A learning algorithm concept for updating look-up tables for automotive applications

C.Guardiola, B.Pla, D.Blanco-Rodriguez, P.Cabrera

CMT Motores Térmicos, Universidad Politécnica de Valencia, Camino de Vera s/n, E-46022 Valencia, Spain

Abstract

Look-up tables are commonly used in the automotive field for handling operating point variations. However, constant maps cannot cope with systems variations and ageing. Methods, such as Kalman filter or Extended Kalman filter for non-linear cases, can be used for table adaptation providing an optimal solution to the problem. But these methods are computationally intensive, making difficult to implement them on commercial engine control units. The current paper proposes a learning method for online updating of look-up tables or maps. This algorithm uses precalculated membership functions based on a standard Kalman filter observer for weighting the adaptation. The main contribution of the method is the derivation of a steady-state Kalman filter observer that lowers the calculation burden and simplifies the implementation, against standard Kalman filter implementation that requires higher computational cost. As far as table is updated online while engine runs, this allows correcting drift errors and the unit-to-unit dispersion. The method is illustrated for mapping engine variables such as λ^{-1} and NO_x in a Diesel engine by using an adaptive look-up table; and its characteristics make it suitable for implementing in commercial engine electronic control units for online purposes.

Keywords: Kalman filter; adaptive models; maps; look-up table; automotive; sensor;
PACS: 5.70.a, 89.40.Bb

1. Introduction

The need of information about systems is crucial in engineering applications. Models are an alternative to physical sensors, which are not always available or their responses are deficient in terms of cost, delay and dynamics. Look-up tables allow engineers to model systems that present complex expressions or are difficult to obtain, by means of mapping outputs with a set of nD heuristic array structures. Table outputs depend on n inputs and are generally calculated using interpolation between table elements. Tables are highly used in different engineering fields but current paper centres on automotive engines. In this field, a lot of parameters and functions must be modelled for the correct performance of systems. The increasing complexity of these (i.e selective catalyst reduction, exhaust gas recirculation, variable geometry turbine or diesel particulate filter, among others) requires each time a higher identification and calibration effort,

Email addresses: carguaga@mot.upv.es (C.Guardiola), benplamo@mot.upv.es (B.Pla), dablarod@mot.upv.es (D.Blanco-Rodriguez), pedcablo@mot.upv.es (P.Cabrera)

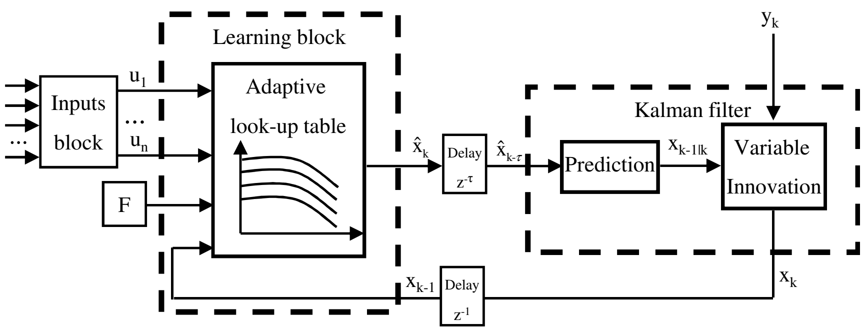

Figure 1: Schematic view of the procedure for estimating x_k [1].

both off-line and on-line. In addition to this, current engine control systems are commonly confided to fixed calibrations, with different maps based on look-up tables for compensating temperature, pressure or ageing variations, complicating even more the identification problem.

Main problem of using models for engine control is that variables change during lifespan due to different reasons, and constant maps do not cover these changes. This leads to a drifted response, where output presents bias that in general varies with time and other variables. For example, common inputs used on Diesel engines are injection (m_f) and speed (n), which generally define the engine operating point. (1) shows how table elements (θ_{ij}) vary with time and engine operating point for a 2D case ($n = 2$),

$$\frac{\mathrm{d}\theta_{ij}}{\mathrm{d}t} = \frac{\partial\theta_{ij}}{\partial t} + \frac{\partial\theta_{ij}}{\partial n}\frac{\mathrm{d}n}{\mathrm{d}t} + \frac{\partial\theta_{ij}}{\partial m_f}\frac{\mathrm{d}m_f}{\mathrm{d}t} + \dots \tag{1}$$

Adaptive modelling is a good solution for coping with this drifted performance. For this, some feedback about the considered variable is needed. This information can be obtained from different paths: sensor outputs (although these suffer delays and dynamical effects and are not available in all cases) and other models (analytical, maps or others).

