_n > 0$, partition the transformed system matrices as

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

Balanced truncation approximation (BTA)

- Define the reduced model as $G_r := (A_r, B_r, C_r, D_r) = (A_{11}, B_1, C_1, D)$
- Alternative computation: partition Z^{-1} and Z as

$$Z^{-1} = \begin{bmatrix} L \\ V \end{bmatrix}, \quad Z = [\, T \;\; U \,]$$

and compute $G_r := (LAT, LB, CT, D)$

Computational approach: determine only the truncation matrices L and T !

Singular perturbation approximation (SPA)

Define the reduced model as $G_r = (A_r, B_r, C_r, D_r)$, where for a continuous-time system

$$\left[\begin{array}{c|c} A_r & B_r \\ \hline C_r & D_r \end{array}\right] = \left[\begin{array}{c|c} A_{11} - A_{12}A_{22}^{-1}A_{21} & B_1 - A_{12}A_{22}^{-1}B_2 \\ \hline C_1 - C_2A_{22}^{-1}A_{21} & D - C_2A_{22}^{-1}B_2 \end{array}\right]$$

Main advantage: G and G_r have the same DC gains!

Hankel-norm approximation (HNA)

Define the reduced model as $G_r = (A_r, B_r, C_r, D_r)$, where A_r, B_r, C_r, and D_r are computed according to formulas developed by Glover (1984) to solve the optimal Hankel-norm approximation problem

$$\|G - G_r\|_H = \min$$

Main feature: lower guaranteed error bound.

Summary of additive error methods

- BTA, SPA and HNA are the basic methods for reduction of stable systems

- Approximation properties:
  - guaranteed stability of reduced models
  - guaranteed a priori error bound

$$\|G - G_r\|_\infty \leq 2 \sum_{j=r+1}^{n} \sigma_j$$

- Easy extensibility to reduce unstable systems in combination with modal or coprime factorization techniques

- Applicability: dense problems with $n \leq 1000$.

Accuracy enhancing square-root method

- solve the matrix Lyapunov equations to compute Gramians

$$\begin{aligned} AP + PA^T + BB^T &= 0 \\ A^TQ + QA + C^TC &= 0 \end{aligned}$$

directly for Cholesky factors S and R: $P = SS^T$, $Q = R^TR$.

- compute the SVD: $RS = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \mathrm{diag}(\Sigma_1, \Sigma_2) \begin{bmatrix} V_1 & V_2 \end{bmatrix}^T$

- compute the truncation matrices: $L = \Sigma_1^{-1/2} U_1^T R$, $T = SV_1\Sigma_1^{-1/2}$

Main feature: reduced model is balanced !

Alternative balancing-free approach

- solve the matrix Lyapunov equations to compute Gramians

$$\begin{aligned} AP + PA^T + BB^T &= 0 \\ A^TQ + QA + C^TC &= 0 \end{aligned}$$

directly for Cholesky factors S and R: $P = SS^T$, $Q = R^TR$.

- compute the SVD: $RS = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \mathrm{diag}(\Sigma_1, \Sigma_2) \begin{bmatrix} V_1 & V_2 \end{bmatrix}^T$
- compute the QR-decompositions: $SV_1 = XW, \quad R^TU_1 = YZ$
- compute the truncation matrices: $L = (Y^TX)^{-1}Y^T, \qquad T = X$

Main feature: L and T are well-conditioned !

Handling unstable systems: modal approach

- Compute Z such that

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

where A_{11} contains the dominant (or unstable) modes
A_{22} contains the non-dominant modes

- Apply BTA, SPA or HNA to (A_{22}, B_2, C_2) to obtain $(A_{r,22}, B_{r,2}, C_{r,2})$.

- Construct $\left[\begin{array}{c|c} A_r & B_r \\ \hline C_r & D_r \end{array}\right] = \left[\begin{array}{cc|c} A_{11} & O & B_1 \\ O & A_{r,22} & B_{r,2} \\ \hline C_1 & C_{r,2} & D \end{array}\right]$

Handling unstable systems: coprime factorization approach

- Compute a stable left coprime factorization $G = M^{-1}N$, where M and N are stable.

- Apply BTA, SPA or HNA to the extended system $[\, M \quad N \,]$ to obtain the reduced factors $[\, M_r \quad N_r \,]$.

- Compute the reduced system $G_r = M_r^{-1} N_r$.

Relative error methods: balanced stochastic truncation (BST)

- Model reduction problem: Given a high order stable plant model G, determine a reduced order model G_r such that

$$\|G^{-1}(G - G_r)\| = \min$$

- Computational approach:
  - compute the left spectral factor W such that $GG^* = W^*W$
  - compute P, the controllability Gramian of G and,
    Q, the observability Gramian of W
  - compute the reduced model G_r using square-root & balancing-free BTA or SPA techniques

- BST is often suitable to perform model reduction in order to obtain low order design models for controller synthesis !

- Approximation properties:
  – guaranteed stability of reduced models
  – approximates simultaneously gain and phase
  – preserves non-minimum phase zeros
  – guaranteed a priori error bound

$$\|G^{-1}(G - G_r)\|_\infty \leq 2 \sum_{j=r+1}^{n} \frac{1+\sigma_j}{1-\sigma_j} - 1$$

- Applicability: dense problems with $n \leq 200$
  – appropriate as final reduction step for absolute error methods
  – use $[\, G \;\; \alpha I \,]$ to combine absolute and relative error methods

- Restrictions: G must be full row rank

Frequency-weighted balanced truncation

- Model reduction problem: Given a high order stable plant model G, stable weighting-matrices W_o and W_i, determine a reduced order model G_r such that

$$\|W_o(G - G_r)W_i\| = \min$$

- Computational approach:
  - compute P, the controllability Gramian of GW_i and,
    Q, the observability Gramian of W_oG
  - compute truncation matrices L and T using square-root & balancing-free techniques to obtain $G_r = (LAT, LB, CT, D)$.

No nice error bounds are known !!

Frequency-weighted Hankel norm approximation

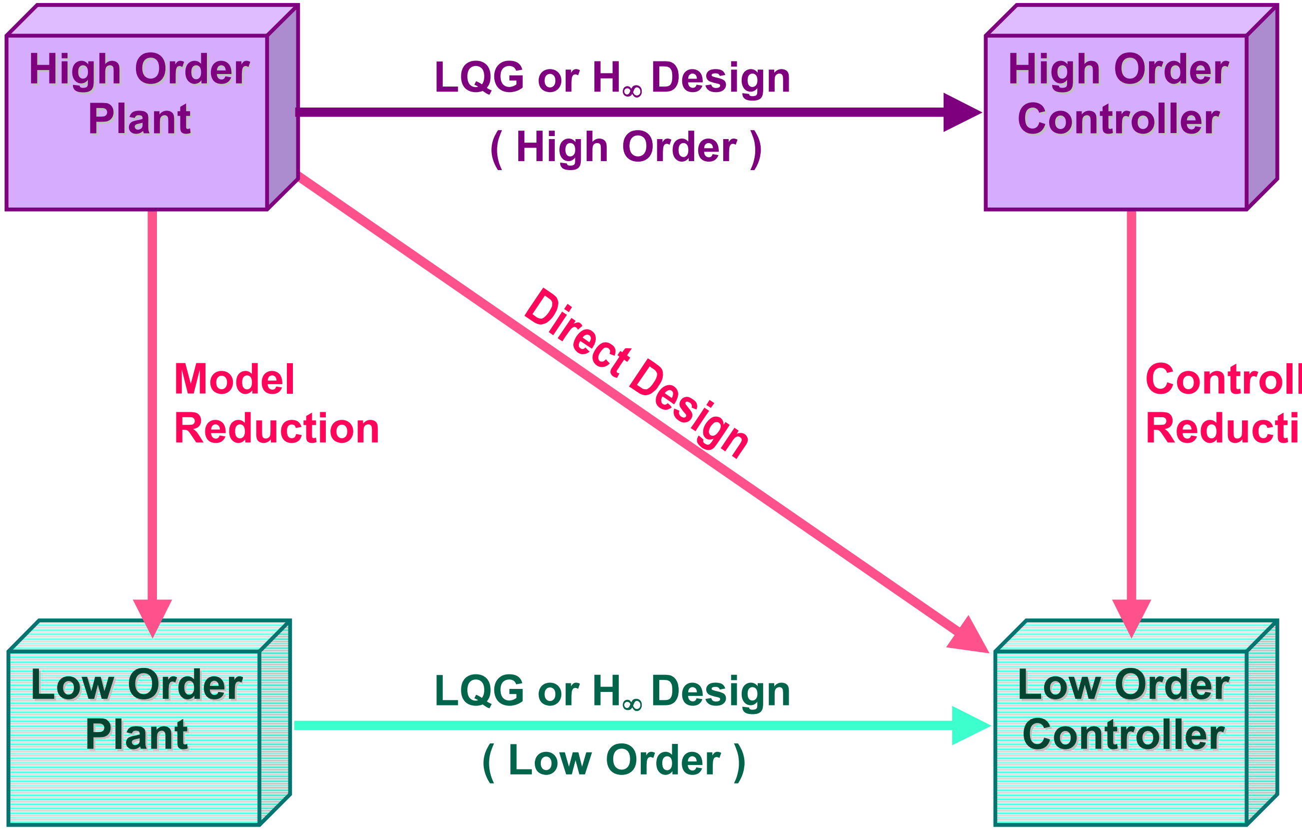
- Model reduction problem: Given a high order stable plant model G, and anti-stable weighting-matrices W_o and W_i, determine a reduced order model G_r such that

$$\|W_o(G - G_r)W_i\| = \min$$

- Computational approach:
  - compute G_1, the Hankel-norm approximation of the stable projection of $W_o G W_i$
  - compute G_r, the stable projection of $W_o^{-1} G_1 W_i^{-1}$.

