On Learning Machines for Engine Control

Gérard Bloch[1], Fabien Lauer[1], and Guillaume Colin[2]

[1] Centre de Recherche en Automatique de Nancy (CRAN), Nancy-University, CNRS, 2 rue Jean Lamour, 54519 Vandoeuvre lès Nancy, France, gerard.bloch@esstin.uhp-nancy.fr, fabien.lauer@esstin.uhp-nancy.fr
[2] Laboratoire de Mécanique et d'Energétique (LME), University of Orléans, 8 rue Léonard de Vinci, 45072 Orléans Cedex 2, France, guillaume.colin@univ-orleans.fr

Summary. The chapter deals with neural networks and learning machines for engine control applications, particularly in modeling for control. In the first section, basic features of engine control in a layered engine management architecture are reviewed. The use of neural networks for engine modeling, control and diagnosis is then briefly described. The need for descriptive models for model-based control and the link between physical models and black box models are emphasized by the grey box approach discussed in this chapter. The second section introduces the neural models frequently used in engine control, namely, MultiLayer Perceptrons (MLP) and Radial Basis Function (RBF) networks. A more recent approach, known as Support Vector Regression (SVR), to build models in kernel expansion form is also presented. The third section is devoted to examples of application of these models in the context of turbocharged Spark Ignition (SI) engines with Variable Camshaft Timing (VCT). This specific context is representative of modern engine control problems. In the first example, the airpath control is studied, where open loop neural estimators are combined with a dynamical polytopic observer. The second example considers modeling the in-cylinder residual gas fraction by Linear Programming SVR (LP-SVR) based on a limited amount of experimental data and a simulator built from prior knowledge. Each example demonstrates that models based on first principles and neural models must be joined together in a grey box approach to obtain effective and acceptable results.

1 Introduction

The following gives a short introduction on learning machines in engine control. For a more detailed introduction on engine control in general, the reader is referred to [20]. After a description of the common features in engine control (Sect. 1.1), including the different levels of a general control strategy, an overview of the use of neural networks in this context is given in Sect. 1.2. Section 1 ends with the presentation of the grey box approach considered in this chapter. Then, in Sect. 2, the neural models that will be used in the illustrative applications of Sect. 3, namely, the MultiLayer Perceptron (MLP), the Radial Basis Function Network (RBFN) and a kernel model trained by Support Vector Regression (SVR) are exposed. The examples of Sect. 3 are taken from a context representative of modern engine control problems, such as airpath control of a turbocharged Spark Ignition (SI) engine with Variable Camshaft Timing (VCT) (Sect. 3.2) and modeling of the in-cylinder residual gas fraction based on very few samples in order to limit the experimental costs (Sect. 3.3).

1.1 Common Features in Engine Control

The main function of the engine is to ensure the vehicle mobility by providing the power to the vehicle transmission. Nevertheless, the engine torque is also used for peripheral devices such as the air conditioning or the power steering. In order to provide the required torque, the engine control manages the engine actuators, such as ignition coils, injectors and air path actuators for a gasoline engine, pump and valve for diesel engine. Meanwhile, over a wide range of operating conditions, the engine control must satisfy some constraints: driver pleasure, fuel consumption and environmental standards.

G. Bloch et al.: *On Learning Machines for Engine Control*, Studies in Computational Intelligence (SCI) **132**, 125–144 (2008)
www.springerlink.com

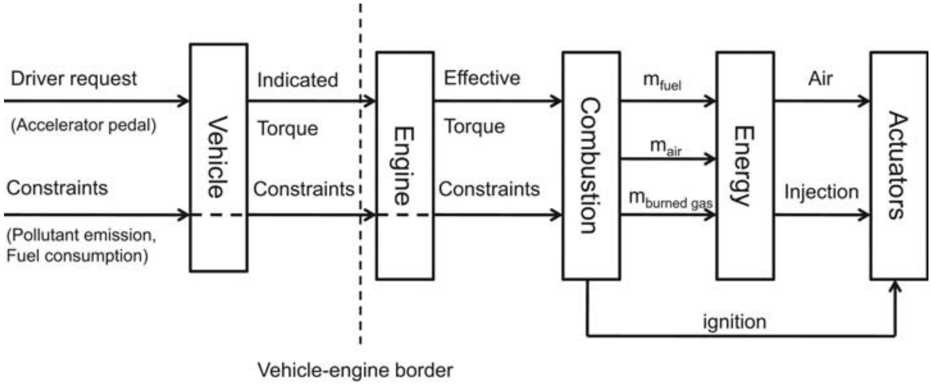

Fig. 1. Hierarchical torque control adapted from [13]

In [13], a hierarchical (or stratified) structure, shown in Fig. 1, is proposed for engine control. In this framework, the engine is considered as a torque source [18] with constraints on fuel consumption and pollutant emission. From the global characteristics of the vehicle, the *Vehicle layer* controls driver strategies and manages the links with other devices (gear box. . .). The *Engine layer* receives from the *Vehicle layer* the effective torque set point (with friction) and translates it into an indicated torque set point (without friction) for the combustion by using an internal model (often a map). The *Combustion layer* fixes the set points for the in-cylinder masses while taking into account the constraints on pollutant emissions. The *Energy layer* ensures the engine load with, e.g. the Air to Fuel Ratio (AFR) control and the turbo control. The lower level, specific for a given engine, is the *Actuator layer*, which controls, for instance, the throttle position, the injection and the ignition.

With the multiplication of complex actuators, advanced engine control is necessary to obtain an efficient torque control. This notably includes the control of the ignition coils, fuel injectors and air actuators (throttle, Exhaust Gas Recirculation (EGR), Variable Valve Timing (VVT), turbocharger. . .). The air actuator controllers generally used are PID controllers which are difficult to tune. Moreover, they often produce overshooting and bad set point tracking because of the system nonlinearities. Only model-based control can enhance engine torque control.

Several common characteristics can be found in engine control problems. First of all, the descriptive models are dynamic and nonlinear. They require a lot of work to be determined, particularly to fix the parameters specific to each engine type ("mapping"). For control, a sampling period depending on the engine speed (very short in the worst case) must be considered. The actuators present strong saturations. Moreover, many internal state variables are not measured, partly because of the physical impossibility of measuring and the difficulties in justifying the cost of setting up additional sensors. At a higher level, the control must be multi-objective in order to satisfy contradictory constraints (performance, comfort, consumption, pollution). Lastly, the control must be implemented in on-board computers (Electronic Control Units, ECU), whose computing power is increasing, but remains limited.

1.2 Neural Networks in Engine Control

Artificial neural networks have been the focus of a great deal of attention during the last two decades, due to their capabilities to solve nonlinear problems by learning from data. Although a broad range of neural network architectures can be found, MultiLayer Perceptrons (MLP) and Radial Basis Function Networks (RBFN) are the most popular neural models, particularly for system modeling and identification [47]. The universal approximation and flexibility properties of such models enable the development of modeling approaches, and then control and diagnosis schemes, which are independent of the specifics of the considered systems. As an example, the linearized neural model predictive control of a turbocharger is described in [12]. They allow the construction of nonlinear global models, static or dynamic. Moreover, neural models can be easily and generically differentiated so that a linearized model can be extracted at each sample time and used for the control design. Neural systems can then replace a combination of control algorithms and look-up tables used in traditional control systems and reduce the development effort and expertise required for the control system calibration of new engines. Neural networks can be used as observers or software sensors, in the context of a low number of measured variables. They enable the diagnosis of complex malfunctions by classifiers determined from a base of signatures.

First use of neural networks for automotive application can be traced back to early 90s. In 1991, Marko tested various neural classifiers for online diagnosis of engine control defects (misfires) and proposed a direct control by inverse neural model of an active suspension system [32]. In [40], Puskorius and Feldkamp, summarizing one decade of research, proposed neural nets for various subfunctions in engine control: AFR and idle speed control, misfire detection, catalyst monitoring, prediction of pollutant emissions. Indeed, since the beginning of the 90s, neural approaches have been proposed by numerous authors, for example, for:

- *Vehicle control.* Anti-lock braking system (ABS), active suspension, steering, speed control
- *Engine modeling.* Manifold pressure, air mass flow, volumetric efficiency, indicated pressure into cylinders, AFR, start-of-combustion for Homogeneous Charge Compression Ignition (HCCI), torque or power
- *Engine control.* Idle speed control, AFR control, transient fuel compensation (TFC), cylinder air charge control with VVT, ignition timing control, throttle, turbocharger, EGR control, pollutants reduction
- *Engine diagnosis.* Misfire and knock detection, spark voltage vector recognition systems

The works are too numerous to be referenced here. Nevertheless, the reader can consult the publications [1, 4, 5, 39, 45] and the references therein, for an overview.

More recently, Support Vector Machines (SVMs) have been proposed as another approach for nonlinear black box modeling [24, 41, 53] or monitoring [43] of automotive engines.

