Classic Gain Scheduling

Overview

Gain scheduling has been one of the dominant control strategies for the design of high performance controllers in the industry for the last fifty years at least. The gain scheduling practice can be roughly divided into two basic categories: *linearization-based* (or classic) and *LPV* (or modern) gain scheduling. In this chapter the first category is detailed whereas in Chapter 2 some classic results on the second category are mentioned. The *classic* method can be also divided into two subcategories: approaches that do offer some stability guaranties for the gain-scheduled system and others that do not. This chapter begins with a general introduction to the adaptive control framework (encompassing also gain scheduling) and then proceeds to a detailed citation of all classic gain scheduling methods existing in the bibliography. At the end of the chapter some additional tools used in the gain scheduling context are also detailed.

Chapter contents

1.1 Adaptive Control Schemes

Introduction

Adaptive control has risen due to the need of changing/updating afeedback controller K in order to conform to the changing parameters of a process $\mathcal{S}$. As a simple example consider the dynamics of an aircraft: this type of systems operate in different altitudes and with different speeds and thus due to several physical reasons their dynamics change drastically as a function of time. A *robust* controller designed to cope with the different operating conditions cannot always guarantee, or at least offer some good indications, that the aircraft will behave in a good way for all altitudes and speeds of its flight envelope.

To solve this problem an adaptive control system may be used in order to update the controller parameters for changing operating conditions. Three basic types of adaptive control systems exist (see Figure 1.1):

Indirect Adaptive Control

Indirect Adaptive Control (IAC): In this control scheme (see Fig. 1.1a) the controller parameters (or gains) ϑ_{c} are updated in real-time by an auto-tuner. This auto-tuner is based on an identified process model $\hat{\mathcal{S}}$ provided by an estimator that uses I/O plant information. This auto-tuner then calculates ϑ_{c} as if $\hat{\mathcal{S}} = \mathcal{S}$. The control scheme has two feedback loops: an internal loop that is fast enough to control the plant and an external one that is slower and detects any potential changes in the system's model through an estimator. An example of an IAC scheme is adaptive pole placement control: the poles of the closed loop plant are assigned in real-time to a specified location on the complex plane based on the estimate of $\mathcal{S}$ and on a given controller structure (e.g. PID).

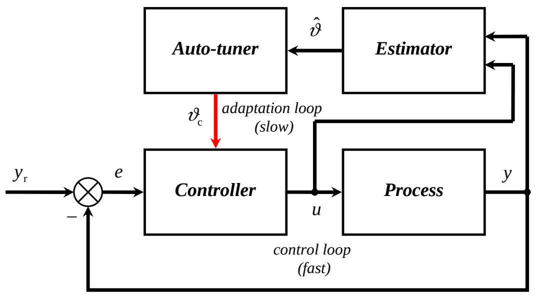
Direct Adaptive Control

Direct Adaptive Control (DAC): In this control scheme the controller parameters ϑ_{c} are estimated directly and the use of a plant parameter estimator is not needed. Take for example a frequently used topology of DAC: the direct Model Reference Adaptive Control (d-MRAC) configuration of Fig. 1.1b. The auto-tuner here computes the difference e_{d} between the outputs of the real plant $\mathcal{S}$ and of a *target* plant model $\bar{\mathcal{S}}$ and tries to find a value for ϑ_{c} so that this difference goes to zero. A way to do that is the famous *MIT* rule [15, 66].

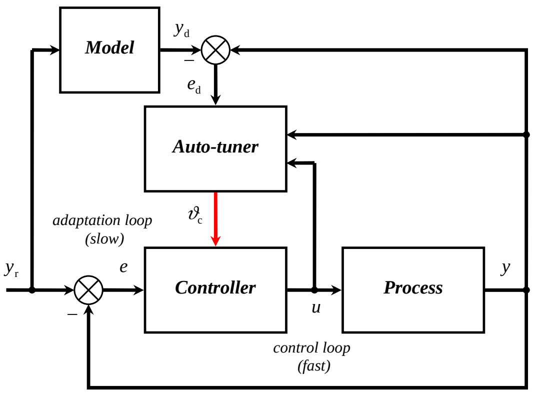
Gain Scheduling Control

Gain Scheduling Control (GSC): In this control scheme (see Fig. 1.1c) no complex algorithm is demanded for updating the controller parameters but only a parameter (or scheduling) vector ϱ (that can sufficiently capture the plant's change of dynamics) and an interpolation method. The controller parameters ϑ_{c} are then updated by combining/interpolating different controllers K^i designed for the plant $\mathcal{S}$, for some family of *critical* values of ϱ[1]. The simplest form of GSC is *controller switching* where no smooth controller parameter update is performed and a single controller is used, being valid for a pre-defined operating region over ϱ.

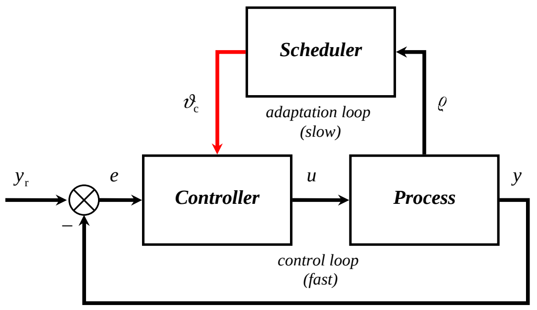
[1]For the aircraft example considered before, this vector ϱ could be the Mach and the altitude.



(a) *Indirect adaptive control configuration*



(b) *Direct adaptive control configuration*



(c) *Gain scheduling control configuration*

Figure 1.1: Various adaptive control schemes.

LPV vs. LBGS

In this monograph the latter method will be considered for the control of generic nonlinear parameter/time dependent systems. The gain scheduling control practice can be further divided into two major categories: the *linearization-based* and *LPV/q-LPV* gain scheduling procedures. The major distinction between these two has to do on the one hand with the approach taken in order to obtain the final nonlinear gain-scheduled controller, and on the other hand on the way that the system nonlinearity is treated. These two methods sometimes overlap and there exist a considerable disagreement over the scientific community on which one is the best suited for a particular problem.

The linearization-based gain scheduling procedure (LBGS)[2] is mostly based on linearized plants of the initial nonlinear system, calculates a number of controllers of possibly, not the same structure, and finally interpolates them in order to obtain the gain-scheduled controller. The existence of a controller is (almost) always guaranteed but stability issues arise due to the ad-hoc linearization-interpolation. There exist however some notable exceptions that they do consider stability for the linearized scheduled system, but obviously not on the initial nonlinear one. In this chapter both types of methods will be discussed and some key results as well as references to real world applications will be given.

The LPV procedure tries to camouflage the nonlinear dynamics and obtain thus a linear system with time varying state space matrices. These time varying matrices can be treated either as time varying uncertainty thus leading to the so-called *LFT* formulation, or as parameter-dependent matrices that may form convex hyper-cubes for frozen values of the parameter leading to the *Polytopic* formulation. In both cases there exist stability guarantees for the overall scheduled system. However, the fact that is not clear enough (see [88], pp. 1012) is for which system the stability guarantees are offered[3].

Briefly it can be said that the first class of methods offers a systematic and unconservative design methodology that provides always a controller whereas the second offers a more theoretically sound, yet sometimes conservative in terms of system operation & controller existence, procedure that guarantees global stability of the gain-scheduled plant. In this work the first class of methods will be used and several of its problems addressed.

Contents

In this chapter the first class of methods (*classic*) is extensively detailed whereas in the next one the second class ones (*modern*) are briefly reviewed. The following section (Section 1.2) considers some general results in system modeling whereas the next one (Section 1.3.1) details the LBGS following the famous five (2+3) step procedure (see [88]).

Finally, subsections 1.3.2, 1.3.3 consider both the ad-hoc and stability preserving methods existing in the bibliography whereas Section 1.4 presents some related to the gain scheduling practice results concerning interpolation and operating domain triangulation.

[2]Also called classic or divide and conquer method (see [88], pp. 1005-1008).

[3]For more details see Chapter 2 being also based on the analysis of the next section.

1.2 System Modeling

In this section general modeling issues in the context of gain scheduling are reviewed. Some system equilibrium notions are initially introduced before passing to a citation of various ways to model a physical process. Finally, some material on Jacobian linearization is covered.

1.2.1 Equilibrium Notions

System modeling

Consider a generic non-autonomous[4] forced nonlinear dynamic system $\mathcal{S}$ whose state and output dynamics are described by a number of coupled first-order differential equations (see Fig. 1.2):

$$\mathcal{S}: \quad \begin{aligned} \dot{x}(t) &= \mathbf{f}[x(t), u(t), t] \\ y(t) &= \mathbf{h}[x(t), u(t), t]. \end{aligned} \tag{1.1}$$

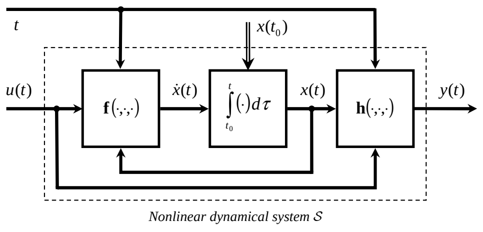
The vectors x, u, y represent the *states*, *inputs* and *outputs* of the system with $x \in \mathbb{R}^n, u \in \mathbb{R}^{n_u}, y \in \mathbb{R}^{n_y}$ respectively[5]. The vector-valued functions $\mathbf{f}, \mathbf{h}$ where $\mathbf{f} := \big[f_1(x,u), \dots, f_n(x,u)\big]^T$ and $\mathbf{h} := \big[h_1(x,u), \dots, h_{n_y}(x,u)\big]^T$ perform the following nonlinear mappings:

$$\mathbf{f}: \quad \mathbb{R}^n \times \mathbb{R}^{n_u} \times \mathbb{R} \mapsto \mathbb{R}^n \tag{1.2}$$

$$\mathbf{h}: \quad \mathbb{R}^n \times \mathbb{R}^{n_u} \times \mathbb{R} \mapsto \mathbb{R}^{n_y}. \tag{1.3}$$

The nonlinear system $\mathcal{S}$ in fact is a mathematical representation of a physical process and thus for $\mathcal{S}$ to provide a valid reproduction of its behavior, several additional hypotheses need to be made. These hypotheses are mostly related to the existence and uniqueness of a solution $x\big(t; t_0, x(t_0)\big)$ given a set of initial state conditions $x(t_0)$ and the differentiability of the functions $\mathbf{f}, \mathbf{h}$ with respect to an equilibrium point or trajectory (see [75], Ch. 3 for more details).

The analysis concerning equilibrium notions in the next two subsections considers both autonomous nonlinear and linear systems and their non-autonomous extensions. Another extension is also given for parameter-varying systems used mostly to model processes controlled by gain-scheduled control schemes.



t
$x(t_0)$
$u(t)$
$\mathbf{f}(\cdot,\cdot,\cdot)$
$\dot{x}(t)$
$\int_{t_0}^{t} (\cdot) d\tau$
$x(t)$
$\mathbf{h}(\cdot,\cdot,\cdot)$
$y(t)$
Nonlinear dynamical system $\mathcal{S}$

Figure 1.2: System block diagram.

[4] *Time-invariant /autonomous* are equivalent as are *time-varying/non-autonomous*.
[5] Dependence on t will be dropped when needed and a derivative is taken with respect to t.

1.2.1.1 Autonomous Systems

Unforced case

To introduce the notion of an equilibrium point it would be simpler to consider first an *unforced* nonlinear autonomous system $\mathcal{S}_a$ whose state dynamics are described by:

$$\mathcal{S}_a : \quad \dot{x} = \mathbf{f}(x). \tag{1.4}$$

An equilibrium state x_{eq} for this system $\mathcal{S}_a$ is defined as the point in the state space from which when the state starts it never leaves, for every $t \geq t_0, t_0 > 0$. The condition that defines such a state is written as:

$$x_{\text{eq}} : \quad \left.\frac{dx(t)}{dt}\right|_{\text{eq}} = \dot{x}_{\text{eq}} \stackrel{\Delta}{=} 0. \tag{1.5}$$

Equilibrium condition

The above condition means briefly that $x\big(t; t_0, x(t_0)\big) = x(t_0) = x_{\text{eq}}$ and as a consequence from Eqs. 1.4, 1.5:

$$\mathbf{f}(x_{\text{eq}}) \stackrel{\Delta}{=} 0. \tag{1.6}$$

Now in order to find the equilibrium points of such a system, a collection of n coupled algebraic equations (given by Eq. 1.6) should be solved. These equations may yield a finite or even an infinite number of equilibrium points, depending on their structure. When studying the stability of the system $\mathcal{S}_a$, using for example the Lyapunov's stability theory, it may be useful to study the stability of the state vector at the origin. This can be done by translating all its equilibrium points via a change of variables; indeed, suppose a $x_{\text{eq}} \neq 0$ and the change of variables $z = x - x_{\text{eq}}$. Then:

$$\begin{aligned} \dot{z} &= \dot{x} - \dot{x}_{\text{eq}} \\ &= \mathbf{f}(x) - 0 \\ &= \mathbf{f}(z + x_{\text{eq}}) \stackrel{\Delta}{=} \mathbf{g}(z). \end{aligned} \tag{1.7}$$

The function $\mathbf{g}(\cdot)$ does not depend *explicitly* on time and thus the unforced nonlinear equivalent system $\dot{z} = \mathbf{g}(z)$ is also autonomous with an equilibrium point at the origin $z = 0$.

Forced case

Consider now the case where the autonomous nonlinear system is *forced*, i.e. it's dynamics are described by:

$$\mathcal{S}_{a,f} : \quad \dot{x} = \mathbf{f}(x, u). \tag{1.8}$$

Equilibrium manifold

Then non-zero equilibrium states x_{eq} could be now *imposed* by using a constant corresponding equilibrium control input vector $u_{\text{eq}} = \mathbf{u}(x_{\text{eq}})$, that will maintain the state to its equilibrium value for all $t \geq t_0, t > 0$. The system's *equilibrium manifold* $\mathcal{E}_{a,f}$ is defined as the set of all the admissible states/inputs for which the right-hand side of Eq. 1.8 may go to zero:

$$\mathcal{E}_{a,f} : \quad \big\{(x_{\text{eq}}, u_{\text{eq}}) | \mathbf{f}(x_{\text{eq}}, u_{\text{eq}}) = 0\big\}. \tag{1.9}$$

Output condition

To the equilibrium manifold $\mathcal{E}_{a,f}$, corresponds also an equilibrium value y_{eq} for the output of the nonlinear autonomous system[6]:

$$y_{\text{eq}} = \mathbf{h}(x_{\text{eq}}, u_{\text{eq}}). \tag{1.10}$$

Once again, the equilibrium points of this *forced* system $\mathcal{S}_{a,f}$ may be translated to the origin for an *unforced* equivalent system. To illustrate this, consider the change of variables $z = x - x_{\text{eq}}, v = u - u_{\text{eq}}$. Then:

$$\begin{aligned}\dot{z} &= \dot{x} - \dot{x}_{\text{eq}} \\ &= \mathbf{f}(x, u) - 0 \\ &= \mathbf{f}(z + x_{\text{eq}}, v + u_{\text{eq}}) \triangleq \mathbf{g}(z, v).\end{aligned} \tag{1.11}$$

The right hand side of Eq. 1.10 does not depend explicitly on time and thus the system described by the transformed equation $\mathbf{g}(z, v)$ admits an equilibrium point at its origin since by definition $\mathbf{f}(x_{\text{eq}}, u_{\text{eq}}) = 0$. So because of the fact that in this case $v = 0$, the transformed plant is now *unforced* and once again the analysis ends up to the study of a system like the one in Eq. 1.7 around $z = 0$.