No nice error bounds are known !!

Basic approaches to low order controller design



Controller reduction problem

- Controller reduction problem: Given a plant G and a stabilizing controller K, determine a reduced order controller K_r, such that the closed-loop system is stable and closed-loop performances are preserved.

- Specific aspects:
  - controllers often unstable
  - information on plant can be used
  - controller reduction problems highly structured
  - closed-loop stability/performance preserving necessary

Controller reduction approaches

- Stability/performance preserving reduction using frequency-weighted balancing techniques:
  - special methods for general and state feedback-observer based controllers
- Coprime factorization based reduction:
  - direct reduction of coprime factors
  - frequency-weighted balanced truncation of coprime factors

Controller reduction using frequency-weighted balanced truncation

- Stability enforcing one-sided weights (Anderson & Liu, 1989):

$$W_o = (I + GK)^{-1}G, \quad W_i = I \;\text{ or }\; W_o = I, \quad W_i = G(I + KG)^{-1}$$

- Stability and performance enforcing weights (Anderson & Liu, 1989):

$$W_o = (I + GK)^{-1}G, \quad W_i = (I + GK)^{-1}$$

- Solve the frequency-weighted balanced truncation approximation problem

$$\|W_o(G - G_r)W_i\| = \min$$

Square-root methods: Varga & Anderson (2002,2003)

Controller reduction using coprime factorization techniques

- apply coprime factorization model reduction to K by exploiting the special structure of state feedback-observer based controllers
  Anderson & Liu (1989); Liu, Anderson & Ly (1990)

- stability preserving coprime factor reduction
  basic approach: Zhou, Doyle & Glover (1996)
  efficient square-root methods: Varga (ACC'2003)

- performance preserving reduction of $\mathcal{H}_\infty$ controllers
  basic approach: Goddard & Glover (1999)
  efficient square-root methods: Varga (ECC'2003)

Software for model/controller reduction

Software	SLICOT	Control Toolbox	Robust Toolbox	μ-Toolbox	MATRIX$_X$	WOR–Toolbox
Provided features						
continuous-time	+	+	+	+	+	+
discrete-time	+	+	−	−	−	−
unstable	+	−	+	−	−	+
non-minimal	+	−	+	+	+	+
Methods						
balancing	+	+	+	+	+	+
balancing-free (BF)	+	−	+	−	+	−
square-root (SR)	+	−	−	+	−	+
BF-SR	+	−	−	−	−	−
Problem classes						
additive error	+	+	+	+	+	+
relative error	+	−	+	+	+	−
frequency weighted	+	−	−	+	+	+
controller reduction	+	−	−	−	+	+

Model reduction software in SLICOT

- reduction of stable models

Name	Function
AB09AD	Balanced truncation approximation (BTA)
AB09BD	Singular perturbation approximation (SPA)
AB09CD	Hankel norm approximation (HNA)
AB09DD	Singular perturbation approximation formulas

- reduction of unstable models

Name	Function	mex-function	m-function
AB09ED	HNA for the stable part	sysred	hna
AB09FD	BTA of coprime factors	sysred	bta_cf
AB09GD	SPA of coprime factors	sysred	spa_cf
AB09MD	BTA for the stable part	sysred	bta, btabal
AB09ND	SPA for the stable part	sysred	spa, spabal
AB09HD	BST with BTA or SPA	bstred	bst
AB09ID	frequency-weighted BTA or SPA	fwered	fwbred
AB09JD	frequency-weighted HNA	fwehna	fwhna

Controller reduction software in SLICOT

- reduction of general controllers

Name	Function	mex-function	m-function
SB16AD	frequency-weighted BTA or SPA	`conred`	`fwbconred`

- reduction of state-feedback-observer based controllers

Name	Function	mex-function	m-function
SB16BD	coprime factor reduction with BTA or SPA	`sfored`	`cfconred`
SB16CD	frequency-weighted BTA of coprime factors	`sfored`	`cfconred`

SLICOT - Matlab comparison

`sysred` – SLICOT based mex-function (square-root BTA)
`sqrmr` – plain MATLAB implementation of the square-root BTA
`balreal` – MATLAB Control Toolbox (balancing method)

Order	Times [sec]		
	sysred	sqrmr	balreal
16	0.003	0.17	0.04
32	0.01	0.5	0.17
64	0.11	2.14	failed
128	0.78	10.55	failed
256	6.12	63.75	failed
512	76.23	478.69	failed

Timing results for a Pentium II 400 MHz PC

Advanced Technologies Testing Aircraft System: ATTAS

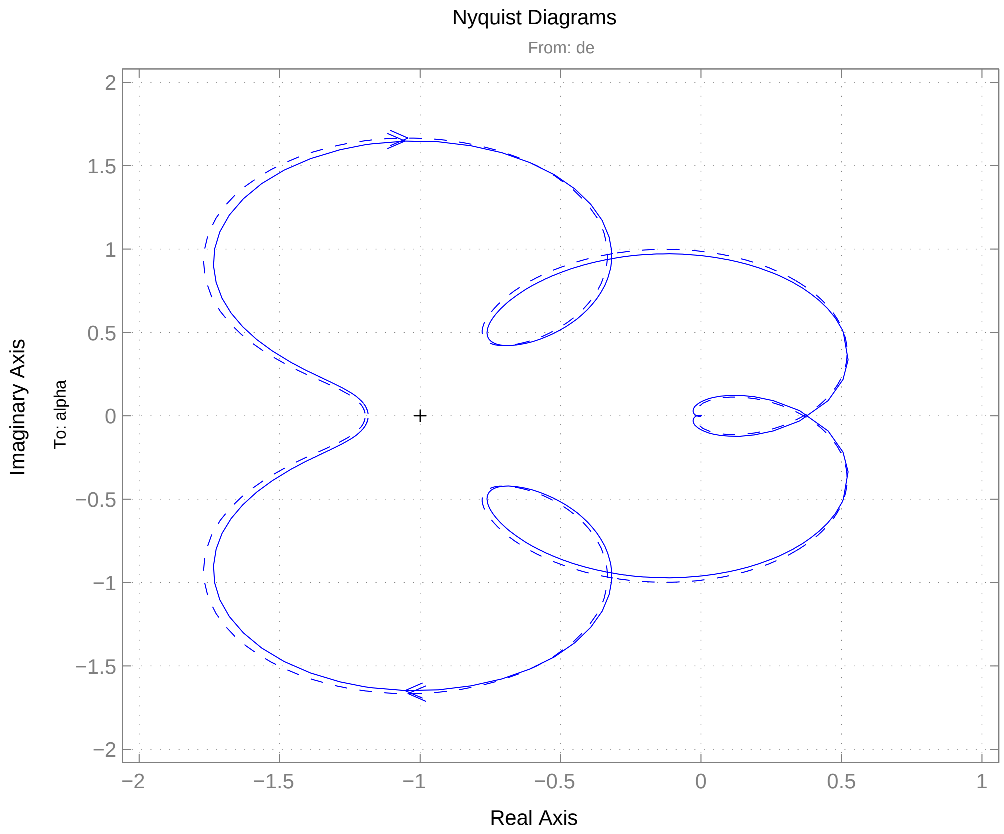
Original linearized aircraft model: $n = 55$, $m = 9$, $p = 9$

Characteristics: continuous-time, non-minimal, unstable

Reduced models obtained with BTA:

global dynamics	$r = 15, \quad m = 9, \quad p = 9$
longitudinal dynamics	$r = 7, \quad m = 4, \quad p = 4$
lateral dynamics	$r = 10, \quad m = 2, \quad p = 5$

