From Linearization to Lazy Learning: A Survey of Divide-and-Conquer Techniques for Nonlinear Control (Invited Paper)

Gianluca Bontempi and Mauro Birattari

***Abstract*— In the field of system identification and control a mismatch exists between the available theoretical tools and most of the problems encountered in practice. On the one hand, researchers developed plenty of theoretical analysis and methods concerning linear systems; on the other hand practitioners are often confronted with the apparent nonlinearity of the real world. Although several nonlinear identification and control techniques have been proposed in the last decades, these still appear to be less robust and reliable than their linear counterparts. An appealing approach to bridge the existing gap consists in decomposing a complex nonlinear control problem in a number of simpler linear problems, each associated with a restricted operating region. This paper will review a number of *divide-and-conquer* techniques proposed in the nonlinear control literature and more recently in the machine learning community to address the problem of linearly controlling a nonlinear system. Two families of divide-and-conquer approaches will be taken into consideration: *analytical* approaches which require the knowledge of the system dynamics and *learning* approaches which rely on powerful approximators to estimate a model from input/output data.**

***Index Terms*— Nonlinear control, divide-and-conquer approaches, lazy learning.**

I. Introduction

THE idea of adopting a linear approach for nonlinear problems is not new in the literature. The persisting use of linear techniques, as the PID controllers [24], in a large part of industrial plants is an obvious demonstration of the effectiveness and the reliability of linear control theory for a number of real tasks. Anyway, in most cases these control systems require the plant to be in a stationary regime and are inefficient in transient operating conditions.

In recent years, the issue of controlling nonlinear plants over a wide range of operating conditions has made of *divide-and-conquer* approaches an appealing solution. These approaches decompose a complex nonlinear control problem in a number of simpler problems, each associated with a restricted operating region. This can often give a more manageable and transparent representation of the control system, leaving at the same time the opportunity of reusing conventional linear techniques in nonlinear tasks.

This paper provides the reader with a state-of-the-art of the methods that reuse the linear approach for nonlinear problems and surveys the existing divide-and-conquer control strategies. For the sake of clarity, we will distinguish between two main classes of approaches to divide-and-conquer control: the *analytical* approaches and the *learning* approaches. While the first group of approaches requires the knowledge of the analytical form of the dynamical system to be controlled, the learning approaches aim at synthesizing a control policy only on the basis of a number of input-output observations.

Three main analytical divide-and-conquer approaches will be discussed and compared in the paper (Figure 1):

- Linearization: this is the most intuitive and historically the first approach to reuse linear techniques in a nonlinear control setting [13].
- Gain scheduling: this is probably the most systematic approach to the control of nonlinear systems in practice [46], [42]. Along the years, it remains an attractive control strategy for its experimented advantages in terms of simple design and low computational complexity.
- Feedback linearization: this control strategy, introduced in recent years [48], [29], transforms the closed-loop problem in a linear control problem, more manageable in theoretical and practical terms.

When no analytical model is available but the assumption of linearity is reasonable, linear system identification techniques are used to estimate the system on the basis of observed data [26]. Linear techniques have been also employed to deal with nonlinear dynamics by making use of adaptive algorithms [17]. Here an on-line identification algorithm (Recursive Least Squares) updates a linear approximation of the system, using a *forgetting factor* in order to track variations in the dynamics. However, such an approximation provides a satisfactory performance only if the operating regime changes slowly. The idea of learning techniques (Figure 2) is to approximate the unknown relation between state of the system and control actions by using nonlinear approximation techniques. The traditional approach to supervised learning is *global* modeling. Global models make the assumption that the relationship between the inputs and the output values can be described by an analytical function over the whole input domain. Examples of functional estimators are linear models, nonlinear statistical

Manuscript received October 15, 2003; revised October 16, 2003.

Gianluca Bontempi and Mauro Birattari, Université Libre de Bruxelles Brussels, Belgium. [gbonte@ulb.ac.be (G. Bontempi), mbiro@ulb.ac.be (M. Birattari)]

Publisher Item Identifier S 1542-5908(05)10106-7/$20.00

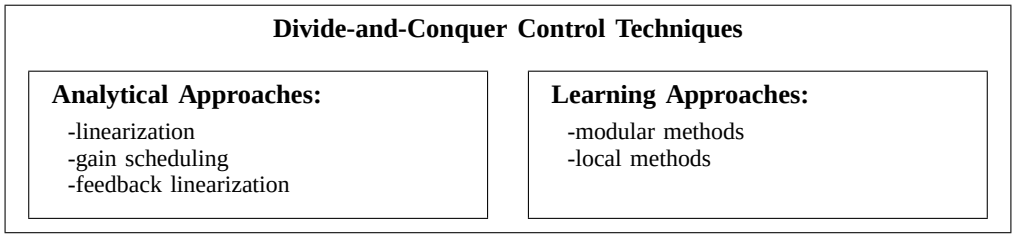


Fig. 1. A pictorial representation of the taxonomy of the divide-and-conquer control techniques discussed in the paper. A more exhaustive classification of supervised learning techniques is proposed in Figure 2.

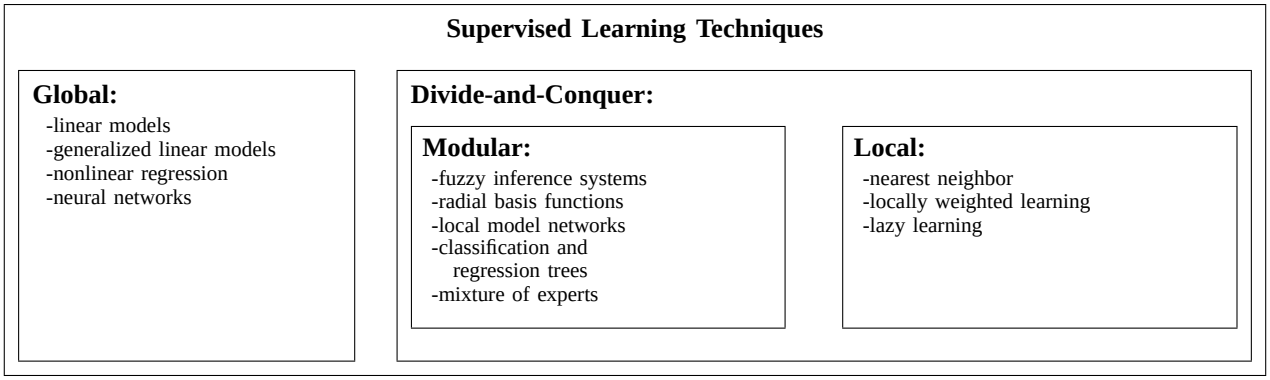


Fig. 2. A pictorial representation of the taxonomy of the supervised learning techniques.

regressions [45], and Neural Networks [43].

The *divide-and-conquer* paradigm in supervised learning originates from the idea of relaxing the global modeling assumptions. It attacks a complex problem by dividing it into simpler problems whose solutions can be combined to yield a solution to the original problem. This principle presents two main advantages. The first is that the choice of the model complexity (*structural identification*) and the estimation of parameters (*parametric identification*) can rely on linear techniques, well studied and developed over the years. The second is that the learning method can better adjust to the properties of the available dataset.

The divide-and-conquer idea has taken two different forms: the *modular architectures* and the *local modeling* approach.

Modular techniques replace a global model with a modular architecture where the modules cover different parts of the input space. This is the idea of *operating regimes* which assume a partitioning of the operating range of the system in order to solve modeling and control problems [20]. Fuzzy Inference Systems [50], Radial Basis Functions [30], [38], Local Model Networks [31], Classification and Regression Trees [12], and Mixture of Experts [23] are well-known examples of this approach. Although modular architectures are a combination of local models, their identification is still performed on the basis of the whole dataset. Hence, the learning procedure remains a function estimation problem, with the advantage that the parametric identification can be made simpler by the adoption of local linear modules. However, in terms of structural identification the problem is still nonlinear and requires the same procedures used for generic global models.

An alternative example of divide-and-conquer methods are *local modeling* techniques [16], which turn the problem of function estimation into a problem of value estimation. The well-known nearest neighbor method can be considered the prototype for this class of techniques: Local methods do not aim to return a complete description of the input/output mapping but rather to approximate the function in a neighborhood of the point to be predicted (Figure 3). The local feature makes possible the adoption of linear techniques both in parametric and structural identification with a gain in terms of analytical tractability and fast design. Locally weighted learning techniques [3] are well-known instances of local modeling. Among them we mention the Lazy Learning (LL) algorithm for modeling and control developed by the authors in [10], [11], [7]

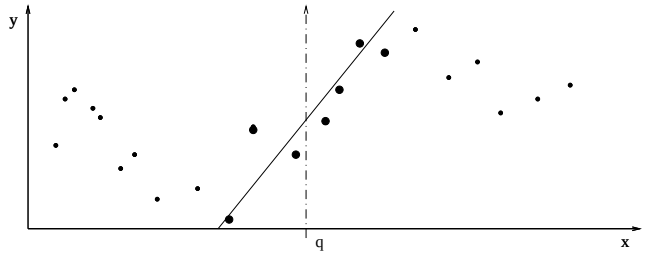


Fig. 3. Local modeling of the input/output relationship between the input variable x and the output variable y, on the basis of a finite set of observations (dots). The value of the variable y for $x = q$ is returned by a linear model (solid line) which fits the samples in a neighborhood of the query point (bigger dots).

Consistently with the above classification we will present in this survey two learning approaches to divide-and-conquer control.

Local modular approach: the idea consists in approximating the dynamics of the system by using a modular locally-linear representation and in exploiting this information to control the system by adopting locally linear techniques. Local Model Networks [20], [21] and the techniques of Fuzzy Control [35] belong to this category. Three ways of combining the local linear representations (local controller, local gain scheduler and self-tuning controller) will be reviewed.

Lazy Learning approach: this approach combines a Lazy Learning approximator, which performs the on-line local linearization of the system, with conventional linear control techniques (minimum variance, pole placement, optimal control). To make the presentation more concise, we will limit ourselves to present a LL self-tuning approach. Other examples of LL approaches are presented in [10], [4]. Note that each LL approach can be considered as an *intrinsically adaptive* control methods. This derives from the fact that, at each time step, the input/output observation is added at no computational cost to the stored dataset, adapting the control policy to changing dynamics. It is important to remark that the adaptive feature requires no modification of the control algorithm unlike nonlinear global techniques where heavy recursive formulations are generally necessary to cope with sequential stream of data.

This is the outline of the survey paper: After an introductory statement on the nonlinear control problem in Section II, Section III discusses the linearization method. Section IV and V introduces gain scheduling and feedback linearization, respectively.

Local modular approaches will be described in Section VI. A self-tuning Lazy Learning regulator will be described in Section VII. A summary comparison of the approaches according to a series of criteria is presented in Section VIII.

II. Nonlinear Systems and Equilibrium Points

A continuous nonlinear dynamic system [27] is represented by a set of differential equations in the form

$$\dot{x} = f\big(x(t), u(t), t\big), \tag{1}$$

where f is a $[n \times 1]$ nonlinear vector function $f\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \rightarrow \mathbb{R}^n$, x is the *state* vector in $\mathbb{R}^n$ and $u \in \mathbb{R}^m$ is the *control input* vector.

Linear systems can be seen as a special class of nonlinear systems. The dynamics of linear systems is represented by

$$\dot{x} = A(t)x(t) + B(t)u(t), \tag{2}$$

where $A(t)$ is a $[n \times n]$ matrix and $B(t)$ is a $[n \times m]$ matrix.

A linear system can be classified as either *time-varying* (LTV) or *time-invariant* (LTI), depending on whether the system matrix A varies with time or not. In the nonlinear context, these terms are replaced by "autonomous" and "non-autonomous".

