Proceedings of the
47th IEEE Conference on Decision and Control
Cancun, Mexico, Dec. 9-11, 2008

Distributed Control: A Sequentially Semi-Separable Approach

Justin K. Rice, Michel Verhaegen

***Abstract*— We consider the problem of designing controllers for spatially-varying interconnected systems distributed in one spatial dimension. The matrix structure of such systems can be exploited to allow fast analysis and design of centralized controllers with simple distributed implementations. Iterative algorithms are provided for stability analysis, H_2 analysis, and sub-optimal controller synthesis. For practical implementation of the algorithms, approximations can be used, and the computational efficiency and accuracy of the algorithms incorporating these approximations are demonstrated on an example.**

I. Introduction

The control of spatially distributed interconnected systems has been of great interest in practical applications involving discretized partial differential equations(PDE's), such as boundary layer and transition control in fluid mechanics [1][2], flexible structures [3], and also in networks of spatially discrete interconnected subsystems, such as highway traffic control[4] and vehicle platooning [5], building anti-earthquake systems[6], formation flight [7], large adaptive telescope mirrors [8], etc.

The challenge has been in the computational cost of designing effective controllers and the complexity of implementing them. For PDE's, when directly solving for the optimal control is not viable, the system is often approximated as a large but finite number of coupled ODE's or interconnected subsystems. The system matrix describing the input-state-output behavior of N interconnected subsystems(ODE's), each of size(order) n, will be $nN \times nN$, and thus most matrix operations will be $\mathcal{O}(n^3N^3)$ floating point operations, making traditional robust or optimal controller design prohibitively expensive for fine discretizations or large numbers of discrete subsystems. Much research has been dedicated to surmounting this computational obstacle. In [9] and [10], multilevel techniques and the special matrix structure($\mathcal{H}$-matrix) have been exploited in iterative methods for finding fast ($\mathcal{O}(N^2)$, $\mathcal{O}(N\log(N))$) approximate solutions to Lyapunov and Riccati equations for systems governed by discretized PDE's. In [11] an efficient LMI method for spatially homogenous interconnected structures was developed, which was extended to finitely many heterogeneous subsystems in an array with boundary conditions [12] in $\mathcal{O}(n^{2\alpha}N^{\alpha})$(where $3.5 < \alpha < 5$). There has also been a conservative extension of the results of [11] to heterogeneous systems through robust synthesis and by treating the heterogeneity as norm bounded uncertainty[13]. In [14], it is shown that for infinite arrays of spatially invariant interconnected systems, a spatial Fourier transform may be used for solving optimal control problems, and that the optimal controller may then be well approximated using spatial truncation, allowing the designer to pick the connectivity of the controller. These results have been extended in [15] to show that the optimal controllers for infinite dimensional exponentially spatial decaying heterogeneous systems are also exponentially spatial decaying, but there are no results on efficiently calculating such controllers.

This research is supported by the Dutch Technology Foundation STW.

Justin K. Rice and Michel Verhaegen are with the Delft Center for Systems and Control, Delft University, 2628CD. email: J.K.Rice@TUDelft.nl

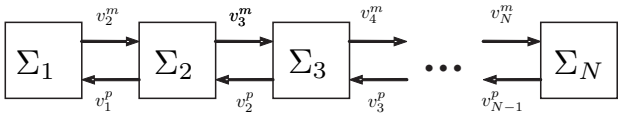

Fig. 1. String interconnection

In this paper we address the analysis and control of spatially varying systems on finite connected one-dimensional strings, such as those discussed in [12]. We approach the problem from a structured matrix algorithmic point of view; by revealing the Sequentially Semi-Separable(SSS) structure of the string connected system in Section II, we show that efficient($\mathcal{O}(n^3N)$) matrix operations are possible on such systems, and that given a controller with the same structure, direct controller distribution is possible. We then show in Section III that using iterative methods under which the SSS structure is closed, we can efficiently check matrix stability, and calculate solutions to Lyapunov and Riccati equations, thus producing controllers with SSS structure, also in $\mathcal{O}(n^3N)$. The *caveat emptor* is that the extreme computational efficiency will come at the cost of successive matrix approximations during the iterative methods to be discussed in Section IV. However, the stability and performance of the synthesized controller may be efficiently verified *a posteriori* in a manner robust to these approximations.

To allay fears brought on by such an iterative approximate approach, in Section V we will conduct an example of distributed H_2 control of a car platooning problem, to show the potential accuracy of such methods and the $\mathcal{O}(n^3N)$ complexity.

II. Subsystem model/Interconnection Structure

The subsystem models considered will most generally consist of state space realizations of the sort Σ_s:

$$\begin{bmatrix} \dot{x}_s \\ v_{s-1}^p \\ v_{s+1}^m \\ z_s \\ y_s \end{bmatrix} = \begin{bmatrix} A_s & B_s^p & B_s^m & B_s^1 & B_s^2 \\ C_s^p & W_s^p & 0 & L_s^p & V_s^p \\ C_s^m & 0 & W_s^m & L_s^m & V_s^m \\ C_s^1 & J_s^p & J_s^m & D_s^{11} & D_s^{12} \\ C_s^2 & H_s^p & H_s^m & D_s^{21} & D_s^{22} \end{bmatrix} \begin{bmatrix} x_s \\ v_s^p \\ v_s^m \\ w_s \\ u_s \end{bmatrix} \quad (2)$$

where v_s^m and v_s^p are interconnections to other subsystems(see Figure 1), z_s and w_s are performance outputs and

disturbance inputs, and y_s and u_s are measured outputs and controlled inputs. The W_s^* terms represent information feedthrough between subsystems Σ_{s+1} and Σ_{s-1}. A generalization of this subsystem has appeared in [12] and associated papers. We will generally allow each subsystem Σ_s to be arbitrarily different from every other subsystem, even having different state, input, and output dimensions, as long as the interconnections are of correct size. An example of such a subsystem model will be shown in Section V, and others are available in the literature, such as multiple vehicle systems [13], flight formations [7], offshore bases [16], and discretizations of various PDE's, [11], [3] etc.

