# Introduction to System Level Synthesis

Marichi Gupta

May 12, 2021

## 1 Introduction

Control theory and the modeling of physical systems has increasing importance in today's world, and there has been a great deal of work over the past several decades to develop rigorous theory for describing such systems. However, such mathematical models often make idealistic assumptions of the system that are not realistic of applicable - for example, assuming there is infinite communication speed or that there is no uncertainty on the state values of the system. Indeed, there is a need to extend control theory results to a wider range of systems: For instance, one wonders how to model distributed systems, where system components are only privy to information from other local system components and where there is a communication delay between all system parts. Coupled with the need to represent such systems accurately is the need to *optimize* over such representations efficiently. While there is substantial theory to account for such complexities of system representation, common optimization techniques break down for especially complicated systems. For instance, Youla Parameterization, first described in [1], offers a useful and common optimization approach by optimizing over the transfer function representation of the controller. However, for particular systems (especially sparse ones), this optimization approach breaks down.

System Level Synthesis (SLS), described at length in [6], [3], offers such a solution. SLS takes a slightly different approach by seeking to optimize over internal components of the controller, rather than over the controller's action on test inputs. With respect to the distributed control problem, SLS offers ways to incorporate the locality constraints of the system (i.e. that system components can only communicate with those close to it) as well as factoring in delays and temporal constraints. The field is a relatively young area of research, with most theoretical development being limited to the past five years or so.

In this term paper we offer a (hopefully) friendly introduction into SLS, assuming basic familiarity with control theory (particularly aimed at those who have taken COMP 590, PIPS, at UNC-CH). We begin by posing the general optimization problem in terms of the input and output of the system, and develop the necessary mathematical theory to do so. With this in hand, we offer some motivation behind SLS and a simple presentation of its formulation.

## 2 Introduction to the Optimal Control Problem

We begin by building up the machinery to state the optimal control problem for linear time-invariant systems (LTIs). We begin by specifying the type of systems that we work with, and will eventually formulate the relation between system input and system output. Since the inputs and outputs will be signals, we will specify the function spaces these signals lie in, and then cast the optimization problem in terms of a map from the input signal to the output. Much of the following exposition mirrors that of Matthew Peet in Lectures 7, 8 of [4].

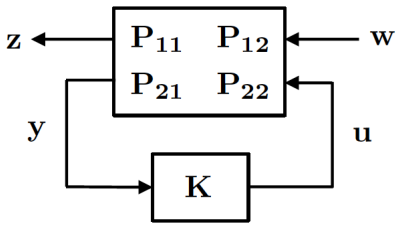


Figure 1: The plant diagram for the systems considered here. Taken from [2], Figure 3.

## 2.1 Two-in-two-out Systems

We consider systems of the following form shown in Figure 1. This system structure is in a slightly different form than the one seen in COMP590. As is familiar, the plant that we model is represented by $P$, and the controller which provides input (that we can specify) to the plant is given by $K$.

There are now two inputs to the system: one is the familiar *control input*, $u$, which the controller tunes and sends to the plant; the other is the *exogenous input*, $w$. The exogenous input is outside of our control, and could represent several things: potentially sensor noise, a tracking signal (for instance, a cruise control setting where a vehicle needs to maintain a desired speed) or other unmodeled dynamics affecting the system.[1]

The outputs are the *regulated output*, $z$, which is the signal we wish to control, and the *sensed output* $y$, which we are able to measure and pass along to the controller. The inputs and outputs may be vector valued, and it is worth noting that there may be overlap in the components between outputs $z$ and $y$, depending on precisely what we able to measure in the output. However, there will be no overlap between inputs $u$ and $w$ since by assumption the exogenous inputs are beyond our control. Generally, we will suppose the controller specifies the control input as

$$u = Ky. \tag{2.1}$$

We are interested now in formulating exactly how the two inputs impact the two output under the aggregate linear system $P$. To this end, we write $P$ as the collection of four subsystems:

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix}. \tag{2.2}$$

With this formulation, we can observe that the $P_{ij}$ are maps with the following actions:

$$P_{11} : w \mapsto z, P_{12} : u \mapsto z, P_{21} : w \mapsto y, \text{ and } P_{22} : u \mapsto y. \tag{2.3}$$

In the case where $u, w, y$, and $z$ are vector-valued, we will call the $P_{ij}$ "multiple-in-multiple-out" (*MIMO*). We call (2.2) the *four system* representation of the overall aggregate system.

We can also write the state-space formulation of the system:

$$\begin{cases} \dot{x} &= Ax + B_1 w + B_2 u \\ z &= C_1 x + D_{11} w + D_{12} u \\ y &= C_2 x + D_{21} w + D_{22} u \end{cases}, \tag{2.4}$$

where $x$ represents the internal state of the plant and $A, B_i, C_j$, and $D_{ij}$ are appropriately shaped matrices. Of course, we also have the more compact representation:

$$\begin{bmatrix} \dot{x} \\ z \\ y \end{bmatrix} = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} x \\ z \\ u \end{bmatrix}. \tag{2.5}$$

[1] It is worth noting that even in COMP590, we have accounted for exogenous inputs in the past, specifically when we took our controller to be of form $u = Ky + Fr$, where $F$ was a feedthrough parameter and $r$ was some reference value. The above presentation simply generalizes this.