LTI case

In the case of a *linear time-invariant* (LTI) dynamical system having the following state space representation (with $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{B} \in \mathbb{R}^{n\times n_u}, \mathbf{C} \in \mathbb{R}^{n_y\times n}, \mathbf{D} \in \mathbb{R}^{n_y\times n_u}$):

$$\mathcal{S}_{\text{LTI}} : \begin{aligned}\dot{x} &= \mathbf{A}x + \mathbf{B}u \\ y &= \mathbf{C}x + \mathbf{D}u\end{aligned} \tag{1.12}$$

the things are simple; the origin is *always* an equilibrium state for the unforced system whereas for the *forced* one, under certain controllability conditions, one may be able to maintain the state to a given equilibrium value x_{eq} using a suitable equilibrium (or *open loop*) control u_{eq} that satisfies:

$$\mathbf{A}x_{\text{eq}} + \mathbf{B}u_{\text{eq}} = 0 \tag{1.13}$$

with a corresponding equilibrium output:

$$y_{\text{eq}} = \mathbf{C}x_{\text{eq}} + \mathbf{D}u_{\text{eq}}. \tag{1.14}$$

Summarizing, the equilibrium points of any linear or nonlinear, forced or unforced autonomous system may be translated to the origin with the resulting system being also autonomous. However, as it will be shown in the next section, the resulting equivalent system $\mathbf{g}(\cdot)$ for an equilibrium *trajectory* $x_{\text{eq}}(t)$ that is a solution to the initial *autonomous* nonlinear system $\mathcal{S}_a$ and satisfying:

$$\dot{x}_{\text{eq}}(t) = \mathbf{f}[x_{\text{eq}}(t)] \tag{1.15}$$

will be *non-autonomous* even when the initial system is.

[6]For the autonomous *unforced* system the output equilibrium value is given by $y_{\text{eq}} = \mathbf{h}(x_{\text{eq}})$.

1.2.1.2 Non-autonomous Systems

The state dynamics of the *unforced* nonlinear non-autonomous system $\mathcal{S}_{na}$ are:

Unforced case

$$\mathcal{S}_{na}: \quad \dot{x} = \mathbf{f}(x,t). \tag{1.16}$$

This system has an equilibrium point at $t = 0$, if $\mathbf{f}(0,t) = 0, \forall t \geq 0$ (any equilibrium point at $t = 0$ is also an equilibrium point at all times; see for example [142], pp. 5). In addition, any constant non-zero equilibrium point can be translated to the origin (for $t = 0$) with the same procedure as in Section 1.2.1.1, both for the forced and unforced cases. In fact this may be done not only for a constant equilibrium point but also on a nonzero system equilibrium trajectory $x_{\text{eq}}(t)$. Indeed, consider the change of variables $z = z(t) = x - x_{\text{eq}}(t)$; then the time derivative of z for $\mathcal{S}_{na}$ will be:

$$\begin{aligned}\dot{z} &= \dot{x} - \dot{x}_{\text{eq}}(t)\\ &= \mathbf{f}(x,t) - \dot{x}_{\text{eq}}(t)\\ &= \mathbf{f}[z + x_{\text{eq}}(t), t] - \dot{x}_{\text{eq}}(t) \overset{\Delta}{=} \mathbf{g}(z,t).\end{aligned} \tag{1.17}$$

Thus the equivalent system described by $\mathbf{g}(\cdot)$ has an equilibrium at the origin for $t = 0$ like the one in Eq. 1.16. From the above analysis it is also evident that even if the initial state dynamics are autonomous, the equivalent system $\mathbf{g}(\cdot)$ is non-autonomous in the case of an equilibrium trajectory since the transformed variable z depends also *explicitly* on time due to $x_{\text{eq}}(t)$.

Forced case

The equilibrium analysis for the *forced* non-autonomous system $\mathcal{S}$ of Eq. 1.1 is related to its control as in the analysis of the previous section concerning this case (see Eq. 1.11). The control to maintain the system $\mathcal{S}$ to a *given* equilibrium state x_{eq} is time-varying and is composed by a steady-state value u_{eq} (translating the state to its steady-state value) plus a time-varying one u_δ that regulates the time-varying system around the origin (see [75], pp. 469-471). The analysis for the linear time-varying (LTV) version (see Eq. 1.18) is also similar.

$$\mathcal{S}_{\text{LTV}}: \quad \begin{aligned}\dot{x} &= \mathbf{A}(t)x + \mathbf{B}(t)u\\ y &= \mathbf{C}(t)x + \mathbf{D}(t)u\end{aligned} \tag{1.18}$$

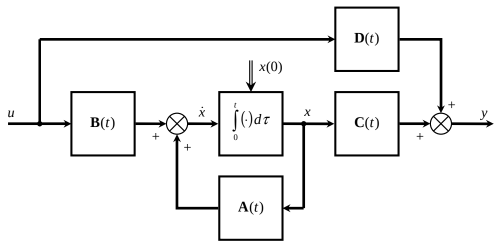


Figure 1.3: LTV system block diagram.

1.2.1.3 Parameter-dependent Systems

A certain class of systems is characterized by a dependence of their dynamics on a time-varying vector of parameters $\varrho = \varrho(t)$. This parameter vector is often called the scheduling vector and it is assumed that *it can be measured in real time* for gain-scheduled systems.

NLPD system

These systems are also called *nonlinear parameter-dependent systems* (NLPD) and their dynamics are given by:

$$\mathcal{S}_{\text{pd}}: \quad \begin{aligned} \dot{x} &= \mathbf{f}(x, u, \varrho) \\ y &= \mathbf{h}(x, u, \varrho). \end{aligned} \tag{1.19}$$

Equilibrium manifold

Leaving other modeling details for the next section, such a system representation has a meaning if an equilibrium manifold $\mathcal{E}_{\text{pd}}$ is defined and *smoothly* parameterized as a function of the scheduling vector ϱ:

$$\mathcal{E}_{\text{pd}}: \quad \Big\{ \big[x(\varrho_{\text{eq}}), u(\varrho_{\text{eq}})\big] \big| \mathbf{f}\big[x(\varrho_{\text{eq}}), u(\varrho_{\text{eq}}), \varrho_{\text{eq}}\big] = 0 \Big\}. \tag{1.20}$$

To illustrate this consider for example the trivial case of a first order SISO system with a single time-varying parameter. The equilibrium point locus may look like the one visualized in Fig. 1.4; an equilibrium state x_{eq} and a corresponding equilibrium input u_{eq} are assigned for any admissible value of the scheduling parameter ϱ.

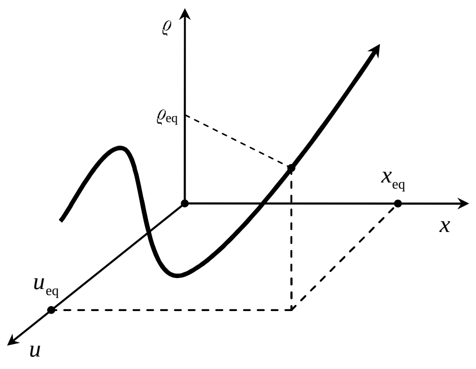


Figure 1.4: Equilibrium point locus.

Output manifold

For each value of the scheduling vector ϱ_{eq} the corresponding equilibrium output may also be defined as:

$$y_{\text{eq}} = \mathbf{h}\big[x(\varrho_{\text{eq}}), u(\varrho_{\text{eq}}), \varrho_{\text{eq}}\big]. \tag{1.21}$$

Technical Note. The computation of the equilibrium manifold of a parameter-dependent nonlinear system is by no means a trivial problem and is mainly done either by solving directly the algebraic equation $\mathbf{f}(x, u, \varrho) = 0$ for every admissible ϱ (if possible), or by numerical iterative optimization techniques (see for example the function '`findop`' of the MATLAB® Simulink Control Design toolbox).

1.2.2 System Descriptions

In this section the discussion will involve around various types of systems involved in gain-scheduled control schemes. The discussion necessitates the analysis of Section 1.2.1 on equilibrium points since for the scope of gain-scheduled control a process is often studied around such points. Details on linearization are given in the next subsection.

NLPD system

A physical process is usually modeled using a collection of nonlinear first order differential equations representing its *state dynamics* along with a second set of nonlinear algebraic equations describing its *output dynamics*. These modeling equations are often dependent to a number of external or internal variables that are regarded as parameters of the process. This parameter vector is called the *scheduling vector* $\varrho = \varrho(t)$ and gives a time-varying sense to such systems. This modeling results to the following equations for this *nonlinear parameter-dependent* system (or NLPD):[7]

$$\mathcal{S}_{\text{pd}} : \quad \begin{aligned} \dot{x} &= \mathbf{f}(x, u, \varrho) \\ y &= \mathbf{h}(x, u, \varrho). \end{aligned} \tag{1.22}$$

Besides the usual properties for the state $x(t)$, input $u(t)$, output $y(t)$ and for the functions $\mathbf{f}, \mathbf{h}$ discussed in the beginning of Section 1.2.1, additional hypotheses are made for the scheduling vector ϱ. Specifically it is assumed that $\varrho \in \mathbf{\Gamma}$, where $\mathbf{\Gamma}$ is a connected compact set with $\mathbf{\Gamma} \subset \mathbb{R}^{n_\varrho}$.

Operating domain discussion

Remark. A set is said to be *connected* if it is impossible to express it as the union of two or more disjoint open subsets. For example the set $[0, 1]$ is connected whereas the union of the sets $[0, 0.5), (0.5, 1]$ is disconnected. In fact any convex set is connected. Moreover a set is said to be *compact* if it is closed and bounded. For example the set $[0, 1]$ is closed but the sets $(0, 1), (0, 1]$ are not closed. In addition all these sets are bounded since they have finite sizes (see [19] for more details on set theory).
The above assumptions on ϱ are quite logical since in physical systems most variables take values on closed, finite and sometimes convex intervals. For example the Mach number of a missile takes values between a minimum and a maximum value; the same holds for the control inputs of a system which are bounded or a varying resistor in an electrical network.

Alternative formulation

An extended method to model a parameter-dependent system used mainly for output tracking adopts a linear robust control-type notation (see [79, 80, 128, 129, 131, 133] or even [75], pp. 474-475):

$$\mathcal{S}^*_{\text{pd}} : \quad \begin{aligned} \dot{x} &= \mathbf{f}(x, u, w) \\ \zeta &= \mathbf{h}_\zeta(x, u, w) \\ y &= \mathbf{h}_y(x, u, w). \end{aligned} \tag{1.23}$$

[7]Note that the explicit dependence on time is omitted but is assumed because $\varrho = \varrho(t)$. Many authors prefer to omit also the explicit dependence on the scheduling vector.

In this notation the signals $w(t)$ are external perturbations whereas $\zeta(t)$ are errors to be minimized (or signals to be treated) and $\mathbf{h}_\zeta, \mathbf{h}_y$ are the corresponding nonlinear algebraic functions with suitable dimensions. The scheduling vector ϱ may or may not appear directly[8] but it is always assumed to exist and defines a smooth equilibrium manifold as detailed in Section 1.2.1.3.

q-LPV system

Return now to the initial parameter-dependent system of Eq. 1.22. Performing state space transformations it is sometimes possible to transform the state/output dynamics so that an equivalent *quasi-linear* parameter-varying system $S_{q-\text{LPV}}$ may appear (with σ being now the measured parameter vector):

$$\mathcal{S}_{\text{q}-\text{LPV}}: \quad \begin{aligned} \dot{x} &= \mathbf{A}(x,u,\sigma)x + \mathbf{B}(x,u,\sigma)u \\ y &= \mathbf{C}(x,u,\sigma)x + \mathbf{D}(x,u,\sigma)u \end{aligned} \tag{1.24}$$

A similar modeling in q-LPV form is when the state x is divided into two parts: the part that is regarded as a parameter x_ϱ and the part that keeps its state variable notion $x^\star$. Then the final scheduling variable ϱ is consisted of the parameter-varying variable σ and of $x^\star$ (see [114], §3.2, pp. 1407).

LPV system

The (nonlinear) dependence of the system matrices on the state, input & measured parameter may also be regarded as a general time-varying parameter vector $\varrho = \varrho(x,u,\sigma)$. The trajectories of the scheduling vector ϱ are considered to be measured in real time. In this case a *linear* parameter-varying system is obtained:

$$\mathcal{S}_{\text{LPV}}: \quad \begin{aligned} \dot{x} &= \mathbf{A}(\varrho)x + \mathbf{B}(\varrho)u \\ y &= \mathbf{C}(\varrho)x + \mathbf{D}(\varrho)u. \end{aligned} \tag{1.25}$$

A delicate issue arises here however: if the scheduling vector is considered to be a function of the state also, then the gain-scheduled controller is assumed to be a state feedback one [9]. As a result, it is preferable to use the *output* rather than the *state* to parameterize the system. Hence, the scheduling vector and the LPV dynamics are dependent directly on the output, the parameter vector σ, and possibly on the input (see [114], §3.2, pp. 1408-09, [120]).

Discussion on LPV systems

The solutions of the nonlinear system (and hence of the q-LPV one) are also solutions of the LPV formulation in Eq. 1.25 and thus the former is over-bounded by the latter (see [88], pp. 1012). This modeling adds some conservativeness but for gain scheduling control it may taken as a basis for controller design. Both classic/modern gain scheduling tools consider these types of models (i.e. see the fundamental work of [13, 18])[10] but for the former there exists a major difference: the LPV models used with classic (or linearization-based) gain scheduling are only valid close to equilibrium points and do not directly describe the behavior of the initial parameter-dependent nonlinear system $\mathcal{S}_{\text{pd}}$ of Eq. 1.22.

[8]In some cases (see [79, 80]) the scheduling vector appears indirectly but is assumed to be a nonlinear function of the controller input and of measured external or signals. In other cases (see [128, 129, 131, 133]) it is considered as a parameter and does not appear directly.

[9]The same holds for the equilibrium manifold being also a function of the state.

[10]Modern gain scheduling design tools will be considered in the next chapter.

More precisely, such linearization-based LPV models are of the form[11]:

$$\mathcal{S}_{\text{LPV}}: \quad \begin{aligned} \dot{x}_\delta &= \mathbf{A}(\varrho)x_\delta + \mathbf{B}(\varrho)u_\delta \\ y_\delta &= \mathbf{C}(\varrho)x_\delta + \mathbf{D}(\varrho)u_\delta \end{aligned} \tag{1.26}$$

where:

$$x_\delta = x - x_{\text{eq}}(\varrho) \tag{1.27}$$
$$u_\delta = u - u_{\text{eq}}(\varrho) \tag{1.28}$$
$$y_\delta = y - y_{\text{eq}}(\varrho). \tag{1.29}$$

The distinction between the two LPV models of Eqs. 1.25, 1.26 is now clear: the first is a 'superset' of the initial nonlinear system with x, u, y being its actual variables whereas the second is a family of linear(ized) systems permitting only local description (around equilibrium points) of the nonlinear system at best.