If N of these subsystems(2) are connected together in a string (see Figure 1) with zero boundary inputs ($v_1^m = 0$,$v_N^p = 0$) and the interconnection variables are resolved, we obtain the interconnected system:

$$\overline{\Sigma}: \quad \begin{bmatrix} \dot{\overline{x}} \\ \overline{z} \\ \overline{y} \end{bmatrix} = \begin{bmatrix} \overline{A} & \overline{B}_1 & \overline{B}_2 \\ \overline{C}_1 & \overline{D}_{11} & \overline{D}_{12} \\ \overline{C}_2 & \overline{D}_{21} & \overline{D}_{22} \end{bmatrix} \begin{bmatrix} \overline{x} \\ \overline{w} \\ \overline{u} \end{bmatrix} \tag{3}$$

where the $\overline{\text{overline}}$ indicates a 'lifted' variable; for vectors: $\overline{x} = \begin{bmatrix} x_1^T & x_2^T & ... & x_N^T \end{bmatrix}^T$, and the interconnected system matrices $(\overline{A}, \overline{B}_1, \overline{B}_2, \overline{C}_1, \overline{D}_{11}, \overline{D}_{12}, \overline{C}_2, \overline{D}_{21}, \overline{D}_{22})$ have a very special structure, called 'Sequentially Semi Separable'(SSS). For example, for $N = 5$, we obtain equation (1) at the bottom of the page. Such matrices will be denoted:

$$\overline{A} = \mathcal{SSS}(B_s^m, W_s^m, C_s^m, A_s, B_s^p, W_s^p, C_s^p) \tag{4}$$

where the arguments of $\mathcal{SSS}()$ are called the 'generator' matrices of $\overline{A}$, and the $\mathcal{SSS}$ notation of the other matrices can be easily derived. This type of data-sparse structured matrix has recently been studied with respect to LTV systems theory and inversion [17], scattering theory [18], and for their own sake [19]. The facts in which we are interested are that SSS matrices can be stored using only a linear amount of memory, there exist algorithms of only linear computational complexity($\mathcal{O}(N)$) for SSS matrix-matrix addition and multiplication, and inversion, and further, that the class of SSS matrices is closed under these operations, that is, they are structure preserving. These properties(many of which are similar to those possessed by $\mathcal{H}$ matrices[10]) are especially important, since they allow the effective use of iterative algorithms incorporating inverses, in contrast to other classes of data sparse matrices(such as banded), which are not closed under inversion.

The consideration of such systems leads to the control design problem, extended to SSS distributed systems:

Problem 1 (Control Synthesis): Given $\overline{\Sigma}$, find a stabilizing controller $\overline{K}$ with SSS structure.

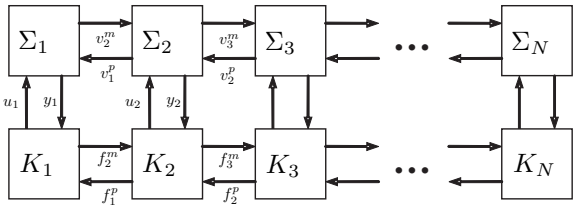

Fig. 2. Controller implementation

Because a distributed implementation of the controller is desired, not a centralized one, the controller $\overline{K}$ is constrained to have a realization of SSS matrices, as we will now explain.

Suppose that some method has been used to design a controller $\overline{K}$: $\left[\begin{array}{c|c} \overline{A}_K & \overline{B}_K \\ \hline \overline{C}_K & \overline{D}_K \end{array}\right]$, for the distributed system (3), where the SSS structure of each matrix is:

$$\begin{aligned} \overline{A}_K &= \mathcal{SSS}(G_s^m, M_s^m, N_s^m, F_s, G_s^p, M_s^p, N_s^p) \\ \overline{B}_K &= \mathcal{SSS}(K_s^m, R_s^m, Q_s^m, G_s, K_s^p, R_s^p, Q_s^p) \\ \overline{C}_K &= \mathcal{SSS}(S_s^m, T_s^m, U_s^m, N_s, S_s^p, T_s^p, U_s^p) \\ \overline{D}_K &= \mathcal{SSS}(X_s^m, Z_s^m, P_s^m, Y_s, X_s^p, Z_s^p, P_s^p) \end{aligned}$$

then it can be verified that such a controller can be directly distributed into subcontrollers K_s:

$$\begin{bmatrix} \dot{\xi}_s \\ f_{s-1}^p \\ f_{s+1}^m \\ u_s \end{bmatrix} = \begin{bmatrix} F_s & \hat{G}_s^p & \hat{G}_s^m & G_s \\ \hat{N}_s^p & \hat{R}_s^p & 0 & \hat{Q}_s^p \\ \hat{N}_s^m & 0 & \hat{R}_s^m & \hat{Q}_s^m \\ N_s & \hat{S}_s^p & \hat{S}_s^m & Y_s \end{bmatrix} \begin{bmatrix} \xi_s \\ f_s^p \\ f_s^m \\ y_s \end{bmatrix} \tag{5}$$

where

$$\hat{G}_s^* = \begin{bmatrix} G_s^* & K_s^* & 0 & 0 \end{bmatrix}, \qquad \hat{Q}_s^* = \begin{bmatrix} 0 & (Q_s^*)^T & 0 & (P_s^*)^T \end{bmatrix}^T$$
$$\hat{S}_s^* = \begin{bmatrix} 0 & 0 & S_s^* & X_s^* \end{bmatrix}, \qquad \hat{N}_s^* = \begin{bmatrix} (N_s^*)^T & 0 & (U_s^*)^T & 0 \end{bmatrix}^T$$
$$\hat{R}_s^* = diag(M_s^*, R_s^*, T_s^*, Z_s^*)$$

where * is held constant as either m or p in each term. This is obviously the same structure as the subsystems, connected as in Figure 2. Interpreted in this way, the f_s^* channels represent the communications between each subcontroller. This illustrates a key advantage of SSS over $\mathcal{H}$ matrix or frequency domain controller design methods for distributed systems: SSS structured controllers admit a simple distributed controller implementation, similar in structure to those sought in [11] and [12], without any additional computation.

III. Computational Methods

For computational complexity, we use 'big O' notation,

Definition 1: A positive function is $f(N) \in \mathcal{O}(N^\alpha)$ if there exist finite positive constants, $\infty > c, \kappa > 0$ such that $f(N) < cN^\alpha, \forall N > \kappa$.

$$\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \end{bmatrix}}_{\dot{\overline{x}}} = \underbrace{\begin{bmatrix} A_1 & B_1^p C_2^p & B_1^p W_2^p C_3^p & B_1^p W_2^p W_3^p C_4^p & B_1^p W_2^p W_3^p W_4^p C_5^p \\ B_2^m C_1^m & A_2 & B_2^p C_3^p & B_2^p W_3^p C_4^p & B_2^p W_3^p W_4^p C_5^p \\ B_3^m W_2^m C_1^m & B_3^m C_2^m & A_3 & B_3^p C_4^p & B_3^p W_4^p C_5^p \\ B_4^m W_3^m W_2^m C_1^m & B_4^m W_3^m C_2^m & B_4^m C_3^m & A_4 & B_4^p C_5^p \\ B_5^m W_4^m W_3^m W_2^m C_1^m & B_5^m W_4^m W_3^m C_2^m & B_5^m W_4^m C_3^m & B_5^m C_4^m & A_5 \end{bmatrix}}_{\overline{A}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}}_{\overline{x}} + \overline{B}_1 \overline{w} + \overline{B}_2 \overline{u} \tag{1}$$

Informally, we will say that a procedure 'is' $\mathcal{O}(N^\alpha)$ if it can be computed in $f(N) \in \mathcal{O}(N^\alpha)$ flops.