One matrix in (2.4) is worth pointing out is $D_{22}$, which we call the *feedforward* matrix since it represents the control input's ($u$) direct impact on the sensor output ($y$).

Closely related to these two representations is the *transfer function* describing a given system, defined as the function $G(s)$ satisfying

$$Y(s) = G(s)U(s), \tag{2.6}$$

where $Y(s)$ and $U(s)$ are the Laplace transforms of time-domain signals $y(t)$ and $u(t)$. In terms of the state-space representation (2.5), aggregating $B_j, C_i$ and $D_{ij}$ into the block matrices $B, C$, and $D$ respectively, we may express the transfer function as

$$G(s) = C(sI - A)^{-1}B + D := \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right], \tag{2.7}$$

where the right-most term of (2.7) denotes a convenient shorthand for $G(s)$ [5]. One notices from (2.7) that $G(s)$ is only defined for appropriate values of $s$. Namely, whenever $s$ is an eigenvalue of $A$, the matrix $(sI - A)$ will not be invertible: we call such $s$ the *poles* of the system. As suggested by the terminology, the $G(s)$ will blow up for such $s$. The transfer function is fundamentally related to the optimal control problem as described later on. It is worth noting that the four-system model admits a transfer function representation as well: indeed, one can write [2]

$$P_{ij} = C_i(sI - A)^{-1}B_j + D_{ij}. \tag{2.8}$$

The three formulations offer different mathematical representations of the system, each of which has its own uses. The four-system approach allows to take a more algebraic approach to understanding the system dynamics: For instance, if two plants $P_a$ and $P_b$ are connected in series within a system block, their combined action on an input will be given by $P_a(P_b(y)) = (P_aP_b)(y)$ (see Chapter 3.4, [5]). The state-space model does not provide this same flexibility and ease of manipulation: instead, one has to trace through the matrix algebra to understand the action on an input. In turn, having explicit matrix representations of the system from the state-space model allows us to optimize over the system using matrix inequalities. Transfer functions over a convenient representation of the system in the frequency domain. indeed, as shown in the action of some system $P$ upon input $u(t)$ may be seen in terms of the multiplication of $u$'s Laplace transform by the system's transfer function $G(s)$. In this way, the transfer function setting offers an algebraic viewpoint to studying systems very similar to the one given by the four-system model, and the two representations can be considered nearly equivalent.

## 2.2 Framing the Optimization Problem

We have mentioned that we have an optimization problem before us, and it is worth specifying what exactly we will optimize. We will choose to optimize some choice of *map* from exogenous input to regulated output. By choosing appropriate metrics (or more formally, norms) to optimize with respect to, we will see that we can achieve certain behaviors of the system for provided exogenous inputs.

We can do some algebra using system blocks from the four-system model to see what this map from $w$ to $z$ looks like more concretely. (The following is adapted from Lecture 8 of [4]). Revisiting (2.2)

$$\begin{cases} z &= P_{11}w + P_{12}u \\ y &= P_{21}w + P_{22}u \\ u &= Ky. \end{cases} \tag{2.9}$$

We seek to remove variables $u, y$ above (i.e., remove the controller input and output variables). Substituting the expression for $u = Ky$ into the expression for $y$ gives

$$y = P_{21}w + P_{22}Ky.$$

Isolating $y$, we obtain

$$\begin{aligned}(I - P_{22}K)y &= P_{21}w \\ \Rightarrow y &= (I - P_{22}K)^{-1}P_{21}w.\end{aligned}$$

Substituting the expressions for $u$ and $y$ into the one for $z$ then gives

$$z = \left[P_{11} + P_{12}K(I - P_{22}K)^{-1}P_{21}\right] w. \tag{2.10}$$

Set the operator $S(K, P)$ to be

$$S(P, K) = P_{11} + P_{12}K(I - P_{22}K)^{-1}P_{21}. \tag{2.11}$$

This is the map from $w$ to $z$ that was desired, and we seek to control its behavior and norm. The map $S(K, P)$ is called a *linear fractional transformation.*[2] We can then frame the central optimization problem as follows:

$$\min_{K \in \text{Admissible Controllers}} \|S(P, K)\|. \tag{2.12}$$

The above raises two questions: what exactly does the space of "Admissible Controllers" look like, what type of norm is chosen for $\|S(P, K)\|$, and why are we minimizing it? To answer these, we develop the notion of the *Hardy Spaces* $\mathcal{H}_2$ and $\mathcal{H}_\infty$, which contain the types of signals that we work with.