Comparison of frequency responses for element $g_{22}(s)$ of ATTAS

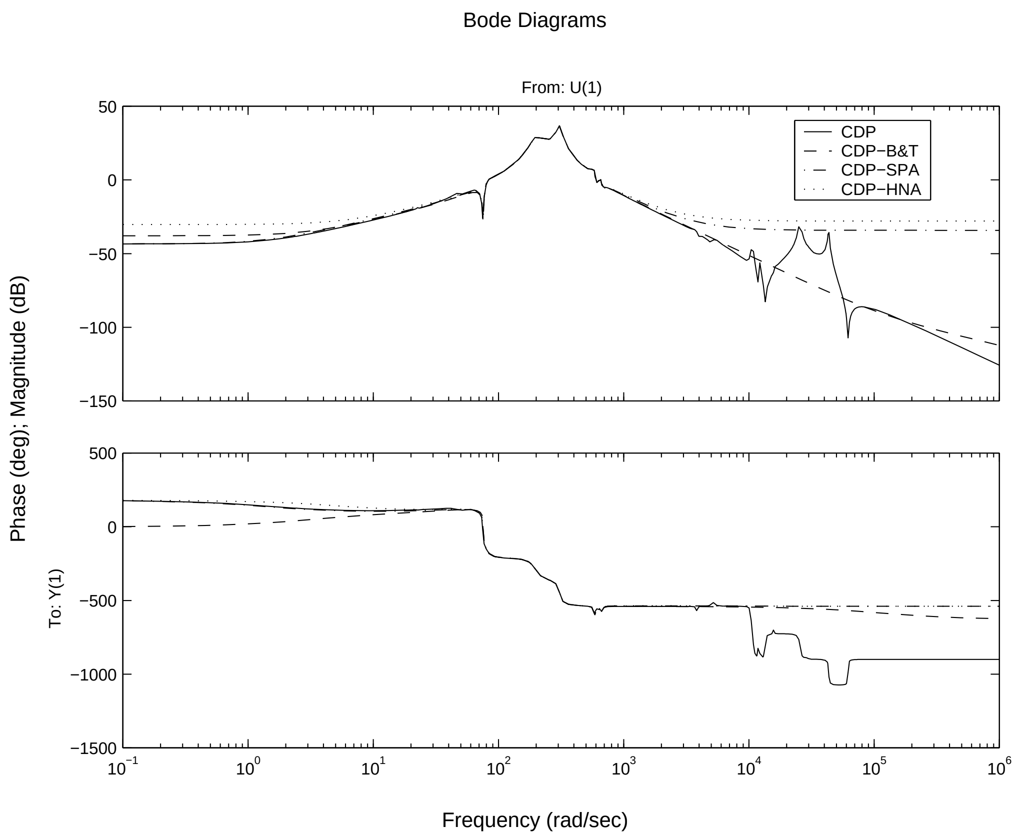


CD-player finite element model

Original CD-player model:
$n = 120,\ m = 1,\ p = 1$

Characteristics:
continuous-time, stable

Reduced models obtained
with BTA, SPA, HNA: $r = 10$

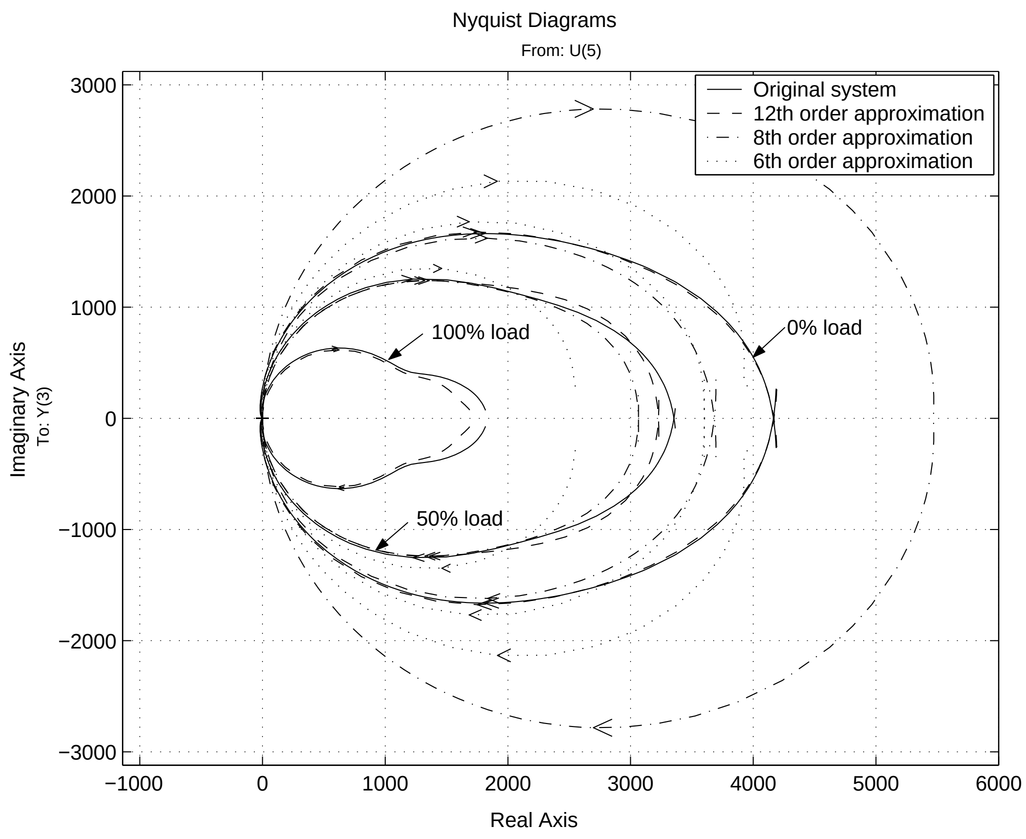


Linearized gasifier models

Gasifier models linearized at 0%, 50%, 100% loads:
$n = 25$, $m = 6$, $p = 5$

Characteristics: continuous-time, stable, non-minimal, badly scaled

Reduced models with BTA:
$r = 6, 8, 12$



Final remarks

- The best numerical software for model reduction is available for free in SLICOT!
- MATLAB/Scilab mex- and m-functions offer flexible interfaces to model reduction software in SLICOT.
- Special software for controller reduction available.
- Software for reduction of very high order systems using parallel computations also available!

Further Reading

[1] K. Zhou, J. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice-Hall, Upper Saddle River, NJ, 1996.

[2] G. Obinata and B. Anderson, *Model Reduction for Control System Design*, Communications and Control Engineering Series, Springer-Verlag, London, UK, 2001.

[3] A. Varga, *Model reduction software in the SLICOT library*, in Applied and Computational Control, Signals, and Circuits, B. Datta, ed., vol. 629 of The Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, Boston, MA, 2001, pp. 239–282.[3]

[4] A. Varga, *New numerical software for model and controller reduction*, SLICOT Working Note SLWN2002-5, 2002.[4]

[3] http://www.robotic.dlr.de/control/publications/2001/varga_ACCSC01.pdf

[4] http://www.robotic.dlr.de/control/publications/2002/varga_SLWN2002-5.pdf

Robust Control Design Using H_∞ Methods

Da-Wei Gu, Petko Petkov & Mihail Konstantinov
Control & Instrumentation Group
Department of Engineering
University of Leicester (UK)

E-mail: dag@leicester.ac.uk
URL: http://www.le.ac.uk/engineering/dag

Abstract

As control systems are vulnerable to external perturbations and measurement noise, robust control design methods aim at computing controllers that stabilize the given plant and guarantee certain performance levels in the presence of disturbance signals, noise interference, unmodeled plant dynamics, and model uncertainties. H_∞-optimization has become a standard method in this area, but its implementation in a CACSD environment is challenging due to several subtle numerical aspects. In this talk, reliable tools for H_∞-optimization, H_∞-loop shaping design, and μ-synthesis, available in SLICOT based toolboxes for MATLAB and Scilab, will be discussed. A mass-damper-spring system will serve as case study to exemplify the steps taken in a typical robust control design.

Outline

- A Mass-Damper-Spring System
- Modeling of Uncertainties
- Robust Designs of MDS System
- Robust Design Routines in SLICOT
- Results and Conclusions

Mass-Damper-Spring System

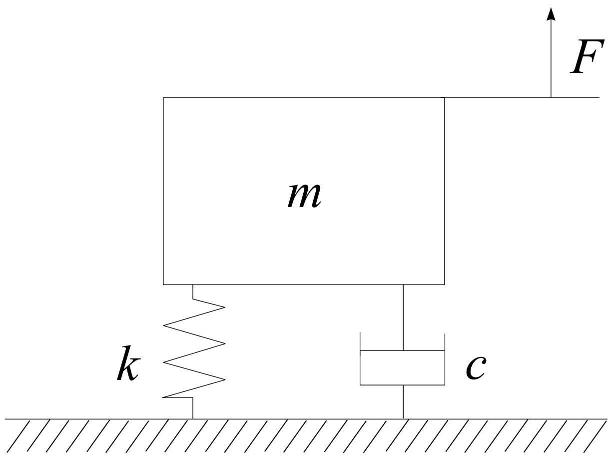


Figure 2: Mass-Damper-Spring System

The dynamics:

$$m\ddot{x} + c\dot{x} + kx = u,$$

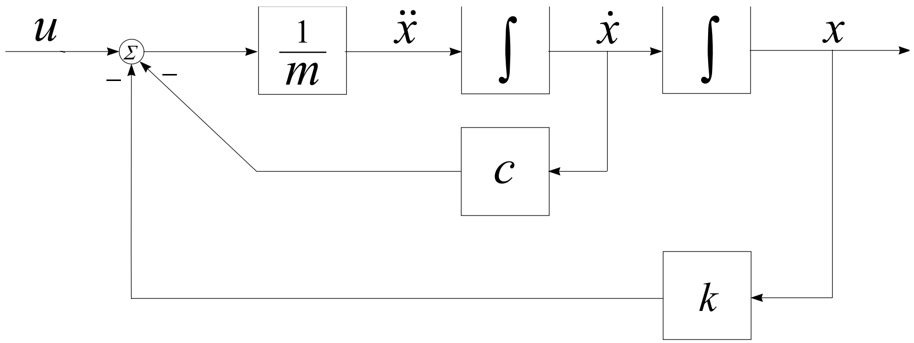


Figure 3: Block Diagram of MDS System

x**:** displacement of the mass block

$u = F$**:** force

m**:** mass

c**:** damper constant

k**:** spring constant

Modeling of Uncertainties

The coefficients are NOT exactly known, hence assume the parametric uncertainties

$$m = \overline{m}(1 + p_m\delta_m), \;\; c = \overline{c}(1 + p_c\delta_c), \;\; k = \overline{k}(1 + p_k\delta_k)$$

with nominal values

$$\overline{m} = 3, \;\; \overline{c} = 1, \;\; \overline{k} = 2$$

Relative perturbations:

$$p_m = 0.4, \;\; p_c = 0.2, \;\; p_k = 0.3$$

and

$$-1 \leq \delta_m, \;\; \delta_c, \;\; \delta_k \leq 1$$

(40% uncertainty in the mass, 20% in the damping coefficient and 30% in the spring constant)