1.3 Grey Box Approach

Let us now focus on the development cycle of engine control, presented in Fig. 2, and the different models that are used in this framework. The design process is the following:

1. Building of an engine simulator mostly based on prior knowledge
2. First identification of control models from data provided by the simulator
3. Control scheme design
4. Simulation and pre-calibration of the control scheme with the simulator
5. Control validation with the simulator
6. Second identification of control models from data gathered on the engine
7. Calibration and final test of the control with the engine

This shows that, in current practice, more or less complex simulation environments based on physical relations are built for internal combustion engines. The great amount of knowledge that is included is consequently available. These simulators are built to be accurate, but this accuracy depends on many physical parameters which must be fixed. In any case, these simulation models cannot be used online, contrary to real time control models. Such control models, e.g. neural models, must be identified first from the simulator and then re-identified or adapted from experimental data. If the modeling process is improved, much gain can be expected for the overall control design process.

Relying in the control design on meaningful physical equations has a clear justification. This partially explains that the fully black box modeling approach has a difficult penetration in the engine control engineering community. Moreover the fully black box (e.g. neural) model based control solutions have still to practically prove their efficiency in terms of robustness, stability and real time applicability. This issue motivates the material presented in this chapter, which concentrates on developing modeling and control solutions, through several examples, mixing physical models and nonlinear black box models in a grey box approach. In

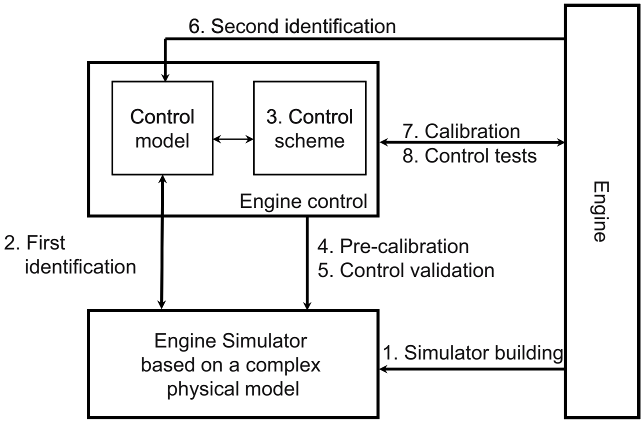

Fig. 2. Engine control development cycle

short, *use neural models whenever needed, i.e. whenever first-principles models are not sufficient*. In practice, this can be expressed in two forms:

- Neural models should be used to enhance – not replace – physical models, particularly by extending two-dimensional static maps or by correcting physical models when applied to real engines. This is developed in Sect. 3.2.
- Physical insights should be incorporated as prior knowledge into the learning of the neural models. This is developed in Sect. 3.3.

2 Neural Models

This section provides the necessary background on standard MultiLayer Perceptron (MLP) and Radial Basis Function (RBF) neural models, before presenting kernel models and support vector regression.

2.1 Two Neural Networks

As depicted in [47], a general neural model with a single output may be written as a function expansion of the form

$$f(\boldsymbol{\varphi}, \boldsymbol{\theta}) = \sum_{k=1}^{n} \alpha_k g_k(\boldsymbol{\varphi}) + \alpha_0, \tag{1}$$

where $\boldsymbol{\varphi} = [\varphi_1 \dots \varphi_i \dots \varphi_p]^T$ is the regression vector and $\boldsymbol{\theta}$ is the parameter vector.

The restriction of the multilayer perceptron to only one hidden layer and to a linear activation function at the output corresponds to a particular choice, the sigmoid function, for the basis function g_k, and to a "ridge" construction for the inputs in model (1). Although particular, this model will be called MLP in this chapter. Its form is given, for a single output f_{nn}, by

$$f_{nn}(\boldsymbol{\varphi}, \boldsymbol{\theta}) = \sum_{k=1}^{n} w_k^2 \; g\left(\sum_{j=1}^{p} w_{kj}^1 \varphi_j + b_k^1 \right) + b^2, \tag{2}$$

where $\boldsymbol{\theta}$ contains all the weights w_{kj}^1 and biases b_k^1 of the n hidden neurons together with the weights and bias w_k^2, b^2 of the output neuron, and where the activation function g is a sigmoid function (often the hyperbolic tangent $g(x) = 2/(1+e^{-2x}) - 1$).

On the other hand, choosing a Gaussian function $g(x) = \exp\left(-x^2/\sigma^2\right)$ as basis function and a radial construction for the inputs leads to the radial basis function network (RBFN) [38], of which the output is given by

$$\begin{aligned} f(\boldsymbol{\varphi}, \boldsymbol{\theta}) &= \sum_{k=1}^{n} \alpha_k g\left(\|\boldsymbol{\varphi} - \boldsymbol{\gamma}_k\|_{\boldsymbol{\sigma}_k}\right) + \alpha_0 \\ &= \sum_{k=1}^{n} \alpha_k \exp\left(-\frac{1}{2}\sum_{j=1}^{p} \frac{(\varphi_j - \gamma_{kj})^2}{\sigma_{kj}^2}\right) + \alpha_0, \end{aligned} \tag{3}$$

where $\boldsymbol{\gamma}_k = [\gamma_{k1} \dots \gamma_{kp}]^T$ is the "center" or "position" of the kth Gaussian and $\boldsymbol{\sigma}_k = [\sigma_{k1} \dots \sigma_{kp}]^T$ its "scale" or "width", most of the time with $\sigma_{kj} = \sigma_k,\ \forall j$, or even $\sigma_{kj} = \sigma,\ \forall j, k$.

The process of approximating nonlinear relationships from data with these models can be decomposed in several steps:

- Determining the structure of the regression vector $\boldsymbol{\varphi}$ or selecting the inputs of the network, see, e.g. [46] for dynamic system identification
- Choosing the nonlinear mapping f or, in the neural network terminology, selecting an internal network architecture, see, e.g. [42] for MLP's pruning or [37] for RBFN's center selection
- Estimating the parameter vector $\boldsymbol{\theta}$, i.e. (weight) "learning" or "training"
- Validating the model

This approach is similar to the classical one for linear system identification [29], the selection of the model structure being, nevertheless, more involved. For a more detailed description of the training and validation procedures, see [7] or [36].

Among the numerous nonlinear models, neural or not, which can be used to estimate a nonlinear relationship, the advantages of the one hidden layer perceptron, as well as those of the radial basis function network, can be summarized as follows: they are *flexible and parsimonious nonlinear black box models, with universal approximation capabilities* [6].

2.2 Kernel Expansion Models and Support Vector Regression

In the past decade, kernel methods [44] have attracted much attention in a large variety of fields and applications: classification and pattern recognition, regression, density estimation, etc. Indeed, using kernel functions, many linear methods can be extended to the nonlinear case in an almost straightforward manner, while avoiding the curse of dimensionality by transposing the focus from the data dimension to the number of data. In particular, Support Vector Regression (SVR), stemming from statistical learning theory [52] and based on the same concepts as the Support Vector Machine (SVM) for classification, offers an interesting alternative both for nonlinear modeling and system identification [16, 33, 54].

SVR originally consists in finding the kernel model that has at most a deviation ε from the training samples with the smallest complexity [48]. Thus, SVR amounts to solving a constrained optimization problem known as a quadratic program (QP), where both the ℓ_1-norm of the errors larger than ε and the ℓ_2-norm of the parameters are minimized. Other formulations of the SVR problem minimizing the ℓ_1-norm of the parameters can be derived to yield linear programs (LP) [31, 49]. Some advantages of this latter approach can be noticed compared to the QP formulation such as an increased sparsity of support vectors or the ability to use more general kernels [30]. The remaining of this chapter will thus focus on the LP formulation of SVR (LP-SVR).

Nonlinear Mapping and Kernel Functions

A kernel model is an expansion of the inner products by the N training samples $\mathbf{x}_i \in \mathbb{R}^p$ mapped in a higher dimensional feature space. Defining the *kernel function* $k(\mathbf{x}, \mathbf{x}_i) = \mathbf{\Phi}(\mathbf{x})^T \mathbf{\Phi}(\mathbf{x}_i)$, where $\mathbf{\Phi}(\mathbf{x})$ is the image of the point $\mathbf{x}$ in that feature space, allows to write the model as a kernel expansion

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b = \mathbf{K}(\mathbf{x}, \mathbf{X}^T)\boldsymbol{\alpha} + b, \tag{4}$$

where $\boldsymbol{\alpha} = [\alpha_1 \dots \alpha_i \dots \alpha_N]^T$ and b are the parameters of the model, the data $(\mathbf{x}_i, y_i)$, $i = 1, \dots, N$, are stacked as rows in the matrix $\mathbf{X} \in \mathbb{R}^{N \times p}$ and the vector $\mathbf{y}$, and $\mathbf{K}(\mathbf{x}, \mathbf{X}^T)$ is a vector defined as follows. For $\mathbf{A} \in \mathbb{R}^{p \times m}$ and $\mathbf{B} \in \mathbb{R}^{p \times n}$ containing p-dimensional sample vectors, the "kernel" $\mathbf{K}(\mathbf{A}, \mathbf{B})$ maps $\mathbb{R}^{p \times m} \times \mathbb{R}^{p \times n}$ in $\mathbb{R}^{m \times n}$ with $\mathbf{K}(\mathbf{A}, \mathbf{B})_{i,j} = k(\mathbf{A}_i, \mathbf{B}_j)$, where $\mathbf{A}_i$ and $\mathbf{B}_j$ are the ith and jth columns of $\mathbf{A}$ and $\mathbf{B}$. Typical kernel functions are the linear $(k(\mathbf{x}, \mathbf{x}_i) = \mathbf{x}^T \mathbf{x}_i)$, Gaussian RBF $(k(\mathbf{x}, \mathbf{x}_i) = \exp(-\|\mathbf{x} - \mathbf{x}_i\|_2^2 / 2\sigma^2))$ and polynomial $(k(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x}^T \mathbf{x}_i + 1)^d)$ kernels. The kernel function defines the feature space $\mathcal{F}$ in which the data are implicitly mapped. The higher the dimension of $\mathcal{F}$, the higher the approximation capacity of the function f, up to the universal approximation capacity obtained for an infinite feature space, as with Gaussian RBF kernels.

