Squaring down of general MIMO systems to invertible uniform rank systems via pre- and/or post-compensators

Peddapullaiah Sannuti\(^1\), Ali Saberi\(^2\), Meirong Zhang\(^3\)

Pre- and/or post-compensators, post-compensators, and both pre- and post-compensators are designed to square down a general linear multivariable system to a uniform rank system.

\(^1\) sannuti@ece.rutgers.edu
Department of Electrical and Computer Engineering, Rutgers University
94 Brett Road, Piscataway, NJ 08854-8058, U.S.A.

\(^2\) saberi@eecs.wsu.edu
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752, U.S.A.

\(^3\) meirong.zhang@email.wsu.edu
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752, U.S.A.
Chapter 1

Introduction and problem statement

Structural properties of single-input single-output (SISO) systems are easy to decipher in comparison with multi-input multi-output (MIMO) systems. It is a common knowledge that MIMO systems are structurally much more complex than SISO systems. Because of this, there have been attempts in the past to simplify the complex structure of general MIMO systems by transforming them to a particular class of MIMO systems known as uniform rank systems, which are structurally similar to SISO systems, [9] and [4, 5]. (Such a process of transforming is termed as squaring down process.) These attempts utilize state feedback, and integrators either at the input side or at the output side or at both the input and output sides in an appropriate coordinate system such as the special coordinate basis (SCB) as developed in [11]. Although certain applications allow state feedback to square down MIMO systems, squaring down without the use of state feedback is significant in many fields, including decentralized control [14], synchronization of multi-agent systems where the agents are non-introspective, see for example [4, 5]. This calls for designing just either pre-compensators or post-compensators or both pre- and post-compensators to square down general MIMO systems (see Figure 1.1). We emphasize that when state feedback is not allowed, the said squaring down problem becomes complex.

In order to rationalize the squaring down problem, we need to understand and compare the structural differences between the general MIMO systems, uniform rank systems, and SISO systems. Since SISO systems are square and invertible, for simplicity, let us first focus only on square invertible MIMO systems. Both SISO systems and MIMO systems contain the dynamics associated with finite and infinite zeros. In an appropriate coordinate basis, the dynamics associated with finite zeros (finite zero structure) and that associated with infinite zeros (infinite zero structure) can be delineated. The dynamics associated with finite zeros is obviously referred to as finite zero dynamics. Such finite zero dynamics associated with MIMO systems and with SISO systems contain basi-
ally the same features. However, the dynamics associated with infinite zeros of MIMO systems and with SISO systems are profoundly different. As such, the structural complexity of MIMO systems as compared to SISO systems arises due to its infinite zero structure. For the benefit of readers, let us recall the definition of infinite zero structure of general square invertible MIMO systems. In a seminal paper [6], Morse discusses the infinite zero structure by introducing what is known as a \( C^* \) structural invariant indices list \( I_4 \). This list is invariant with respect to output injection, state feedback, and nonsingular transformations of state, input, and output. For strictly proper square invertible systems, the list \( I_4 \) contains several subsets, a subset of \( n_1 \) integers all equal to 1, a subset of \( n_2 \) integers all equal to 2, \( \cdots \), and finally a subset of \( n_k \) integers all equal to \( k \) for a certain \( k \). That is,

\[
I_4 = \{ 1, 1, \ldots, 1, 2, 2, \ldots, 2, \ldots, k, k, \ldots, k \}.
\]

(For non-strictly proper systems, another sublist of the type \{ 0, 0, \ldots, 0 \} needs to be included in the above set. For clarity of presentation, we consider strictly proper case.) The total number of integers in \( I_4 \) as expressed above equals the number of inputs as well as the number of outputs.\(^1\) In contrast to the above, the
\( I_4 \) for a SISO system is a single integer, namely its relative degree \( r \), \( I_4 = \{ r \} \). The \( I_4 \) for a uniform rank system is of the form,

\[
I_4 = \{ r, r, \ldots, r \},
\]

for some integer \( r \). That is, the infinite zero structure of a uniform rank system contains \( n_r \) integers all equal to \( r \). In this sense, uniform rank systems are similar to SISO systems. While \( I_4 \) of a SISO system is only a single integer, \( I_4 \) of a uniform rank system is again a single integer \( r \) but repeats itself \( n_r \) times. This is unlike several integers repeating themselves many times for a general MIMO system.

The infinite zero structure represented by \( I_4 \) has several interpretations. We quote below two such interpretations. The first one interprets \( I_4 \) as the number of inherent integrator chains between certain components of input \( u \) and output \( y \) of a square invertible MIMO system \( \Sigma \). In an appropriate state, input, and output coordinate basis, we can decompose the input \( u \) and the output \( y \) of \( \Sigma \) into certain components,

\[
u = \text{col}(u_1, u_2, \cdots, u_k), \quad y = \text{col}(y_1, y_2, \cdots, y_k).
\]

Notationally, we dedenote by \( \text{col}(z_1, z_2, \cdots, z_m) \) the column vector obtained by stacking the vectors \( z_1, z_2, \cdots, z_m \) in that order. For each \( i, 1 \leq i \leq k \), the dimensions of \( u_i \) and \( y_i \) are exactly the same and equal \( n_i \). There exists a chain of \( i \) integrator blocks from the input vector \( u_i \) to the output vector \( y_i \). Each integrator block contains \( n_i \) scalar integrators corresponding to the dimension of \( u_i \) as well as \( y_i \). Let us emphasize again that there are \( i \) integrator blocks in the chain, each block containing \( n_i \) scalar integrators. This signifies the meaning behind the invariant subset \( \{ i, i, \ldots, i \} \) of \( I_4 \).

We need to introduce now certain terminology. The subset \( \{ i, i, \ldots, i \} \) of \( I_4 \) is termed as the \( i \)-th order infinite zeros; the dynamics associated with it is termed as the \( i \)-th order infinite zero dynamics. We observe that there are \( in_i \) number of poles associated with the \( i \)-th order infinite zero dynamics.

The second interpretation of infinite zero structure is a multi-variable root-loci interpretation. In this regard, we mention here the results of [7, 13] which show that, under high gain feedback, the \( in_i \) number of poles associated with the \( i \)-th order infinite zero dynamics form a certain \( i \)-th order root loci that tend to infinity as the gain tends to infinity.

The above discussion and inherent integration interpretation of the infinite zero structure of a MIMO system and that of a uniform rank system lead us to visualize graphically the structures of square invertible MIMO systems and uniform rank systems as depicted in Figures 1.2 and 1.3 respectively. These figures show clearly the fundamental difference between them, namely, while a MIMO system exhibits several orders of infinite zero dynamics, the uniform number of inputs or the the number of outputs whichever is smaller.
rank system has only one order of infinite zero dynamics (say, \(r\)-th order for some integer \(r\)).

Now that the structural differences between a MIMO system and a uniform rank system are clear and transparent, we can formally state the general squaring down problem as described below. Consider a general linear time invariant MIMO system,

\[
\Sigma_{\text{plant}}: \begin{cases}
\dot{x} = Ax + Bu \\
y = Cx,
\end{cases}
\]

where \(x \in \mathbb{R}^n, \ u \in \mathbb{R}^m\), and \(y \in \mathbb{R}^p\) are respectively state, input, and output of the system \(\Sigma_{\text{plant}}\), and \(\dot{x}\) denotes \(\frac{dx}{dt}\).

**Problem Statements:** Consider a general MIMO system \(\Sigma_{\text{plant}}\) as described in (1.1). Let \(k\) be the highest order of infinite zeros of \(\Sigma_{\text{plant}}\), that is, the list 

\[I_4 = \{n_1, 1, \ldots, 1, n_2, 2, \ldots, 2, \ldots, k, k, \ldots, k\}.\]

Consider a goal as follows:

**Goal:** Square down \(\Sigma_{\text{plant}}\) to an invertible uniform rank system with the list \(I_4\) as

\[I_4 = \{k, k, \ldots, k\},\]

and that the squared down system contains the invariant zeros of the given system \(\Sigma_{\text{plant}}\) and possibly additional invariant zeros which can be freely assigned in the open left half \(s\) plane.

We have the following three problem statements:

1. Design a pre-compensator \(\Sigma_{\text{Pre}}\) whenever possible such that the cascade connection of \(\Sigma_{\text{Pre}}\) and \(\Sigma_{\text{plant}}\) in that order as depicted in Figure 1.1 meets the goal stated above.

2. Design a post-compensator \(\Sigma_{\text{Post}}\) whenever possible such that a cascade connection of \(\Sigma_{\text{plant}}\) and \(\Sigma_{\text{Post}}\) in that order as depicted in Figure 1.1 meets the goal stated above.

3. Design a pre-compensator \(\Sigma_{\text{Pre}}\) and a post-compensator \(\Sigma_{\text{Post}}\) such that a cascade connection of \(\Sigma_{\text{Pre}}, \Sigma_{\text{plant}}, \text{ and } \Sigma_{\text{Post}}\) in that order as depicted in Figure 1.1 meets the goal stated above.

It is clear that squaring down a general MIMO system to a uniform rank system requires a careful structural examination of the infinite zero structure of the given MIMO system. We utilize here the powerful tool of SCB that captures all the inherent structures of a MIMO system. Since its introduction, SCB has been used as a tool in a large body of research, for details, we refer to the recent books [2, 3, 12] and the references there in as well as to some articles [1, 8, 10, 13].

The algorithmic procedures developed here to square down a given system to a uniform rank system are presented for continuous-time systems, however the same procedures can be utilized with appropriate modifications for discrete-time systems as well.
Figure 1.2: MIMO system - SCB format
Figure 1.3: Uniform rank system - SCB format

\[ a_{r,0}x_0 + \sum_{i=1}^{r} a_{r,r,i}x_{r,i} \]

Finite zero structure

Subsystem with infinite zero structure of order \( r \)
Chapter 2

Results

We have the following results.

**Theorem 1** Consider a general MIMO system $\Sigma_{\text{plant}}$ as described in (1.1) with its highest order of infinite zeros equal to $k$. Let $\Sigma_{\text{plant}}$ be stabilizable and detectable. We have the following statements:

1. There exists a post-compensator that squares down a left invertible system $\Sigma_{\text{plant}}$ to an invertible uniform rank system with its order of infinite zeros equal to $k$ and meets the goal stated in the problem statement.

2. There exists a pre-compensator that squares down a right invertible system $\Sigma_{\text{plant}}$ to an invertible uniform rank system with its order of infinite zeros equal to $k$ and meets the goal stated in the problem statement.

3. There exists a combination of pre- and post-compensators that square down a general system $\Sigma_{\text{plant}}$ to an invertible uniform rank system with its order of infinite zeros equal to $k$ and meets the goal stated in the problem statement.

**Remark 1** We consider here only strictly proper systems. The results can be extended trivially to the non-strictly proper system. For non-strictly proper case, the list of $\mathbb{I}_4$ includes another sublist of the type $\{0,0,\ldots,0\}$ which can be taken into account in a straightforward way. However, this adds another layer of notational complexity. For clarity of presentation, we consider only strictly proper systems.

**Remark 2** In addition to being stabilizable and detectable, if the given system $\Sigma_{\text{plant}}$ is of minimum phase, then the squared down systems, the cascade of $\Sigma_{\text{pre}}$ and $\Sigma_{\text{plant}}$, the cascade of $\Sigma_{\text{plant}}$ and $\Sigma_{\text{post}}$, and the cascade of $\Sigma_{\text{pre}}$, $\Sigma_{\text{plant}}$, and $\Sigma_{\text{post}}$ are also stabilizable, detectable, and of minimum phase.

The proof of the above theorem is by explicit construction of pre- and post-compensators that accomplish the tasks itemized in the theorem. There are
two phases to such a construction. In the first phase, a given general linear
time-invariant MIMO system is squared down to an invertible square MIMO
system, while in the second phase an invertible square MIMO system is squared
down to an invertible uniform rank system.

Towards the goal of first phase, let us recall the squaring down results de-
veloped in [10]:
(1). A general left invertible system $\Sigma_{plant}$ which is stabilizable and detectable
can be squared down to an invertible square system via a post-compensator.
(2). A general right invertible system $\Sigma_{plant}$ which is stabilizable and detectable
can be squared down to an invertible square system via a pre-compensator.
(3). A general neither a left invertible nor a right invertible system $\Sigma_{plant}$ which
is stabilizable and detectable can be squared down to an invertible square sys-
tem via the use of both pre- as well as post-compensators.
The above squaring down procedure introduces additional invariant zeros be-
yond those the given system $\Sigma_{plant}$ possesses. However, all such additional
invariant zeros can be freely assigned in the open left half $s$ plane.

These results accomplish the goal of the first phase. In view of this, we need
to concentrate only on the second and crucial phase of constructing pre- and
post-compensators to square down a general invertible square MIMO system
to an invertible uniform rank system such that the constructed pre- and post-
compensators preserve the invariant zero structure of the considered invertible
square MIMO system. Chapter 3 focuses on constructing a pre-compensator
that accomplished the said goal, while Chapter 4 focuses on constructing a
post-compensator.
Chapter 3

Pre-compensator design

The goal of this chapter is to construct a pre-compensator that squares down a general linear invertible multi-input multi-out system to a uniform rank system. Our development to do so is divided into several sections. Section 3.1 details the description of a general square invertible system in a SCB format, while Section 3.2 does the same for a uniform rank system. Section 3.3 describes how the construction of a pre-compensator can be sub-divided into the construction of a number of simple pre-compensators, each one of which can be constructed by an algorithm termed as \( P \)-algorithm given in Section 3.4. The data required for the construction of each of simple pre-compensator is obtained in Section 3.5. Finally, Section 3.6 presents a numerical example.

3.1 A general invertible system in SCB format

It is clear from introduction that squaring down a general MIMO system to a uniform rank system requires a careful structural decomposition of finite and infinite zero structures of the given MIMO system. Such a decomposition is captured by the special coordinate basis (SCB) as developed in [11]. We recall below the SCB for square invertible systems by considering a system \( \Sigma \) with the highest order of infinite zeros equal to \( k \). As depicted in Figure 1.2 such a system \( \Sigma \) can be decomposed into \( k + 1 \) systems, \( \Sigma_0 \) and \( \Sigma_i, \ i = 1 \) to \( k \). Here \( \Sigma_0 \) represents the zero dynamics, while \( \Sigma_i, \ i = 1 \) to \( k \) represents the dynamics of \( i \)-th order infinite zeros. Thus, the dynamics of the given system \( \Sigma \) in SCB format can be described as follows where \( g_i(x) \) for an integer \( i \) denotes linear functions of all state variables \( x \):

\[
\Sigma_0 : \{ \dot{x}_0 = A_0 x_0 + \sum_{i=1}^{k} A_{0,i} y_i, \quad \text{Invariant Zero dynamics} \} \\
\Sigma_1 : \{ \dot{x}_{1,1} = g_1(x) + u_1, \quad \text{First order infinite zero dynamics} \}
\]  

(3.1a)  

(3.1b)
\[ \Sigma_2 : \begin{cases} \dot{x}_{2,1} &= x_{2,2} + \nu_{2,1,1} x_{1,1} \\ x_{2,2} &= g_2(x) + u_2, \quad \text{(Second order infinite zero dynamics)} \end{cases} \]

\[ \Sigma_3 : \begin{cases} \dot{x}_{3,1} &= x_{3,2} + \nu_{3,1,1} x_{1,1} + \nu_{3,1,2} x_{2,1} \\ \dot{x}_{3,2} &= x_{3,3} + \nu_{3,2,1} x_{1,1} + \nu_{3,2,2} x_{2,1} \\ \dot{x}_{3,3} &= g_3(x) + u_3, \quad \text{(Third order infinite zero dynamics)} \end{cases} \]

\[ \vdots \]

\[ \begin{cases} \dot{x}_{i,1} &= x_{i,2} + \sum_{j=1}^{i-1} \nu_{i,1,j} x_{j,1} \\ \dot{x}_{i,2} &= x_{i,3} + \sum_{j=1}^{i-1} \nu_{i,2,j} x_{j,1} \\ \ddots \\ \dot{x}_{i,i-1} &= x_{i,i} + \sum_{j=1}^{i-1} \nu_{i,i-1,j} x_{j,1} \\ \dot{x}_{i,i} &= g_i(x) + u_i, \quad \text{\textbf{(i-th order infinite zero dynamics)}} \end{cases} \]

\[ \vdots \]

\[ \begin{cases} \dot{x}_{k,1} &= x_{k,2} + \sum_{j=1}^{k-1} \nu_{k,1,j} x_{j,1} \\ \dot{x}_{k,2} &= x_{k,3} + \sum_{j=1}^{k-1} \nu_{k,2,j} x_{j,1} \\ \ddots \\ \dot{x}_{k,k-1} &= x_{k,k} + \sum_{j=1}^{k-1} \nu_{k,k-1,j} x_{j,1} \\ \dot{x}_{k,k} &= g_k(x) + u_k, \quad \text{\textbf{(k-th order infinite zero dynamics)}} \end{cases} \]

The state \( x \) of the given system \( \Sigma \) and its states are defined as

\[
x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix}, \quad x_1 = [x_{1,1}], \quad x_2 = \begin{bmatrix} x_{2,1} \\ x_{2,2} \end{bmatrix}, \quad x_i = \begin{bmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,i} \end{bmatrix}, \quad x_k = \begin{bmatrix} x_{k,1} \\ x_{k,2} \\ \vdots \\ x_{k,k} \end{bmatrix}.
\]

Notationally, the states \( x_0 \), and \( x_i, \: i = 1 \text{ to } k \) pertain respectively to the dynamics of invariant zeros, and \( i \)-th order infinite zeros. Their dimensions are respectively \( n_0 \), and \( n_{i,j}, \: i = 1 \text{ to } k \). Each state \( x_i \) has \( i \) components denoted by \( x_{i,j}, \: j = 1 \text{ to } i \). The dimension of each \( x_{i,j} \) is \( n_i \). In the notation \( x_{i,j} \), the first subscript \( i \) always refers to the order of infinite zeros, and the second subscript \( j \) refers to a particular component of \( x_i \) with \( j = i \) being its last component. A word about the subscript notation of the coefficient matrix \( \nu_{i,j,m} \) is in order. In general, the subscripts \( i \) and \( j \) in \( \nu_{i,j,m} \) refer to the dynamic equation of the state \( x_{i,j} \) in which \( \nu_{i,j,m} \) appears, and the subscript \( m \) refers to the output component \( y_m \) to which the coefficient belongs.
The input \( u \) and the output \( y \) are defined as
\[
\begin{bmatrix}
u_1 \\ u_2 \\ \vdots \\ u_k
\end{bmatrix},
\begin{bmatrix}
y_1 \\ y_2 \\ \vdots \\ y_k
\end{bmatrix}
\begin{bmatrix}
x_{1,1} \\ x_{2,1} \\ \vdots \\ x_{k,1}
\end{bmatrix}.
\]

Notationally, \( u_i \) is the input to the dynamics of \( i \)-th order infinite zeros, while the first component \( x_{i,1} \) is the output \( y_i \) of \( i \)-th order infinite zero chain. The dimension of both \( u_i \) and \( y_i \) is \( n_i \).

**Remark 3** We observe that the dynamics associated with different orders of infinite zeros are intertwined. Such an interweaving occurs in two ways. The first one is output injection through the coefficients of the type \( \nu_{i,j,m} \) in equation (3.1), while the second one is through the linear functions of the type \( g_i(x) \) which are additive to the input \( u_i \) for \( i = 1 \) to \( k \).

### 3.2 A general invertible uniform rank system in SCB format

As depicted in Figure 1.3, an invertible uniform rank system \( \Sigma_w \) in its SCB format consists of two systems \( \Sigma_{w0} \) and \( \Sigma_{wk} \) of the form,
\[
\begin{align*}
\Sigma_{w0} & : \begin{cases}
iw_0 = A_0w_0 + A_0y_w, & \text{(Invariant Zero dynamics)} \\
\end{cases} \\
\Sigma_{wk} & : \begin{cases}
iw_j = w_{j+1}, & \text{for } j = 1, 2, \ldots, k-1, \\
iw_k = g(w) + u_w, & \text{(k-th order infinite zero dynamics)}
\end{cases}
\end{align*}
\]

with \( y_w := w_1 \) considered as its output. The integer \( k \) here denotes the order of infinite zeros of \( \Sigma_w \). Also, \( w_i, i = 0, 1, 2, \ldots, k \) are the states, \( y_w \) is the output, and \( u_w \) is the input of \( \Sigma_w \). Furthermore, \( g(w) \) denotes a linear function of all state variables \( w_i, i = 0, 1, 2, \ldots, k \).

We observe that, in the format of SCB, \( \Sigma_w \) consists of \( \Sigma_{w0} \) representing its finite zero structure and \( \Sigma_{wk} \) representing its infinite zero structure. We note further that \( \Sigma_{w0} \) is fed only by the output of \( \Sigma_{wk} \). We note also that the infinite zero structure \( \Sigma_{wk} \) consists of \( k \) integrator blocks forming a chain; the output of first integrator is its output as well as the output of \( \Sigma_w \), while the last integrator is fed by a linear combinations of all the states and the input of \( \Sigma_w \). We note also that each integrator block in the chain contains \( n_k \) scalar integrators.

### 3.3 A general procedure of transforming any \( \Sigma_i \) to the corresponding uniform rank system \( \Sigma_{wi} \) in SCB format

In what follows, our goal is to design a pre-compensator, denoted by \( P_i \), to square down \( \Sigma_i, i = 1 \) to \( k-1 \), to a uniform rank system \( \Sigma_{wi} \) in SCB format.
such that its order of infinite zeros is $k$.

There are two phases in accomplishing our goal. We observe from equation (3.1) that the dynamic equations of $\Sigma_i$, for each $i = 2$ to $k$, are not in the form of a uniform rank system in SCB format as given in equation (3.2) in Section 3.2 because of the presence of output injection through the coefficients of the type $\nu_{i,j,m}$. The first phase consists of rewriting such dynamic equations in the SCB format by an appropriate selection of state variables. A successful completion of this phase gives rise to a uniform rank system $\tilde{\Sigma}_i,i$ in SCB format whose order of infinite zeros is the same as that of the given system $\Sigma_i$, namely $i$.

The second phase consists of designing the pre-compensator $P_i$ that enables squaring down $\tilde{\Sigma}_i,i$ to $\Sigma_{wi}$. This is done step by step. That is, we decompose $P_i$ as a cascade of $k - i$ simple pre-compensators as,

$$P_i = P_{i,k-1} P_{i,k-2} \cdots P_{i,i+1} P_{i,i}. \quad (3.3)$$

Each simple pre-compensator $P_{i,j}$, $j = i$ to $k - 1$, is designed by following what is termed as $P$-algorithm that is developed soon in Section 3.4. The goal of each simple pre-compensator $P_{i,j}$ is to enable increasing the order of infinite zeros of a certain given system by a value $1$. To be precise, let us define

$$\tilde{P}_{i,i} = P_{i,i}, \quad (3.4)$$

and for $i + 1 \leq j \leq k - 1$,

$$\tilde{P}_{i,j} = P_{i,j} P_{i,j-1} \cdots P_{i,i+1} P_{i,i}. \quad (3.5)$$

Let us also denote, for $i + 1 \leq j \leq k - 1$,

$$\tilde{\Sigma}_{i,j+1} = \tilde{P}_{i,j} \tilde{\Sigma}_{i,i}. \quad (3.6)$$

The above notations allow us to explain the step by step design procedure we envision. We design first $P_{i,i}$ such that the system

$$\tilde{\Sigma}_{i,i+1} := \tilde{P}_{i,i} \tilde{\Sigma}_{i,i} := P_{i,i} \tilde{\Sigma}_{i,i}$$

is a uniform rank system in SCB format with its order of infinite zeros equal to $i + 1$. We design next $P_{i,i+1}$ such that the system

$$\tilde{\Sigma}_{i,i+2} := \tilde{P}_{i,i+1} \tilde{\Sigma}_{i,i} := P_{i,i+1} P_{i,i} \tilde{\Sigma}_{i,i}$$

is a uniform rank system in SCB format with its order of infinite zeros equal to $i + 2$. We continue this iterative procedure until we design $P_{i,k-1}$ such that $\tilde{\Sigma}_{i,k}$ is a uniform rank system in SCB format with its order of infinite zeros equal to $k$. By our design, $\tilde{\Sigma}_{i,k}$ is indeed $\Sigma_{wi}$ and $P_i$ is indeed $\tilde{P}_{i,k-1}$.
3.4 A building block of pre-compensator design

In this section, we design each $P_{i,j}$ given in (3.3) step by step starting from $j = i$. To do so, an algorithm termed as $P$-algorithm is presented next.

As a preliminary to the constructive details of $P$-algorithm, we state first its objective. Consider the last equation of $\dot{\Sigma}_{i,j}$ for $i = 1$ to $k - 1$ and $i \leq j \leq k - 1$,

$$\dot{w}_{i,j} = g_{i,j}(x) + f_{i,j}(w) + x_{p_{i,j}}, \quad (3.7)$$

where $g_{i,j}(x)$ is a linear function of all states of $x$, $x_{p_{i,j}}$ denotes an input variable which turns out to be the state of a simple pre-compensator $P_{i,j}$, $w_{i,j}$ is a substate variable, and the function $f_{i,j}(w)$ has a certain structure, namely it is a linear function of all variables $w_{p,q}$ for $p < i$, $q \leq j$.

Remark 4 As will be evident shortly and in subsequent sections, the function $f_{i,j}(w)$ is identically zero for $i = 1$. For the sake of generality of the algorithm, we just keep the presence of $f_{i,j}(w)$ in the algorithm. The expression for $f_{i,i}(w)$ and the expression for each $g_{i,i}(x)$ are derived in the subsequent section that pertains to phase 1. Based on $g_{i,i}(x)$ and $f_{i,i}(w)$, the algorithm developed below will generate recursively $g_{i,j}(x)$ and $f_{i,j}(w)$ for $i < j \leq k - 1$.

Define $w_{i,j+1}$ as

$$w_{i,j+1} := \dot{w}_{i,j} = g_{i,j}(x) + f_{i,j}(w) + x_{p_{i,j}}. \quad (3.8)$$

Given $g_{i,j}(x)$ and $f_{i,j}(w)$, the goal of $P$-algorithm is to design the simple pre-compensator $P_{i,j}$ containing one block of integral operation such that the cascade connection of $P_{i,j}$ and the subsystem with its dynamic equation as (3.7) is of the form,

$$\dot{w}_{i,j} = w_{i,j+1}$$
$$\dot{w}_{i,j+1} = g_{i,j+1}(x) + f_{i,j+1}(w) + x_{p_{i,j+1}}, \quad (3.9)$$

for an appropriately defined linear functions $g_{i,j+1}(x)$ and $f_{i,j+1}(w)$, and a new input variable $x_{p_{i,j+1}}$.

Remark 5 Equations (3.7), (3.8), and (3.9) imply that the goal of the $P$-algorithm developed below that constructs the pre-compensator $P_{i,j}$ is two fold, to generate the input variable $x_{p_{i,j}}$ to $\dot{w}_{i,j}$ as shown in (3.8) and to generate the linear functions $g_{i,j+1}(x)$ and $f_{i,j+1}(w)$ which define the dynamics of $\dot{w}_{i,j+1}$ as shown in (3.9) where $x_{p_{i,j+1}}$ can be considered as a new input variable to the pre-compensator $P_{i,j}$. Towards accomplishing such a dual goal, we envision the structure of pre-compensator $P_{i,j}$ as shown in Figure 3.1.

Construction of $P$-algorithm that designs $P_{i,j}$:
As input data to this algorithm, we are given the linear functions $g_{i,j}(x)$ and $f_{i,j}(w)$. We need to design the input variable $x_{pi,j}$ such that we get the system in (3.9). We do so in three stages:

**Stage a:** In this stage, we decompose the given linear function $g_{i,j}(x)$ into two appropriately defined linear functions. Let us first define the following:

$$
\tilde{x}_0 = x_0, \quad x_r = \begin{bmatrix} x_r,1 \\ x_r,2 \\ \vdots \\ x_{r,r-1} \end{bmatrix}, \quad \bar{x}_r = \begin{bmatrix} \tilde{x}_0 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_k \end{bmatrix} \quad \text{for } 2 \leq r \leq k, \quad \bar{x} = \begin{bmatrix} \tilde{x}_0 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_k \end{bmatrix}.
$$

For each $1 < r \leq k$, we observe that $\tilde{x}_r$ is obtained from $x_r$ by omitting its last element, namely $x_{r,r}$. Note that $\tilde{x}_1$ does not exist. Then, it is easy to see that

$$
g_{i,j}(x) = \tilde{g}_{i,j}(\bar{x}) + \sum_{r=1}^{k} \alpha_{i,j,r} x_{r,r},
$$

for an appropriately defined linear function $\tilde{g}_{i,j}(\bar{x})$ and coefficients $\alpha_{i,j,r}$. We can then rewrite the dynamic equation (3.7) as

$$
\dot{w}_{i,j} = \tilde{g}_{i,j}(\bar{x}) + \sum_{r=1}^{k} \alpha_{i,j,r} x_{r,r} + f_{i,j}(w) + x_{pi,j} := w_{i,j+1}.
$$

**Stage b:** In this stage, we construct the variable $x_{pi,j}$ via a simple pre-compensator $P_{i,j}$ with an integral operation as shown in Figure 3.1. That is, we construct $x_{pi,j}$ as

$$
x_{pi,j} := I_1 \left[ x_{pi,j+1} + \mathcal{L}_{i,j}(u) \right],
$$

where the operator $\mathcal{I}_1[h]$ denotes the integral of $h(t)$,

$$\mathcal{I}_1[h] = \int h(\tau)d\tau.$$
In equation (3.12), \( x_{pi,j+1} \) is the input to the simple pre-compensator, and the linear function \( L_{i,j}(u) \) is defined as
\[
L_{i,j}(u) = \sum_{r=1}^{k} \beta_{i,j,r} u_r,
\]
where the coefficients \( \beta_{i,j,r} \), \( 1 \leq r \leq k \) are yet unknown. The above two equations lead to
\[
\dot{x}_{pi,j} := x_{pi,j+1} + \sum_{r=1}^{k} \beta_{i,j,r} u_r.
\]

**Stage c:** In this stage, we determine the unknown coefficients \( \beta_{i,j,r} \) for \( 1 \leq r \leq k \). To do so, let us differentiate \( w_{i,j+1} \) as defined in (3.11) with respect to time to obtain,
\[
\dot{w}_{i,j+1} = \bar{g}_{i,j+1}(x) + \sum_{r=1}^{k} \alpha_{i,j,r} g_r(x) + u_r + f_{i,j+1}(w) + x_{pi,j+1},
\]
where the linear function \( \bar{g}_{i,j+1}(x) := \bar{g}_{i,j}(\dot{x}) \), \( f_{i,j+1}(w) := f_{i,j}(\dot{w}) \).