In this section we will discuss iterative methods for solving some matrix equations commonly encountered in control, which may be conducted very efficiently for SSS matrices. The key to the method is that there are 'fast' converging iterative algorithms for solving these problems, and that each iteration may be computed using SSS arithmetic algorithms in only $\mathcal{O}(N)$. We will begin with a discussion of some of the properties of SSS matrices under arithmetic operations, which are important to understanding our techniques.

A. $\mathcal{SSS}$ orders

For an SSS matrix: $\overline{A} = \mathcal{SSS}(B_s^m, W_s^m, C_s^m, A_s, B_s^p, W_s^p, C_s^p)$, $s \in \{1, 2, ...N\}$, many matrix-matrix operations are $\mathcal{O}(N)$, but cubic in the sizes of the generator matrices. For example, if $B_s^m, W_s^m, C_s^m, A_s, B_s^p, W_s^p, C_s^p \in \mathbb{R}^{n\times n}$, $\forall s \in \{1, 2, ...N\}$, then computing $\overline{A}^2$ will take $40n^3N$ flops [18]. However, it has be shown[17] that the 'orders' of the SSS matrices, and thus their generators, grow in size additively with each SSS addition and multiplication operation. To be explicit, we define:

Definition 2: The upper and lower order of an SSS matrix is the largest size of its upper and lower multiplier terms(W_s^m and W_s^p in $\overline{A}$), respectively. The class of SSS matrices of maximum lower and upper orders w_l and w_u with N diagonal terms is denoted as $\mathcal{SSS}^{w_l,w_u,N}$.

Thus the growing order of SSS matrices can be related as:

Lemma 1: For conformably partitioned matrices $\overline{A} \in \mathcal{SSS}^{a_l,a_u,N}$ and $\overline{B} \in \mathcal{SSS}^{b_l,b_u,N}$, then $\overline{A} + \overline{B} = \overline{C} \in \mathcal{SSS}^{c_l,c_u,N}$ where $c_l \leq a_l + b_l$, $c_u \leq a_u + b_u$, and $\overline{AB} = \overline{D} \in \mathcal{SSS}^{d_l,d_u,N}$ where $d_l \leq a_l + b_l$, $d_u \leq a_u + b_u$.

Proof: This can be seen from the addition and multiplication algorithms[18] [19]. ∎

This is important since we intend to use iterative algorithms for controller synthesis, and evidently each iteration will cost more than the previous. However, in the following subsections we will bound the number of iterations, k, for each algorithm independently of N, and thus still have $\mathcal{O}(N)$ computation, although with a very large multiplicative constant in front. To remedy this, in Section IV we will show that for practical purposes, low order approximations may be used to considerably cut down on the order growth.

B. Matrix Sign Function

The matrix sign function[20][21], defined for a square matrix X with no eigenvalues on the imaginary axis, may be calculated as

Algorithm 1 Sign Iteration[20]

$$\begin{aligned} & Z_0 = X \\ Z_{k+1} &= \frac{1}{2}(Z_k + Z_k^{-1}) \quad \text{for k} = 0, 1, 2, ... \\ sign(X) &= \lim_{k\to\infty} Z_k \end{aligned}$$

This iteration has been proven to be equivalent to a Newton's method, and correspondingly converges quadratically near the solution, but may start out slower. However, from the condition and damping of the spectrum of X, the number of iterations to convergence can be bounded:

Lemma 2: For a matrix X with no purely imaginary eigenvalues, the number of sign iterations k to reach $\|Z_k - sign(X)\|_2 \leq \epsilon$ for Jordan decomposition $X = PJP^{-1}$ will be $\mathcal{O}(\log_2(\eta)^2 + \log_2(\log_2(\epsilon^{-1} + cond(P))))$, where $\eta = \max_i\{1 + |\Re(\lambda_i)| + |\Re(\lambda_i)|^{-1} + \frac{|\Im(\lambda_i)|}{|\Re(\lambda_i)|}\}$ *Proof:* [10], Lemma 3.5 ∎

So, given some $\overline{X}$ of size N with SSS structure, we may compute iterations, each of only $\mathcal{O}(N)$(since SSS addition, inversion, and scalar multiplication are $\mathcal{O}(N)$), to compute the matrix sign of $\overline{X}$. However, to claim $\mathcal{O}(N)$ sign computation, we must also upper bound k with a constant independently of N, $\forall N \in \mathbb{N}$, using the following assumptions:

- A1: $\rho(\overline{X}) < \beta_1 < \infty$
- A2: $\min_i |\Re(\lambda_i(\overline{X}))| > \beta_2 > 0$
- A3: $cond(P) < \beta_3 < \infty$ for $\overline{X} = PJP^{-1}$

A1 and A2 imply that η will always be finite, but do not seem very restrictive, as they just imply some analytic continuity around the imaginary axis and a finitely bounded spectrum. A3 is the least intuitive, but keep in mind that β_3 can be *very* large without unduly increasing k, for example $log_2(log_2(10^{100})) < 9$. From now on the set of matrices that satisfy A1, A2, and A3 will be denoted as $\mathcal{A}$. We can thus state:

Lemma 3: For the set of SSS matrices $\overline{X} \in SSS^{\bullet,\bullet,N}$, $\overline{X} \in \mathcal{A}$ $\forall N \in \mathbb{N}$, an approximation, $\overline{S} \in SSS^{\bullet,\bullet,N}$ to $sign(\overline{X})$ can be calculated to within some prespecified positive tolerance $\epsilon > \|\overline{S} - sign(\overline{X})\|_2$ in $\mathcal{O}(N)$.

We will use the conservative Frobenius norm to check convergence, since we can calculate it exactly in $\mathcal{O}(N)$ for SSS matrices($\|\overline{X}\|_F^2 = Tr(\overline{X}^T\overline{X})$).