Support Vector Regression by Linear Programming

In Linear Programming Support Vector Regression (LP-SVR), the model complexity, measured by the ℓ_1-norm of the parameters $\boldsymbol{\alpha}$, is minimized together with the error on the data, measured by the ε-insensitive loss function l, defined by [52] as

$$l(y_i - f(\mathbf{x}_i)) = \begin{cases} 0 & \text{if } |y_i - f(\mathbf{x}_i)| \le \varepsilon, \\ |y_i - f(\mathbf{x}_i)| - \varepsilon & \text{otherwise.} \end{cases} \tag{5}$$

Minimizing the complexity of the model allows to control its generalization capacity. In practice, this amounts to penalizing non-smooth functions and implements the general smoothness assumption that two samples close in input space tend to give the same output.

Following the approach of [31], two sets of optimization variables, in two positive slack vectors $\mathbf{a}$ and $\boldsymbol{\xi}$, are introduced to yield a linear program solvable by standard optimization routines such as the MATLAB *linprog* function. In this scheme, the LP-SVR problem may be written as

$$\begin{array}{lccc} & \min\limits_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi} \ge \mathbf{0}, \mathbf{a} \ge \mathbf{0})} & \mathbf{1}^T \mathbf{a} + C \mathbf{1}^T \boldsymbol{\xi} & \\ \text{s.t.} & -\boldsymbol{\xi} \le & \mathbf{K}(\mathbf{X}, \mathbf{X}^T)\boldsymbol{\alpha} + b\mathbf{1} - \mathbf{y} & \le \boldsymbol{\xi} \\ & 0 \le & \mathbf{1}\varepsilon & \le \boldsymbol{\xi} \\ & -\mathbf{a} \le & \boldsymbol{\alpha} & \le \mathbf{a}, \end{array} \tag{6}$$

where a hyperparameter C is introduced to tune the trade-off between the minimization of the model complexity and the minimization of the error. The last set of constraints ensures that $\mathbf{1}^T \mathbf{a}$, which is minimized, bounds $\|\boldsymbol{\alpha}\|_1$. In practice, sparsity is obtained as a certain number of parameters α_i will tend to zero. The input vectors $\mathbf{x}_i$ for which the corresponding α_i are non-zero are called *support vectors* (SVs).

2.3 Link Between Support Vector Regression and RBFNs

For a Gaussian kernel, the kernel expansion (4) can be interpreted as a RBFN with N neurons in the hidden layer centered at the training samples $\mathbf{x}_i$ and with a unique width $\boldsymbol{\sigma}_k = [\sigma \dots \sigma]^T$, $k = 1, \dots, N$. Compared to neural networks, SVR has the following advantages: automatic selection and sparsity of the model, intrinsic regularization, no local minima (convex problem with a unique solution), and good generalization ability from a limited amount of samples.

It seems though that least squares estimates of the parameters or standard RBFN training algorithms are most of the time satisfactory, particularly when a sufficiently large number of samples corrupted by Gaussian noise is available. Moreover, in this case, standard center selection algorithms may be faster and yield a sparser model than SVR. However, in difficult cases, the good generalization capacity and the better behavior with respect to outliers of SVR may help. Even if non-quadratic criteria have been proposed to train [9] or prune neural networks [25, 51], the SVR loss function is intrinsically robust and thus allows accommodation to non-Gaussian noise probability density functions. In practice, it is advised to employ SVR in the following cases:

- Few data points are available.
- The noise is non-Gaussian.
- The training set is corrupted by outliers.

Finally, the computational framework of SVR allows for easier extensions such as the one described in this chapter, namely, the inclusion of prior knowledge.

3 Engine Control Applications

3.1 Introduction

The application treated here, the control of the turbocharged Spark Ignition engine with Variable Camshaft Timing, is representative of modern engine control problems. Indeed, such an engine presents for control the common characteristics mentioned in Sect. 1.1 and comprises several air actuators and therefore several degrees of freedom for airpath control.

More stringent standards are being imposed to reduce fuel consumption and pollutant emissions for Spark Ignited (SI) engines. In this context, downsizing appears as a major way for reducing fuel consumption while maintaining the advantage of low emission capability of three-way catalytic systems and combining several well known technologies [28]. (Engine) downsizing is the use of a smaller capacity engine operating at higher specific engine loads, i.e. at better efficiency points. In order to feed the engine, a well-adapted turbocharger seems to be the best solution. Unfortunately, the turbocharger inertia involves long torque transient responses [28]. This problem can be partially solved by combining turbocharging and Variable Camshaft Timing (VCT) which allows air scavenging from the intake to the exhaust.

The air intake of a turbocharged SI Engine with VCT, represented in Fig. 3, can be described as follows. The compressor (pressure p_{int}) produces a flow from the ambient air (pressure p_{amb} and temperature T_{amb}). This air flow Q_{th} is adjusted by the intake throttle (section S_{th}) and enters the intake manifold (pressure p_{man} and temperature T_{man}). The flow that goes into the cylinders Q_{cyl} passes through the intake valves, whose timing is controlled by the intake Variable Camshaft Timing VCT_{in} actuator. After the combustion, the gases are expelled into the exhaust manifold through the exhaust valve, controlled by the exhaust Variable Camshaft Timing VCT_{exh} actuator. The exhaust flow is split into turbine flow and wastegate flow. The turbine flow powers up the turbine and drives the compressor through a shaft. Thus, the supercharged pressure p_{int} is adjusted by the turbine flow which is controlled by the wastegate WG.

The effects of Variable Camshaft Timing (VCT) can be summarized as follows. On the one hand, cam timing can inhibit the production of nitrogen oxides (NO_x). Indeed, by acting on the cam timing, combustion products which would otherwise be expelled during the exhaust stroke are retained in the cylinder during the subsequent intake stroke. This dilution of the mixture in the cylinder reduces the combustion temperature and limits the NO_x formation. Therefore, it is important to estimate and control the back-flow of burned gases in the cylinder. On the other hand, with camshaft timing, air scavenging can appear, that is air passing directly from the intake to the exhaust through the cylinder. For that, the intake manifold pressure must be greater than the exhaust pressure when the exhaust and intake valves are opened together. In that case, the engine torque dynamic behavior is improved, i.e. the settling times decreased. Indeed, the flow which

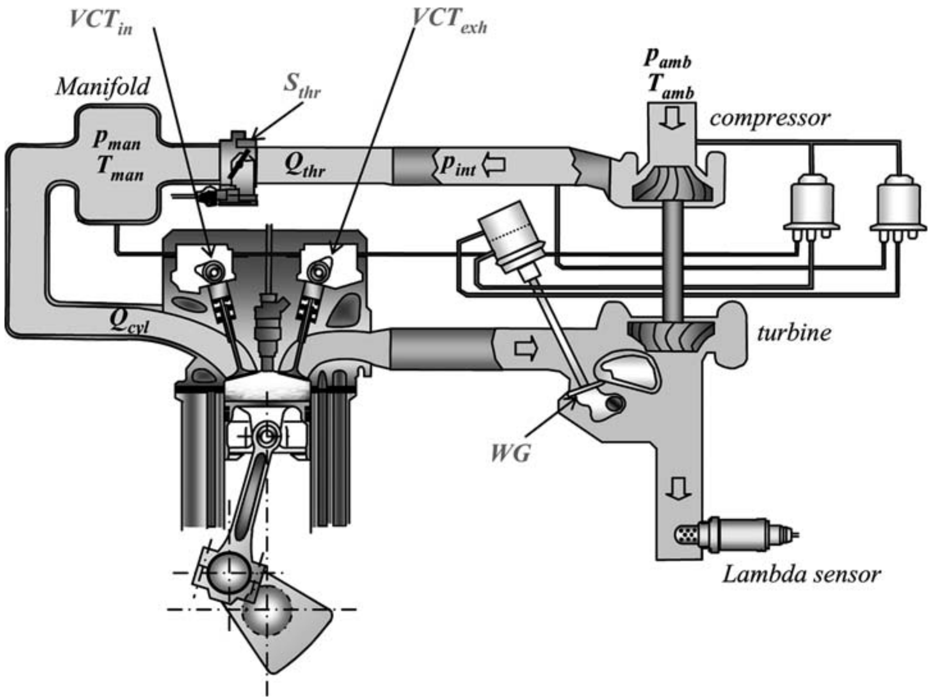

Fig. 3. Airpath of a turbocharged SI engine with VCT

passes through the turbine is increased and the corresponding energy is transmitted to the compressor. In transient, it is also very important to estimate and control this scavenging for torque control.