## 2.3 The Spaces $\mathcal{H}_\infty$ and $\mathcal{H}_2$.

To begin, we introduce the Lebesgue spaces of square-integrable functions, $L_2$, and of bounded functions, $L_\infty$, loosely following [4], Lecture 7 as well as [5], Chapter 4. Given some set $D \subset \mathbb{R}$, we define $L^2(D)$ to be the set of all functions $f$ that satisfy

$$\|f\|^2_{L^2(D)} := \int_D |f(x)|^2 \, dx < \infty. \tag{2.13}$$

The term denoted $\|f\|_{L^2(D)}$ is the $L^2$ *norm* of $f$, . Often, the set $D$ is taken to be $[0, \infty)$; in this case, one can see that functions lying in $L_2([0, \infty))$ must decay to 0 fast enough. For instance, one can compute that $\int_1^\infty x^{-p} < \infty$ for $p > \frac{1}{2}$, so functions in $L^2([0, \infty))$ must decay faster than $x^{-\frac{1}{2}}$. On the other hand, this means that for some choices of $p$, however, the signal will decay too slowly and the integral describing the $L^2$ norm will not converge. Similarly, we let $L^\infty(D)$ be the set of functions $f$ obeying

$$\|f\|_{L^\infty(D)} := \sup_{x \in D} |f(x)| < \infty. \tag{2.14}$$

We call the above expression the $L^\infty$ *norm* of $f$. Such functions are easy to interpret: they are bounded functions.[3]

The spaces $\mathcal{H}_2$ and $\mathcal{H}_\infty$ are defined analogously. Let $f$ be a complex-valued function: we say $f$ is *analytic* in $\mathbf{C}$ if it is differentiable[4] for all $z \in \mathbb{C}$. In particular, if $f$ is analytic on some set $D$, then $f$ can have no singularities (in particular, no poles) in $D$. Writing $z = \sigma + i\omega$, the requirement that $f$ be

[2]The terminology comes from seeing these as the matrix extension of the one-dimensional maps $x \mapsto \frac{ax+b}{cx+d}$. A minor point: Traditionally, this is called the *Lower Star Product*, and is denoted by $\underline{S}(P, K)$. This suggests the presence of an "Upper Star Product" - indeed, this type of structure is used in Robust Control Theory to also account for the impact of uncertainty in parameters on the system stability. For more details, see Chapter 10 of [5].

[3]When $f$ is matrix-valued, an extension of the $L^\infty$ space requires that $\sup_{x \in D} \overline{\sigma}(f(x)) < \infty$, where $\overline{\sigma}(f(x))$ is the largest-singular value of the matrix-valued function $f$. Of course, when $f$ is scalar-valued, the two are equivalent, and we restrict ourselves to scalar-valued functions here.

[4]Usually, "analytic" pertains to infinitely differentiable functions, or to having a series expansion converging everywhere. But for complex-valued functions these are equivalent to $f$ being once differentiable.

analytic for $\sigma > 0$ dictates that $f$ have no poles in the right-half plane, which has obvious importance when considering these spaces as sets of transfer functions.

We define $\mathcal{H}_2$ as

$$\mathcal{H}_2 = \{f(\sigma + i\omega) : \mathbb{C} \to \mathbb{C} \,:\, f \text{ analytic for } \sigma > 0, \text{ and } \|f(i\omega)\|_{L^2(\mathbb{R})} < \infty\}. \tag{2.15}$$

One observes that functions in $\mathcal{H}_2$ have finite $L^2$ norm along the imaginary axis. Indeed, it turns out that $\mathcal{H}_2$ and $L_2$ are related (in fact, isomorphic) via the Laplace Transform [4]. Namely, we have that $H_2 = \mathcal{L}[L_2([0,\infty)]$ where $\mathcal{L}$ is the Laplace Transform.

Similarly $\mathcal{H}_\infty$ as

$$\mathcal{H}_\infty = \{f(\sigma + i\omega) : \mathbb{C} \to \mathbb{C} \,:\, f \text{ analytic for } \sigma > 0, \text{ and } \|f(i\omega)\|_{L^\infty(\mathbb{R})} < \infty\}. \tag{2.16}$$

We say that $\|G\|_{H_\infty} = \sup_{\omega\in\mathbb{R}} |G(i\omega)|$. Unlike with $\mathcal{H}_2$, there is no isomorphism between $L_\infty$ and $\mathcal{H}_\infty$. Still, the space $\mathcal{H}_\infty$ is an important one: it is the space of transfer functions of bounded linear time-invariant systems.

Indeed, the $\mathcal{H}_\infty$ offers a convenient system norm to apply to the optimization problem stated in 2.12. Given a system $P$ that maps input $u(t)$ to output $y(t) = Pu(t)$, we define the *gain* of $P$ as the relative size, measured in $L^2$, between the input and output signals. We can think of the gain as the following ratio:

$$\frac{\|Pu\|_{L^2}}{\|u\|_{L^2}}.$$

The gain of a system of a natural quantity to control: if a system has a bounded gain, say $M$, then the magnitude of the output signal can be no greater than $M$ times the energy (i.e., $L^2$-norm) of the input signal. This has immediate use if we wish to control the output signal (either in terms of maximum value or asymptotic stability). It turns out that when one considers the transfer function $G(s)$ associated with system $P$, the gain is precisely equal to $\|G\|_{\mathcal{H}_\infty}$. In fact, with a bit more mathematical theory, one can show that $\|G\|_{\mathcal{H}_\infty}$ measures the *operator norm* of the system $P$ [4].