**Remark 6** Clearly, in view of (3.1), \( \dot{x} \) contains only \( x \) variables and no input variables, thus \( \bar{g}_{i,j}(\dot{x}) \) is of the form \( \bar{g}_{i,j+1}(x) \). Similarly, since \( i \leq j \leq k-1 \) and since \( f_{i,j}(w) \) is a linear function of only variables \( w_{p,q} \) for \( p < i \) and \( q \leq j \), we observe that \( f_{i,j+1}(w) := f_{i,j}(\dot{w}) \) is a linear function of only variables \( w_{p,q} \) for \( p < i \) and \( q \leq j+1 \).

We select
\[
\beta_{i,j,r} = -\alpha_{i,j,r}
\]
in order to cancel all the input variables in (3.15). This specifies completely \( x_{pi,j} \) given in (3.12). In other words, by the above selection of \( L_{i,j}(u) \), we complete the task of designing the simple pre-compensator \( \mathcal{P}_{i,j} \) shown in Figure 3.1.

All the above development, lets us rewrite \( \dot{w}_{i,j+1} \) as
\[
\dot{w}_{i,j+1} = \bar{g}_{i,j+1}(x) + \sum_{r=1}^{k} \alpha_{i,j,r} g_r(x) + f_{i,j+1}(w) + x_{pi,j+1}
\]
\[
= g_{i,j+1}(x) + f_{i,j+1}(w) + x_{pi,j+1},
\]
where
\[
g_{i,j+1}(x) = \bar{g}_{i,j+1}(x) + \sum_{r=1}^{k} \alpha_{i,j,r} g_r(x).
\]
This completes the task of designing the simple pre-compensator \( \mathcal{P}_{i,j} \).

At this time, it is wise to pause and reflect on the above constructional details. Equations (3.12) to (3.15) are crucial aspects of the algorithm. Expressing \( x_{pi,j} \) as an integral of some terms as in (3.12) enables the cancellation of unwanted input variables in \( \dot{w}_{i,j+1} \) as seen in (3.15) and (3.16). Although the mechanics of writing these equations are straightforward, the underlying concepts as to why such cancellations are done are profound. These concepts enable us to move the unwanted input variables from one step to another until they are accumulated in the last step where they are permissible.

We summarize now different notations and inputs and outputs of the \( \mathcal{P} \)-algorithm whose main task is to design the simple pre-compensator \( \mathcal{P}_{i,j} \) which
is essentially prescribed by prescribing the linear function $L_{i,j}(u)$ that cancels some unwanted variables. We encounter here the following terminology:

$g_{i,j}(x)$ Input data to the algorithm

$g_{i,j+1}(x)$ Output data of the algorithm

$f_{i,j}(w)$ Input data to the algorithm

$f_{i,j+1}(w)$ Output data of the algorithm

$L_{i,j}(u)$ Output data of the algorithm

$\beta_{i,j,r}$ Coefficients that define $L_{i,j}(u)$

$\alpha_{i,j,r}$ Internal coefficients used to rewrite a part of $g_{i,j}(x)$

$x_{pi,j}$ Output variable and state of simple pre-compensator $P_{i,j}$

$x_{pi,j+1}$ Input variable to simple pre-compensator $P_{i,j}$

$w_{i,j+1}$ Prescribed variable ahead of the algorithm

$\dot{w}_{i,j+1}$ A variable which is the output of the algorithm

**Remark 7** It is evident how we can make use of $P$-algorithm repetitively to design all the simple pre-compensators $P_{i,j}$, $i = 1$ to $k - 1$ and $j = i$ to $k - 1$. To be explicit, we narrate the design procedure here. For each $i$, $i = 1$ to $k - 1$, we start with the design of $P_{i,i}$. This requires $g_{i,i}(x)$ and $f_{i,i}(w)$ as inputs to the $P$-algorithm. We will show in the subsequent section how to generate these inputs. This involves transforming each $\Sigma_i$ to a uniform rank system $\breve{\Sigma}_{i,i}$ in SCB format having the same order of infinite zeros as $\Sigma_i$ does, namely $i$. For the sake of our present discussion we assume that both $g_{i,i}(x)$ and $f_{i,i}(w)$ are known and can be fed as input data to the $P$-algorithm. The algorithm not only designs $P_{i,i}$ (and thus $x_{pi,i} = u_i$), it also generates the linear functions $g_{i,i+1}(x)$ and $f_{i,i+1}(w)$. By feeding $g_{i,i+1}(x)$ and $f_{i,i+1}(w)$ as input data to the $P$-algorithm, we can design $P_{i,i+1}$ and at the same time generate the linear functions $g_{i,i+2}(x)$ and $f_{i,i+2}(w)$. In turn, $g_{i,i+2}(x)$ and $f_{i,i+2}(w)$ can be used as input data to the $P$-algorithm to design $P_{i,i+2}$ and to generate the linear functions $g_{i,i+3}(x)$ and $f_{i,i+3}(w)$. Thus, successively and recursively we can design all $P_{i,j}$, $j = i$ to $k - 1$ once we have $g_{i,i}(x)$ and $f_{i,i}(w)$.

**Remark 8** It is critical to understand the structure of the pre-compensator $P_{i,j}$ shown in Figure 3.1. Clearly, $x_{pi,j}$ represents the state of $P_{i,j}$. Also, as emphasized in Remark 7, the goal of $P_{i,j}$ is to generate as its outputs, $x_{pi,j}$ which is an input to $\dot{w}_{i,j+1}$, and the linear functions $g_{i,j+1}(x)$ and $f_{i,j+1}(w)$ which define the dynamics of $\dot{w}_{i,j+1}$. To accomplish its goal, $P_{i,j}$ requires two types of input data. One of them is $x_{pi,j+1}$ which indeed is a new input variable to the pre-compensator $P_{i,j}$. Besides this input, as displayed clearly in Figure 3.1, the linear function $L_{i,j}(u)$ is an input to $P_{i,j}$. Equation (3.13) shows that $L_{i,j}(u)$ is a linear function of the input variables of the given system $\Sigma$, namely
u_r, r = 1 to k. However, each u_r := \chi_{pr,r} for r = 1 to k− 1, is generated by the simple pre-compensator \mathcal{P}_{r,r}, r = 1 to k− 1. This leads us to emphasize two aspects of pre-compensator design, (1) all simple pre-compensators \mathcal{P}_{\ell,m}, \ell = 1 to k− 1, m = \ell to k− 1 are linked via linear functions of the type \mathcal{L}_{\ell,m}(u). When all the pre-compensators \mathcal{P}_{\ell,m} are successfully and iteratively designed, they can all be put together as a single over-all pre-compensator whose new inputs are \chi_{pr,k} which can be designated as u_{\text{new},i}, i = 1 to k− 1 and one of the original inputs to the given system \Sigma, namely u_k which can be called u_{\text{new},k}. The outputs of the over-all pre-compensator are the variables u_r := \chi_{pr,r} for r = 1 to k− 1 to the given system \Sigma. This is illustrated via an example in Section 3.6 by presenting the dynamic equations of each simple pre-compensator and then the over all pre-compensator. Moreover, the over-all pre-compensator, by generating the linear functions of the type g_{i,i}(x) and f_{i,i}(w), enables us to write the entire squared down uniform rank system in the format of SCB as given in equation (3.2) in Section 3.2.

Remark 9 Once all the subsystems \Sigma_i, i = 1 to k are transformed to uniform rank systems, they can all be put together as one subsystem \Sigma_r envisioned in Figure 1.3 by defining the output y_r whose components are outputs of individual subsystems \Sigma_i, i = 1 to k, and by defining the input u_{\text{new}} = u_r to consist of individual inputs u_{\text{new},i}, i = 1 to k. The design of each pre-compensator does not affect the finite zero structure. To relate the finite zero structures shown in Figures 1.2 and 1.3 we define A_{0,r}y_r = \Sigma_{i=1}^k A_{o,i}y_{r,i}.

3.5 Transformation of \Sigma_i to a uniform rank system \bar{\Sigma}_{i,i}

As we discussed in Section 3.3, squaring down each \Sigma_i, i = 1 to k− 1, to a uniform rank system \Sigma_{w,i} is done in two phases, phase 1 during which \Sigma_i is transformed to a uniform rank system \bar{\Sigma}_{i,i} in SCB format whose order of infinite zeros is the same as that of \Sigma_i, and phase 2 during which the pre-compensator \mathcal{P}_i is designed that squares down \bar{\Sigma}_{i,i} to a uniform rank system \Sigma_{w,i} having its order of infinite zeros equal to k. Clearly, phase 2 can be accomplished by the \mathcal{P}-algorithm as explained in Remark 7. What remains is the determination of the expressions for g_{i,i}(x) and f_{i,i}(w) that are required to get started for phase 2. This remaining task is what is pursued below by successfully completing phase 1.

In the processing of determining \bar{\Sigma}_{i,i}, we introduce variables w_{i,j} for 1 \leq i \leq k and 1 \leq j \leq i such that there exists a recursive relationship between these variables,
\[ \dot{w}_{i,j} = w_{i,j+1} \quad \text{for} \quad 1 < i \leq k \quad \text{and} \quad 1 \leq j < i, \]
and
\[ \dot{w}_{i,i} = g_{i,i}(x) + f_{i,i}(w) + u_i \quad \text{for} \quad 1 \leq i \leq k. \]
Also, \( \dot{w}_{i,j} \) for \( 1 < i \leq k \) and \( 1 \leq j < i \) can be expressed in terms of \( x \) and certain known variable \( w \). In fact, \( \dot{w}_{i,j} \) in this case is of the form,

\[
\dot{w}_{i,j} = x_{i,j+1} + f_{i,j}(w),
\]

(3.18)

where \( f_{i,j}(w) \) is a function of \( w \) with a certain structure, namely it is a linear function of all variables \( w_{p,q} \) for \( p < i, q \leq j \). The form of equation (3.18) is depicted in Figure 3.2. We observe that Figure 3.2 complements Figure 3.1 to show the structure of each \( \dot{w}_{i,j} \) as related to \( x \) and \( w \) for all possible indices \( i \) and \( j \).

![Figure 3.2: Structure of \( \dot{w}_{i,j} \) for \( 1 < i \leq k \) and \( 1 \leq j < i \)](image)

In what follows, expressions for all \( f_{i,j}(w) \) are obtained recursively leading to the determination of \( g_{i,i}(x) \) and \( f_{i,i}(w) \). We illustrate this at first by considering specific values of \( i \), namely \( i = 1 \) to \( 4 \), before considering a general index \( i \).

### 3.5.1 Determination of \( g_{1,1}(x) \) and \( f_{1,1}(w) \)

As alluded to, our goal here is to transform \( \Sigma_1 \) to \( \tilde{\Sigma}_{1,1} \), and in that process determine the expressions for \( g_{1,1}(x) \) and \( f_{1,1}(w) \).

Let us recall the dynamic equation of \( \Sigma_1 \),

\[
\Sigma_1: \{ \dot{x}_{1,1} = g_1(x) + u_1 \}.
\]

(3.19)

Clearly, this is already in the form of a uniform rank system with its order of infinite zeros equal to 1. Nevertheless, for uniformity of notations to follow, we let

\[ x_{1,1} = u_{1,1}, \]

and re-write equation (3.19) as

\[ \tilde{\Sigma}_{1,1}: \{ \dot{w}_{1,1} = g_1(x) + u_1 \} \]
Comparing the above equation with the equation (3.7) for \( i = 1 \) and \( j = 1 \), we find that
\[
\dot{g}_{1,1}(x) = g_1(x) \quad \text{and} \quad f_{1,1}(w) = 0.
\]
This completes the task of this subsection.

3.5.2 Determination of \( g_{2,2}(x) \) and \( f_{2,2}(w) \)
To start with, we assume that the pre-compensator \( P_1 \) is designed that squares down \( \Sigma_{1,1} \) to a uniform rank system \( \Sigma_{w1} \) by utilizing appropriately the \( P \)-algorithm, and its notations are consistent with those in Figures 3.1 and 3.2.

We recall next the dynamic equations of \( \Sigma_2 \) as
\[
\Sigma_2 : \begin{cases}
\dot{x}_{2,1} &= x_{2,2} + \nu_{2,1,1} w_{1,1} \\
\dot{x}_{2,2} &= g_2(x) + u_2,
\end{cases}
\]
where \( y_1 = x_{1,1} = w_{1,1} \) by the notation introduced in the previous subsection. We denote
\[
w_{2,1} = x_{2,1} \quad \text{and} \quad \dot{w}_{2,1} = w_{2,2} = x_{2,2} + \nu_{2,1,1} w_{1,1}.
\]
In view of the notations introduced in equation (3.18) and in Figure 3.2, we note that
\[
f_{2,1}(w) = \nu_{2,1,1} w_{1,1}.
\]
We have
\[
\dot{w}_{2,2} = g_2(x) + u_2 + \nu_{2,1,1} \dot{w}_{1,1} = g_2(x) + \nu_{2,1,1} w_{1,2} + u_2.
\]
Comparing the expression for \( \dot{w}_{2,2} \) with the equation (3.7) for \( i = 2 \) and \( j = 2 \), we find that
\[
g_{2,2}(x) = g_2(x) \quad \text{and} \quad f_{2,2}(w) = \nu_{2,1,1} w_{1,2}.
\]
This lets us rewrite the above system as
\[
\Sigma_{2,2} : \begin{cases}
\dot{w}_{2,1} &= w_{2,2} := x_{2,2} + f_{2,1}(w) \\
\dot{w}_{2,2} &= g_2(x) + f_{2,2}(w) + u_2.
\end{cases}
\]
This completes the task of this subsection.

3.5.3 Determination of \( g_{3,3}(x) \) and \( f_{3,3}(w) \)
To start with, we assume that the pre-compensators \( P_1 \) and \( P_2 \) are designed that square down \( \Sigma_{1,1} \) and \( \Sigma_{2,2} \) to uniform rank systems \( \Sigma_{w1} \) and \( \Sigma_{w2} \) by utilizing appropriately the \( P \)-algorithm, and the notations are consistent with those in Figures 3.1 and 3.2.

We recall below the dynamic equations of \( \Sigma_3 \) as
\[
\Sigma_3 : \begin{cases}
\dot{x}_{3,1} &= x_{3,2} + \nu_{3,1,1} w_{1,1} + \nu_{3,1,2} w_{2,1} \\
\dot{x}_{3,2} &= x_{3,3} + \nu_{3,2,1} w_{1,1} + \nu_{3,2,2} w_{2,1} \\
\dot{x}_{3,3} &= g_3(x) + u_3.
\end{cases}
\]
where \( y_1 = x_{1,1} = w_{1,1} \) and \( y_2 = x_{2,1} = w_{2,1} \) by the notations introduced in the previous two subsections.

We denote
\[
\dot{w}_{3,2} = x_{3,2} + \nu_{3,1,1} w_{1,1} + \nu_{3,1,2} w_{2,1}.
\]
This yields
\[
\dot{w}_{3,1} = w_{3,2} = x_{3,2} + f_{3,1}(w),
\]
where
\[
f_{3,1}(w) = \nu_{3,1,1} w_{1,1} + \nu_{3,1,2} w_{2,1},
\]
consistent with the notations introduced in Figure 3.2. We have
\[
\dot{w}_{3,2} = x_{3,3} + f_{3,2}(w),
\]
where
\[
f_{3,2}(w) = \nu_{3,2,1} w_{1,1} + \nu_{3,2,2} w_{2,1} + \nu_{3,1,1} w_{1,2} + \nu_{3,1,2} w_{2,2},
\]
consistent with the notations introduced in Figure 3.2. We have
\[
\dot{w}_{3,2} = x_{3,3} + f_{3,2}(w),
\]
where
\[
f_{3,2}(w) = \nu_{3,2,1} w_{1,1} + \nu_{3,2,2} w_{2,1} + \nu_{3,1,1} w_{1,2} + \nu_{3,1,2} w_{2,2},
\]
consistent with the notations introduced in Figure 3.2. We have
\[
\dot{w}_{3,2} = x_{3,3} + f_{3,2}(w),
\]
and obtain
\[
\dot{w}_{3,3} = g_3(x) + u_3 + f_{3,2}(\dot{w}) = g_{3,3}(x) + f_{3,3}(w) + u_3, \quad (3.20)
\]
where \( g_{3,3}(x) = g_3(x) \) and
\[
f_{3,3}(w) = f_{3,2}(\dot{w}) = \nu_{3,2,1} w_{1,1} + \nu_{3,2,2} w_{2,1} + \nu_{3,1,1} w_{1,2} + \nu_{3,1,2} w_{2,2}.
\]
We observe that all \( f_{3,j}(w) \) has a definite structure, namely \( f_{3,j}(w) \) is a function of only \( w_{p,q} \) for \( p < 3 \) and \( q \leq j \). We can see easily that
\[
\dot{\Sigma}_{3,3} : \begin{cases}
\dot{w}_{3,1} = w_{3,2} \\
\dot{w}_{3,2} = w_{3,3} \\
\dot{w}_{3,3} = g_{3,3}(x) + f_{3,3}(w) + u_3.
\end{cases}
\]
This completes the task of this subsection.

### 3.5.4 Determination of \( g_{4,4}(x) \) and \( f_{4,4}(w) \)

To start with, once again we assume that the pre-compensators \( P_1 \) to \( P_3 \) are designed that square down \( \Sigma_{1,1} \) to \( \Sigma_{4,3} \) to uniform rank systems \( \Sigma_{w1} \) to \( \Sigma_{w3} \) by utilizing appropriately the \( P \)-algorithm, and the notations are consistent with those in Figures 3.1 and 3.2.

We recall next the dynamic equations of \( \Sigma_4 \) as given,
\[
\Sigma_4 : \begin{cases}
\dot{x}_{4,1} = x_{4,2} + \sum_{j=1}^{3} \nu_{4,1,j} x_{j,1} \\
\dot{x}_{4,2} = x_{4,3} + \sum_{j=1}^{3} \nu_{4,2,j} x_{j,1} \\
\dot{x}_{4,3} = x_{4,4} + \sum_{j=1}^{3} \nu_{4,3,j} x_{j,1} \\
\dot{x}_{4,4} = g_4(x) + u_4,
\end{cases}
\]

21
where $y_i = x_{i,1} = w_{i,1}, i = 1$ to $3$ by the notations introduced in the previous two subsections.

We denote

$$w_{4,2} = \dot{x}_{4,1} = x_{4,2} + \Sigma_{j=1}^{3} \nu_{4,1,j} w_{j,1}. $$

This yields

$$w_{4,1} = w_{4,2} = x_{4,2} + \Sigma_{j=1}^{3} \nu_{4,1,j} w_{j,1} = x_{4,2} + f_{4,1}(w),$$

where

$$f_{4,1}(w) = \Sigma_{j=1}^{3} \nu_{4,1,j} w_{j,1}. $$

We have

$$\dot{w}_{4,2} = x_{4,3} + f_{4,2}(w),$$

where

$$f_{4,2}(w) = \Sigma_{j=1}^{3} \nu_{4,2,j} w_{j,1} + f_{4,1}(\dot{w}) = \Sigma_{m=1}^{2} \Sigma_{j=1}^{3} \nu_{4,3-m,j} w_{j,m}. $$

We denote

$$w_{4,3} = x_{4,3} + f_{4,2}(w),$$

and obtain

$$\dot{w}_{4,3} = x_{4,4} + f_{4,3}(w),$$

where

$$f_{4,3}(w) = \Sigma_{m=1}^{3} \Sigma_{j=1}^{3} \nu_{4,4-m,j} w_{j,m}. $$

By denoting

$$w_{4,4} = x_{4,4} + f_{4,3}(w),$$

we obtain

$$\dot{w}_{4,4} = g_{4,4}(x) + f_{4,4}(w) + u_{4},$$

where $g_{4,4}(x) = g_{4}(x)$ and

$$f_{4,4}(w) = f_{4,3}(\dot{w}) = \Sigma_{m=1}^{3} \Sigma_{j=1}^{3} \nu_{4,4-m,j} w_{j,m+1}. $$

As in the previous subsection, we observe that all $f_{4,j}(w), j = 1$ to $3$, are consistent with (3.18) and the notations of Figure 3.2. Also, all $f_{4,j}(w), j = 1$ to $4$, have a definite structure, namely $f_{4,j}(w)$ is a linear function of only $w_{p,q}$ for $p < 4$ and $q \leq j$. Moreover, we can see easily that

$$\dot{\Sigma}_{4,4} : \begin{cases} 
\dot{w}_{4,1} &= w_{4,2} \\
\dot{w}_{4,2} &= w_{4,3} \\
\dot{w}_{4,3} &= w_{4,4} \\
\dot{w}_{4,4} &= g_{4,4}(x) + f_{4,4}(w) + u_{4}.
\end{cases}$$

This completes the task of this subsection.
3.5.5 Determination of \( g_{i,j}(x) \) and \( f_{i,j}(w) \)

The previous subsections lead the way to generalize the recursive procedure of determining \( g_{i,j}(x) \) and \( f_{i,j}(w) \) for any \( i = 2 \) to \( k - 1 \), and determining \( \Sigma_{i,j} \) for any \( i = 2 \) to \( k \). This is pursued in this subsection. A good grasp of notations of Figures 3.1 and 3.2 facilitates an easy understanding of what follows.

We first recall below the dynamic equations of \( \Sigma_{i} \) as given,

\[
\Sigma_{i} : \begin{cases}
\dot{x}_{i,1} = x_{i,2} + \sum_{j=1}^{i-1} \nu_{i,1,j} x_{j,1} \\
\dot{x}_{i,2} = x_{i,3} + \sum_{j=1}^{i-1} \nu_{i,2,j} x_{j,1} \\
\quad \vdots \\
\dot{x}_{i,i-1} = x_{i,i} + \sum_{j=1}^{i-1} \nu_{i,i-1,j} x_{j,1} \\
\dot{x}_{i,i} = g_{i}(x) + u_{i}.
\end{cases}
\]

We denote

\[
w_{i,1} = x_{i,1} \quad \text{and} \quad w_{i,2} = x_{i,2} + \sum_{j=1}^{i-1} \nu_{i,1,j} w_{j,1}.
\]

We have

\[
\dot{w}_{i,1} = w_{i,2} := x_{i,2} + f_{i,1}(w),
\]

where \( f_{i,1}(w) = \sum_{j=1}^{i-1} \nu_{i,1,j} w_{j,1} \).

If \( i = 2 \), we have

\[
\dot{w}_{2,1} = g_{2}(x) + u_{2} + f_{2,1}(\dot{w}) = g_{2,2}(x) + f_{2,2}(w) + u_{2},
\]

where \( g_{2,2}(x) = g_{2}(x) \) and

\[
f_{2,2}(w) = f_{2,1}(\dot{w}) = \nu_{2,1,1} w_{1,2}.
\]

If \( i \neq 2 \), but \( 3 \leq i \leq k \), we have

\[
\dot{w}_{i,2} = x_{i,3} + \sum_{j=1}^{i-1} \nu_{i,2,j} w_{j,1} + f_{i,1}(\dot{w}) := x_{i,3} + f_{i,2}(w),
\]

where

\[
f_{i,2}(w) = \sum_{j=1}^{i-1} \nu_{i,2,j} w_{j,1} + \sum_{j=1}^{i-1} \nu_{i,1,j} w_{j,2} = \sum_{m=1}^{i-1} \nu_{i,3-m,j} w_{j,m}.
\]

We denote \( w_{i,3} = \dot{w}_{i,2} \). If \( i = 3 \), we have

\[
\dot{w}_{3,3} = g_{3}(x) + u_{3} + \sum_{m=1}^{2} \nu_{i,3,m} \dot{w}_{j,m} = g_{3,3}(x) + f_{3,3}(w) + u_{3},
\]

where \( g_{3,3}(x) = g_{3}(x) \) and

\[
f_{3,3}(w) = f_{3,2}(\dot{w}) = \sum_{m=1}^{2} \nu_{3,3-m,j} \dot{w}_{j,m+1}.
\]

If \( i \neq 3 \), but \( 4 \leq i \leq k \), we have

\[
\dot{w}_{i,3} = x_{i,4} + \sum_{j=1}^{i-1} \nu_{i,3,j} w_{j,1} + f_{i,2}(\dot{w}) := x_{i,4} + f_{i,3}(w),
\]
where
\[ f_{i,3}(w) = \sum_{m=1}^{3} \sum_{j=1}^{i-1} \nu_{i,4-m,j} w_{j,m}. \]

We denote \( w_{i,4} = \dot{w}_{i,3}. \) If \( i = 4, \) we have
\[ \dot{w}_{4,4} = g_4(x) + u_4 + f_{4,3}(\dot{w}) = g_{4,4}(x) + f_{4,4}(w) + u_4, \]
where \( g_{4,4}(x) = g_4(x) \) and
\[ f_{4,4}(w) = f_{4,3}(\dot{w}) = \sum_{m=1}^{3} \sum_{j=1}^{4-m} \nu_{4,4-m,j} w_{j,m+1}. \]

Remark 10 For the purpose of phase 2, as needed by \( P \)-algorithm, it is important to recognize the structure of \( f_{i,\ell}(w) \); it is a function of only \( w_{p,q} \) for \( p < i \) and \( q \leq \ell. \)

If \( i = \ell + 1, \) we observe that
\[ \dot{w}_{i,i} = g_{i,i}(x) + f_{i,i}(w) + u_i, \]
where \( g_{i,i}(x) = g_i(x) \) and
\[ f_{i,i}(w) = f_{i,i-1}(\dot{w}) = \sum_{m=1}^{i-1} \sum_{j=1}^{i-1} \nu_{i,i-m,j} w_{j,m+1}. \]

We emphasize that the structure of \( f_{i,i}(w) \) conforms to what is discussed in Remark 10.

Finally, the above development yields for all \( 2 \leq i \leq k, \)
\[ \dot{\Sigma}_{i,i} : \begin{cases} \dot{w}_{i,1} = w_{i,2} \\ \dot{w}_{i,2} = w_{i,3} \\ \vdots \\ \dot{w}_{i,i} = g_{i,i}(x) + f_{i,i}(w) + u_i. \end{cases} \]

This completes the task of this subsection.
3.5.6 Conversion of $x_{i,j}$, $i = 1$ to $k$ and $j = 1$ to $i$, to corresponding $w_{i,j}$

In phase 1 of each stage $i$ of transforming $\Sigma_i$ to $\tilde{\Sigma}_{i,i}$, $i = 1$ to $k$, we define $w_{i,j}$, $i = 1$ to $k$ and $j = 1$ to $i$, in terms of $x_{i,j}$, $i = 1$ to $k$ and $j = 1$ to $i$. After all these stages are complete, we can solve for all $x_{i,j}$, $i = 1$ to $k$ in terms of $w_{i,j}$. From the notations summarized in Figure 4.2 and from equations (3.21) and (3.22), we can obtain easily such relationships. In fact, we have

\[
\begin{align*}
x_{i,1} &= w_{i,1} \quad \text{for all } i = 1 \text{ to } k, \\
x_{i,j} &= w_{i,j} - f_{i,j-1}(w) \\
&= w_{i,j} - \sum_{m=1}^{j-1} \nu_{i,j-m} w_{m,i} \quad \text{for all } 1 < i \leq k \text{ and } 1 < j \leq 3.
\end{align*}
\]

This enables all linear functions $g_{i,j}(x)$ to be re-written as $g_{i,j}(w)$.

**Remark 11** We observe that all our transformations in all stages never touch $\Sigma_0$, and hence the finite zero structure is preserved by the design of a pre-compensator.

3.6 Numerical Examples

**Example 1:** We consider first an example with the highest order of infinite zeros $k$ equal to 3. For ease of illustration, we set all the coefficients of the type $\nu_{i,j,m}$ equal to one. Also, we set the dimensions $n_i = 1$ for $i = 1$ to 3. We assume that

\[
g_{i,i}(x) = g_i(x) = x_0 + \sum_{i=1}^{3} \sum_{m=1}^{i} x_{\ell,m} \quad \text{for all } i = 1 \text{ to } 3.
\]

For the finite zero subsystem, $A_{0,i} = 1$, $i = 1, 2, 3$ and $A_i = -1$.

**Transformation of $\Sigma_i$ to $\tilde{\Sigma}_{i,i}$, $i = 1$ to 3:** Following the methodology of phase 1 as developed in Section 3.3, we get

\[
\begin{align*}
\tilde{\Sigma}_{1,1} : & \quad \begin{cases}
\dot{w}_{1,1} = g_1(x) + f_{1,1}(w) + u_1 \\
\dot{w}_{2,1} = \dot{w}_{2,2} \\
\dot{w}_{2,2} = g_2(x) + f_{2,2}(w) + u_2
\end{cases} \\
\tilde{\Sigma}_{2,2} : & \quad \begin{cases}
\dot{w}_{3,1} = \dot{w}_{3,2} \\
\dot{w}_{3,2} = \dot{w}_{3,3} \\
\dot{w}_{3,3} = g_3(x) + f_{3,3}(w) + u_3
\end{cases}
\end{align*}
\]

where

\[
f_{1,1}(w) = 0, \quad f_{2,2}(w) = w_{1,2}, \quad \text{and} \quad f_{3,3}(w) = w_{1,2} + w_{2,2} + w_{1,3} + w_{2,3}.
\]

We proceed now to design $P_{1,1}$ for the subsystem $\tilde{\Sigma}_{1,1}$ by using $P$-algorithm. By feeding the algorithm with $g_{1,1}(x) = g_1(x)$ and $f_{1,1}(w) = 0$, we get $f_{1,2}(w) = 0$,

\[
g_{1,2}(x) = 2x_0 + 7x_{1,1} + 6x_{2,1} + 4x_{2,2} + 4x_{3,1} + 4x_{3,2} + 4x_{3,3}.
\]
and
\[ \dot{x}_{p1,1} = -x_{p1,1} + x_{p1,2} - u_2 - u_3, \]
where \( x_{p1,1} := u_1 \). This yields the last equation of \( \ddot{\Sigma}_{1,2} \) as
\[ \ddot{w}_{1,2} = g_{1,2}(x) + x_{p1,2}. \]

We proceed now to design \( P_{1,2} \) for the subsystem \( \ddot{\Sigma}_{1,2} \). By feeding the \( P \)-algorithm with \( g_{1,2}(x) \) and \( f_{1,2}(w) = 0 \), we get \( f_{1,3}(w) = 0 \),
\[ g_{1,3}(x) = 13x_0 + 31x_{1,1} + 25x_{2,1} + 21x_{2,2} + 17x_{3,1} + 19x_{3,2} + 19x_{3,3}, \]
and
\[ \dot{x}_{p1,2} := -7x_{p1,1} + u_{new,1} - 4u_2 - 4u_3, \]
where \( u_{new,1} := x_{p1,1} \). This yields the last equation of \( \ddot{\Sigma}_{1,3} \) as
\[ \ddot{w}_{1,3} = g_{1,3}(x) + u_{new,1}. \]

This completes the design for the subsystem \( \Sigma_1 \).