Definition 1 (Autonomous systems): The nonlinear system (1) is said to be autonomous if the function f does not depend explicitly on time, i.e., if the system's state equation can be written in the form

$$\dot{x} = f(x, u), \tag{3}$$

Otherwise, the system is called non-autonomous.

III. Linearization

Linearization methods are the simplest way to adopt linear control techniques in a nonlinear setting. In this section, we will distinguish between *linearization about an equilibrium point* and *linearization about a trajectory*.

A. Linearization About an Equilibrium Point

We introduce first the definition of equilibrium point:

Definition 2 (Equilibrium point): The pair (x_0, u_0) is an equilibrium of the system (3) if once $x(t)$ is equal to x_0 and $u(t)$ is equal to u_0 for all future time, $x(t)$ remains equal to x_0 for all future time. In mathematical terms this means that

$$f(x_0, u_0) = 0. \tag{4}$$

Consider the autonomous system (3), and assume that f is continuously differentiable at the equilibrium (x_0, u_0). Then, the system dynamics can be written as

$$\begin{aligned} \dot{x}(t) &= f\big(x(t), u(t)\big) = \\ &= f(x_0, u_0) + G_{1f}(x_0, u_0)\big(x(t) - x_0\big) \\ &\quad + G_{2f}(x_0, u_0)\big(u(t) - u_0\big) + r\big(x(t), u(t)\big), \end{aligned} \tag{5}$$

where $r\big(x(t), u(t)\big)$ stands for higher-order terms in $\big(x(t), u(t)\big)$ and G_{1f} and G_{2f} are defined in Appendix . Define the deviation from equilibrium terms as

$$\tilde{x}(t) = x(t) - x_0, \tag{6}$$

$$\tilde{u}(t) = u(t) - u_0. \tag{7}$$

Then, the system

$$\dot{\tilde{x}}(t) = G_{1f}(x_0, u_0)\tilde{x}(t) + G_{2f}(x_0, u_0)\tilde{u}(t) \tag{8}$$

is called the *linearization* (or *linear approximation*) of the original nonlinear system (3) about the equilibrium point (x_0, u_0).

B. Linearization About a Trajectory

Let $x = x^*(t)$ and $u = u^*(t)$ satisfy the equation (3) with f continuously differentiable, that is $\forall t : \dot{x^*}(t) = f\big(x^*(t), u^*(t)\big)$. We call $\big(x^*(t), u^*(t)\big)$ a *trajectory* of the nonlinear system.

We examine the behavior of the system near this trajectory and we define

$$\begin{aligned} \tilde{x}(t) &= x(t) - x^*(t), \\ \tilde{u}(t) &= u(t) - u^*(t). \end{aligned} \tag{9}$$

The function f near the trajectory may be approximated by

$$\begin{aligned} f(x(t), u(t)) \;\approx\; & f\big(x^*(t), u^*(t)\big) + G_{1f}\big(x^*(t), u^*(t)\big)\tilde{x}(t) \\ & + G_{2f}\big(x^*(t), u^*(t)\big)\tilde{u}(t) \end{aligned} \tag{10}$$

then leading to the linear time-varying dynamics:

$$\dot{\tilde{x}}(t) = G_{1f}\big(x^*(t), u^*(t)\big)\tilde{x}(t) + G_{2f}\big(x^*(t), u^*(t)\big)\tilde{u}(t). \tag{11}$$

In the next section, we show that the linearized dynamics (8) and (11) may be used to infer properties of the nonlinear system in a neighborhood of an equilibrium point and of a known trajectory, respectively.

C. Linearization and Local Stability

In this section we report the conditions of stability for a dynamic system, when a linearization about a point or a trajectory is given.

As seen in the previous sections, a linearization about an equilibrium returns a time invariant linear dynamics while a linearization about a trajectory returns a linear time-varying (LTV) system. Different stability analysis are then required.

1) Linearization About an Equilibrium and Local Stability: Consider the equilibrium x_0 of an unforced system

$$\dot{x}(t) = f\big(x(t)\big) \tag{12}$$

that satisfies $f(x_0) = 0$. The following result makes precise the relationship between the stability of the linearized system and the original nonlinear system [48].

Theorem 1 (Lyapunov's linearization method):

- If the linearized system is strictly stable (i.e. all of the eigenvalues of $G_{1f}(x_0)$ have negative real parts), then the equilibrium point x_0 is asymptotically stable for the nonlinear system.
- If the linearized system is unstable (i.e. at least one of the eigenvalues of $G_{1f}(x_0)$ is in the right-half complex plane), then the equilibrium point x_0 is unstable for the nonlinear system.
- If the linearized system is marginally stable (i.e. all of the eigenvalues of $G_{1f}(x_0)$ have negative real parts but at least one is on the imaginary axis), then one cannot conclude anything from the linear approximation about the stability of the nonlinear system.

2) Linearization About a Trajectory and Local Stability: The stability analysis of the linearization of an autonomous system about a trajectory can be reduced to the analysis of the linearization of a non-autonomous system about the equilibrium point. As in the equilibrium case, consider an unforced system (12) for which a solution $x^*(t)$, with initial condition $x^*(0) = x_0^*$, is available.

Let us now perturb the initial condition to be $x(0) = x_0^* + \delta x_0$ and study the associated variation of the motion error

$$\tilde{x}(t) = x(t) - x^*(t). \tag{13}$$

Since both $x^*(t)$ and $x(t)$ are solutions of (12), we have that $\tilde{x}(t)$ satisfies the following non-autonomous differential equation

$$\dot{\tilde{x}}(t) = g(\tilde{x}, t), \tag{14}$$

whose equilibrium point is at the origin. Then each particular nominal motion of an autonomous system corresponds to an equivalent non-autonomous system. Therefore, we will refer to the theorems concerning the linearization method for non-autonomous systems about the origin. Let $A(t)$ be the linearization of the system (14) around the equilibrium 0. Then the following theorem states:

Theorem 2: If the linearized system $A(t)$ is uniformly asymptotically stable, then the equilibrium point 0 of the original non-autonomous system is also uniformly asymptotically stable.

D. Some Considerations on Linearization

Linearization methods are easy to interpret and shed a light on the behavior of a nonlinear system about some operating conditions. However, they present some drawbacks:

- a control design based on the linearized dynamics could have stability problems when operating away from the equilibrium or trajectory;
- the equilibrium points and/or the trajectories must be known in advance and this knowledge is often not available.

In the next section, we will present a design approach which addresses the restrictions of linearization: the *gain scheduling*.

IV. Gain-scheduled Control Design

The idea of gain scheduling [46], [42] consists in breaking the control design process into two steps. The first step designs a set of local linear controllers at different equilibria, called *operating points*. In the second step, a global nonlinear controller is obtained by interpolating (scheduling) the set of local controllers. The resulting global controller is called a *gain scheduler*.

Let us see this approach in detail. Consider the nonlinear system

$$\dot{x}(t) = f(x(t), u(t), s), \tag{15}$$

$$y(t) = g(x(t)), \tag{16}$$

where $y(t)$ denotes the measured output, and s, called the *scheduling* variable, is a continuous and measurable quantity. The gain scheduling approach consists in

1) defining a finite set $S = \{s_1, s_2, \ldots s_m\}$ of m representative values of the variable s,
2) decomposing the original system in m local subsystems parameterized by the variable s,
3) designing a controller for each local system and
4) controlling the system by interpolating or switching between the m controllers.

A typical application of gain scheduling is flight control, where the state x of the system contains the position and speed of the aircraft and the scheduling variable s measures the velocity of the external wind.

In real cases, a main issue is how to find suitable scheduling variables. This is normally done based on the knowledge of the physics of the system and considering the following alternatives.

- State variables or a subset of them.
- Exogenous variables: these are variables that would be state variables in an extended system which is not generally modeled because of its complexity.
- Reference state trajectories: in this case it is implicit the assumption that the system state is near to the reference command.

Once scheduling variables have been chosen, the controller parameters are designed at a number of operating conditions, using some suitable design method.

Assume that the family of equilibrium points (x_{eq}, u_{eq}) is parameterized by the variable s, i.e.

$$0 = f\big(x_{eq}(s), u_{eq}(s)\big). \tag{17}$$

For each value s_j, $j = 1, \ldots, m$, gain scheduling linearizes the original system about the equilibrium $x_{eq}(s_j)$. We obtain a family of m linearizations, also called *frozen* configurations,

$$\dot{\tilde{x}}(t) = A(s_j)\tilde{x}(t) + B(s_j)\tilde{u}(t), \tag{18}$$
$$\tilde{y} = C(s_j)\tilde{x}(t), \tag{19}$$

with

$$A(s_j) = G_{1f}\big(x_{eq}(s_j), u_{eq}(s_j)\big), \tag{20}$$
$$B(s_j) = G_{2f}\big(x_{eq}(s_j), u_{eq}(s_j)\big), \tag{21}$$
$$C(s_j) = G_g\big(x_{eq}(s_j)\big), \tag{22}$$
$$\tilde{x}(t) = x(t) - x_{eq}(s_j), \tag{23}$$
$$\tilde{u}(t) = u(t) - u_{eq}(s_j), \tag{24}$$
$$\tilde{y}(t) = y(t) - g\big(x_{eq}(s_j)\big). \tag{25}$$

Once a linearized system is detectable and stabilizable, a linear controller can be designed by using a variety of linear design methods. The result is a parameterized family of linear controllers:

$$\dot{z}(t) = \bar{A}(s_j)z(t) + \bar{B}(s_j)y(t), \tag{26}$$
$$u(t) = \bar{C}(s_j)z(t) + \bar{D}(s_j)y(t). \tag{27}$$

Upon operation of the control system, the variable s is measured and used to infer to which of the m conditions the system is nearest. Two are the methods of proceeding:

Discontinuous (switching) In this case the domain of s is divided into a set of regions R_j, $j = 1, \ldots, m$, each containing a representative value s_j of the scheduling variable. As a consequence, the control system shifts between a finite number of controllers indexed by the respective scheduling value.

Smooth For each value of s the controller matrices are obtained by interpolating the m local controllers. The contribution of the j^{th} controller is a function of the distance between s and s_j.

Note that although each controller (26) is designed for a fixed value of s (frozen design) and is based on a linear time-invariant approximation to the plant, the resulting controller is time-varying. This causes some difficulties in terms of stability analysis.

A. Gain Scheduling and Stability

For each fixed value of the scheduling variable, the designer can guarantee the desired properties of stability, performance and robustness of the closed loop system. However, being the resulting control system time-varying, it does not necessarily inherit any of the frozen time properties. This means that one cannot assess a priori the properties of the gain scheduled design starting from the frozen time design but that a further analysis is required.

Shamma [46] introduced the formalism of *linear parameter-varying* systems (LPV) to study the behavior of gain scheduled control systems. A linear parameter varying system is defined as a linear system whose coefficients depend on an exogenous time-varying parameter $\theta(t)$:

$$\dot{x} = A\big(\theta(t)\big)x(t) + B\big(\theta(t)\big)u(t), \tag{28}$$
$$y(t) = C\big(\theta(t)\big)x(t). \tag{29}$$

The exogenous parameter θ is unknown a priori; however it can be measured or estimated upon operation of the system. This distinguishes LPV from LTV systems where the time variations are known beforehand.

The evident similarity between (18) and (28) makes of LPV a useful paradigm for the study of stability gain-scheduled controller. Gain scheduled controllers fix a set of controllers for a set of parameters values $\{\theta^1, \ldots \theta^m\}$ such that for all frozen values of the parameters, the closed loop has desired feedback properties. Since the parameters are actually time-varying, none of these properties need carry over to the overall closed loop system.