As an aside, a nice way to interpret the $\mathcal{H}_\infty$ and $\mathcal{H}_2$ norms of a signal come from their *Bode plots*, which plot the magnitude of the frequency response of a transfer function. Figure 1 shows such a plot for sample signals; the plot with a notable spike would be optimized in the $\mathcal{H}_2$ norm, while as the other may be optimized by the $\mathcal{H}_\infty$ norm. It is worth noting that $\mathcal{H}_2$ norms are relatively easy to compute (though require the development of the system's Gramian matrices, which we avoid), whereas the $\mathcal{H}_\infty$ is much more difficult to compute, and is often computed (or approximated) using iterative techniques [4].

To ensure that we work only with transfer functions that can be represented in state-space (otherwise shifting back to state-space and optimizing wouldn't work), we restrict ourselves to *rational* transfer functions. Let

$$\mathcal{R} = \left\{ \frac{p(s)}{q(s)} \,:\, p, q \text{ are polynomials} \right\},$$

and then define ([4], Lecture 7)

$$\mathcal{RH}_\infty = \mathcal{R} \cap \mathcal{H}_\infty \text{ and } \mathcal{RH}_2 = \mathcal{R} \cap \mathcal{H}_2.$$

With the above, we can say something very nice about the space $R\mathcal{H}_\infty$: it turns out that $G \in R\mathcal{H}_\infty$ if and only if $G$ is proper (i.e., $\deg(p) \le \deg(q)$) and has no poles on right half plane. When $G$ is *strictly proper*, that is if $\deg(p) < \deg(q)$, we may write $G \in z^{-1}\mathcal{RH}_\infty$. The notation makes some sense in emphasizing that $G(s)$ must decay like $z^{-1}$ for large $s$.

With the above formulation, we can revisit (2.12) and state it more precisely:

$$\min_{K\in\mathcal{H}_\infty} \|S(P,K)\|_{\mathcal{H}_\infty}. \tag{2.17}$$

Recall from the earlier discussion that $S(P,K)$ is precisely the map relating the exogenous input $w(t)$ to the regulated output $z(t)$. To parse the expression into words, we seek to minimize the norm

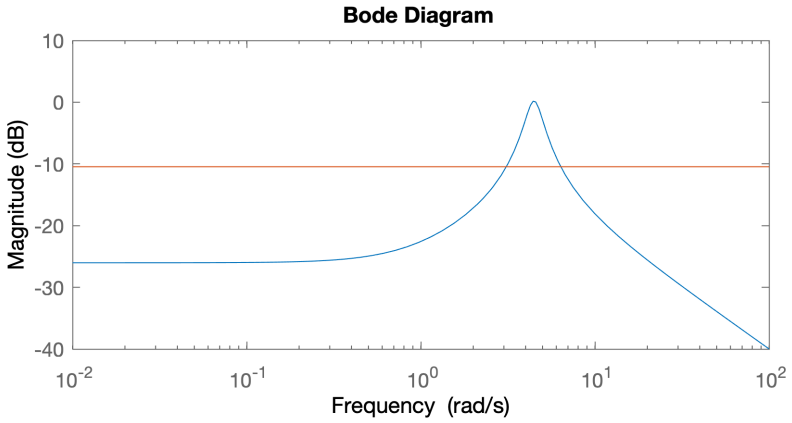


Figure 2: An example of a *Bode Plot*, showing the frequency response for two different signals. The blue signal has lower $\mathcal{H}_2$-norm than the orange signal, and similarly the orange signal has lower $\mathcal{H}_\infty$ norm. Example generated using `bode()` and `tf()` in MATLAB.

$\|S(P,K)\|_{\mathcal{H}_\infty}$, which measures the gain of the system, over the space of all transfer functions that describe bounded linear time-invariant systems. In other words, by controlling $\|S(P,K)\|_{\mathcal{H}_\infty}$ we ensure that even for large $w(t)$, the output $z(t)$ not be too large.

With the optimization problem posed, we now turn to discuss how one goes about solving the optimization problem. The two main approaches mirror our two different representations of the system:

1. One option involves reverting back to the state space representation of the system, and finding the choice of matrices $A, B_j, C_i, D_{ij}$ that minimizes (2.17). The nice thing about this approach is that it is fairly tractable computationally. One theoretical result that enables this approach is the *KYP Lemma* (also known as the *Bounded Real Lemma*), which provides an upper bound on the $\mathcal{H}_\infty$ norm by way of a matrix inequality involving the state-space matrices (see pg. 7, 238 of [5]). For detail on how this is conducted algorithmically, see, for instance, MIT Course Notes available at [11].

2. The other approach involves optimizing over the system/transfer function representation of the system. *Youla Parametrization* is an example of such an approach, detailed further below.