Representations of uncertainties in LFTs

$$\begin{aligned}\frac{1}{m} &= \frac{1}{\overline{m}(1+p_m\delta_m)} = \frac{1}{\overline{m}} - \frac{p_m}{\overline{m}}\delta_m(1+p_m\delta_m)^{-1} \\ &= F_U(M_{mi}, \delta_m)\end{aligned}$$

with

$$M_{mi} = \begin{bmatrix} -p_m & \frac{1}{\overline{m}} \\ -p_m & \frac{1}{\overline{m}} \end{bmatrix}.$$

Similarly,

$$c = F_U(M_c, \delta_c),\ \ k = F_U(M_k, \delta_k)$$

with

$$M_c = \begin{bmatrix} 0 & \overline{c} \\ p_c & \overline{c} \end{bmatrix}, \quad M_k = \begin{bmatrix} 0 & \overline{k} \\ p_k & \overline{k} \end{bmatrix}.$$

Block Diagrams

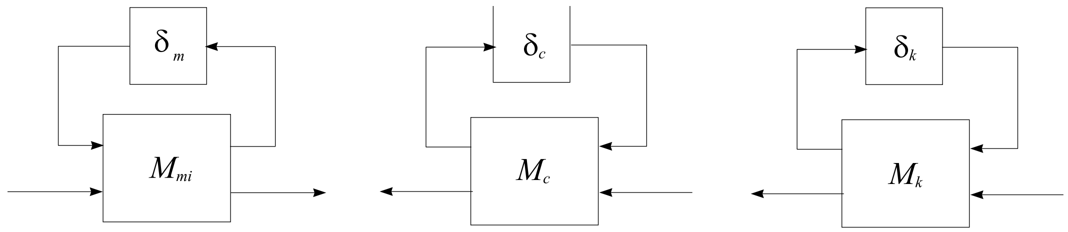


Figure 4: Uncertain Parameters in LFTs

and

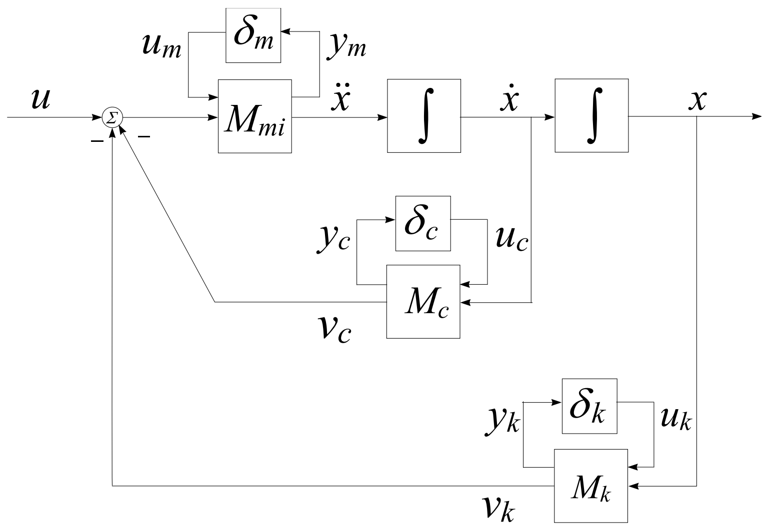


Figure 5: MDS with Uncertain Parameters

Further, define the nominal system G_{mds} with consideration of parametric uncertainties.

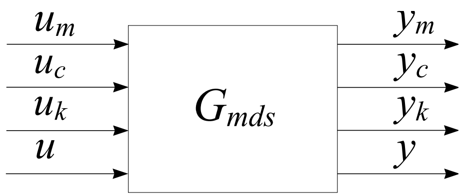


Figure 6: Input/Output Block Diagram of MDS System

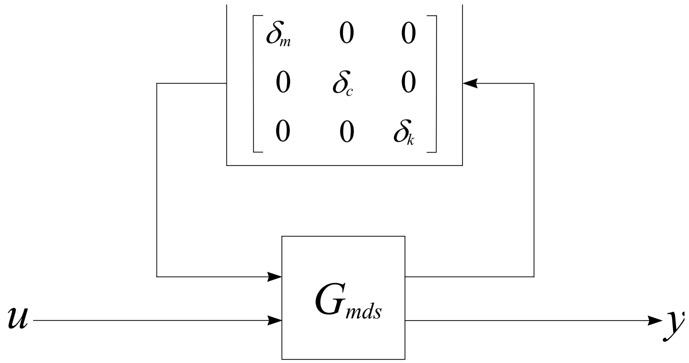


Figure 7: LFT Representation of MDS System with Uncertainties

Designs of Robust Controllers

Design Objectives:

1. Robust Stability
2. Robust Performance:

$$\left\| \begin{bmatrix} W_p(I+GK)^{-1} \\ W_uK(I+GK)^{-1} \end{bmatrix} \right\|_\infty < 1$$

for all $G = F_U(G_{mds}, \Delta)$

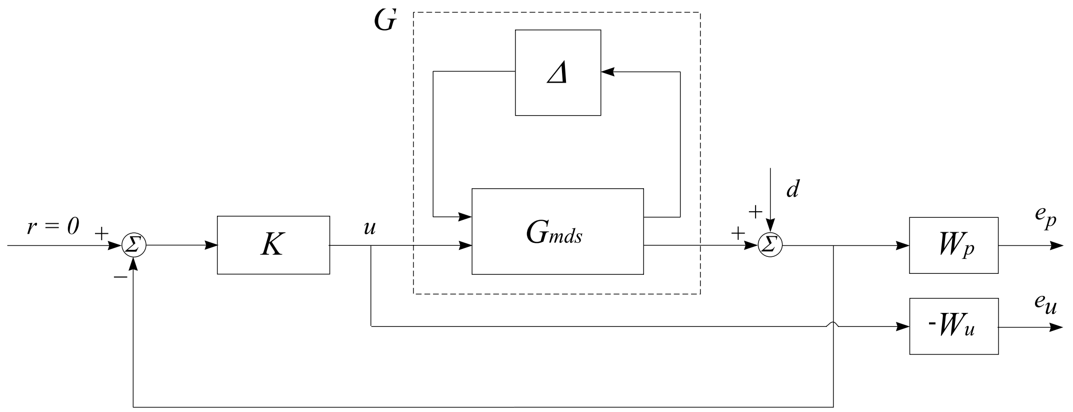


Figure 8: Closed-Loop System Structure

3 designs tried

- Sub-Optimal $\mathcal{H}_\infty$ (S over KS) Design (K_{hin})
- $\mathcal{H}_\infty$ Loop Shaping Design Procedure (LSDP) (K_{lsh})
- μ-Synthesis (D-K Iterations) (K_{mu})

Weighting functions selected:

$$w_p(s) = 0.95\frac{s^2 + 1.8s + 10}{s^2 + 8.0s + 0.01}, \quad w_u = 10^{-2}$$

and, in $\mathcal{H}_\infty$ LSDP,

$$W_1(s) = 2\frac{8s + 1}{0.9}, \quad W_2(s) = 1$$

Robust Design Routines in SLICOT

Routines available for $\mathcal{H}_2$, $\mathcal{H}_\infty$, $\mathcal{H}_\infty$ LSDP and μ-analysis and synthesis.

$\mathcal{H}_2$ design for continuous-time systems

Routine	Functionality
SB10HD	Design of optimal $\mathcal{H}_2$ output controllers (main subroutine)
SB10UD	Transformation of system matrices to standard form
SB10VD	Computation of the state feedback and output injection matrices of the optimal $\mathcal{H}_2$ regulator
SB10WD	Computation of the $\mathcal{H}_2$ optimal controller
AB13BD	Computation of the $\mathcal{H}_2$ or $\mathcal{L}_2$ norm of continuous- and discrete-time systems

$\mathcal{H}_\infty$ design for continuous-time systems

Routine	Functionality
SB10FD	Design of suboptimal $\mathcal{H}_\infty$ output controllers (main subroutine)
SB10PD	Transformation of system matrices to standard form
SB10QD	Computation of the state feedback and output injection matrices of the suboptimal $\mathcal{H}_\infty$ regulator
SB10RD	Computation of the suboptimal controller
SB10LD	Computation of the closed-loop system matrices
AB13DD	Computation of the $\mathcal{H}_\infty/\mathcal{L}_\infty$ norm of continuous- and discrete-time systems

$\mathcal{H}_\infty$ and $\mathcal{H}_2$ synthesis routines for discrete-time systems

Routine	Functionality
SB10DD	Design of $\mathcal{H}_\infty$ suboptimal output controllers (main routine)
SB10ED	Design of optimal $\mathcal{H}_2$ output controllers (main routine)
SB10SD	Computation of the $\mathcal{H}_2$ controller for the normalized system
SB10TD	Computation of the $\mathcal{H}_2$ controller for the original system

Routine for $\mathcal{H}_\infty$ LSDP and μ computation

Routine	Functionality
SB10ID	Loop shaping design of output controllers for continuous-time cases (main subroutine)
SB10JD	Transformation of a descriptor system into regular form
SB10KD	Loop shaping design of output controllers for discrete-time cases (main subroutine)
AB13MD	Computation of upper bound on the structured singular value

Other related routines, including matrix algebraic Riccati and Lyapunov equation solvers with condition and accuracy estimates

Routine	Functionality
SB01DD	Pole and eigenstructure assignment of a multi-input system
SB02QD	Estimation of the condition number of a continuous-time Riccati equation and estimation of the forward error
SB02RD	Solution of the continuous- or discrete-time matrix Riccati equation with condition and forward error estimation
SB02PD	Solution of the continuous-time matrix Riccati equation by the matrix sign function method
SB02OD	Solution of the continuous- or discrete-time algebraic Riccati equation

Other related routines (Cont.)