C. Stability Check Through Matrix Sign

It can be shown that:

Lemma 4: For some matrix X, $sign(X) = -I$ if and only if $\Re(\lambda(X)) < 0$.

Proof: Follows from the Jordan-decomposition based definition of the sign function in[21] ∎

Because of the fast, size independent, convergence of the sign iterations, this can be used as an $\mathcal{O}(N)$ check on the stability of SSS matrices $\in \mathcal{A}, \forall N \in \mathbb{N}$.

Remark 1: Note that we are using 'check' here in a practical, finite precision, sense. In exact arithmetic, Algorithm 1 will never converge exactly to $sign(\overline{X})$ in finite k unless $\lambda(\overline{X}) \in \{-1, 1\}$, but in practice we 'check' stability by checking the approximation from Lemma 3 : $\|\overline{S} + I\|_F$ should be small.

D. Structure Preserving Permutation

Since we will be designing controllers, it is also necessary to check the stability of closed loop systems, which will have matrices with SSS blocks. We could just use block matrix algebra to do the matrix sign iteration, but this could

fail when using Schur complements to calculate the inverses. Instead, we can permute the rows and columns of SSS block matrices to form single SSS matrices. The following lemma is for a 2×2 block, but it can be easily extended to other dimensions.

Lemma 5: Given matrices $\overline{R}, \overline{X}, \overline{Y}, \overline{Z}$ with SSS representations:

$$\begin{aligned}
\overline{R} &= \mathcal{SSS}(B_s^{Rm}, W_s^{Rm}, C_s^{Rm}, A_s^R, B_s^{Rp}, W_s^{Rp}, C_s^{Rp}) \\
\overline{X} &= \mathcal{SSS}(B_s^{Xm}, W_s^{Xm}, C_s^{Xm}, A_s^X, B_s^{Xp}, W_s^{Xp}, C_s^{Xp}) \\
\overline{Y} &= \mathcal{SSS}(B_s^{Ym}, W_s^{Ym}, C_s^{Ym}, A_s^Y, B_s^{Yp}, W_s^{Yp}, C_s^{Yp}) \\
\overline{Z} &= \mathcal{SSS}(B_s^{Zm}, W_s^{Zm}, C_s^{Zm}, A_s^Z, B_s^{Zp}, W_s^{Zp}, C_s^{Zp})
\end{aligned}$$

and lifted vectors $\overline{e}, \overline{f}, \overline{g}, \overline{h}$, the relations

$$\begin{bmatrix} \overline{e} \\ \overline{f} \end{bmatrix} = \begin{bmatrix} \overline{R} & \overline{X} \\ \overline{Y} & \overline{Z} \end{bmatrix} \begin{bmatrix} \overline{g} \\ \overline{h} \end{bmatrix}, \quad \overline{\begin{bmatrix} e \\ f \end{bmatrix}} = \overline{P} \overline{\begin{bmatrix} g \\ h \end{bmatrix}} \tag{6}$$

are equivalent, to within a permutation, where $\overline{P} = \mathcal{SSS}(\hat{B}_s^m, \hat{W}_s^m, \hat{C}_s^m, \hat{A}_s, \hat{B}_s^p, \hat{W}_s^p, \hat{C}_s^p)$ with

$$\begin{aligned}
\hat{A}_s &= \begin{bmatrix} A_s^R & A_s^X \\ A_s^Y & A_s^Z \end{bmatrix}, \quad \hat{C}_s^* = \begin{bmatrix} diag(C_s^{R*}, C_s^{X*}) \\ diag(C_s^{Y*}, C_s^{Z*}) \end{bmatrix} \\
\hat{B}_s^* &= diag(\begin{bmatrix} B_s^{R*} & B_s^{X*} \end{bmatrix}, \begin{bmatrix} B_s^{Y*} & B_s^{Z*} \end{bmatrix}) \\
\hat{W}_s^* &= diag(W_s^{R*}, W_s^{X*}, W_s^{Y*}, W_s^{Z*})
\end{aligned}$$

and the $*$'s are held constant as m or p in each term. *Proof:* This is easily verified by substitution. ∎

The transformation from $\begin{bmatrix} \overline{e} \\ \overline{f} \end{bmatrix} \rightarrow \overline{\begin{bmatrix} e \\ f \end{bmatrix}}$ could be called a 'shuffle' permutation, since $\overline{e}$ and $\overline{f}$ are shuffled like a deck of cards to get $\overline{\begin{bmatrix} e \\ f \end{bmatrix}}$. The reverse operation is also possible:

Lemma 6: Given an SSS matrix $\overline{P}$ with conformably partitioned generators: $\overline{P} = \mathcal{SSS}(\begin{bmatrix} B_s^{1m} \\ B_s^{2m} \end{bmatrix}, W_s^m, \begin{bmatrix} C_s^{1m} & C_s^{2m} \end{bmatrix}, \begin{bmatrix} A_s^{11} & A_s^{12} \\ A_s^{21} & A_s^{22} \end{bmatrix}, \begin{bmatrix} B_s^{1p} \\ B_s^{2p} \end{bmatrix}, W_s^p, \begin{bmatrix} C_s^{1p} & C_s^{2p} \end{bmatrix})$, the relations (6) are equivalent, to within a permutation, where:

$$\begin{aligned}
\overline{R} &= \mathcal{SSS}(B_s^{1m}, W_s^m, C_s^{1m}, A_s^{11}, B_s^{1p}, W_s^p, C_s^{1p}) \\
\overline{X} &= \mathcal{SSS}(B_s^{1m}, W_s^m, C_s^{2m}, A_s^{12}, B_s^{1p}, W_s^p, C_s^{2p}) \\
\overline{Y} &= \mathcal{SSS}(B_s^{2m}, W_s^m, C_s^{1m}, A_s^{21}, B_s^{2p}, W_s^p, C_s^{1p}) \\
\overline{Z} &= \mathcal{SSS}(B_s^{2m}, W_s^m, C_s^{2m}, A_s^{22}, B_s^{2p}, \hat{W}_s^p, C_s^{2p})
\end{aligned}$$

and everything is assumed conformably partitioned. *Proof:* This is easily verified by substitution. ∎

This method will allow us to perform sign iterations for block SSS matrices without using block algorithms, and then repermute to obtain the block solution. This will be implicitly used frequently in the sequel.