## 2.4 Youla Paramaterization

In broad terms, Youla Parametrization can be thought of as an approach to optimize the system/transfer-function representations of the controller and plant. It makes use of a mathematical trick in using a special matrix factorization of the system plant to recast the optimization problem as one over a convex space, where algorithms perform much better.

We follow the exposition of [6], Section 3.3. Recall the four system representation described in (2.2), and that we wish to minimize (2.17). We say that the matrix describing $P_{22}$ has a "double coprime" factorization if there exist transfer matrices $U_r, V_r, X_r, Y_r, U_l, V_l, X_l, Y_l \in \mathcal{RH}_\infty$ such that

$$P_{22} = V_r U_r^{-1} = U_l^{-1} V_l$$

and

$$\begin{bmatrix} X_l & -Y_l \\ -V_l & U_l \end{bmatrix} \begin{bmatrix} U_r & Y_r \\ V_l & X_l \end{bmatrix} = I.$$

This factorization exists so long as $P_{22}$ is stabilizable[5] and detectable[6] [6]. As for how one practically goes about computing such matrices, see [9], [10] for some programmatic approaches. Assuming one has them in hand, define the following auxiliary matrices

$$T_{11} = P_{11} + P_{12}Y_rU_lP_{21},\ T_{12} = -P_{12}U_r,\ \text{ and } T_{21} = U_lP_{21}.$$

The optimization problem then becomes

$$\min_{Q \in \mathcal{RH}_\infty} \|T_{11} + T_{12}QT_{21}\|_{\mathcal{H}_\infty}. \tag{2.18}$$

The transfer function $Q$ is called the *Youla Parameter*. To see the usefulness of this, compare the expression $T_{11} + T_{12}QT_{21}$ from (2.18) to $S(P,K) = P_{11} + P_{12}K(I - P_{22}K)^{-1}P_{21}$ from (2.11). The principal advantage is that (2.18) is convex with respect to $Q$, whereas (2.11) is not convex with respect to $K$, involving the product of $K$ with a matrix inverse in terms of $K$. For details on computing this parameter (i.e. performing the optimization), [6] points to [7] (see Ch. 15 for especially). Once the parameter $Q$ is found (or approximated), the optimal controller can be taken as $K = (Y_r - U_rQ)(X_r - V_rQ)^{-1}$.

One important note is that to impose additional constraints on viable controllers $K$ (for instance, to incorporate real world communication or information-sharing constraints), we will restrict ourselves to a subspace of controllers - denote this subspace $\mathcal{C} \subset \mathcal{H}_\infty$. Let's revisit the non-linear term from $S(P,K)$, and call it $h(K) = K(I - P_{22}K)^{-1}$. One observes that if it happens that $h(K) = K$, then the expression $S(P,K)$ becomes convex in $K$, and optimization is thus better posed in terms of $K$. This is the motivation of the application of *quadratic invariance* to the optimization problem. We say that a set $\mathcal{M}$ is quadratically invariant with respect to some $B$ (not necessarily in $\mathcal{M}$) if for all $A \in \mathcal{M}$, we have that $ABA \in \mathcal{M}$. It turns out that the subspace $\mathcal{C}$ will be quadratically invariant with respect to $P_{22}$ precisely when $h(S) = S$ [8]. Using the definitions of the coprime factorization matrices, one also always has that $h(S) = (Y_r - U_rQ)U_l$. By way of this relation, it also happens that quadratic invariance of $\mathcal{C}$ under $P_{22}$ is a necessary and sufficient condition for such stabilizing controllers $K \in \mathcal{C}$ to be described by a convex constraint on $Q$ ([6], Section 3.4).

# 3 System Level Synthesis

SLS is described as a framework that overcomes some of the shortcomings of Youla Parameterization, and we can easily consider one such example in light of the above discussion on quadratic invariance. In Section 3.5, [6] raises the issue of controller sparsity violating the quadratic invariance assumption.

For an example, suppose that we have a system of several controllers $\mathbf{K} = K_{ij}$, where $K_{ij}$ is a controller for the $(ij)$th subsystem, which all together control some composite system. It may not be feasible for each controller to share information with one another. Indeed, suppose that the information constraint $\mathcal{C}$ required controllers $K$ to be lower block diagonal matrices.[7] Suppose also that the plant $P_{22}$ is fully connected: recalling that $P_{22}$ maps $u$ to $y$ as described in (2.3), this could represent the case where each controller $K_{ij}$'s control input $u_{ij}$ directly impacts each contoller's sensed output $y_{ij}$. However, under these circumstances, the information constraint $\mathcal{C}$ *will not* be quadratically invariant with respect to $Q_{22}$: this is because there will exist some $\mathbf{K}' \in \mathcal{C}$ where $\mathbf{K}'P_{22}\mathbf{K}' \notin \mathcal{C}$.