Shamma summarized the conditions which guarantee that the closed loop retain the feedback properties of the frozen time design. Suppose that one has carried out the gain scheduled design procedure. Then along any particular parameter vector trajectory, the closed loop unforced dynamics are of the form

$$\dot{x} = A_c(t)x(t), \tag{30}$$

where A_c represents the closed loop dynamics matrix. Let us make the following assumption:

Assumption 1: The dynamics matrix $A_c : \mathbb{R}^+ \to \mathbb{R}^{n\times n}$ is bounded and globally Lipschitz continuous with constant L_A, i.e.

$$\|A_c(t) - A_c(\tau)\| \le L_A\|t - \tau\| \quad \forall t, \tau \in \mathbb{R}^+. \tag{31}$$

Then the following theorem can be stated.

Theorem 3: Consider the linear system of (30) under the assumption (1). Assume that at each instant: (1) $A_c(t)$ is stable and (2) there exist μ and $\lambda \ge 0$ such that

$$\|e^{A_c(\tau)t}\| \le \mu e^{-\lambda t}, \quad \forall t, \tau \ge 0. \tag{32}$$

Under these conditions, given any $\eta \in [0, \lambda]$,

$$L_A \le \frac{(\lambda - \eta)^2}{4\mu \ln \mu} \Rightarrow \|x(t)\| \le \mu e^{-\eta t}\|x_0\| \qquad \forall t \ge 0, x_0 \in \mathbb{R}^n. \tag{33}$$

The above theorem states that a time-varying system retains its frozen-time exponential stability provided that the time-variations of the dynamics are sufficiently slow.

The reasoning behind the theorem is that a time varying dynamics stabilized for a set of frozen time instants can be approximated by a piecewise constant dynamics. On each piecewise constant interval, the linear system can be decomposed in a LTI stable system and a time-varying perturbation. Along this time interval, the state will decay exponentially but may experience a certain amount of amplification before its eventual decay. The final global behavior is related to

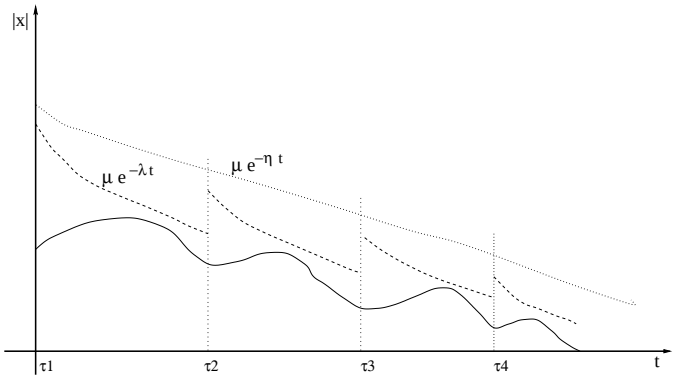


Fig. 4. Gain scheduling: stable closed loop for a stable frozen design at the instants τ_i. The sequence of stable dynamics with sufficient decays (dashed lines) induces a global behavior of state x (solid line) which is bounded by a stable trajectory (dotted line).

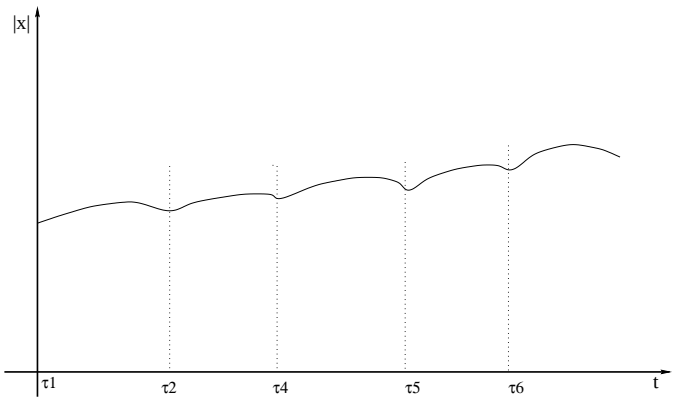


Fig. 5. Gain scheduling: unstable closed loop for a stable frozen design at the instants τ_i. The sequence of stable dynamics with insufficient rate of decay causes a global unstable behavior (solid line) of the state x.

how the sequence of amplifications/decays are combined. The system will result exponentially stable only if each LTI approximations will be long enough to allow sufficient decay (see Fig. 4 for a stable case and Fig. 5 for an unstable case). The eventual amplification is related to the parameter μ, called *overshoot*. In the special case of $\mu = 1$ none of the frozen-time systems experience any amplification. It follows that in this case the time variations can be arbitrarily fast with no instability occurring.

It is possible to have a sufficient condition for stability inferred from the worst-case values of μ and λ. Shamma [46] presented some experimental design where the Lyapunov method and the Matrix Exponential Method are used to test the overall stability of the gain scheduled controller.

B. Considerations on Gain Scheduling

The gain scheduling approach was the first approach in control design literature to address these relevant issues:

- how to extend linear methods to nonlinear control,
- how to control on larger operating regions than the neighborhood of a single equilibrium,
- how to solve the problem of introducing time variations in the overall control systems.

However, there are also a number of shortcomings and limitations associated with this approach:

- the designer must know a priori the location of the equilibrium points,
- the state of the nonlinear system must be close to one of the equilibrium points.

In the following sections we will see alternative approaches to the idea of extending linear control techniques to nonlinear control.

V. Feedback Linearization

Feedback linearization is an approach to nonlinear control design which has attracted a great deal of research in recent years [48], [29]. The main idea behind feedback linearization is to transform the nonlinear system model into a fully, or partially, linear model so that linear control techniques can be applied.

The idea of canceling the nonlinearities and imposing a desired linear dynamics, can be simply applied to a class of nonlinear systems described by the so-called *companion form* or *controllability canonical form* represented by

$$\dot{x} = f(x) + b(x)u, \tag{34}$$

where x is the state vector and u is the scalar control input. The dynamics (34) is linear in terms of the control input u but nonlinear in the states. Hence, by using the control input

$$u = \frac{1}{b}(v - f), \tag{35}$$

we can cancel the nonlinearities and obtain a simple input/output relation. When a nonlinear dynamics is not in the controllability canonical form, one may use algebraic transformations to first put the dynamics into the controllability form. Consider, for instance, a single input nonlinear system of the form

$$\dot{x} = f(x, u). \tag{36}$$

The technique of *input-state linearization* solves this problem in two steps. First one finds a state transformation $z = w(x)$ and an input transformation $u = g(x, v)$ so that the nonlinear system dynamics is transformed into an equivalent linear time-invariant dynamics

$$\dot{z} = Az + bv. \tag{37}$$

Second, one uses standard linear techniques to design v. A number of remarks should be made on this approach:

- The result, though valid in a large region of the input space, may not return a global solution. This may be due to the existence of singularity points.
- The input-state linearization is achieved by a combination of a state transformation and an input transformation with state feedback. It should be noted that this is a linearization by feedback, fundamentally different from the Jacobian linearization (5) which holds for small range operations.
- In general feedback linearization relies on the system model (34) both for the controller design and for the

computation of z. If there were uncertainty in the model, this would produce an inaccurate control action.

The feedback design can be extended also to general nonlinear systems. Consider the system in the state space form:

$$\dot{x} = f(x, u), \tag{38}$$

$$y = h(x), \tag{39}$$

and assume that our goal is to make $y(t)$ track a reference trajectory. The idea of *input/output linearization* is to find a direct relation between the system output y and the control input u. It consists basically in differentiating the output of a system d times, where d is called the *relative degree*, in order to generate an explicit relationship between the output and the input. Once this relation is established, one of the feedback methods sketched above can be applied.

A. Feedback Linearization and Stability

The effects of input/output linearization on the system dynamics has to be examined carefully. By means of input/output linearization the dynamics of a nonlinear system is decomposed into an external input/output part and an internal, not observable part. The latter will be called the *internal dynamics* because it cannot be seen from the external input/output relationship. If this dynamics is stable the input/output linearization solves the tracking problem, otherwise the control design is not effective.

In general, it is very difficult to determine the stability of the internal dynamics, due to nonlinear, non-autonomous and coupling effects. So far, global results have been obtained only by defining proper Lyapunov functions for the internal dynamics [48].

B. Considerations on Feedback Linearization

Feedback linearization can be used for stabilization and tracking control problems in SISO and MIMO configurations. So far, feedback linearization has been successfully applied to a number of practical nonlinear control problems. However, the method has a number of important limitations:

- it cannot be used for all nonlinear systems,
- the state has to be measured,
- no robustness is guaranteed in the presence of parameter uncertainty or unmodeled dynamics.

The approaches discussed so far demand a knowledge of the analytical form of the system dynamics in order to apply a linear control strategy. The remaining part of the survey will be dedicated to learning methods for divide-and-conquer control, which require no information about the system apart from a limited amount of input-output observations. The philosophy of these methods is to identify a nonlinear model of the system which could be easily and efficiently exploited by a linear controller.

VI. Local Modular Control

The local modular approach to control consists of two step: first, the identification of a locally-linear representation of the system dynamics and, second, the synthesis of a control strategy based on the adoption of locally linear techniques.

The idea of local modular control appeared first in literature thanks to the research on Fuzzy Control Systems [35].

Fuzzy control was initially proposed as an heuristic-based design technique [28]. The motivation for a fuzzy design of controllers is given by the fact that many industrial processes are typically controlled in one of the following two ways:

- a manual control by an human operator,
- an automatic control supported by a human operator.

In these situations it is useful to model the human control action in terms of fuzzy *if-then* rules in order to obtain a similar or better control system. However, this approach implies that a control algorithm or an *a priori* expert knowledge should pre-exist to the fuzzy design. Also, it is admitted in the literature that an heuristic-based design has a low reliability in MIMO problems, which represents the largest part of challenging applications. Finally, the heuristic-based design lacks systematic and formally verifiable tuning techniques, and the study of stability can only be done via extensive simulations.

Our survey will neglect the fuzzy heuristic approach and will focus instead on data driven modular techniques where the only information about the plant comes from a set of input/output observations. Local Model Networks is an example of identification method which combines the local modular philosophy with learning algorithms. This technique is briefly introduced in the following section.