E. Lyapunov, Riccati, and H_2 through Matrix Sign

It has been shown([20][22]) that general symmetric algebraic Riccati equations and stable Lyapunov equations:

$$\begin{aligned}
\mathcal{R}(X) = XA + A^T X + Q - XRX = 0 \qquad (7) \\
\mathcal{L}(X) = XA + A^T X + Q = 0
\end{aligned}$$

$Q = Q^T$, $R = R^T$, can be solved by forming the appropriate Hamiltonian matrix and calculating its matrix sign. Such methods have also been used for solving centralized optimal control problems for partial differential equations with $\mathcal{H}$ matrices in [10], and the idea with SSS matrices is similar. By applying Lemma 5, the block SSS Hamiltonian, $H = \begin{bmatrix} \overline{A} & -\overline{R} \\ -\overline{Q} & -\overline{A}^T \end{bmatrix}$ may be permuted into a single SSS matrix $\overline{H}$, on which the sign iterations may be performed using Algorithm 1, and assuming that $H \in \mathcal{A}$, the matrix sign iteration will converge within some tolerance, $sign(\overline{H})$ may be repermuted using Lemma 6, and a solution, $\overline{X} \in \mathcal{SSS}^{\bullet,\bullet,N}$, may be solved for in $\mathcal{O}(N)$.

The extension to $\mathcal{O}(N)$ H_2 optimal control of SSS systems is then obvious. It is well known [23] that all that is needed to synthesize such controllers is the solution of two Riccati equations and a few matrix multiplications and additions, and due to the closedness properties of the SSS structure, the resulting H_2-optimal controller matrices will also be SSS. Likewise, analyzing the stability and H_2 norm of the closed loop system, $\left[\begin{array}{c|c} \overline{A}_{cl} & \overline{B}_{cl} \\ \hline \overline{C}_{cl} & 0 \end{array}\right]$ amounts to a stability check of $\overline{A}_{cl}$, and the solution of a Lyapunov equation to compute a closed loop Grammian, each of which may be computed in $\mathcal{O}(N)$, assuming the relevant matrices are $\in \mathcal{A}$.

Remark 2: We have not specified any method of checking the assumptions for the existence of H_2 optimal controllers, because we do not, in fact, have an efficient method for explicitly checking stabilizability, detectability, and the location of transmission zeros. Instead, we suggest performing the SSS H_2 synthesis calculations and checking the closed loop stability and performance of the resulting controller(if the Riccati equations have solutions!) to see if it is valid.

IV. Approximations

As mentioned in Section III, for SSS matrix multiplication and addition, some of the generator matrices of the resulting SSS matrix will grow with each operation. Because of this, we cannot go on adding and multiplying SSS matrices in an efficient manner forever, as after each operation the SSS matrices will grow in order, and the SSS operations are cubically dependent on the orders. Since we have assumed that the number of iterations needed for controller synthesis is bounded independently of N, the overall procedure will still be $\mathcal{O}(N)$, but the hidden constant(c in Definition 1) may be prohibitively large for practical computation(notice that the SSS order will double at each step of the sign iteration in Algorithm 1). The solution to this problem lies in order-reducing SSS approximations, discussed in the next section.

A. Optimal order reduction

For infinite dimensional systems, many systems can be shown to have exponentially spatially decaying(ESD)[15] operators, a characterization that basically implies that the norm of the couplings between subsystems is bounded by some exponential decay in space. In [14] it is shown that quadratically optimal controllers for ESD spatially invariant

distributed systems are also ESD, and in [15] this is extended to the spatially varying case for the solutions of Riccati equations of ESD systems. This leads to arguments for spatially truncating the controller for an almost optimal distributed implementation [14].

For very large($N >> 0$) finite dimensional systems, we may also take this route: given a Riccati or Lyapunov equation solution with SSS structure, $\overline{X}$, we could compute a p^{th} spatial truncation approximation of SSS structure by calculating the middle $2p+1$ diagonals, and then finding the generator matrices of the truncation $\overline{X}_p \in \mathcal{SSS}^{p,p,N}$ using the construction algorithm[18] specialized to banded matrices. However, within the SSS framework, there is a *Hankel norm* optimal order reduction [17][18]. That is, given some SSS matrix $\overline{X} \in \mathcal{SSS}^{q,q,N}$, we may calculate a lower order approximation $\tilde{\overline{X}}$ that achieves:

$$\inf_{\tilde{\overline{X}} \in \mathcal{SSS}^{p,p,N}} \|\tilde{\overline{X}} - \overline{X}\|_H \tag{8}$$

where $\|Y\|_H = \max_i \|H_i(Y)\|$ and $H_i(Y)$ are the Hankel blocks[17] of matrix Y, and $p < q$. The Hankel norm is intimately related to the spectral norm as $\|Y\|_H \leq \|Y\|_2 \leq \sqrt{N}\|Y\|_H$ for some $Y \in \mathbb{R}^{N\times N}$, and there exist $\mathcal{O}(N)$ algorithms for computing (8) which work very well(much better than spatial truncation) in practice.

This also provides an answer to the problem of the SSS order growing after each operation; after every few order-increasing operations, we can do an SSS order reducing approximation, making the iterative schemes very efficient in practice. However, even if the order of the SSS matrices of the Riccati solutions and state space matrices ($\overline{A}_K$, $\overline{B}_K$, $\overline{C}_K$, $\overline{D}_K$) are limited, using spatial truncation or $\|\cdot\|_H$ optimal approximation, the distributed controller(5) might still be very inefficiently represented(especially considering the construction of the $\hat{R}_s$ terms). Therefore, the large size of the communication vectors might surpass the abilities of the controller communication links.

However, through Lemmas 5 and 6, the input-output and state dynamics of the controller(5) may be expressed as a single block SSS matrix: $\overline{\begin{bmatrix}\dot{\xi}_s \\ u_s\end{bmatrix}} = \overline{P}\begin{bmatrix}\xi_s \\ y_s\end{bmatrix}$ with $\overline{P} = \mathcal{SSS}(\begin{bmatrix}\hat{G}_s^m \\ \hat{S}_s^m\end{bmatrix}, \hat{R}_s^m, \begin{bmatrix}\hat{N}_s^m & \hat{Q}_s^m\end{bmatrix}, \begin{bmatrix}F_s & G_s \\ N_s & Y_s\end{bmatrix}, \begin{bmatrix}\hat{G}_s^p \\ \hat{S}_s^p\end{bmatrix}, \hat{R}_s^p, \begin{bmatrix}\hat{N}_s^p & \hat{Q}_s^p\end{bmatrix})$, and by doing a model order reduction on this matrix, the size of the $\hat{R}$ terms may be reduced without destroying the distributed structure. However, it should be noted that neither the spatial truncation nor the $\|\cdot\|_H$ optimal approximation of $\overline{P}$ is guaranteed to remain stabilizing, so the closed loop stability and performance should be checked for each order-reduced controller.