A special case of the LPV modeling of Eq. 1.25 is the so-called LFT-based approach. This approach treats the LPV model as an LTI one with all time-varying parameters ϱ being regrouped as (measurable) uncertainties. This approach is also used in modern gain scheduling methods and it assumes that the plant $\mathcal{S}_{\text{LPV}}$ of Eq. 1.25 may be rewritten as the upper LFT (u-LFT) of an LTI standard plant $\mathbb{P}(s)$ (following a robust control-type modeling of the parameter-dependent plant) and a time-varying block operator Θ, specifying how the scheduling vector components ϱ_i enter the LPV plant dynamics: `LFT case`

$$\begin{bmatrix} \zeta \\ y \end{bmatrix} = \mathcal{F}_u\big(\mathbb{P}(s), \Theta\big) \begin{bmatrix} w \\ u \end{bmatrix} \tag{1.30}$$

with variables ζ, w being error/external perturbations signals respectively and:

$$\Theta = \text{blockdiag}(\varrho_1 \mathbb{I}_{\varrho 1}, \ldots, \varrho_1 \mathbb{I}_{\varrho n_\varrho}) \tag{1.31}$$

and also:

$$\zeta_\theta = \Theta w_\theta \tag{1.32}$$

representing the I/O's of the uncertainty block (see Fig. 1.3).

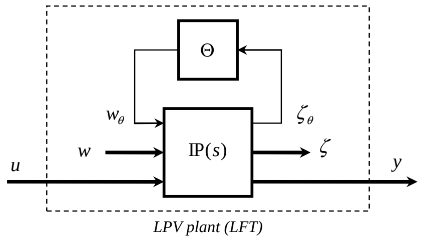


Figure 1.5: LFT description of an LPV system.

[11]More details on this formulation and linearization are given are given in the next section.

LTV systems

Another important class of systems are the linear time-varying (LTV) systems of the form (see Fig. 1.3):

$$\mathcal{S}_{\text{LTV}}: \quad \begin{aligned} \dot{x} &= \mathbf{A}(t)x + \mathbf{B}(t)u \\ y &= \mathbf{C}(t)x + \mathbf{D}(t)u \end{aligned} \tag{1.33}$$

These types of systems represent either a physical process directly, or (more often) they stem from the linearization of a nonlinear system (or also parameter-dependent one) around an *equilibrium trajectory* $x_{\text{eq}}(t), u_{\text{eq}}(t), y_{\text{eq}}(t)$ (for more details see the next section). In this case the LTV model is rather written as:

$$\mathcal{S}_{\text{LTV}}: \quad \begin{aligned} \dot{x}_\delta &= \mathbf{A}(t)x_\delta + \mathbf{B}(t)u_\delta \\ y_\delta &= \mathbf{C}(t)x_\delta + \mathbf{D}(t)u_\delta \end{aligned} \tag{1.34}$$

with:

$$x_\delta = x - x_{\text{eq}}(t) \tag{1.35}$$
$$u_\delta = u - u_{\text{eq}}(t) \tag{1.36}$$
$$y_\delta = y - y_{\text{eq}}(t). \tag{1.37}$$

LTI system

The previous model is valid in the vicinity of the equilibrium trajectory as is the following LTI system a valid approximation model of a nonlinear parameter-dependent system around an *equilibrium point*.

$$\mathcal{S}_{\text{LTI}}: \quad \begin{aligned} \dot{x}_\delta &= \mathbf{A}x_\delta + \mathbf{B}u_\delta \\ y_\delta &= \mathbf{C}x_\delta + \mathbf{D}u_\delta \end{aligned} \tag{1.38}$$

Discussion

The following figure shows an illustration of all the different ways to model a physical process. Starting from a nonlinear parameter-dependent (NLPD) model and going inwards to lesser degrees of complexity, one gets the simplest possible model which is a linear time invariant (LTI) one. An LPV model can be seen either as a conservative way to approximate a q-LPV or NLPD model or a more complex way to model a possibly time-varying LTI system. The last holds also for the LTV one that can be either seen as a linear approximation of a NLPD system around an equilibrium trajectory or a more 'realistic' way to model an LTI system. Finally, an LTI system is an approximation of a NLPD model around an equilibrium point.

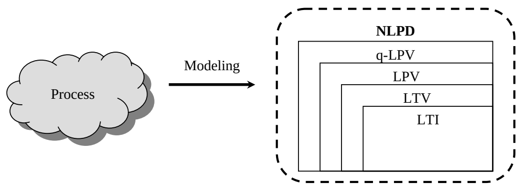


Figure 1.6: Process modeling.

1.2.3 Linearization Notions

In this section some results are presented concerning the approximation of nonlinear parameter-dependent systems by linearizing their dynamics around equilibrium points or trajectories. This section is also linked to the analysis of the two previous ones and offers the necessary material for the next section concerning linearization-based gain scheduling.

NLPD system

Suppose a given forced nonlinear parameter-dependent system $\mathcal{S}_{\text{pd}}$ is described by the following first-order differential equations:

$$\mathcal{S}_{\text{pd}}: \quad \begin{aligned} \dot{x} &= \mathbf{f}(x, u, \varrho) \\ y &= \mathbf{h}(x, u, \varrho). \end{aligned} \tag{1.39}$$

The functions $\mathbf{f}, \mathbf{h}$ with $\mathbf{f} := \big[f_1, f_2, \ldots, f_n\big]^T$ and $\mathbf{h} := \big[h_1, h_2, \ldots, h_{n_y}\big]^T$ perform the following nonlinear mappings on the state $x \in \mathbb{R}^n$, input $u \in \mathbb{R}^{n_u}$, output $y \in \mathbb{R}^{n_y}$ and scheduling vector $\varrho \in \mathbb{R}^{n_\varrho}$:

$$\mathbf{f}: \quad \mathbb{R}^n \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\varrho} \mapsto \mathbb{R}^n \tag{1.40}$$

$$\mathbf{h}: \quad \mathbb{R}^n \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\varrho} \mapsto \mathbb{R}^{n_y}. \tag{1.41}$$

As it has been detailed in Section 1.2.1.3, the scheduling vector $\varrho(t)$ defines an equilibrium manifold $\mathcal{E}_{\text{pd}}$ (see Eq. 1.20). This means that it spans the equilibrium points of the system inside its domain of operation $\mathbf{\Gamma}$, with $\mathbf{\Gamma}$ being a connected compact set (see the corresponding remark in Section 1.2.2).

Reformulation

The trajectory $x\big(t; t_0, x(t_0)\big)$ of $\mathcal{S}_{\text{pd}}$ may be *approximated* via the solution $\tilde{x}\big(t; t_0, x(t_0)\big)$ of a linearized model $\mathcal{S}_{\text{LTI}}$ in the close vicinity of an equilibrium point, defined for a constant (or frozen) value of the scheduling vector ϱ_{eq}. To obtain this approximation, reformulate first the dynamics of Eq. 1.39 (see [85], §2.1, pp. 291) as:

$$\dot{x}_\delta = \mathbf{A}(\varrho_{\text{eq}}) x_\delta + \mathbf{B}(\varrho_{\text{eq}}) u_\delta + \varepsilon_{\mathbf{f}} \tag{1.42}$$

$$y_\delta = \mathbf{C}(\varrho_{\text{eq}}) x_\delta + \mathbf{D}(\varrho_{\text{eq}}) u_\delta + \varepsilon_{\mathbf{h}} \tag{1.43}$$

Deviation quantities

where the errors $x_\delta, u_\delta, y_\delta$ are defined as:

$$x_\delta = x - x(\varrho_{\text{eq}}) \tag{1.44}$$

$$u_\delta = u - u(\varrho_{\text{eq}}) \tag{1.45}$$

$$y_\delta = y - y(\varrho_{\text{eq}}). \tag{1.46}$$

The matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are obtained by linearization (or first-order Taylor expansion) of the functions $\mathbf{f}, \mathbf{h}$ around the equilibrium point ϱ_{eq}[12] and having assumed that they have the appropriate differentiability properties.

[12]Even though the dependence of the matrices is shown to be only on ϱ_{eq} (see Eqs. 1.42, 1.43), it is assumed that there exist also a dependence on $x_{\text{eq}} = x(\varrho_{\text{eq}}), u_{\text{eq}} = u(\varrho_{\text{eq}})$. However it is omitted for notational simplicity.

System matrix computation

These matrices are computed as:

$$\mathbf{A}(\varrho_{\text{eq}}) = \nabla_x \mathbf{f}\Big|_{\varrho_{\text{eq}}} \tag{1.47}$$
$$\mathbf{B}(\varrho_{\text{eq}}) = \nabla_u \mathbf{f}\Big|_{\varrho_{\text{eq}}} \tag{1.48}$$
$$\mathbf{C}(\varrho_{\text{eq}}) = \nabla_x \mathbf{h}\Big|_{\varrho_{\text{eq}}} \tag{1.49}$$
$$\mathbf{D}(\varrho_{\text{eq}}) = \nabla_u \mathbf{h}\Big|_{\varrho_{\text{eq}}} \tag{1.50}$$

where:

$$\nabla_x \mathbf{f} = \begin{bmatrix} \frac{\partial f_1}{x_1} & \cdots & \frac{\partial f_1}{x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{x_1} & \cdots & \frac{\partial f_n}{x_n} \end{bmatrix} \tag{1.51}$$

$$\nabla_u \mathbf{f} = \begin{bmatrix} \frac{\partial f_1}{u_1} & \cdots & \frac{\partial f_1}{u_{n_u}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{u_1} & \cdots & \frac{\partial f_n}{u_{n_u}} \end{bmatrix} \tag{1.52}$$

$$\nabla_x \mathbf{h} = \begin{bmatrix} \frac{\partial h_1}{x_1} & \cdots & \frac{\partial h_1}{x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_{n_y}}{x_1} & \cdots & \frac{\partial h_{n_y}}{x_n} \end{bmatrix} \tag{1.53}$$

$$\nabla_u \mathbf{h} = \begin{bmatrix} \frac{\partial h_1}{u_1} & \cdots & \frac{\partial h_1}{u_{n_u}} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_{n_y}}{u_1} & \cdots & \frac{\partial h_{n_y}}{u_{n_u}}. \end{bmatrix} \tag{1.54}$$

H.O.T.

The quantities $\varepsilon_{\mathbf{f}}, \varepsilon_{\mathbf{h}}$ are in fact the higher order terms (H.O.T.) in the Taylor series expansion of Eqs. 1.42, 1.43 and may be written as:

$$\varepsilon_{\mathbf{f}} = \mathbf{f}(x, u, \varrho) - \mathbf{f}(x_{\text{eq}}, u_{\text{eq}}, \varrho_{\text{eq}}) - \mathbf{A}(\varrho_{\text{eq}})x_\delta - \mathbf{B}(\varrho_{\text{eq}})u_\delta \tag{1.55}$$
$$\varepsilon_{\mathbf{h}} = \mathbf{h}(x, u, \varrho) - \mathbf{h}(x_{\text{eq}}, u_{\text{eq}}, \varrho_{\text{eq}}) - \mathbf{C}(\varrho_{\text{eq}})x_\delta - \mathbf{D}(\varrho_{\text{eq}})u_\delta. \tag{1.56}$$

Linearized system

The dynamics of the initial nonlinear parameter-dependent system may now be *approximated* by the following LTI system by *truncating* the higher-order terms $\varepsilon_{\mathbf{f}}, \varepsilon_{\mathbf{h}}$:

$$\mathcal{S}_{\text{LTI}}: \quad \begin{aligned} \dot{\tilde{x}}_\delta &= \mathbf{A}(\varrho_{\text{eq}})\tilde{x}_\delta + \mathbf{B}(\varrho_{\text{eq}})u_\delta \\ y_\delta &= \mathbf{C}(\varrho_{\text{eq}})\tilde{x}_\delta + \mathbf{D}(\varrho_{\text{eq}})u_\delta. \end{aligned} \tag{1.57}$$

A solution $x\big(t;t_0,x(t_0)\big)$ to the nonlinear system $\mathcal{S}_{\text{pd}}$ with $x(t_0)$ being 'close enough' to x_{eq} may now be written as:

$$x\big(t;t_0,x(t_0)\big) \simeq x_{\text{eq}} + \tilde{x}_\delta(t;t_0,0) \tag{1.58}$$

or $x_\delta \simeq \tilde{x}_\delta$, with $x_\delta = x - x_{\text{eq}}$[13]. The question that rises now *is to what extent the approximation $\tilde{x}_\delta$ remains close to x_δ.* The answer to this important question is that the peak absolute difference between the two is bounded provided that the LTI system $\mathcal{S}_{\text{LTI}}$ is stable (i.e. the eigenvalues of $\mathbf{A}$ have negative real parts) and the excitation u_δ is *sufficiently* small (see [31], Ch. 5, §9 or [88], §2.1).

Further results

The same results hold when an approximation of the nonlinear system about an equilibrium *trajectory* $x_{\text{eq}}(t)$ is needed for some constant value of the scheduling vector ϱ. However this time, the state space matrices of the approximate model are *time-varying* and as a result the resulting system is LTV (see also the discussion in Section 1.2.2):

$$\mathcal{S}_{\text{LTV}}: \quad \begin{aligned} \dot{\tilde{x}}_\delta &= \mathbf{A}(\varrho_{\text{eq}},t)\tilde{x}_\delta + \mathbf{B}(\varrho_{\text{eq}},t)u_\delta \\ y_\delta &= \mathbf{C}(\varrho_{\text{eq}},t)\tilde{x}_\delta + \mathbf{D}(\varrho_{\text{eq}},t)u_\delta \end{aligned} \tag{1.59}$$

with:

$$\tilde{x}_\delta \simeq x - x_{\text{eq}}(t) \tag{1.60}$$
$$u_\delta \simeq u - u_{\text{eq}}(t) \tag{1.61}$$
$$\tilde{y}_\delta \simeq y - y_{\text{eq}}(t). \tag{1.62}$$

In the gain scheduling context however, the designer is interested to approximate the behavior of a nonlinear parameter-dependent system for a *family* of equilibrium points rather than a *single* equilibrium point. In this context, the approximation results to an LPV system, being a very different object from a nonlinear system *disguised* in LPV form via state transformations (see discussion of the previous section), parameterized by the scheduling variable ϱ as:

$$\mathcal{S}_{\text{LPV}}: \quad \begin{aligned} \dot{\tilde{x}}_\delta &= \mathbf{A}(\varrho)\tilde{x}_\delta + \mathbf{B}(\varrho)u_\delta \\ y_\delta &= \mathbf{C}(\varrho)\tilde{x}_\delta + \mathbf{D}(\varrho)u_\delta. \end{aligned} \tag{1.63}$$

with:

$$\tilde{x}_\delta = x - x(\varrho) \tag{1.64}$$
$$u_\delta = u - u(\varrho) \tag{1.65}$$
$$\tilde{y}_\delta = y - y(\varrho). \tag{1.66}$$

For any frozen value $\varrho_{\text{eq}} \in \mathbf{\Gamma}$ of the scheduling vector, the LTV plant becomes an LTI and describes the dynamics of the nonlinear parameter-dependent system locally around the corresponding equilibrium point[14].

[13]Dependence on time and initial conditions are omitted.

[14]With a small abuse in notation, the approximated state is almost always noted as x_δ instead of $\tilde{x}_\delta$ to underline our self-satisfaction in the case that $\tilde{x}_\delta \mapsto x_\delta$.