For such an engine, the following presents the inclusion of neural models in various modeling and control schemes in two parts: an air path control based on an in-cylinder air mass observer, and an in-cylinder residual gas estimation. In the first example, the air mass observer will be necessary to correct the manifold pressure set point. The second example deals with the estimation of residual has gases for a single cylinder naturally-aspirated engine. In this type of engine, no scavenging appears, so that the estimation of burned gases and air scavenging of the first example are simplified into a residual gas estimation.

3.2 Airpath Observer Based Control

Control Scheme

The objective of engine control is to supply the torque requested by the driver while minimizing the pollutant emissions. For a SI engine, the torque is directly linked to the air mass trapped in the cylinder for a given engine speed N_e and an efficient control of this air mass is then required. The air path control, i.e. throttle, turbocharger and variable camshaft timing (VCT) control, can be divided in two main parts: the air mass control by the throttle and the turbocharger and the control of the gas mix by the variable camshaft timing (see [12] for further details on VCT control). The structure of the air mass control scheme, described in Fig. 4, is now detailed block by block. The supervisor, that corresponds to a part of the *Combustion layer*

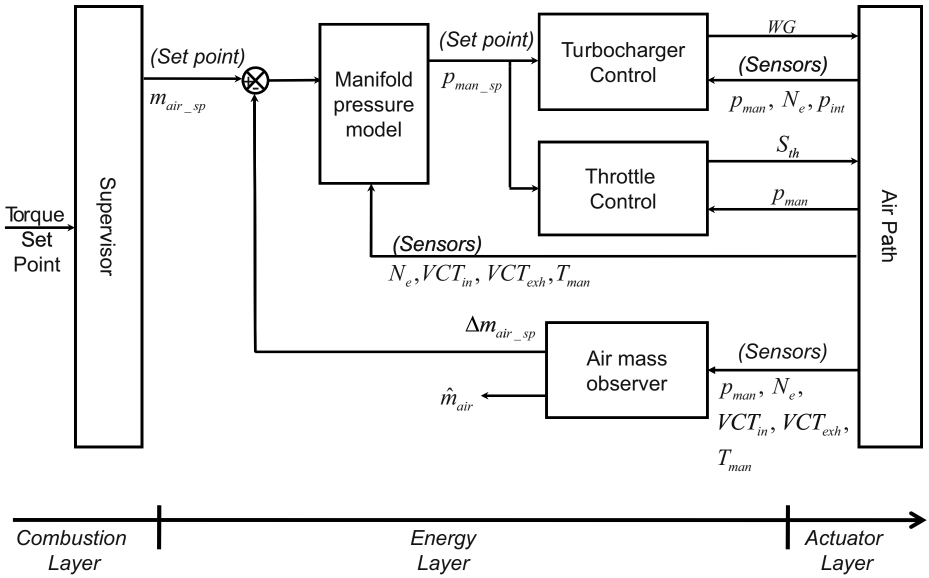

Fig. 4. General control scheme

of Fig. 1, builds the in-cylinder air mass set point from the indicated torque set point, computed by the *Engine layer*. The determination of manifold pressure set points is presented at the end of the section. The general control structure uses an in-cylinder air mass observer discussed below that corrects the errors of the manifold pressure model. The remaining blocks are not described in this chapter but an Internal Model Control (IMC) of the throttle is proposed in [12] and a linearized neural Model Predictive Control (MPC) of the turbocharger can be found in [11, 12]. The IMC scheme relies on a grey box model, which includes a neural static estimator. The MPC scheme is based on a dynamical neural model of the turbocharger.

Observation Scheme

Here two nonlinear estimators of the air variables, the recirculated gas mass RGM and the in-cylinder air mass m_{air}, are presented. Because these variables are not measured, data provided by a complex but accurate high frequency engine simulator [27] are used to build the corresponding models.

Because scavenging and burned gas back-flow correspond to associated flow phenomena, only one variable, the Recirculated Gas Mass (RGM), is defined

$$RGM = \begin{cases} m_{bg}, & \text{if } m_{bg} > m_{sc} \\ -m_{sc}, & \text{otherwise,} \end{cases} \tag{7}$$

where m_{bg} is the in-cylinder burned gas mass and m_{sc} is the scavenged air mass. Note that, when scavenging from the intake to the exhaust occurs, the burned gases are insignificant. The recirculated gas mass RGM estimator is a neural model entirely obtained from the simulated data.

Considering in-cylinder air mass observation, a lot of references are available especially for air-fuel ratio (AFR) control in a classical engine [21]. More recently, [50] uses an “input observer” to determine the engine cylinder flow and [3] uses a Kalman filter to reconstruct the air mass for a turbocharged SI engine.

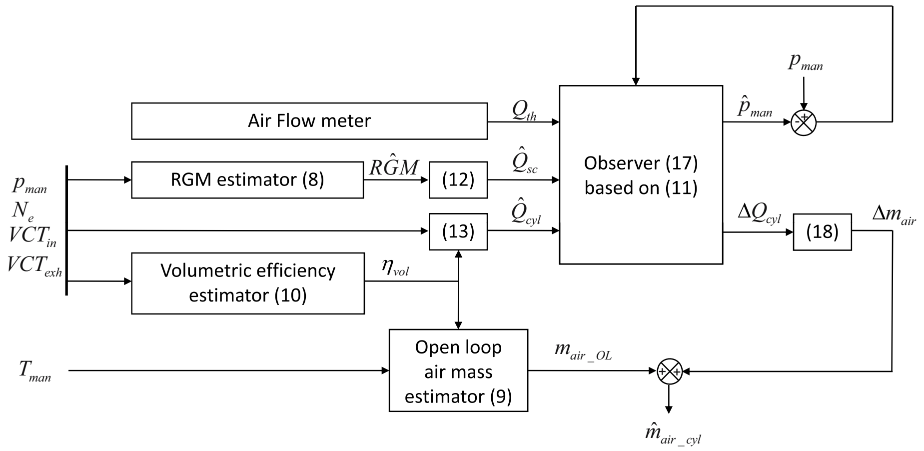

Fig. 5. Air mass observer scheme

A novel observer for the in-cylinder air mass m_{air} is presented below. Contrary to the references above, it takes into account a non measured phenomenon (scavenging), and can thus be applied with advanced engine technology (turbocharged VCT engine). Moreover, its on-line computational load is low. As presented in Fig. 5, this observer combines open loop nonlinear neural based statical estimators of RGM and m_{air}, and a "closed loop" polytopic observer. The observer is built from the Linear Parameter Varying model of the intake manifold and dynamically compensates for the residual error ΔQ_{cyl} committed by one of the estimators, based on a principle similar to the one presented in [2].

Open Loop Estimators

Recirculated Gas Mass Model

Studying the RGM variable (7) is complex because it cannot be measured on-line. Consequently, a static model is built from data provided by the engine simulator. The perceptron with one hidden layer and a linear output unit (2) is chosen with a hyperbolic tangent activation function g.

The choice of the regressors φ_j is based on physical considerations and the estimated Recirculated Gas Mass $\widehat{RGM}$ is given by

$$\widehat{RGM} = f_{nn}(p_{man}, N_e, VCT_{in}, VCT_{exh}), \tag{8}$$

where p_{man} is the intake manifold pressure, N_e the engine speed, VCT_{in} the intake camshaft timing, and VCT_{exh} the exhaust camshaft timing.

Open Loop Air Mass Estimator

The open loop model m_{air_OL} of the in-cylinder air mass is based on the volumetric efficiency equation

$$m_{air\_OL} = \eta_{vol} \frac{p_{amb} V_{cyl}}{r T_{man}}, \tag{9}$$

where T_{man} is the manifold temperature, p_{amb} the ambient pressure, V_{cyl} the displacement volume, r the perfect gas constant, and where the volumetric efficiency η_{vol} is described by the static nonlinear function f of four variables: p_{man}, N_e, VCT_{in} and VCT_{exh}.

In [15], various black box models, such as polynomial, spline, MLP and RBFN models, are compared for the static prediction of the volumetric efficiency. In [10], three models of the function f, obtained from engine simulator data, are compared: a polynomial model linear in manifold pressure proposed by Jankovic [23] $f_1(N_e, VCT_{in}, VCT_{exh})p_{man} + f_2(N_e, VCT_{in}, VCT_{exh})$, where f_1 et f_2 are fourth order polynomials, complete with 69 parameters, then reduced by stepwise regression to 43 parameters; a standard fourth order polynomial model $f_3(p_{man}, N_e, VCT_{in}, VCT_{exh})$, complete with 70 parameters then reduced to 58 parameters; and a MLP model with six hidden neurons (37 parameters)

$$\eta_{vol} = f_{nn}(p_{man}, N_e, VCT_{in}, VCT_{exh}). \tag{10}$$

Training of the neural model has been performed by minimizing the mean squared error, using the Levenberg–Marquardt algorithm. The behavior of these models is similar, and the most important errors are committed at the same operating points. Nevertheless, the neural model, that involves the smallest number of parameters and yields slightly better approximation results, is chosen as the static model of the volumetric efficiency. These results illustrate the parsimony property of the neural models.