Routine	Functionality
SB02SD	Estimation of the condition number of a discrete-time Riccati equation and estimation of the forward error
SB03QD	Estimation of the condition number of a continuous-time Lyapunov equation and estimation of the forward error
SB03RD	Solution of the continuous-time matrix Lyapunov equation with condition and forward error estimation
SB03SD	Estimation of the condition number of a discrete-time Lyapunov equation and estimation of the forward error
SB03PD	Solution of the discrete-time matrix Lyapunov equation with condition and forward error estimation

- **SLICOT** routines used in all 3 robust designs and model reduction
- Mex files available, well integrated in MATLAB environment
- Good numerical performance in comparison to those in MATLAB
- Numerical accuracy and perturbation error estimations available in **SLICOT**

Design Results & Comparisons

1. $\mathcal{H}_\infty$ Controller: 4^{th} order, minimum γ achieved 0.9506

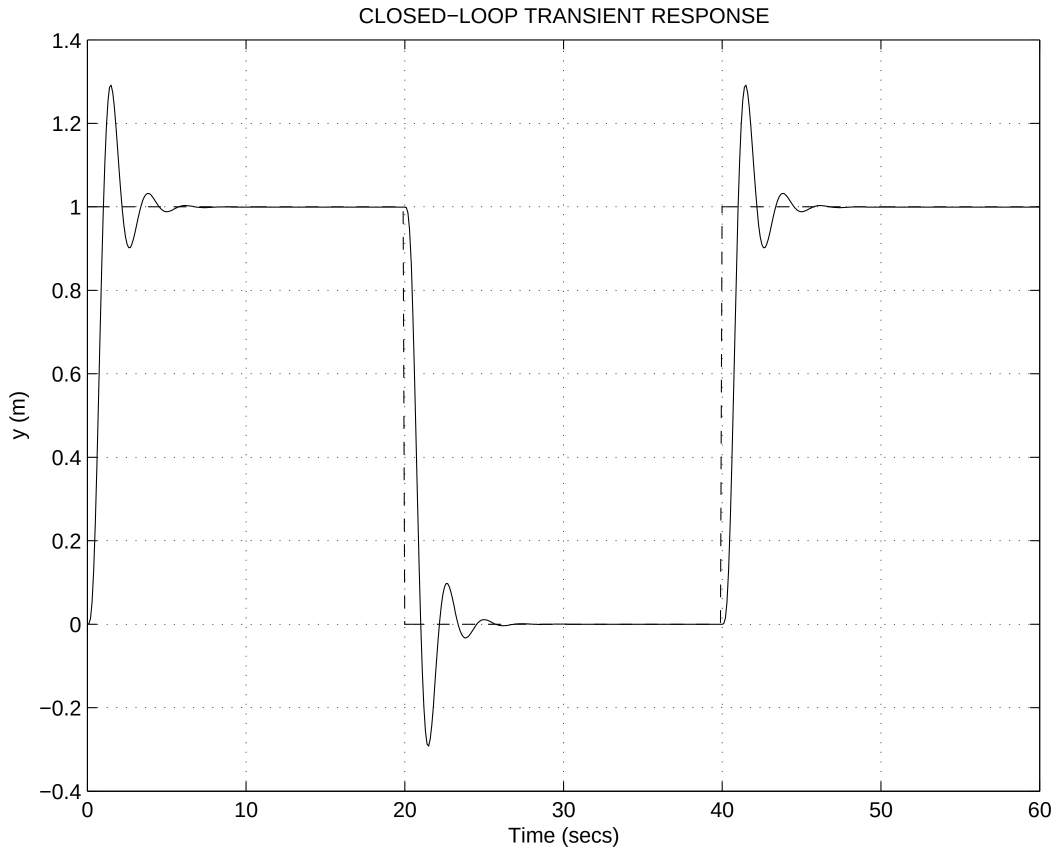


Figure 9: Transient Response to Reference Input (K_{hin})

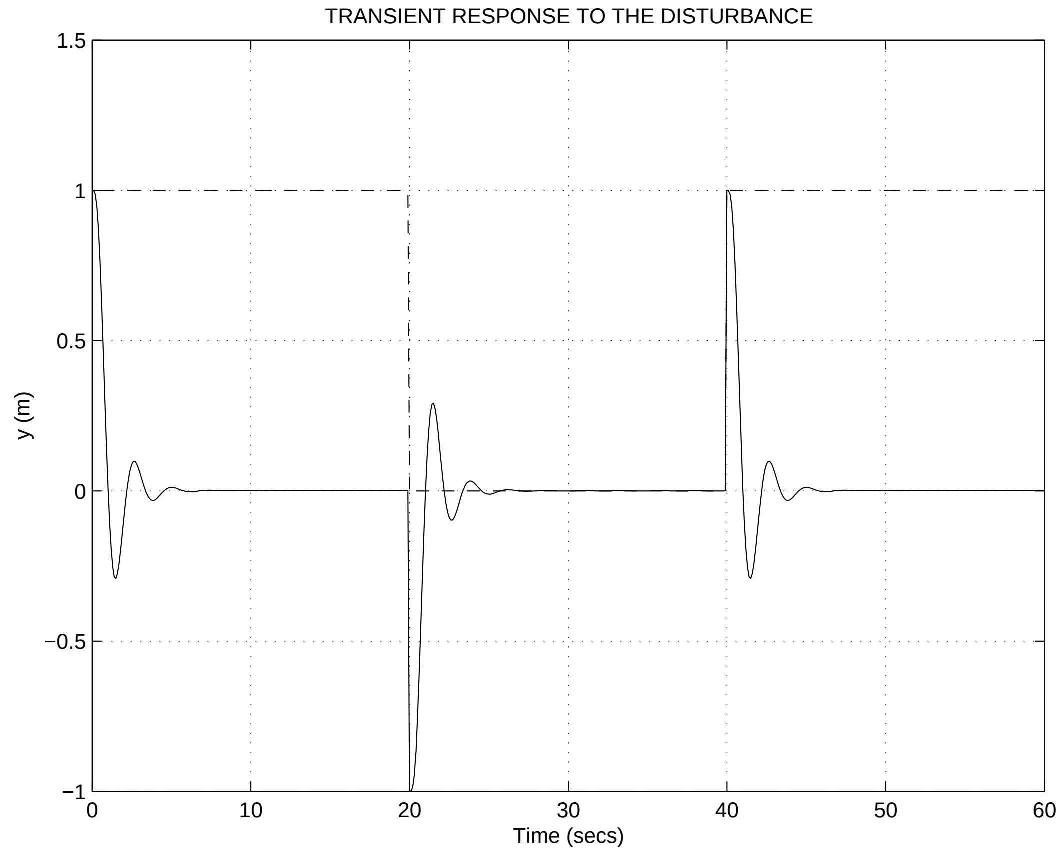


Figure 10: Transient Response to Disturbance Input (K_{hin})

2. $\mathcal{H}_\infty$ LSDP Controller: 4^{th} order, $\gamma = 0.395$,

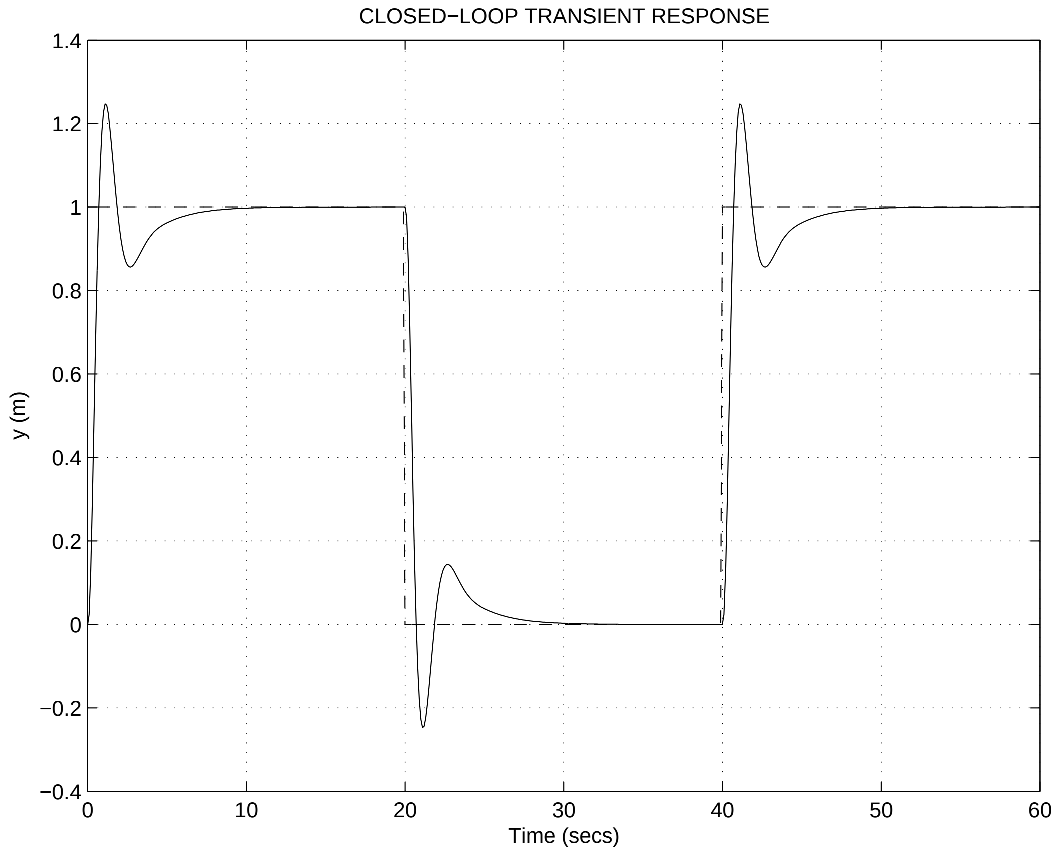


Figure 11: Transient Response to Reference Input (K_{lsh})