Technical Note: As a technical note concerning Section 1.2.3, the extensive computational capabilities of commercial software for linearization should be outlined. In MATLAB® Simulink Control Design Toolbox for example there exists a full suite of specialized functions (`linearize`, `linmod`) that permit linearization of a nonlinear model around user-specified equilibrium points. This may be done for any portions of the nonlinear model by specifying input/output points in open or closed loop operation. This linearization can also be performed in a *frozen-time* context during a system simulation, providing thus the opportunity to analyze the stability of a gain-scheduled control system for a specific trajectory of the scheduling vector.

Linearizing a nonlinear model is not of course a trivial procedure and the algorithms used are either symbolic (block-by block analytic linearization) or numeric (numerical-perturbation linearization). For either case special attention should be made for discontinuous blocks, delays, saturations and also on the properties of each method (e.g. perturbation levels, open or closed loop linearization etc.) since many subsequent errors are due to a black-box conception of this process.

`Other methods`

Except for the traditional linearization methods based on Taylor-series expansion, there exist also other ways to linearize a nonlinear model. The ones briefly outlined here are *velocity-based linearization* and *feedback linearization.*

Velocity-based linearization is a method that approximates the dynamics of a nonlinear system around *any* given solution-trajectory instead of considering only equilibrium operation like the Jacobian-based approach. This method has in fact received great attention in the last twenty years because it has given birth to a new class of gain-scheduled control systems (namely for autopilots, power systems etc.) and has met significant success, even though there exist controversial opinions on its capabilities (see for example the rather amusing discussion appearing in [84]). A resume of the key points of this methodology are detailed in the next chapter.

Feedback linearization in its turn is a pure nonlinear method very popular in the 70's and 80's that tries to transform the state dynamics of a nonlinear system of the form:

$$\dot{x} = \mathbf{f}(x) + \mathbf{g}(x)u \tag{1.67}$$

$$y = \mathbf{h}(x) \tag{1.68}$$

to an LTI system, using a state feedback control law:

$$u = \boldsymbol{\phi}(x) + \boldsymbol{\psi}(x)v \tag{1.69}$$

and a state transformation $z = \mathbf{z}(x)$. Except for the aforementioned input $\mapsto$ state linearization, a full input $\mapsto$ output linearization is possible. This method is outside the scope of this work, however more details can be found in standard nonlinear control textbooks (see for example [75], Ch. 13 and references therein).

1.3 Linearization-based Gain Scheduling

In this section a detailed review of the *Linearization-based Gain Scheduling - (LBGS)* method is presented. This general class of methods is considered for this monograph for the control of nonlinear parameter-dependent systems and is one of the most used in the control literature (see [88, 114] and references therein). The section starts with a detailed description of the corresponding procedure for the design of a nonlinear gain-scheduled controller, whereas the following sections present the methods that do not or do guarantee certain stability properties of the gain-scheduled loop.

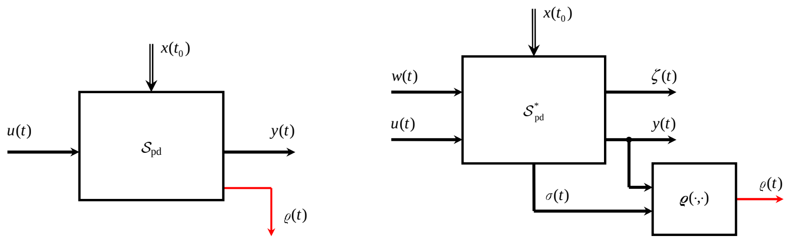
1.3.1 Gain Scheduling Procedure

NLPD system

Start by considering the nonlinear state/output dynamics of a nonlinear parameter-dependent system $\mathcal{S}_{\text{pd}}$ (see Fig. 1.7a):

$$\mathcal{S}_{\text{pd}}: \quad \begin{aligned} \dot{x} &= \mathbf{f}(x, u, \varrho) \\ y &= \mathbf{h}(x, u, \varrho) \end{aligned} \tag{1.70}$$

where $x \in \mathbb{R}^n, u \in \mathbb{R}^{n_u}, y \in \mathbb{R}^{n_y}$ are its state, input and output vectors correspondingly, $\varrho \in \mathbf{\Gamma} \subset \mathbb{R}^{n_\varrho}$ the *measured in real time* scheduling vector with $\mathbf{\Gamma}$ being a connected compact set and $\mathbf{f}, \mathbf{h}$ are nonlinear functions satisfying standard continuity & differentiability conditions.



(a) *Standard modeling* **(b)** *Alternative modeling*

Figure 1.7: *Nonlinear parameter-dependent system.*

Alternative formulation

Remark. Note that the modeling-notation existing in the survey of [88] is used (the scheduling vector however appears explicitly here) for reasons of simplicity. An alternative one is the one appearing in [114] (see Fig. 1.7b), which is more robust control/tracking-oriented:

$$\mathcal{S}^*_{\text{pd}}: \quad \begin{aligned} \dot{x} &= \mathbf{f}(x, u, w, \sigma) \\ \zeta &= \mathbf{h}_\zeta(x, u, w, \sigma) \\ y &= \mathbf{h}_y(x, u, \sigma). \end{aligned} \tag{1.71}$$

The scheduling variable ϱ here is a function of σ that is a vector capturing parametric dependence of the plant and of the output y. Other slightly different formulations are also possible, e.g. see [79].

LBGS The linearization-based gain-scheduling procedure (LBGS) can be divided in five distinct steps:

Step 1 *Trim Analysis.* First the equilibrium states and the corresponding equilibrium inputs (or trim controls) are computed for every value ϱ_{eq} of the scheduling variable inside the domain of operation $\mathbf{\Gamma}$. This can be done either analytically or numerically as detailed in Section 1.2.1.3 and corresponds to finding the equilibrium manifold $\mathcal{E}_{\text{pd}}$ (see Eq. 1.20) of the system. The trim/equilibrium control hyper-surface $u_{\text{eq}} = u(\varrho_{\text{eq}})$ of the system that maintains the state to a corresponding equilibrium value $x_{\text{eq}} = x(\varrho_{\text{eq}})$ is thus obtained. In addition, the equilibrium outputs $y_{\text{eq}} = y(\varrho_{\text{eq}})$ may also be computed. In a noiseless environment, if the system is fed with an initial state $x(t_0) = x_{\text{eq}}$, then $x(t; t_0, x_{\text{eq}}) = x_{\text{eq}}, \forall t > t_0, t_0 \geq 0$ if the input is always $u(t; t_0) = u_{\text{eq}}$ for any value $\varrho_{\text{eq}} \in \mathbf{\Gamma}$.

Step 2 *System Linearization.* In this phase, the nonlinear system dynamics are approximated using Jacobian linearization for any member of the equilibrium manifold and thus, the following LPV system is obtained[15]:

$$\mathcal{S}_{\text{LPV}}: \quad \begin{aligned} \dot{x}_\delta &= \mathbf{A}(\varrho)x_\delta + \mathbf{B}(\varrho)u_\delta \\ y_\delta &= \mathbf{C}(\varrho)x_\delta + \mathbf{D}(\varrho)u_\delta \end{aligned} \tag{1.72}$$

with:

$$x_\delta = x - x(\varrho) \tag{1.73}$$
$$u_\delta = u - u(\varrho) \tag{1.74}$$
$$y_\delta = y - y(\varrho). \tag{1.75}$$

For every frozen value ϱ_{eq} of the scheduling vector, the above linear dynamics describe the initial nonlinear dynamics of Eq. 1.70 in the vicinity of the corresponding equilibrium state $x_{\text{eq}} = x(\varrho_{\text{eq}})$. Here two remarks should be made: first it is sometimes impractical to obtain symbolic expressions for the state matrices of Eq. 1.72 for *every* value of the scheduling vector; a designer may be happy with only a *tabulated* linear model around a significant number of operating points. If a linear model around an intermediate operating point is needed, then interpolation between tabulated points may be performed (this is very common with aeronautical systems where the nonlinear aerodynamic functions are computed in wind tunnels for a family of flight-operating conditions). Second, after the linear model is computed, certain open loop properties may be studied performing eigenvalue or Bode analysis iteratively for every member of the LPV plant; this is a major guideline for the next step: *Local Controller Synthesis.*

[15]The approximation $\tilde{x}_\delta$ of the plant's state is supposed to be near enough to the real value x_δ so that the 'tilde' sign may be omitted. The same also holds for the output y.

Local Controller Synthesis. This step involves the synthesis of a family of LTI controllers $\Sigma\big(K_{\mathrm{LTI}}\big)$ for a set of linearized systems (being frozen instances of the plant $\mathcal{S}_{\mathrm{LPV}}$) being computed for constant values ϱ^i_{eq} of the scheduling vector. These controllers are of the generic form: Step 3

$$K^i_{\mathrm{LTI}} : \quad \begin{aligned} \dot{x}_k &= \mathbf{A}_k(\varrho^i_{\mathrm{eq}})x_k + \mathbf{B}_k(\varrho^i_{\mathrm{eq}})y_\delta \\ u_\delta &= \mathbf{C}_k(\varrho^i_{\mathrm{eq}})x_k + \mathbf{D}_k(\varrho^i_{\mathrm{eq}})y_\delta \end{aligned} \tag{1.76}$$

with $x_k \in \mathbb{R}^{n_k}$ being the controller state vector and u_δ, y_δ defined as:

$$u_\delta = u - u(\varrho^i_{\mathrm{eq}}) \tag{1.77}$$

$$y_\delta = y - y(\varrho^i_{\mathrm{eq}}). \tag{1.78}$$

The matrices $\mathbf{A}_k, \mathbf{B}_k, \mathbf{C}_k$ and $\mathbf{D}_k$ are each time designed in such a way so that stability, performance and robustness properties are met for every member $\mathcal{S}^i_{\mathrm{LTI}}$ of the family of linearized plants obtained for a family of values of the scheduling vector. This is in fact the strong point of the gain scheduling method: use the powerful synthesis methods of linear (mostly robust) control theory (such as $\mathscr{H}_2, \mathscr{H}_\infty$), in order to control an initially nonlinear system.

Local Controller Interpolation. In this important step lies the essence of gain scheduling design: *interpolation.* As it has been already mentioned, the scope of a gain-scheduled controller is to provide a control law *for any value* of the scheduling vector or else *for any point* of the operating domain $\mathbf{\Gamma}$ of the plant; be it a synthesis point or not, and not only for a family of synthesis points. Thus, when coming to on-line implementation, the designer will need only a small memory space for stocking the LTI controllers and an interpolation algorithm able to provide global operation. This may simply be restated as replacing in fact the variable ϱ_{eq} (this implies equilibrium operation) with ϱ and the constant equilibrium quantities $y_{\mathrm{eq}} = y(\varrho_{\mathrm{eq}}), u_{\mathrm{eq}} = u(\varrho_{\mathrm{eq}})$ by $y(\varrho), u(\varrho)$. For more details on this important subject, refer to the following section. Step 4

Controller Implementation & Validation. The last step of the gain-scheduling procedure concerns the final controller implementation. The main problem here is to construct the gain-scheduled controller in such a way that it provides an appropriate trim control input for every value of the scheduling vector. This in fact may be done in many ways (see for example [88], §3.1 or [114], §4.2 or even [149]) but the easiest one is to design the linear controllers so that they contain integral action; as a result $y_\delta \mapsto 0$ and the state of the controller x_k tends to an equilibrium value that corresponds to the u_δ needed for the system's state x to go to the equilibrium value dictated by the scheduling vector's equilibrium value ϱ_{eq}[16]. Step 5

[16]For more details see Section 1.3.3.

1.3.2 Ad-hoc Interpolation Methods

In this section the so-called ad-hoc controller interpolation methods are presented. It is reminded that an interpolation method is needed when the scheduling control law is computed at operating points that do not belong to the set of synthesis points. An interpolation strategy permits calculation of such a control law by combining controllers computed at a small number of synthesis points.

1.3.2.1 Controller Switching

`Controller set`

The *controller switching* method is the simplest one of all interpolation methods; to be more precise it does not involve any interpolation at all. A set $\Sigma(K_{\text{LTI}})$ of LTI controllers is computed:

$$\Sigma(K_{\text{LTI}}) := \left\{K^1, K^2, \dots, K^k\right\} \tag{1.79}$$

`Operating domain`

where each controller $K^i = K(\varrho^i)$ of the set is calculated for fixed-equilibrium values of the scheduling vector $\varrho^i = \varrho^i_{\text{eq}}$ belonging to the system's operating domain $\mathbf{\Gamma}$[17]. Each controller is designed to be *robust* for a given subset Γ^i of the operating domain around the corresponding value ϱ^i (see Fig. 1.8), with:

$$\mathbf{\Gamma} = \bigcup_{i=1}^{k} \Gamma^i \tag{1.80}$$

and:

$$\Gamma^i \bigcap \Gamma^j = \emptyset, \quad \text{with} \quad \{i,j\} = 1,\dots,k \quad \text{and} \quad i \neq j. \tag{1.81}$$

`Switching discussion`

The last condition means that there exists no overlapping in the switching regions Γ^i and the controllers are simply switched according to the scheduling vector trajectory $\varrho(t)$. Now this may be the source of control signal discontinuities and chattering behavior when passing from one scheduling-switching region to the next. This may easily seen by considering the control signal produced from a controller K^1 up to a critical switching time $t = t_{\text{sw}}$. Suppose this controller to be of the standard form:

$$K^1: \quad \begin{aligned} \dot{x}^1_k &= \mathbf{A}^1_k x^1_k + \mathbf{B}^1_k y_\delta \\ u_\delta &= \mathbf{C}^1_k x^1_k + \mathbf{D}^1_k y_\delta. \end{aligned} \tag{1.82}$$

Then the control signal is:

$$u_\delta(t) = \mathbf{C}^1_k \left[e^{\mathbf{A}^1_k (t-t_0)} x^1_k(t_0) + \int_{t_0}^{t} e^{\mathbf{A}^1_k (t-\tau)} \mathbf{B}^1_k y_\delta(\tau) \right] + \mathbf{D}^1_k y_\delta(t). \tag{1.83}$$

From Eq. 1.83 it is evident that if at $t = t_{\text{sw}}$, the controller matrices change their values, then the control signal will be discontinuous. The solution to this problem is fairly simple but it is not very often implemented in real systems.

[17]The controllers are of the form as in Eq. 1.76.

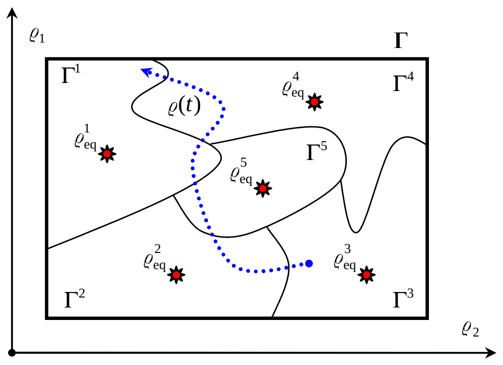


Figure 1.8: Controller switching.