Air Mass Observer

Principle

The air mass observer is based on the flow balance in the intake manifold. As shown in Fig. 6, a flow Q_{th} enters the manifold and two flows leave it: the flow that is captured in the cylinder Q_{cyl} and the flow scavenged from the intake to the exhaust Q_{sc}. The flow balance in the manifold can thus be written as

$$\dot{p}_{man}(t) = \frac{rT_{man}(t)}{V_{man}}\left(Q_{th}(t) - Q_{cyl}(t) - \Delta Q_{cyl}(t) - Q_{sc}(t)\right), \tag{11}$$

where, for the intake manifold, p_{man} is the pressure to be estimated (in Pa), T_{man} is the temperature (K), V_{man} is the volume (m^3), supposed to be constant and r is the ideal gas constant. In (11), Q_{th} can be measured by an air flow meter (kg s^{-1}). On the other hand, Q_{sc} (kg s^{-1}) and Q_{cyl} (kg s^{-1}) are respectively estimated by differentiating the Recirculated Gas Mass $\widehat{RGM}$ (8)

$$\hat{Q}_{sc} = \min(-\widehat{RGM}, 0)/t_{tdc}, \tag{12}$$

where $t_{tdc} = \frac{2 \times 60}{N_e\ n_{cyl}}$ is the variable sampling period between two intake top dead center (TDC), and by

$$\hat{Q}_{cyl}(t) = \eta_{vol}(t)\frac{p_{amb}(t)\ V_{cyl}\ N_e(t)\ n_{cyl}}{rT_{man}(t) 2 \times 60}, \tag{13}$$

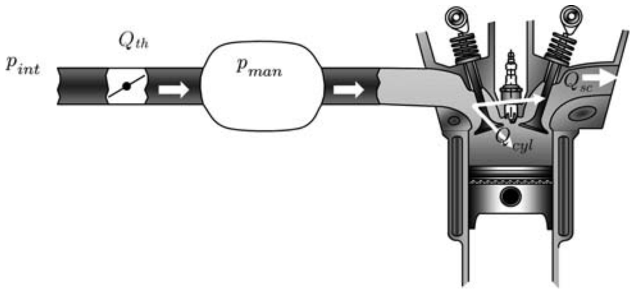

Fig. 6. Intake manifold and cylinder. From the intake manifold, the throttle air flow Q_{th} is divided into in-cylinder air flow Q_{cyl} and air scavenged flow Q_{sc}

where η_{vol} is given by the neural model (10), p_{amb} (Pa) is the ambient pressure, V_{cyl} (m^3) is the displacement volume, N_e (rpm) is the engine speed and n_{cyl} is the number of cylinders. The remaining term in (11), ΔQ_{cyl}, is the error made by the model (13).

Considering slow variations of ΔQ_{cyl}, i.e. $\dot{\Delta Q}_{cyl}(t) = 0$, and after discretization at each top dead center (TDC), thus with a variable sampling period $t_{tdc}(k) = \frac{2 \times 60}{N_e(k)\ n_{cyl}}$, the corresponding state space representation can be written as

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{A}\ \mathbf{x}_k + \mathbf{B}\ \mathbf{u}_k \\ y_k = \mathbf{C}\ \mathbf{x}_k, \end{cases} \tag{14}$$

where

$$\mathbf{x}_k = \begin{bmatrix} p_{man}(k) \\ \Delta Q_{cyl}(k) \end{bmatrix}, \quad \mathbf{u}_k = \begin{bmatrix} Q_{th}(k) \\ Q_{cyl}(k) \\ Q_{sc}(k) \end{bmatrix}, \quad y_k = p_{man}(k), \tag{15}$$

and, defining $\rho(k) = -\frac{r\ T_{man}(k)}{V_{man}} t_{tdc}(k)$, where

$$\mathbf{A} = \begin{bmatrix} 1 & \rho(k) \\ 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} -\rho(k) & \rho(k) & \rho(k) \\ 0 & 0 & 0 \end{bmatrix}. \tag{16}$$

Note that this system is Linear Parameter Varying (LPV), because the matrices **A** and **B** depend linearly on the (measured) parameter $\rho(k)$, which depends on the manifold temperature $T_{man}(k)$ and the engine speed $N_e(k)$.

The state reconstruction for system (14) can be achieved by resorting to the so-called polytopic observer of the form

$$\begin{cases} \hat{\mathbf{x}}_{k+1} = \mathbf{A}(\rho_k)\hat{\mathbf{x}}_k + \mathbf{B}(\rho_k)\mathbf{u}_k + \mathbf{K}(y_k - \hat{y}_k) \\ \hat{y}_k = \mathbf{C}\hat{\mathbf{x}}_k, \end{cases} \tag{17}$$

with a constant gain **K**.

This gain is obtained by solving a Linear Matrix Inequality (LMI). This LMI ensures the convergence towards zero of the reconstruction error for the whole operating domain of the system based on its polytopic decomposition. This ensures the global convergence of the observer. See [34, 35] and [14] for further details.

Then, the state ΔQ_{cyl} is integrated (i.e. multiplied by t_{tdc}) to give the air mass bias

$$\Delta m_{air} = \Delta Q_{cyl} \times t_{tdc}. \tag{18}$$

Finally, the in-cylinder air mass can be estimated by correcting the open loop estimator (9) with this bias as

$$\hat{m}_{air\_cyl} = m_{air\_OL} + \Delta m_{air}. \tag{19}$$

Results

Some experimental results, normalized between 0 and 1, obtained on a 1.8-Liter turbocharged four cylinder engine with Variable Camshaft Timing are given in Fig. 7. A measurement of the in-cylinder air mass, only valid in steady state, can be obtained from the measurement of Q_{th} by an air flow meter. Indeed, in steady state with no scavenging, the air flow that gets into the cylinder Q_{cyl} is equal to the flow that passes through the throttle Q_{th} (see Fig. 6). In consequence, this air mass measurement is obtained by integrating Q_{th} (i.e. multiplying by t_{tdc}). Figure 7 compares this measurement, the open loop neural estimator ((9) with a neural model (10)), an estimation not based on this neural model (observer (17) based on model (11) but with $Q_{cyl} = Q_{sc} = 0$), the proposed estimation ((19) combining the open loop neural estimator (9) and the polytopic observer (17) based on model (11) with Q_{cyl} given by (13) using the neural model (10) and Q_{sc} given by (12) using (8)).

For steps of air flow, the open loop neural estimator tracks very quickly the measurement changes, but a small steady state error can be observed (see for example between 32 s and 34 s). Conversely, the closed loop

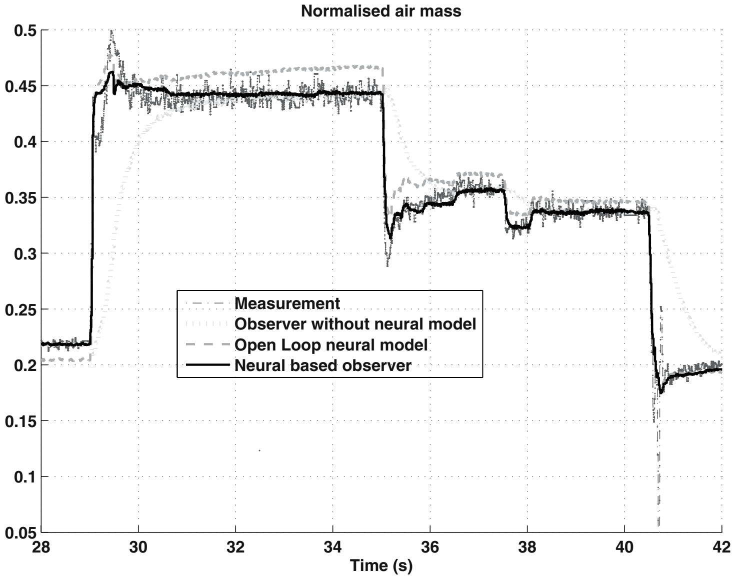

Fig. 7. Air mass observer results (mg) vs. time (s) on an engine test bench

observer which does not take into account this feedforward estimator involves a long transient error while guarantying the convergence in steady state. Finally, the proposed estimator, including feedforward statical estimators and a polytopic observer, combines both the advantages: very fast tracking and no steady state error. This observer can be used to design and improve the engine supervisor of Fig. 5 by determining the air mass set points.

Computing the Manifold Pressure Set Points

To obtain the desired torque of a SI engine, the air mass trapped in the cylinder must be precisely controlled. The corresponding measurable variable is the manifold pressure. Without Variable Camshaft Timing (VCT), this variable is linearly related to the trapped air mass, whereas with VCT, there is no more one-to-one correspondence. Figure 8 shows the relationship between the trapped air mass and the intake manifold pressure at three particular VCT positions for a fixed engine speed.

Thus, it is necessary to model the intake manifold pressure p_{man}. The chosen static model is a perceptron with one hidden layer (2). The regressors have been chosen from physical considerations: air mass m_{air} (corrected by the intake manifold temperature T_{man}), engine speed N_e, intake VCT_{in} and exhaust VCT_{exh} camshaft timing. The intake manifold pressure model is thus given by

$$p_{man} = f_{nn}\left(m_{air}, N_e, VCT_{in}, VCT_{exh}\right). \tag{20}$$

Training of the neural model from engine simulator data has been performed by minimizing the mean squared error, using the Levenberg–Marquardt algorithm.