A possible application is shown in figure 1, where three main blocks can be seen: Learning block, Inputs block and Kalman filter block.

The adaptation and interpolation of the table is included in the Learning block, where an adaptive table is used for mapping a given variable x_k. The table itself is a n-Dimensional matrix that constitutes a static model depending on n inputs ($u_1, ..., u_n$); F that represents the algorithm that controls the updating, which is the core of the paper; and x_{k-1} that is the reference value used for updating the table. Finally, $\hat{x}_k$ is the output of the table that is obtained by linear interpolation using the current table elements θ_{ij}, although other methods are possible.

Inputs block represents all the operations for obtaining the required n inputs ($u_1, ..., u_n$) for the model in the proper way: filtering, delaying, sensor outputs treatment, other modelisations, etc. For example, for modelling NO_x [2], EGR rate would have to be one of the inputs, but EGR is not directly available on Diesel engines, and then a model for getting EGR rate must be built. In addition to this, NO_x output will respond to EGR variation with some delay and some dynamics. This inputs block takes into account all these treatments. The Learning block will receive the inputs already treated.

Although table system represents a static model, matching the table output with a filter allows tracking dynamics. Kalman filter [3] block is built for observing x_k using the delayed table output $\hat{x}_{k-\tau}$ as input for the observer and some information source y_k as feedback. The discrete state-space model built for this observer contains the considered dynamics, which are included in the prediction block of the Kalman filter (see (21)). Variable innovation block uses y_k for correcting the observation x_k. This last is used as reference input for the learning block. Other authors have set out this kind of Kalman filter structure for bias cancellation of models from some information feedback [4, 5, 6, 7].

In addition to these blocks, at least two delay blocks must be included. The first one shifts $\hat{x}_k$ for taking into account possible delay τ of y_k. Typical examples are sensors, which in general present some pure delay, transport delay and some physic delay with respect to inputs actuation (although this last can also be treated in Inputs block). The second delay is included after the observation of x_k for delaying it just one iteration, because in every iteration, table output is the result of the last updating; i.e. table updating will always be one sample shifted.

The key aspect for the adaptation is then the design of the learning algorithm F and its performance depends highly on the existence of a reliable x_k value that gives feedback for compensating the error in every instant. In the field of adaptive look-up tables, some papers have been published treating the table elements as parameters that are identified by means of an extended Kalman filter (EKF) [4, 8]. Nevertheless, these methods present problems of calculation burden and must solve the local unobservability of table elements that are not active. This formulation also requires a big computational effort because heavy matrices must be calculated preventing it from being implemented in commercial electronic control units (ECUs) for map adaptation. The current paper proposes a novel learning algorithm that only affects matrix nodes involved in the interpolation process and updates table elements by means of membership functions. These functions have been previously calculated off-line from a derivation of a Kalman filter gain calculation. As in figure 1, the table system can be linked with a dynamic evolution like the proposed in the Kalman filter block for mapping dynamic systems.

Regarding tuning of the state-space system used in the Kalman block and the delay, there is a wide bibliography [1, 9, 10, 11, 12] on sensor characterisation and dynamical treatment using both on-line and off-line procedures.

The structure of the paper is the following: section 2 presents the problem description, section 3 gives some comment about interpolation phase; section 4 presents a review of methods for updating look-up tables on automotive applications and presents the proposed method for updating table elements; section 5 shows results on simulating the method, one in 1D and a second one in 2D; section 6 shows table updating on real engine and finally section 7 outlines the work conclusions. Section 2 to 5 are centred on Learning block, while section 6 recovers the full example of figure 1 with real engine data.