To ensure that $u_\delta(t_{\text{sw}}^+) = u_\delta(t_{\text{sw}}^-)$, it suffices to initialize the new controller K^2 entering on line to a state that ensures bumpless transfer. Indeed:

$$u_\delta(t_{\text{sw}}^+) = \mathbf{C}_k^2 x_k^2(t_{\text{sw}}^+) + \mathbf{D}_k^2 y_\delta(t_{\text{sw}}^+) \triangleq u_\delta(t_{\text{sw}}^-) \tag{1.84}$$

and one has to solve for the initialization state of the second controller $x_k^2(t_{\text{sw}}^+)$[18]. For digitally implemented systems where the difference $\delta t = t_{\text{sw}}^+ - t_{\text{sw}}^- \to 0$, the bump in the control signal may be done arbitrarily small (see [64] or [59]).

`Features` The major advantage of this interpolation method is that it is fairly simple to implement and has been used extensively in real systems. The operating domain is divided in rectangular regions and each controller K^i is valid for given ranges on each component of the scheduling vector. The major disadvantage was already stressed: control signal continuity and stability during transitions.

`Applications` Applications of switching-based gain scheduling are: in [106] a switched gain-scheduled controller for a two link wafer transfer robot system is designed with the scheduling variable being the rotational angle difference of the two links. In [125], a technique for aircraft control is used, generating smooth control signals using LPV control and LMI's. In a similar context, a switching control scheme for the control of magnetic bearings is used in [152]. A not pure switching strategy (involves interpolation in the union of the switching regions) is used in [6] for the control of the water level of a steam generator. An aircraft control example with the speed as a scheduling variable is considered in [64].

Finally, some more theoretical work on the subject, with extensions to nonlinear control and hybrid systems, can be also found in [22, 91, 95]. The first considers a hierarchical switching controller architecture over a set of moving equilibria and uses equilibria-based Lyapunov functions to guarantee stability. The second considers also Lyapunov-based control and regions of stability for a certain class of nonlinear systems. The third one finally considers performance of switched LPV systems and extensions to hybrid systems.

[18]Notice that in general there is more than one solution to Eq. 1.84 and in order to obtain $x_k^2(t_{\text{sw}}^+)$ the pseudo-inverse of $\mathbf{C}_k^2$ may be used (see eq. 1.91).

1.3.2.2 Controller Blending

The *controller blending* method can be seen as a generalization to *controller switching*. Instead of switching controllers when passing from the one operating region to the next, the output control signals of adjacent controllers are blended in order to provide the final control command.

LTI Controller

Consider (for simplicity) a planar operating region $\mathbf{\Gamma}$ and the corresponding two-dimensional scheduling vector $\varrho = [\varrho_1 \quad \varrho_2]^T$ (see Fig. 1.9a). A set $\Sigma_{\mathcal{K}}$ of linear controllers is computed for fixed equilibrium values $\varrho_{\text{eq}}^{i,j}$ of the scheduling vector (red stars). The controllers are distributed evenly in the horizontal and vertical directions and produce rectangular scheduling regions Γ^i. To each rectangular scheduling region Γ^i correspond four controllers $K^{i,j}$, with 'i' being the region index and 'j' (with $j = 1, \ldots, 4$) the controller index of the i'th region, taken in an anti-clockwise manner[19]. An LTI controller $K^{i,j}$ of the form:

$$K^{i,j}: \quad \begin{aligned} \dot{x}_k^{i,j} &= \mathbf{A}_k^{i,j} x_k^{i,j} + \mathbf{B}_k^{i,j} y_\delta \\ u_\delta^{i,j} &= \mathbf{C}_k^{i,j} x_k^{i,j} + \mathbf{D}_k^{i,j} y_\delta \end{aligned} \tag{1.85}$$

is calculated for every synthesis point corresponding to an equilibrium value $\varrho^{i,j} = \varrho_{\text{eq}}^{i,j}$ of the scheduling vector. The total interpolated control output u_δ, for any value of the scheduling vector inside an operating region Γ^i, is calculated by *blending* the four control signals $u_\delta^{i,j}, j = 1 \ldots 4$.

This is done as a function of the distances of the current operating point (cyan star) to the four synthesis points at the edges of the corresponding operating region (see Fig. 1.9a). These distances a_l (with $l = 1 \ldots n_\varrho$ and n_ϱ being the dimension of the scheduling vector) are normalized quantities with $0 \leq a_l \leq 1, \forall l$.

Normalized distances

For a two-dimensional scheduling vector and any rectangular scheduling region Γ^i they are defined as:

$$a_1(t) = \frac{\varrho_1(t) - \varrho^{i,1}}{\varrho^{i,4} - \varrho^{i,1}} \equiv \frac{\varrho_1(t) - \varrho^{i,2}}{\varrho^{i,3} - \varrho^{i,2}} \tag{1.86}$$

$$a_2(t) = \frac{\varrho_2(t) - \varrho^{i,1}}{\varrho^{i,2} - \varrho^{i,1}} \equiv \frac{\varrho_2(t) - \varrho^{i,4}}{\varrho^{i,3} - \varrho^{i,4}}. \tag{1.87}$$

Control input

The total blended control input u_δ^i, being a function of the normalized distances and the control signals of each controller $u_\delta^{i,j}$, is computed as:

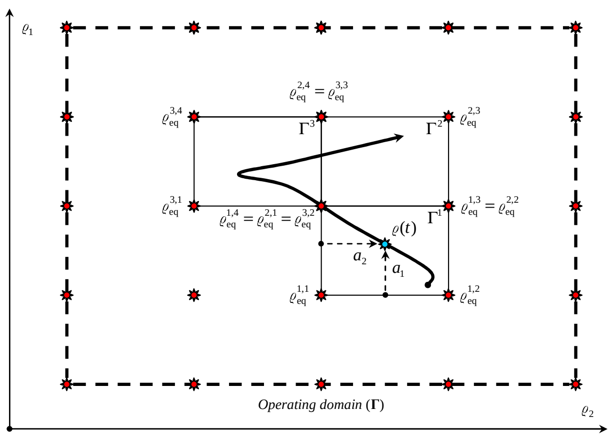
$$u_\delta^i = \big[1 - a_1(t)\big] u_\delta^{i,\{1,2\}} + a_1(t) u_\delta^{i,\{3,4\}} \equiv u_\delta^{i,\{1,2,3,4\}} \tag{1.88}$$

with:

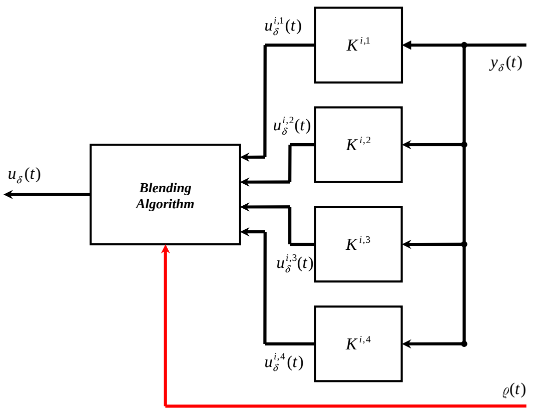
$$u_\delta^{i,\{1,2\}} = \big[1 - a_2(t)\big] u_\delta^{i,1} + a_2(t) u_\delta^{i,2} \tag{1.89}$$

$$u_\delta^{i,\{3,4\}} = \big[1 - a_2(t)\big] u_\delta^{i,4} + a_2(t) u_\delta^{i,3}. \tag{1.90}$$

[19]Obviously some controllers may be used for up to four neighbor regions, depending their position on the scheduling region $\mathbf{\Gamma}$; as a result this numbering is non-unique.



(a) *Operating region visualization*



(b) *Controller realization*

Figure 1.9: Controller blending technique.

A possible interpolation scenario is depicted in Fig. 1.9a, with the trajectory $\varrho(t)$ passing through three scheduling regions ($\Gamma^1 \looparrowright \Gamma^3 \looparrowright \Gamma^2$). During the first transition (which is a little exaggerated since the trajectory passes exactly from the synthesis point $\varrho_{\text{eq}}^{1,4}$) all three controllers $K^{1,1}, K^{1,2}, K^{1,3}$ turn off and controllers $K^{3,1}, K^{3,3}, K^{3,4}$ go on-line to replace them. However, controller $K^{1,4} \equiv K^{3,2}$ remains on-line for both regions Γ^1, Γ^3. Similarly, during the second transition, controller $K^{1,4}$ remains always on-line as well as controller $K^{3,3}$; however controllers $K^{3,1}, K^{3,4}$ give their place to $K^{2,2}, K^{2,3}$ respectively.

`Features` A simplified structure of the interpolator is visualized in Fig. 1.9b. The scheme is rather simplified but it shows the essence of the method: only the *outputs* of the controller are processed and not the controller themselves as with other interpolation strategies (see following sections). However, a hierarchical mechanism should be added so as to decide when and how to switch on and off the controllers. This is a major advantage of this controller interpolation method: it is not obligatory to use controllers of the same structure or of the same complexity for each synthesis point since it is only each controller's output that is processed. This is not the case with other controller interpolation methods such as *gain blending* where the controller structure/order remains the same and the interpolation procedure is done on the controller parameters.

Another advantage of the method is in terms of the numerical computations needed to obtain the control law; in [73, 74] it is argued that this method is significantly faster in terms of multiplications & additions needed to compute the interpolated control signal in comparison for example with *state-space matrix* or *zero-pole-gain* interpolation.

This method however presents some important disadvantages: an important one is *controller initialization.* Consider once again the scenario of Fig. 1.9a where the scheduling vector crosses the boundary of the regions Γ^3 and Γ^2. At the exact moment t_{sw}, where the scheduling vector is on the border of the two regions, the control signal is affected by the outputs of only two controllers and two new controllers should be put on-line and *initialized to some state*, in order to be able to perform interpolation in region Γ^2 for $t > t_{\text{sw}}$ (during of course the time that $\varrho(t) \in \Gamma^2$).

However, this initialization process is not a trivial matter since if these controllers are switched on with zero initial state conditions there will probably be an initial transient on the total control output due to the inconsistency of the newly entered controllers' states added and the operating situation of the system before the switching. This transient may be rendered smaller if these states are initialized in a smarter way. A possible solution is to re-initialize *all four* controllers of the new region Γ^3 to a state dictated by the actual control signal $u_\delta(t_{\text{sw}})$, where t_{sw} is the switching time (see [137] or Chapter 5):

$$x_k^{i,j}(t_{\text{sw}}) = \left(\mathbf{C}_k^{i,j}\right)^+ \left[u_\delta(t_{\text{sw}}) - \mathbf{D}_k^{i,j} y_\delta(t_{\text{sw}})\right].^{20} \tag{1.91}$$

[20]The '+' sign in the exponent denotes the Moore-Penrose pseudo-inverse of a matrix.

These inevitable transients become more annoying when the average period that the scheduling vector stays inside any operating region is small. This may happen either due to the fact that $\varrho(t)$ varies relatively fast with respect to the plant's settling time or if the scheduling regions are too small; i.e. the gridding of the operating domain is overly dense. In this case, even though the designer could expect an increase in performance when the synthesis points augment, the initial transients may render the closed loop system slow. This problem may be corrected if 'sufficiently fast' dynamics are assigned to the LTI controllers that will internally compensate the state inconsistency.

This initial transient problem may yet be amplified if the scheduling vector demonstrates big step changes from one value to the next, making thus the interpolator jump to interpolation regions that are not neighbor. In this case the state initialization may not be useful at all and the transients heavier. A possible solution to this problem is to filter the scheduling vector with a *low-pass* filter in order to force the scheduling vector pass from all regions in between and spend a finite time at each one. However much attention should be paid on the filter's bandwidth so as not to augment the closed-loop rise & settling times.

Another disadvantage of this method is the fact that the scheduler needs four controllers (if there are two parameters, triangles could be considered and thus three controllers are enough) to be implemented, apart from the unit that performs the state initialization. This strategy is more complex than say, a gain interpolation one where only a single controller (e.g. a PID) is implemented and solely its gains interpolated, depending on the location of the scheduling vector.

`Bibliography`

Some solid work on the subject appears on two nearly identical papers (see [73, 74]), where this method is compared to state space & zero-pole-gain (ZPK) interpolation methods[21] and several of its details are discussed. However some of the disadvantages that this method possesses are not stressed out.

Notable work on aeronautical systems (namely missile autopilots) controlled with this type of interpolation are presented in [35, 81] and [137] respectively. In [81], a missile autopilot is designed using the controller blending method but the simulation results are not so thorough, even though local linear equivalence properties for the gain-scheduled controller (see Section 1.3.3) are exploited. In [35] a μ-analysis method is used for the LTI controllers for the design of a 3DOF missile autopilot. The most complete treatment on the subject can be found in [137] (or equivalently in Chapter 5 of this monograph), where extensive simulations are used to validate and compare the approach with an alternative observer based blending strategy, detailed in Section 1.3.2.6. A controller blending approach is also used in [58] for the control of a power plant boiler. The scheduling variables used are the steam temperature and pressure and the effectiveness of the gain-scheduled over a robust control scheme is demonstrated. Finally in [60] an interesting LPV-based method for the control of a vehicle powertrain using static $\mathscr{H}_\infty$ controllers and controller blending is proposed.

[21] These methods are analyzed in the sections to follow.

1.3.2.3 ZPK Interpolation

The *zero-pole-gain interpolation* (ZPK) method is one of the standard techniques used for controller interpolation. For simplicity SISO systems will be considered, however the analysis could be extended for MIMO ones, even though the method is not adapted for truly multivariable setups.

LTI controller

Consider once again a set $\Sigma(K_{\text{LTI}})$ of LTI-SISO controllers designed for fixed values of the scheduling vector inside a scheduling region $\mathbf{\Gamma}$[22]. Each controller $K^{i,j}$ may be represented in the s-domain in a ZPK form as:

$$K(s)^{i,j} = K^{i,j} \frac{\prod_{k=1}^{m} \big(s - z_k^{i,j}\big)}{\prod_{l=1}^{n} \big(s - p_l^{i,j}\big)} \tag{1.92}$$

with $z_k^{i,j}, p_l^{i,j}$ being the k-*th* (respectively l-*th*) zero (respectively pole) and $K^{i,j}$ the dc-gain of the j-*th* controller[23] at the corresponding i-*th* scheduling region Γ^i.

Global controller

Each zero, pole and gain is interpolated in the same way as with the controller blending method. For each value $\varrho(t)$ of the scheduling vector, the normalized distances are given by Eqs. 1.86, 1.87 and thus the final interpolated compensator has the following form:

$$K(s, \varrho) = K(\varrho) \frac{\prod_{k=1}^{m} \big(s - z_k(\varrho)\big)}{\prod_{l=1}^{n} \big(s - p_l(\varrho)\big)}. \tag{1.93}$$

The zeros, poles and gain of the compensator are now dependent on time since $\varrho(t)$ draws a trajectory inside the operating region of the system. Consider for example the k-*th* zero of the interpolated compensator when the scheduling vector is inside the i-*th* scheduling region; its (time-dependent) value is given by:

$$z_k(\varrho) = \big[1 - a_1(t)\big] z_k^{i,\{1,2\}} + a_1(t) z_k^{i,\{3,4\}} \equiv z_k(\varrho)^{i,\{1,2,3,4\}} \tag{1.94}$$

with:

$$z_k^{i,\{1,2\}} = \big[1 - a_2(t)\big] z_k^{i,1} + a_2(t) z_k^{i,2} \tag{1.95}$$

$$z_k^{i,\{3,4\}} = \big[1 - a_2(t)\big] z_k^{i,4} + a_2(t) z_k^{i,3}. \tag{1.96}$$

Features

The major advantage of this method is that it maintains ‘a good engineering feeling’ in the interpolation process. Indeed, it is more natural and straightforward to interpolate the zeros-poles-gains of a controller than say, the coefficients of a transfer function since the effect of a changing pole (or a zero or also a gain) could be easily linked to the step or frequency response of the closed loop system. However this method becomes rather complicated in the case of complex poles/zeros and for multivariable systems.