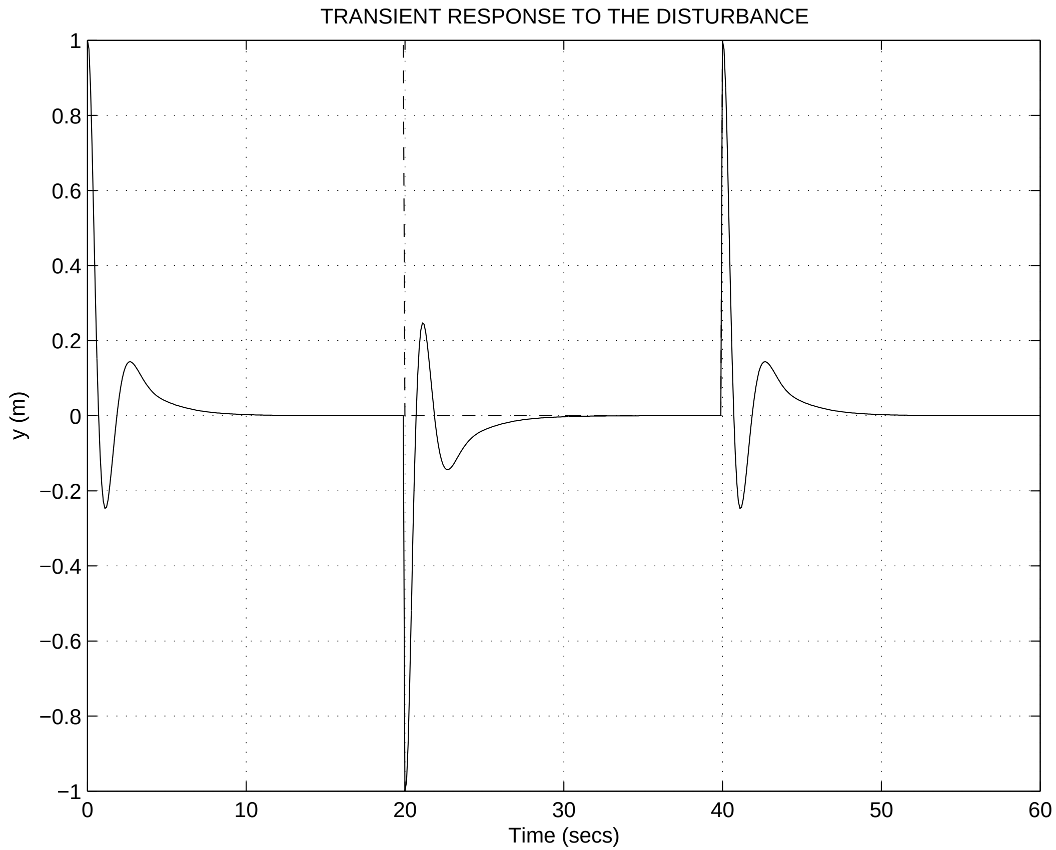


Figure 12: Transient Response to Disturbance Input (K_{lsh})

3. μ Controller: $\mu = 0.965$ after 4 iterations, original order 20 and reduced to 4

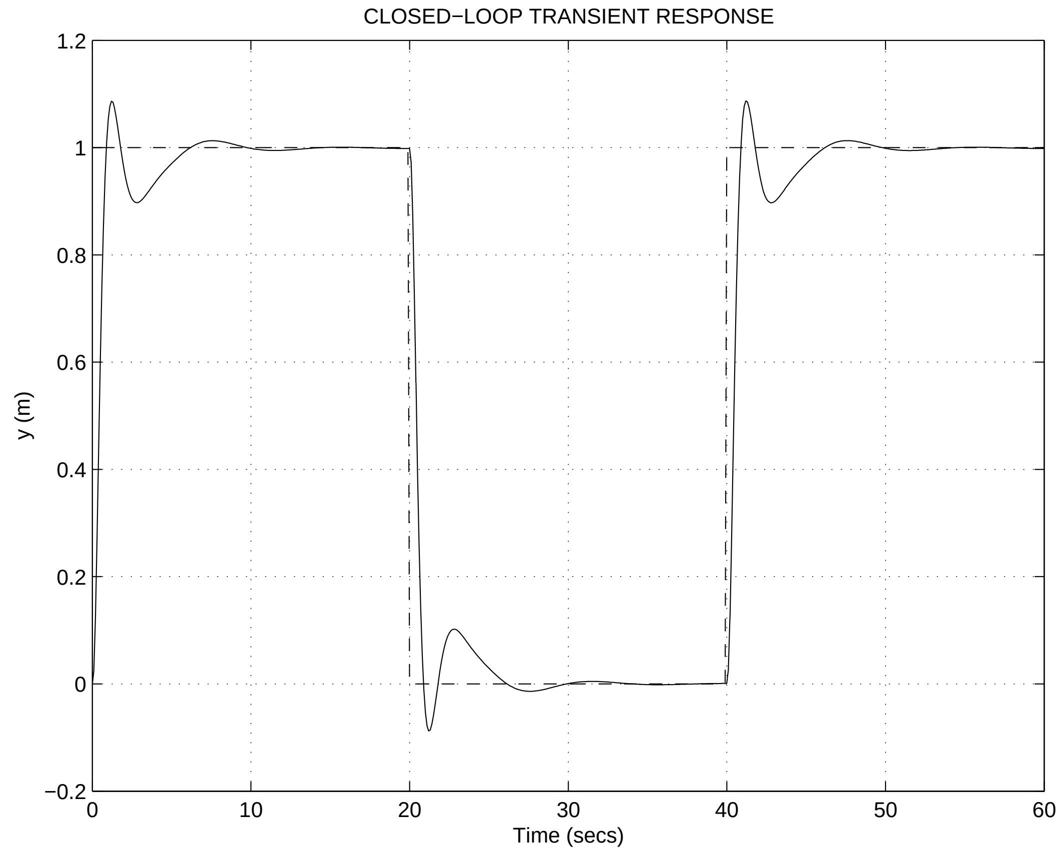


Figure 13: Transient Response to Reference Input (K_{mu})

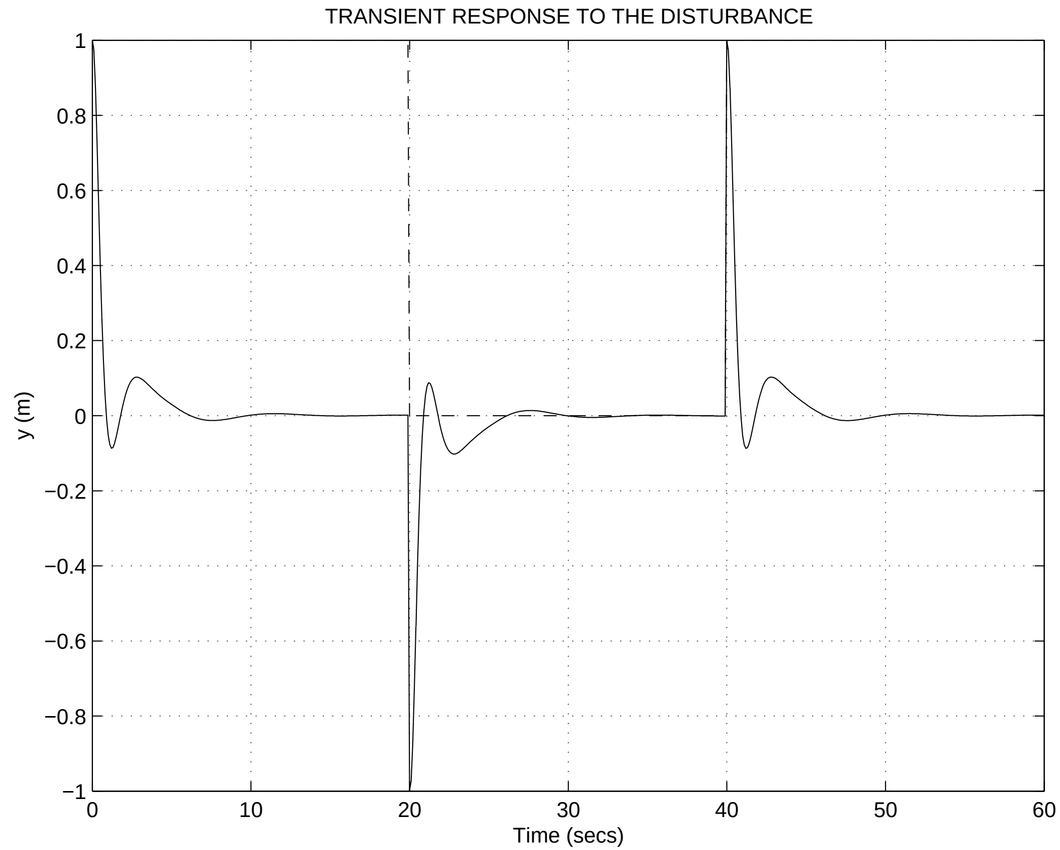


Figure 14: Transient Response to Disturbance Input (K_{mu})