We can proceed now to design \( P_{2,2} \) for the subsystem \( \ddot{\Sigma}_{2,2} \). By feeding the \( P \)-algorithm with \( g_{2,2}(x) = g_2(x) \) and \( f_{2,2}(w) = w_{1,2} \), we get \( f_{2,3}(w) = w_{1,3} \),
\[ g_{2,3}(x) = 2x_0 + 7x_{1,1} + 6x_{2,1} + 4x_{2,2} + 4x_{3,1} + 4x_{3,2} + 4x_{3,3}, \]
and
\[ \dot{x}_{p2,2} = -x_{p1,1} - x_{p2,2} + u_{new,2} - u_3, \]
where \( x_{p2,2} := u_2 \) and \( u_{new,2} := x_{p2,3} \). This yields the last equation of \( \ddot{\Sigma}_{2,3} \) as
\[ \ddot{w}_{2,3} = g_{2,3}(x) + f_{2,3}(w) + u_{new,2}. \]

This completes the design for the subsystem \( \Sigma_2 \).

By now the design of all simple pre-compensators is complete. We can convert all \( x_{i,j}, i = 1 \) to 3 and \( j = 1 \) to \( i \), to corresponding \( w_{i,j} \). In view of (3.24), we have
\[ x_{i,1} = w_{i,1} \text{ for all } i = 1 \text{ to } 3, \]
\[ x_{i,j} = w_{i,j} - \sum_{m=1}^{i-1} w_{p,m} \text{ for all } 1 < i \leq 3 \text{ and } 1 < j \leq i. \]

This in turn lets us express,
\[ g_{w1}(w) = 13w_0 - 28w_{1,1} - 19w_{1,2} - 13w_{2,1} + 2w_{2,2} + 17w_{3,1} + 19w_{3,2} + 19w_{3,3}. \]
\[ g_{w2}(w) = 2w_0 - 5w_{1,1} - 4w_{1,2} + w_{1,3} - 2w_{2,1} + 4w_{3,1} + 4w_{3,2} + 4w_{3,3}. \]
\[ g_{w3}(w) = w_0 - 2w_{1,1} + w_{1,3} - w_{2,1} + w_{2,2} + w_{3,1} + w_{3,2} + w_{3,3}. \]

The above expressions lead to the squared down uniform rank system in SCB format for \( i = 1 \) to 3 as
\[ \ddot{\Sigma}_{i,3} : \begin{cases} \ddot{w}_{i1} &= w_{i,2} \\ \ddot{w}_{i2} &= w_{i,3} \\ \ddot{w}_{i3} &= g_{wi}(w) + u_{new,i} \end{cases} \]
where $u_{\text{new},i}$, $i = 1$ and 2 are as defined earlier, and $u_{\text{new},3} := u_3$.

We can now put together the dynamic equations of all simple pre-compensators of the type $\mathcal{P}_{i,j}$ to obtain the dynamic equations of over all pre-compensator $\Sigma_{\text{pre}}$,

$$
\Sigma_{\text{pre}} : \begin{cases}
\dot{x}_{p1,1} = \begin{bmatrix} -1 & 1 & -1 \\ -7 & 0 & -4 \\ -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_{p1,1} \\ x_{p1,2} \\ x_{p2,2} \end{bmatrix} + \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & -4 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} u_{\text{new},1} \\ u_{\text{new},2} \\ u_{\text{new},3} \end{bmatrix}.
\end{cases}
$$

We observe that the output of $\Sigma_{\text{pre}}$ is the input to the given system $\Sigma$,

$$
u = \text{col}(x_{p1,1}, x_{p1,2}, u_{\text{new},3}) = \text{col}(u_1, u_2, u_3),$$

while the input of $\Sigma_{\text{pre}}$ is

$$
u_{\text{new}} = \text{col}(u_{\text{new},1}, u_{\text{new},2}, u_{\text{new},3}),$$

where $u_{\text{new},3} = u_3$.

**Example 2:** We consider next an example with the highest order of infinite zeros $k$ equal to 4. Again, for ease of illustration, we set all the coefficients of the type $\nu_{i,j,m}$ equal to one. Also, we set the dimensions $n_i = 1$ for $i = 1$ to 4.

We assume that

$$g_{i,i}(x) = g_i(x) = x_0 + \sum_{\ell=1}^{3} \sum_{m=1}^{4} x_{\ell,m}$$

for all $i = 1$ to 4.

For the finite zero subsystem, $A_{0,i} = 1$, $i = 1$ to 4, and $A_0 = -1$.

**Transformation of $\Sigma_i$ to $\tilde{\Sigma}_{i,i}$, $i = 1$ to 4:** Following the methodology of phase 1 as developed in Section 3.3, we get

$$
\tilde{\Sigma}_{1,1} : \begin{cases}
\dot{w}_{1,1} = g_1(x) + f_{1,1}(w) + u_1 \\
\dot{w}_{2,1} = \dot{w}_{2,2} \\
\dot{w}_{2,2} = g_2(x) + f_{2,2}(w) + u_2 \\
\dot{w}_{3,1} = \dot{w}_{3,2} \\
\dot{w}_{3,2} = \dot{w}_{3,3} \\
\dot{w}_{3,3} = g_3(x) + f_{3,3}(w) + u_3 \\
\dot{w}_{4,1} = \dot{w}_{4,2} \\
\dot{w}_{4,2} = \dot{w}_{4,3} \\
\dot{w}_{4,3} = \dot{w}_{4,4} \\
\dot{w}_{4,4} = g_4(x) + f_{4,4}(w) + u_4,
\end{cases}
$$

where $f_{1,1}(w) = 0, f_{2,2}(w) = w_{1,2},$ $f_{3,3}(w) = w_{1,2} + w_{2,2} + w_{1,3} + w_{2,3}$ and $f_{4,4}(w) = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{j,i+1}$.  

27
We proceed now to design $P_{1,1}$ for the subsystem $\Sigma_{1,1}$ by using $P$-algorithm.

By feeding the algorithm with $g_{1,1}(x) = g_1(x)$ and $f_{1,1}(w) = 0$, we get $f_{1,2}(w) = 0,$

\[ g_{1,2}(x) = 3x_0 + 11x_1 + 10x_2 + 5x_3 + 8x_4 + 5x_5 + 5x_6 + 5x_7 + 5x_8, \]

and

\[ \dot{x}_{p,1} = -x_{p,1} + x_{p,2} - u_2 - u_3 - u_4, \]

where $x_{p,1} := u_1$. This yields the last equation of $\tilde{\Sigma}_{1,2}$ as

\[ \dot{w}_{1,2} = g_{1,2}(x) + x_{p,1}. \]

We proceed now to design $P_{1,2}$ for the subsystem $\Sigma_{1,2}$. By feeding the $P$-algorithm with $g_{1,2}(x)$ and $f_{1,2}(w) = 0$, we get $f_{1,3}(w) = 0,$

\[ g_{1,3}(x) = 23x_0 + 67x_1 + 57x_2 + 36x_3 + 44x_4 + 34x_5 + 31x_6 + 29x_7 + 31x_8 + 31x_9, \]

and

\[ \dot{x}_{p,1} = -11x_{p,1} + x_{p,3} - 5u_2 - 5u_3 - 5u_4. \]

This yields the last equation of $\tilde{\Sigma}_{1,3}$ as

\[ \dot{w}_{1,3} = g_{1,3}(x) + x_{p,1}. \]

We can design next $P_{1,3}$ for the subsystem $\Sigma_{1,3}$. By feeding the $P$-algorithm with $g_{1,3}(x)$ and $f_{1,3}(w) = 0$, we get $f_{1,4}(w) = 0,$

\[ g_{1,4}(x) = 142x_0 + 414x_1 + 57x_2 + 222x_3 + 279x_4 + 209x_5 + 199x_6 \\
+ 188x_7 + 194x_8 + 196x_9 + 196x_{10}, \]

and

\[ \dot{x}_{p,1} = -67x_{p,1} + u_{\text{new},1} - 36u_2 - 31u_3 - 31u_4, \]

where $u_{\text{new},1} := x_{p,4}$. This yields the last equation of $\tilde{\Sigma}_{1,4}$ as

\[ \dot{w}_{1,4} = g_{1,4}(x) + u_{\text{new},1}. \]

This completes the design for the subsystem $\Sigma_1$.

We can proceed now to design $P_{2,2}$ for the subsystem $\Sigma_{2,2}$. By feeding the $P$-algorithm with $g_{2,2}(x) = g_2(x)$ and $f_{2,2}(w) = w_{1,2}$, we get $f_{2,3}(w) = w_{1,3},$

\[ g_{2,3}(x) = 3x_0 + 11x_1 + 10x_2 + 5x_3 + 8x_4 + 5x_5 + 5x_6 + 5x_7 + 5x_8 + 5x_9, \]

and

\[ \dot{x}_{p,2} = -x_{p,1} - x_{p,2} + x_{p,3} - u_3 - u_4, \]

where $x_{p,2} := u_2$. This yields the last equation of $\tilde{\Sigma}_{2,3}$ as

\[ \dot{w}_{2,3} = g_{2,3}(x) + f_{2,3}(w) + x_{p,2}. \]
We design next \( P_{2,3} \) for the subsystem \( \tilde{\Sigma}_{2,3} \). By feeding the \( P \)-algorithm with \( g_{2,3}(x) \) and \( f_{2,3}(w) = w_{1,3} \), we get \( f_{2,4}(w) = w_{1,4} \).

\[
g_{2,4}(x) = 23x_0 + 67x_{1,1} + 57x_{2,1} + 36x_{2,2} + 44x_{3,1} + 34x_{3,2} + 31x_{3,3} + 29x_{4,1} + 31x_{4,2} + 31x_{4,3} + 31x_{4,4},
\]
and
\[
\dot{x}_{p2,3} = -11x_{p1,1} - 5x_{p2,2} + u_{new,2} - 5u_3 - 5u_4,
\]
where \( u_{new,2} := x_{p2,4} \). This yields the last equation of \( \tilde{\Sigma}_{2,4} \) as
\[
\dot{w}_{2,4} = g_{2,4}(x) + f_{2,4}(w) + u_{new,2}.
\]
This completes the design for the subsystem \( \tilde{\Sigma}_{2} \).

We can proceed now to design \( P_{3,3} \) for the subsystem \( \tilde{\Sigma}_{3,3} \). By feeding the \( P \)-algorithm with \( g_{3,3}(x) = g_{3}(x) \) and
\[
f_{3,4}(w) = w_{1,3} + w_{2,3} + w_{1,4} + w_{2,4},
\]
we get
\[
g_{3,4}(x) = 3x_0 + 11x_{1,1} + 10x_{2,1} + 5x_{2,2} + 8x_{3,1} + 5x_{3,2} + 5x_{3,3} + 5x_{4,1} + 5x_{4,2} + 5x_{4,3} + 5x_{4,4},
\]
and
\[
\dot{x}_{p3,3} = -x_{p1,1} - x_{p2,2} - x_{p3,3} + u_{new,3} - u_4,
\]
where \( x_{p3,3} := x_{3} \) and \( u_{new,3} := x_{p3,4} \). This yields the last equation of \( \tilde{\Sigma}_{3,4} \) as
\[
\dot{w}_{3,4} = g_{3,4}(x) + f_{3,4}(w) + u_{new,3}.
\]
This completes the design for the subsystem \( \tilde{\Sigma}_{3} \).

By now the design of all simple pre-compensators is complete. We can convert all \( x_{i,j} \), \( i = 1 \) to 4 and \( j = 1 \) to \( i \), to corresponding \( w_{i,j} \). In view of (3.24), we have
\[
x_{i,1} = w_{i,1} \text{ for all } i = 1 \text{ to } 4,
\]
\[
x_{i,j} = w_{i,j} - \sum_{m=1}^{j-1} \sum_{p=1}^{i-1} w_{p,m} \text{ for all } 1 < i \leq 4 \text{ and } 1 < j \leq i.
\]

This in turn lets us express,
\[
g_{w1}(w) = g_{1,4}(x) = 142w_0 + [-802 - 591 - 196 0]w_1 + [-637 - 369 - 196 0]w_2
+ [-307 - 183 3 0]w_3 + [188 194 196 196]w_4
\]
\[
g_{w2}(w) = g_{2,4}(x) + f_{2,4}(w) = 23w_0 + [-127 - 93 - 31 1]w_1 + [-101 - 57 - 31 0]w_2
+ [-49 - 28 0 0]w_3 + [29 31 31 31]w_4
\]
\[
g_{w3}(w) = g_{3,4}(x) + f_{3,4}(w) = 3w_0 + [-19 - 15 - 4 1]w_1 + [-15 - 10 - 4 1]w_2
+ [-7 - 5 0 0]w_3 + [5 5 5 5]w_4
\]
\[
g_{w4}(w) = g_{4,4}(x) + f_{4,4}(w) = w_0 + [-5 - 2 0 1]w_1 + [-4 - 1 0 1]w_2
+ [-2 0 1 1]w_3 + [1 1 1 1]w_4,
\]

29
where \( w_i = \text{col}(w_{i,1}, w_{i,2}, w_{i,3}, w_{i,4}) \). The above expressions lead to the squared down uniform rank system in SCB format for \( i = 1 \) to \( 4 \) as

\[
\dot{\Sigma}_{i,4} : \begin{cases}
\dot{w}_{i,1} = w_{i,2} \\
\dot{w}_{i,2} = w_{i,3} \\
\dot{w}_{i,3} = w_{i,4} \\
\dot{w}_{i,4} = g_{wi}(w) + u_{\text{new},i},
\end{cases}
\]

where \( u_{\text{new},i}, i = 1 \) to \( 3 \) are as defined earlier, and \( u_{\text{new},4} := u_4 \).

We can now put together the dynamic equations of all simple pre-compensators of the type \( P_{i,j} \) to obtain the dynamic equations of over all pre-compensator \( \Sigma_{\text{pre}} \):

\[
\Sigma_{\text{pre}} : \begin{bmatrix}
\dot{x}_{p_{i,1}} \\
\dot{x}_{p_{i,2}} \\
\dot{x}_{p_{i,3}} \\
\dot{x}_{p_{i,4}}
\end{bmatrix} =
\begin{bmatrix}
-1 & 1 & 0 & -1 & 0 & -1 \\
-11 & 0 & 0 & -5 & 0 & -5 \\
-67 & 0 & 0 & -36 & 0 & -31 \\
-1 & 0 & 0 & -1 & 1 & -1 \\
-11 & 0 & 0 & -5 & 0 & -5 \\
-1 & 0 & 0 & -1 & 0 & -1
\end{bmatrix}
\begin{bmatrix}
x_{p_{i,1}} \\
x_{p_{i,2}} \\
x_{p_{i,3}} \\
x_{p_{i,4}}
\end{bmatrix}
+ \begin{bmatrix}
0 & 0 & 0 & -1 \\
0 & 0 & 0 & -5 \\
1 & 0 & 0 & -31 \\
0 & 0 & 0 & -1 \\
0 & 1 & 0 & -5 \\
0 & 0 & 1 & -1
\end{bmatrix}
\begin{bmatrix}
u_{\text{new},1} \\
u_{\text{new},2} \\
u_{\text{new},3} \\
u_{\text{new},4}
\end{bmatrix}.
\]

We observe that the output of \( \Sigma_{\text{pre}} \) is the input to the given system \( \Sigma \),

\[
u = \text{col}(x_{p_{1,1}}, x_{p_{2,2}}, x_{p_{3,3}}, u_{\text{new},4}) = \text{col}(u_1, u_2, u_3, u_4),
\]

while the input of \( \Sigma_{\text{pre}} \) is

\[
u_{\text{new}} = \text{col}(u_{\text{new},1}, u_{\text{new},2}, u_{\text{new},3}, u_{\text{new},4}),
\]

where \( u_{\text{new},3} = u_4 \).
Chapter 4

Post-compensator design

Chapter 3 develops the design of a pre-compensator to square down a general linear invertible multi-input multi-out system to a uniform rank system. The goal of this chapter is to design a post-compensator that accomplishes the same task.

4.1 A general invertible system in SCB format — re-written

Section 3.1 describes a general linear time invariant invertible system \( \Sigma \) with the highest order of infinite zeros equal to \( k \) in the format of a SCB. The notations for state variables given there are suitable for pre-compensator design. We rewrite here the equations of SCB presented there by changing the notations conveniently for post-compensator design. Once again, in SCB format, a given system \( \Sigma \) can be decomposed into \( k + 1 \) subsystems, \( \Sigma_0 \) and \( \Sigma_m \), \( m = 1 \) to \( k \). As before, \( \Sigma_0 \) represents the dynamics of invariant zeros, while \( \Sigma_m \), \( m = 1 \) to \( k \) represents the dynamics of \( m \)-th order infinite zeros. We have the following equations where, as before, \( g_m(x) \) for an integer \( m \) denotes a linear function of all state variables \( x \):

\[
\Sigma_0 : \begin{align*}
\dot{x}_0 &= A_0 x_0 + \sum_{i=1}^{k} A_{0,i} y_i, \quad \text{(Invariant Zero dynamics)} \\
\end{align*}
\]

\[
\Sigma_1 : \begin{align*}
\dot{x}_{1,k} &= g_1(x) + u_1, \quad \text{(First order infinite zero dynamics)} \\
\end{align*}
\]

\[
\Sigma_2 : \begin{align*}
\dot{x}_{2,k-1} &= x_{2,k} + \ell_{2,k-1,1} x_{1,k} \\
\dot{x}_{2,k} &= g_2(x) + u_2, \quad \text{(Second order infinite zero dynamics)} \\
\end{align*}
\]

\[
\Sigma_3 : \begin{align*}
\dot{x}_{3,k-2} &= x_{3,k-1} + \ell_{3,k-2,1} x_{1,k} + \ell_{3,k-2,2} x_{2,k-1} \\
\dot{x}_{3,k-1} &= x_{3,k} + \ell_{3,k-1,1} x_{1,k} + \ell_{3,k-1,2} x_{2,k-1} \\
\dot{x}_{3,k} &= g_3(x) + u_3, \quad \text{(Third order infinite zero dynamics)} \\
\end{align*}
\]

::
\[
\begin{cases}
\dot{x}_{m,k+1-m} = x_{m,k+2-m} + \sum_{j=1}^{m-1} \ell_{m,k+1-m,j} x_{j,k+1-j} \\
\dot{x}_{m,k+2-m} = x_{m,k+3-m} + \sum_{j=1}^{m-1} \ell_{m,k+2-m,j} x_{j,k+1-j} \\
\vdots \\
\dot{x}_{m,k-1} = x_{m,k} + \sum_{j=1}^{m-1} \ell_{m,k-1,j} x_{j,k+1-j} \\
\dot{x}_{m,k} = g_m(x) + u_m, \quad (m\text{-th order infinite zero dynamics})
\end{cases}
\]

\[
\begin{cases}
\dot{x}_{k,1} = x_{k,2} + \sum_{j=1}^{k-1} \ell_{k,1,j} x_{j,k+1-j} \\
\dot{x}_{k,2} = x_{k,3} + \sum_{j=1}^{k-1} \ell_{k,2,j} x_{j,k+1-j} \\
\vdots \\
\dot{x}_{k,k-1} = x_{k,k} + \sum_{j=1}^{k-1} \ell_{k,k-1,j} x_{j,k+1-j} \\
\dot{x}_{k,k} = g_k(x) + u_k, \quad (k\text{-th order infinite zero dynamics})
\end{cases}
\]

The state \( x \) of the given system \( \Sigma \) and its substates are defined as

\[
x = \begin{bmatrix}
x_0 \\
x_1 \\
x_2 \\
v_k \\
\vdots \\
x_k \end{bmatrix}, \quad x_1 = [x_{1,k}] \\
x_2 = \begin{bmatrix} x_{2,k-1} \\
x_{2,k} \end{bmatrix}, \quad x_m = \begin{bmatrix} x_{m,k+1-m} \\
x_{m,k+2-m} \\
\vdots \\
x_{m,k} \end{bmatrix}, \quad x_k = \begin{bmatrix} x_{k,1} \\
x_{k,2} \\
\vdots \\
x_{k,k} \end{bmatrix}
\]

Notationally, the states \( x_0 \), and \( x_m \), \( m = 1 \) to \( k \) pertain respectively to the dynamics of invariant zeros, and \( m\text{-th order infinite zeros}. Their dimensions are respectively \( n_0 \), and \( mn_m \), \( m = 1 \) to \( k \). Each state \( x_m \) has \( m \) components (substates) denoted by \( x_{m,j} \), \( j = k+1-m \) to \( k \). The dimension of each \( x_{m,j} \) is \( n_m \). In the notation \( x_{m,j} \), the first subscript \( m \) always refers to the order of infinite zeros, and the second subscript \( j \) refers to a particular component of \( x_m \) with \( j = k \) being its last component. A word about the subscript notation of the coefficient matrix \( \ell_{m,i,j} \) is in order. In general, the subscripts \( m \) and \( i \) in \( \ell_{m,i,j} \) refer to the dynamic equation of the state \( x_{m,i} \) in which \( \ell_{m,i,j} \) appears, and the subscript \( j \) refers to the output component \( y_j \) to which the coefficient belongs.

The input \( u \) and the output \( y \) are defined as

\[
u = \begin{bmatrix} u_1 \\
u_2 \\
\vdots \\
u_k \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\
y_2 \\
\vdots \\
y_k \end{bmatrix} = \begin{bmatrix} x_{1,k} \\
x_{2,k-1} \\
\vdots \\
x_{k,1} \end{bmatrix}
\]

Notationally, \( u_m \) is the input to the dynamics of \( m\text{-th order infinite zeros}, while the first component \( x_{m,k+1-m} \in x_m \) is the output \( y_m \) of \( m\text{-th order infinite zero chain. The dimension of both } u_m \text{ and } y_m \text{ is } n_m. \)
4.2 Utilization of integral operations at the output side

Our aim at first is to transform the $m$-th order infinite zero dynamics of $\Sigma_m$, $m = 1$ to $k - 1$, into $k$-th order infinite zero dynamics. This can be done by adding $k - m$ number of integrators at the output end of $\Sigma_m$. This modifies and extends the subsystems $\Sigma_m$, $m = 1$ to $k - 1$. For clarity of notation, all the state variables are denoted by $z$ instead of $x$. This way, whenever we encounter a variable of the type $z_{m,j}$, we recognize easily that it belongs to the extended system. Similarly, a subsystem $\Sigma_m$ of $\Sigma$ is denoted in extended system as $\Sigma_{zm}$, and the entire extended system is denoted by $\Sigma_z$. We use the integral operator $I_n$ to denote the $n$-th integral of $h$, that is

$$I_n h = \int \int \ldots \int h(t) dt.$$ 

Similarly, we define a differential operator $D_n$ to denote the $n$-th differential of $h$, that is

$$D_n h = \frac{d^n h(t)}{dt^n}.$$ 

We note that $z_{m,k+1-m}$ is the output of the subsystem $\Sigma_m$ of the given system $\Sigma$ in the notation of $z$ variables. Let

$$z_{m,j} = I_{k+1-(m+j)} z_{m,k+1-m}$$

for $m = 1$ to $k - 1$, and $1 \leq j \leq k - m$.

Thus,

$$z_{1,1} = I_{k-1} z_{1,k}, \quad z_{1,2} = I_{k-2} z_{1,k}, \quad \ldots \quad z_{1,k-1} = I_1 z_{1,k},$$

$$z_{2,1} = I_{k-2} z_{2,k-1}, \quad z_{2,2} = I_{k-3} z_{2,k-1}, \quad \ldots \quad z_{2,k-2} = I_1 z_{2,k-1},$$

$$z_{3,1} = I_{k-3} z_{3,k-2}, \quad z_{3,2} = I_{k-4} z_{3,k-2}, \quad \ldots \quad z_{3,k-3} = I_1 z_{3,k-2},$$

$$\vdots$$

$$z_{m,1} = I_{k-m} z_{m,k+1-m}, \quad z_{m,2} = I_{k-m-1} z_{m,k+1-m}, \quad \ldots \quad z_{m,k-m} = I_1 z_{m,k+1-m},$$

$$\vdots$$

$$z_{k-1,1} = I_1 z_{k-1,2}.$$  

Let

$$z = \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_k \end{bmatrix}, \quad z_1 = \begin{bmatrix} z_{1,1} \\ z_{1,2} \\ \vdots \\ z_{1,k} \end{bmatrix}, \quad z_2 = \begin{bmatrix} z_{2,1} \\ z_{2,2} \\ \vdots \\ z_{2,k} \end{bmatrix}, \quad \ldots \quad z_m = \begin{bmatrix} z_{m,1} \\ z_{m,2} \\ \vdots \\ z_{m,k} \end{bmatrix}, \quad \ldots \quad z_k = \begin{bmatrix} z_{k,1} \\ z_{k,2} \\ \vdots \\ z_{k,k} \end{bmatrix}. $$
We point out explicitly that all the states $z_m$, have $k$ substates and are of dimension $kn_m$, $m = 1, \ldots, k$. The substate $z_{m,j}$ is the $j$-th component of $z_m$ and has dimension $n_m$. The inputs and outputs of $\Sigma_z$ are respectively given by

$$
\begin{align*}
  u_z &= u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_k \end{bmatrix}, \\
  y_z &= y = \begin{bmatrix} y_{z_1} \\ y_{z_2} \\ \vdots \\ y_{z_k} \end{bmatrix} = \begin{bmatrix} z_{1,1} \\ z_{1,2} \\ \vdots \\ z_{1,k} \\ z_{2,1} \\ z_{2,2} \\ \vdots \\ z_{2,k} \\ \vdots \\ z_{k,1} \end{bmatrix} = \begin{bmatrix} I_{k-1} & z_{1,k} \\ I_{k-2} & z_{2,k-1} \\ \vdots & \vdots \\ z_{k,1} \end{bmatrix}.
\end{align*}
$$

We consider the first component $z_{m,1}$ of each $z_m$ is now the new output variable of $\Sigma_{zm}$, while $z_{m,k+1-m}$ is the output of the subsystem $\Sigma_m$ of the given system $\Sigma$ in the notation of $z$ variables. The dynamics of $\Sigma_z$ can then be described as

$$
\begin{align*}
  \Sigma_{z_0} : & \{ \dot{z}_0 = A_0 z_0 + \sum_{j=1}^{k} A_{z_0,j} z_{j,k+1-j}, \quad \text{(Invariant Zero dynamics)} \}
  \\
  \Sigma_{z_1} : & \begin{cases} 
    \dot{z}_{1,1} = z_{1,2} \\
    \dot{z}_{1,2} = z_{1,3} \\
    \vdots \\
    \dot{z}_{1,k} = g_1(z) + u_1, \quad \text{($k$-th order infinite zero dynamics)}
  \end{cases}
  \\
  \Sigma_{z_2} : & \begin{cases} 
    \dot{z}_{2,1} = z_{2,2} \\
    \dot{z}_{2,2} = z_{2,3} \\
    \vdots \\
    \dot{z}_{2,k-2} = z_{2,k-1} \\
    \dot{z}_{2,k-1} = z_{2,k} + \ell_{2,k-1,1} z_{1,k} \\
    \dot{z}_{2,k} = g_2(z) + u_2. \quad \text{($k$-th order infinite zero dynamics)}
  \end{cases}
  \\
  \text{In general,} \\
  \Sigma_{z_i} : & \begin{cases} 
    \dot{z}_{m,1} = z_{m,2} \\
    \dot{z}_{m,2} = z_{m,3} \\
    \vdots \\
    \dot{z}_{m,k-m} = z_{m,k+1-m} \\
    \dot{z}_{m,k+1-m} = z_{m,k+2-m} + \sum_{j=1}^{m-1} \ell_{m,k+1-m,j} \dot{z}_{j,k+1-j} \\
    \vdots \\
    \dot{z}_{m,m-1} = z_{m,m} + \sum_{j=1}^{m-1} \ell_{m,m-1,j} \dot{z}_{j,m-1-j} \\
    \dot{z}_{m,k} = g_m(z) + u_m. \quad \text{($k$-th order infinite zero dynamics)}
  \end{cases}
  \\
  \text{Finally,} \quad \Sigma_{zk} \text{ is defined by} \\
  \Sigma_{zk} : & \begin{cases} 
    \dot{z}_{k,1} = z_{k,2} + \sum_{j=1}^{k} \ell_{k,1,j} \dot{z}_{j,k+1-j} \\
    \dot{z}_{k,2} = z_{k,3} + \sum_{j=1}^{k} \ell_{k,2,j} \dot{z}_{j,k+1-j} \\
    \vdots \\
    \dot{z}_{k,i} = z_{k,i+1} + \sum_{j=1}^{k} \ell_{k,i,j} \dot{z}_{j,k+1-j} \\
    \vdots \\
    \dot{z}_{k,k} = g_k(z) + u_k. \quad \text{($k$-th order infinite zero dynamics)}
  \end{cases}
\end{align*}
$$

34
The above new system $\Sigma_z$ is obtained with the use of a post-compensator in the form of integrators. Clearly, all $\Sigma_{zm}$, $m = 1$ to $k$, together describe the dynamics of $k$-th order infinite zeros. However, while $\Sigma_{z1}$ is in the format of a uniform rank system in SCB format, the other subsystems are not because of the presence of output injection via the coefficients of the type $\ell_{m,i,j}$. That is, the subsystem $\Sigma_{zm}$, for $m = 2$ to $k$, is linked to the other subsystems $\Sigma_{zm}$, for $j < m$ via the links of the type $\ell_{m,i,j}$. One of our goals then is to rewrite the dynamics of $\Sigma_{zm}$, $m = 2$ to $k$, in the format of a uniform rank system in SCB format.