2. Problem description

The paper addresses the updating of a n dimensional look-up table T, where each dimension n presents d_n nodes. Table output in time t is referred as $\hat{x}(t)$, which is calculated as function of n inputs u_n by means of the interpolation of the 2^n observed table elements involved in every iteration:

$$\hat{x}(t) = f(T(t), u_1(t), ..., u_n(t)) \tag{2}$$

Current computer and programming systems are based on discrete formulation as far as systems are sampled, and the paper formulation will be presented on a discrete basis (3). Normal frequency for engine variables is in the order of $50 - 100Hz$ when these change fast.

$$\hat{x}_k = f(T_k, u_{1,k}, ..., u_{n,k}) \tag{3}$$

Electronic control units (ECUs) are extensively programmed using fixed calibrated look-up tables ($T_k = T_{k-1} = ... = T_1$) where ageing effects, manufacturing discrepancies and in general, effects that produce variations in the function are not taken into account. Main reason for this, is the time and memory resources that adaptive algorithms consume, without forgetting the stability of the programming against perturbations that can affect the model. Furthermore, tables are usually calibrated off-line with specific test rigs that consume an important amount of human and material resources.

T_k in (4) defines the table T in the instant k as a function of the previous elements of the table, the inputs for the function $u_{n,k-1}$ and the learning algorithm F, according to the available reference x_{k-1}.

$$T_k = F(T_{k-1}, u_{1,k-1}, ..., u_{n,k-1}, x_{k-1}) \tag{4}$$

With this scenery, the solution of the problem is designing a learning algorithm F, which from certain reference about the system x_{k-1}, updates the matrix T_k for inferring the estimation $\hat{x}_k$. Uncertainties in x_k can be taken into account as some noise ζ_k around the actual value $x_{r,k}$ that in general is not known:

$$x_k = x_{r,k} + \zeta_k \tag{5}$$

The ability of the learning method for coping with errors in the reference is studied in section 5. Without any lack of generality and for the comprehension of the reader, the matrix T in instant k can be defined for a 2D case as follows:

$$T_k = \begin{bmatrix} \theta_{1,1,k} & \theta_{1,2,k} & ... & \theta_{1,d_2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{d_1,1,k} & \theta_{d_1,2,k} & ... & \theta_{d_1,d_2,k} \end{bmatrix} \tag{6}$$

being $\prod_{i=1}^{n} d_i$ the number of elements of the table and d_i the number of nodes in each dimension.

For giving numbers of the method performance (see section 5), T_r matrix is defined as the actual reference of the matrix, where T_k must converge to T_r. $\theta_{r,ij}$ are the elements of T_r, in the same way of T_k elements in (6). During each iteration, an scalar error e_k in the estimation is used for updating T_k:

$$e_k = x_k - \hat{x}_{k-\tau} \tag{7}$$

Although different error metrics can be defined, the current paper defines an scalar quadratic error percentage for each iteration for seeing the convergence rate of the method, and is defined in (8) for the 2D case. If e_k tends to zero, convergence of the matrix can be ensured.

$$eq_k = 100 \sum_{i=1}^{d_1} \sum_{j=1}^{d_2} (\theta_{ij} - \theta_{r,ij})^2 / eq_u \tag{8}$$

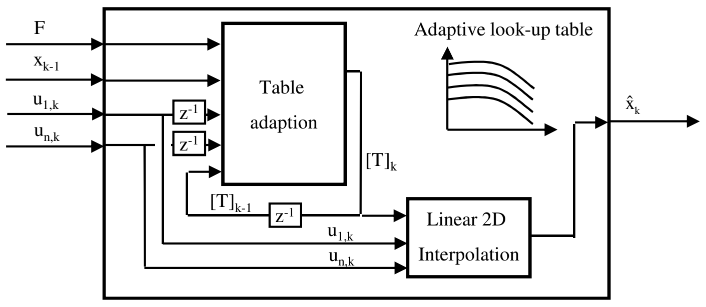

Figure 2: Adaptive scheme including adaptation and interpolation blocks.

where eq_u normalises the error for getting $eq_1 = 1$.

To sum up, the system must keep the following properties:

- Convergence of the estimated matrix T_k to the actual value when T_r is the actual value.

$$\lim_{k \to +\infty} T_k = T_r \tag{9}$$

- Method must ensure noise rejection in the updated elements when ζ_k appears in x_k and must cope with changes in T_r, ensuring convergence as in (9).
- Low memory use and reduced computational resources of the adaptation algorithm, for allowing its use in commercial ECUs, which is one of the main contributions of the presented method.

And to solve the problem, two main parts must be addressed:

- Interpolation problem; which defines the table output $\hat{x}_k$.
- Adaptation problem; which defines the learning algorithm F that updates the table.