The supervisor gives an air mass set point m_{air_sp} from the torque set point (Fig. 4). The intake manifold pressure set point, computed by model (20), is corrected by the error Δm_{air} (18) to yield the final set point p_{man_sp} as

$$p_{man\_sp} = f_{nn}\left(m_{air\_sp} - \Delta m_{air\_sp}, N_e, VCT_{in}, VCT_{exh}\right). \tag{21}$$

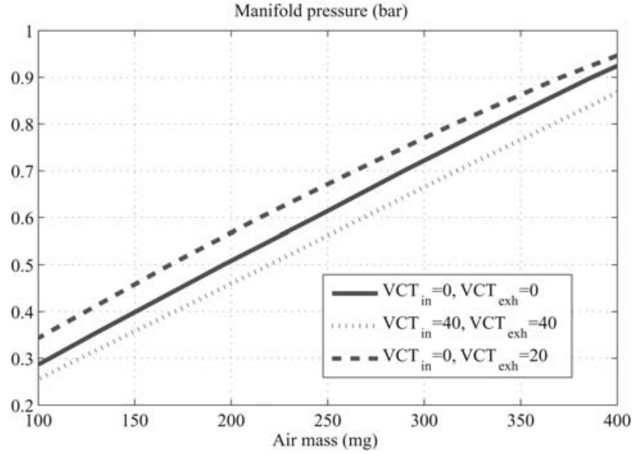

Fig. 8. Relationship between the manifold pressure (in bar) and the air mass trapped (in mg) for a SI engine with VCT at 2,000 rpm

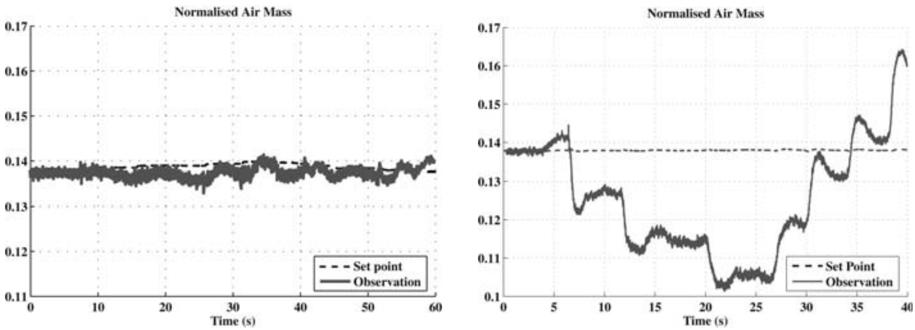

Fig. 9. Effect of the variation of VCTs on air mass with the proposed control scheme (*left*) and without taking into account the variation of VCTs in the control scheme (*right*)

Engine Test Bench Results

The right part of Fig. 9 shows an example of results for air mass control, in which the VCT variations are not taken into account. Considerable air mass variations (nearly $\pm 25\%$ of the set point) can be observed. On the contrary, the left part shows the corresponding results for the proposed air mass control. The air mass is almost constant (nearly $\pm 2\%$ of variation), illustrating that the manifold pressure set point is well computed with (21). This allows to reduce the pollutant emissions without degrading the torque set point tracking.

3.3 Estimation of In-Cylinder Residual Gas Fraction

The application deals with the estimation of residual gases in the cylinders of Spark Ignition (SI) engines with Variable Camshaft Timing (VCT) by Support Vector Regression (SVR) [8]. More precisely, we are interested in estimating the residual gas mass fraction by incorporating prior knowledge in the SVR learning with the general method proposed in [26]. Knowing this fraction allows to control torque as well as pollutant emissions. The residual gas mass fraction χ_{res} can be expressed as a function of the engine speed N_e, the ratio p_{man}/p_{exh}, where p_{man} and p_{exh} are respectively the (intake) manifold pressure and the exhaust pressure, and an overlapping factor OF (in °/m) [17], related to the time during which the valves are open together.

The available data are provided, on one hand, from the modeling and simulation environment Amesim [22], which uses a high frequency zero-dimensional thermodynamic model and, on the other hand, from off line measurements, which are accurate, but complex and costly to obtain, by direct in-cylinder sampling [19]. The problem is this. How to obtain a simple, embeddable, black box model with a good accuracy and a large validity range for the real engine, from precise real measurements as less numerous as possible and a representative, but possibly biased, prior simulation model? The problem thus posed, although particular, is very representative of numerous situations met in engine control, and more generally in engineering, where complex models, more or less accurate, exist and where the experimental data which can be used for calibration are difficult or expensive to obtain.

The simulator being biased but approximating rather well the overall shape of the function, the prior knowledge will be incorporated in the derivatives. Prior knowledge of the derivatives of a SVR model can be enforced in the training by noticing that the kernel expansion (4) is linear in the parameters $\boldsymbol{\alpha}$, which allows to write the derivative of the model output with respect to the scalar input x^j as

$$\frac{\partial f(\mathbf{x})}{\partial x^j} = \sum_{i=1}^{N} \alpha_i \frac{\partial k(\mathbf{x}, \mathbf{x}_i)}{\partial x^j} = \mathbf{r}_j(\mathbf{x})^T \boldsymbol{\alpha}, \tag{22}$$

where $\mathbf{r}_j(\mathbf{x}) = [\partial k(\mathbf{x}, \mathbf{x}_1)/\partial x^j \ldots \partial k(\mathbf{x}, \mathbf{x}_i)/\partial x^j \ldots \partial k(\mathbf{x}, \mathbf{x}_N)/\partial x^j]^T$ is of dimension N. The derivative (22) is linear in $\boldsymbol{\alpha}$. In fact, the form of the kernel expansion implies that the derivatives of any order with respect to any component are linear in $\boldsymbol{\alpha}$. Prior knowledge of the derivatives can thus be formulated as linear constraints.

The proposed model is trained by a variant of algorithm (6) with additional constraints on the derivatives at the points $\tilde{\mathbf{x}}_p$ of the simulation set. In the case where the training data do not cover the whole input space, extrapolation occurs, which can become a problem when using local kernels such as the RBF kernel. To avoid this problem, the simulation data $\tilde{\mathbf{x}}_p$, $p = 1, \ldots, N^{pr}$, are introduced as potential support vectors (SVs). The resulting global model is now

$$f(\mathbf{x}) = \mathbf{K}(\mathbf{x}, [\mathbf{X}^T\ \tilde{\mathbf{X}}^T])\boldsymbol{\alpha} + b, \tag{23}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{N+N^{pr}}$ and $[\mathbf{X}^T\ \tilde{\mathbf{X}}^T]$ is the concatenation of the matrices $\mathbf{X}^T = [\mathbf{x}_1 \ldots \mathbf{x}_i \ldots \mathbf{x}_N]$, containing the real data, and $\tilde{\mathbf{X}}^T = [\tilde{\mathbf{x}}_1 \ldots \tilde{\mathbf{x}}_p \ldots \tilde{\mathbf{x}}_{N^{pr}}]$, containing the simulation data. Defining the $N^{pr} \times (N + N^{pr})$-dimensional matrix $\mathbf{R}(\tilde{\mathbf{X}}^T, [\mathbf{X}^T\ \tilde{\mathbf{X}}^T]) = [\mathbf{r}_1(\tilde{\mathbf{x}}_1) \ldots \mathbf{r}_1(\tilde{\mathbf{x}}_p) \ldots \mathbf{r}_1(\tilde{\mathbf{x}}_{N^{pr}})]^T$, where $\mathbf{r}_1(\mathbf{x})$ corresponds to the derivative of (23) with respect to the input $x^1 = p_{man}/p_{exh}$, the proposed model is obtained by solving

$$\begin{aligned}
\min_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}, \mathbf{a}, \mathbf{z})} \quad & \frac{1}{N + N^{pr}} \mathbf{1}^T \mathbf{a} + \frac{C}{N} \mathbf{1}^T \boldsymbol{\xi} + \frac{\lambda}{N^{pr}} \mathbf{1}^T \mathbf{z} \\
\text{s.t.} \quad & -\boldsymbol{\xi} \leq \mathbf{K}(\mathbf{X}^T, [\mathbf{X}^T\ \tilde{\mathbf{X}}^T])\boldsymbol{\alpha} + b\mathbf{1} - \mathbf{y} \leq \boldsymbol{\xi} \\
& \mathbf{0} \leq \mathbf{1}\varepsilon \leq \boldsymbol{\xi} \\
& -\mathbf{a} \leq \boldsymbol{\alpha} \leq \mathbf{a} \\
& -\mathbf{z} \leq \mathbf{R}(\tilde{\mathbf{X}}^T, [\mathbf{X}^T\ \tilde{\mathbf{X}}^T])\boldsymbol{\alpha} - \mathbf{y}' \leq \mathbf{z},
\end{aligned} \tag{24}$$

where $\mathbf{y}'$ contains the N^{pr} known values of the derivative with respect to the input p_{man}/p_{exh}, at the points $\tilde{\mathbf{x}}_p$ of the simulation set. In order to be able to evaluate these values, a *prior model*, $\tilde{f}(\mathbf{x}) = \sum_{p=1}^{N^{pr}} \tilde{\alpha}_p k(\mathbf{x}, \tilde{\mathbf{x}}_p) + \tilde{b}$, is first trained on the N^{pr} simulation data $(\tilde{\mathbf{x}}_p, \tilde{y}_p)$, $p = 1, \ldots, N^{pr}$, only. This prior model is then used to provide the prior derivatives $\mathbf{y}' = [\partial \tilde{f}(\tilde{\mathbf{x}}_1)/\partial \tilde{x}^1 \ldots \partial \tilde{f}(\tilde{\mathbf{x}}_p)/\partial \tilde{x}^1 \ldots \partial \tilde{f}(\tilde{\mathbf{x}}_{N^{pr}})/\partial \tilde{x}^1]^T$.