Another aspect that needs to be rectified is this. As it can be observed easily, the inputs to the subsystem $\Sigma_{z0}$ that represents the finite zero structure of $\Sigma_z$ are the original outputs of $\Sigma$. In the process of transforming each $\Sigma_{zm}$, for $m = 2$ to $k$, to a uniform rank system, it is necessary to redefine the outputs. This necessitates re-writing the dynamics of $\Sigma_{z0}$ so that its inputs are only the designated outputs.

The first goal can be accomplished by appropriately re-defining the output of each subsystem as a certain linear combination of the outputs of the given system $\Sigma$ and their integrals. As we transform each subsystem into a uniform rank system, we change the notation of state variables from $z$ to $w$ to signify that a uniform rank system is being obtained or already obtained.

We construct below two types of post-compensators.

The first type of post-compensator keeps the same number of integrators as used in transforming the given system $\Sigma$ to $\Sigma_z$ without using any more additional integrators. In this sense, this type of post-compensator uses absolutely minimum number of integrators that are necessary for the construction of a post-compensator. We pursue this in Section 4.3.

The second type of post-compensator uses more integrators than those used in the first type of post-compensator, however, it is easy to construct. This is pursued in Section 4.4.

### 4.3 Construction of a uniform rank system with the use of a post-compensator that uses absolutely minimum number of integrators

In this section, we construct the first type of post-compensator as alluded to in the above discussion. In order to motivate and develop the required concepts, we present first five examples in Subsections 4.3.1 to 4.3.5. In Subsection 4.3.6 we develop the general results.
4.3.1 Transformation of $\Sigma_{z1}$ to a uniform rank system $\Sigma_{w1}$

We first consider $\Sigma_{z1}$, which is already in the format of a uniform rank system. All we need to do is simply change the state variables from $z$ to $w$.

$$\Sigma_{w1} : \left\{ \begin{array}{ll}
\dot{w}_{1,j} &= w_{1,j+1} \quad \text{for } j = 1 \text{ to } k - 1, \\
\dot{w}_{1,k} &= g_1(z) + u_1, \quad (k\text{-th order infinite zero dynamics})
\end{array} \right.$$  (4.3)

which is in the format of a uniform rank system with $y_{w1} := w_{1,1}$ considered as its output.

4.3.2 Transformation of $\Sigma_{z2}$ to a uniform rank system $\Sigma_{w2}$

We consider next $\Sigma_{z2}$ and transform it to a uniform rank system which is denoted by $\Sigma_{w2}$. At first, we select the output of $\Sigma_{w2}$ as a linear combination of $z_{2,1}$ (the output of $\Sigma_{z2}$), and $z_{1,1}, z_{1,2}, \cdots, z_{1,k}$ (the output and integrals of the output of $\Sigma_{1}$).

$$y_{w2} := w_{2,1} := z_{2,1} + \beta_{2,k-1,1} z_{1,1} + \beta_{2,0,1,1} z_{1,k},$$

$$y_{w2} := w_{2,1} := z_{2,1} + \beta_{2,k-1,1} \int_{k-1} z_{1,k} + \beta_{2,1,1} z_{1,k} + \beta_{2,0,1,1} z_{1,k},$$

$$y_{w2} := w_{2,1} := z_{2,1} + \sum_{j=0}^{k-1} \beta_{2,j,1} \int_j z_{1,k}.$$  

Here various coefficients $\beta_{2,i,j}$ are yet unknown. The notation for these coefficients at a first glance seems to be chaotic and formidable. However, the first index 2 in $\beta_{2,i,j}$ simply refers to the subsystem we are considering here. The second index $i$ and the third index $j$ have a deeper meaning. The variable to which $\beta_{2,i,j}$ is a coefficient is obtained after $i$ integrations of the output of subsystem $\Sigma_j$ of the given system $\Sigma$. To make the notation self evident, the output $y_{w2}$ is expressed in different but equivalent ways.

The unknown coefficients $\beta_{2,i,j}$ are chosen as we proceed to remove the existing feedback terms in $\Sigma_{z2}$, which emanate from the output $z_{1,k}$ of $\Sigma_{1}$. To reach such feedback terms for the first time, $w_{2,1}$ needs to be differentiated $k - 1$ times. In the process of such differentiation, all those output terms associated with the integral operator $\int_j$ for $j < k - 1$ will be differentiated. Such differentials are not among the feedback terms in $\Sigma_{z2}$ which we plan to eliminate or cancel, and hence, with the exception of $z_{2,1}$, all those terms associated with the integral operator $\int_j$, for $j < k - 1$, need to be removed from the assumed $y_{w2}$ as given above. This discussion leads us to choose $y_{w2}$ as

$$y_{w2} := w_{2,1} := z_{2,1} + \beta_{2,k-1,1} z_{1,1},$$

$$y_{w2} := w_{2,1} := z_{2,1} + \beta_{2,k-1,1} \int_{k-1} z_{1,k}.$$  

This yields

$$\dot{z}_{2,j} = \dot{w}_{2,j+1} + \beta_{2,k-1,1} \int_{j-1} z_{1,k} := w_{2,j+1}, \quad \text{for } j = 1 \text{ to } k - 2$$

$$\dot{w}_{2,k-1} = \dot{z}_{2,k} + \sum_{j=1}^{k-1} \dot{z}_{2,j} + \beta_{2,k-1,1} z_{1,k}.$$
The last expression enables us to select
\[ \beta_{2,k-1,1} = -\ell_{2,k-1,1} \]
in order to cancel the feedback term represented by the spade ♠. A quick glance at the above expression reveals that the seemingly chaotic notation we used for an unknown coefficient pays itself in terms of simplicity in identifying the coefficient of canceled feedback term in \( w_{2,k-1} \).

We select next \( w_{2,k} := z_{2,k} \) to get,
\[ \dot{w}_{2,k} = g_2(z) + u_2. \]
It is now straightforward to summarize the above development as,
\[ y_{w_{2}} := w_{2,1} := z_{2,1} - \ell_{2,k-1,1}z_{1,1} \]
\[ w_{2,j} := z_{2,j} - \ell_{2,k-1,1}z_{1,j}, \text{ for } j = 2 \text{ to } k - 1 \]
\[ w_{2,k} := z_{2,k} \]
and
\[ \Sigma_{w_{2}} : \begin{cases} \dot{w}_{2,j} &= w_{2,j+1} \text{ for } j = 1 \text{ to } k - 1, \\
\dot{w}_{2,k} &= g_2(z) + u_2, \end{cases} \tag{4.4} \]
which is in the format of a uniform rank system in SCB format with \( y_{w_{2}} := w_{2,1} \) as its output.

**Observation 1:** We emphasize that selecting the output appropriately is the key that leads to a uniform rank system. To illustrate this, select erroneously the output as \( \tilde{y}_{w_{2}} = z_{2,1} \). After \( k - 1 \) differentiations of it, we find that
\[ \dot{w}_{2,k-1} = z_{2,k} + \ell_{2,k-1,1}z_{1,k}. \]
Obviously, the feedback term cannot be canceled. Nevertheless, in an effort to obtain a uniform rank system, we might define,
\[ w_{2,k} := z_{2,k} + \ell_{2,k-1,1}z_{1,k}. \]
Then,
\[ \dot{w}_{2,k} := g_2(w) + \ell_{2,k-1,1}\dot{z}_{1,k} + u_2. \]
However, the presence of the term \( \dot{z}_{1,k} \) is not allowed in a uniform rank system, and hence the chosen output \( \tilde{y}_{w_{2}} = z_{2,1} \) is incorrect.

### 4.3.3 Transformation of \( \Sigma_{z_{3}} \) to a uniform rank system \( \Sigma_{w_{3}} \)
We consider next \( \Sigma_{z_{3}} \) and transform it to a uniform rank system \( \Sigma_{w_{3}} \).

At first, we select the output of \( \Sigma_{w_{3}} \) as a linear combination of \( z_{3,1} \) (the output of \( \Sigma_{z_{3}} \)), \( z_{2,1}, z_{2,2}, \cdots, z_{2,k-1} \) (output and integrals of the output of \( \Sigma_{2} \)), and \( z_{1,1}, z_{1,2}, \cdots, z_{1,k} \) (the output and integrals of the output of \( \Sigma_{1} \)),
\[ y_{w_{3}} := z_{3,1} := z_{3,1} + \sum_{j=0}^{k-2} \beta_{3,j,2}\dot{I}_j z_{2,k-1} + \sum_{j=0}^{k-1} \beta_{3,j,1}\dot{I}_j, \]
\[ := z_{3,1} + \beta_{3,k-1,1}\dot{I}_k - 1 z_{1,k} + \sum_{j=0}^{k-2} \sum_{m=1}^{2} \beta_{3,j,m}\dot{I}_j z_{m,k+1-m}. \]
Various coefficients $\beta_{3,i,j}$ are yet unknown. Once again, the notation for these coefficients at a first glance seems to be chaotic and formidable. However, the first index 3 in $\beta_{3,i,j}$ simply refers to the subsystem we are considering here. The second index $i$ and the third index $j$ have a deeper meaning. The variable to which $\beta_{3,i,j}$ is a coefficient is obtained after $i$ integrations of the output of subsystem $\Sigma_j$ of the given system $\Sigma$. Note that the output $y_{w3}$ is expressed in two different but equivalent ways as each representation has a merit in subsequent work.

The unknown coefficients $\beta_{3,i,j}$ are chosen as we proceed to remove the existing feedback terms in $\Sigma_{z3}$, which are the outputs of the form $z_{j,k-j+1}$ of $\Sigma_j$ for $j = 1$ and 2. To reach such feedback terms for the first time, $w_{3,1}$ needs to be differentiated $k - 2$ times. In the process of such differentiation, all those outputs associated with the integral operator $I_j$ for $j < k - 2$ will be differentiated. Such differentials are not among the feedback terms in $\Sigma_{z3}$ which we plan to eliminate or cancel, and hence, with the exception of $z_{3,1}$, all those terms associated with the integral operator $I_j$, for $j < k - 2$, need to be removed from the assumed $y_{w3}$ as given above. This discussion leads us to choose $y_{w3}$ as

$$y_{w3} := w_{3,1} := z_{3,1} + I_{k-2} \beta_{3,k-2,1} z_{1,k} + I_{k-1} \beta_{3,k-1,1} z_{1,k} + I_{k-2} \beta_{3,k-2,2} z_{2,k-1}$$

$$:= z_{3,1} + I_{k-2} \Sigma_{j=1}^{2} \beta_{3,k-2,j} z_{j,k+1-j} + I_{k-1} \beta_{3,k-1,1} z_{1,k}.$$ 

This yields

$$\dot{w}_{3,m} = z_{3,m+1} + \beta_{3,k-1,1} I_{k-m-1} z_{1,k} + I_{k-m-2} \Sigma_{j=1}^{2} \beta_{3,k-2,j} z_{j,k+1-j} := w_{3,m+1}, \text{ for } m = 1 \text{ to } k-3$$

$$\dot{w}_{3,k-2} = z_{3,k-1} + \ell_{3,k-2,1} z_{1,k} + \ell_{3,k-2,2} z_{2,k-1}$$

$$+ \beta_{3,k-1,1} I_{1} z_{1,k} + \beta_{3,k-2,1} z_{1,k} + \beta_{3,k-2,2} z_{2,k-1} := w_{3,k-1}.$$ 

Examination of the last expression enables us to decide,

$$\beta_{3,k-2,1} = -\ell_{3,k-2,1} \quad \text{and} \quad \beta_{3,k-2,2} = -\ell_{3,k-2,2},$$

so that the feedback terms represented by the spade ♠ are eliminated. This implies that

$$\dot{w}_{3,k-2} = z_{3,k-1} + \beta_{3,k-1,1} I_{1} z_{1,k} = z_{3,k-1} + \beta_{3,k-1,1} z_{1,k-1} := w_{3,k-1}.$$ 

This leads to

$$\dot{w}_{3,k-1} = z_{3,k} + \ell_{3,k-1,1} z_{1,k} + \ell_{3,k-1,2} z_{2,k-1}$$

$$+ \beta_{3,k-1,1} z_{1,k} := w_{3,k}.$$ 

This enables us to choose,

$$\beta_{3,k-1,1} = -\ell_{3,k-1,1}, \quad \text{and thus} \quad w_{3,k} := z_{3,k} + \ell_{3,k-1,2} z_{2,k-1}.$$ 

38
Note that not all feedback terms are canceled. The one represented by the diamond ♦ is not canceled, it is simply passed on to the last step to be absorbed by \( \tilde{g}_3(z) \) for some linear function of \( z \). Finally, we get

\[
\hat{w}_{3,k} = \tilde{g}_3(w) + u_3.
\]

We can now summarize the above development as

\[
\hat{w}_{3,1} := z_{3,1} - \ell_{3,k-1,1}z_{1,1} - \ell_{3,k-2,1}z_{1,2} - \ell_{3,k-2,2}z_{2,1}
\]

\[
w_{3,m} := z_{3,m} - \ell_{3,k-1,1}z_{1,m} - \ell_{3,k-2,1}z_{1,m+1} - \ell_{3,k-2,2}z_{2,m}, \text{ for } m = 2 \text{ to } k - 2
\]

\[
w_{3,k-1} := z_{3,k-1} - \ell_{3,k-1,1}z_{1,k-1}
\]

\[
w_{3,k} := z_{3,k} + \ell_{3,k-1,2}z_{2,k-1},
\]

and

\[
\Sigma_{w3} : \left\{ \begin{array}{l}
\dot{w}_{3,j} = w_{3,j+1} \text{ for } j = 1 \text{ to } k - 1, \\
\dot{w}_{3,k} = \tilde{g}_3(w) + u_3,
\end{array} \right.
\]

which is in the format of a uniform rank system in SCB format with \( y_{w3} := w_{3,1} \) as its output.

### 4.3.4 Transformation of \( \Sigma_{z4} \) to a uniform rank system \( \Sigma_{w4} \)

We consider next \( \Sigma_{z4} \) and transform it to a uniform rank system \( \Sigma_{w4} \). At first, we select the output of \( \Sigma_{w4} \) as a linear combination of the output of \( \Sigma_{z4} \), and the output and their integrals of subsystems \( \Sigma_j \) for \( j = 1 \) to 3 of the given system \( \Sigma \) as given below:

\[
y_{w4} := w_{4,1} := z_{4,1} + \sum_{j=0}^{k-3} I_j \beta_{4,j,1} z_{3,j-2}
\]

\[
+ \sum_{j=0}^{k-2} I_j \beta_{4,j,2} z_{2,k-1}
\]

\[
+ \sum_{j=0}^{k-1} I_j \beta_{4,j,1} z_{1,k}.
\]

As before, various coefficients \( \beta_{4,i,j} \) are yet unknown. The notation for these unknown coefficients is as described earlier in previous subsections.

The unknown coefficients \( \beta_{4,i,j} \) are chosen as we proceed to remove the existing feedback terms in \( \Sigma_{z4} \), which are the outputs of the form \( z_{j,k-j+1} \) of \( \Sigma_j \) for \( j = 1 \) to 3. To reach such feedback terms for the first time, \( w_{4,1} \) needs to be differentiated \( k - 3 \) times. In the process of such differentiation, all those outputs associated with the integral operator \( I_j \) for \( j < k - 3 \) will be differentiated. Such differentials are not among the feedback terms in \( \Sigma_{z4} \) which we plan to eliminate or cancel, and hence, with the exception of \( z_{4,1} \), all those terms associated with the integral operator \( I_j \), for \( j < k - 3 \), need to be removed from the assumed
We note that, in the notation of \( z \) variables, \( z_{j,k+1-j} \) is the output of the subsystem \( \Sigma_j \) of the given system \( \Sigma \), \( j = 1 \) to \( 3 \). Thus, we have 
\[
\Sigma_j^3 = \beta_{4,k-3,j} z_{j,k+1-j}
\]
as a linear combination of the outputs of the subsystems \( \Sigma_j \), \( j = 1 \) to \( 3 \),
\[
\Sigma_j^2 = \beta_{4,k-2,j} z_{j,k+1-j}
\]
as a linear combination of the outputs of the subsystems \( \Sigma_j \), \( j = 1 \) to \( 2 \),
and \( \beta_{4,k-1,1} z_{1,k} \) as a constant times the output of the subsystem \( \Sigma_1 \).

We proceed now to describe the algorithmic process of determining the coefficients \( \beta_{4,i,j} \) in \( y_{w4} = w_{4,1} \).

**Steps 1 to \( k - 4 \):** At first, we define,
\[
w_{4,m} = D_{m-1} w_{4,1}, \quad \text{for } m = 2 \text{ to } k - 3.
\]

**Step \( k - 3 \):** When \( y_{w4} = w_{4,1} \) is differentiated \( k - 3 \) times, we get
\[
\dot{w}_{4,k-3} = z_{4,k-2} + \sum_{j=1}^{3} \ell_{4,k-3,j} \dot{z}_{j,k+1-j}
\]
\[
+ \sum_{j=1}^{3} \beta_{4,k-3,j} z_{j,k+1-j}
\]
\[
+ I_1 \sum_{j=1}^{2} \beta_{4,k-2,j} z_{j,k+1-j}
\]
\[
+ I_2 \beta_{4,k-1,1} z_{1,k}.
\]

Examination of the above expression enables us to decide,
\[
\beta_{4,k-3,j} = -\ell_{4,k-3,j}, \quad j = 1 \text{ to } 3.
\]

We choose,
\[
w_{4,k-2} := z_{4,k-2} + I_1 \sum_{j=1}^{2} \beta_{4,k-2,j} z_{j,k+1-j} + I_2 \beta_{4,k-1,1} z_{1,k}.
\]

**Step \( k - 2 \):** We obtain,
\[
\dot{w}_{4,k-2} = z_{4,k-1} + \sum_{j=1}^{2} \ell_{4,k-2,j} \dot{z}_{j,k+1-j} + \sum_{j=1}^{3} \beta_{4,k-2,j} z_{j,k+1-j}
\]
\[
+ I_1 \sum_{j=1}^{2} \beta_{4,k-2,j} z_{j,k+1-j} + I_2 \beta_{4,k-1,1} z_{1,k}.
\]
We emphasize that not all feedback terms in \( \dot{w}_{4,k-2} \) are canceled. The feedback term designated by the diamond ♦ in \( \dot{w}_{4,k-2} \) is not canceled. This uncanceled term is the feedback from the subsystem \( \Sigma_3 \), and it is included as a component of \( w_{4,k-1} \). We will discuss shortly the consequence of passing on an uncanceled term to later stages.

**Step \( k-1 \):** We obtain next,
\[
\dot{w}_{4,k-1} = z_{4,k-1} + \ell_{4,k-3,3}z_{3,k-2} + \mathcal{I}_1\beta_{4,k-1,1} z_{1,k}.
\]

The uncanceled term \( \ell_{4,k-2,3}z_{3,k-2} \) in \( \dot{w}_{4,k-2} \), after one differentiation, generates other secondary feedback terms in \( \dot{w}_{4,k-1} \), namely
\[
\ell_{4,k-2,3}\left[z_{3,k-1} + \ell_{3,k-2,1}z_{1,k} + \ell_{3,k-2,2}z_{2,k-1}\right].
\]

The meaning behind designating a term by a club ♠ or by a spade ♠ or by a diamond ♦ will be explained clearly in a later section when we discuss transforming a general \( \Sigma_{zm} \) to a \( \Sigma_{wm} \).

In general, secondary feedback terms arise because of certain uncanceled feedback terms in the previous steps which are passed on to later stage substates. It is important to cancel the presence of \( z_{1,k} \) in \( \dot{w}_{4,k-1} \) of the above expression, irrespective of the way it arises, by choosing appropriately \( \beta_{4,k-1,1} \). Otherwise, it will generate a term \( \dot{z}_{1,k} \) or \( \dot{z}_{1,k+1} \) in a subsequent differentiation that leads to \( \dot{w}_{4,k} \), and the presence of such \( \dot{z}_{1,k} \) in \( \dot{w}_{4,k} \) is unwarranted. Thus, we choose
\[
\beta_{4,k-1,1} = -\ell_{4,k-1,1} - \ell_{4,k-2,3}\ell_{3,k-2,1}.
\]

Unlike in the previous cases, the coefficient \( \beta_{4,k-1,1} \) chosen above cancels two feedback terms both arising by a feedback from \( z_{1,k} \), one that was naturally present in \( \dot{w}_{4,k-1} \) and the other that arose because of a non-canceled term in \( \dot{w}_{4,k-2} \) as explained above. We choose next
\[
w_{4,k} := z_{4,k} + \mathcal{I}_j^{3}\ell_{4,k-3,j}z_{j,k+1-j} + \ell_{4,k-2,3}\left[z_{3,k-1} + \ell_{3,k-2,2}z_{2,k-1}\right].
\]

**Step \( k \):** Finally, we get
\[
\dot{w}_{4,k} = \tilde{g}_4(z) + u_4,
\]
for some linear function \( \tilde{g}_4(z) \). We can now summarize the above development as,
\[
y_{w4} := w_{4,1} := \begin{cases} 
  z_{4,1} - \mathcal{I}_{k-3}\mathcal{I}_{j=1}\ell_{4,k-3,j}z_{j,k+1-j} \\
  -\mathcal{I}_{k-2}\mathcal{I}_{j=1}\ell_{4,k-2,j}z_{j,k+1-j} \\
  -\mathcal{I}_{k-1}\ell_{4,k-1,1}z_{1,k},
\end{cases}
\]
where
\[ \ell_{4,k-1,1} = \ell_{4,k-1,1} + \ell_{4,k-2,3}\ell_{3,k-2,1}. \]

Similarly, we define
\[
\begin{align*}
w_{4,m} &:= z_{4,m} - \mathcal{I}_{k-m-2} \sum_{j=1}^{3} \ell_{4,k-3,j} z_{j,k+1-j} \\
 &\quad - \mathcal{I}_{k-m-1} \sum_{j=1}^{2} \ell_{4,k-2,j} z_{j,k+1-j} \\
 &\quad - \mathcal{I}_{k-m} \ell_{4,k-1,1} z_{1,k}, \quad \text{for } m = 2 \text{ to } k - 3 \\
w_{4,k-2} &:= z_{4,k-2} - \mathcal{I}_{1} \sum_{j=2}^{3} \ell_{4,k-2,j} z_{j,k+1-j} - \mathcal{I}_{2} \ell_{4,k-1,1} z_{1,k} \\
w_{4,k-1} &:= z_{4,k-1} + \ell_{4,k-2,3} z_{3,k-2} - \mathcal{I}_{1} \ell_{4,k-1,1} z_{1,k} \\
w_{4,k} &:= z_{4,k} + \sum_{j=2}^{3} \ell_{4,k-1,j} z_{j,k+1-j} + \ell_{4,k-2,3} [z_{3,k-1} + \ell_{3,k-2,2} z_{2,k-1}].
\end{align*}
\]

The above definition of substates leads us to
\[
\Sigma_{w4} : \begin{cases} 
\dot{w}_{4,j} = w_{4,j+1} & \text{for } j = 1 \text{ to } k - 1, \\
\dot{w}_{4,k} = \tilde{g}_4(z) + u_4, \end{cases} \tag{4.6}
\]

for some linear function \( \tilde{g}_4(z) \). The above system is in the format of a uniform rank system in SCB format with \( y_{w4} := w_{4,1} \) as its output.

### 4.3.5 Transformation of \( \Sigma_{z7} \) to a uniform rank system \( \Sigma_{w7} \)

We consider next \( \Sigma_{z7} \) and transform it to a uniform rank system \( \Sigma_{w7} \) in SCB format. At first, we select the output of \( \Sigma_{w7} \) as a linear combination of the output of \( \Sigma_{z7} \), and the output and their integrals of subsystems \( \Sigma_j \) for \( j = 1 \) to 6 of the given system \( \Sigma \) as given below:

\[
y_{w7} := w_{7,1} := z_{7,1} + \sum_{j=0}^{k-6} \mathcal{I}_j \beta_{7,j,6} z_{0,k-5} \\
+ \sum_{j=0}^{k-5} \mathcal{I}_j \beta_{7,j,5} z_{0,k-4} \\
+ \sum_{j=0}^{k-4} \mathcal{I}_j \beta_{7,j,4} z_{4,k-3} \\
+ \sum_{j=0}^{k-3} \mathcal{I}_j \beta_{7,j,3} z_{3,k-2} \\
+ \sum_{j=0}^{k-2} \mathcal{I}_j \beta_{7,j,2} z_{2,k-1} \\
+ \sum_{j=0}^{k-1} \mathcal{I}_j \beta_{4,j,1} z_{1,k}.
\]

As before, various coefficients \( \beta_{7,i,j} \) are yet unknown. The notation for these unknown coefficients is as described earlier in previous subsections.

The unknown coefficients \( \beta_{7,i,j} \) are chosen as we proceed to remove the existing feedback terms in \( \Sigma_{z7} \), which are the outputs of the form \( z_{j,k-j+1} \) of \( \Sigma_j \) for \( j = 1 \) to 6. To reach such feedback terms for the first time, \( w_{7,1} \) needs to be differentiated \( k-6 \) times. In the process of such differentiation, all those outputs associated with the integral operator \( \mathcal{I}_j \) for \( j < k-6 \) will be differentiated. Such differentials are not among the feedback terms in \( \Sigma_{z7} \) which we plan to eliminate or cancel, and hence, with the exception of \( z_{7,1} \), all those terms associated with
the integral operator $I_j$, for $j < k - 6$, need to be removed from the assumed $y_{w7}$ as given above. This discussion leads us to choose $y_{w7}$ as

$$y_{w7} := w_{7,1} := z_{7,1} + \sum_{n=1}^{6} I_{k+n-7} z_{j,k+n-7} \beta_{7,k+n-7,j}$$

$$:= z_{7,1} + \sum_{n=1}^{6} \sum_{j=1}^{\gamma} \beta_{7,k-n,j} z_{j,k+1-j}$$

$$+ \sum_{j=1}^{5} \sum_{j=1}^{\gamma} \beta_{7,k-5,j} z_{j,k+1-j}$$

$$+ \sum_{j=1}^{4} \sum_{j=1}^{\gamma} \beta_{7,k-4,j} z_{j,k+1-j}$$

$$+ \sum_{j=1}^{3} \sum_{j=1}^{\gamma} \beta_{7,k-3,j} z_{j,k+1-j}$$

$$+ \sum_{j=1}^{2} \sum_{j=1}^{\gamma} \beta_{7,k-2,j} z_{j,k+1-j}$$

$$+ \sum_{j=1}^{1} \sum_{j=1}^{\gamma} \beta_{7,k-1,j} z_{j,k+1-j}$$

We note that, in the notation of $z$ variables, $z_{j,k+1-j}$ is the output of the subsystem $\Sigma_j$ of the given system $\Sigma_j$, $j = 1$ to 6. Thus, we have

$\Sigma_j^1 = \beta_{7,k-6,j} z_{j,k+1-j}$ as a linear combination of the outputs of the subsystems $\Sigma_j$, $j = 1$ to 6,

$\Sigma_j^2 = \beta_{7,k-5,j} z_{j,k+1-j}$ as a linear combination of the outputs of the subsystems $\Sigma_j$, $j = 1$ to 5,

$\Sigma_j^3 = \beta_{7,k-4,j} z_{j,k+1-j}$ as a linear combination of the outputs of the subsystems $\Sigma_j$, $j = 1$ to 4,

$\Sigma_j^4 = \beta_{7,k-3,j} z_{j,k+1-j}$ as a linear combination of the outputs of the subsystems $\Sigma_j$, $j = 1$ to 3,

$\Sigma_j^5 = \beta_{7,k-2,j} z_{j,k+1-j}$ as a linear combination of the outputs of the subsystems $\Sigma_j$, $j = 1$ to 2,

and $\beta_{4,k-1,1} z_{1,k}$ as a constant times the output of the subsystem $\Sigma_1$.

We proceed now to describe the algorithmic process of determining the coefficients $\beta_{7,i,j}$ in $y_{w7} = w_{7,1}$.

**Steps 1 to $k - 7$:** At first, we define,

$$w_{7,m} = D_{m-1} w_{7,1}, \quad \text{for } m = 2 \text{ to } k - 6.$$  

**Step $k - 6$:** When $y_{w7} = w_{7,1}$ is differentiated $k - 6$ times, we get

$$w_{7,k-6} = z_{7,k-5} + \sum_{j=1}^{6} \sum_{j=1}^{\gamma} \beta_{7,k-6,j} z_{j,k+1-j}$$

$$+ \sum_{j=1}^{5} \sum_{j=1}^{\gamma} \beta_{7,k-5,j} z_{j,k+1-j}$$

$$+ \sum_{j=1}^{4} \sum_{j=1}^{\gamma} \beta_{7,k-4,j} z_{j,k+1-j}$$

$$+ \sum_{j=1}^{3} \sum_{j=1}^{\gamma} \beta_{7,k-3,j} z_{j,k+1-j}$$

$$+ \sum_{j=1}^{2} \sum_{j=1}^{\gamma} \beta_{7,k-2,j} z_{j,k+1-j}$$

$$+ \sum_{j=1}^{1} \sum_{j=1}^{\gamma} \beta_{7,k-1,j} z_{j,k+1-j}$$

Here the term or terms represented by a spade $\spadesuit$ are the outputs of previous subsystems that can be canceled by choosing $\beta$ coefficients appropriately. In
fact, examination of the above expression enables us to decide,

\[ \beta_{7,k-6,j} = -\ell_{7,k-6,j}, \quad j = 1 \text{ to } 6. \]

We choose,

\[
w_{7,k-5} := z_{7,k-5} + \sum_{j=1}^{5} \beta_{7,k-5,j} z_{j,k+1-j} + \sum_{j=1}^{4} \beta_{7,k-4,j} z_{j,k+1-j} + \sum_{j=1}^{3} \beta_{7,k-3,j} z_{j,k+1-j} + \sum_{j=1}^{2} \beta_{7,k-2,j} z_{j,k+1-j} + \sum_{j=1}^{1} \beta_{7,k-1,1} z_{1,k}.\]

**Step** \( k-5 \): We obtain,

\[
w_{7,k-5} = z_{7,k-4} + \left( \sum_{j=1}^{5} \ell_{7,k-5,j} z_{j,k+1-j} \right) + \left( \sum_{j=1}^{4} \ell_{7,k-4,j} z_{j,k+1-j} \right) + \left( \sum_{j=1}^{3} \ell_{7,k-3,j} z_{j,k+1-j} \right) + \left( \sum_{j=1}^{2} \ell_{7,k-2,j} z_{j,k+1-j} \right) + \ell_{7,k-1,1} z_{1,k}.\]

As before, the term or terms represented by a spade ♠ are the outputs of previous subsystems that can be canceled by choosing \( \beta \) coefficients appropriately, that is by choosing

\[ \beta_{7,k-5,j} = -\ell_{7,k-5,j} \quad \text{for} \quad j = 1 \text{ to } 5. \]

On the other hand, although it is an output of a previous subsystem, the term represented by a diamond ♦ cannot be canceled because there is no longer any \( \beta \) coefficient associated with such an output. This term becomes a part of \( w_{7,k-4} \) defined as

\[
w_{7,k-4} := z_{7,k-4} + \ell_{7,k-5,6} z_{6,k-5} + \sum_{j=1}^{4} \beta_{7,k-4,j} z_{j,k+1-j} + \sum_{j=1}^{3} \beta_{7,k-3,j} z_{j,k+1-j} + \sum_{j=1}^{2} \beta_{7,k-2,j} z_{j,k+1-j} + \beta_{7,k-1,1} z_{1,k}.\]
Step \(k - 4\): We obtain next,

\[
\dot{w}_{7,k-4} = z_{7,k-3} + \sum_{j=1}^{4} \ell_{7,k-4,j} z_{j,k+1-j} + \ell_{7,k-5,6} \sum_{j=1}^{5} \ell_{6,k-5,j} z_{j,k+1-j} + \sum_{j=1}^{5} \beta_{7,k-4,j} z_{j,k+1-j} + \sum_{j=1}^{5} \beta_{7,k-3,j} z_{j,k+1-j} + \sum_{j=1}^{5} \beta_{7,k-2,j} z_{j,k+1-j} + \sum_{j=1}^{5} \beta_{7,k-1,1} z_{1,k}.
\]

We re-write and re-order all the terms in \(\dot{w}_{7,k-4}\) to identify what can be canceled and what cannot be,

\[
\dot{w}_{7,k-4} = z_{7,k-3} + \sum_{j=1}^{4} \ell_{7,k-4,j} z_{j,k+1-j} + \sum_{j=1}^{6} \ell_{7,k-5,6} \sum_{j=1}^{5} \ell_{6,k-5,j} z_{j,k+1-j} + \sum_{j=1}^{5} \beta_{7,k-4,j} z_{j,k+1-j} + \sum_{j=1}^{5} \beta_{7,k-3,j} z_{j,k+1-j} + \sum_{j=1}^{5} \beta_{7,k-2,j} z_{j,k+1-j} + \sum_{j=1}^{5} \beta_{7,k-1,1} z_{1,k}.
\]

Here besides spades ♠ and diamonds ♦, we get another term represented by a club ♣. This term is not an output of any subsystem and can never be canceled by appropriately selecting \(\beta\) coefficients, since such \(\beta\) coefficients are associated only with the outputs of previous subsystems.