Both issues are shown in figure 2 and will be discussed in section 3 and 4 respectively.

3. Look-up table interpolation

$\hat{x}_k$ is obtained by means of linear interpolation of the table elements that are active in every iteration (see (3)), 2^n in total for the nD problem. These active nodes coincide with the observable ones for every iteration. Linear interpolation, depending on the node distribution and system to be modelled, is sufficient for getting enough accuracy with a reasonable calculation burden. Anyway, other principles could be used, such as the nearest interpolation or B-splines. The defined grid must have enough nodes $(d_1, ..., d_n)$ and a sufficient density for tracking x_k slope variations [13].

For a 1D case, linear interpolation can be mathematically expressed as:

$$\hat{x}_k = [(1-\eta) \quad \eta]\,[\theta_i \quad \theta_{i+1}]' \tag{10}$$

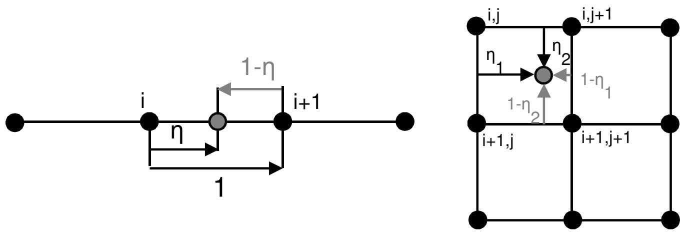

Figure 3: Definition of distance axis for membership functions.

where variable η is defined according to left plot in figure 3 that considers the normalised distance of the new measurement point u_k to the abscissa of the nodes i and $i + 1$.

For the 2D case and linear interpolation, η_1 and η_2 variables are defined in the right plot of figure 3 according to the 2 degree of freedom of the table:

$$\hat{x}_k = \left[(1-\eta_1)(1-\eta_2) \quad \eta_1(1-\eta_2) \quad (1-\eta_1)\eta_2 \quad \eta_1\eta_2\right]\left[\theta_{i,j} \quad \theta_{i+1,j} \quad \theta_{i,j+1} \quad \theta_{i+1,j+1}\right]' \tag{11}$$

4. Look-up table adaptation

The design of the learning algorithm F is the key aspect of the problem and motivation of the paper. Different kinds of adaptation principles for look-up tables can be found in the literature, and some, specially those based on Kalman filter and proportional correction, are commented in the following:

- [4] and [14] propose an extended Kalman filter (EKF) for updating table elements where they are treated as states to be observed and updated using some system feedback. The error covariance matrix P_k estimates the observation error adjusting a Kalman gain K_k in every iteration. P_k elements related with locally unobservable elements have a linear growth of noise (defined by the user), diminishing and approaching to zero when such elements become observables, coping with own table ageing. A bigger covariance leads to a higher K_k. The first problem of this system is that although unobservable elements do not affect the updating in the given iteration, EKF must manipulate all elements of P_k. Global stability of the method relies on the observability of the states and although during each iteration only 4 elements (in a 2D example) are locally observables and the rest of them unobservables, the calculation involves all table areas. The second problem relies on the fact that system matrices vary with time making impossible to derive an steady-state Kalman filter [15, 1]. This forces the use of a huge memory and computational resources for inferring K_k in every instant. If the 2D matrix T has 100 elements (10×10), and 1 state variable is reserved for the system dynamics, then the total size of the state vector is 101×1 (100 nodes plus 1 state). Hence P_k used in the EKF is 101×101. This matrix needs to be updated and calculated recursively and, since some of their elements can present an unbounded growth (due to unobservability issues), it supposes severe computational instability challenges. Furthermore, this method requires the storing of at least two dynamic memories for P_k and K_k (101×1) extra with respect to the steady-state Kalman filter. Anyway, this idea is used as inspiration for the proposed algorithm.

- [16] treats the problem of table updating as a reverse interpolation problem but considering that this is an ill conditioned one. A proportional weighting is used for updating all the elements that were involved in the previous interpolation calculation. The author cites it literally as *multiple nodes proportional distribution*. This weighting is calculated only from inputs and previous output. In that way, the method does not take into account noise in the measurement, and is not optimal by definition, as Kalman filter is. Main advantage of the method is that during every iteration, only table values related with observable elements are updated. This makes the system updating fairly simple, where only a simple expression is solved for defining the weighting gain (equivalent to K_k in our method). The method proposed in the current paper remains this property.