B. Numerical issues relating to the Approximations

Of course, iteratively approximating SSS matrices using lower orders will call into question the use of the sign function methods for checking stability and solving the Riccati equation. With respect to rounding errors, the sign function has been found to often work just as well as invariant subspace techniques [24], but as expected, larger intermediate approximations often cause larger residual errors. Hence we cannot 'solve' the Lyapunov and Riccati equations; there will always be small non-zero residuals, $\mathcal{L}(\overline{X})$, $\mathcal{R}(\overline{X})$, the norms of which are not necessarily a good measure of the backwards error[25][26], and the criteria suggested for which are not easily calculable using SSS methods. Furthermore, due to the potential fragility of optimal controllers[27] it is not acceptable to blindly use an approximated controller and assume that it will have the expected closed loop stability and performance; some kind of satisfactory closed loop test is needed, which we will next describe.

For checking matrix stability, it is infeasible to run Algorithm 1 approximation-free, since the SSS orders of $\overline{Z}_k$ grow as 2^k, but an unsuitable low-order approximation could bump an unstable eigenvalue into the left half plane, or vice-versa, falsifying the result. For stable matrices, $\Re(\lambda(\overline{X})) < 0$, $\overline{X} \in \mathcal{A}$ the SSS order necessary to represent $\overline{Z}_k$ exactly will eventually approach 0, since $\lim_{k\to\infty} \overline{Z}_k = sign(\overline{X}) = -I \in \mathcal{SSS}^{0,0,N}$, but due to the erratic behavior of the spectrum $\lambda(\overline{Z}_k)$ during the iterations, any approximation, $\tilde{\overline{Z}}_k$, must be very accurate: $\|\overline{Z}_k - \tilde{\overline{Z}}_k\|_2 < \frac{\beta_2}{\beta_1\beta_3}$ to guarantee that no eigenvalues cross the imaginary axis.

However, for *symmetric* matrices $\overline{Z}_k = \overline{Z}_k^T$, $\lambda(\overline{Z}_k) \in \mathbb{R}$, standard perturbation theory [28] shows that

$$\max_{t\in\{1,2,\ldots,N\}} |\lambda_t(\overline{Z}_k) - \lambda_t(\tilde{\overline{Z}}_k)| \leq \|\overline{Z}_k - \tilde{\overline{Z}}_k\|_F$$

and also, for *symmetric* sign iterations, it can be shown that any eigenvalues initially inside the unit circle $|\lambda_t(\overline{X})| < 1$ will be transported outside after the first iteration: $|\lambda_t(\overline{Z}_k)| \geq 1, \quad \forall k \in \mathbb{N}$, allowing the use of aggressive(any $\|\overline{Z}_k - \tilde{\overline{Z}}_k\|_F < 1$) approximations without the risk of eigenvalues crossing the imaginary axis. Fortunately, using Lyapunov stability theory, we can convert non-symmetric stability problems into symmetric problems, as follows.

It is well known[29] for some $E, P \in \mathbb{R}^{N\times N}$, $P = P^T$ with $EP + PE^T \prec 0$, that $\Re(\lambda(E)) < 0$ if and only if $P \succ 0$. Hence for SSS matrices, one can use the matrix sign function to solve $\overline{A}_{cl}P + P\overline{A}_{cl}^T + I = 0$ for P using iterative approximations, and if the resulting $\tilde{\overline{P}} \succ 0$ and $\overline{A}_{cl}\tilde{\overline{P}} + \tilde{\overline{P}}\overline{A}_{cl}^T \prec 0$, which are both symmetric stability problems(and thus robust to approximation errors), then it is guaranteed that $\Re(\lambda(\overline{A}_{cl})) < 0$.

As for the closed loop performance, after stability has been verified, the Lyapunov equation: $\overline{A}_{cl}S + S\overline{A}_{cl}^T + \overline{B}_{cl}\overline{B}_{cl}^T = 0$, may be relaxed to a strict inequality: $\|\overline{C}_{cl}(sI - \overline{A}_{cl})^{-1}\overline{B}_{cl}\|_2 < \gamma$ if and only if $\exists\Pi$: $Tr(\overline{C}_{cl}\Pi\overline{C}_{cl}^T) < \gamma^2$ where $\overline{A}_{cl}\Pi + \Pi\overline{A}_{cl}^T + \overline{B}_{cl}\overline{B}_{cl}^T \prec 0$. For any such γ, there exists some small $\epsilon > 0$ such that, $Tr(\overline{C}_{cl}S_\epsilon\overline{C}_{cl}^T) < \gamma^2$, where S_ϵ is the solution to the preturbed Lyapunov equation[30]: $\mathcal{L}(S_\epsilon) = \overline{A}_{cl}S_\epsilon + S_\epsilon\overline{A}_{cl}^T + \overline{B}_{cl}\overline{B}_{cl}^T + \epsilon I = 0$. In practice, this means that we can use the matrix sign function with iterative approximations to solve such a perturbed Lyapunov equation for some $\tilde{\overline{S}}_\epsilon$ using a small ϵ, and $(Tr(\overline{C}_{cl}\tilde{\overline{S}}_\epsilon\overline{C}_{cl}^T))^{1/2} < \gamma_u$

is an upper bound on the H_2 norm, as long as the residual satisfies $\mathcal{L}(\overline{\tilde{S}}_\epsilon) \prec \epsilon I$, which is just a symmetric stability problem. Since $S_\epsilon = S + \epsilon P$, then given the residuals of the closed loop stability performance equations, any ϵ such that $\left(\overline{A}_{cl}\tilde{\overline{S}} + \tilde{\overline{S}}\overline{A}_{cl}^T + \overline{B}_{cl}\overline{B}_{cl}^T\right) + \epsilon\left(\overline{A}_{cl}\tilde{\overline{P}} + \tilde{\overline{P}}\overline{A}_{cl}^T\right) \prec 0$, with corresponding $\tilde{\overline{S}}_\epsilon = \tilde{\overline{S}} + \epsilon\tilde{\overline{P}}$ will work.

Thus, the use of iterative low-order approximations in checking matrix stability, calculating solutions to Lyapunov and Riccati equations, and synthesizing sub-optimal H_2 controllers is acceptable, since the *a posteriori* verifications of closed loop stability and performance may be converted to symmetric matrix stability problems, and thus efficiently *and* robustly(with respect to approximation errors) checked.