By re-ordering the terms in the above expression, we get

\[
\dot{w}_{7,k-4} = z_{7,k-3} + \sum_{j=1}^{4} \ell_{7,k-4,j} z_{j,k+1-j} + \sum_{j=1}^{6} \ell_{7,k-5,6} \sum_{j=1}^{5} \ell_{6,k-5,j} z_{j,k+1-j} + \sum_{j=1}^{5} \beta_{7,k-4,j} z_{j,k+1-j} + \sum_{j=1}^{5} \beta_{7,k-3,j} z_{j,k+1-j} + \sum_{j=1}^{5} \beta_{7,k-2,j} z_{j,k+1-j} + \sum_{j=1}^{5} \beta_{7,k-1,1} z_{1,k}.
\]
We define new coefficients as

\[ \tilde{\ell}_{7,k-4,j} = \ell_{7,k-4,j} + \ell_{7,k-5,6} \ell_{6,k-5,j} \quad \text{for} \quad j = 1 \text{ to } 5, \quad (4.7) \]

and re-write the above as

\[
\begin{align*}
\hat{w}_{7,k-4} &= \hat{w}_{7,k-3} + \sum_{j=1}^{4} \tilde{\ell}_{7,k-4,j} z_{j,k+1-j} \\
&\quad + \sum_{j=1}^{4} \beta_{7,k-4,j} z_{j,k+1-j} \\
&\quad + \ell_{7,k-4,6} z_{6,k-5} + \ell_{7,k-4,5} z_{5,k-4} \\
&\quad + \ell_{7,k-5,6} z_{6,k-4} \\
&\quad + I_1 \sum_{j=1}^{5} \beta_{7,k-3,j} z_{j,k+1-j} \\
&\quad + I_2 \sum_{j=1}^{5} \beta_{7,k-2,j} z_{j,k+1-j} \\
&\quad + I_3 \beta_{7,k-1,1} z_{1,k}.
\end{align*}
\]

We select

\[ \beta_{7,k-4,j} = -\tilde{\ell}_{7,k-4,j} \quad \text{for} \quad j = 1 \text{ to } 4. \]

By relabeling \( \tilde{\ell}_{7,k-5,6} = \ell_{7,k-5,6} \) and \( \tilde{\ell}_{7,k-4,6} = \ell_{7,k-4,6} \), we define \( w_{7,k-3} \) as,

\[
\begin{align*}
w_{7,k-3} &= \hat{w}_{7,k-3} + \sum_{j=1}^{6} \tilde{\ell}_{7,k-3,j} z_{j,k+1-j} \\
&\quad + \sum_{j=1}^{6} \beta_{7,k-3,j} z_{j,k+1-j} \\
&\quad + \ell_{7,k-4,6} z_{6,k-5} + \ell_{7,k-4,5} z_{5,k-4} \\
&\quad + \ell_{7,k-5,6} z_{6,k-4} \\
&\quad + I_1 \sum_{j=1}^{5} \beta_{7,k-3,j} z_{j,k+1-j} \\
&\quad + I_2 \sum_{j=1}^{5} \beta_{7,k-2,j} z_{j,k+1-j} \\
&\quad + I_3 \beta_{7,k-1,1} z_{1,k}.
\end{align*}
\]

**Step k – 3:** We obtain next,

\[
\begin{align*}
\hat{w}_{7,k-3} &= \hat{w}_{7,k-2} + \sum_{j=1}^{6} \tilde{\ell}_{7,k-3,j} z_{j,k+1-j} \\
&\quad + \sum_{j=1}^{6} \beta_{7,k-3,j} z_{j,k+1-j} \\
&\quad + \ell_{7,k-4,6} \left[ z_{6,k-4} + \sum_{j=1}^{4} \ell_{6,k-5,j} z_{j,k+1-j} \right] \\
&\quad + \ell_{7,k-4,5} \left[ z_{5,k-3} + \sum_{j=1}^{4} \ell_{5,k-4,j} z_{j,k+1-j} \right] \\
&\quad + \ell_{7,k-5,6} \left[ z_{6,k-3} + \sum_{j=1}^{5} \ell_{6,k-4,j} z_{j,k+1-j} \right] \\
&\quad + \sum_{j=1}^{3} \beta_{7,k-3,j} z_{j,k+1-j} \\
&\quad + I_1 \sum_{j=1}^{5} \beta_{7,k-2,j} z_{j,k+1-j} \\
&\quad + I_2 \beta_{7,k-1,1} z_{1,k}.
\end{align*}
\]

We re-write and re-order all the terms in \( \hat{w}_{7,k-3} \) to identify what can be canceled.
and what cannot be,

\[ \dot{w}_{7,k-3} = z_{7,k-2} + \sum_{j=1}^{3} \ell_{7,k-3,j} \dot{z}_{j,k+1-j} + \sum_{j=1}^{3} \ell_{7,k-4,6} \dot{z}_{6,k-5,j} \dot{z}_{j,k+1-j} \]

\[ + \sum_{j=1}^{5} \ell_{7,k-4,5} \dot{z}_{j,k+1-j} \dot{z}_{j,k+1-j} \]

\[ + \sum_{j=1}^{5} \ell_{7,k-5,6} \dot{z}_{j,k+1-j} \dot{z}_{j,k+1-j} \]

\[ + \sum_{j=1}^{5} \beta_{7,k-3,j} \dot{z}_{j,k+1-j} \]

\[ + \sum_{j=4}^{6} \ell_{7,k-4,4} \dot{z}_{4,k-3} + \sum_{j=4}^{6} \ell_{7,k-5,6} \dot{z}_{6,k-4,j} \dot{z}_{j,k+1-j} \]

\[ + \ell_{7,k-6} \dot{z}_{6,k-4} + \ell_{7,k-4,5} \dot{z}_{5,k-3} + \ell_{7,k-5,6} \dot{z}_{6,k-3} \]

\[ + I_{1} \sum_{j=1}^{2} \beta_{7,k-2,j} \dot{z}_{j,k+1-j} \]

\[ + I_{2} \beta_{7,k-1,1} \dot{z}_{1,k}. \]

Let us define

\[ \ell_{7,k-3,j} = \ell_{7,k-3,j} + \ell_{7,k-4,6} \ell_{6,k-5,j} + \ell_{7,k-4,5} \ell_{5,k-4,j} + \ell_{7,k-5,6} \ell_{6,k-4,j} \] (4.8)

for \( j = 1 \) to 4, and

\[ \ell_{7,k-3,5} = \ell_{7,k-3,5} + \ell_{7,k-4,6} \ell_{6,k-5,5} + \ell_{7,k-5,6} \ell_{6,k-4,5}, \]

\[ \ell_{7,k-3,6} = \ell_{7,k-3,6}. \] (4.9)

This enables us to re-write \( \dot{w}_{7,k-3} \) as

\[ \dot{w}_{7,k-3} = z_{7,k-2} + \sum_{j=1}^{3} \ell_{7,k-3,j} \dot{z}_{j,k+1-j} \]

\[ + \sum_{j=1}^{3} \beta_{7,k-3,j} \dot{z}_{j,k+1-j} \]

\[ + \sum_{j=4}^{6} \ell_{7,k-4,4} \dot{z}_{4,k-3} + \sum_{j=4}^{6} \ell_{7,k-5,6} \dot{z}_{6,k-4,j} \dot{z}_{j,k+1-j} \]

\[ + \ell_{7,k-6} \dot{z}_{6,k-4} + \ell_{7,k-4,5} \dot{z}_{5,k-3} + \ell_{7,k-5,6} \dot{z}_{6,k-3} \]

\[ + I_{1} \sum_{j=1}^{2} \beta_{7,k-2,j} \dot{z}_{j,k+1-j} \]

\[ + I_{2} \beta_{7,k-1,1} \dot{z}_{1,k}. \]

We select

\[ \beta_{7,k-3,j} = -\ell_{7,k-3,j} \text{ for } j = 1 \text{ to } 3, \]
and define

$$w_{7,k-2} = z_{7,k-2} + \sum_{j=4}^{6} \tilde{\ell}_{7,k-3,j} z_{j,k+1-j} + \tilde{\ell}_{7,k-4,6} z_{6,k-4} + \tilde{\ell}_{7,k-4,5} z_{5,k-3} + \tilde{\ell}_{7,k-5,6} z_{6,k-3} + I_1 \sum_{j=1}^{3} \beta_{7,k-2,j} \tilde{z}_{j,k+1-j} + I_2 \beta_{7,k-1,1} \tilde{z}_{1,k}.$$ 

**Step k - 2:** We obtain next,

$$\tilde{w}_{7,k-2} = z_{7,k-1} + \sum_{j=1}^{6} \ell_{7,k-2,j} \tilde{z}_{j,k+1-j} + \tilde{\ell}_{7,k-3,4} \left[ z_{4,k-2} + \sum_{j=1}^{3} \ell_{4,k-3,j} \tilde{z}_{j,k+1-j} \right] + \tilde{\ell}_{7,k-3,5} \left[ z_{5,k-3} + \sum_{j=1}^{4} \ell_{5,k-4,j} \tilde{z}_{j,k+1-j} \right] + \tilde{\ell}_{7,k-3,6} \left[ z_{6,k-4} + \sum_{j=1}^{5} \ell_{6,k-5,j} \tilde{z}_{j,k+1-j} \right] + \tilde{\ell}_{7,k-4,6} \left[ z_{6,k-3} + \sum_{j=1}^{5} \ell_{6,k-4,j} \tilde{z}_{j,k+1-j} \right] + \tilde{\ell}_{7,k-4,5} \left[ z_{5,k-2} + \sum_{j=1}^{5} \ell_{5,k-3,j} \tilde{z}_{j,k+1-j} \right] + \tilde{\ell}_{7,k-5,6} \left[ z_{6,k-2} + \sum_{j=1}^{5} \ell_{6,k-3,j} \tilde{z}_{j,k+1-j} \right] + \sum_{j=1}^{3} \beta_{7,k-2,j} \tilde{z}_{j,k+1-j} + I_1 \beta_{7,k-1,1} \tilde{z}_{1,k}.$$ 

We re-write and re-order all the terms in \( \tilde{w}_{7,k-2} \) enabling us to identify what can be canceled and what cannot be,

$$\tilde{w}_{7,k-2} = z_{7,k-1} + \sum_{j=1}^{2} \ell_{7,k-2,j} \tilde{z}_{j,k+1-j} + \sum_{j=3}^{6} \ell_{7,k-2,j} \tilde{z}_{j,k+1-j} + \tilde{\ell}_{7,k-3,4} \left[ z_{4,k-2} + \sum_{j=1}^{2} \ell_{4,k-3,j} \tilde{z}_{j,k+1-j} + \ell_{4,k-3,3} z_{3,k-2} \right] + \tilde{\ell}_{7,k-3,5} \left[ z_{5,k-3} + \sum_{j=1}^{2} \ell_{5,k-4,j} \tilde{z}_{j,k+1-j} + \sum_{j=3}^{4} \ell_{5,k-4,j} z_{j,k+1-j} \right] + \tilde{\ell}_{7,k-3,6} \left[ z_{6,k-4} + \sum_{j=1}^{2} \ell_{6,k-5,j} \tilde{z}_{j,k+1-j} + \sum_{j=3}^{5} \ell_{6,k-5,j} z_{j,k+1-j} \right] + \tilde{\ell}_{7,k-4,6} \left[ z_{6,k-3} + \sum_{j=1}^{2} \ell_{6,k-4,j} \tilde{z}_{j,k+1-j} + \sum_{j=3}^{5} \ell_{6,k-4,j} z_{j,k+1-j} \right] + \tilde{\ell}_{7,k-4,5} \left[ z_{5,k-2} + \sum_{j=1}^{2} \ell_{5,k-3,j} \tilde{z}_{j,k+1-j} + \sum_{j=3}^{5} \ell_{5,k-3,j} z_{j,k+1-j} \right] + \tilde{\ell}_{7,k-5,6} \left[ z_{6,k-2} + \sum_{j=1}^{2} \ell_{6,k-3,j} \tilde{z}_{j,k+1-j} + \sum_{j=3}^{5} \ell_{6,k-3,j} z_{j,k+1-j} \right] + \sum_{j=1}^{3} \beta_{7,k-2,j} \tilde{z}_{j,k+1-j} + I_1 \beta_{7,k-1,1} \tilde{z}_{1,k}.$$
We can re-order the terms as

\[
\psi_{7,k-2} = z_{7,k-1} + \sum_{j=1}^{2} \tilde{\ell}_{7,k-2,j} z_{j,k+1-j} + \tilde{\ell}_{7,k-3,4} \sum_{j=1}^{2} \ell_{4,k-3,j} z_{j,k+1-j} \\
+ \tilde{\ell}_{7,k-3,5} \sum_{j=1}^{2} \ell_{5,k-4,j} z_{j,k+1-j} + \tilde{\ell}_{7,k-3,6} \sum_{j=1}^{2} \ell_{6,k-5,j} z_{j,k+1-j} \\
+ \tilde{\ell}_{7,k-4,6} \sum_{j=1}^{2} \ell_{6,k-4,j} z_{j,k+1-j} + \tilde{\ell}_{7,k-4,5} \sum_{j=1}^{2} \ell_{5,k-3,j} z_{j,k+1-j} \\
+ \tilde{\ell}_{7,k-5,6} \sum_{j=1}^{2} \ell_{6,k-3,j} z_{j,k+1-j} + \sum_{j=1}^{3} \beta_{7,k-2,j} z_{j,k+1-j} \\
+ \sum_{j=3}^{6} \tilde{\ell}_{7,k-2,j} z_{j,k+1-j} + \tilde{\ell}_{7,k-3,4} \ell_{4,k-3,3} z_{3,k-2} \\
+ \tilde{\ell}_{7,k-3,5} \sum_{j=3}^{4} \ell_{5,k-4,j} z_{j,k+1-j} + \tilde{\ell}_{7,k-3,6} \sum_{j=3}^{5} \ell_{6,k-5,j} z_{j,k+1-j} \\
+ \tilde{\ell}_{7,k-4,6} \sum_{j=3}^{5} \ell_{6,k-4,j} z_{j,k+1-j} + \tilde{\ell}_{7,k-4,5} \sum_{j=3}^{4} \ell_{5,k-3,j} z_{j,k+1-j} \\
+ \tilde{\ell}_{7,k-5,6} \sum_{j=3}^{5} \ell_{6,k-3,j} z_{j,k+1-j} \\
+ \tilde{\ell}_{7,k-3,4} z_{4,k-2} + \tilde{\ell}_{7,k-3,5} z_{5,k-3} + \tilde{\ell}_{7,k-3,6} z_{6,k-4} \\
+ \tilde{\ell}_{7,k-4,6} z_{6,k-3} + \tilde{\ell}_{7,k-4,5} z_{5,k-2} + \tilde{\ell}_{7,k-5,6} z_{6,k-2} \\
+ \mathcal{I} \beta_{7,k-1,1} z_{1,k}.
\]

Let us define

\[
\tilde{\ell}_{7,k-2,j} = \ell_{7,k-2,j} + \tilde{\ell}_{7,k-3,4} \ell_{4,k-3,j} + \tilde{\ell}_{7,k-3,5} \ell_{5,k-4,j} \\
+ \ell_{7,k-3,6} \ell_{6,k-5,j} + \tilde{\ell}_{7,k-4,6} \ell_{6,k-4,j} \\
+ \ell_{7,k-4,5} \ell_{5,k-3,j} + \tilde{\ell}_{7,k-5,6} \ell_{6,k-3,j}
\]

(4.10)

for \(j = 1\) and \(2\), and

\[
\tilde{\ell}_{7,k-3,2,3} = \ell_{7,k-2,3} + \ell_{7,k-3,6} \ell_{6,k-3,3} + \ell_{7,k-4,5} \ell_{5,k-3,3} + \ell_{7,k-4,6} \ell_{6,k-4,3} \\
+ \ell_{7,k-3,4} \ell_{4,k-3,3} + \ell_{7,k-3,5} \ell_{5,k-4,3} + \ell_{7,k-3,6} \ell_{6,k-5,3} \\
\tilde{\ell}_{7,k-3,2,4} = \ell_{7,k-2,4} + \ell_{7,k-3,6} \ell_{6,k-3,4} + \ell_{7,k-4,5} \ell_{5,k-3,4} \\
+ \ell_{7,k-4,6} \ell_{6,k-4,4} + \ell_{7,k-3,5} \ell_{5,k-4,4} + \ell_{7,k-3,6} \ell_{6,k-5,4} \\
\tilde{\ell}_{7,k-3,2,5} = \ell_{7,k-2,5} + \ell_{7,k-3,6} \ell_{6,k-3,5} + \ell_{7,k-4,6} \ell_{6,k-4,5} \\
+ \ell_{7,k-3,6} \ell_{6,k-5,5}
\]

(4.11)
Also, select
\[ \beta_{7,k-2,j} = -\tilde{\ell}_{7,k-2,j} \]
for \( j = 1 \) and \( 2 \), and define
\[
w_{7,k-1} = z_{7,k-1} + \sum_{j=3}^{6} \tilde{\ell}_{7,k-2,j} z_{j,k+1-j} \\
+ \tilde{\ell}_{7,k-3,3} z_{4,k-2} + \tilde{\ell}_{7,k-3,5} z_{5,k-3} + \tilde{\ell}_{7,k-3,6} z_{6,k-4} \\
+ \tilde{\ell}_{7,k-4,6} z_{5,k-3} + \tilde{\ell}_{7,k-4,5} z_{5,k-2} + \tilde{\ell}_{7,k-5,6} z_{6,k-2} \\
+ L_{1} \beta_{7,k-1,1} z_{1,k}.\]

**Step k - 1:** We obtain next,
\[
w_{7,k-1} = z_{7,k} + \sum_{j=1}^{6} \tilde{\ell}_{7,k-1,j} z_{j,k+1-j} \\
+ \tilde{\ell}_{7,k-2,3} \left[ z_{3,k-1} + \sum_{j=1}^{2} \tilde{\ell}_{3,k-2,j} z_{j,k+1-j} \right] \\
+ \tilde{\ell}_{7,k-2,4} \left[ z_{4,k-2} + \sum_{j=1}^{3} \tilde{\ell}_{4,k-3,j} z_{j,k+1-j} \right] \\
+ \tilde{\ell}_{7,k-2,5} \left[ z_{5,k-3} + \sum_{j=1}^{4} \tilde{\ell}_{5,k-4,j} z_{j,k+1-j} \right] \\
+ \tilde{\ell}_{7,k-2,6} \left[ z_{6,k-4} + \sum_{j=1}^{5} \tilde{\ell}_{6,k-5,j} z_{j,k+1-j} \right] \\
+ \tilde{\ell}_{7,k-3,4} \left[ z_{4,k-1} + \sum_{j=1}^{2} \tilde{\ell}_{4,k-2,j} z_{j,k+1-j} \right] \\
+ \tilde{\ell}_{7,k-3,5} \left[ z_{5,k-2} + \sum_{j=1}^{3} \tilde{\ell}_{5,k-3,j} z_{j,k+1-j} \right] \\
+ \tilde{\ell}_{7,k-3,6} \left[ z_{6,k-3} + \sum_{j=1}^{4} \tilde{\ell}_{6,k-4,j} z_{j,k+1-j} \right] \\
+ \tilde{\ell}_{7,k-4,6} \left[ z_{6,k-2} + \sum_{j=1}^{5} \tilde{\ell}_{6,k-3,j} z_{j,k+1-j} \right] \\
+ \tilde{\ell}_{7,k-4,5} \left[ z_{5,k-1} + \sum_{j=1}^{4} \tilde{\ell}_{5,k-2,j} z_{j,k+1-j} \right] \\
+ \tilde{\ell}_{7,k-5,6} \left[ z_{6,k-1} + \sum_{j=1}^{5} \tilde{\ell}_{6,k-2,j} z_{j,k+1-j} \right] \\
+ \beta_{7,k-1,1} z_{1,k}.\]

We define,
\[
\tilde{\ell}_{7,k-1,j} = \ell_{7,k-1,j} + \tilde{\ell}_{7,k-2,3} \ell_{3,k-2,j} + \tilde{\ell}_{7,k-2,4} \ell_{4,k-3,j} \\
+ \tilde{\ell}_{7,k-2,5} \ell_{5,k-4,j} + \tilde{\ell}_{7,k-2,6} \ell_{6,k-5,j} \\
+ \tilde{\ell}_{7,k-3,4} \ell_{4,k-2,j} + \tilde{\ell}_{7,k-3,5} \ell_{5,k-3,j} \\
+ \tilde{\ell}_{7,k-3,6} \ell_{6,k-4,j} + \tilde{\ell}_{7,k-4,6} \ell_{6,k-3,j} \\
+ \tilde{\ell}_{7,k-4,5} \ell_{5,k-2,j} + \tilde{\ell}_{7,k-5,6} \ell_{6,k-2,j},\]
for \( j = 1 \) and \( 2 \), and
\[
\tilde{\ell}_{7,k-1,3} = \ell_{7,k-1,3} + \tilde{\ell}_{7,k-2,4} \ell_{4,k-3,3} \\
+ \tilde{\ell}_{7,k-2,5} \ell_{5,k-4,3} + \tilde{\ell}_{7,k-2,6} \ell_{6,k-5,3} \\
+ \tilde{\ell}_{7,k-3,4} \ell_{4,k-2,3} + \tilde{\ell}_{7,k-3,5} \ell_{5,k-3,3} \\
+ \tilde{\ell}_{7,k-3,6} \ell_{6,k-4,3} + \tilde{\ell}_{7,k-4,6} \ell_{6,k-3,3} \\
+ \tilde{\ell}_{7,k-4,5} \ell_{5,k-2,3} + \tilde{\ell}_{7,k-5,6} \ell_{6,k-2,3},\]
for \( j = 1 \) and \( 2 \).
\[ \ddot{\ell}_{7,k-1} = \ell_{7,k-1} + \ddot{\ell}_{7,k} - 2,5 \ell_{5,k-4,4} + \ddot{\ell}_{7,k-2,6} \ell_{6,k-3,5} + \ddot{\ell}_{7,k-3,6} \ell_{6,k-4,4} + \ddot{\ell}_{7,k-4,6} \ell_{6,k-3,4} + \ddot{\ell}_{7,k-5,6} \ell_{6,k-2,4}, \]  

(4.14)

\[ \ddot{\ell}_{7,k-1} = \ell_{7,k-1} + \ddot{\ell}_{7,k-2,6} \ell_{6,k-5,5} + \ddot{\ell}_{7,k-3,6} \ell_{6,k-4,5} + \ddot{\ell}_{7,k-4,6} \ell_{6,k-3,5} + \ddot{\ell}_{7,k-5,6} \ell_{6,k-2,5}, \]  

(4.15)

and

\[ \ddot{\ell}_{7,k-1,6} = \ell_{7,k-1,6}. \]  

(4.16)

With the above definitions, we can re-write \( w_{7,k-1} \) as

\[ w_{7,k-1} = z_{7,k} + \ddot{\ell}_{7,k-1} - 1,1 z_{1,k} + \beta_{7,k-1,1} \beta_{7,k-1,1} z_{1,k} \]

\[ + \sum_{j=2}^{6} \ddot{\ell}_{7,k-1,j} j \sum_{k+j} \ell_{j,k-1,j} z_{j,k+1-j} \]

\[ + \ddot{\ell}_{7,k-2,3} z_{3,k-1} + \ddot{\ell}_{7,k-2,4} z_{4,k-2} \]

\[ + \ddot{\ell}_{7,k-2,5} z_{5,k-3} + \ddot{\ell}_{7,k-2,6} z_{6,k-4} \]

\[ + \ddot{\ell}_{7,k-3,4} z_{4,k-1} + \ddot{\ell}_{7,k-3,5} z_{5,k-2} \]

\[ + \ddot{\ell}_{7,k-3,6} z_{6,k-3} + \ddot{\ell}_{7,k-4,6} z_{6,k-2} \]

\[ + \ddot{\ell}_{7,k-4,5} z_{5,k-1} + \ddot{\ell}_{7,k-5,6} z_{6,k-1}. \]

We choose

\[ \beta_{7,k-1,1} = -\ddot{\ell}_{7,k-1,1}. \]

We can now define

\[ w_{7,k} = z_{7,k} + \sum_{j=2}^{6} \ddot{\ell}_{7,k-1,j} j \sum_{k+j} \ell_{j,k-1,j} z_{j,k+1-j} \]

\[ + \ddot{\ell}_{7,k-2,3} z_{3,k-1} + \ddot{\ell}_{7,k-2,4} z_{4,k-2} \]

\[ + \ddot{\ell}_{7,k-2,5} z_{5,k-3} + \ddot{\ell}_{7,k-2,6} z_{6,k-4} \]

\[ + \ddot{\ell}_{7,k-3,4} z_{4,k-1} + \ddot{\ell}_{7,k-3,5} z_{5,k-2} \]

\[ + \ddot{\ell}_{7,k-3,6} z_{6,k-3} + \ddot{\ell}_{7,k-4,6} z_{6,k-2} \]

\[ + \ddot{\ell}_{7,k-4,5} z_{5,k-1} + \ddot{\ell}_{7,k-5,6} z_{6,k-1}. \]

51
Step \( k \): We obtain next,

\[
\dot{w}_{7,k} = g_k(z) + u_k \\
+ \sum_{j=2}^{4} \ell_{7,k-1,j} \left[ z_{j,k+2-j} + \sum_{p=1}^{3} \ell_{7,k+1-j,p} \hat{z}_{p,k-p-1} \right] \\
+ \ell_{7,k-2,3} \left[ z_{3,k} + \sum_{p=1}^{4} \ell_{3,k-1,p} \hat{z}_{p,k-p-1} \right] \\
+ \ell_{7,k-2,4} \left[ z_{4,k-1} + \sum_{p=1}^{4} \ell_{4,k-2,p} \hat{z}_{p,k-p-1} \right] \\
+ \ell_{7,k-2,5} \left[ z_{5,k-2} + \sum_{p=1}^{4} \ell_{5,k-3,p} \hat{z}_{p,k-p-1} \right] \\
+ \ell_{7,k-2,6} \left[ z_{6,k-3} + \sum_{p=1}^{4} \ell_{6,k-4,p} \hat{z}_{p,k-p-1} \right] \\
+ \ell_{7,k-3,4} \left[ z_{4,k} + \sum_{p=1}^{4} \ell_{4,k-1,p} \hat{z}_{p,k-p-1} \right] \\
+ \ell_{7,k-3,5} \left[ z_{5,k-1} + \sum_{p=1}^{4} \ell_{5,k-2,p} \hat{z}_{p,k-p-1} \right] \\
+ \ell_{7,k-3,6} \left[ z_{6,k-2} + \sum_{p=1}^{4} \ell_{6,k-3,p} \hat{z}_{p,k-p-1} \right] \\
+ \ell_{7,k-4,6} \left[ z_{6,k-1} + \sum_{p=1}^{4} \ell_{6,k-2,p} \hat{z}_{p,k-p-1} \right] \\
+ \ell_{7,k-4,5} \left[ z_{5,k} + \sum_{p=1}^{4} \ell_{5,k-1,p} \hat{z}_{p,k-p-1} \right] \\
+ \ell_{7,k-5,6} \left[ z_{6,k} + \sum_{p=1}^{4} \ell_{6,k-1,p} \hat{z}_{p,k-p-1} \right]
\]

We conclude now the process of transforming \( \Sigma_{z7} \) to a uniform rank system \( \Sigma_{w7} \) in SCB format by simply saying that the above algorithmic procedure gives us,

\[
\Sigma_{w7} : \begin{cases} \\
\dot{w}_{7,j-1} = w_{7,j+1}, \text{ for } j = 1, 2, \ldots, k-1, \\
\dot{w}_{7,k} = \hat{g}_7(z) + u_7,
\end{cases}
\]

for some linear function \( \hat{g}_7(z) \). The above system is in the format of a uniform rank system in SCB format with \( y_{w7} := w_{7,1} \) as its output.