- There exist other possibilities in the literature, such as least-squares identification [13] or non-uniform rational b-spline interpolation (NURBS) [17]. The first is fairly simple and requires that table grid and data is well distributed for avoiding robustness problems (as the authors literally claim); the second fits well with complicated profiles but is more indicated for mapping complicated functions than for look-up tables on the automotive field, because of the high computational resources needed.

4.1. A learning algorithm for look-up table adaptation

A novel algorithm is presented combining the accuracy and optimal solution presented by [4] with the fast and simple one presented by [16]. Basic idea is deriving membership functions similar to the ones used in [16], but from a Kalman filter representation as in [4]. In that way, an steady-state representation of the Kalman filter is achieved under certain simplifications allowing off-line calculation of Kalman gain K_k. This permits getting membership functions that depend uniquely on the relative active position. Hence the resulting algorithm keeps some of the properties of the Kalman filter but with a lower computational effort. This also avoids numerical problems associated to the system unobservability due the unbounded growth of P_k. Anyway, it remains evident that complete table adaptation requires enough excitation in all areas of the matrix.

For that, the problem is stated as follows:

$$\vec{\theta}_k = A\vec{\theta}_{k-1} + Q \tag{12}$$
$$x_k = D_k\vec{\theta}_k + R \tag{13}$$

For the 2D case, this system is applied uniquely to the table area that is active during every iteration. This is similar to consider a system of dimension 4 that works only with observable elements. System matrices and state vector $\vec{\theta}$, built from expanding active elements of T_k, are shown in (14). This allows solving a reduced system every time.

$$A = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} ; D_k = \begin{bmatrix} (1-\eta_1)(1-\eta_2) \\ \eta_1(1-\eta_2) \\ (1-\eta_1)\eta_2 \\ \eta_1\eta_2 \end{bmatrix}' ; \vec{\theta} = \begin{bmatrix} \theta_{i,j} \\ \theta_{i+1,j} \\ \theta_{i,j+1} \\ \theta_{i+1,j+1} \end{bmatrix} \tag{14}$$

Two noise matrices representing the instant variation of the system and the measurement noise as white noises with a certain covariance are considered:

$$Q = \begin{bmatrix} q(\sigma_\theta^2) & & & \\ & q(\sigma_\theta^2) & & \\ & & q(\sigma_\theta^2) & \\ & & & q(\sigma_\theta^2) \end{bmatrix} \; ; \; R = r(\sigma_x^2) \tag{15}$$

Optimal observation of θ_k is ensured if the reference x_k exists and then it can be obtained through the Kalman filter formulation:

$$\vec{\theta}_k = \vec{\theta}_{k|k-1} + K_k(x_k - D\vec{\theta}_{k|k-1})) \tag{16}$$

where Kalman gain correction K_k is recursively defined through:

$$\begin{aligned} P_{k|k-1} &= AP_{k-1|k-1}A' + Q \\ K_k &= P_{k|k-1}D_k' \left(D_k P_{k|k-1} D_k' + R\right)^{-1} \\ P_{k|k} &= (I - K_k D_k) P_{k|k-1} \end{aligned} \tag{17}$$

where first and second equation define the prediction phase and third equation defines the updating phase for Kalman gain correction; being $P_{k|k-1}$ the predicted estimate covariance, K_k the kalman gain and $P_{k|k}$ the updated covariance for observing the state vector $\vec{\theta}_k$ of the system. This formulation, similar to the one in [14], would allow to compute the optimal estimate of the table nodes (θ) but with a high computational cost.

Now, if the problem is considered stationary (the same input is repeated once and again), then $D_k = D$. In addition to this, if a noise ratio $\sigma_x^2/\sigma_\theta^2$ is given, K_k in (17) converges after a given number of iterations. That occurs although P does not converge and can be ill conditioned (see Appendix A for a wider explanation). Then, a stationary equivalent is obtained and an analytical expression can be derived, resulting:

$$k_{i,j} = f(\eta_1, \eta_2, \sigma_x^2/\sigma_\theta^2) \tag{18}$$

where weighting functions $k_{i,j}$ are shown in (A.7) and only depend on the current engine operating point defined by η_1 and η_2 (derived from u_1 and u_2) in the 2D case.