A simple computation can give some intuition for why not: suppose that we have the matrix representations $\mathbf{K} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \in \mathcal{C}$, and that $P_{22} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. Direct computation shows that $\mathbf{K}P_{22}\mathbf{K} = \begin{bmatrix} 2 & 1 \\ 4 & 2 \end{bmatrix}$ is not lower-block-triangular and hence does not lie in $\mathcal{C}$.

Since quadratic invariance of $P_{22}$ over the constraint space $\mathcal{C}$ was equivalent to convexity of the Youla Parameter $Q$, one sees from the above example that controller sparsity can turn the optimization over

[5] I.e. if there exists some input such that the resulting system matrix has negative eigenvalues [5].

[6] I.e. if the initial system state $x(0)$ can be reconstructed from the history of $u(t)$ and $y(t)$ [5].

[7] This could represent a system where only one-way communication between the controllers is possible: system controller $K_{mj}$ can receive information from $K_{nj}$ only if $m \leq n$, but otherwise cannot transmit information back.

$Q$ into a non-convex problem, taking us back to where we started. In turn, SLS is posed as a framework that solves this weakness of Youla Parameterization.

## 3.1 Basic Formulation

In this section, we now describe the basics of SLS, following exposition from [6]. We first describe at a high-level the main differences. Youla parameterization (and even optimizing over the explicit state-space matrices as mentioned earlier) optimizes over the controller representation of $K$, by way of finding the appropriate transfer function (or state-space matrices) describing $K$'s action. SLS instead seeks to optimize over the *system responses*, which relate the disturbances on the state vector and regulated output to the state vector and control outputs.

To start, we perform some simple algebraic manipulation similar to what we did in defining $S(P, K)$, this time to write the state $x$ in terms of the exogenous input $w$ and the state-space matrices from (2.4). Let $u = Kx$ describe the controller. Substituting this into (2.4) gives

$$\begin{aligned}\dot{x}(t) &= Ax(t) + B_1 w(t) + B_2 u(t) \\ &= (A + B_2 K)x(t) + B_1 w(t).\end{aligned}$$

Let's shift to the transfer function representation. Taking the Laplace Transform of the above (with zero initial conditions) gives

$$\begin{aligned}sX(s) &= (A + B_2 K)X(s) + B_1 W(s) \\ \Rightarrow X(s) &= (sI - (A + B_2 K))^{-1} B_1 W(s).\end{aligned}$$

Using $U(s) = KX(s)$, this also implies

$$U(s) = K\left(sI - (A + B_2 K)\right)^{-1} B_1 W(s).$$

Let $\delta_x = B_1 W(s)$ be the disturbance of the exogenous input on the state in line with 2.5. We then define the *system response* $\Phi_x : \delta_x \mapsto X(S), \Phi_u : \delta_x \mapsto U(S)$ to be the operators

$$\Phi_x = (sI - (A + B_2 K))^{-1} \text{ and } \Phi_u = K\left(sI - (A + B_2 K)\right)^{-1}. \tag{3.1}$$

Just as we formerly varied over Youla Parameter $Q$ (which was really a transfer function), we now seek optimal choices of controller $K$ by varying $\Phi_x$ and $\Phi_u$. We have the following theorem, which is Theorem 4.1 in [6]

**Theorem 3.1.** *(System Level Synthesis). Suppose that our controller is of form $u = Kx$. Then for the system (2.4), the following are true:*

1. *Solutions in $\Phi_x, \Phi_u \in z^{-1}\mathcal{RH}_\infty$ as given by*

$$\begin{bmatrix}(sI - A) & -B_2\end{bmatrix} \begin{bmatrix}\Phi_x \\ \Phi_u\end{bmatrix} = I \tag{3.2}$$

   *span an affine subspace of $\mathcal{RH}_\infty$, and parameterizes all responses mapping $\delta_x$ to $X(s)$ and $U(s)$.*

2. *The choice of $K = \Phi_u \Phi_x^{-1}$ stabilizes (2.4).*

In the above, recall that $G \in z^{-1}\mathcal{RH}_\infty$ means that $G(s)$ is strictly proper. The proof of the above is provided in [6]. Key takeaways are that we keep track of the internal state $x$ (and its frequency-domain representation) much more closely than in Youla parameterization, and that since the search space for $\Phi_x, \Phi_u$ is an affine set, convex optimization methods are more easily applied. Moreover, [6] emphasizes that focusing on "controller realization", i.e. on $\Phi_x, \Phi_u$, rather than the map $K = \Phi_u \Phi_x$ allows for more flexibility compared to Youla parameterization (see Section 4.1). In other words, it gives us a

more granular representation of the system to work with. With this, the optimization problem from (2.17) becomes, for some objective function $g$,

$$\min_{\Phi_x, \Phi_u} g(\Phi_x, \Phi_u) \tag{3.3}$$

$$\text{provided } \begin{bmatrix}(sI - A) & -B_2\end{bmatrix} \begin{bmatrix}\Phi_x \\ \Phi_u\end{bmatrix} = I, \tag{3.4}$$

$$\Phi_x, \Phi_u \in z^{-1}\mathcal{RH}_\infty \cap \mathcal{S} \tag{3.5}$$

Here, $\mathcal{S}$ is an additional constraint set encapsulating other properties of the system (e.g. sparsity, communication delays, etc). The objective $g$ could be, for instance, the gain of the system in terms of $\Phi_x$ and $\Phi_u$.