### 4.3.6 Transformation of \( \Sigma_{zm} \) to a uniform rank system \( \Sigma_{wm} \)

So far, we have successfully transformed \( \Sigma_{zm} \), for \( m = 1 \) to 4 and for \( m = 7 \), to their respective uniform rank systems \( \Sigma_{wm} \) in SCB format. In doing so, we gained enough experience that guides us in transforming \( \Sigma_{zm} \) to a \( \Sigma_{wm} \) for a general index \( m \). As seen in previous subsections, the elimination or cancelation of feedback terms is undertaken sequentially one step at a time. One of the typical observations we emphasized earlier is the presence of uncanceled feedback terms in a given step. Such uncanceled terms are passed on to the definition of a substate variable that follows next in the chain of steps that ensue. In any given step, the uncanceled feedback terms of the previous steps, when they are differentiated, generate their own feedback terms besides the primary feedback terms that are naturally present in \( \Sigma_{zm} \). Such feedback terms can be called secondary feedback terms. In every step, irrespective of whether they are primary or secondary feedback terms, some of them are canceled and others are passed on to the definition of a substate variable that follows next. Eventually, all uncanceled feedback terms accumulate in the last step and are absorbed into the linear function \( g_m(w) \) which is additive to the control variable...
of that step. An important aspect to watch for is that no uncanceled terms of the type \( \dot{w}_{m,k} \) or further differentiations of such terms are passed on to \( g_m(w) \) because such terms are prohibited in \( g_m(w) \), which is just a linear function of \( w \).

We now proceed to transform \( \Sigma_{zm} \) for \( 3 < m \leq k \) to a uniform rank system \( \Sigma_{wm} \) in SCB format. Based on the experience gained so far, we select,

\[
y_{wm} := w_{m,1} := z_{m,1} + \sum_{n=1}^{m-1} \mathcal{I}_{k+n+m} \sum_{j=1}^{m-n} \beta_{m,k+n-m,j} z_{j,k+1-j}
\]

\[
= z_{m,1} + \mathcal{I}_{k+1-m} \sum_{j=1}^{m-1} \beta_{m,k+1-m,j} z_{j,k+1-j} 
+ \mathcal{I}_{k+2-m} \sum_{j=1}^{m-2} \beta_{m,k+2-m,j} z_{j,k+1-j} 
+ \cdots 
+ \mathcal{I}_{k+m} \beta_{m,k,m-j} z_{j,k+1-j} 
+ \mathcal{I}_{k+1} \beta_{m,k-1,j} z_{j,k+1-j} 
+ \mathcal{I}_{k} \beta_{m,k-2,j} z_{j,k+1-j} 
+ \cdots 
+ \mathcal{I}_{k-1} \beta_{m,k-1} z_{1,k}.
\]

To understand the above selection of \( y_{wm} \), we first need to make it clear that for each \( j = 1 \) to \( k \), the substate \( z_{j,k+1-j} \) is the output of \( \Sigma_j \) of the given system \( \Sigma \) when the output of \( \Sigma_j \) is written in terms of \( z \) variables. In view of this, the above selection of \( y_{wm} \) of \( \Sigma_{wm} \) requires exactly the same number of minimum integral operators as required by \( \Sigma_{zi} \) in order to obtain them from the given subsystems \( \Sigma_i, i = 1 \) to \( m \). In fact, \( y_{wm} \) is simply a linear combination of the substates of \( \Sigma_{zi}, i = 1 \) to \( m-1 \) and \( z_{m,1} \). This means that the post-compensator developed here utilizes absolutely the minimum number of integrators that are necessary.

Here, as before, various coefficients \( \beta_{m,i,j} \) are yet unknown. Let us recall the notation of \( \beta_{m,i,j} \). The first index \( m \) in it simply refers to the subsystem we are considering here. The second index \( i \) and the third index \( j \) have a deeper meaning. The variable to which \( \beta_{m,i,j} \) is a coefficient is obtained after \( i \) integrations of the output of subsystem \( \Sigma_j \) of the given system \( \Sigma \). These unknown coefficients \( \beta_{m,i,j} \) are chosen in \( i \)-th step as we proceed to remove the existing feedback terms in \( \Sigma_{zm} \) as well as secondary feedback terms step by step. We list below the algorithmic process of selecting these coefficients as well as the substates \( w_{m,j}, j = 1 \) to \( k \). We emphasize that the process of choosing the coefficients \( \beta_{m,i,j} \) for a general index \( m \) conceptually mimics that in the previous subsection for \( m = 7 \).

To start with, we first outline the algorithmic process of selecting the unknown coefficients. We recognize that all \( \beta \) coefficients are associated with the outputs of subsystems and they are selected step by step to cancel out the feedback terms from such outputs. We forewarn that we cannot select \( \beta \) coefficients to cancel any feedback other than the feedback from such outputs.

**Step 1:** We define

\[
D_1 w_{m,1} := w_{m,2}.
\]
Step 2: We define
\[ D_2 w_{m,1} := w_{m,3}. \]

\[ \vdots \]

Step \( k - m \): We define
\[ D_{k-m} w_{m,1} := w_{m,k+1-m}. \]

Step \( k - (m - 1) \): After \( k - (m - 1) \) differentiations of the output \( y_{wm} \), we reach \( w_{m,k-(m-1)} \). Then, the coefficients \( \beta_{m,k-(m-1),j} \) for \( j = 1 \) to \( m - 1 \) are selected to cancel out all the feedback terms present in \( w_{m,k-(m-1)} \). Also, \( w_{m,k-(m-2)} \) is chosen such that \( w_{m,k-(m-1)} = w_{m,k-(m-2)} \).

Step \( k - (m - 2) \): After \( k - (m - 2) \) differentiations of the output \( y_{wm} \), we reach \( w_{m,k-(m-2)} \). Then, the coefficients \( \beta_{m,k-(m-2),j} \) for \( j = 1 \) to \( m - 2 \) are selected to cancel out the feedback terms from all subsystems except from the subsystem \( \Sigma_{m-1} \). Note that because of the way \( y_{wm} \) is structured, there exists no coefficient \( \beta_{m,k-(m-2),m-1} \) present in \( y_{wm} \) to cancel out the feedback term from the output of \( \Sigma_{m-1} \). Such an uncanceled feedback term is \( \ell_{m,k-(m-2),m-1} z_{m-1,k-(m-2)} \), and it needs to be passed on as a component in the next substate variable \( w_{m,k-(m-3)} \) which is chosen such that \( w_{m,k-(m-2)} = w_{m,k-(m-3)} \). We can foresee the effect of passing on \( \ell_{m,k-(m-2),m-1} z_{m-1,k-(m-2)} \) to \( w_{m,k-(m-3)} \). Obviously, the term gets differentiated in the next step, and generates some other feedback terms which can be called secondary feedback terms. Some such secondary feedback terms need to be canceled in the next step and others need to be passed on to subsequent steps. It is worth examining what type of variables are passed on to subsequent steps. We will describe in detail this examination shortly when we present the details of selecting the \( \beta \) coefficients.

\[ \vdots \]

Step \( k - (m - i) \): After \( k - (m - i) \) differentiations of the output \( y_{wm} \), we reach \( w_{m,k-(m-i)} \). Then, the coefficients \( \beta_{m,k-(m-i),j} \) for \( j = 1 \) to \( m - i \) are selected to cancel out the feedback terms from all subsystems except from the subsystems \( \Sigma_n \) for \( m - i < n \leq m - 1 \). Note that because of the way \( y_{wm} \) is structured, there exists no coefficients \( \beta_{m,k-(m-i),n} \) for \( m - i < n \leq m - 1 \) present in \( y_{wm} \) to cancel out the feedback terms from the outputs of such subsystems \( \Sigma_n \). Such uncanceled feedback terms need to be passed on as components in the next substate variable \( w_{m,k-(m-i-1)} \) which is chosen such that \( w_{m,k-(m-i)} = w_{m,k-(m-i-1)} \).

\[ \vdots \]

Step \( k - 2 \): After \( k-2 \) differentiations of the output \( y_{wm} \), coefficients \( \beta_{m,k-2,j} \) for \( j = 1 \) and \( j = 2 \) are selected to cancel out the feedback terms from the
subsystems $\Sigma_1$ and $\Sigma_2$. The feedback terms from the outputs of all subsystems $\Sigma_n$ for $n > 2$ are not canceled and are passed on as components of $w_{m,k-1}$ which is chosen such that $\dot{w}_{m,k-2} = w_{m,k-1}$.

**Step $k-1$:** After $k-1$ differentiations of the output $y_{wm}$, the coefficient $\beta_{m,k-1,1}$ is selected to cancel out the feedback term from the subsystem $\Sigma_1$. The feedback terms from the outputs of all subsystems $\Sigma_n$ for $n > 1$ are not canceled and are passed on as components of $w_{m,k}$ which is chosen such that $\dot{w}_{m,k-1} = w_{m,k}$.

**Step $k$:** Finally, after $k$ differentiations of the output, we reach $\dot{w}_{m,k}$. Whatever terms were passed on as components of $w_{m,k}$ are such that $\dot{w}_{m,k}$ consists of only linear combinations of all substates of $w$ besides the control input $u_m$. That is,

$$\dot{w}_{m,k} = g_m(w) + u_m.$$

We proceed now to describe the details of the algorithmic process of determining the coefficients $\beta_{m,i,j}$ in $y_{wm} = w_{m,1}$. The first $k - m$ steps are exactly the same as we described above. There exists $m$ more steps before we reach $\dot{w}_{m,k}$, and we describe next the details of these steps.

**Step $k - (m - 1)$:** When $y_{wm} = w_{m,1}$ is differentiated $k - (m - 1) = k + 1 - m$ times, we get

$$\dot{w}_{m,k+1-m} = z_{m,k+2-m} + \sum_{j=1}^{m-1} \ell_{m,k+1-m,j} \dot{z}_{j,k+1-j}$$

$$\quad + \sum_{j=1}^{m-1} \beta_{m,k+1-m,j} \dot{z}_{j,k+1-j}$$

$$\quad + \sum_{n=1}^{m-2} \sum_{j=1}^{m-n-1} \beta_{m+k+1-m,j} \dot{z}_{j,k+1-j}.$$ 

Examination of the above expression enables us to decide,

$$\beta_{m,k+1-m,j} = -\ell_{m,k+1-m,j}, \text{ for } j = 1 \text{ to } m - 1,$$

in order to cancel the feedback represented by the spade ♦. We choose next,

$$w_{m,k+2-m} := z_{m,k+2-m} + \sum_{n=1}^{m-2} \sum_{j=1}^{m-n-1} \beta_{m,k+n+1-m,j} \dot{z}_{j,k+1-j}$$

$$\quad := z_{m,k+2-m} + \sum_{n=1}^{m-2} \sum_{j=1}^{m-n} \beta_{m+k+2-m.j} \dot{z}_{j,k+1-j}$$

$$\quad + \sum_{n=1}^{m-3} \sum_{j=1}^{m-n-1} \beta_{m+k+n+2-m,j} \dot{z}_{j,k+1-j}.$$ 

The above implies that

$$\dot{w}_{m,k+1-m} = w_{m,k+2-m}.$$ 

**Step $k - (m - 2)$:** We differentiate once $w_{m,k+2-m}$, and obtain

$$\dot{w}_{m,k+2-m} := z_{m,k+3-m} + \sum_{j=1}^{m-2} \ell_{m,k+2-m,j} \dot{z}_{j,k+1-j} + \ell_{m,k+2-m,m-1} \dot{z}_{m-1,k+2-m}$$

$$\quad + \sum_{j=1}^{m-2} \beta_{m,k+2-m,j} \dot{z}_{j,k+1-j}$$

$$\quad + \sum_{n=1}^{m-3} \sum_{j=1}^{m-n} \beta_{m+k+n+2-m,j} \dot{z}_{j,k+1-j}.$$
We see clearly that we can choose

\[ \beta_{m,k+2-m,j} = -\ell_{m,k+2-m,j}, \quad \text{for } j = 1 \text{ to } m - 2, \]

in order to cancel the feedback term represented by the spade ♠. The above selection leaves uncanceled the term represented by the diamond ♦. This uncanceled term is the feedback from the output of subsystem \( \Sigma_{m-1} \) of the given system \( \Sigma \) and it could not be canceled because there is no \( \beta \) coefficient associated with such an output. Consequently, it needs to be passed on to become a component of \( w_{m,k+3-m} \) defined as,

\[
 w_{m,k+3-m} := z_{m,k+3-m} + \ell_{m,k+2-m,m-1} z_{m-1,k+2-m} \\
+ \sum_{n=1}^{m-3} \sum_{j=1}^{m-n-2} \beta_{m,k+n+2-m,j} z_{j,k+1-j} \\
+ I_1 \sum_{j=1}^{m-3} \beta_{m,k+3-m,j} z_{j,k+1-j} \\
+ \sum_{n=1}^{m-4} \sum_{j=1}^{m-n-3} \beta_{m,k+n+3-m,j} z_{j,k+1-j}.
\]

The above implies that

\[
 \dot{w}_{m,k+2-m} = w_{m,k+3-m}.
\]

As said above, the term represented by the diamond ♦ could not be canceled in this step \( k - (m - 2) \) and needs to be passed on to the next step \( k - (m - 3) \). We will see shortly the consequence of passing on such a term to the next step.

**Step \( k - (m - 3) \):** We differentiate once \( w_{m,k+3-m} \) to obtain,

\[
 \dot{w}_{m,k+3-m} := z_{m,k+4-m} + \sum_{j=1}^{m-3} \ell_{m,k+3-m,j} z_{j,k+1-j} \\
+ \ell_{m,k+2-m,m-1} z_{m-1,k+2-m} \\
+ \sum_{j=1}^{m-3} \beta_{m,k+3-m,j} z_{j,k+1-j} \\
+ \sum_{n=1}^{m-4} \sum_{j=1}^{m-n-3} \beta_{m,k+n+3-m,j} z_{j,k+1-j},
\]

\[
:= z_{m,k+4-m} + \sum_{j=1}^{m-3} \ell_{m,k+3-m,j} z_{j,k+1-j} \quad \text{(Primary feedback terms)}
\]

\[
+ \ell_{m,k+2-m,m-1} z_{m-1,k+3-m} + \sum_{j=1}^{m-2} \ell_{m-1,k+2-m,j} z_{j,k+1-j} \quad \text{(Secondary feedback terms)}
\]

\[
+ \sum_{j=1}^{m-3} \beta_{m,k+3-m,j} z_{j,k+1-j} \\
+ \sum_{n=1}^{m-4} \sum_{j=1}^{m-n-3} \beta_{m,k+n+3-m,j} z_{j,k+1-j}.
\]

Before proceeding further, we need to rearrange the terms in the above expres-
sion for easy recognition of the terms that can be canceled,

\[ w_{m,k+3-m} := z_{m,k+4-m} + \sum_{j=1}^{m-3} [\ell_{m,k+3-m,j} z_{j,k+1-j} \]

\[ + \sum_{j=1}^{m-1} \beta_{m,k+3-m,j} z_{j,k+1-j} \]  

\[ + \sum_{j=m-2}^{m-1} \ell_{m,k+3-m,j} z_{j,k+1-j} \]  

\[ + \sum_{j=m-2}^{m-1} z_{m-1,k+3-m} \]  

\[ + \sum_{j=m-1}^{m-2} \epsilon_{m,k+2-m,m-1} z_{m-2,k+3-m} \]  

(4.17)  

The terms represented by the spade ♠ can be canceled by the terms in the expression (4.17), provided the coefficients \( \beta_{m,k+3-m,j} \) for \( j = 1 \) to \( m-3 \) are chosen appropriately; that is, by the choice of

\[ \beta_{m,k+3-m,j} = -\epsilon_{m,k+3-m,j} - \sum_{j=m-2}^{m-1} \beta_{m,k+3-m,j} z_{j,k+1-j} \]  

(4.18)  

The terms in (4.18) are those left over among the primary feedback terms as they cannot be canceled. These are the feedback terms from the outputs of subsystems \( \Sigma_{m-1} \) and \( \Sigma_{m-2} \). The terms in (4.20) are those left over among the secondary feedback terms as they cannot be canceled either. The marking of club ♣ in the expression (4.19), and the markings of diamonds ♦ in the expressions (4.18) and (4.20) are consistent with what we used earlier, that is clubs can never be canceled because there are not associated with outputs of any system, while the diamonds could not be canceled although they are associated with outputs of some system or other.

We define,

\[ w_{m,k+4-m} := z_{m,k+4-m} + \sum_{j=m-2}^{m-1} \ell_{m,k+3-m,j} z_{j,k+1-j} \]

\[ + \sum_{j=m-1}^{m-2} \ell_{m,k+2-m,m-1} z_{m-1,k+3-m} + \sum_{j=1}^{m-4} \beta_{m,k+4-m,j} z_{j,k+1-j} \]

\[ + \sum_{j=m-1}^{m-4} \epsilon_{m,k+2-m,m-1} z_{m-2,k+3-m} \]

where

\[ \epsilon_{m,k+2-m,m-1} = \ell_{m,k+2-m,m-1} - \ell_{m-1,k+2-m,m-2} \]

This leads to

\[ \dot{w}_{m,k+3-m} := w_{m,k+4-m} \]

At this time, it is imperative that we examine carefully the consequences of passing on an uncanceled output term represented by the diamond ♦ from
the previous Step \( k - (m - 2) \) to this Step \( k - (m - 3) \). We observe that it got differentiated and in that process generated several terms which we called secondary feedback terms. Some of these secondary terms got canceled in this step and some others did not, and thus need to be passed on to the next step. It is indeed incumbent on us to learn in depth the nature of all the secondary terms. We see that:

\[
\hat{z}_{m-1,k+2-m} = z_{m-1,k+3-m} + \sum_{j=1}^{m-2} \ell_{m-1,k+2-m,j} \hat{z}_{j,k+1-j} \quad \text{(Secondary feedback terms)}
\]

\[
= \sum_{j=1}^{m-3} \ell_{m-1,k+2-m,j} \hat{z}_{j,k+1-j} + \hat{z}_{m-1,k+3-m} + \ell_{m-1,k+2-m,m-2} \hat{z}_{m-2,k+3-m}.
\]

The term or terms represented by a spade \( \clubsuit \) are outputs of previous subsystems and got canceled in this step. The term represented by a club \( \clubsuit \) is not an output of any subsystem and can never be beaten by any club to have it canceled by appropriately selecting \( \beta \) coefficients, since such \( \beta \) coefficients are associated only with the outputs of previous subsystems. The term represented by a diamond \( \diamondsuit \), although it is an output of a previous subsystem, it is hard to be canceled by appropriately selecting \( \beta \) coefficients because there is no longer any \( \beta \) coefficient associated with such an output.

It is worthwhile to examine the clubs \( \clubsuit \) and diamonds \( \diamondsuit \) one at a time. Let us first focus on the clubs. A club \( \clubsuit \), when differentiated once, generates its own spades \( \clubsuit \), clubs \( \clubsuit \), and diamonds \( \diamondsuit \). We are interested at this time to focus only on clubs. By differentiating the club \( z_{m-1,k+3-m} \) once, we get

\[
\hat{z}_{m-1,k+3-m} = \hat{z}_{m-1,k-(m-3)} = \hat{z}_{m-1,k-(m-4)} + \clubsuit + \diamondsuit.
\]

We observe that any club \( \clubsuit \), say \( z_{m-1,k-(m-3)} \), when differentiated once generates a club \( \clubsuit \) given by \( z_{m-1,k-(m-4)} \). Note that by differentiating once \( z_{m-1,k-(m-3)} \), the second suffix in it increases by one. In the remaining \((m-3)\) steps before reaching \( \hat{w}_{m,k} \) in \( k \)-th step, the club \( \clubsuit \) \( z_{m-1,k-(m-3)} \) gets differentiated \((m-3)\) times (once in each step), and yields a club \( z_{m-1,k} \), which is then safely absorbed in to \( g_m(w) \) that is present in \( k \)-th step that describes \( \hat{w}_{m,k} \). Following this reasoning and by examining the second subscript of the clubs generated in each step, it is straightforward to confirm that any club \( \clubsuit \) that is present in any step will eventually and safely be passed on to \( g_m(w) \) that is present in \( k \)-th step.

This brings us next to examine the terms represented by the diamonds \( \diamondsuit \). These are the feedback terms from the outputs of subsystems \( \Sigma_{m-1} \) and \( \Sigma_{m-2} \) that cannot be canceled in this step \( k - (m - 3) \). These feedback terms contain:

\[
z_{m-2,k+3-m} \quad \text{(Output of} \Sigma_{m-2} \text{)} \quad \text{and} \quad z_{m-1,k+2-m} \quad \text{(Output of} \Sigma_{m-1} \text{)}.
\]

These terms are passed on from this step \( k - (m - 3) \) to the next step \( k - (m - 4) \). In step \( k - (m - 4) \), they will be differentiated once and generate their own
spades (♠s), clubs (♥s), and diamonds (♦s). Since spades can be canceled, we need to examine only clubs and diamonds. However, we already examined the clubs. Thus, we need to discuss only about diamonds. Any of the above terms when differentiated once generates a diamond ♦ in which the second suffix is increased by one. In the remaining \((m - 3)\) steps before reaching \(w_{m,k}\) in \(k\)-th step, these diamonds (♦s) gets differentiated \((m - 3)\) times (once in each step), and result in diamonds of the form \(z_{i,k}\) and \(z_{i,k-1}\) for some \(i < m - 1\). Such resulting diamonds are safely absorbed in to \(g_m(w)\) that is present in \(k\)-th step. Following this reasoning and by examining the second subscript of the diamonds generated in each step, it is straight forward to confirm that any diamond ♦ that is present in any step will eventually and safely be passed on to \(g_m(w)\) that is present in \(k\)-th step.

Having explained the process of determining the coefficients \(\beta_{m,i,j}\), we can summarize below all the pertinent equations. To start with, all the coefficients \(\ell_{m,i,j}\) can be viewed depicted in a matrix \(L_m\), for each \(m, 2 \leq m \leq k\), as

\[
L_m = \begin{bmatrix}
\ell_{m,1} & \ell_{m,2} & \cdots & \ell_{m,m-1} \\
\ell_{m+1,1} & \ell_{m+1,2} & \cdots & \ell_{m+1,m-1} \\
\vdots & \vdots & \ddots & \vdots \\
\ell_{m,k-1} & \ell_{m,k-2} & \cdots & \ell_{m,k-1}
\end{bmatrix},
\]

where the coefficients of \(i\)-th row are those that belong to \(\tilde{z}_{m,k-m+i,j}\) for \(i = 1\) to \(k - 1\) and \(j = 1\) to \(m - 1\). In order to determine the coefficients \(\beta_{m,i,j}\), we update the above coefficients by

\[
\tilde{\ell}_{m,i,j} = \ell_{m,i,j}, \text{ for } m = 2 \text{ and } m = 3,
\]

\[
\tilde{\ell}_{m,i,j} = \ell_{m,i,j} + \sum_{p=k+1}^{i-1} \sum_{q=p+1}^{m} \tilde{\ell}_{m,p,q}\ell_{q,i-k-q+p,j} \quad (4.21)
\]

for \(4 \leq m \leq k\). In fact, only when \(k + 1 - m + 2 \leq i \leq k - 1\) and \(1 \leq j \leq m - 2\), \(\ell_{m,i,j}\) is updated, otherwise it stays unchanged. The updated coefficient matrix \(\tilde{L}_m\) can be viewed as

\[
\tilde{L}_m = \begin{bmatrix}
\tilde{\ell}_{m+1,1} & \tilde{\ell}_{m+1,2} & \cdots & \tilde{\ell}_{m+1,m-1} \\
\tilde{\ell}_{m,k} & \tilde{\ell}_{m,k+1,n} & \cdots & \tilde{\ell}_{m,k,m-1} \\
\vdots & \vdots & \ddots & \vdots \\
\tilde{\ell}_{m,k-1} & \tilde{\ell}_{m,k-2} & \cdots & \tilde{\ell}_{m,k-1}
\end{bmatrix}
\]

Let us defines the coefficient matrix \(\beta_m\) for \(2 \leq m \leq k\) as

\[
\beta_m = \begin{bmatrix}
\beta_{m,1} & \beta_{m,2} & \cdots & \beta_{m,m-1} \\
\beta_{m+1,1} & \beta_{m+1,2} & \cdots & \beta_{m+1,m-1} \\
\vdots & \vdots & \ddots & \vdots \\
\beta_{m,k-1,1} & \beta_{m,k-1,2} & \cdots & \beta_{m,k-1,m-1} \\
\end{bmatrix}.
\]

59
We then have,

\[ \beta_m = \begin{bmatrix}
-\ell_{m,k+1-m,1} & -\ell_{m,k+1-m,2} & \cdots & -\ell_{m,k+1-m,m-1} \\
-\ell_{m,k+1-m,1} & -\ell_{m,k+1-m,2} & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots \\
-\ell_{m,k-1,1} & 0 & \cdots & 0
\end{bmatrix}. \]

We observe that the elements that form the upper triangular portion including the diagonal of \( \tilde{L}_m \) define \( \beta_m \). These are the terms associated with spades ♠️s, the remaining elements of \( \tilde{L}_m \) are associated with diamonds ♦️s.

We also have for \( k+1-m \leq i < k, \)

\[ \dot{w}_{m,i} = z_{m,i+1} + \sum_{p=1}^{k-i} \ell'_{m,i,p} z_{p,k+1-p} + \sum_{p=k+1-i}^{m-1} \ell'_{m,i,p} z_{p,k+1-p} \]

\[ + \sum_{p=k+1-m+1}^{i-1} \sum_{q=k+1-p}^{m-1} \ell'_{m,p,q} z_{q,k+1+i-p} - \sum_{p=k+1-(j-1)}^{i-1} \sum_{q=k+1-p}^{j-1} \ell'_{i,k+j-(p+q),p} z_{p,q}. \]  

For \( 1 \leq j < k+1-i \), we just use \( z_{i,j} = \int z_{i,j} \) to get \( z_{i,j} \) (for \( 1 \leq j < k+1-i \)). Because we know that \( z_{1,j} = w_{1,j} \), so by using equation (4.22) it is easy to find all \( z_{i,j} \) in the form

\[ z_{i,j} = \sum_{p=1}^{k} \sum_{q=1}^{k} r_{p,q} w_{p,q} \]

for an appropriate \( r_{p,q} \). A similar expression \( z_0 \) to \( w_0 \) is obtained subsequently in Section 4.5 (See equation (4.24)).

We conclude now the process of transforming \( \Sigma_{zm} \) to a uniform rank system \( \Sigma_{wm} \) by simply saying that the above algorithmic procedure gives us,

\[ \Sigma_{wm} : \begin{cases} 
\dot{w}_{m,j} = w_{m,j+1}, & \text{for } j = 1, 2, \ldots, k-1, \\
\dot{w}_{m,k} = \tilde{g}_m(z) + u_m,
\end{cases} \]

for some linear function \( \tilde{g}_m(z) \). Expressions (4.23) and (4.24) enable us to rewrite \( \tilde{g}_m(z) \) as a function of \( w \).
4.4 Construction of a uniform rank system with the use of a post-compensator that uses more integrators than absolutely necessary

In this section, we construct the second type of post-compensator we alluded to earlier. This second type of post-compensator uses more integrators than absolutely necessary, but much simpler to construct than the one discussed in the previous section.

We select the output \( y_{wm} \) as,

\[
y_{wm} := w_{m,1} := z_{m,1} + \sum_{i=1}^{m-1} \mathcal{I}_{k+i-m} \sum_{j=1}^{m-1} \beta_{m,k+i-m,j} \tilde{z}_{j,k+1-j}
\]

\[
:= z_{m,1} + \mathcal{I}_{k+1-m} \sum_{j=1}^{m-1} \beta_{m,k+1-m,j} \tilde{z}_{j,k+1-j}
\]

\[
+ \mathcal{I}_{k+2-m} \sum_{j=1}^{m-1} \beta_{m,k+2-m,j} \tilde{z}_{j,k+1-j}
\]

\[
\vdots
\]

\[
+ \mathcal{I}_{k+i-m} \sum_{j=1}^{m-1} \beta_{m,k+i-m,j} \tilde{z}_{j,k+1-j}
\]

\[
+ \mathcal{I}_{k-2} \sum_{j=1}^{m-1} \beta_{m,k-2,j} \tilde{z}_{j,k+1-j}
\]

\[
+ \mathcal{I}_{k-1} \sum_{j=1}^{m-1} \beta_{m,k-1,j} \tilde{z}_{j,k+1-j}.
\]

The above selection of \( y_{wm} \) of \( \Sigma_{wm} \) requires more than the minimum number of integral operators than those required by \( \Sigma_{zm} \) to obtain it from the given subsystem \( \Sigma_m \). To illustrate this, let \( m = 4 \). We have then,

\[
y_{w4} := z_{4,1} + \mathcal{I}_{k-3} \sum_{j=1}^{3} \beta_{4,k-3,j} \tilde{z}_{j,k+1-j}
\]

\[
+ \mathcal{I}_{k-2} \sum_{j=1}^{3} \beta_{4,k-2,j} \tilde{z}_{j,k+1-j}
\]

\[
+ \mathcal{I}_{k-1} \sum_{j=1}^{3} \beta_{4,k-1,j} \tilde{z}_{j,k+1-j}.
\]

The above \( y_{w4} \) requires extra integral operators than the \( y_{w4} \) of Subsection 4.3.4, and these extra terms are

\[
\mathcal{I}_{k-2} \beta_{4,k-2,3} z_{3,k-2} + \mathcal{I}_{k-1} \sum_{j=2}^{3} \beta_{4,k-1,j} \tilde{z}_{j,k+1-j}.
\]

In this case, the unknown coefficients \( \beta_{m,i,j} \) turn out to be simply,

\[
\beta_{m,k+i-m,j} = -\ell_{m,k+i-m,j}, \text{ for all } j = 1 \text{ to } m-1, \text{ and } i = 1 \text{ to } m-1.
\]

We list below the algorithmic procedure that shows this equivalence.

**Step 1:** We define

\[
\mathcal{D}_1 w_{m,1} := w_{m,2}.
\]

**Step 2:** We define

\[
\mathcal{D}_2 w_{m,1} := w_{m,3}.
\]
Step $k - m$: We define
\[ D_{k-m} w_{m,1} := w_{m,k+1-m}. \]

Step $k - (m-1)$: When $y_{wm} = w_{m,1}$ is differentiated $k - (m-1) = k + 1 - m$ times, we get
\[
\dot{w}_{m,k+1-m} = z_{m,k+2-m} + \sum_{j=1}^{m-1} \ell_{m,k+1-m,j} z_{j,k+1-j} \\
+ \sum_{j=1}^{m-2} \beta_{m,k+1-m,j} z_{j,k+1-j} \\
+ \sum_{n=1}^{m-3} T_n \sum_{j=1}^{m-1} \beta_{m,k+n+1-m,j} z_{j,k+1-j}.
\]

Examination of the above expression enables us to decide,
\[ \beta_{m,k+1-m,j} = -\ell_{m,k+1-m,j}, \text{ for } j = 1 \text{ to } m - 1, \]
in order to cancel the feedback terms represented by the spade ♠.