Calculating only one of the $k_{i,j}$ functions is enough, as far as the rest of them are symmetrical as shown in figures 4 and 5 for 1 and 2D cases (this is forced because the effect of variable P_k is being neglected). The user can evaluate (A.7) for the 2D case in every iteration for a variable noise or can map the functions off-line and fasten even more the calculation, which is the option suggested by the authors. Table elements are then corrected during every iteration:

$$\theta_{i,j,k} = \theta_{i,j,k-1} + k_{i,j}(\eta_1, \eta_2, \sigma_x^2/\sigma_\theta^2)e_k \tag{19}$$

Figures 4 and 5 also show how when operating point is close to one node, the correction is generally bigger than when the node is further. Furthermore, when $\sigma_x^2/\sigma_\theta^2$ relationship increases is equivalent to increase the confidence in x_k, and then $k_{i,j}$ tends to be higher. Non-observable elements in every iteration will not present any correction, which is equivalent to say that $k_{i,j}$ function for these is zero.

To sum up, $k_{i,j}$ functions for updating table elements have been derived from an standard Kalman filter under certain hypothesis, and these are calculated and stored off-line. These hypothesis neglect the effect of the variation of D_k along time and P_k variation. However, this allows keeping a very low calculation burden; P is not calculated nor updated.

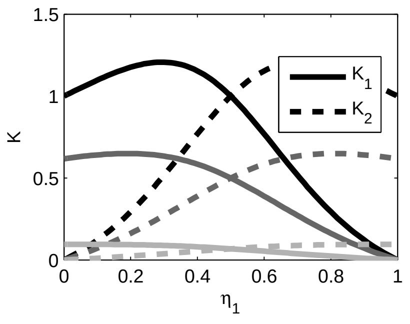

Figure 4: Membership functions for Kalman gain correction for the 1D case for $\sigma_x^2/\sigma_\theta^2 = 0$ (black), $\sigma_x^2/\sigma_\theta^2 = 0.01$ (medium grey) and $\sigma_x^2/\sigma_\theta^2 = 0.1$ (light grey).

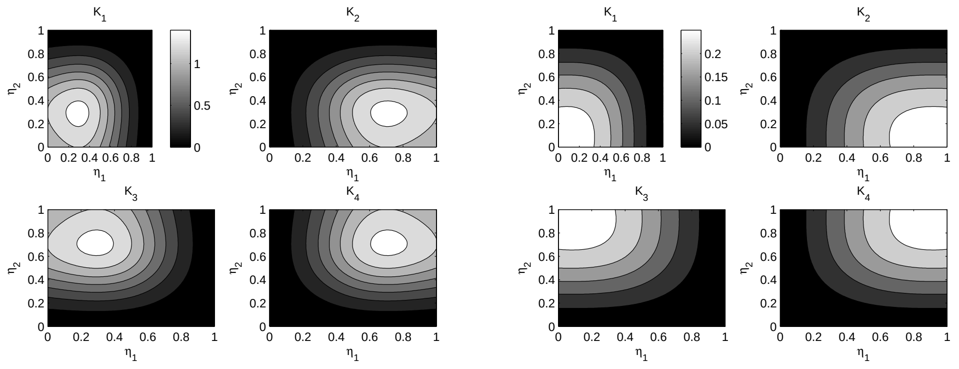

Figure 5: Membership functions for Kalman gain correction for the 2D case for $\sigma_x^2/\sigma_\theta^2 = 0$ (left) and $\sigma_x^2/\sigma_\theta^2 = 0.1$ (right).

This method itself is capable of tracking engine system ageing but does not estimate observation error in every iteration. If the user requires this, a simple count vector could be added for increasing or decreasing the noise ratio in (18), and then increasing or decreasing $k_{i,j}$; and for this case, (A.4) analytical expression cannot be mapped off-line. Anyway, this vector would require an extra dynamic memory of 100×1 and a simple evaluation of (A.4) for getting $k_{i,j}$, which is still much less than for the standard Kalman filter. This can be used during first iterations for accelerating convergence in areas that have not been updated before.