### 3.2 Factoring in Constraints

We detail two types of constraints that SLS handles well as discussed in [6], namely *locality constraints* and *temporal constraints*. As mentioned in the example at the start of the section, often we may have systems that have sparse controller representations, potentially due to the presence of distributed subcontrollers that only communicate with a few other controllers that are close to it. The central issue that arose with Youla Parameterization was the need to satisfy a quadratic invariance condition for the Youla space to be convex. Here, such issues are circumvented: the search space for $\Phi_x, \Phi_u$ is affine independent of the controller structure. Section 4.1, Example 1 of [6] offers an explicit example of an optimal control problem that fails for Youla Paramaterization but succeeds with SLS. Generally, such locality restraints are enforced by requiring the transfer function matrices themselves to be sparse (see Definition 2, Section 4.2 of [6]). Further, temporal constraints, such as delays (specifically, actuator, communication, and sensing delays) or the need for a finite impulse response time can be enforced by imposing conditions on the transfer functions describing the systems: see Definition 4, Section 4.2 and discussion thereafter for such conditions on transfer functions for discrete-time systems.

The principal example presented in [6] and [2] demonstrating both spatial and temporal restraints is that of a chain system, where $k$ system nodes are laid out in a chain, and where each node is connected only to those adjacent to it. More precisely, the discrete-time system describing node $x_i$ is [2]

$$x_i[t+1] = \alpha(x_i[t] + \kappa x_{i-1}[t] + \kappa x_{i+1}[t]) + b_i u_i[t] + w_i[t].$$

Figure 3 describes the general set-up: locality constraints are modeled by the horizontal lines and the temporal constraints by the vertical line, and should the system response extend beyond these lines the constraints would be considered violated. Figure 4 shows a numerical example for system dynamics for a chain with and without accounting for delay. Such numerical examples can be further explored using the SLSpy Python package described in [12].

## 4 Current Research in System Level Synthesis

We conclude by briefly reporting on some of the current research being conducted into SLS. One current issue with SLS frameworks, as noted in [6], [2] is that running such optimizations are slow. To address this, [13] describes applying dynamic programming algorithms to more quickly solve the optimization problem, and report solving the problem 4-12 times faster than with standard methods. Additionally on the programmatic side, as mentioned, SLSPy [12] offers a Python toolbox for controller synthesis and design, and emphasizes ease of use and being able to customize workflows. In a slightly different direction, [14] explores a data-driven approach to optimizing SLS systems by analyzing past system trajectories rather than computing the optimal control through a system model, which may be difficult (or impossible) to explicitly develop.

In other work, more tailored to applications, [15] explores the application of SLS for model predictive control (MPC) on large-scale and distributed systems. Such an approach is needed to circumvent locality

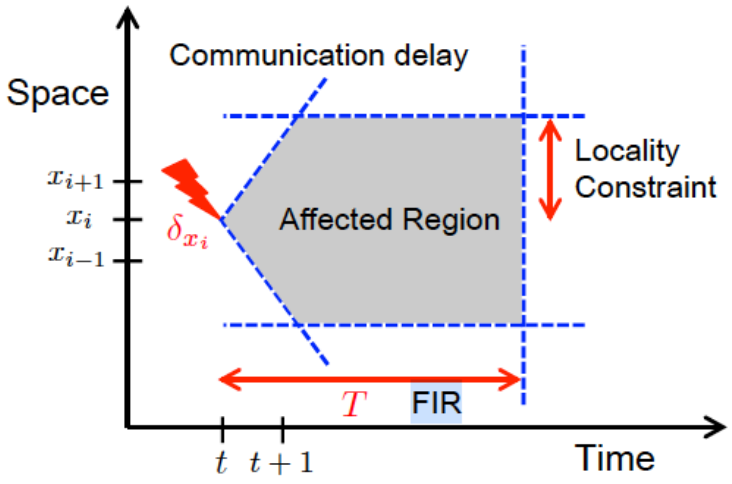


Figure 3: Figure 3 from [6], indicating the space-time diagram for a disturbance propagating through the chain system described. The $x_i$ indicate nodes on the system; the disturbance strikes $x_i$ at time $t$ and propagates with some delay to neighboring vertices. The horizontal dashed lines indicate the locality restraint that the disturbance's effect not spread too far, and the vertical dashed line represent the temporal constraint that the response needs to settle by then.