V. Application: Vehicle Platooning

As a demonstration of the method, we will consider the control of a car platoon. We note that H_2 performance isn't necessarily a very good measure of the 'string stability' usually desired in such applications(although it has been considered before [31]). Rather, the point of this example is to both show how coupled systems may be fitted into the subsystem structure (2), and to demonstrate the $\mathcal{O}(N)$ computational complexity for calculating sub-optimal distributed controllers, in the hope that these techniques will be extended to other control methods and applications.

The dynamics of each car is modeled as a simple point mass with an actuator gain(g_s) and lag(τ_s):

$$\begin{bmatrix} \dot{x}_s^1 \\ \dot{x}_s^2 \\ \dot{x}_s^3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & \frac{-1}{\tau_s} \end{bmatrix} \begin{bmatrix} x_s^1 \\ x_s^2 \\ x_s^3 \end{bmatrix} + \begin{bmatrix} 0 \\ q_s^1 \\ 0 \end{bmatrix} v_s + \begin{bmatrix} 0 \\ 0 \\ g_s \end{bmatrix} u_s$$

which is similar to models previously considered in the literature[32][31][5], with a force disturbance input $v_s(t)$ representing wind gusts. Each car measures its own velocity, and the relative position between itself and the car in front of it:

$$y_s = \begin{bmatrix} -c_s^1 & 0 \\ 0 & c_s^2 \end{bmatrix} \begin{bmatrix} x_s^1 \\ x_s^2 \end{bmatrix} + \begin{bmatrix} c_s^1 \\ 0 \end{bmatrix} x_{s-1}^1 + \begin{bmatrix} q_s^3 & 0 \\ 0 & q_s^4 \end{bmatrix} n_s$$

except for the front car, which measures its own position. The cost function will be based on each car's input, $z_s^1 = f_s^1 u_s$, and the difference between its following distance and a reference, $z_s^2 = (x_{s-1}^1 - x_s^1 - \hat{r}_s)$, where the reference: $\dot{\hat{r}} = \frac{-1}{\kappa_s}\hat{r} + r_s$, is treated as a disturbance, but filtered through a lag to keep the H_2 norm finite and better represent a real situation. Note that the dynamics are uncoupled in this example(although linear draft dynamics could easily be added), but the vehicles are coupled through their measurements and cost functions. Such an interconnected system can be put in subsystem form(2) simply as Σ_s :

$$\left[\begin{array}{cccc|cc|cccc|c} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & q_s^1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-1}{\tau_s} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g_s \\ 0 & 0 & 0 & \frac{-1}{\kappa_s} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & f_s^1 \\ -f_s^2 & 0 & 0 & -f_s^2 & 0 & f_s^2 & 0 & 0 & 0 & 0 & 0 \\ \hline -c_s^1 & 0 & 0 & 0 & 0 & c_s^1 & 0 & 0 & q_s^3 & 0 & 0 \\ 0 & c_s^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & q_s^4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right]$$

where the disturbance input is partitioned as: $w_s = \begin{bmatrix} v_s & r_s & n_s^T \end{bmatrix}^T$, and each car is allowed to measure it's own unfiltered reference: $y_s^3 = r_s$.

This subsystem model can then be used to form the lifted system of SSS matrices in (3), allowing the use of the computational tools described in Section III to perform H_2 controller synthesis.

1) $\mathcal{O}(N)$ Demonstration for H_2 Synthesis: For this example, we will allow the coefficients to vary randomly in space. Each coefficient will have the following mean values: $\{\overline{\tau}, \overline{\kappa}, \overline{q^1}, \overline{g}, \overline{f^1}, \overline{f^2}, \overline{c^1}, \overline{c^2}, \overline{q^3}, \overline{q^4}\} = \{0.1, 0.5, 0.1, 1, 1, 1, 1, 1, 0.1, 0.1\}$, but at each point $s \in \{1, 2, ...N\}$ they will be allowed to randomly vary within 20% of this value(except for κ which is held constant). For example, at each $s \in \{1, 2, ...N\}$, $g_s = \overline{g}(1 + \frac{1}{5}\mathcal{U}(-1, 1))$, where $\mathcal{U}$ is a uniform distribution. For problems of size $N = \{5, 10, 25, 50, 100, 150, 200\}$ we did this 50 times each using our SSS solver and MATLAB's `h2syn`. The closed loop $\|\cdot\|_2$ norms varied between about 2 and 15, and the performance of the SSS controllers was always within 10^{-5} of that of the MATLAB centrally optimal controllers.

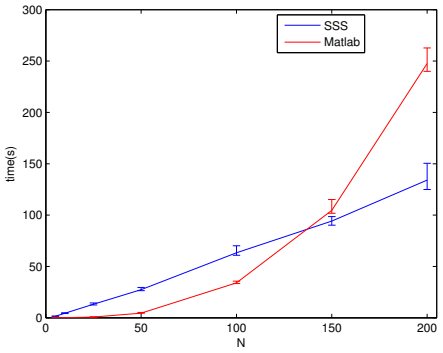

Fig. 3. Average computational time comparison of SSS vs MATLAB H_2 synthesis routines. Error bars indicate maximum and minimum times. Note the linear trend for the SSS based solver compared to the approximately cubic trend of MATLAB's unstructured solver.

In figure 3 we see a comparison of the synthesis computation times, where the bars show the maximum and minimum time for each value of N, and the linear complexity of our approach becomes an advantage after about $N \approx 150$.

For reference to the algorithmic discussions, SSS orders of $w_u = w_l = 16$ were used in all iterative schemes, and the sign iterations took about 11 iterations to converge.

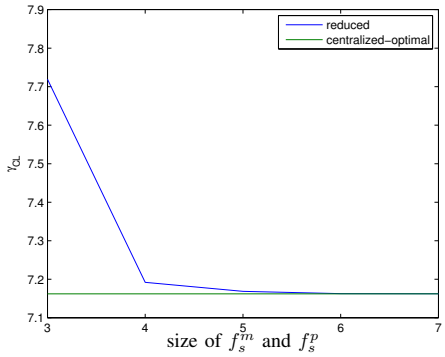

Fig. 4. Closed loop performance of $\|\cdot\|_H$ optimal communication-reduced controllers

2) Distributedness: For a typical example with $N = 50$, figure 4 shows the decrease in closed loop H_2 norm with the increase in communication order, using the optimal order reduction discussed in Section IV. All reduced controllers with communication links of size at least 3 were stable, and as we see, there is an exponential-like decrease, with high performance controllers even for very small communication links.