Note that the knowledge of the derivatives is included by soft constraints, thus allowing to tune the trade-off between the data and the prior knowledge. The weighting hyperparameters are set to C/N and λ/N^{pr} in order to maintain the same order of magnitude between the regularization, error and prior knowledge terms in the objective function. This allows to ease the choice of C and λ based on the application goals and confidence in the prior knowledge. Hence, the hyperparameters become problem independent.

The method is now evaluated on the in-cylinder residual gas fraction application. In this experiment, three data sets are built from the available data composed of 26 experimental samples plus 26 simulation samples:

- The training set $(\mathbf{X}, \mathbf{y})$ composed of a limited amount of real data (N samples)
- The test set composed of independent real data ($26 - N$ samples)
- The simulation set $(\tilde{\mathbf{X}}, \tilde{\mathbf{y}})$ composed of data provided by the simulator ($N^{pr} = 26$ samples)

The test samples are assumed to be unknown during the training and are retained for testing only. It must be noted that the inputs of the simulation data do not exactly coincide with the inputs of the experimental data as shown in Fig. 10 for $N = 3$.

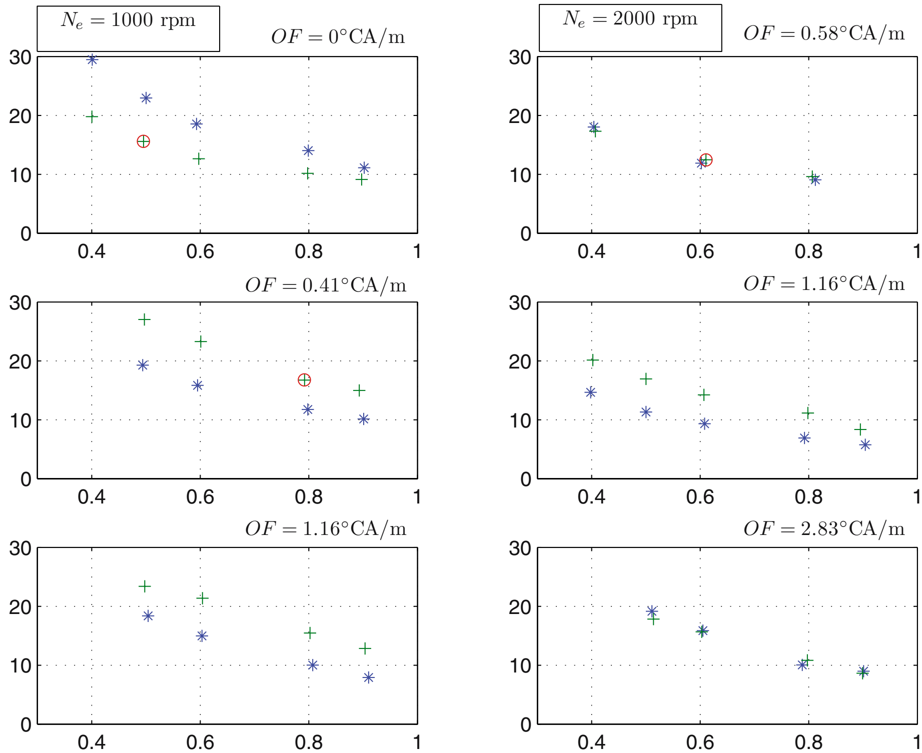

Fig. 10. Residual gas mass fraction χ_{res} in percentages as a function of the ratio p_{man}/p_{exh} for two engine speeds N_e and different overlapping factors OF. The 26 experimental data are represented by *plus signs* (+) with a *superposed circle* (⊕) for the three points retained as training samples. The 26 simulation data appear as *asterisks* (∗)

Table 1. Errors on the residual gas mass fraction with the number of real and simulation data used for training

Model	# of real data N	# of simulation data N^{pr}	RMSE test	RMSE	MAE
Experimental model	6		6.84	6.00	15.83
Prior model		26	4.86	4.93	9.74
Mixed model	6	26	4.85	4.88	9.75
Proposed model	6	26	**2.44**	**2.15**	**5.94**
Experimental model	3		–	–	–
Prior model		26	4.93	4.93	9.74
Mixed model	3	26	4.89	4.86	9.75
Proposed model	3	26	**2.97**	**2.79**	**5.78**

'–' appears when the result is irrelevant (model mostly constant)

The comparison is performed between four models:

- The *experimental model* trained by (6) on the real data set $(\mathbf{X}, \mathbf{y})$ only
- The *prior model* trained by (6) on the simulation data $(\tilde{\mathbf{X}}, \tilde{\mathbf{y}})$ only
- The *mixed model* trained by (6) on the real data set simply extended with the simulation data $([\mathbf{X}^T \ \tilde{\mathbf{X}}^T]^T, [\mathbf{y}^T \ \tilde{\mathbf{y}}^T]^T)$ (the training of this model is close in spirit to the virtual sample approach, where extra data are added to the training set)
- The *proposed model* trained by (24) and using both the real data $(\mathbf{X}, \mathbf{y})$ and the simulation data $(\tilde{\mathbf{X}}, \tilde{\mathbf{y}})$

These models are evaluated on the basis of three indicators: the root mean square error (RMSE) on the test set (*RMSE test*), the RMSE on all experimental data (*RMSE*) and the maximum absolute error on all experimental data (*MAE*).

Before training, the variables are normalized with respect to their mean and standard deviation. The different hyperparameters are set according to the following heuristics. One goal of the application is to obtain a model that is accurate on both the training and test samples (the training points are part of the performance index *RMSE*). Thus C is set to a large value ($C = 100$) in order to ensure a good approximation of the training points. Accordingly, ε is set to 0.001 in order to approximate the real data well. The trade-off parameter λ of the proposed method is set to 100, which gives as much weight to both the training data and the prior knowledge. Since all standard deviations of the inputs are equal to 1 after normalization, the RBF kernel width σ is set to 1.

Two sets of experiments are performed for very low numbers of training samples $N = 6$ and $N = 3$. The results in Table 1 show that both the *experimental* and the *mixed* models cannot yield a better approximation than the *prior model* with so few training data. Moreover, for $N = 3$, the *experimental model* yields a quasi-constant function due to the fact that the model has not enough free parameters (only three plus a bias term) and thus cannot model the data. In this case, the RMSE is irrelevant. On the contrary, the *proposed model* does not suffer from a lack of basis functions, thanks to the inclusion of the simulation data as potential support vectors. This model yields good results from very few training samples. Moreover, the performance decreases only slightly when reducing the training set size from 6 to 3. Thus, the proposed method seems to be a promising alternative to obtain a simple black box model with a good accuracy from a limited number of experimental data and a prior simulation model.

4 Conclusion

The chapter exposed learning machines for engine control applications. The two neural models most used in modeling for control, the MultiLayer Perceptron and the Radial Basis Function network, have been described, along with a more recent approach, known as Support Vector Regression. The use of such black box models has been placed in the design cycle of engine control, where the modeling steps constitute the bottleneck of the whole process. Application examples have been presented for a modern engine, a turbocharged Spark Ignition engine with Variable Camshaft Timing. In the first example, the airpath control was studied, where open loop neural estimators are combined with a dynamical polytopic observer. The second example considered modeling a variable which is not measurable on-line, from a limited amount of experimental data and a simulator built from prior knowledge.

The neural black box approach for modeling and control allows to develop generic, application independent, solutions. The price to pay is the loss of the physical interpretability of the resulting models. Moreover, the fully black box (e.g. neural) model based control solutions have still to practically prove their efficiency in terms of robustness or stability. On the other hand, models based on first principles (white box models) are completely meaningful and many control approaches with good properties have been proposed, which are well understood and accepted in the engine control community. However, these models are often inadequate, too complex or too difficult to parametrize, as real time control models. Therefore, intermediate solutions, involving grey box models, seem to be preferable for engine modeling, control and, at a higher level, optimization. In this framework, two approaches can be considered. First, besides first principles models, black box neural sub-models are chosen for variables difficult to model (e.g. volumetric efficiency, pollutant emissions). Secondly, black box models can be enhanced thanks to physical knowledge. The examples presented in this chapter showed how to implement these grey box approaches in order to obtain efficient and acceptable results.