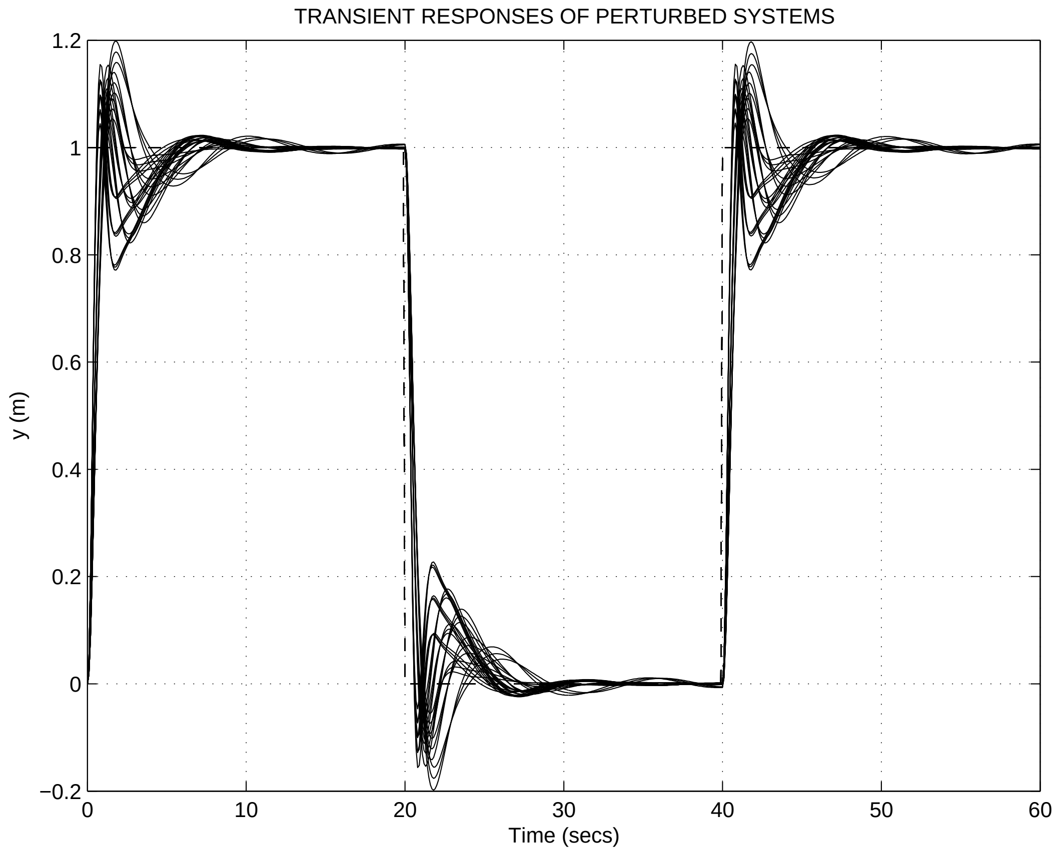


Figure 15: Transient Responses of Perturbed Systems (K_{mu})