VI. Conclusion

We have shown how the SSS structure of 1-D spatially varying interconnected systems may be exploited to compute distributed controllers with almost centrally optimal H_2 performance in only $\mathcal{O}(n^3N)$, compared to $\mathcal{O}(n^3N^3)$ for centralized methods. Our techniques use iterative approximations, potentially introducing errors, but sufficient conditions for checking closed loop stability and performance may be posed as symmetric stability problems, robust to these errors.

The H_∞ analysis and synthesis problem, while more complicated, is similar, and the details are discussed in [33]. The discrete time Riccati equation may also be solved with the matrix sign function[34], so many results should be extendable. Our stability and performance analysis should also extend to $\mathcal{H}$ matrices[10], but it is not clear what the distributed implementation of such controllers could be.

Acknowledgment

The authors thank Paolo Massioni, Jan-Willem van Wingerden, Tamas Keviczky, and the reviewers.

References

[1] D. S. Henningson, M. Hogberg, and M. Chevalier, "Optimal feedback control applied to boundary layer flow," *Journal of Turbulence*, vol. 6, no. 25, 2005.

[2] T. R. Bewley and S. Liu, "Optimal and robust control and estimation of linear paths to transition," *Journal of Fluid Mechanics*, vol. 365, pp. 205–349, 1998.

[3] F. Wu and S. E. Yildizoglu, "Distributed parameter-dependent modeling and control of flexible structures," *Transactions of the ASME*, vol. 127, pp. 230–239, 2005.

[4] R. Horowitz and P. Varaiya, "Control design of an automated highway system," *Proceedings of the IEEE*, vol. 88, pp. 913–925, 2000.

[5] S. Sheikholeslam and C. Desoer, "Longitudinal control of a platoon of vehicles," *In Proc. Amer. Control Conf.*, pp. 291–296, 1990.

[6] J. P. Lynch and K. H. Law, "Decentralized control techniques for large-scale civil structural systems," *Proceedings of the IMAC*, 2002.

[7] J. Fowler and R. D'Andrea, "A formation flight experiment," *IEEE Control Systems Magazine*, vol. 23, pp. 35–43, 2003.

[8] S. Jiang, P. G. Voulgaris, L. E. Holloway, and L. A. Thompson, "Distributed control of large segmented telescopes," in *Proc. Amer. Control Conf.*, 2006.

[9] I. Rosen and C. Wang, "A multilevel technique for the approximate solution of operator Lyapunov and algebraic Riccati equations," *SIAM Journal of Numerical Analysis*, vol. 32, pp. 514–541, 1995.

[10] L. Grasedyck, W. Hackbusch, and B. Khoromskij, "Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices," *Computing*, vol. 70, pp. 121–165, 2003.

[11] R. D'Andrea and G. E. Dullerud, "Distributed control design for spatially interconnected systems," *IEEE Trans. Autom. Control*, vol. 48, pp. 1478–1495, 2003.

[12] G. E. Dullerud and R. D'Andrea, "Distributed control of heterogeneous systems," *IEEE Trans. Autom. Control*, vol. 49, pp. 2113–2128, 2004.

[13] F. Wu, "Distributed control for interconnected linear parameter-dependent systems," *IEEE Proceedings on Control Theory Applications*, vol. 150, pp. 518–527, 2003.

[14] B. Bamieh, F. Paganini, and M. A. Dahleh, "Distributed control of spatially invariant systems," *IEEE Trans. Autom. Control*, vol. 47, pp. 1091–1106, 2002.

[15] N. Motee and A. Jadbabaie, "Optimal control of spatially distributed systems," *IEEE Trans. Autom. Control*, to be published.

[16] P. Sharma and C. Beck, "Modelling and distributed control of mobile offshore bases," *Proc. Amer. Control Conf.*, pp. 5238–5243, 2004.

[17] P. Dewilde and A. J. V. D. Veen, *Time-Varying systems and Computations*. Kluwer Academic Publishers, 1998.

[18] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, A.-J. van der Veen, and D. White, "Fast stable solvers for sequentially semi- separable linear systems of equations," *Report, Lawrence Livermore National Laboratory*, 2003.

[19] Y. Eidelman and I. Gohberg, "On a new class of structured matrices," *Integral Equations and Operator Theory*, vol. 34, pp. 292–324, 1999.

[20] J. Roberts, "Linear model reduction and solution of the algebraic Riccati equation by use of the sign function," *International Journal of Control*, vol. 32, pp. 677–687, 1980.

[21] C. S. Kenney and A. J. Laub, "The matrix sign function," *IEEE Trans. Autom. Control*, vol. 40, pp. 1330–1348, 1995.

[22] C. Kenney, A. J. Laub, and E. Jonckheere, "Positive and negative solutions of dual Riccati equations by matrix sign function iteration," *Systems and Control Letters*, vol. 13, pp. 109–116, 1989.

[23] K. Zhou, K. Glover, and J. C. Doyle, *Robust and Optimal Control*. Prentice Hall, 1996.

[24] R. Byers, C. He, and V. Mehrmann, "The matrix sign function method and the computation of invariant subspaces," *SIAM Journal of Matrix Analysis and Applications*, vol. 18, pp. 615–632, 1997.

[25] N. J. Higham, "Perturbation theory and backward error for $AX - XB = C$," *BIT Numerical Mathematics*, pp. 124–136, 1993.

[26] C. Kenney, A. J. Laub, and M. Wette, "Error bounds for Newton refinement of solutions to algebraic Riccati equations," *Mathematics of Control, Signals, and Systems*, vol. 3, pp. 211–224, 1990.

[27] L. Keel and S. Bhattacharyya, "Robust, fragile or optimal," *Proc. Amer. Control Conf.*, pp. 1307–1313, 1997.

[28] G. H. Golub and C. F. V. Loan, *Matrix Computations*. JHU Press, 1996.

[29] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[30] M. Dettori, "LMI techniques for control with application to a compact disc player mechanism," *Phd Thesis*, 2001.

[31] B. Shu and B. Bamieh, "Robust H_2 control of vehicular strings," *Submitted to ASME Journal of Dynamics, Meas. and Control.*

[32] E. Shaw and J. K. Hedrick, "String stability analysis for heterogeneous vehicle strings," in *Proc. Amer. Control Conf.*, 2007.

[33] J. K. Rice and M. Verhaegen, "Distributed control: A sequentially semi-separable approach for heterogeneous linear systems," *IEEE Trans. Autom. Control*, Accepted for Publication, temporarily available at http://www.dcsc.tudelft.nl/∼jrice/Publications.htm.

[34] J. D. Gardiner and A. J. Laub, "Generalization of the matrix-sign-function solution for algebraic Riccati equations," *International Journal of Control*, vol. 44, pp. 823–832, 1986.