A. Local Model Networks

Local Model Networks (LMN) were first introduced by Johansen and Foss [20] to model complex nonlinear relationships by generalizing the idea of Basis Function Networks [39]. Let us consider an unknown input-output mapping $f : \mathbb{R}^n \to \mathbb{R}$ where $\phi \in \Phi \subset \mathbb{R}^n$ is the input variable and $y \in \mathbb{R}$ is the output. A Local Model Network uses an approximator of the function $f(\cdot)$ having the form

$$\hat{y}(\phi) = \sum_{j=1}^{m} \rho_j(\phi, \eta_j) h_j(\phi, \alpha_j), \tag{40}$$

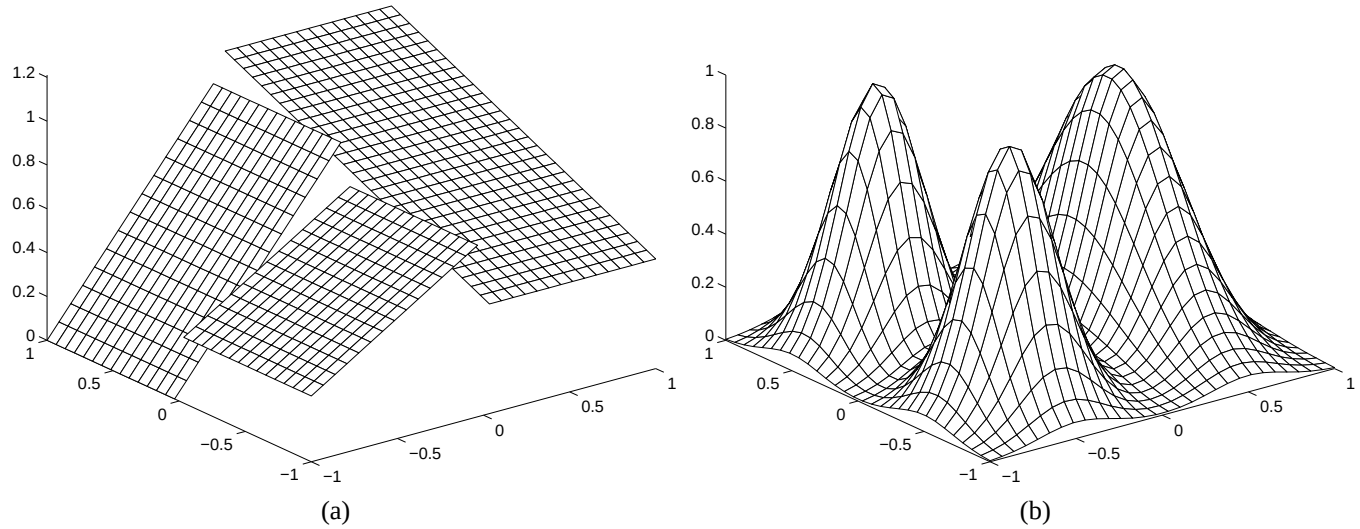
where the shape of functions $\rho_j(\cdot)$ and $h_j(\cdot)$, the number m of basis functions and the set of parameters η_j and α_j have to be estimated on the basis of a set of N training data.

The terms $\rho_j(\cdot)$ are denoted as *basis* or *activation* functions and are constrained to satisfy

$$\sum_{j=1}^{m} \rho_j(\phi, \eta_j) = 1 \quad \forall \phi \in \Phi. \tag{41}$$

This means that the basis functions form a *partition of unity* of the input domain Φ. This ensures that every point in the input space has equal weight, so that any variation in the output over the input space is due only to the models $h_j(\cdot)$ [31].

The smooth combination provided by the LMN formalism enables the representation of complex nonlinear mappings on the basis of simpler modules. See the example in Fig. 6 which shows the combination in a two dimensional input space of



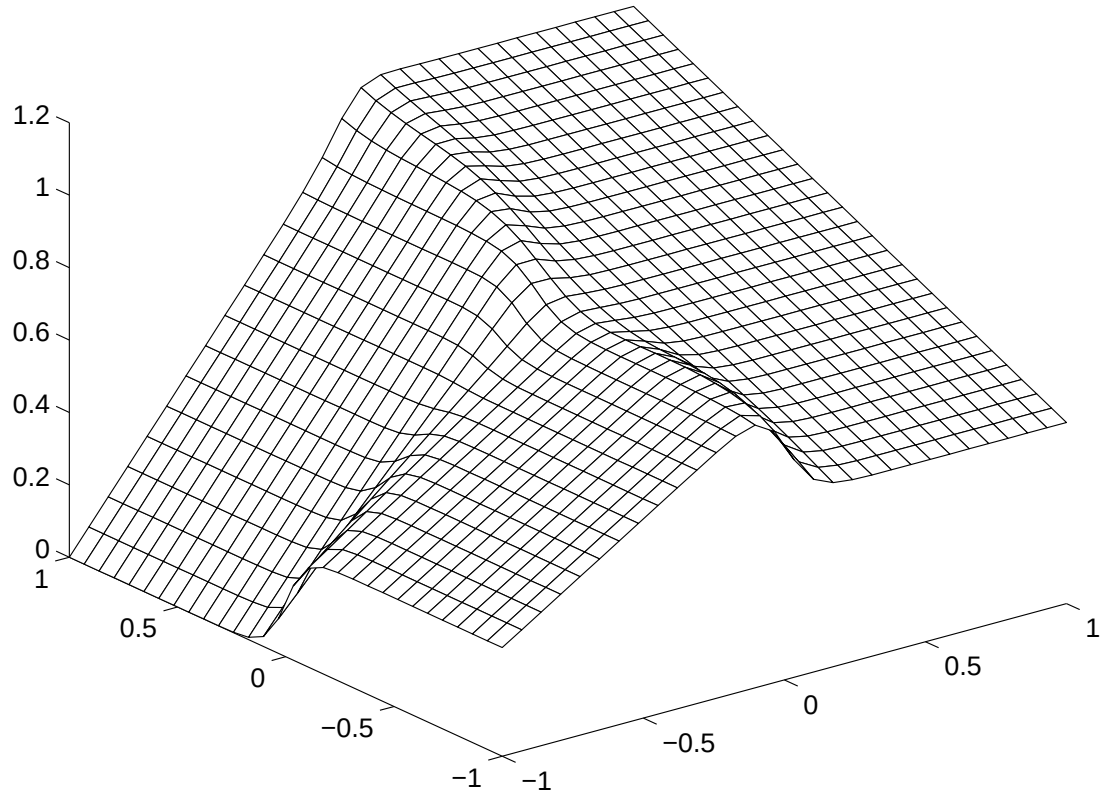


Fig. 6. A Local Model Network with $m = 3$ local models: the nonlinear input/output approximator in (c) is obtained by combining the three local linear models in (a) according to the three basis functions in (b).

three local linear models whose validity regions is represented by Gaussian basis functions.

In general, the local models $h_j(\cdot, \alpha)$ in Eq. (40) can be of any form: linear, nonlinear, physical models or black-box parametric models. In the case of local linear models

$$h_j(\phi, \alpha_j) = \sum_{i=1}^{n} a_{ji}\phi_i + b_j, \tag{42}$$

where the vector of parameters of the local model is $\alpha_j = [a_{j1}, \ldots, a_{jn}, b_j]$ and ϕ_i is the i^{th} term of the vector ϕ. Note that this kind of LMN architecture returns one further information about the input/output phenomenon: the local linear approximation h_{lin} of the input/output mapping about a generic point ϕ

$$h_{lin}(\phi) = \sum_{j=1}^{m} \rho_j(\phi, \eta_j) \left(\sum_{i=1}^{n} a_{ji}\phi_i + b_j \right). \tag{43}$$

The learning of a LMN from a training dataset requires two steps: structural identification and parametric identification. The structural identification step aims to find the optimal number and shape of the basis functions $\rho_j(\cdot)$. Once the structure of the network is defined, the parametric identification searches for the optimal set of parameters η_j of the basis functions (e.g. center and width in the Gaussian case) and the optimal set of parameters α_j of the local models (e.g. linear coefficients in the case of local linear models). For more information on LMN learning, we refer the reader to [32].

The following sections will discuss how a local modular representation of the system dynamics can be integrated in a divide-and-conquer control architecture. In particular, Section VI-B, VI-C and VI-D will present three applications of modular techniques to control design. As we will see, these architectures differ for the domain on which the partitioning is made and for the composition method of the local controllers.

B. The Local Controller Technique

The local controller approach is characterized by three steps (Fig. 7):

1) the plant is modeled by a modular architecture, e.g. a Local Model Network (40) composed of m local linear models,
2) a feedback controller is designed for each local model in order to guarantee local properties of stability and robustness,
3) a global controller is obtained by composing the m local controllers according to the weighting functions in (40).

At a first sight the local controller approach appears identical to the gain scheduling technique, described in Section IV. In fact, two are the main differences:

1) in the local controller approach the local linear models are not linearizations about equilibria points but about generic operating points
2) unlike gain scheduling, the local controller technique requires no analytical description of the plant.

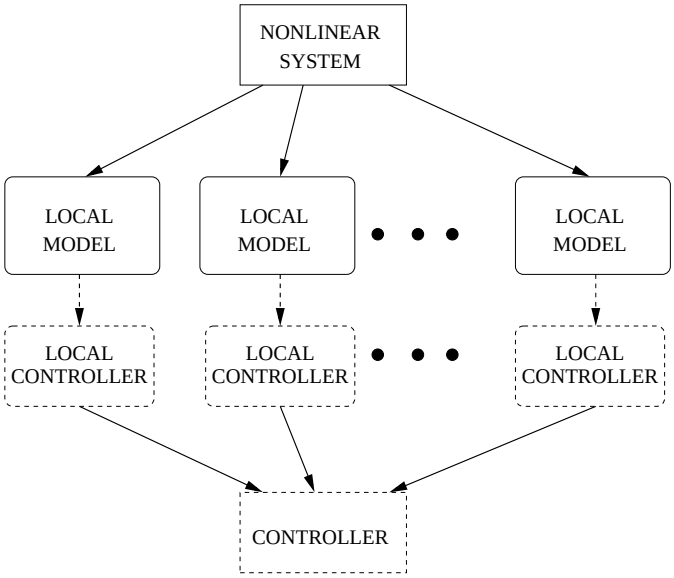


Fig. 7. The local controller approach. From top to bottom: the plant is modeled by a modular architecture made of m local models, a controller is designed for each local model, the outputs of the controllers are weighted to form the global controller.

The local controller approach requires only the existence of a modular description (40) of the nonlinear autonomous system (3) to be controlled. The modular description can be given by the designer or identified from a set of observed data.

We consider a modular architecture made of a set of m local linear models, where the j^{th} one describes the model dynamics within the region specified by the j^{th} validity function. The dynamics of the overall system is then represented by the following expression:

$$\dot{x} = \sum_{j=1}^{m} \rho_j(x)(A_j x + B_j u), \tag{44}$$

where

- $\rho_j : \mathbb{R}^n \to [0, 1]$, $j = 1 \ldots m$, is the validity function of the j^{th} local region with
$$\sum_{j=1}^{m} \rho_j(x) = 1; \tag{45}$$
- $\dot{x} = A_j x + B_j u$ is a linear model describing the system dynamics in the j^{th} region.

Once the plant is represented in the form (44), the local controller technique returns a controller architecture which combines a set of m control rules

$$u = K_k x, \tag{46}$$

where K_k is the feedback gain of the linear control law which stabilizes the linear dynamics in the k^{th} region, $k = 1, \ldots, m$. The overall controller, obtained by aggregating the m controllers, is given by:

$$u = \sum_{k=1}^{m} \rho_k(x) K_k x, \tag{47}$$

where $\rho_k : \mathbb{R}^n \to [0,1]$, $k = 1 \dots m$, is the validity region of the k^{th} controller. Replacing (47) in (44), we obtain the expression for the closed loop system

$$\dot{x} = \sum_{j=1}^{m} \sum_{k=1}^{m} \rho_j(x)\rho_k(x)(A_j + B_j K_k)x \tag{48}$$

and further, denoting $(A_j + B_j K_k)$ by A_{jk}, we can write

$$\dot{x} = \sum_{j=1}^{m} \sum_{k=1}^{m} \rho_j(x)\rho_k(x)A_{jk}x. \tag{49}$$

This controller architecture was presented as the FLC1 architecture in [35].

1) Stability of the Local Controller: The dynamics of the closed loop system is nonlinear due to the nonlinearities introduced by $\rho_j(x)$ and $\rho_k(x)$. However, results from [51] show that the global stability is independent of the weights $\rho_j(x)$ and $\rho_k(x)$, as stated by the following theorem:

Theorem 4: The equilibrium of a control system (49) is globally asymptotically stable if there exists a common positive definite matrix P such that

$$(A_{jk})^T P + P A_{jk} < 0. \tag{50}$$

This theorem establishes a sufficient condition. Therefore the system (49) may be globally asymptotically stable though it is not possible to find a common positive definite matrix which satisfies (50).

An important issue is whether the system (49) may be stable when all the A_{jk} are stable matrices. Unfortunately, the answer is no in general. An example is reported by Tanaka and Sugeno [51] who present a control system (49) which is not globally asymptotically stable though all the A_{jk} matrices are stable.