[22] Consider a two-dimensional scheduling variable ϱ and the same setup and notation as in Section 1.3.2.2.

[23] Once again $j = 1, \ldots, 4$, since rectangular scheduling regions Γ^i are considered.

There is however a significant issue for this type of interpolation, concerning implementation. Modern gain-scheduled controllers are implemented using digital components thus a fundamental question arises: should a controller be first discretized and then interpolated in the z-domain or first interpolated in the s-domain and then discretized?

Example

To answer this question consider a single pole (real for simplicity) s_p and the mapped to the z-domain equivalent one $z_p = e^{s_p T}$, where T is the sampling period. Suppose both poles are *equally* perturbed to a new value s_p^* and z_p^* with $s_p^* = s_p + \delta, z_p^* = z_p + \delta$ and $\delta > 0$ a small real number. The quotient $q(s_p, T)$[24] of the difference between the z-domain mapped perturbed pole $z_p^* = \mathscr{Z}(s_p^*)$ (perturbation in the s-domain) and the nominal pole z_p, and the difference between the s-domain mapped perturbed pole $s_p^* = \mathscr{S}(z_p^*)$ (perturbation in the z-domain) and the nominal pole s_p is:

$$q(s_p, T) = \frac{z_p^* - z_p}{s_p^* - s_p} (\equiv \frac{z_\delta}{s_\delta}) = \frac{e^{(s_p+\delta)T} - e^{s_p T}}{\dfrac{\ln\left(e^{s_p T} + \delta\right)}{T} - s_p}. \tag{1.97}$$

From the following figure it can be seen that this difference quotient is rather small; this means that a perturbation on the s-domain pole results to a much smaller perturbation to the corresponding pole on the z-domain than the inverse. As a result, it is preferable to perform the interpolation first to the s-domain and then discretize the controller since numerical sensitivity is bigger in the z-domain (a similar but approximative analysis may be found in [74], pp. 178).

Applications

A recent paper addresses the control of a pick and place machine and performs interpolation using the length of the beam used for transportation [108]. Also in the reference paper [103], a missile autopilot using robust $\mathscr{H}_\infty$ controllers scheduled on the vertical acceleration and Mach number but the control scheme is complicated and more performing controllers are proposed in later works.

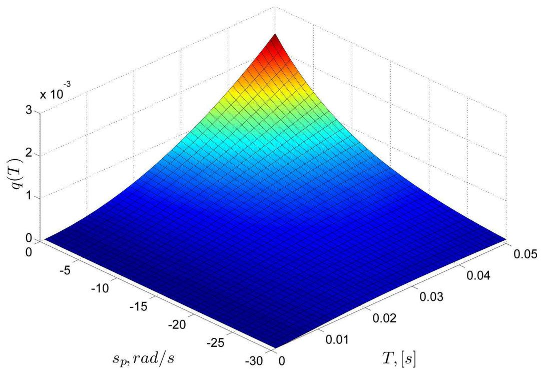


Figure 1.10: Quotient concerning ZPK mapping.

[24]The dependence on the perturbation δ is considered constant; here a value $\delta = 0.1$ rad/s is taken for both mappings.

1.3.2.4 Transfer Function Coefficient Interpolation

The *transfer function coefficient* is similar to the ZPK interpolation technique. Once again SISO systems will be considered for simplicity; in any case this method also is not particularly suited for MIMO setups.

LTI controller

The transfer function of an LTI SISO controller may be written in the following (alternative to Eq. 1.92 ZPK) form[25]:

$$\begin{aligned} K(s)^{i,j} &= \frac{\sum_{k=1}^{m} \beta_k^{i,j} s^k}{\sum_{l=1}^{n} \alpha_l^{i,j} s^l} \\ &= \frac{\beta_k^{i,j} s^k + \beta_{k-1}^{i,j} s^{k-1} + \ldots + \beta_1^{i,j} s + \beta_0^{i,j}}{\alpha_l^{i,j} s^l + \alpha_{l-1}^{i,j} s^{l-1} + \ldots + \alpha_1^{i,j} s + \alpha_0^{i,j}}. \end{aligned} \tag{1.98}$$

Global controller

The numerator and denominator have m and n coefficients respectively that define their dynamics. In the gain-scheduling context, a set of controllers is again designed at a number of synthesis points and the corresponding controller coefficients for the i-th scheduling region and the j-th controller of this region are denoted by $\beta_k^{i,j}$ and $\alpha_l^{i,j}$. To obtain a global interpolated controller:

$$\begin{aligned} K(s,\varrho) &= \frac{\sum_{k=1}^{m} \beta_k(\varrho) s^k}{\sum_{l=1}^{n} \alpha_l(\varrho) s^l} \\ &= \frac{\beta_k(\varrho) s^k + \beta_{k-1}(\varrho) s^{k-1} + \ldots + \beta_1(\varrho) s + \beta_0(\varrho)}{\alpha_l(\varrho) s^l + \alpha_{l-1}(\varrho) s^{l-1} + \ldots + \alpha_1(\varrho) s + \alpha_0(\varrho)} \end{aligned} \tag{1.99}$$

the transfer function coefficients of adjacent controllers are interpolated using the same formulas as in the previous section.

Features

Now this method seems well suited for SISO systems and relatively low order controllers (lead, lag, PID) since if the controller's order rises the effect of interpolation on its stability becomes less clear. This means that there is no guarantee that the interpolated controller will have linearly varying dynamics even if the coefficients are updated linearly (this is not however the case with ZPK interpolation which is more direct).

Applications

Relative work on the subject may be found in [150], where a missile autopilot is obtained using interpolation of controller coefficients. In [62] a similar system is considered but this time $\mathscr{H}_\infty$ loop shaping controllers are designed and a least-squares analytic approach to obtain the global controller coefficients is adopted. In [92] a robust controller for aircraft is designed whereas finally in [26] a MIMO controller for a frigate ship is computed using sea state data and the ship's velocity. In this final work, an LPV gain-scheduled controller is compared with the linearization-based with coefficient interpolation and with a robust LTI; in all cases the gain-scheduled schemes perform better than the robust one.

[25]Very often the coefficients of the transfer function are normalized so that the *highest power* coefficients of both the numerator and the denominator become unitary.

1.3.2.5 State Space Matrix Interpolation

The *state-space matrix interpolation* is a method that offers a nonlinear gain-scheduled controller by blending the coefficients of the state space representation of local LTI controllers. Consider for example the following state space representation of a controller $K^{i,j}$ designed at the j-*th* point of the i-*th* scheduling region of the operating domain $\mathbf{\Gamma}$ of a nonlinear parameter-dependent system:

LTI controller

$$K^{i,j}: \quad \begin{aligned} \dot{x}_k^{i,j} &= \mathbf{A}_k^{i,j} x_k^{i,j} + \mathbf{B}_k^{i,j} y_\delta \\ u_\delta^{i,j} &= \mathbf{C}_k^{i,j} x_k^{i,j} + \mathbf{D}_k^{i,j} y_\delta \end{aligned} \tag{1.100}$$

In contrast to the *controller blending* method which interpolates the outputs of the controllers $u_\delta^{i,j}$, the *state-space matrix interpolation* method interpolates directly their internal structure. As a result, the structure of the gain-scheduled controller $K(\varrho)$ will be:

Global controller

$$K(\varrho): \quad \begin{aligned} \dot{x}_k &= \mathbf{A}_k(\varrho) x_k + \mathbf{B}_k(\varrho) y_\delta \\ u_\delta &= \mathbf{C}_k(\varrho) x_k + \mathbf{D}_k(\varrho) y_\delta. \end{aligned} \tag{1.101}$$

Now each element of each matrix $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ is obtained linearly by interpolating the four adjacent controller matrices using the normalized distances a_1, a_2 (when scheduling on the plane) that are a function of the scheduling vector as in the previous sections. Consider for example the matrix $\mathbf{A}$:

$$\mathbf{A}_k(\varrho) = \big[1 - a_1(t)\big] \mathbf{A}_k^{i,\{1,2\}} + a_1(t) \mathbf{A}_k^{i,\{3,4\}} \equiv \mathbf{A}_k(\varrho)^{i,\{1,2,3,4\}} \tag{1.102}$$

with:

$$\mathbf{A}_k^{i,\{1,2\}} = \big[1 - a_2(t)\big] \mathbf{A}_k^{i,1} + a_2(t) \mathbf{A}_k^{i,2} \tag{1.103}$$

$$\mathbf{A}_k^{i,\{3,4\}} = \big[1 - a_2(t)\big] \mathbf{A}_k^{i,4} + a_2(t) \mathbf{A}_k^{i,3}. \tag{1.104}$$

Features

This method, even though it seems rather straightforward, it presents several disadvantages compared to other methods. First, it is rather demanding on calculations since all elements of a state space realization need to be interpolated; this may be very conservative. For example the state space representation of a SISO, 2^{nd} order controller may have up to nine coefficients whereas a ZPK or transfer function realization up to five (for a 3^{rd} order controller it is even worse: sixteen and seven respectively). Second, the effect of interpolation to the zeros and poles and thus to the final controller dynamics is not straightforward and may also lead to numerical problems for ill-conditioned realizations. Third, a similar controller structure/realization is assumed for all synthesis points in order for the interpolation to have a meaning since interpolating between different states is not a sound strategy (see [73] for more details).

For these reasons, there does not exist significant applications using this type of interpolation, even though some work on stability preserving interpolation can be found in [133], as it will be presented in the next part of this chapter.

1.3.2.6 Observer/State Feedback Interpolation

The *observer/state feedback interpolation* method is somewhat close to the *state-space matrix interpolation* one in the sense that controller matrices are interpolated in order to obtain a gain-scheduled controller. Consider once again the setup used in the previous sections concerning the synthesis points and the scheduling vector.

A well-known control strategy for MIMO LTI systems is the observer/state feedback compensator. For the initial nonlinear parameter-dependent system in Eq. 1.70, consider a family of (strictly proper for simplicity) linearized plants computed at a number of operating points[26]:

$$\mathcal{S}^{i,j}_{\text{LTI}} : \quad \begin{aligned} \dot{x}_\delta &= \mathbf{A}^{i,j} x_\delta + \mathbf{B}^{i,j} u_\delta \\ y_\delta &= \mathbf{C}^{i,j} x_\delta . \end{aligned} \tag{1.105}$$

`LTI Controller`

An observer/state feedback controller for each of the above linearized systems is written as:

$$K^{i,j} : \quad \begin{aligned} \dot{\hat{x}}_\delta &= \mathbf{A}^{i,j} \hat{x}_\delta + \mathbf{B}^{i,j} y_\delta + \mathbf{K}^{i,j}_{\text{o}} (y_\delta - \mathbf{C}^{i,j} \hat{x}_\delta) \\ u_\delta &= \mathbf{K}^{i,j}_{\text{c}} \hat{x}_\delta \end{aligned} \tag{1.106}$$

with $\mathbf{K}^{i,j}_{\text{c}}, \mathbf{K}^{i,j}_{\text{o}}$ being respectively the observer and controller matrices for each design point. Now the observer is estimating the state error close to the equilibrium point whereas the controller uses this estimation to perform a pole placement. The interpolation procedure here updates *all* matrices as a function of the scheduling vector (the plants' included) in order to provide the following gain-scheduled controller:

`Global controller`

$$K(\varrho) : \quad \begin{aligned} \dot{\hat{x}}_\delta &= \mathbf{A}(\varrho) \hat{x}_\delta + \mathbf{B}(\varrho) y_\delta + \mathbf{K}_{\text{o}}(\varrho) \big(y_\delta - \mathbf{C}(\varrho) \hat{x}_\delta\big) \\ u_\delta &= \mathbf{K}_{\text{c}}(\varrho) x_\delta \end{aligned} \tag{1.107}$$

`Features`

Now here arise several issues: first, it is clear that this interpolation method is very demanding in calculations since both the controller structure (matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$) *and* the controller dynamics (matrices $\mathbf{K}_{\text{c}}, \mathbf{K}_{\text{o}}$) need to be updated as a function of the scheduling vector $\varrho(t)$. Second, an important issue is the update of the controller structure itself; normally, the observer should reconstruct the state of the linearized system at any possible operating point. This is done by considering the system matrices at the specific operating point; however given that these matrices are in general computed at a small number of operating points (along with the controller/observer gain matrices), it is clear that an interpolation used for any other operating point may yield a different linearized plant than the one *explicitly or symbolically* computed at this operating point.

[26]Once again 'i' denotes the scheduling region and 'j' the index of the controller as in the previous sections. Also the indexes i,j are omitted for the state, input and output of each linearized plant for simplicity. However it remains evident that the errors $x_\delta, u_\delta, y_\delta$ are taken with respect to the equilibrium values of *each* synthesis point.

To illustrate this fact, consider the following (oversimplified) example: suppose that a nonlinear parameter-dependent system has been linearized around the origin and the following LPV plant describes its dynamics around this point: `Example`

$$\dot{x}_\delta = 2\varrho^2 x_\delta = \mathbf{A}(\varrho)x_\delta. \tag{1.108}$$

Suppose also that the scheduling vector ϱ may vary arbitrarily between 1 and 2 and the state deviation is defined as:

$$x_\delta = x - x_{\text{eq}} \equiv x. \tag{1.109}$$

Suppose that an observer for the system's state is to be constructed using the two extremal values $\mathbf{A}(\varrho_{\min}) = 2 \cdot 1^2 = 2$ and $\mathbf{A}(\varrho_{\max}) = 2 \cdot 2^2 = 8$ of the system dynamics, for the operating point corresponding to $\varrho = 1.5$. If a linear interpolation between the two values is taken, then the interpolated matrix used in the observer will be $\tilde{\mathbf{A}} = 2 + \frac{8-2}{2} = 5$.

However the true value, if exact calculations were to be used, is computed as $\mathbf{A}(\varrho)\big|_{1.5} = 2 \cdot 1.5^2 = 4.5$. Now, if the synthesis point were four (for $\varrho = 1, 1.33, 1.66$ and 2 respectively) then the interpolated values would have been 4.55; much closer to the explicitly calculated one.

This simplistic example shows the danger of estimating the dynamics of a 'wrong' plant which may result in poor performance, if the operating domain gridding is not done correctly. A solution to this problem is to perform a denser gridding when linearizing the plant but calculate the controller/observer matrices at a smaller number of points. A possible scenario is depicted in Fig. 1.11 where the blue stars denote linearization points and the red both linearization & controller synthesis-interpolation ones[27].

A third issue has to do with appropriate realization of the controller and observer so that there is a meaning in interpolation. In addition, given that the interpolation is on the controller matrices, it is not very clear if the interpolated quantities will behave well as far as stability is concerned (this in fact is the same problem with state-space matrix interpolation).

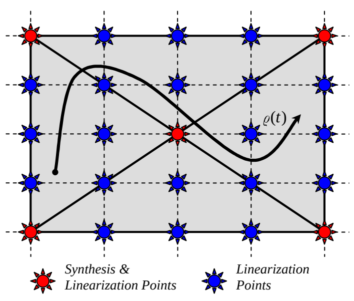


Figure 1.11: A potential gridding scenario.