We choose next,
\[
\dot{w}_{m,k+2-m} = z_{m,k+2-m} + \sum_{n=1}^{m-2} T_n \sum_{j=1}^{m-1} \beta_{m,k+n+1-m,j} z_{j,k+1-j} \\
+ \sum_{n=1}^{m-3} T_n \sum_{j=1}^{m-1} \beta_{m,k+n+2-m,j} z_{j,k+1-j}.
\]

The above implies that
\[ \dot{w}_{m,k+1-m} = w_{m,k+2-m}. \]

Step $k - (m-2)$: We differentiate once $w_{m,k+2-m}$, and obtain,
\[
\dot{w}_{m,k+2-m} = z_{m,k+3-m} + \sum_{j=1}^{m-1} \ell_{m,k+2-m,j} z_{j,k+1-j} \\
+ \sum_{j=1}^{m-1} \beta_{m,k+2-m,j} z_{j,k+1-j} \\
+ \sum_{n=1}^{m-3} T_n \sum_{j=1}^{m-1} \beta_{m,k+n+2-m,j} z_{j,k+1-j}.
\]

We see clearly that we can choose
\[ \beta_{m,k+2-m,j} = -\ell_{m,k+2-m,j}, \text{ for } j = 1 \text{ to } m - 1, \]
in order to cancel the feedback term represented by the spade ♠.

We repeat the above procedure $m - 4$ times

more, and go to the following step.
Step $k - 1$: We differentiate once $w_{m,k-1}$, and obtain
\[
\dot{w}_{m,k-1} = z_{m,k} + \sum_{j=1}^{m-1} \ell_{m,k-1,j} \dot{z}_{j,k+1-j} + \sum_{j=1}^{m-1} \beta_{m,k-1,j} \dot{z}_{j,k+1-j}.
\]

We choose
\[
\beta_{m,k-1,j} = -\ell_{m,k-1,j}
\]
for $j = 1$ to $m - 1$, so that the feedback represented by the spade ♠ can be canceled. This lets us to choose
\[
w_{m,k} := z_{m,k},
\]
to get
\[
\dot{w}_{m,k-1} = w_{m,k}.
\]

Step $k$: By differentiating $w_{m,k}$ once, we get
\[
\dot{w}_{m,k} = g_m(w) + u_m.
\]

We conclude now the process of transforming $\Sigma_{z0}$ to a uniform rank system $\Sigma_{w0}$ in SCB format by simply saying that the above algorithmic procedure gives us,
\[
y_{wm} := w_{m,1} := z_{m,1} - \sum_{i=1}^{m-1} \sum_{j=1}^{m-i} \ell_{m,k-i-m,j} z_{j,k+1-j},
\]
and
\[
\Sigma_{w0} : \begin{cases}
\dot{w}_{m,j} = w_{m,j+1}, & \text{for } j = 1, 2, \ldots, k - 1, \\
\dot{w}_{m,k} = \tilde{g}_m(w) + u_m,
\end{cases}
\]
for some linear function $\tilde{g}_m(w)$. The above system is in the format of a uniform rank system in SCB format with $y_{wm} := w_{m,1}$ as its output.

4.5 Post–compensator design – Transformation of $\Sigma_{z0}$ to a proper form $\Sigma_{w0}$

In connection with post-compensator design, so far in previous sections, we concentrated to transform the dynamics of infinite zero structure to the format of a uniform rank system. In this process, we redefined the outputs. This redefinition affects the way the dynamics of finite zero structure is given. As it can be observed from (4.1), $\dot{z}_0$ is defined in terms of the original outputs of $\Sigma$ and not the new outputs of $\Sigma_w$. This needs to be rectified, and it can be done by observing the relationship between the outputs of the given system $\Sigma$ and those of $\Sigma_w$. From the way the outputs of $\Sigma_w$ (namely, $w_{i,1}$, $i = 1$ to $k$) are defined, it is easy to ascertain that a linear combination of the outputs of $\Sigma$ is equivalent to a linear combination of the outputs of $\Sigma_w$ and their first, second, \ldots, and $k - 1$-th order differentials. Without even worrying about what linear
combinations of \( w_{i,j} \) needs to be counted to form \( \Sigma_{i=1}^{k} A_{0,i} y_{i} \), where \( y_{i} \), is the output of subsystem \( \Sigma_{i} \), \( i = 1 \) to \( k \), of the given system \( \Sigma \), we can assume that it is a linear combination of all the states \( w_{i,j} \) of \( \Sigma_{w} \),

\[
\Sigma_{i=1}^{k} A_{0,i} y_{i} = \Sigma_{i=1}^{k} \Sigma_{j=1}^{i} \gamma_{i,j} w_{i,j}.
\]

This enables us to rewrite \( \Sigma_{0} \) as

\[
\dot{x}_{0} = A_{0} x_{0} + \sum_{i=1}^{k} \sum_{j=1}^{i} \gamma_{i,j} w_{i,j}
\]

\[
= A_{0} x_{0} + \sum_{i=1}^{k} \gamma_{i,1} w_{i,1} + \sum_{i=1}^{k} \gamma_{i,2} w_{i,2} + \sum_{i=1}^{k} \gamma_{i,3} w_{i,3} + \cdots + \sum_{i=1}^{k} \gamma_{i,k} w_{i,k}
\]

\[
= A_{0} x_{0} + \sum_{i=1}^{k} \gamma_{i,1} w_{i,1} + \sum_{i=1}^{k} \gamma_{i,2} \dot{w}_{i,1} + \sum_{i=1}^{k} \gamma_{i,3} \dot{w}_{i,2} + \cdots + \sum_{i=1}^{k} \gamma_{i,k} \dot{w}_{i,k-1}.
\]

We present next a step by step algorithm of eliminating differentials in the last expression of the above equation.

**Step 1:** In order to eliminate the differential coefficients on the right side, we can redefine \( x_{0} \) as

\[
x_{10} = x_{0} - \sum_{i=1}^{k} \gamma_{i,2} w_{i,1} - \sum_{i=1}^{k} \gamma_{i,3} w_{i,2} - \cdots - \sum_{i=1}^{k} \gamma_{i,k} w_{i,k-1}.
\]

This leads to

\[
\dot{x}_{10} = A_{0} x_{10} + \sum_{i=1}^{k} \gamma_{i,1} w_{i,1} + \sum_{i=1}^{k} \gamma_{i,2} A_{0} w_{i,1} + \sum_{i=1}^{k} \gamma_{i,3} w_{i,2} + \cdots + \sum_{i=1}^{k} \gamma_{i,k} A_{0} w_{i,k-1},
\]

\[
= A_{0} x_{10} + \sum_{i=1}^{k} \gamma_{i,1} w_{i,1} + \sum_{i=1}^{k} \gamma_{i,2} \dot{w}_{i,1} + \cdots + \sum_{i=1}^{k} \gamma_{i,k} \dot{w}_{i,k-2},
\]

for some appropriately defined coefficients \( \gamma_{i,l,j} \). We observe that, in this step 1, we eliminated the presence of \( \dot{w}_{i,k-1} \) on the right side of the above expression.

**Step 2:** In order to eliminate the differential coefficients on the right side of the last equation, we can redefine,

\[
x_{20} = x_{10} - \sum_{i=1}^{k} \gamma_{i,3} w_{i,1} - \cdots - \sum_{i=1}^{k} \gamma_{i,k} w_{i,k-1}.
\]

This gives rise to

\[
\dot{x}_{20} = A_{0} x_{20} + \sum_{i=1}^{k} \gamma_{i,1} w_{i,1} + \sum_{i=1}^{k} \gamma_{i,2} A_{0} w_{i,1} + \sum_{i=1}^{k} \gamma_{i,3} w_{i,2} + \cdots + \sum_{i=1}^{k} \gamma_{i,k} A_{0} w_{i,k-2},
\]

\[
= A_{0} x_{20} + \sum_{i=1}^{k} \gamma_{i,1} w_{i,1} + \sum_{i=1}^{k} \gamma_{i,2} \dot{w}_{i,1} + \cdots + \sum_{i=1}^{k} \gamma_{i,k} \dot{w}_{i,k-3},
\]

for some appropriately defined coefficients \( \gamma_{i,l,j} \). We observe that, in this step 2, we eliminated the presence of \( \dot{w}_{i,k-2} \) on the right side of the above expression.

Proceeding step by step as illustrated above, it is easy to see that in the \((k-1)\)-th step, we get

\[
\dot{w}_{0} = A_{0} w_{0} + \sum_{i=1}^{k} \gamma_{i,1} w_{i,1},
\]

for some appropriately defined coefficients \( \gamma_{i,1} \). We can also obtain an expression relating \( z_{0} \) to \( w_{0} \) in the form,

\[
z_{0} = w_{0} + \sum_{p=1}^{k} \sum_{q=1}^{k} \gamma_{0,p,q} w_{p,q}, \quad (4.24)
\]

\[64\]
for an appropriate $r_{p,q}^0$.

The above equation describes the finite zero structure of $\Sigma_w$ in the format of SCB as the right hand side of the above equation consists of only the variable $w_0$ and the outputs of $\Sigma_w$. We observe that the finite zeros (or invariant zeros) of $\Sigma_w$ are the eigenvalues of $A_0$. This implies that the finite zeros (or invariant zeros) of $\Sigma_w$ are exactly the same as those of the given system $\Sigma$. 
Chapter 5

Codes to transform a square strictly proper system to a uniform rank system in SCB format

This chapter gives two functions that transform any invertible strictly proper square system to a uniform rank system, one function that uses pre-compensators and another that uses post-compensators.

5.1 Transformation code by using pre-compensator

The function that transforms any invertible square strictly proper system to a uniform rank system in SCB format by using a pre-compensator is:

\[ [A W, B W, C W, D W, A c, B c, C c, D c, dim, qdim] = \text{rankuniform}_\text{pre}(A, B, C, D). \]  

(5.1)

5.1.1 Main function: rankuniform_pre

\[ [A W, B W, C W, D W, A c, B c, C c, D c, dim, qdim] = \text{rankuniform}_\text{pre}(A, B, C, D) \]

\[ [A A, B B, C C, D D, G s, G o, G i, d i m s, l v, r v, q v, m 0, e r r _ s c b, \ldots, \]

\[ \text{for}_g m 8, G m o r] = \text{zzscb}(A, B, C, D); \]
Dim = dims;

%find the dimension of each infinite-zero subsystem
K = max(qv); % highest order of infinite-zero subsystem
Colofqv = numel(qv); % number of elements of qv
DimofSubsys = zeros(1, K);
j = 1;

for i = 1:Colofqv
    while (qv(i) ~= j)
        j = j + 1;
    end
    DimofSubsys(j) = DimofSubsys(j) + 1;
end

% Modify AA by adding zero coefficients for any
% DimofSubsys(j) = 0, j = 1 to K. Then DimofSubsys(j) = 1
m = 1;
Zerodim = 0;
DimA0 = Dim(1);
N = max(size(AA));
j = 1;
t = DimA0;
while (j <= K)
    if (DimofSubsys(j) == 0)
for i=1:j-1
    t=t+Dim(i)*i;
end

AA=[AA(1:t,:);zeros(j,N);AA(t+1:N,:)];
AA=[AA(:,1:t),zeros(N+j,j),AA(:,t+1:N)];
N=N+j;
DimofSubsys(j)=1;
Zerodim(:,m)=j;
m=m+1;
end

j=j+1;
end

Move columns and rows in AA (generated by SCB code)
% to the SCB form in the paper
% step 1: move columns in AA
% for DimofSubsys(1), these columns are not moved.
% So we start from DimofSubsys(2)

j=2;
t=DimA0;
while (j<=K)
    if (DimofSubsys(j)>1)
        for i=1:j-1
            t=t+DimofSubsys(i)*i;
        end
    end
    for m=1:j

68
for \( i = 1 : \text{DimofSubsys}(j) \)
\[
A1(:, \text{DimofSubsys}(j) * (m-1) + i) = AA(:, t + (i - 1) * j);
\]
end
end

\[ AA(:, t + 1: t + \text{DimofSubsys}(j) * j) = A1; \]
end

\( j = j + 1; \)
end
%
Move rows in AA
%for \text{DimofSubsys}(1), these columns are not moved.
%So we start from \text{DimofSubsys}(2)

\( j = 2; \)
\( t = \text{DimA0}; \)
while \( j \leq K \)
  if \( \text{DimofSubsys}(j) > 1 \)
    for \( i = 1: j - 1 \)
      \( t = t + \text{DimofSubsys}(i) * i; \)
    end
  end
  for \( m = 1 : j \)
    for \( i = 1 : \text{DimofSubsys}(j) \)
      \( A2(\text{DimofSubsys}(j) * (m-1) + i,:) = AA(t + m + (i - 1) * j,:); \)
    end
  end
end

\[ AA(t + 1: t + \text{DimofSubsys}(j) * j,:) = A2; \]
end

j=j+1;
end

%Get garbage Matrix G and coefficients of bias function V

%Get G
j=1;
m=0;
t=dimA0;
while (j<=K)
    G(m+1:m+dimofSubsys(j),:)=AA((t+1+dimofSubsys(j)*j-1),...,:+(t+dimofSubsys(j)*j,:),:);
    m=m+dimofSubsys(j);
    t=t+dimofSubsys(j)*j;
    j=j+1;
end

%Get V

t=0;
m=0;
for j=2:K
    t=t+dimofSubsys(j)*(j-1);
    m=m+dimofSubsys(j-1);
end

70
V=zeros(t,m);

j=2;
m1=0;
m2=0;
t1=DimA0+DimofSubsys(1);
t2=DimA0;
while (j<=K)
    for i=1:j-1
        V(m1+1:m1+DimofSubsys(j)*(j-1),m2+1:m2+DimofSubsys(i))= ... , AA(t1+1:t1+DimofSubsys(j)*(j-1),t2+1:t2+DimofSubsys(i));
        m2=m2+DimofSubsys(i);
        t2=t2+DimofSubsys(i)*i;
    end
    m1=m1+DimofSubsys(j)*(j-1);
    m2=0;
    t2=DimA0;
    t1=t1+DimofSubsys(j)*j;
    j=j+1;
end

% So far, garbage matrix G and coefficients matrix V of % bias function \( f_{\{i,j\}}(w) \) are ready. Next step is to updated % garbage matrix and generate uniform-rank system

s=0;
for i=1:K-1
s = s + DimofSubsys(i) * (K - i);
end
m = size(G);
CompG = zeros(s, m(2));
s = 0; % for CompGs
G1 = G;
for i = 1:K - 1
    t = 0;
    for m = 1:i - 1
        t = t + DimofSubsys(m);
    end
    A0 = AA(1: DimA0, :);
a = G(t + 1:t + DimofSubsys(i), :);
CompG(s + 1:s + DimofSubsys(i), :) = a;
s = s + DimofSubsys(i);
UpdateTime = K - i;
while (UpdateTime >= 1)
    [abar] = updateG(G, V, A0, a, K, DimofSubsys, DimA0);
a = abar;
    UpdateTime = UpdateTime - 1;
    if (UpdateTime >= 1)
        CompG(s + 1:s + DimofSubsys(i), :) = a;
    end
end
\[ s = s + \text{DimofSubsys}(i); \]

end

end

\[ G1(t+1:t+\text{DimofSubsys}(i),:) = \text{abar}; \]

end

\[ G = G1; \]

% Generate R matrix whose elements are \( r_{p,i,j} \)
\[ t = \text{DimA0}; \]

for \( m = 1:K \)
\[ t = t + \text{DimofSubsys}(m) \times K; \]
end

\[ s = 0; \]

for \( m = 1:K \)
\[ s = s + \text{DimofSubsys}(m); \]
end

\[ R = \text{zeros}(s,t); \]

% Update the updated-garbage matrix G to garbage matrix R
\[ \text{for } i = 1:K \]
\[ t = 0; \]

\[ \text{for } m = 1:i-1 \]
\[ t = t + \text{DimofSubsys}(m); \]
end

\[ A0 = \text{AA}(1: \text{DimA0},:); \]
a=G(t+1:t+DimofSubsys(i),:);
[ rk]=getRk(V,a,K,DimofSubsys,DimA0);
%[ ri]=getRi(rk,V,a,A0,K,DimofSubsys,DimA0);
R(t+1:t+DimofSubsys(i),:)=rk;
end

%The last step is to add the bias term to matrix R, %and generate garbage matrix Gw
Gw=zeros(size(R));
for i=1:K
    t=0;
    for m=1:i-1
        t=t+DimofSubsys(m);
    end
    a=R(t+1:t+DimofSubsys(i),:);
    nb=i;
    [ gw]=getGw(nb,V,a,K,DimofSubsys,DimA0);
    Gw(t+1:t+DimofSubsys(i),:)=gw;
end

%Generate the coefficients for uniform rank system AW
%Change A0 to the size of Gw
m=size(Gw);
Aw0=zeros(DimA0,m(2));
Aw0( :, 1:DimA0) = A0(:, 1:DimA0);
for i = 1:K
    s1 = DimA0; s2 = DimA0;
    for m = 1:i - 1
        s1 = s1 + DimofSubsys(m) * K;
    end
    for m = 1:i - 1
        s2 = s2 + DimofSubsys(m) * m;
    end
    Aw0(:, s1 + 1: s1 + DimofSubsys(i) * i) = A0(:, s2 + 1: s2 + DimofSubsys(i) * i);
end
AG = [Aw0; Gw];

% Add zero rows to AG
m = size(Gw);
AW = zeros(m(2), m(2));
AW(1: DimA0, :) = AG(1: DimA0, :);
for i = 1:K
    s1 = DimA0; s2 = DimA0;
    for m = 1:i - 1
        s1 = s1 + DimofSubsys(m) * K;
    end
    for m = 1:i - 1
        % Further code
    end
end
s2 = s2 + DimofSubsys(m);
end

AW(s1+1:DimofSubsys(i)∗(K−1):s1+DimofSubsys(i)∗K,:) = ... ,
AG(s2+1:s2+DimofSubsys(i),:);
end

% Add identity matrix to AW
for i = 1:K
s1 = DimA0;
for m = 1:i − 1
s1 = s1 + DimofSubsys(m)∗K;
end
ni = DimofSubsys(i);
Ii1 = eye(ni∗(K−1));
z1i = zeros(ni∗(K−1), ni);
Ii = [z1i, Ii1];
AW(s1 +1:s1 +ni∗(K−1), s1 +1:s1 +ni∗K) = Ii;
end

% Move columns and rows to be a standard SCB form
% Move columns
j = 1;
t = DimA0;
while (j <= K)
if (DimofSubsys(j) > 1)
for i = 1:j − 1
    t = t + DimofSubsys(i) * K;
end
for m = 1:DimofSubsys(j)
    for i = 1:K
        A3(:, K * (m − 1) + i) = AW(:, t + m + (i − 1) * DimofSubsys(j));
    end
end
AW(:, t + 1:t + DimofSubsys(j) * K) = A3;
end
j = j + 1;
end

% Move rows
j = 1;
t = DimA0;
while (j <= K)
    if (DimofSubsys(j) > 1)
        for i = 1:j − 1
            t = t + DimofSubsys(i) * K;
        end
        for m = 1:DimofSubsys(j)
            for i = 1:K
\[
A_4(K*(m-1)+i,:) = AW(t+m+(i-1)*\text{DimofSubsys}(j,:),:);
\]

end
end

\[
AW(t+1:t+\text{DimofSubsys}(j)*K,:) = A_4;
\]
end

j=j+1;
end

% Remove zero coefficients added at the beginning
numofzero = 0;
if (Zerodim~ = 0)
    numofzero = max(size(Zerodim));
end

for i = 1:numofzero
    nsub = Zerodim(i);
    s1 = DimA0;
    for j = 1:nsub-1
        s1 = s1 + \text{DimofSubsys}(j)*K;
    end
AW(s1+1:s1+K,:) = [];
AW(:, s1+1:s1+K) = [];
DimofSubsys(nsub) = 0;
end
%Generate BW,CW,DW
m = size(AW);
t = 0;
for j = 1:K
    t = t + DimofSubsys(j);
end
BW = zeros(m(1), t);
CW = zeros(t, m(1));
DW = zeros(t, t);
s = DimA0;
for j = 1:t
    BW(s + K, j) = 1;
    CW(j, s + 1) = 1;
    s = s + K;
end

%Generate Ac, Bc, Cc, Dc
%Remove zero rows and columns in CompG
%remove zero rows
for i = 1:K - 1
    if DimofSubsys(i) == 0
        t = 0;
    end
end
for j=1:i-1  
t=t+DimofSubsys(j)*(K-j);  
end  
CompG(t+1:t+(K-i),:)=[];  
end  
end

%remove zero columns and the first column (A0);  
CompG(:,1:DimA0)=[];

for i=1:K  
if (i==1)  
    if (DimofSubsys(i)==0)  
        CompG(:,1)=[];  
    end  
    t=t+DimofSubsys(i);  
else  
    if (DimofSubsys(i)==0)  
        CompG(:,t+1:t+i)=[];  
    else  
        CompG(:,t+1:t+DimofSubsys(i)*(i-1))=[];  
        t=t+DimofSubsys(i);  
end  
end
Beta coefficient is negative of the garbage coefficients
CompG = CompG * (-1);

t1 = 0; t2 = 0;
for i = 1:K-1
    t1 = t1 + DimofSubsys(i) * (K-i);
    t2 = t2 + DimofSubsys(i);
end
t2 = t2 + DimofSubsys(K);

Ac = zeros(t1, t1);
Bc = zeros(t1, t2);
Cc = zeros(t2, t1);
Dc = zeros(t2, t2);

% Ac
Nm = size(CompG);
Nn = Nm(1);
t = 0; s = 0; p = 0;
for i = 1:K-1
    if (DimofSubsys(i)^2 = 0)
Ac(:,t+1:t+DimofSubsys(i))=CompG(:,s+1:s+DimofSubsys(i));
s=s+DimofSubsys(i);
t=t+DimofSubsys(i);

m=DimofSubsys(i)*(K-i-1);
if (m=0)
   In=[zeros(p,m);eye(m);zeros(Nn-p-m,m)];
   Ac(:,t+1:t+m)=In;
   t=t+m;
end
p=p+DimofSubsys(i)*(K-i);
end

%the above s should not be reset, it is used below

%Bc

t=0;p=0;
for i=1:K-1
   if (DimofSubsys(i)=0)
      p=p+DimofSubsys(i)*(K-i-1);
      In=[zeros(p,DimofSubsys(i));eye(DimofSubsys(i));..., zeros(Nn-p-DimofSubsys(i),DimofSubsys(i))];
      Bc(:,t+1:t+DimofSubsys(i))=In;
   end
end
\[ p = p + \text{DimofSubsys}(i); \]
\[ t = t + \text{DimofSubsys}(i); \]
end
end

\[ Bc(:, t+1:t+\text{DimofSubsys}(K)) = \text{CompG}(:, s+1:s+\text{DimofSubsys}(K)); \]

\%Cc
\[ t1 = 0; t2 = 0; \]
\[ \text{for } i = 1:K-1 \]
\[ \text{if } (\text{DimofSubsys}(i) \neq 0) \]
\[ Cc(t1+1:t1+\text{DimofSubsys}(i), t2+1:t2+\text{DimofSubsys}(i)) = \ldots, \]
\[ \text{eye(DimofSubsys}(i)) ; \]
\[ t1 = t1 + \text{DimofSubsys}(i) ; \]
\[ t2 = t2 + \text{DimofSubsys}(i) \times (K-i) ; \]
end
end

\%Dc
\[ t1 = 0; \]
\[ \text{for } i = 1:K-1 \]
\[ t1 = t1 + \text{DimofSubsys}(i) ; \]
end

\[ Dc(t1+1:t1+\text{DimofSubsys}(K), t1+1:t1+\text{DimofSubsys}(K)) = \ldots, \]
\[ \text{eye(DimofSubsys}(K)) ; \]
%Return the updated dim vector

dim=dims;

t=0;

for i=1:K
    t=t+DimofSubsys(i);
end

dim(6)=t*K;

qdim=size(1,t);

for j=1:t
    qdim(j)=K;
end

5.1.2 Subfunction: updateG

function [abar]=updateG(G,V,A0,a,K,DimofSubsys,DimA0)

%This is a function to update the coefficients of g_{i,j}(x)
%after adding an integrator (not the last added integrator)
%before subsystems

%update b_{10}
abar=zeros(size(a));

t=DimA0;

b10=a(:,1:t)*A0(:,1:t);

m=0;

for i=1:K;
    a_1pp(:,m+1:m+DimofSubsys(i))= ...,
a(:,t+1:DimofSubsys(i)+(i-1):t+DimofSubsys(i)*i);

m=m+DimofSubsys(i);
t=t+DimofSubsys(i)*i;
end

a_p0=G(:,1:DimA0);
b10=b10+a_pp*a_p0;
abar(:,1:DimA0)=b10;

update b_{1ij}
for i=1:K
    t=DimA0;
    for m=1:i-1
        t=t+DimofSubsys(m)*m;
    end
    for j=1:i
        b1=0;
        b2=0;
        b3=0;
        b4=0;
        if (j==1)
            b1=a(:,1:DimA0)*A0(:,t+1:t+DimofSubsys(i));
        end
        for p=i+j:K
            s1=0;
        end
    end
    end
end

85
for m=2:p-1
    s1=s1+DimofSubsys(m)*(m-1);
end
s2=0;
for m=1:i-1
    s2=s2+DimofSubsys(m);
end
t1=DimA0;
for m=1:p-1;
    t1=t1+DimofSubsys(m)*m;
end
for q=1:p-1
    a_{1pq}=a(:,t1+1+(q-1)*DimofSubsys(p):...,t1+q*DimofSubsys(p));

    if (i>=K)
        v_{pqi}=0;
    else
        v_{pqi}=V(s1+1+(q-1)*DimofSubsys(p):...,s1+q*DimofSubsys(p),s2+1:s2+DimofSubsys(i));
    end
end
b4=b4+a_{1pq}*v_{pqi};
end
end
else

\[ b_2 = a (:, t+1+(j-2)*\text{DimofSubsys}(i):t+(j-1)*\text{DimofSubsys}(i)) \]

end

\[ a_{pij} = G(:, t+1+(j-1)*\text{DimofSubsys}(i):t+j*\text{DimofSubsys}(i)) \]

\[ b_3 = a_{pp} \cdot a_{pij} \]

\[ \text{abar} (:, t+1+(j-1)*\text{DimofSubsys}(i):t+j*\text{DimofSubsys}(i)) = b_1 + b_2 + b_3 + b_4 \]

end

5.1.3 Subfunction: getGw

function \([gw] = \text{getGw}(nb, V, a, K, \text{DimofSubsys}, \text{DimA0})\)

% This function is used to get the rank-uniformized garbage matrix Gw

\[ gw = \text{zeros(size}(a)); \]

\[ gw(:, 1:\text{DimA0}) = a(:, 1:\text{DimA0}); \]

for \( i = 1:K \)

\[ t = \text{DimA0}; \]

for \( m = 1:i-1 \)

\[ t = t + \text{DimofSubsys}(m) \cdot K; \]

end

for \( j = 1:K \)

\[ a_{1ij} = a (:, t+1+(j-1)*\text{DimofSubsys}(i):t+j*\text{DimofSubsys}(i)) ; \]

\[ s1 = 0; s2 = 0; \]
for m=2:nb−1
    s1=s1+DimofSubsys(m)∗(m−1);
end
for m=1:i−1
    s2=s2+DimofSubsys(m);
end
if ((i>=K)||(K−j+1>=nb))
    v_nbi=0;
else
    v_nbi=V(s1+(K−j+1)+DimofSubsys(nb):...,
              s1+(K−j+1)∗DimofSubsys(nb),s2+1:s2+DimofSubsys(i));
end
for m=2:nb−1
    gw(:,t+1+(j−1)∗DimofSubsys(i):t+j∗DimofSubsys(i))=a_ij+v_nbi;
end
end

5.1.4 Subfunction: getRk

function rk=getRk(V,a,K,DimofSubsys,DimA0)
	%This function is used to get the coefficient $r_{k,i,j}$ in the algorithm. This is from transformation of $\Sigma_k$

m=size(a);
T=DimA0;
for i=1:K
    t=t+DimofSubsys(i)*K;
end
rk=zeros(m(1),t);

%$r_{k,0}$ is not changed
rk(:,1:DimA0)=a(:,1:DimA0);
for i=1:K
    for j=1:K
        r1=0;
        r2=0;
        t1=DimA0;
        for m=1:i-1;
            t1=t1+DimofSubsys(m)*m;
        end
        if (j<=i)
            r1=a(:,t1+1+(j-1)*DimofSubsys(i):t1+j*DimofSubsys(i));
        end
        for q=j+1:K
            for p=max(i+1,q):K
                t1=DimA0;
                for m=1:p-1;
                    t1=t1+DimofSubsys(m)*m;
                end
            end
        end
    end
end
end

\[ a_{kpq} = a(:, t1 + 1 + (q - 1) \times \text{DimofSubsys}(p): t1 + q \times \text{DimofSubsys}(p)) ; \]

\[ s1 = 0; \]
\[ s2 = 0; \]

for \( m = 2:p - 1 \)
\[ s1 = s1 + \text{DimofSubsys}(m) \times (m - 1); \]
end

for \( m = 1:i - 1 \)
\[ s2 = s2 + \text{DimofSubsys}(m); \]
end

if \( i \geq K \)
\[ v_{pqji} = 0; \]
else
\[ v_{pqji} = V(s1 + 1 + (q - j - 1) \times \text{DimofSubsys}(p): \ldots, \]
\[ s1 + (q - j) \times \text{DimofSubsys}(p), s2 + 1:s2 + \text{DimofSubsys}(i)); \]
end

\[ r2 = r2 + a_{kpq} \times v_{pqji}; \]
end
end

t = \text{DimA0};

for \( m = 1:i - 1; \)
\[ t = t + \text{DimofSubsys}(m) \times K; \]
end
rk (: , t+1+(j−1)*DimofSubsys(i):t+j*DimofSubsys(i))=r1−r2;
end

end

5.1.5 Subfunction: zzscb
:
[AA,BB,CC,DD,Gs,Go,Gi,dims,lv,rv,qv,m0,err_scb,for_gm8_Gmor]= ...,
zzscb(A,B,C,D,tol,dc)

%ZZSCB Special Coordinate Basis for Continuous−time Systems
% [ At , Bt , Ct , Dt , Gms,Gmo,Gmi,dim ] = zzscb( A, B, C, D )
% decomposes a continuous−time system characterized by (A,B,C,D)
% into the standard SCB form with state subspaces x_a being
% separated into stable, marginally stable and unstable parts
% (in continuous−time sense), and x_d being decomposed into the
% form of integration chains.
% Input Parameters:
% x = A x + B u , y = C x + D u
% Output Parameters:
% x_t = At x_t + Bt u_t , y_t = Ct x_t + Dt u_t
% where x_t = [ x_a⁻ x_a⁺0 x_a⁺ x_b x_c x_d ]’ with
% dim = [ n_a⁻ , n_a⁺0 , n_a⁺ , n_b , n_c , n_d ],
% and Gms, Gmo & Gmi = state, output & input transformations.
% See also SCBRAW, DSCB and SSD.
% dc : dc=0, for continuous−time system; (default)
% dc=1, for discrete−time system.