For that reason a complex procedure is generally required to design stable local controller architectures.

2) Design of a Local Controller: The design of a stable local controller architecture is strictly related to the problem of finding a set of gains K_k such that a common matrix P satisfying the theorem 4 exists. Unfortunately, this is a conservative property and no systematic procedure is provided in the existing literature. Recently, a number of Lyapunov methods for the definition of the matrix P, aimed specifically at local linear controller architectures, have received a lot of attention. They are based on the use of well-behaved numerical convex optimization methods to search for Lyapunov stabilizing functions. These methods focus on Lyapunov functions having arbitrary structure, as long as the conditions on positive and negative definiteness are true. In particular, in fuzzy control it is common to consider the following types of Lyapunov function:

- quadratic Lyapunov functions,
- piecewise quadratic Lyapunov functions ,
- piecewise affine Lyapunov functions.

An interesting overview of Lyapunov methods for piecewise models is presented in [22].

C. The Local Gain Scheduler

The local version of the gain scheduling approach was first proposed in fuzzy control literature by Palm and Rehfuess [36]. This local architecture has been also presented as the FLC2 architecture in [35].

As in the local controller approach, this method linearizes the original system in m operating points

$$\begin{aligned} \dot{x} &= A(x_j, u_j)(x - x_j) + B(x_j, u_j)(u - u_j) \\ &= A_j(x - x_j) + B_j(u - u_j), \\ & \quad j = 1, \dots, m \end{aligned} \tag{51}$$

and derives the control laws

$$u = K(x_k, u_k)(x - x_k) + u_k = K_k(x - x_k) + u_k \quad k = 1, \dots, m \tag{52}$$

that stabilize each local model.

If we assume that the equilibrium point (x_{eq}, u_{eq}) is in the neighborhood of the operating points, the system dynamics can be approximated by

$$\dot{x} = \sum_{j=1}^{m} \rho_j(x_{eq})\big(A_j(x - x_{eq}) + B_j(u - u_{eq})\big). \tag{53}$$

Note that, once fixed a certain x_{eq}, all the ρ_j return a constant value and consequently the model (53) is linear.

The resulting control action is then

$$u = \sum_{k=1}^{m} \rho_k(x_{eq})\big(K_k(x - x_{eq}) + u_{eq}\big). \tag{54}$$

Hence, by replacing (54) in (53) we obtain the equation for the closed-loop configuration

$$\dot{x} = \sum_{j=1}^{m} \sum_{k=1}^{m} \rho_j(x_{eq})\rho_k(x_{eq})(A_j + B_j K_k)(x - x_{eq}). \tag{55}$$

By comparing the above equation with Eq. (48) a substantial difference between the local gain-scheduler approach and the local controller approach emerges. In local gain-scheduling the validity functions ρ_k, $k = 1, \dots, m$, measure the "appropriateness" of the k^{th} control law for controlling the system about a given equilibrium x_{eq}; in the local controller approach they measure the "appropriateness" of the k^{th} control law for a given operating condition.

1) Stability and Design of the Local Gain Scheduler Controller: In order to derive the conditions under which (55) is asymptotically stable, some results about robust stability have to be introduced. Consider a linear system with linear perturbations

$$\dot{x} = Ax + \sum_{j=1}^{m} k_j \delta A_j x, \tag{56}$$

where A is a Hurwitz matrix, δA_j are constant matrices of the same dimensions as A, and k_j are uncertain parameters with values in an arbitrary interval around zero. Let P be the unique solution of the Lyapunov equation

$$A^T P + P A = -2I, \tag{57}$$

and let us define the matrices P_j as

$$P_j = \frac{\delta A_j^T P + P \delta A_j}{2}. \quad (58)$$

Then the following theorem holds

Theorem 5: [53] The linear system (56) is asymptotically stable if

$$\sum_{j=1}^{m} \|k_j\| \sigma_{max}(P_j) < 1, \quad (59)$$

where σ_{max} is the largest singular value.[1]

This theorem can be used for a stable design of the local gain scheduler [35]. First of all, we denote by A the common closed loop matrix which can be obtained with a proper feedback of each local A_j. That is

$$A_1 + B_1 K_1 = A_2 + B_2 K_2 = \cdots = A_m + B_m K_m = A, \quad (60)$$

where the matrices K_j are designed by a pole assignment technique [2] which assigns the same set of desired poles to the m local closed loop systems. By putting

$$A + \delta A_{jk} = A_j + B_j K_k = A_{jk} \quad (61)$$

we can easily obtain the matrices P_{jk}. Since $\sum_{j=1}^{m} \sum_{k=1}^{m} \rho_j(x_{eq}) = 1$, the system (55) can be written in the form

$$\dot{x} = A(x - x_{eq}) + \sum_{j=1}^{m} \sum_{k=1}^{m} \rho_j(x_{eq}) \rho_k(x_{eq}) \delta A_{jk} (x - x_{eq}). \quad (62)$$

According to Theorem 5 the linear system (62) is asymptotically stable if

$$\sum_{i=1}^{m} \sum_{k=1}^{m} \rho_j(x_{eq}) \rho_k(x_{eq}) \sigma_{max}(P_{jk}) < 1. \quad (63)$$

The above condition holds if the stronger condition

$$\sum_{i=1}^{m} \sum_{k=1}^{m} \sigma_{max}(P_{jk}) < 1 \quad (64)$$

is satisfied. Hence, in order to verify whether the gain matrices K_j satisfy condition (64) we have to compute the set of singular values for each P_{jk}. If condition (64) holds we have a stable design, otherwise the gain K_j have to be redesigned over and over until the condition is met.

D. The Local Self-tuning Controller

In the two previous approaches the overall controller is obtained by synthesizing a local controller for each rule and then combining their outputs. In this section we discuss an indirect control approach where the combination is made at the modeling level and not at the controller level (Fig. 8). We suppose that the plant is described by a parametric model:

$$\dot{x} = f(x(t), u(t), \vartheta, t). \quad (65)$$

An *indirect control* scheme [2], [34] combines a parameter estimator, which computes an estimate $\hat{\vartheta}$ of the parameter ϑ in (65) with a control law $u = K(x, \vartheta)$ implemented as a function of ϑ. In the adaptive version, the estimator generates the estimate $\hat{\vartheta}(t)$ at each time instant by processing the observed input/output behavior. This estimate is then assumed to be a specification of the real plant and used to compute the control law $u = K(x, \hat{\vartheta}(t))$ (certainty equivalence assumption) (Fig 9). A controller of this type is also called a *self-tuning regulator* (STR) to emphasize that the controller automatically tunes its parameters to obtain the desired properties of the closed-loop system. In conventional adaptive control theory, to make the problem analytically tractable, the plant is assumed to be a linear time-invariant system with unknown parameters.

Local self-tuning control addresses nonlinear configurations by combining a divide-and-conquer model with conventional

[1]Refer to [40] for the computation of singular values.

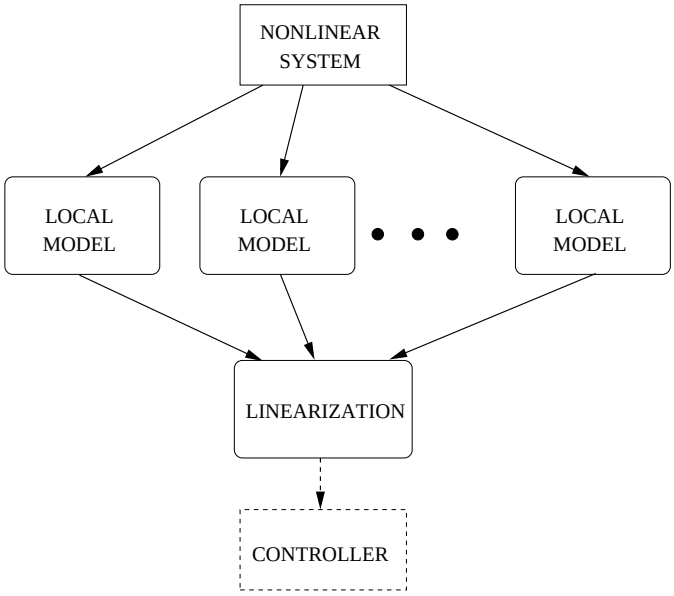


Fig. 8. The local self-tuning approach. From top to bottom: the plant is modeled by a modular architecture made of m local models, a local linearization is obtained by composing the m local models, a feedback controller is designed to stabilize the local linearization.

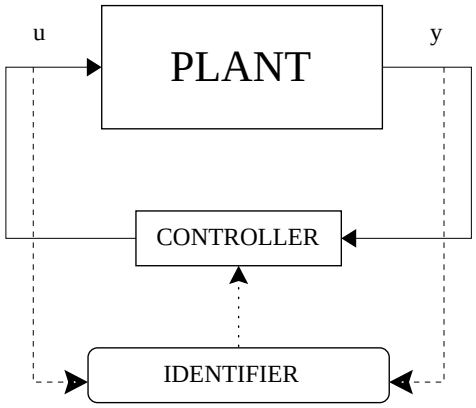


Fig. 9. The self-tuning regulator architecture. The parameters of the controller are updated by an identification module.

linear control techniques. The main difference with respect to conventional adaptive control techniques lies in the parameter estimation scheme. In the linear self-tuning regulators, identification is performed by a recursive parameter estimator which updates the same linear model each time an input/output sample is observed. In local self-tuning control there is not a global linear description but at each time-step the system dynamics is linearized by a modular architecture in the neighborhood of the current state. It is important to remark that in this case the linearization is performed also in configurations which are far from the equilibrium locus.

The modular architecture is made of a set of local models

$$\dot{x} = A_j x + B_j u + b_j \quad j = 1, \ldots, m, \tag{66}$$

where b_j represents the offset term in the validity region ρ_j. The dynamics of the overall system is then modeled by

$$\dot{x} = \sum_{j=1}^{m} \rho_j(x)(A_j x + B_j u + b_j). \tag{67}$$

Unlike previous approaches, the local self-tuning controller is not a combination of local feedback gains. Instead, the model (67) is used to return at each instant a local linearization of the global dynamics. For instance, suppose that at time $\bar{t}$ the state is $\bar{x}$. In a neighborhood of $\bar{x}$ the system dynamics can be represented by the linear model

$$\dot{x} = \sum_{j=1}^{m} \rho_j(\bar{x})(A_j x + B_j u + b_j). \tag{68}$$

Once this description is available, a linear control technique, as minimum variance (MV) or pole placement (PP) [2] can be used to stabilize the system (68). However, these design techniques require a model formulation in the form

$$\dot{x} = Ax + Bu, \tag{69}$$

while the local STR approach performs linearizations (68) also in configurations which are far from the equilibrium locus. This means that the local STR architecture returns in the neighborhood of a state $\bar{x}$ a local description in the form

$$\dot{x} = Ax + Bu + b, \tag{70}$$

where b is an offset term. This requires a slight modification to the equations of MV and PP controllers. The formulas for MV and PP controllers which take into account the offset term b in the case of a discrete-time system, are derived in Appendix A and B, respectively.

1) Stability of the Local Self-tuning Control: In the local STR approach a nonlinear plant is parameterized as a linear system whose parameters change with the current state $\bar{x}$:

$$\begin{aligned} \dot{x} &= \sum_{j=1}^{m} \rho_j(\bar{x}) A_j x + \sum_{j=1}^{m} \rho_j(\bar{x}) B_j u + \sum_{j=1}^{m} \rho_j(\bar{x}) b_j \\ &= A(\bar{x})x + B(\bar{x})u + b(\bar{x}) \end{aligned} \tag{71}$$

This configuration reminds both the linear parameter varying (LPV) configuration introduced by Shamma and Athans [47] in their analytical study of gain scheduling controllers, and the state-dependent models presented by [41].