[27]In this example four triangular interpolation regions are defined that reduce conservatism at it will be discussed in Section 1.4.

The major advantage of this control configuration is that it is based on an estimation of the nonlinear plant's state itself; this feature may be *highly desirable* for the validation of a gain-scheduled control scheme. In addition, the synthesis of observer-based state feedback schemes is rather simple and may easily treat MIMO systems, in contrast to say, *ZPK* or *transfer function coefficient* interpolation. In addition, discretization-implementation is by far more straightforward in state-space setups. Finally, there exist significant work in the field of *stability preserving interpolation* schemes using this type of control loops (see later sections).

Applications

As far as applications of this interpolation method are concerned, in [7, 137] the observed-based interpolation technique was compared to the controller blending one for an integrated flight and propulsion control system and a missile autopilot respectively. In [23], a gain-scheduled controller was used to attenuate disturbances due to engine-induced vibrations whereas in [147], arbitrary $\mathscr{H}_\infty$ compensators are converted to state feedback/observer form for the control of a launcher. In [100] a discrete time controller is interpolated for the stabilization of an electrostatic levitator whereas in [29] a nice application in a multi-motor web transport system is presented. Finally [65], is a rather good reference on the subject.

Youula param/tion

An interesting extension to the state feedback/observer interpolation method is the *Youla parameter-based interpolation* method. It is known that any stabilizing LTI controller K for an LTI plant $\mathcal{S}_{\mathrm{LTI}}$ may be written as the l-LFT of a stabilizing observer-based state feedback controller $\mathcal{J}$ *plus* a free, stable system $\mathcal{Q}$ which is called the *Youla* parameter, being driven by the *innovations signal.*

If desired stabilizing (but of arbitrary structure) compensators $K^{i,j}$ are designed for some specific operating points of the plant, it is possible to retain the same observer-state feedback controller for every operating point and change only the corresponding Youla parameter $\mathcal{Q}^{i,j}$ so as to obtain:

$$K^{i,j} = \mathcal{F}_l(\mathcal{J}, \mathcal{Q}^{i,j}). \tag{1.110}$$

The Youla parameter may then be scheduled in order to obtain a scheduler transfer function $\mathcal{Q}(\varrho)$ and therefore a scheduled global controller $K(\varrho)$. Possible stability preserving extensions to this method are considered in the next section.

Applications

An interesting application of this method is found in [102] where the scheduled parameter $\mathcal{Q}$ is used to achieve rejection of vibrations in magnetic bearing systems. In [134] a SIMO servo controller is scheduled using two extremal LTI controllers (that correspond to different Youla parameters): the first one for performance and the second one for good robustness and error tracking suppression. In [104] a scheme for gain scheduling control is devised when the scheduling parameter ϱ is not known and need to be estimated; this is done using the Youla parameterization procedure mentioned above. Finally in [110] some theoretical work is done in the context of continuation of observer-based structures for issues arising from interpolation in gain scheduling control.

1.3.2.7 Other Interpolation Schemes

In this section some additional interpolation techniques will be cited that are either less used in the bibliography, or they may be regarded as transformations to the existing methods detailed in the previous sections.

Gain blending

The first technique is called *Gain Blending* and is maybe the more standard interpolation technique of all, since due to this method the *Gain Scheduling* terminology rises. A very often used industrial controller is the 'PID' type, existing both in a simple SISO form or in more complex inner-outer loop or MIMO forms. It is not needed to cite the benefits from PID control since they are widely accepted: ease of use and implementation, optimality, engineering intuition preserved etc[28]. The PID controller is nicely tailored for a great variety of systems (automotive, aeronautical etc.) and tuning a PID is the most frequent task a systems engineer may be asked to perform on the field. In addition, adaptive PID controllers are an excellent (and preferable) choice for the control of parameter-dependent systems.

The *Gain Blending* technique is exactly that: for a set of operating points, compute a family of PID controllers of the form:

$$K(s)^{i,j} = K_{\mathrm{p}}^{i,j} + K_{\mathrm{i}}^{i,j}\frac{1}{s} + K_{\mathrm{d}}^{i,j}s. \tag{1.111}$$

Then interpolate the gains $K_{\mathrm{p}}, K_{\mathrm{i}}, K_{\mathrm{d}}$ at each operating region following the scheduling vector evolution in order to obtain a gain-scheduled controller[29].

Applications

Some nice applications of gain blending can be found for example in [138] where a missile autopilot was calculated by scheduling PID controllers of a special type or in [139] where a re-entry vehicle autopilot is considered[30]. Another good practical example can be found in [69] where a PID controller is scheduled for the control of a diesel engine.

Riccati interpolation

Another interpolation method used in the context of robust gain-scheduled control is based on the interpolation of the solutions $\mathbf{X}_\infty, \mathbf{Y}_\infty$ of Riccati equations relevant to $\mathscr{H}_\infty$ control synthesis. For further details on applications of this method see [8, 112].

Other methods

Finally, other methods could be considered such as coprime factor scheduling, fuzzy interpolation schemes or even mixed strategies. In the following table, the most important interpolation schemes described in the previous sections are compared with each other using various criteria (industrial use, implementation complexity, possible use for MIMO systems etc.).

[28] Another interesting feature of the PID controller is that it includes integral action that ensures proper reference tracking; feature that is very important and very often a problem with other interpolation structures that do not necessarily provide a correct trim input for non-synthesis operating points (see Section 1.3.3 for more details)

[29] The PID controller may be seen as a controller with transfer function $K(s) = \frac{K_{\mathrm{d}}s^2+K_{\mathrm{p}}s+K_{\mathrm{i}}}{s}$ or even in ZPK form. That is why a *Gain Blending* terminology does not really define a separate interpolation strategy.

[30] Both articles being part of this thesis are considered in Part II of this manuscript.

Table 1.1: Comparison of interpolation methods.

Features → *Methods* ↓	Industrial Spread	Computational Complexity	MIMO Use	Stability-preserving Extensions	Limitations on Controller Order - Structure	Signal Continuity Interpolation Coherence
Controller Switching	☺	☺	☺	☺☹	☺	☹
Controller Blending	☹	☺	☺	☹	☺	☹
ZPK Interpolation	☺☹	☺☹	☺☹	☹	☹	☺
TF Coefficient Interpolation	☺☹	☺☹	☺☹	☹	☹	☹
SS Matrix Interpolation	☹	☹	☺	☺	☹	☹
Observer-based Interpolation	☺☹	☹	☺	☺	☹	☺☹
Gain Blending	☺	☺	☺☹	☹	☺☹	☺

1.3.3 Stability-preserving Methods

The interpolation procedure used for the construction of the global gain-scheduled controller is crucial since it may cause instability to the closed loop system, even when the LTI controllers are designed to assure stability around the synthesis points. In this section some results concerning the analysis of methods that assure a degree of stability for the gain-scheduled plant are presented.

1.3.3.1 The Origins

A thorough analysis of gain-scheduled control systems lacked in the bibliography for many years, even though many real-world systems used this attractive control tool since the 1950's. Most theoretic work focused on general stability theory for feedback time-varying systems and the connection between theory and practice was not clear. The first systematic work on this subject appears in the bibliography with the PhD thesis of J. S. Shamma (see [119]) in the late 80's. The author considers three major types of gain-scheduled systems: a parameter-dependent linear plant (LPV) scheduling on its time-varying parameters, a nonlinear plant (rendered LPV by linearization) scheduled on a reference trajectory or on its output. In the first case, the author considers LPV systems of the following form[31]:

$$\mathcal{S}_{\text{LPV}}: \quad \begin{aligned} \dot{x} &= \mathbf{A}(\varrho)x + \mathbf{B}(\varrho)u \\ y &= \mathbf{C}(\varrho)x + \mathbf{D}(\varrho)u \end{aligned} \tag{1.112}$$

and it is argued that closed loop stability and the good properties of the feedback loop, around the family of operating points for which LTI controllers have been designed, may be retained provided that the parameter vector ϱ *varies slowly* in some operating domain $\mathbf{\Gamma}$. The overall analysis is rather complicated and conservative but gives for the first time sufficient conditions for the good behavior of a gain-scheduled control system.

In the second case when scheduling on a reference trajectory $y_{\text{ref}}(t)$ (that may or may not be known beforehand), the plant's output y should follow $y_{\text{ref}}(t)$. The author shows that the closed loop system is stable if the controller designed *for all* frozen-values of time (representing distinct values of the reference trajectory and thus distinct LTI snapshots of the LPV plant in Eq. 1.112) provides robust stability/performance *and* the reference trajectory *varies slowly.*

Finally in the third case, a more realistic case is considered where the gain-scheduled controller is updated (using state space matrix interpolation) with the output of the plant in order to ensure following of a reference output value. Again slowness conditions are imposed as well as the notion of 'capturing the plant's nonlinearities' with the scheduling variable, meaning that unmodeled dynamics of the plant should be relatively small. This work has led to two important, yet not so easily exploited, reference papers on the subject (see [120] and [121]).

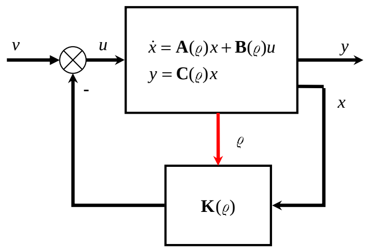
[31]This system may in fact represent linearized dynamics of a nonlinear system with x being in fact $x_\delta = x - x_{\text{eq}}(\varrho)$ and the same for the input and output vector.

Several important points of the gain scheduling practice relevant to the two aforementioned publications are developed in more detail in [122]. These points include loss of stability and non-minimum phase properties of a gain-scheduled plant for rapid parameter variations; in addition real-world examples are given.

Another very interesting approach for the stability analysis of gain-scheduled systems is found in [118]. In this important paper that was published in the early nineties, right after the pioneering work of J.S. Shamma, the important problem of state feedback gain-scheduled regulation is considered. More precisely, for the LPV dynamics of Eq. 1.112[32], the following state feedback control law is considered (see Fig. 1.12):

$$u = -\mathbf{K}(\varrho)x + v \tag{1.113}$$

The state feedback matrix $\mathbf{K}$ is computed at a finite number of points $[\varrho_1, \ldots, \varrho_k]$ so that the eigenvalues of the corresponding frozen closed loop systems $\mathbf{A}(\varrho_j) - \mathbf{B}(\varrho_j)\mathbf{K}(\varrho_j)$ induce exponential stability. The choice of the synthesis points is done in such a way that the closed loop *scheduled* systems $\mathbf{A}(\varrho) - \mathbf{B}(\varrho)\mathbf{K}(\varrho)$ are also exponential stable and their eigenvalues are *close enough* to the frozen designs *if* the gain matrices are interpolated linearly between the synthesis points. In addition, the gain-scheduled loop remains stable *only if* the scheduling vector derivative is smaller to a certain amount. So once again, the notion of slowness in the time-varying parameter is inferred. Finally, conditions and ways of computing the state feedback matrix, so that the bounds on the parameter variation rate may be made arbitrarily high without causing instability, are devised.



v
u
$\dot{x} = \mathbf{A}(\varrho)x + \mathbf{B}(\varrho)u$
$y = \mathbf{C}(\varrho)x$
y
x
-
ϱ
$\mathbf{K}(\varrho)$

Figure 1.12: State feedback gain-scheduling.

The initial period on gain scheduling was also marked by a paper (see [113]) considering the problem of appropriate realization of a gain scheduling controller. This is conformable to the discussion of Section 1.3.1 where the five steps of gain scheduling control were detailed and is relevant with the final step of controller implementation. This premature paper was the beginning of a series of papers concerning this important issue as it will be detailed in the next subsection and triggered a controversy in the scientific community concerning classical and velocity-based gain scheduling (details for the latter are given in the next chapter).

[32]In the paper strictly proper dynamics are in fact assumed.

1.3.3.2 Mature Era

The mature era starts with the famous paper of R.A. Nichols, R.T. Reichert and W.J. Rugh (see [103]) for the gain-scheduled autopilot of an Air-to-air missile and is considered as a benchmark paper. It develops in fact ideas in the controller realization found in the previous cited paper [113]. However the full theoretical results are given in the important paper of D.A. Lawrence and W.J. Rugh in 1995 (see [79]).

Generic nonlinear controller

In this paper, the authors establish some very useful conditions for the gain scheduling controller realization to provide correct trim control *and* avoid the famous *coupling terms* that stem from the gain-scheduling practice. To illustrate this fact, suppose a nonlinear gain-scheduled controller to be written in the following *generic* form:

$$\dot{x}_k(t) = \mathbf{f}_k[x_k(t), x_i(t), \zeta(t), y(t), w(t), r(t)] \tag{1.114}$$

$$\dot{x}_i(t) = \mathbf{f}_i[x_k(t), \zeta(t), y(t), w(t), r(t)] \tag{1.115}$$

$$u(t) = \mathbf{f}_u[x_k(t), x_i(t), \zeta(t), y(t), w(t), r(t)] \tag{1.116}$$

where $\mathbf{f}_k, \mathbf{f}_i, \mathbf{f}_u$ are the nonlinear functions of the dynamic control, integral-error and output portions of the controller respectively[33] and $x_k, x_i, u, y, w, r, \zeta$ are the states of the dynamic control & integral error, control input, measured output, measured external parameter (used for scheduling), reference input and tracking error vectors. The setup is a servo problem one and the goal is to minimize (ideally nullify) the tracking error when the time-varying scheduling vector ϱ, being a nonlinear function of w, r and y, is taking values in a set $\mathbf{\Gamma}$[34].

Controller nonlinear functions

The realization of the nonlinear gain-scheduled controller functions that is proposed by the authors has the following form:

$$\begin{aligned}\mathbf{f}_k =& \mathbf{A}_{kk}(\varrho)[x_k - x_{k,\text{eq}}(\varrho)] + \mathbf{A}_{ki}(\varrho)[x_i - x_{i,\text{eq}}(\varrho)] + \\ & \mathbf{B}_{k\zeta}(\varrho)[\zeta - \zeta_{\text{eq}}(\varrho)] + \mathbf{B}_{ky}(\varrho)[y - y_{\text{eq}}(\varrho)] + \\ & \mathbf{B}_{kw}(\varrho)[w - w_{\text{eq}}(\varrho)] + \mathbf{B}_{kr}(\varrho)[r - r_{\text{eq}}(\varrho)]\end{aligned} \tag{1.117}$$

$$\begin{aligned}\mathbf{f}_i =& \mathbf{A}_{ik}(\varrho)[x_k - x_{k,\text{eq}}(\varrho)] + \\ & \mathbf{B}_{i\zeta}(\varrho)[\zeta - \zeta_{\text{eq}}(\varrho)] + \mathbf{B}_{iy}(\varrho)[y - y_{\text{eq}}(\varrho)] + \\ & \mathbf{B}_{iw}(\varrho)[w - w_{\text{eq}}(\varrho)] + \mathbf{B}_{ir}(\varrho)[r - r_{\text{eq}}(\varrho)]\end{aligned} \tag{1.118}$$

$$\begin{aligned}\mathbf{f}_u =& \mathbf{C}_{uk}(\varrho)[x_k - x_{k,\text{eq}}(\varrho)] + \mathbf{C}_{ui}(\varrho)[x_i - x_{i,\text{eq}}(\varrho)] \\ & \mathbf{D}_{u\zeta}(\varrho)[\zeta - \zeta_{\text{eq}}(\varrho)] + \mathbf{D}_{uy}(\varrho)[y - y_{\text{eq}}(\varrho)] + \\ & \mathbf{D}_{uw}(\varrho)[w - w_{\text{eq}}(\varrho)] + \mathbf{D}_{ur}(\varrho)[r - r_{\text{eq}}(\varrho)].\end{aligned} \tag{1.119}$$

[33]Note that the integral part is particularly important as it ensures appropriate error tracking when the scheduling vector is varying.