if nargin==4,
tol=1e-8; dc=0;

elseif nargin==5
    dc=0;
end

n=size(A,1);
pp=size(C,1);
mm=size(B,2);

%Check whether [B;D] and [C,D] are full rank.
[u0,s0,v0]=svd([B;D]);
m=rank(s0,tol);
if mm==m,
    v0=eye(m);
else %[B;D] is not full rank
    B=B*v0;B=B(:,1:m);
    if isempty(D)
        D=D*v0;D=D(:,1:m);
    end
end

end;
[u1,s1,v1]=svd([C,D]);
p=rank(s1,tol);
if pp==p,
```

u1=eye(p);

else

    \[ C, D \] is not full rank

    C=u1'*C; C=C(1:p,:);
    if ~isempty(D)
        D=u1'*D; D=D(1:p,:);
    end

end


%Compute m0

if norm(D)>tol

    m0=rank(D,tol);
    [u,s,v]=svd(D); tt=[]; ttt=[];
    for kk=1:m0,
        tt(kk,kk)=sqrt(s(kk,kk));
        ttt(kk,kk)=1/sqrt(s(kk,kk));
        tt2(kk,kk)=s(kk,kk);
        ttt2(kk,kk)=1/s(kk,kk);
    end

    T2=u*blkdiag(tt,eye(p-m0));
    invT2=blkdiag(ttt,eye(p-m0))*u';
    T3=v*blkdiag(ttt,eye(m-m0));
    invT3=v'*blkdiag(tt,eye(m-m0));

    if nargin==7
```
T2=u;
invT2=u’;
T3=v*blkdiag(ttt2,eye(m-m0));
invT3=v’*blkdiag(ttt2,eye(m-m0));
end
Bt=B*T3;
B0=Bt(:,1:m0);
B1=Bt(:,m0+1:m);
Ct=invT2*C;
C0=Ct(1:m0,:);
C1=Ct(m0+1:p,:);
A1=A-B0*C0;
elseif norm(D)<=tol
A1=A;B1=B;C1=C;
B0=zeros(n,0);C0=zeros(0,n);D0=[];
m0=0;T2=eye(p);invT2=eye(p);T3=eye(m);invT3=eye(m);
end
D0=eye(m0);

%Structure decomposition of (A1,B1,C1)
rv=[]; lv=[]; qv=[]; Go=[]; Gi=[];
if (isempty(C1)) & (isempty(B1))
AA=A1; BB=[]; CC=[]; di3=[0,0,0]; Gs=eye(size(A1,1));

end

if (~isempty(C1)) & (isempty(B1))
    [AA,CC,Gs,Go,no,lv,k,k,invGs]=osd(A1,C1,tol);
    BB=[]; di3=[sum(lv),0,0];
end

if (isempty(C1)) & (~isempty(B1))
    [AA,BB,Gs,Gi,no,rv,k,k,invGs]=csd(A1,B1,tol);
    di3=[0,sum(rv),0]; CC=[];
end

if (isempty(B1) | isempty(C1))
    nt=size(AA,1)-sum(di3);
    At=AA(1:nt,1:nt);
    if dc==0
        [Aa,tt,nn,n0,np]=ssd(At,tol);
    else
        [Aa,tt,nn,n0,np]=dssd(At,tol);
    end
    t1=blkdiag(tt,eye(n-nt));
    Gs=Gs*t1; invGs=inv(Gs);
    AA(1:nt,1:nt)=Aa; dims=[nn,n0,np,di3];
end

if (~isempty(C1)) & (~isempty(B1))

95
[AA, BB, CC, Gs, Go, Gi, lv, rv, qv, dims, invGs] = zzspscb(A1, B1, C1, tol, dc);
end

DD = zeros(size(CC, 1), size(BB, 2));

na = sum(dims(1:3));
AA(1:na, 1:na) = AA(1:na, 1:na) .* blkdiag(ones(dims(1)), ..., ones(dims(2)), ones(dims(3)));

% Structure decomposition of (A,B,C,D)
B0 = invGs * B0; C0 = C0 * Gs;
BB = [B0, BB]; CC = [C0; CC];
DD = zeros(p, m); DD(1:m0, 1:m0) = eye(m0);
for gm8_Gmor = Go;
    Go = T2 * blkdiag(eye(m0), Go);
    Gi = T3 * blkdiag(eye(m0), Gi);
end

err_scb_A = norm(AA + B0 * C0 - invGs * A * Gs); err_scb_B = 0; err_scb_C = 0;
if ~isempty(B)
    err_scb_B = norm(BB - invGs * B * Gi);
end
if ~isempty(C)
    err_scb_C = norm(CC - inv(Go) * C * Gs);
end

96
% If \([B;D]\) and \([C,D]\) are not of full rank.

\[
BB=[BB,\text{zeros}(n,\text{mm-m})];
\]

\[
CC=[CC;\text{zeros}(pp-p,n)];
\]

\[
DD=\text{blkdiag}(DD,\text{zeros}(pp-p,\text{mm-m}));
\]

\[
Go=u1*\text{blkdiag}(Go,\text{eye}(pp-p));
\]

\[
Gi=v0*\text{blkdiag}(Gi,\text{eye}(mm-m));
\]

\[
err_{scb}=[err_{scb_A},err_{scb_B},err_{scb_C}];
\]

### 5.2 Transformation code by using post-compensator

The function that transforms any invertible square strictly proper system to a uniform rank system in SCB format by using a post-compensator is:

\[
[AW,BW,CW,DW,Ac,Bc,Cc,Dc,\text{dim},\text{qdim}] = \text{rankuniform}_{\text{post}}(A,B,C,D).
\]

(5.2)

#### 5.2.1 Main function: \textit{rankuniform}_{\text{post}}

```matlab
function [AW,BW,CW,DW,Ac,Bc,Cc,Dc,\text{dim},\text{qdim}]=\text{rankuniform}_{\text{post}}(A,B,C,D)

[AA,BB,CC,DD,Gs,Go,Gi,\text{dims},lv,rv,qv,m0,err_{scb},for_{gm8}_{Gmor}]=...,
zzscb(A,B,C,D);

Dim=\text{dims};

% find the dimension of each infinite-zero subsystem
K=max(qv);% highest order of infinite-zero subsystem
Colofqv=numel(qv);% number of elements of qv
DimofSubsys=zeros(1,K);

j=1;

for i=1:Colofqv
```

97
while (qv(i) = j)
    j = j + 1;
end

DimofSubsys(j) = DimofSubsys(j) + 1;

end

% Modify AA by adding zero coefficients for any DimofSubsys(j) = 0,
% j = 1 to K. Then DimofSubsys(j) = 1

m = 1;
Zerodim = 0;
DimA0 = Dim(1);
N = max(size(AA));
j = 1;
t = DimA0;
while (j <= K)
    if (DimofSubsys(j) == 0)
        for i = 1:j - 1
            t = t + Dim(i) * i;
        end
        AA = [AA(1:t, :); zeros(j, N); AA(t + 1:N, :)];
        AA = [AA(:, 1:t), zeros(N + j, j), AA(:, t + 1:N)];
        N = N + j;
        DimofSubsys(j) = 1;
        Zerodim(:, m) = j;
    end
end
m=m+1;

end

j=j+1;

end

Move columns and rows in AA (generated by SCB code)
to the SCB form in the paper

step 1: move columns in AA

for DimofSubsys(1), these columns are not moved.
So we start from DimofSubsys(2)

j=2;
t=DimA0;
while (j<=K)
    if (DimofSubsys(j)>1)
        for i=1:j-1
            t=t+DimofSubsys(i)*i;
        end
        for m=1:j
            for i=1:DimofSubsys(j)
                A1(:,DimofSubsys(j)*(m-1)+i)=AA(:,t+m+(i-1)*j);
            end
        end
    end
    AA(:,t+1:t+DimofSubsys(j)*j)=A1;
end

j=j+1;
end

%Move rows in AA
%for DimofSubsys(1), these columns are not moved. So we start from
%DimofSubsys(2)

j = 2;
t = DimA0;

while (j <= K)
    if (DimofSubsys(j) > 1)
        for i = 1:j - 1
            t = t + DimofSubsys(i) * i;
        end
        for m = 1:j
            for i = 1:DimofSubsys(j)
                A2(DimofSubsys(j) * (m - 1) + i, :) = AA(t + m + (i - 1) * j, :);
            end
        end
    end
    AA(t + 1:t + DimofSubsys(j) * j, :) = A2;
end

j = j + 1;
end

%Get garbage Matrix G and coefficients of bias function V

%Get G

100
j = 1;
m = 0;
t = DimA0;

while (j <= K)
    G(m + 1:m + DimofSubsys(j), :) = AA((t + 1 + DimofSubsys(j) * (j - 1)) ... ,
                                      : (t + DimofSubsys(j) * j), :);

    m = m + DimofSubsys(j);
    t = t + DimofSubsys(j) * j;
    j = j + 1;
end

% Get V

j = 2;
m1 = 0;
m2 = 0;
t1 = DimA0 + DimofSubsys(1);
t2=DimA0;

while (j<=K)
    for i=1:j-1
        V(m1+1:m1+DimofSubsys(j)*(j-1),m2+1:m2+DimofSubsys(i))= ..., AA(t1+1:t1+DimofSubsys(j)*(j-1),t2+1:t2+DimofSubsys(i));
        m2=m2+DimofSubsys(i);
        t2=t2+DimofSubsys(i)*i;
    end
    m1=m1+DimofSubsys(j)*(j-1);
    m2=0;
    t2=DimA0;
    t1=t1+DimofSubsys(j)*j;
    j=j+1;
end

% So far, garbage matrix G and coefficients matrix V of 
% bias function \( L_{i,j}(w) \) are ready. Next step is 
% to updated V matrix to L matrix according to the updated algorithm.

% Update bias coefficients V, 
% and get the updated coefficients named L

L=V;

for m=4:K
    s=0;
    for t=2:m-1
        s=s+DimofSubsys(t)*(t-1);
    end
end
for i=K+1-m+2:K-1
    s1=s+(i-(K+1-m))*DimofSubsys(m);
    for j=1:m-1
        w1=0;w2=0;
        s2=0;
        for t=1:j-1
            s2=s2+DimofSubsys(t);
        end
        w1=V(s1+1:s1+DimofSubsys(m),s2+1:s2+DimofSubsys(j));
    end
    w1=V(s1+1:s1+DimofSubsys(m),s2+1:s2+DimofSubsys(j));
    for p=K+1-m+1:i-1
        s3=s+(p-(K+1-m))*DimofSubsys(m);
        for q=max(K+1-p,j+1):m-1
            s4=0;
            for t=1:q-1
                s4=s4+DimofSubsys(t);
            end
            lmpq=L(s3+1:s3+DimofSubsys(m),s4+1:s4+DimofSubsys(q));
        end
        u=0;
        for t=2:q-1
            u=u+DimofSubsys(t)*(t-1);
        end
        s5=u+(i+K-p-q-(K+1-q))*DimofSubsys(q);
        lqij=V(s5+1:s5+DimofSubsys(q),s2+1:s2+DimofSubsys(j));
    end
\[ w_2 = w_2 + l_{mpq} \cdot l_{ij}; \]

\[ \text{end} \]

\[ \text{end} \]

\[ L(s_1 + 1 : s_1 + \text{DimofSubsys}(m), s_2 + 1 : s_2 + \text{DimofSubsys}(j)) = w_1 + w_2; \]

\[ \text{end} \]

\[ \text{end} \]

\[ \% \text{Get beta matrix} \]

\[ \text{Beta} = -L; \]

\[ \text{for } m = 3 : K \]

\[ s = 0; \]

\[ \text{for } t = 2 : m - 1 \]

\[ s = s + \text{DimofSubsys}(t) \cdot (t - 1); \]

\[ \text{end} \]

\[ \text{for } i = K + 1 - m + 1 : K - 1 \]

\[ nd = i - (K + 1 - m); \]

\[ s_1 = s + nd \cdot \text{DimofSubsys}(m); \]

\[ \text{for } j = m - nd : m - 1 \]

\[ s_2 = 0; \]

\[ \text{for } t = 1 : j - 1 \]

\[ s_2 = s_2 + \text{DimofSubsys}(t); \]

\[ \text{end} \]

\[ \text{Beta}(s_1 + 1 : s_1 + \text{DimofSubsys}(m), s_2 + 1 : s_2 + \text{DimofSubsys}(j)) = 0; \]
end
end
end

%Generate complete matrix Rx for Z.{i,j}
s1=0;
for t=1:K
    s1=s1+DimofSubsys(t)*K;
end
Rx=zeros(s1,s1);
%generate for z.{1,j}
s1=DimofSubsys(1)*K;
Rx(1:s1,1:s1)=eye(s1);
%generate for z.{i,j} and i>=2
for i=2:K
    s2=0;
    for t1=2:i-1
        s2=s2+DimofSubsys(t1)*(t1-1);
    end
    ss=0;
    for t1=1:i-1
        ss=ss+DimofSubsys(t1)*K;
    end
end
for \( j = K + 1 - i : K \)

\[
s_1 = s + \text{DimofSubsys}(i) \cdot (j - 1);
\]

\[
\text{Rx}(s_1 + 1:s_1 + \text{DimofSubsys}(i), s_1 + 1:s_1 + \text{DimofSubsys}(i)) = \ldots, \quad \text{eye}(\text{DimofSubsys}(i));
\]

for \( s = j : K - 1 \)

\[
s_3 = s_2 + (s - (K + 1 - i)) \cdot \text{DimofSubsys}(i);
\]

for \( t = 1:K - s \)

\[
s_4 = 0;
\]

for \( t_1 = 1:t - 1 \)

\[
s_4 = s_4 + \text{DimofSubsys}(t_1);
\]

end

\[
\text{Bist}_1 = \text{Beta}(s_3 + 1:s_3 + \text{DimofSubsys}(i), s_4 + 1:s_4 + \text{DimofSubsys}(t));
\]

\[
s_5 = 0;
\]

for \( t_1 = 1:t - 1 \)

\[
s_5 = s_5 + \text{DimofSubsys}(t_1) \cdot K;
\]

end

\[
\text{nd} = K - t - (s - j);
\]

\[
s_5 = s_5 + (\text{nd} - 1) \cdot \text{DimofSubsys}(t);
\]

\[
\text{ztkt} = \text{Rx}(s_5 + 1:s_5 + \text{DimofSubsys}(t), :);
\]

\[
\text{Rx}(s_1 + 1:s_1 + \text{DimofSubsys}(i), :) = \ldots, \quad \text{Rx}(s_1 + 1:s_1 + \text{DimofSubsys}(i), :) + \text{Bist}_1 \cdot \text{ztkt};
\]

end

end

if \( j \geq K + 1 - i + 2 \)
for $p=K+1-(j-1):i-1$

\[s6=0; s8=0;\]

for $t1=1:p-1$

\[s6=s6+\text{DimofSubsys}(t1);\]
\[s8=s8+\text{DimofSubsys}(t1)*K;\]
end

for $q=K+1-p:j-1$

\[nl=K+j-(p+q);\]
\[s7=s2+(nl-(K+1-i))*\text{DimofSubsys}(i);\]
\[\text{lipq1}=-L(s7+1:s7+\text{DimofSubsys}(i),s6+1:s6+\text{DimofSubsys}(p));\]
\[s9=s8+(q-1)*\text{DimofSubsys}(p);\]
\[zpq=\text{Rx}(s9+1:s9+\text{DimofSubsys}(p),:);\]
\[\text{Rx}(s1+1:s1+\text{DimofSubsys}(i),:) = \ldots,\]
\[\text{Rx}(s1+1:s1+\text{DimofSubsys}(i),:)+\text{lipq1}*zpq;\]
end
end

end

if $(j=K+1-i)$

\[\text{Rx0}=\text{Rx}(s1+1:s1+\text{DimofSubsys}(i),:);\]
\[\text{Rx1}=\text{zeros}(\text{size}(%\text{Rx0}));\]
\[m1=K+1-i-1;\]

while $(m1>=1)$

\[s11=ss+(m1-1)*\text{DimofSubsys}(i);\]
for $m2=1:i$

107
s10 = 0;

for t1 = 1:m2 - 1
    s10 = s10 + DimofSubsys(t1) * K;
end

Rx1(:, s10 + 1:s10 + DimofSubsys(m2) * (K - 1)) = ...,
Rx0(:, s10 + 1 + DimofSubsys(m2):s10 + DimofSubsys(m2) * K);
end

Rx(s11 + 1:s11 + DimofSubsys(i), :) = Rx1;
Rx0 = Rx1;
ml = ml - 1;

end
end
end

end

% Transform Z. \{0\}
s1 = 0;
for t1 = 1:K
    s1 = s1 + DimofSubsys(t1) * K;
end

R0 = zeros(DimA0, s1);
A0 = AA(1:DimA0, :);
for i = 1:K
    s2 = 0; s1 = DimA0;

for \( t = 1: i - 1 \)
\[
s_1 = s_1 + \text{DimofSubsys}(t) \times t;
\]
\[
s_2 = s_2 + \text{DimofSubsys}(t) \times K;
\]
end
\[
nd = K + 1 - i;
\]
\[
s_2 = s_2 + (nd - 1) \times \text{DimofSubsys}(i);
\]
\[
A_{0i} = A_0(:, s_1 + 1: s_1 + \text{DimofSubsys}(i));
\]
\[
R_0 = R_0 + A_{0i} \times R_x(s_2 + 1: s_2 + \text{DimofSubsys}(i),:);
\]
end

\( R_{x0} = R_0; \)
\( A_{00} = A_0(:, 1: \text{DimA0}); \)

for \( i = 1: K \)
\[
s_1 = 0;
\]
for \( t = 1: i - 1 \)
\[
s_1 = s_1 + \text{DimofSubsys}(t) \times K;
\]
end
for \( j = 1: K \)
\[
s_2 = s_1 + (j - 1) \times \text{DimofSubsys}(i);
\]
\[
ss = 0;
\]
if \( j < K \)
for \( p = j + 1: K \)
\[
s_3 = s_1 + (p - 1) \times \text{DimofSubsys}(i);
\]
ss=ss+(A00°(p−j−1))∗Rx0(:,s3+1:s3+DimofSubsys(i));

end

Rx0(:,s2+1:s2+DimofSubsys(i))=ss;

end

end

%Get Aw0

s1=0;
for t=1:K
    s1=s1+DimofSubsys(t)*K;
end

Aw0=zeros(DimA0,DimA0+s1);

Aw0(:,1:DimA0)=A00;
for i=1:K
    s1=0;
    for t=1:i−1
        s1=s1+DimofSubsys(t)*K;
    end
    ss=0;
    for j=1:K
        s2=s1+(j−1)*DimofSubsys(i);

110
ss=ss+A0ˆ( j−1)∗R0(:,s2+1:s2+DimofSubsys(i));
end

Aw0(:,s1+DimA0+1:s1+DimA0+DimofSubsys(i))=ss;
end

G1=G;

% Update G
for i=1:K
  s1=0;
  for t=1:i−1
    s1=s1+DimofSubsys(t);
  end
  gi=G(s1+1:s1+DimofSubsys(i),:);
  m=i;
  [gibar]=updateGpost(m,gi,L,V,K,DimofSubsys,DimA0);
  G(s1+1:s1+DimofSubsys(i),:)=gibar;
end

% generate Gw
SG=size(G);
s1=SG(1);
SG=size(Rx);
s2=SG(2);
Gw=zeros(s1,s2);
for i=1:K
s1=0;
for t=1:i-1
    s1=s1+DimofSubsys(t);
end

gi=G(s1+1:s1+DimofSubsys(i),:);
[gw]=getGwpost(gi,Rx,Rx0,K,DimofSubsys,DimA0);
Gw(s1+1:s1+DimofSubsys(i),:)=gw;
end

Gw=[G(:,1:DimA0),Gw];
AG=[Aw0;Gw];

% Add zero rows to AG
m=size(Gw);
AW=zeros(m(2),m(2));
AW(1:DimA0,:)=AG(1:DimA0,:);

for i=1:K
    s1=DimA0; s2=DimA0;
    for m=1:i-1
        s1=s1+DimofSubsys(m)*K;
    end
    for m=1:i-1
        s2=s2+DimofSubsys(m);
    end
    AW(s1+1:DimofSubsys(i)*(K-1):s1+DimofSubsys(i)*K,:)=...;
    AG(s2+1:s2+DimofSubsys(i),:);
end
end

% Add identity matrix to AW

for i = 1:K
    s1 = DimA0;
    for m = 1:i-1
        s1 = s1 + DimofSubsys(m) * K;
    end
    ni = DimofSubsys(i);
    Ii1 = eye(ni * (K-1));
    zi1 = zeros(ni * (K-1), ni);
    Ii = [zi1, Ii1];
    AW(s1 + 1:s1 + ni * (K-1), s1 + 1:s1 + ni * K) = Ii;
end

% Move columns and rows to be a standard SCB form

% Move columns

j = 1;
t = DimA0;

while (j <= K)
    if (DimofSubsys(j) > 1)
        for i = 1:j-1
            t = t + DimofSubsys(i) * K;
        end
        for m = 1:DimofSubsys(j)

113
for i=1:K
    A3(:,K*(m-1)+i)=AW(:,t+m+(i-1)*DimofSubsys(j));
end
end

AW(:,t+1:t+DimofSubsys(j)*K)=A3;
end

j=j+1;
end

%Move rows
j=1;
t=DimA0;
while (j<=K)
    if (DimofSubsys(j)>1)
        for i=1:j-1
            t=t+DimofSubsys(i)*K;
        end
        for m=1:DimofSubsys(j)
            for i=1:K
                A4(K*(m-1)+i,:)=AW(t+m+(i-1)*DimofSubsys(j),:);
            end
        end
    end
    AW(t+1:t+DimofSubsys(j)*K,:)=A4;
end
j = j + 1;

end

end

% Remove zero coefficients added at the beginning
DimofSubsysM = DimofSubsys;
numofzero = 0;
if (Zerodim ~= 0)
    numofzero = max(size(Zerodim));
end
for i = 1: numofzero
    nsub = Zerodim(i);
    s1 = DimA0;
    for j = 1: nsub - 1
        s1 = s1 + DimofSubsys(j) * K;
    end
    AW(s1 + 1: s1 + K, :) = [];
    AW(:, s1 + 1: s1 + K) = [];
    DimofSubsys(nsub) = 0;
end

% Generate BW, CW, DW
m = size(AW);
t = 0;
for j=1:K
    t=t+dimofSubsys(j);
end
BW=zeros(m(1),t);
CW=zeros(t,m(1));
DW=zeros(t,t);
s=dimA0;
for j=1:t
    BW(s+K,j)=1;
    CW(j,s+1)=1;
    s=s+K;
end

% Generate Ac, Bc, Cc, Dc
s1=0;s2=0;
for t=1:K-1
    s1=s1+(K-t);
    s2=s2+dimofSubsys(t);
end
s2=s2+dimofSubsys(K);
Ac=zeros(s1,s1);
Bc=zeros(s1,s2);
Cc=zeros(s2,s1);
Dc=zeros(s2,s2);
%Ac

for  i=1:K-1
    s1=0;
    s2=0;
    for  t=1:i-1
        s1=s1+DimofSubsys(t)*(K-t);
        s2=s2+DimofSubsysM(t)*K;
    end
    if ( DimofSubsys(i) )
        r_{ij}=zeros(DimofSubsys(i),s1+DimofSubsys(i)*(K-i));
        t1=(K-i-1)*DimofSubsys(i);
        Ac(s1+1:s1+t1,s1+DimofSubsys(i)+1:s1+DimofSubsys(i)+t1)=eye(t1);
        nd=K+1-i;
        s3=s2+(nd-1)*DimofSubsys(i);
        Rxr=Rx(s3+1:s3+DimofSubsys(i),:);
        for  j=1:i
            s4=0;
            s5=0;
            for  t=1:j-1
                s4=s4+DimofSubsysM(t)*K;
                s5=s5+DimofSubsys(t)*(K-t);
            end
    end
if (DimofSubsys(j))
    rij ( :, s5 +1:s5+DimofSubsys(j) *(K−j)) = ...,
    −Rxr ( :, s4 +1:s4+(K−j) * DimofSubsys(j));
end
end

Ac(s1+t1 +1:s1+t1+DimofSubsys(i) , 1:s1+DimofSubsys(i)+t1) = r ij ;
end
end

%Bc, Cc

for i = 1:K−1
    s1 = 0; s2 = 0;
    for t = 1:i−1
        s1 = s1 + DimofSubsys(t) *(K−t);
        s2 = s2 + DimofSubsys(t);
    end
    if (DimofSubsys(i))
        s3 = s1 + (K−i−1) * DimofSubsys(i);
        Bc(s3 +1:s3+DimofSubsys(i) , s2 +1:s2+DimofSubsys(i)) = eye(DimofSubsys(i));
        Cc(s2 +1:s2+DimofSubsys(i) , s1 +1:s1+DimofSubsys(i)) = eye(DimofSubsys(i));
    end
end
%Dc
s1 = 0;
for t = 1:K - 1
    s1 = s1 + DimofSubsys(t);
end
Dc(s1 + 1:s1 + DimofSubsys(K), s1 + 1:s1 + DimofSubsys(K)) = eye(DimofSubsys(K));

%Return the updated dim vector
dim = dims;
t = 0;
for i = 1:K
    t = t + DimofSubsys(i);
end
dim(6) = t * K;
qdim = size(1, t);
for j = 1:t
    qdim(j) = K;
end

5.2.2 updateGpost

function [gibar] = updateGpost(m, gi, L, V, K, DimofSubsys, DimA0)
%This is a function used to update garbage matrix G

gibar = gi;
sL = 0;
for t = 2:m - 1
\[ s_L = s_L + \text{DimofSubsys}(t)*(t-1); \]

end

for \( i = 1:K \)

\[ s_1 = \text{DimA}0; s_6 = 0; \]

for \( t = 1:i-1 \)

\[ s_1 = s_1 + \text{DimofSubsys}(t)*t; \]

\[ s_6 = s_6 + \text{DimofSubsys}(t); \]

end

for \( j = 1:i \)

\[ s_2 = s_1 + (j-1)*\text{DimofSubsys}(i); \]

\[ a_1 = \text{gi}(:, s_2+1:s_2+\text{DimofSubsys}(i)); \]

\[ a_2 = 0; \]

\[ a_3 = 0; \]

if \( j == 1 \)

for \( p = i+1:m-1 \)

\[ s_3 = 0; s_5 = 0; \]

for \( t = 1:p-1 \)

\[ s_3 = s_3 + \text{DimofSubsys}(t); \]

\[ s_5 = s_5 + \text{DimofSubsys}(t)*(t-1); \]

end

for \( q = K+1-p:K-1 \)

\[ s_4 = s_L + (q-(K+1-m))*\text{DimofSubsys}(m); \]

end

end
\[ \text{lmpq} = L(s_4 + 1:s_4 + \text{DimSubsys}(m), s_3 + 1:s_3 + \text{DimSubsys}(p)) \]
\[ s_7 = s_5 + (2K - p - q - (K+1 - p)) \times \text{DimSubsys}(p) \]
\[ \text{lphi} = V(s_7 + 1:s_7 + \text{DimSubsys}(p), s_6 + 1:s_6 + \text{DimSubsys}(i)) \]
\[ a_3 = a_3 + \text{lmpq} \times \text{lphi} \]
\end{verbatim}
\begin{verbatim}
end
end
end
if (j > 1)
\[ s_8 = s_L + (K+1 - j - (K+1 - m)) \times \text{DimSubsys}(m) \]
if (i < m)
\[ a_2 = L(s_8 + 1:s_8 + \text{DimSubsys}(m), s_6 + 1:s_6 + \text{DimSubsys}(i)) \]
end
end
gibar(:, s_2 + 1:s_2 + \text{DimSubsys}(i)) = a_1 + a_2 + a_3 ;
end
end

5.2.3 getGwpost

function \([gw] = \text{getGwpost}(\text{gi}, R_x, R_x0, K, \text{DimSubsys}, \text{DimA0})\)

% This is a function to generate garbage matrix Gw in uniform rank system
\[
\$ \Sigma \{w_i\}$
\]

am0 = gi(:, 1:DimA0);
gw = am0 * R_x0;

121
for i = 1:K
    ss = DimA0;
    sr = 0;
    for t = 1:i - 1
        ss = ss + DimofSubsys(t) * t;
        sr = sr + DimofSubsys(t) * K;
    end
    for j = 1:i
        s1 = ss + (j - 1) * DimofSubsys(i);
        amij = gi(:, s1 + 1:1 + DimofSubsys(i));
        nd = K + j - i;
        s2 = sr + (nd - 1) * DimofSubsys(i);
        zikj = Rx(s2 + 1:s2 + DimofSubsys(i), :);
        gw = gw + amij * zikj;
    end
end

Subfunction [zzscb] is the same one called in main function [rankuniform_pre].
Chapter 6

Conclusions

Design of a pre-compensator or a post-compensator or both pre- and post-compensators is pursued to square down a general MIMO system to a uniform rank system, the structure of which is almost similar to a SISO system. The significance of such a squaring down lies in the simplicity of the structure of a uniform rank system lending itself for easy analysis and control design. This simplicity is transparent in several applications including decentralized control, and synchronization of multi-agent systems where the agents are non-introspective.
Bibliography