In our stability analysis of the local STR approach, we assume that there exists a LPV model which represents in a sufficiently accurate manner the nonlinear system. Another important assumption is the *stabilizability* of the system [18], i.e. we assume that for each linearized model (71) the hypothesis of complete controllability holds. This is an assumption usually required in adaptive control theory for deriving proof of stability for indirect control schemes.

Hence, the closed-loop system may be put into the state-space form

$$\dot{x} = A_c(\bar{x})x, \tag{72}$$

which makes easier the analysis of stability. If the regulator is designed such that the eigenvalues of A_c are stable, then the system (72) will be asymptotically stable for any fixed value of $\bar{x}$ (frozen time stability). However, this is not a sufficient condition for uniform asymptotic stability of the system. A sufficient condition for uniform stability, once we know the set of values assumed by the matrix $A_c(\bar{x})$, is given in Theorem 3.

In particular, if $A_c(\bar{x}) = A_c$ for all the future values of $\bar{x}$, i.e. the closed loop transfer function is kept fixed and stable, it follows that the controlled system is globally asymptotically stable.

VII. Lazy Learning for Divide-and-conquer Control

The modular techniques presented in the previous section identify models which are made of several simple modules covering different parts of the state space. Notwithstanding their modularity, these techniques produce a functional description of the system for all the possible operating regimes. Local modeling techniques, on the contrary, do not aim to return a complete description of the system but rather to estimate a reliable prediction and local linearization of the system once a specific operating point is given. Lazy Learning (LL) is a local modeling algorithm which has been proposed and applied by the authors to several practical problems in control, data analysis and time series prediction [9].

We will first introduce the Lazy Learning algorithm in Section VII-A. The integration of the algorithm within a self-tuning regulator control strategy will be presented in Section VII-B.

A. The Lazy Learning Method

Given two variables $\phi \in \mathbb{R}^n$ and $y \in \mathbb{R}$, let us consider the mapping $f\colon \mathbb{R}^n \to \mathbb{R}$, known only through a set of N examples $\{(\phi_i, y_i)\}_{i=1}^{N}$ obtained as follows:

$$y_i = f(\phi_i) + \varepsilon_i, \tag{73}$$

where $\forall i$, ε_i is a random variable such that $E[\varepsilon_i] = 0$ and $E[\varepsilon_i \varepsilon_j] = 0$, $\forall j \neq i$, and such that $E[\varepsilon_i^r] = \mu_r(\phi_i)$, $\forall r \geq 2$, where $\mu_r(\cdot)$ is the unknown r^{th} moment of the distribution of ε_i and is defined as a function of ϕ_i.

Given a query point ϕ_q, the parameter $\boldsymbol{\beta}_1$ of a local first-degree polynomial approximating $f(\cdot)$ in a neighborhood of ϕ_q, can be obtained solving the local polynomial regression:

$$\sum_{i=1}^{N} \left\{ (y_i - \phi_i' \boldsymbol{\beta})^2 K\left(\frac{D(\phi_i, \phi_q)}{h} \right) \right\}, \tag{74}$$

where, given a metric on the space $\mathbb{R}^n$, $D(\phi_i, \phi_q)$ is the distance from the query point to the i^{th} example, $K(\cdot)$ is a weight function, h is the bandwidth, and where the vectors ϕ_i have been obtained by pre-appending a constant value 1 to each vector ϕ_i in order to consider a constant term in the regression.

Once obtained the local first-degree polynomial approximation $\hat{\beta}$, a prediction of $y_q = f(\phi_q)$, is finally given by

$$\hat{y}_q = \phi_q' \hat{\beta}. \tag{75}$$

By exploiting the linearity in the parameters of the local approximator, a leave-one-out cross-validation estimation of the error variance $E[(y_q - \hat{y}_q)^2]$ can be obtained without any significant overload. In fact, using the PRESS statistic [33], it is possible to calculate the leave-one-out error $e_j^{cv} = y_j - \phi_j' \hat{\boldsymbol{\beta}}^{-j}$, without explicitly identifying the regression parameters $\hat{\boldsymbol{\beta}}^{-j}$ with the j^{th} case set aside.

If a rectangular weight function $K(\cdot)$ is adopted, the optimization of the parameter h can be conveniently reduced to the optimization of the number k of neighbors to which a unitary weight is assigned in the local regression evaluation. In other words, we reduce the problem of bandwidth selection to a search in the space of $h(k) = D(\phi(k), \phi_q)$, where $\phi(k)$ is the k^{th} nearest neighbor of the query point.

The main advantage deriving from the adoption of an indicator weight function, is that, simply by updating the parameter $\hat{\boldsymbol{\beta}}(k)$ of the model identified using the k nearest neighbors, it is straightforward and inexpensive to obtain $\hat{\boldsymbol{\beta}}(k+1)$. The recursive algorithm described in [6], [11] returns for a given query point ϕ_q, a set of predictions $\hat{y}_q(k) = \phi_q' \hat{\boldsymbol{\beta}}(k)$, together with a set of associated leave-one-out error vectors $e^{cv}(k)$.

On the basis of this information a local linearization (and consequently the prediction $\hat{y}_q$ of the value of the regression function) in ϕ_q can be obtained in two different ways: the first is based on the selection of the *best* approximator according to a given criterion, while the second returns a prediction as a *combination* of more local models.

If the selection paradigm, frequently called *winner-takes-all*, is adopted, the most natural way to extract the best local linearization $\hat{\beta}(\hat{k})$ consists in comparing, on the basis of the classical *mean square error* criterion, the prediction obtained for each value of k, given the degree d of the local approximator.

$$\hat{y}_q = \phi_q' \hat{\boldsymbol{\beta}}(\hat{k}), \quad \text{with } \hat{k} = \arg\min_{k \in \mathcal{K}} mse^{cv}(k); \tag{76}$$

where $\mathcal{K}$ is a range, defined by the analyst, from which the optimal number of neighbors is selected.

A variant of the method consists in adopting the local combinations of estimates [52], [37] as an alternative to the *winner-takes-all* paradigm. In this case the resulting linearization is obtained as a weighted average of the best b models, where b is a parameter of the algorithm. Suppose the predictions $\hat{y}_q(k)$ and the error vectors $e^{cv}(k)$ have been ordered creating a sequence of integers $\{k_i\}$ so that $mse^{cv}(k_i) \leq mse^{cv}(k_j)$, $\forall i < j$. The prediction of y_q is given by

$$\hat{y}_{d,q} = \frac{\sum_{i=1}^{b} \zeta_i \, \hat{y}_q(k_i)}{\sum_{i=1}^{b} \zeta_i}, \tag{77}$$

where the weights are the inverse of the mean square errors: $\zeta_i = 1/mse^{cv}(k_i)$. The corresponding combination rule applied to the linear parameters $\hat{\beta}$ returns the linearization of $f(\cdot)$ in ϕ_q.

The LL algorithm is made publicly available by the authors as a MATLAB toolbox [5].[2] For more information on theoretical aspects and practical applications of the method, we refer the reader to [10], [11], [7].

B. The Lazy Learning Self-tuning Control

This section will discuss a divide-and-conquer method, based on the Lazy Learning technique, for the control design of nonlinear discrete-time systems in the input/output form.

The reasons for focusing on discrete-time configurations are essentially two: the growing importance of digital computers in practical applications and the fact that a discrete-time configuration makes easier the extension of the Lazy Learning approach to control design.

For the sake of simplicity, we will restrict to consider single-input single-output (SISO) discrete-time dynamic systems whose equations of motion are expressed in the form:

$$\begin{aligned} y(t) = f\big(&y(t-1), \ldots, y(t-n_y), u(t-d), \ldots, u(t-d-n_u), \\ &w(t-1), \ldots, w(t-n_e)\big) + w(t), \end{aligned} \tag{78}$$

where $t \in \mathbb{N}$ denotes the discrete time, n_y, n_u and n_e are positive numbers, $y(t)$ is the system output, $u(t)$ the input, $w(t)$ is a zero-mean disturbance term, $d > 0$ is the input/output time delay and $f(\cdot)$ is some nonlinear function. This model is known as the NARMAX (Nonlinear AutoRegressive Moving Average with eXternal input) model [25].[3] Let us assume we have no a priori information about the function $f(\cdot)$ except a training set D_N made of N pairs $\langle u(t), y(t) \rangle$. Defining the *information vector* as

$$\begin{aligned} \phi(t-1) = \big[&y(t-1), \ldots, y(t-n_y), u(t-d), \ldots, u(t-d-n_u), \\ &w(t-1), \ldots, w(t-n_e)\big], \end{aligned} \tag{80}$$

the system (78) can be written in the form:

$$y(t) = f\left(\phi(t-1)\right) + w(t). \tag{81}$$

[2] HTTP://IRIDIA.ULB.AC.BE/∼LAZY

[3] It is demonstrated [25] that models (78) can describe the input/output behavior of general state-space nonlinear dynamic systems:

$$\begin{aligned} x(t+1) &= g\left(x(t), u(t)\right) + w_x(t) \\ y(t) &= h\left(x(t)\right) + w_y(t), \end{aligned} \tag{79}$$

where $x \in \mathbb{R}^n$ is the state vector, $w_x \in \mathbb{R}^n$ and $w_y \in \mathbb{R}$ are zero-mean disturbances, and $g : \mathbb{R}^{n+1} \to \mathbb{R}^n$, $h : \mathbb{R}^n \to \mathbb{R}$ are some nonlinear functions.

1) Acquisition of the vector (83).
2) Linearization of the function $f(\cdot)$ about (83). The linearization is computed by the Lazy Learning algorithm.
3) Derivation of the polynomials A, B, C and the offset b (see Appendix) from the linearized model.
4) Design of a MVG/PP controller which satisfies the required properties (stability, accuracy, speed...) of the closed loop behavior.
5) Computation of the control signal.
6) Updating of the database by storing the new input/output observation.

Repeat these steps at each sampling period.

Fig. 10. The Lazy Learning self-tuning controller algorithm.

We consider a locally linear indirect control scheme where the Lazy Learning technique is used to return the set of linear parameters which describe locally the nonlinear plant (81).

In linear control theory the plant is typically modeled in the input/output parametric form

$$A(z)y(t) = z^{-d}B(z)u(t) + C(z)w(t), \tag{82}$$

where the matrices A, B and C are the set of parameters. The indirect control scheme consists in combining a parameter estimator, which computes an estimate $\hat{\vartheta}$ of the unknown plant parameters, with a control law $u(t) = K(\phi(t), \vartheta)$ implemented as a function of the real plant parameters. Once the parameters in (82) have been estimated, conventional linear control employs methods like the minimum-variance (MV) or the pole-placement (PP) control techniques [2].

The idea of the LL self-tuning (LL-ST) controller consists in first linearizing the system dynamics in the neighborhood of the current operating point, then in computing the linear control law. The current operating point is represented by the information vector (80). Note that the selection of neighbors in the Lazy Learning identification procedure (Section VII-A) is made considering only the subset vector

$$\phi_s(t-1) = \big[y(t-1), \ldots, y(t-n_y), u(t-2), \ldots, u(t-n_u), w(t-1), \ldots, w(t-n_e)\big], \tag{83}$$

of the information vector (80). The reason is that $u(t-1)$ cannot be available as it is the expected outcome of the procedure. Anyway, the local weighted regression is performed in the space of the complete information vector (80).

Once the linear parameters in Eq. (82) are returned by the Lazy Learning identification module, a linear control algorithm is used to implement the controller. The resulting control algorithm is described in detail in Fig. 10. An experimental validation of the LL-ST control strategy can be found in [10], [7], [8]. Other applications of LL to practical control problems can be found in [4].