[34]This setup is rather complicated and may be simplified for particular applications but encompasses most cases.

Now the above equations[35] have a linear form with respect to deviations from equilibrium values and should provide the appropriate control input (trim control) for equilibrium operation. In addition they linearize to an LTI controller with matrices $\mathbf{A}_{kk}(\varrho_{\mathrm{eq}}), \ldots, \mathbf{D}_{ur}(\varrho_{\mathrm{eq}})$ for *fixed* values of the scheduling vector that may be designed for a collection of equilibrium points in the operating domain $\mathbf{\Gamma}$ of the system.

Coupling terms

The gain-scheduled controller however has to update the control input as a function of the scheduling vector and thus one should consider its behavior when $\varrho = \varrho(t)$ is not constant. Indeed, in this case *the scheduling vector is function of w, r, y and when linearizing Eqs. 1.117-1.119 there appear additional terms besides the ones corresponding to the linear controllers designed for constant equilibrium point operation.* These terms are the famous **hidden coupling terms**, they are time-varying and add up to the overall system dynamics causing problems to the closed loop operation. It turns out from the analysis in [79] that the inclusion of the integral-error component in the nonlinear controller (see Eq. 1.115) guarantees with appropriate use the existence of a controller *without the coupling terms*. Examples for cases that this may be achieved can be found in [79] or even in the survey of [114].

The second series of articles concerning linearization-based gain scheduling starts to appear in the late 90's with the work of D.J. Stilwell. The first paper (see [132]) addresses a similar problem initially discussed in [118]. A parameter-dependent system $\mathcal{S}_{\mathrm{pd}}$ as the one in Eq. 1.70 is considered, as well as an LPV one as in Eq. 1.72 stemming from linearization around equilibrium points dictated by the scheduling variable.

Stability covering condition

Given now this LPV system, an observer/state feedback control setup is assumed (see Section 1.3.2.6). The issue addressed is how to compute the control matrices $\mathbf{K}_{\mathrm{c}}, \mathbf{K}_{\mathrm{o}}$ as a function of the scheduling vector so as to guarantee stability of the LPV system, both for fixed and varying values of ϱ. It turns out that controllers are calculated at a number of points in the operating domain $\mathbf{\Gamma}$ of the system satisfying a certain *stability covering condition* and then are interpolated in a specific way that guarantees closed loop stability up to a certain extent of variation rate of the scheduling vector. Concerning this condition[36], suppose that a set $\Sigma(K_{\mathrm{LTI}}) = [\mathbf{K}_{\mathrm{c}}^1, \ldots, \mathbf{K}_{\mathrm{c}}^k]$ of state feedback gains are computed for corresponding constant values of the scheduling vector[37] $[\varrho^1, \ldots, \varrho^k] \in \mathbf{\Gamma}$ in such a way that each gain stabilizes the LTI plant (i.e. $\mathbf{A}(\varrho^i)+\mathbf{B}(\varrho^i)\mathbf{K}_{\mathrm{c}}^i$ is stable) obtained by a frozen LPV one for $\varrho = \varrho^i$. In addition each gain also stabilizes the LPV plant inside a region (i.e. $\mathbf{A}(\varrho) + \mathbf{B}(\varrho)\mathbf{K}_{\mathrm{c}}^i$ is stable) Γ^i with $\varrho^i \in \Gamma^i$. *If* $\mathbf{\Gamma} \subset \bigcup_{i=1}^{k} \Gamma^i$ (their intersection is not necessarily empty) then the family of gains is said to satisfy the *stability covering condition*.

[35]The 'eq' notation means equilibrium operation for fixed values of the scheduling vector.

[36]Pure state feedback is assumed since the analysis for the full observer-based state feedback proceeds in an analogous manner.

[37]Suppose that $\varrho \in \mathbb{R}^1$ for simplicity.

This condition means in brief that the LPV plant must be stabilized at (possibly mutually overlapping) regions, whose union overbounds the operating region of the plant, using a single feedback gain per region. Of course at overlapping regions the LPV system may be stabilized by more than one gain. See for example Fig. 1.13 where a two dimensional scheduling vector is considered and three overlapping scheduling regions (light shading) for each one of the three synthesis points (red stars) inside the rectangular operating domain $\mathbf{\Gamma}$ of the plant. Three regions (dark shaded) exhibit overlapping of two controllers whereas one (small central blue region) exhibits overlapping of all controllers.

Returning to the one dimensional scheduling vector discussion, the global gain-scheduled controller will apply only one stabilizing LTI controller (robust functioning) for non-overlapping regions whereas in overlapping ones a gain blending approach is used that guarantees stability and control signal continuity. For more details on this particular type of interpolation, refer to [132], pp.1227, Theorem II.2.

`SS & NCF interpolation`

Another paper from the same authors (see [133] or equally [131]) generalizes the results of the previous paper [132] for two other cases. The first is *state space matrix interpolation* (as in Section 1.3.2.5) and the second one is a particular type of interpolation using a *normalized coprime factor* development of the controllers. As it has been already noted, the first method has several computational and other disadvantages. The second one has not received practically any attention in the scientific communities and it would may be interesting in a $\mathscr{H}_\infty$ loop-shaping control context.

`J-Q method`

Finally a very interesting theoretically justified interpolation approach uses LTI controllers of arbitrary structure (but of the same order) that may all be parameterized by a *single* dynamic system $\mathscr{J}$ and different stable Youla parameters $\mathscr{Q}^i$ [38]. The method uses once again the stability covering condition and interpolates the state space matrices of the Youla parameter in order to obtain a gain-scheduled controller. This article (see [129] or equally [128]) also offers a missile autopilot example in order to justify the practicality of the approach.

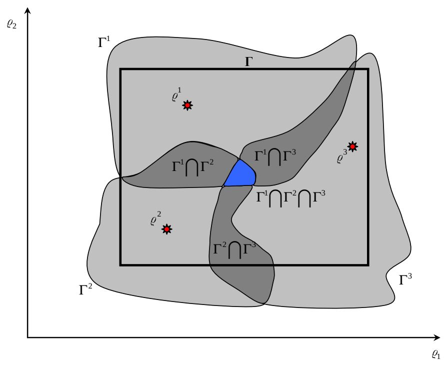

ϱ_2
Γ^1
$\mathbf{\Gamma}$
ϱ^1
$\Gamma^1 \bigcap \Gamma^2$
$\Gamma^1 \bigcap \Gamma^3$
ϱ^3
$\Gamma^1 \bigcap \Gamma^2 \bigcap \Gamma^3$
ϱ^2
$\Gamma^2 \bigcap \Gamma^3$
Γ^2
Γ^3
ϱ_1

Figure 1.13: Stability covering regions.

[38]See also Section 1.3.2.6.

1.3.3.3 Modern Approaches

The modern era on gain scheduling seems to have started by the two surveys on the subject of gain scheduling in general published the same year (see [88, 114]). These surveys cover most gain-scheduled control schemes (linearization-based, LPV-based, fuzzy etc.) known to the scientific community and offer some good insights on how the gain scheduling research context should be defined in the future. However, one can distinguish a bias to velocity-based gain scheduling (see Chapter 2) in the first, and to linearization-based gain scheduling (with the appropriate controller realizations detailed in Section 1.3.3.2) in the second, mostly due to the corresponding to either methods work, of each pair of authors. Despite this fact, these survey should be taken as a landmark for future work.

A piece of work that should be noted is the extension of the one by D.J. Stilwell and D.A. Lawrence detailed in the previous section on sampled-data systems and it may be found in [130]. In the paper it is shown that certain linearization properties of a continuous time gain-scheduled plant *carry on* to the sampled data one, a fact that is important for real world systems.

Finally, a work that stands out from the rest in the gain-scheduling control community can be found in [40] where it is shown that the control objectives of a gain-scheduled controller can be expressed as the weighted incremental norm minimization of a nonlinear operator. However this method has yet to prove its effectiveness in practical situations.

1.4 Operating Domain Issues

In this section some tools used for controller interpolation will be presented. It has been already remarked that the 2nd and 3rd steps of the linearization-based gain scheduling procedure involve linearization of the nonlinear plant at a certain set of equilibrium points inside its operating domain $\mathbf{\Gamma}$. These points emerge from constant (or frozen-time) values of the scheduling vector $\varrho(t)$ and the LTI systems obtained are used in order to obtain a set of LTI controllers $\Sigma(K_{\mathrm{LTI}})$. Then the LTI controllers are somehow blended to form a gain-scheduled controller. The latter is done by dividing the operating region $\mathbf{\Gamma}$ to appropriate scheduling regions Γ^i considering a number of neighbor controllers being placed at the edges of these regions. Until now, the issues of both choosing these points in a systematic way and forming the regions have been avoided.

`Operating domain gridding`

Given that very often the operating domain $\mathbf{\Gamma}$ of a system may be considered/rendered convex, an equidistant gridding is performed (see Fig. 1.11) with no idea on the density of the considered points. As a result, either only region corners are considered, or the gridding is performed in a trial-error manner until satisfactory performance is obtained[39].

[39]An exception is the work found in [132] but it remains rather conservative in terms of the maximum of the scheduling vector rate of variation.

Thus, the designer is not sure if he has considered too many or too few points in his study. Additionally, it is not obligatory that the points be taken equidistantly or even in a rectangular gridding[40]. This problem is addressed for the first time in this thesis using appropriate operating point selection algorithms as it will be detailed in Chapter 5.

The result of these algorithms are a number of operating points dispersed in an *non-uniform* way on the operating domain $\mathbf{\Gamma}$. Thus, for implementation purposes, a way to define scheduling regions Γ^i should be found. It turns out that the rectangular ones are nor redundant, given that a plane may be defined using only three points and not four, nor the interpolation is straightforward since the operating points will not be in general aligned. As a result, the more efficient way should be to triangulate the operating domain of the system and interpolate the controllers in triads (see Fig. 1.14).

`Triangulation`

An efficient way to triangulate a convex polygonal domain is the famous *Delaunay Triangulation.* A triangulation $\triangle$ of a convex domain $\mathbf{\Gamma}$ is a Delaunay triangulation provided that for every triangle Γ^j in $\triangle$, there is no vertex $\varrho^i \notin \Gamma^j$[41] inside the circumcircle around Γ^j. An extension for non-convex operating domains can be found in [78], pp. 109-110. A related notion are the *Voronoi Diagrams* which are polygonal domains formed around each vertex ϱ^i and define a certain zone of influence around every ϱ^i.

Technical Note. In the context of Gain Scheduling, given a set of points, one can use the MATLAB function `TRI=delaunay`(ϱ_x, ϱ_y), where ϱ_x, ϱ_y are the one to one ordered vectors of the x and y coordinates of all points, in order to obtain a Delaunay triangulation. Then, for any value $\varrho(t)$ of the scheduling vector the function `tsearch` (taking also as inputs all the triangle edges triplets `TRI`) gives the corresponding triangle (or else scheduling region Γ^j). Thus, controller interpolation may be performed by using the LTI controllers stored for every edge of the triangle chosen (see Figure 1.15 for an example of Delaunay triangulation and Voronoi tesselation).

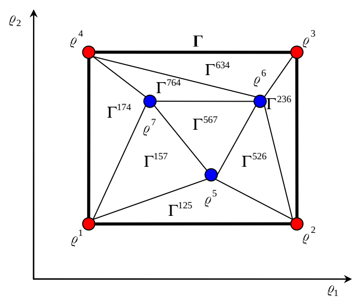
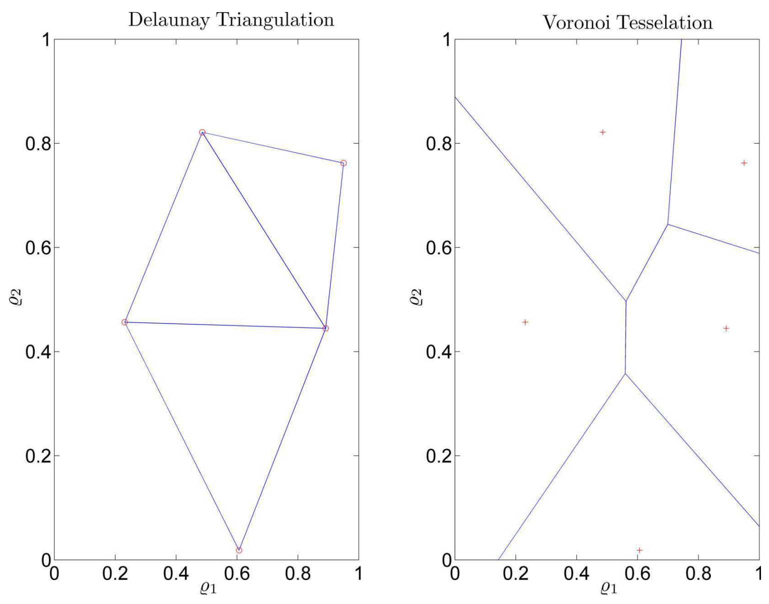


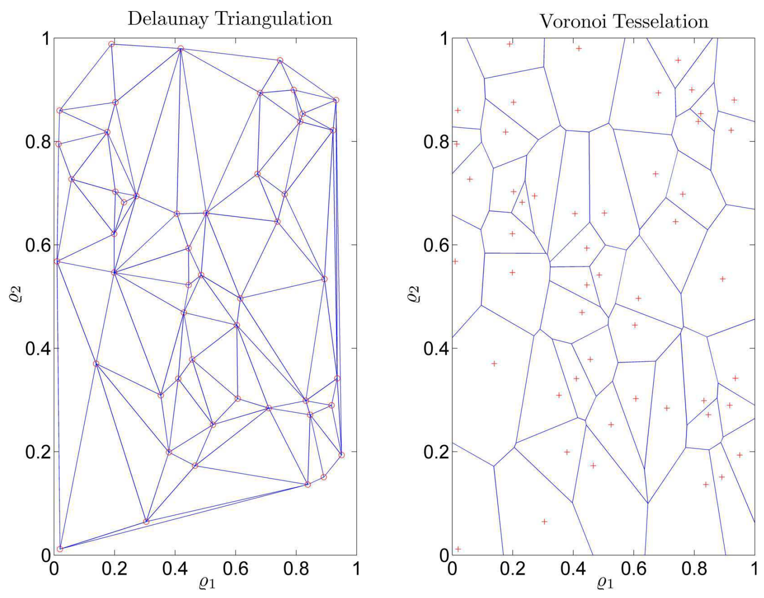
Figure 1.14: Operating domain triangulation.

[40]Planar operating regions are considered for simplicity.

[41]Here it is supposed that the vertices are dictated by constant values of the scheduling vector ϱ^i on the operating domain $\mathbf{\Gamma}$ of the system.



(a) *Five points*



(b) *Fifty points*

Figure 1.15: Delaunay triangulation and Voronoi Tesselation.