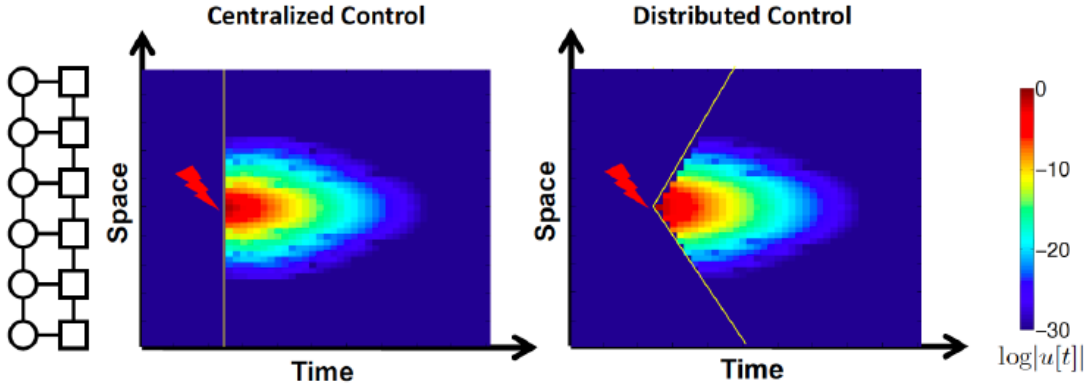


Figure 4: Figure 8 from [2]. The left panel models the system response assuming zero communication delay, showing that neighboring nodes, despite being steps away from the disturbance, respond immediately. The right panel models some communication delay (modeled by the yellow line) from the nodes sensing the disturbance, computing the input, and actuating it. No locality or temporal constraints are applied here.

conditions and information restraints as imposed by the system architecture. If one has full knowledge of the system and friendly constraints to work with, one theoretically could compute in advance the ideal stabilizing controller; however, such controllers may have idealistic assumptions or be difficult (i.e. NP-hard) to compute. Along these lines, [15] describes how to parametrize the closed loop controller and MPC problem into a convex, structured one, where controllers only need to make use of local controller information, which respects the distributed nature of the system. Still, there is large room for further research into the application of SLS-based controllers.

# References


[1] D. C. Youla, H. A. Jabr, and J. J. B. Jr., "Modern wiener-hopf design of optimal controllers Part II: The multivariable case," IEEE Transactions on Automatic Control, 1976.

[2] J. C. Doyle, N. Matni, Y. Wang, J. Anderson and S. Low, "System level synthesis: A tutorial," 2017 IEEE 56th Annual Conference on Decision and Control (CDC), 2017, pp. 2856-2867, doi: 10.1109/CDC.2017.8264074.

[3] Y. -S. Wang, N. Matni and J. C. Doyle, "A System-Level Approach to Controller Synthesis," in IEEE Transactions on Automatic Control, vol. 64, no. 10, pp. 4079-4093, Oct. 2019, doi: 10.1109/TAC.2018.2890753.

[4] M. Peet, "A Course on LMIs in Systems and Control." Lecture videos for MAE598 at ASU. https://www.youtube.com/playlist?list=PL5ebyVGQORm6n158o-I_liUZ7Q5Od43li.

[5] K. Zhou, and J. C. Doyle. "Essentials of Robust Control", Prentice Hall, 1998.

[6] J. Anderson, J. C. Doyle, S. H. Low, N. Matni, "System level synthesis," Annual Reviews in Control, vol. 47, 2019, pp 364-393. https://doi.org/10.1016/j.arcontrol.2019.03.006.

[7] S. Boyd and C. Barratt, "Linear Controller Design: Limits of Performance," Prentice Hall, 1991. https://web.stanford.edu/ boyd/lcdbook/lcdbook.pdf.

[8] L. Lessard and S. Lall, "Convexity of Decentralized Controller Synthesis," IEEE Transactions on Automatic Control, Vol. 61, No. 10, 2016.

[9] K. Sugimoto and Y. Yamamoto. "A polynomial matrix method for computing stable rational doubly coprime factorizations," Systems & Control Letters, 1990, Volume 14, Issue 3, pp. 267-273. https://doi.org/10.1016/0167-6911(90)90023-N.

[10] A. Varga, "Computation of coprime factorizations of rational matrices," Linear Algebra and its Applications, 1998, Volume 271, Issues 1-3, pp. 83-115. https://doi.org/10.1016/S0024-3795(97)00256-5.

[11] A. Megretski, "Algorithms for H-Infinity Optimization," MIT Lecture Notes for 6.245, Multivariable Control Systems, 2004. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-245-multivariable-control-systems-spring-2004/lecture-notes/lec7_6245_2004.pdf.

[12] S-H Tseng and J. S. Li, "SLSpy: Python-Based System-Level Controller Synthesis Framework", Preprint 2020. https://arxiv.org/abs/2004.12565.

[13] S-H Tseng, C. A. Alonso, and S. Han, "System Level Synthesis via Dynamic Programming," Proc. IEEE CDC, 2020.

[14] A. Xue and N. Matni, "Data-Driven System Level Synthesis," Preprint 2021. https://arxiv.org/abs/2011.10674.

[15] C. A. Alonso, N. Matni and J. Anderson, "Explicit Distributed and Localized Model Predictive Control via System Level Synthesis," 2020 59th IEEE Conference on Decision and Control (CDC), 2020, pp. 5606-5613, doi: 10.1109/CDC42340.2020.9304349.