C. Lazy Learning Self-tuning Control Analysis

In the Lazy Learning self-tuning regulator, the nonlinear plant (81) is parameterized as a linear system where the parameters are changing with the observable state. This means that the nonlinear model can be written as a linear parameter varying (LPV) model (71) where the parameters vary with the state of the system.

In a discrete-time formulation the models can be represented as follows:

$$A(\phi(t))y(t) = z^{-d}B(\phi(t))u(t) + C(\phi(t))w(t). \tag{84}$$

Let us now assume that there exists a LPV model which represents in a sufficiently accurate manner the nonlinear system (81). If we make the hypothesis of complete controllability and observability, the closed-loop system may be put into the state-space form

$$x(t+1) = F\big(\phi(t)\big)x(t) + v(t). \tag{85}$$

This representation allows us to analyze the stability of the LL-ST controller. If the regulator is designed such that the eigenvalues of $F(\phi(t))$ are stable, then the system (85) will be asymptotically stable for any fixed value of ϕ (frozen time stability). However, this is not a sufficient condition for uniform asymptotic stability of the system. If we consider the set of values assumed by the matrix $F(\phi(t))$ a sufficient condition for uniform stability [51] is that it exists a common matrix $P > 0$ such that:

$$F(\phi(t))^T P F(\phi(t)) - P < 0 \qquad \text{for all t.} \tag{86}$$

With the pole placement technique we can impose the same stable closed loop transfer function for all t. It follows that there exists a matrix P that satisfies the equation (86). Then the following theorem holds

Theorem 6: If

1) the system (81) is completely controllable and observable;
2) the system (81) can be put in the form (84);
3) the approximation error of the Lazy Learning identifier is negligible;

then the equilibrium of the nonlinear system (81) controlled by the LL PP self-tuning controller is globally asymptotically stable.

D. Considerations on Lazy Learning Controllers

Lazy Learning is an example of supervised techniques which allows the reuse of a number of results and techniques of linear control in a nonlinear setting. Since, as showed in this paper, this is not a prerogative of the LL approach it is interesting to present a comparative analysis of the Lazy Learning approach with the other data driven methods outlined in the paper.

Lazy Learning vs. linear adaptive control: Conventional adaptive control is based on a recursive identification procedure which updates on-line a linear approximation to the plant on the basis of upcoming observations. In order to cope both with nonlinear and time-varying configurations a forgetting factor, which weights more the latest observations, is introduced in the recursive computation [49].

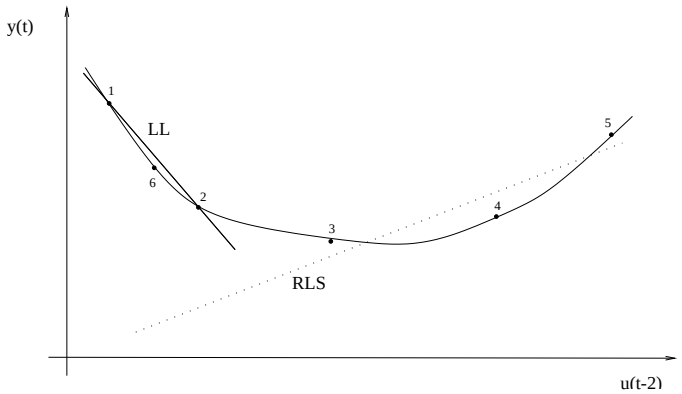


Fig. 11. Recursive Least-Squares with forgetting factor (dotted) vs. Lazy Learning (solid) identification of the model $y(t) = f(u(t-2))$ in the neighborhood of the configuration 6. The figure compares the linearizations returned by the two models when the input/output samples 1 to 5 have been collected (the sequence of numbers represents the temporal sequence of the states in the state space). While the RLS is adversely affected by the interference between the previous states and the actual one, the Lazy Learning algorithm makes a better use of the past observations.

Unlike conventional adaptive control, the Lazy Learning approach treats separately nonlinear and time-varying configurations.

In nonlinear tasks, LL fits a local linear model by using the samples which are the most representative ones of the current operating point. This approach returns different local models for different operating regimes and avoids the problem of *data interference* [19] [44] — a.k.a. *stability-plasticity dilemma* [14]. Consider, as illustrative example, the simple nonlinear dynamics $y(t) = f(u(t-2))$ of which we plot six input/output observations in Fig. 11. Let the numbers represent the temporal order with which we observed the samples. If the system is identified with a forgetting factor recursive approach, when the entry no. 6 is encountered, the linear model (dotted line) has lost memory of the dynamics existing in the neighborhood of the points 1 and 2. As a result, the accuracy of the RLS approximation (dotted line) in 6 is poor due to its limited tracking speed. On the other hand, the Lazy Learning approach is not affected by any interference phenomenon (from data 4 and 5) and returns a better local approximation (solid line).

As far as nonstationarity is concerned, the LL technique deals with time-varying configurations with minor changes. It is sufficient to extend the input domain by adding the current time variable t to the set of input features. Once a linearization is required, the nearest samples in space and time are used to fit the local model.

Lazy Learning vs. modular control: The two approaches share the common idea of decomposing a difficult problem into simpler local problems. The main differences concern the model identification procedure. Modular methods (Section VI) return a piecewise description which covers the whole system operating domain, whereas local modeling techniques focus exclusively on the current operating point. Modular techniques are more time consuming in identification but faster in prediction. However when new data are observed, the updating of a modular architecture requires the repetition of the whole modeling process from scratch. On this matter Lazy Learning takes an advantage from the absence of a global model: once a new input/output example is observed, it is sufficient to update the database which stores the set of input/output pairs.

VIII. Concluding Remarks

This paper surveyed a set of control design techniques which share the same idea of decomposing a nonlinear complex problem into a set of simpler linear problems. However, notwithstanding the common underlying principle, divide-and-conquer control techniques can still differentiate for a number of reasons:

Analytical assumptions: Some methods, like linearization, gain scheduling and feedback linearization, require an analytical description of the system, while others, like modular control and lazy learning, are based on parametric models identified from data.

Locus of linearization: On this matter, the main distinction is between methods that linearize in the neighborhood of equilibrium points (e.g. gain scheduling) and methods that linearize off equilibria (e.g. local controller).

Range of validity of local representations: In linearization there is an intuitive notion of proximity to the equilibrium point, in gain scheduling the notion is extended to consider a set of equilibrium points, in modular architecture previous knowledge or identification from data determine the local regions of validity.

How the local controllers are combined: We range from discontinuous switching in gain scheduling, composition of controllers in local controller architecture, composition of local models in local self-tuning control to the local linearization of Lazy Learning.

Adaptation of local models: While analytical methods do not take into consideration the availability of input/output observations, the performance of data driven techniques is strictly related to the capacity of updating the local descriptions once new data are collected.

Table 12 can be used as a concise summary of the techniques illustrated in the survey according to the above mentioned criteria.

Acknowledgments. The work of Mauro Birattari was supported by the *Metaheuristics Network*, a Research Training Network funded by the Improving Human Potential Programme of the Commission of the European Community, grant HPRN-CT-1999-00106. The information provided is the

Method	Analytical	Off-equilibrium	Combination	Adaptation
Gain scheduling	YES	NO	switching/smooth	NO
Feedback lin.	YES	NO	NO	NO
Local controller	NO	YES	at control level	difficult
Local self-tuning	NO	YES	at model level	difficult
Lazy Learning	NO	YES	local linearization	easy

Fig. 12. Taxonomy of divide-and-conquer methods for nonlinear control

sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

APPENDIX

Let $f\colon \mathbb{R}^n \to \mathbb{R}^r$ be a function which maps vectors in $\mathbb{R}^n$ to values in $\mathbb{R}^r$. In terms of individual components,

$$f(x) = \begin{bmatrix} f_1(x_1, \ldots, x_n) \\ \cdots \\ f_r(x_1, \ldots, x_n) \end{bmatrix} \tag{87}$$

where the x_i are scalar components of the $\mathbb{R}^n$ vector x, and the f_i are scalar valued functions of $\mathbb{R}^n$. The *Jacobian matrix* of f is denoted by G_f and is defined as the $[r \times n]$ matrix of partial derivatives

$$G_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_r}{\partial x_1} & \cdots & \frac{\partial f_r}{\partial x_n} \end{bmatrix} \tag{88}$$

In case f is continuously differentiable at x_0, then the Jacobian matrix can be used to approximate f. A multivariable Taylor series expansion takes the form

$$f(x) = f(x_0) + G_f(x_0)(x - x_0) + \rho(x) \tag{89}$$

where the remainder, $\rho(x)$, represents higher-order terms which satisfy

$$\lim_{h \to 0} \frac{|\rho(x_0 + h)|}{|h|} = 0 \tag{90}$$

Now let $f: \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^r$ be a function which maps a pair of vectors in $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively, to values in $\mathbb{R}^p$. The notations G_{1f} and G_{2f} denote the Jacobian matrices with respect to the first variable and second variables, respectively. Thus, if

$$f(x, u) = \begin{bmatrix} f_1(x_1, \ldots, x_n, u_1, \ldots u_m) \\ \cdots \\ f_r(x_1, \ldots, x_n, u_1, \ldots u_m) \end{bmatrix} \tag{91}$$

then G_{1f} denotes the $[r \times n]$ matrix

$$G_{1f} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_r}{\partial x_1} & \cdots & \frac{\partial f_r}{\partial x_n} \end{bmatrix} \tag{92}$$

and G_{f2} denotes the $[r \times n]$ matrix

$$G_{1f} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_r}{\partial u_1} & \cdots & \frac{\partial f_r}{\partial u_m} \end{bmatrix} \tag{93}$$

The minimum variance (MV) control algorithm was first formulated in [1]. Since then, the MV technique has had many practical applications and significant theoretical developments. Let us consider a linear discrete-time process (82) and suppose we want to regulate it to $y_{\text{ref}} = 0$. The MV control technique seeks the control law which minimizes the variance of the error. However, the MV controlled closed loop system is stable only if B has all of its roots inside the unit circle (minimum phase). Therefore, more complex formulations are needed in the case of a tracking problem or in the case of non minimum-phase systems. This is the case of Generalized minimum variance (GMV) which selects properly the controller parameters in order to make the closed loop system asymptotically stable.

Pole placement (PP) design is an alternative technique to deal with non minimum-phase configurations. The procedure requires first to choose the desired closed loop pole positions and then to calculate the appropriate controller.

Both these design techniques have been developed to deal with a linear model of the system in the form (82). When the linearization of the nonlinear plant is around configurations far from the equilibrium locus, the linear descriptions takes the form

$$A(z)y(t) = z^{-d}B(z)u(t) + C(z)w(t) + b \tag{94}$$

where the offset term b is nonzero. Hence, a proper modification of the GMV and PP control formulas is required. The derivation of the off-equilibrium control laws is given in the following section.