References

1. C. Alippi, C. De Russis, and V. Piuri. A neural-network based control solution to air fuel ratio for automotive fuel injection system. *IEEE Transactions on Systems, Man, and Cybernetics – Part C*, 33(2):259–268, 2003.
2. P. Andersson. *Intake Air Dynamics on a Turbocharged SI Engine with Wastegate*. PhD thesis, Linköping University, Sweden, 2002.
3. P. Andersson and L. Eriksson. Mean-value observer for a turbocharged SI engine. In *Proc. of the IFAC Symp. on Advances in Automotive Control, Salerno, Italy*, pages 146–151, April 2004.
4. I. Arsie, C. Pianese, and M. Sorrentino. A procedure to enhance identification of recurrent neural networks for simulating air-fuel ratio dynamics in SI engines. *Engineering Applications of Artificial Intelligence*, 19(1):65–77, 2006.
5. I. Arsie, C. Pianese, and M. Sorrentino. Recurrent neural networks for AFR estimation and control in spark ignition automotive engines. Chapter 9 in this volume.
6. A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.
7. G. Bloch and T. Denoeux. Neural networks for process control and optimization: two industrial applications. *ISA Transactions*, 42(1):39–51, 2003.
8. G. Bloch, F. Lauer, G. Colin, and Y. Chamaillard. Combining experimental data and physical simulation models in support vector learning. In L. Iliadis and K. Margaritis, editors, *Proc. of the 10th Int. Conf. on Engineering Applications of Neural Networks (EANN), Thessaloniki, Greece*, volume 284 of *CEUR Workshop Proceedings*, pages 284–295, 2007.
9. G. Bloch, F. Sirou, V. Eustache, and P. Fatrez. Neural intelligent control of a steel plant. *IEEE Transactions on Neural Networks*, 8(4):910–918, 1997.
10. G. Colin. *Contrôle des systèmes rapides non linéaires – Application au moteur à allumage commandé turbocompressé à distribution variable*. PhD thesis, University of Orléans, France, 2006.
11. G. Colin, Y. Chamaillard, G. Bloch, and A. Charlet. Exact and linearised neural predictive control - a turbocharged SI engine example. *Journal of Dynamic Systems, Measurement and Control - Transactions of the ASME*, 129(4):527–533, 2007.
12. G. Colin, Y. Chamaillard, G. Bloch, and G. Corde. Neural control of fast nonlinear systems - Application to a turbocharged SI engine with VCT. *IEEE Transactions on Neural Networks*, 18(4):1101–1114, 2007.
13. G. Corde. Le contrôle moteur. In G. Gissinger and N. Le Fort Piat, editors, *Contrôle commande de la voiture*. Hermès, Paris, 2002.
14. J. Daafouz and J. Bernussou. Parameter dependent Lyapunov functions for discrete time systems with time varying parametric uncertainties. *Systems & Control Letters*, 43(5):355–359, August 2001.

15. G. De Nicolao, R. Scattolini, and C. Siviero. Modelling the volumetric efficiency of IC engines: parametric, non-parametric and neural techniques. *Control Engineering Practice*, 4(10):1405–1415, 1996.
16. P. M. L. Drezet and R. F. Harrison. Support vector machines for system identification. In *Proc. of the UKACC Int. Conf. on Control, Swansea, UK*, volume 1, pages 688–692, 1998.
17. J. W. Fox, W. K. Cheng, and J. B. Heywood. A model for predicting residual gas fraction in spark-ignition engines. *SAE Technical Papers*, (931025), 1993.
18. J. Gerhardt, H. Hönniger, and H. Bischof. A new approach to functionnal and software structure for engine management systems - BOSCH ME7. *SAE Technical Papers*, (980801), 1998.
19. P. Giansetti, G. Colin, P. Higelin, and Y. Chamaillard. Residual gas fraction measurement and computation. *International Journal of Engine Research*, 8(4):347–364, 2007.
20. L. Guzzella and C. H. Onder. *Introduction to Modeling and Control of Internal Combustion Engine Systems.* Springer, Berlin Heidelberg New York, 2004.
21. E. Hendricks and J. Luther. Model and observer based control of internal combustion engines. In *Proc. of the 1st Int. Workshop on Modeling Emissions and Control in Automotive Engines (MECA), Salerno, Italy*, pages 9–20, 2001.
22. Imagine. Amesim web site. www.amesim.com, 2006.
23. M. Jankovic and S. W. Magner. Variable Cam Timing : Consequences to automotive engine control design. In *Proc. of the 15th Triennial IFAC World Congress, Barcelona, Spain*, pages 271–276, 2002.
24. I. Kolmanovsky. Support vector machine-based determination of gasoline direct-injected engine admissible operating envelope. *SAE Technical Papers*, (2002-01-1301), 2002.
25. M. Lairi and G. Bloch. A neural network with minimal structure for maglev system modeling and control. In *Proc. of the IEEE Int. Symp. on Intelligent Control/Intelligent Systems & Semiotics, Cambridge, MA, USA*, pages 40–45, 1999.
26. F. Lauer and G. Bloch. Incorporating prior knowledge in support vector regression. *Machine Learning*, 70(1):89–118, January 2008.
27. F. Le Berr, M. Miche, G. Colin, G. Le Solliec, and F. Lafossas. Modelling of a turbocharged SI engine with variable camshaft timing for engine control purposes. *SAE Technical Paper*, (2006-01-3264), 2006.
28. B. Lecointe and G. Monnier. Downsizing a gasoline engine using turbocharging with direct injection. *SAE Technical Paper*, (2003-01-0542), 2003.
29. L. Ljung. *System Identification: Theory for the User*, 2nd edition. Prentice-Hall, Englewood Cliffs, NJ, 1999.
30. O. Mangasarian. Generalized support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146. MIT, Cambridge, MA, 2000.
31. O. L. Mangasarian and D. R. Musicant. Large scale kernel regression via linear programming. *Machine Learning*, 46(1–3):255–269, 2002.
32. K. A. Marko. Neural network application to diagnostics and control of vehicle control systems. In R. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 537–543. Morgan Kaufmann, San Mateo, CA, 1991.
33. D. Mattera and S. Haykin. Support vector machines for dynamic reconstruction of a chaotic system. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 211–241. MIT, Cambridge, MA, 1999.
34. G. Millérioux, F. Anstett, and G. Bloch. Considering the attractor structure of chaotic maps for observer-based synchronization problems. *Mathematics and Computers in Simulation*, 68(1):67–85, February 2005.
35. G. Millérioux, L. Rosier, G. Bloch, and J. Daafouz. Bounded state reconstruction error for LPV systems with estimated parameters. *IEEE Transactions on Automatic Control*, 49(8):1385–1389, August 2004.
36. O. Nelles. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models.* Springer, Berlin Heidelberg New York, 2001.
37. M. J. L. Orr. Recent advances in radial basis function networks. Technical report, Edinburgh University, UK, 1999.
38. T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of IEEE*, 78(10):1481–1497, 1990.
39. D. V. Prokhorov. Neural networks in automotive applications. Chapter 7 in this volume.
40. G. V. Puskorius and L. A. Feldkamp. Parameter-based Kalman filter training: theory and implementation. In S. Haykin, editor, *Kalman Filtering and Neural Networks*, chapter 2, pages 23–67. Wiley, New York, 2001.
41. A. Rakotomamonjy, R. Le Riche, D. Gualandris, and Z. Harchaoui. A comparison of statistical learning approaches for engine torque estimation. *Control Engineering Practice*, 16(1):43–55, 2008.
42. R. Reed. Pruning algorithms – a survey. *IEEE Transactions on Neural Networks*, 4:740–747, 1993.

43. M. Rychetsky, S. Ortmann, and M. Glesner. Support vector approaches for engine knock detection. In *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN), Washington, DC, USA*, volume 2, pages 969–974, 1999.
44. B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT, Cambridge, MA, 2001.
45. P. J. Shayler, M. S. Goodman, and T. Ma. The exploitation of neural networks in automotive engine management systems. *Engineering Applications of Artificial Intelligence*, 13(2):147–157, 2000.
46. J. Sjöberg and L. S. H. Ngia. Neural nets and related model structures for nonlinear system identification. In J. A. K. Suykens and J. Vandewalle, editors, *Nonlinear Modeling, Advanced Black-Box Techniques*, chapter 1, pages 1–28. Kluwer Academic Publishers, Boston, 1998.
47. J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P. Y. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12):1691–1724, 1995.
48. A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.
49. A. J. Smola, B. Schölkopf, and G. Rätsch. Linear programs for automatic accuracy control in regression. In *Proc. of the 9th Int. Conf. on Artificial Neural Networks, Edinburgh, UK*, volume 2, pages 575–580, 1999.
50. A. Stotsky and I. Kolmanovsky. Application of input estimation techniques to charge estimation and control in automotive engines. *Control Engineering Practice*, 10:1371–1383, 2002.
51. P. Thomas and G. Bloch. Robust pruning for multilayer perceptrons. In P. Borne, M. Ksouri, and A. El Kamel, editors, *Proc. of the IMACS/IEEE Multiconf. on Computational Engineering in Systems Applications, Nabeul-Hammamet, Tunisia*, volume 4, pages 17–22, 1998.
52. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, Berlin Heidelberg New York, 1995.
53. C.-M. Vong, P.-K. Wong, and Y.-P. Li. Prediction of automotive engine power and torque using least squares support vector machines and Bayesian inference. *Engineering Applications of Artificial Intelligence*, 19(3):277–287, 2006.
54. L. Zhang and Y. Xi. Nonlinear system identification based on an improved support vector regression estimator. In *Proc. of the Int. Symp. on Neural Networks, Dalian, China*, volume 3173 of *LNCS*, pages 586–591. Springer, Berlin Heidelberg New York, 2004.