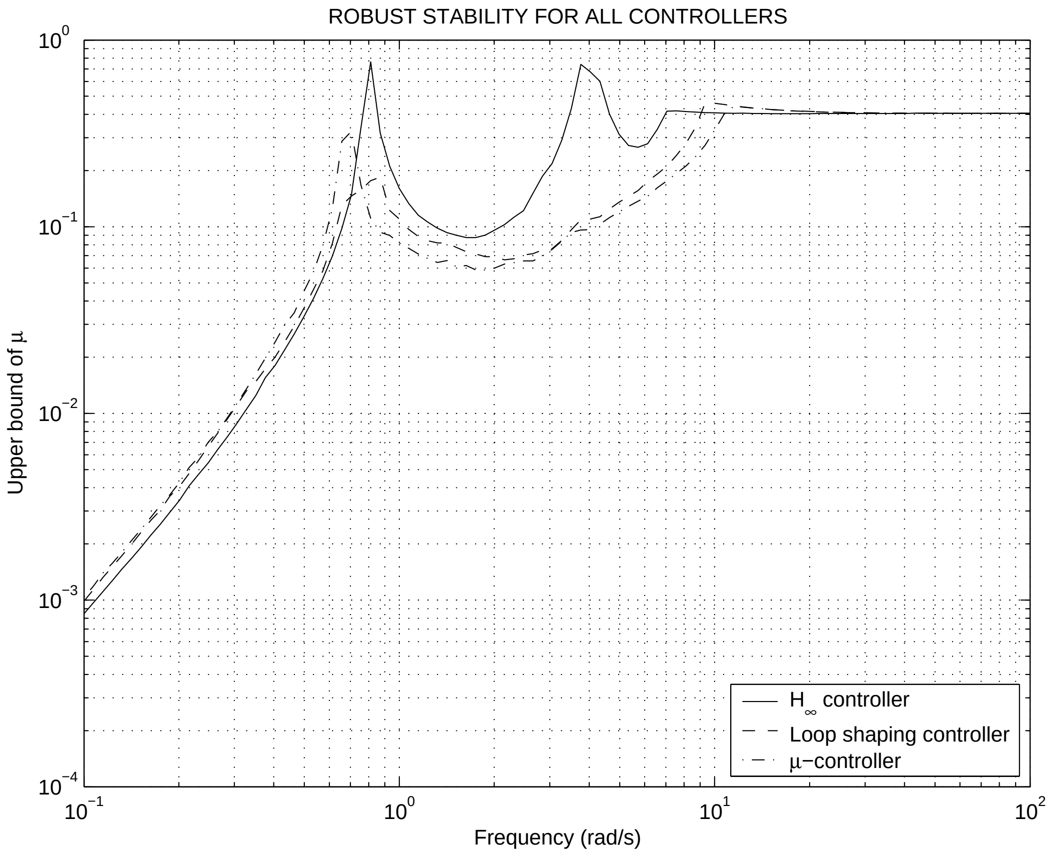


Figure 16: Comparison of Robust Stability for 3 Controllers

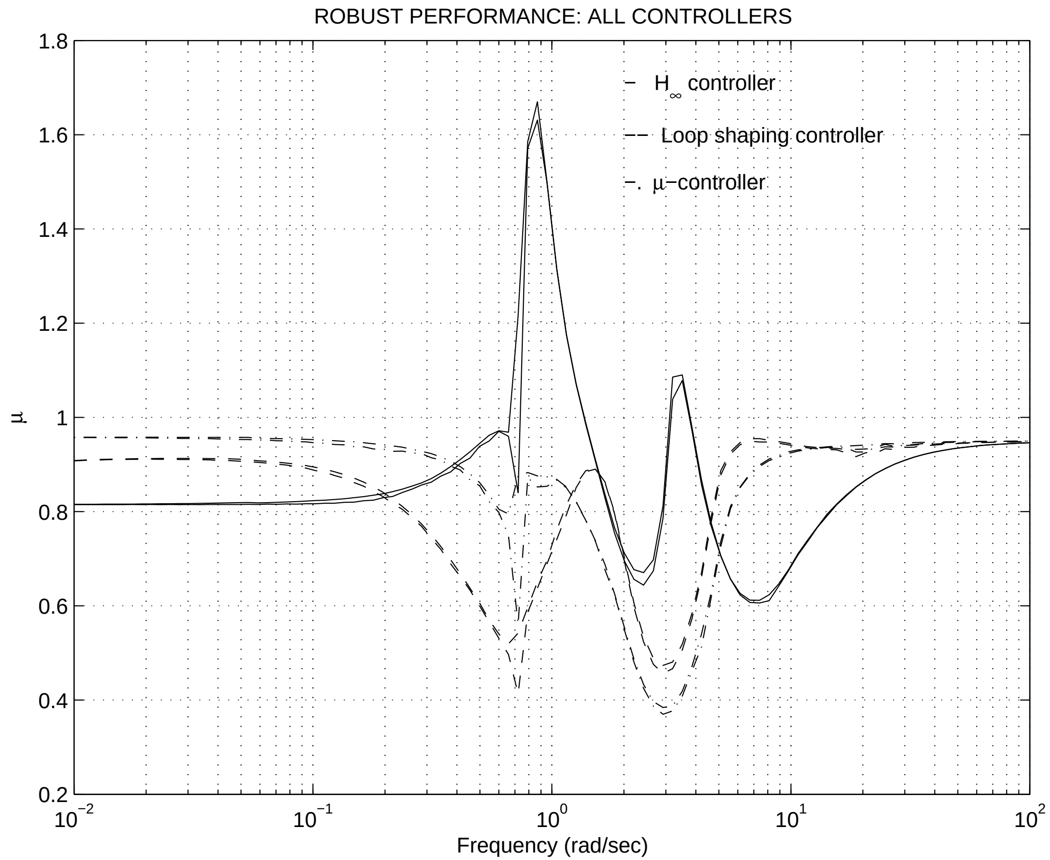


Figure 17: Comparison of Robust Performance for 3 Controllers

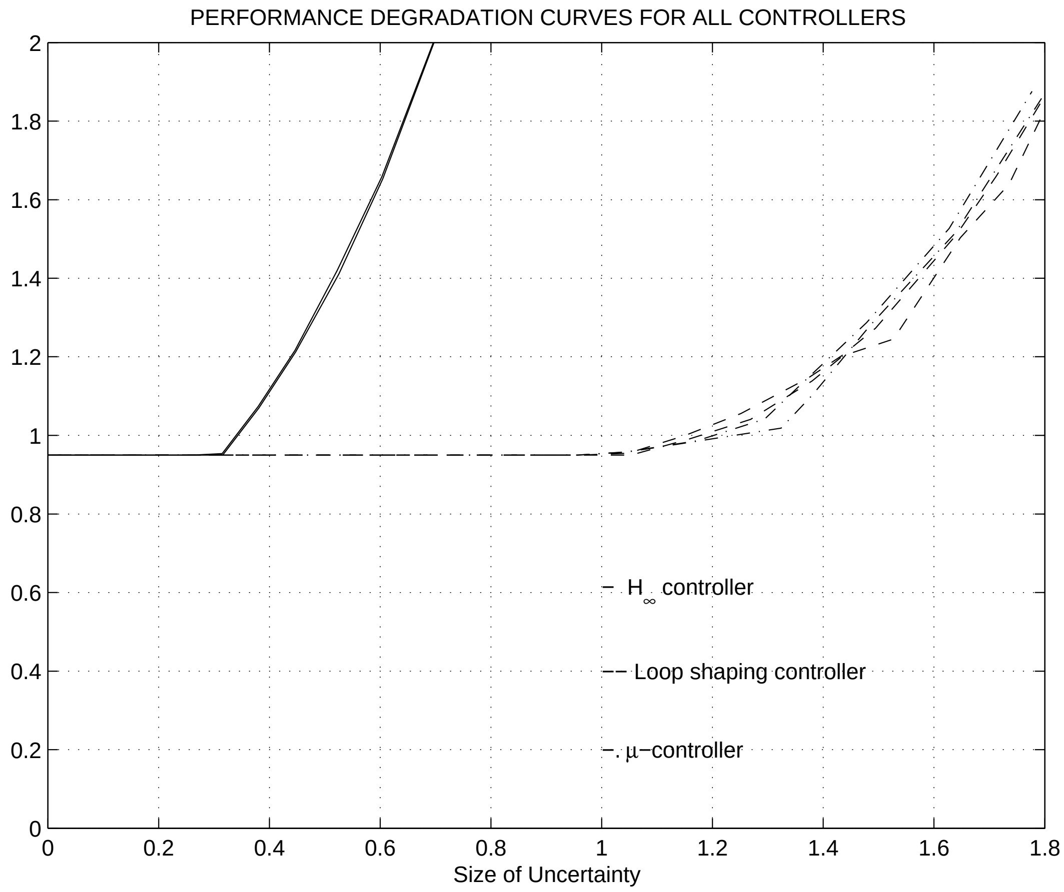


Figure 18: Performance Degradation for 3 Controllers

Model Reduction of μ Controller

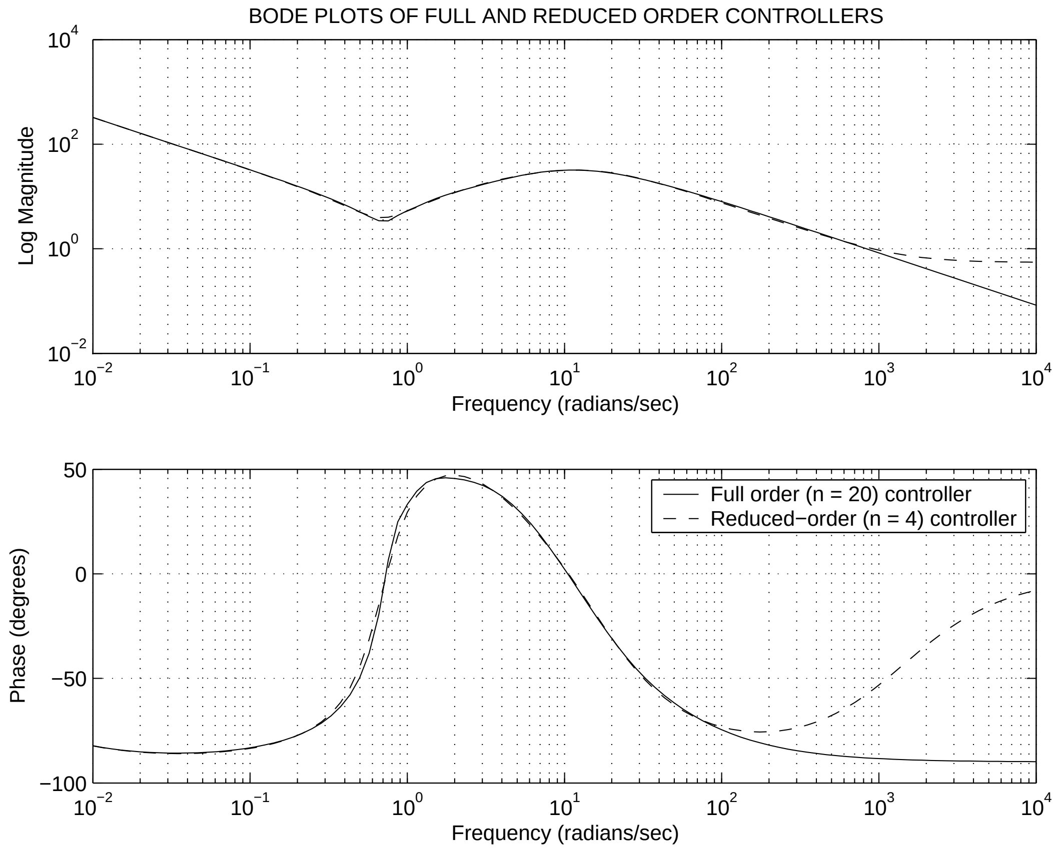


Figure 19: Frequency Responses of Full and Reduced Order Controllers

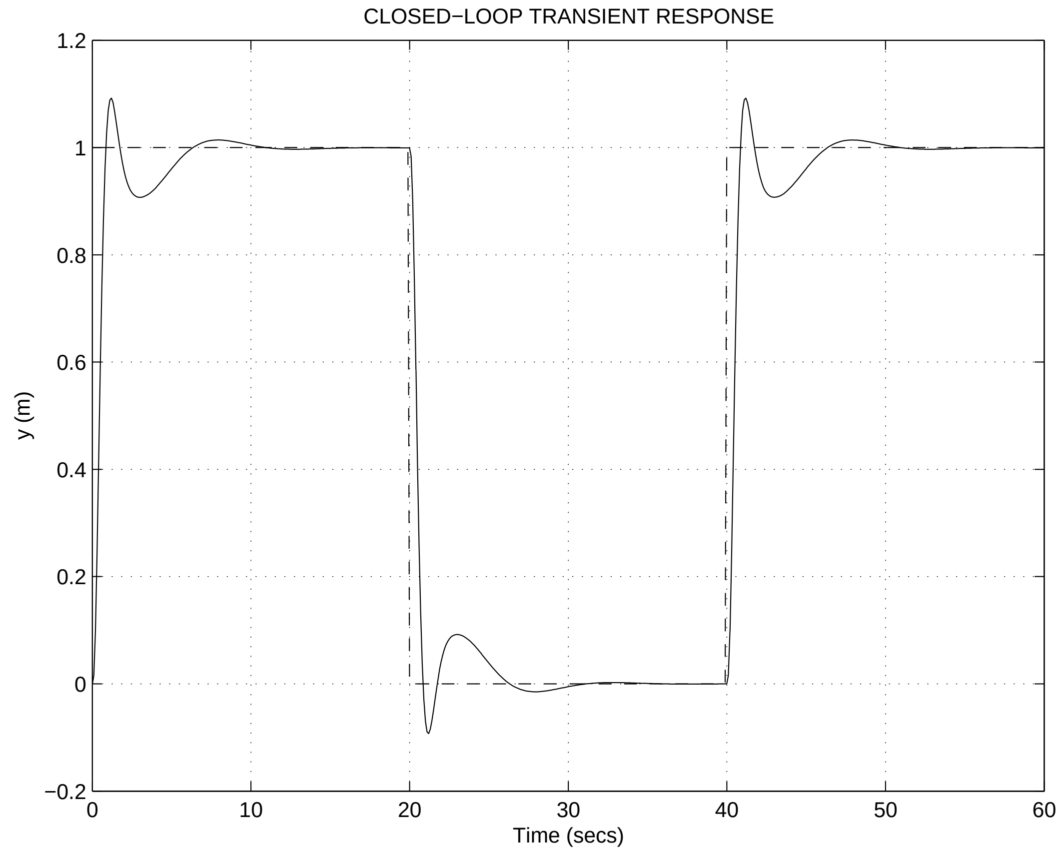


Figure 20: Transient Responses of Full and Reduced Order Controllers

Further Reading

[1] D.-W. Gu, P.Hr. Petkov, and M.M. Konstantinov, *An Introduction to $\mathcal{H}_\infty$ Optimisations Designs*, Niconet Report NIC1999-4, 1999.

[2] D.-W. Gu, P.Hr. Petkov, and M.M. Konstantinov, *$\mathcal{H}_\infty$ and $\mathcal{H}_2$ optimization toolbox in SLICOT*, SLICOT Working Note SLWN1999-12, 1999.

[3] D.-W. Gu, P.Hr. Petkov, and M.M. Konstantinov, *$\mathcal{H}_\infty$ Loop Shaping Design Procedure Routines in SLICOT*, Niconet Report MIC1999-15, 1999.

[4] G.J. Balas, J.C. Doyle, K. Glover, A. Packard, and R. Smith, *μ-Analysis and Synthesis Toolbox: User's Guide*, MUSYN Inc. and The Mathworks, Inc., 1995.

[5] K. Zhou, J. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice-Hall, Upper Saddle River, NJ, 1996.