A. Generalized Minimum Variance Design with an Offset Term

Clarke and Gawthrop [15] developed the Generalized Minimum-Variance Controller (GMVC) by introducing the reference signal and the control variable into the performance index

$$J = E\left[\Big(P(z)y(t+d) + Q(z)u(t) - y_{\text{ref}}(t)\Big)^2\right], \tag{95}$$

where $P(z) = \frac{P_N(z)}{P_D(z)}$ and $Q(z) = \frac{Q_N(z)}{Q_D(z)}$ are the polynomial terms which weight the contribution of the output and the input, respectively. Suppose that data are generated according to model (94). Multiplying both sides of (94) by P we obtain

$$\frac{P_N(z)}{P_D(z)}y(t) = \frac{P_N(z)}{P_D(z)}\left(\frac{B(z)}{A(z)}u(t-d) + \frac{b}{A(z)} + \frac{C}{A(z)}w(t)\right). \tag{96}$$

By setting $\tilde{y} = Py$, $\tilde{y}_{\text{ref}} = y_{\text{ref}} - Qu$, $\tilde{A} = P_D A$, $\tilde{B} = P_N B$, $\tilde{C} = P_N C$ and $\tilde{b} = P_N b$:

$$\tilde{A}(z)\tilde{y}(t) = z^{-d}\tilde{B}(z)u(t) + \tilde{C}w(t) + \tilde{b}. \tag{97}$$

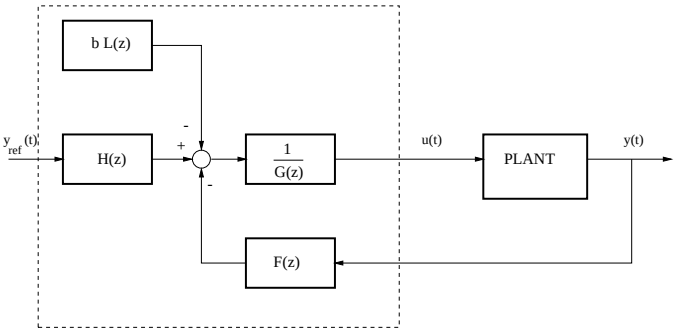


Fig. 13. The MV self-tuning architecture with an offset term. Note the presence of the additional term $bL(z)$ due to the offset $b \neq 0$.

Let the polynomial $W(z)$ and $\tilde{F}(z)$ be the solution of the equation:

$$\tilde{C}(z) = \tilde{A}(z)W(z) + z^{-d}\tilde{F}(z). \tag{98}$$

commonly known as the *Diophantine equation* [2]. Multiplying both sides of (97) by $z^d W(z)$ gives:

$$\tilde{A}(z)W(z)\tilde{y}(t+d) = \tilde{B}(z)W(z)u(t) + \tilde{C}w(t+d) + \tilde{b}W(z). \tag{99}$$

From (98) we have:

$$\tilde{C}(z)\tilde{y}(t+d) = \tilde{B}W(z)u(t) + \tilde{C}w(t+d) + \tilde{b}W(z) + \tilde{F}(z)\tilde{y}(t). \tag{100}$$

The control law that minimizes the cost (95) is then

$$\tilde{C}(z)\left(y_{\rm ref} - \frac{Q_N}{Q_D}u(t)\right) = \tilde{B}W(z)u(t) + \tilde{b}W(z) + \tilde{F}(z)\tilde{y}(t), \tag{101}$$

that is equivalent to

$$\begin{aligned} &P_D(z)Q_D(z)C(z)y_{\rm ref} - Q_N(z)P_D(z)C(z)u(t) = \\ &Q_D(z)\tilde{F}(z)y(t) + Q_D(z)P_D(z)B(z)W(z)u(t) \\ &+Q_D(z)P_D(z)bW(z) \end{aligned} \tag{102}$$

in the plant polynomials. The control law is then

$$u(t) = \frac{H(z)y_{\rm ref} - F(z)y(t) - bL(z)}{G(z)} \tag{103}$$

with $G = Q_N P_D C + Q_D P_D B W$, $L = Q_D P_D W$ and $H = P_D Q_D C$, $F = Q_D \tilde{F}$. The control system architecture is presented in Fig. 13. The result for the basic minimum variance controller can be obtained by setting $P_N = P_D = Q_D = 1$ and $Q_N = 0$.

B. Pole Placement Design with an Offset Term

In the pole-placement formulation the designer fixes the desired closed-loop function by:

$$H_m(z) = \frac{B_m(z)}{A_m(z)} \tag{104}$$

The PP regulator is characterized by one output u and two inputs: the reference signal $y_{\rm ref}$ and the measured output y. A general structure for the regulator is typically represented by

$$u(t) = \frac{T(q)}{R(q)}y_{\rm ref}(t) - \frac{S(q)}{R(q)}y(t) - G(q) \tag{105}$$

where R, T, G and S are polynomials in the forward-shift operator q. The input/output relationship for the closed-loop system is obtained by eliminating u between Equations (94) and (104). Hence:

$$y = \frac{B_d T}{AR + B_d S}y_{\rm ref} + \frac{R(b - GB_d)}{AR + B_d S} + \frac{CR}{AR + B_d S}w. \tag{106}$$

with $B_d = z^{-d}B$. Requiring that this input/output relation be equivalent to (104) gives

$$\frac{B_d T}{AR + BS} = \frac{B_m}{A_m}, \tag{107}$$

$$b = GB_d. \tag{108}$$

The pole-placement design problem with an offset term is then equivalent to the conventional one, once the additional requirement (108) is satisfied.

REFERENCES

[1] K. J. Åström. Computer control of a paper machine. an application of linear stochastic control theory. *IBM J. Res. Dev.*, 11:389–404, 1967.
[2] K. J. Åström and B. Wittenmark. *Computer-controlled Systems: Theory and Design*. Prentice-Hall International Editions, 1990.
[3] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1–5):11–73, 1997.
[4] E. Bertolissi, M. Birattari, G. Bontempi, A. Duchateau, and H. Bersini. Data-driven techniques for direct adaptive control: the lazy and the fuzzy approach. *Fuzzy Sets and Systems*, 128(1):3–14, 2002.
[5] M. Birattari and G. Bontempi. The lazy learning toolbox, for use with matlab. Technical Report TR/IRIDIA/99-7, IRIDIA-ULB, Brussels, Belgium, 1999.
[6] M. Birattari, G. Bontempi, and H. Bersini. Lazy learning meets the recursive least-squares algorithm. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 375–381, Cambridge, 1999. MIT Press.
[7] G. Bontempi. *Local Learning Techniques for Modeling, Prediction and Control*. PhD thesis, IRIDIA- Université Libre de Bruxelles, 1999.
[8] G. Bontempi, H. Bersini, and M. Birattari. The local paradigm for modeling and control: From neuro-fuzzy to lazy learning. *Fuzzy Sets and Systems*, 121(1):59–72, 2001.
[9] G. Bontempi, M. Birattari, and H. Bersini. Lazy learners at work: the lazy learning toolbox. In *Proceeding of the 7th European Congress on Inteligent Techniques and Soft Computing EUFIT '99*, 1999.
[10] G. Bontempi, M. Birattari, and H. Bersini. Lazy learning for modeling and control design. *International Journal of Control*, 72(7/8):643–658, 1999.
[11] G. Bontempi, M. Birattari, and H. Bersini. A model selection approach for local learning. *Artificial Intelligence Communications*, 121(1), 2000.
[12] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.
[13] R. Brockett. *Finite Dimensional Linear Systems*. Wiley, New York, 1970.
[14] G.A. Carpenter and S. Grossberg. The art of adaptive pattern recognition by a self-organising neural network. *IEEE Computer*, 21(3):77–88, 1988.
[15] D. W. Clarke and P. J. Gawthrop. Self tuning controller. *Proceedings IEEE*, 122(9):929–934, 1975.
[16] W. S. Cleveland and C. Loader. Smoothing by local regression: Principles and methods. *Computational Statistics*, 11, 1995.
[17] G. C. Goodwin and K. S. Sin. *Adaptive Filtering Prediction and Control*. Prentice-Hall, 1984.
[18] P. A. Ioannou and J. Sun. *Robust adaptive control*. Prentice-Hall, 1996.
[19] R. A. Jacobs, M. I. Jordan, and A. G. Barto. Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science*, 1991.
[20] T. A. Johansen and B. A. Foss. Constructing narmax models using armax models. *International Journal of Control*, 58:1125–1153, 1993.
[21] T. A. Johansen and B. A. Foss. Semi-empirical modeling of nonlinear dynamic systems through identification of operating regimes and local models. In K. J. Hunt, G. R. Irwin, and K. Warwick, editors, *Neural Network Engineering in dynamic control systems*, pages 105–126. Springer, 1995.

[22] M. Johansson. *Piecewise Linear Control Systems*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, 1999.
[23] M. J. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.
[24] Ching-Hung Lee. A survey of pid controller design based on gain and phase margins. *International Journal of Computational Cognition*, 2(3):63–100, 2004.
[25] I. J. Leontaritis and S. A. Billings. Input-output parametric models for non-linear systems. *International Journal of Control*, 41(2):303–344, 1985.
[26] L. Ljung. *System identification: Theory for the User*. Prentice-Hall, Englewood Cliffs, NJ, 1987.
[27] D. Luenberger. *Introduction to dynamic systems : theory, models, and applications*. Wiley, 1979.
[28] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.
[29] R. Marino and P. Tomei. *Nonlinear Control Design: Geometric, Adaptive and Robust*. Prentice Hall, UK, 1995.
[30] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.
[31] R. Murray-Smith. *A local model network approach to nonlinear modelling*. PhD thesis, Department of Computer Science, University of Strathclyde, Strathclyde, UK, 1994.
[32] R. Murray-Smith and T. A. Johansen, editors. *Multiple Model Approaches to Modeling and Control*. Taylor and Francis, 1997.
[33] R. H. Myers. *Classical and Modern Regression with Applications*. PWS-KENT Publishing Company, Boston, MA, second edition, 1994.
[34] K. S. Narendra and A. M. Annaswamy. *Stable Adaptive Systems*. Prentice Hall, Englewood Cliffs, NJ, 1989.
[35] R. Palm, H. Hellendoorn, and D. Driankov. *Model Based Fuzzy Control*. Springer, 1997.
[36] T. Palm and U. Rehfuess. Fuzzy controllers as gain scheduling approximators. *Fuzzy Sets and Systems*, 85:233–246, 1997.
[37] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Artificial Neural Networks for Speech and Vision*, pages 126–142. Chapman and Hall, 1993.
[38] R. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.
[39] M. J. D. Powell. *Algorithms for Approximation*, chapter Radial Basis Functions for multivariable interpolation: a review, pages 143–167. Clarendon Press, Oxford, 1987.
[40] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992. Second ed.
[41] M.B. Priestley. *Non-linear and Non-stationary time series analysis*. Academic Press, 1988.
[42] W. J. Rugh. Analytical framework for gain scheduling. *IEEE Control Systems Magazine*, 11(1):79–84, 1991.
[43] D. E. Rumelhart, G. E. Hinton, and R. K. Williams. Learning representations by backpropagating errors. *Nature*, 323(9):533–536, 1986.
[44] M. Salganicoff. Tolerating concept and sampling shift in lazy learning using prediction error context switching. *Artificial Intelligence Review*, 11(1–5):133–155, 1997.
[45] G. A. F. Seber and C. J. Wild. *Nonlinear regression*. Wiley, New York, 1989.
[46] J. S. Shamma. *Analysis and Design of Gain Scheduled Control Systems*. PhD thesis, Lab. for Information and Decision Sciences, MIT,, Cambridge, MA, 1988.
[47] J. S. Shamma and M. Athans. Gain scheduling: Potential hazards and possible remedies. *IEEE Control Systems Magazine*, pages 101–107, 1992.
[48] J. J. Slotine and W. Li. *Applied Nonlinear Control*. Prentice-Hall International, 1991.
[49] T. Söderström and P. Stoica. *System Identification*. Prentice Hall International, 1989.
[50] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1):116–132, 1985.
[51] K. Tanaka and M. Sugeno. Stability analysis and design of fuzzy control systems. *Fuzzy Sets and Systems*, 45:135–156, 1992.
[52] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.
[53] K. Zhou and P. Khargonekar. Stability robustness for linear state-space models with structures uncertainty. *IEEE Trans. on Automatic Control*, 32(7):621–623, 1987.