5. Simulation of the learning algorithm

For simulation purposes, a Pentium Dual-Core PC computer with a mathematical program is used. An stochastic distribution of 1000 points is prepared to show the adaptation performance of the table using the proposed method. Table elements are initially zero. Two analytical functions for 1D and 2D case are designed for checking the performance of the method. They are represented in figure 6,including updated T matrix, which is calculated in next subsections. Convergence and speed of the method are numerically studied using error metric eq_k defined in (8).

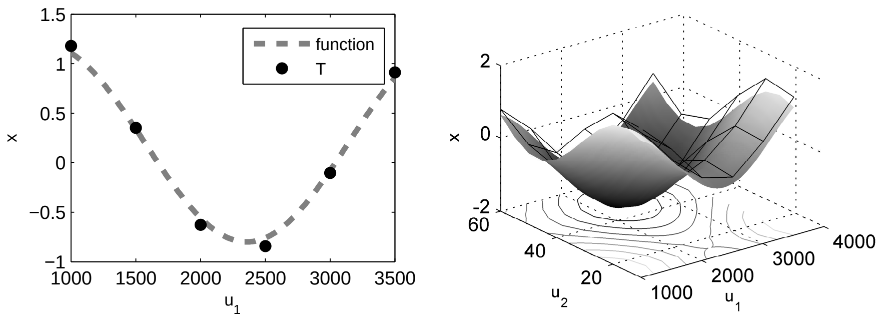

Figure 6: Functions in grey color against fitted one by the proposed method in 1D (left) and 2D case (right).

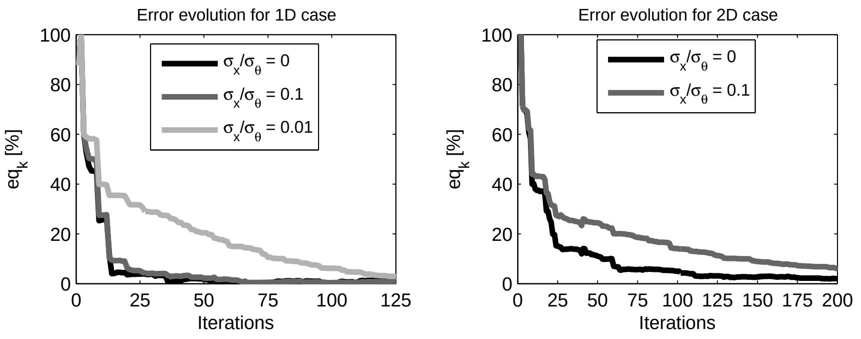

Figure 7: Evolution of the quadratic error in 1D case and 2D case for different noises.

5.1. *Updating with a reliable x_k*

Reference is supposed reliable and well-known, and then (5) leads to $x_k = x_r$. Simulation results are shown in figure 7. eq_k tends to zero for 1D and 2D cases in a short number of iterations, which shows the convergence of the algorithm. In this figure, different noise sets are checked, and best results are with lower σ_x/σ_θ. This is a logical conclusion as far as x_k is now the real reference and it is considered well-known. Then, $\sigma_x^2 = 0$ makes the fastest correction. This case is ideal and demonstrates the feasibility of the method, but some perturbations must be taken into account to see how the adaptation behaves.

5.2. *Bias in the identification*

Automotive engines operate under highly variable conditions, which combined with modelling errors, lead models to suffer from bias. Furthermore, this bias varies with time, operating point and others, provoking drift. This makes that table output $\hat{x}_k$ suffers some drift if table elements are fixed. For testing this, a constant bias to all elements of the function in the 2D case is applied just after first 200 iterations of the method, provoking an step on the error and a sharp variation of x_k that anyway is considered known. This is by far the worst drift situation possible. The result is shown in figure 8, showing clearly again how the noise set with $\sigma_x = 0$ is the optimal one again.

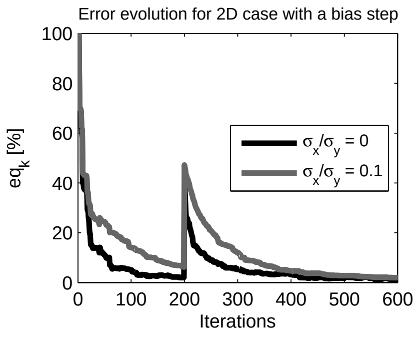

Figure 8: Evolution of the quadratic error in 2D case with a step of 0.6 in function in iteration number 200.

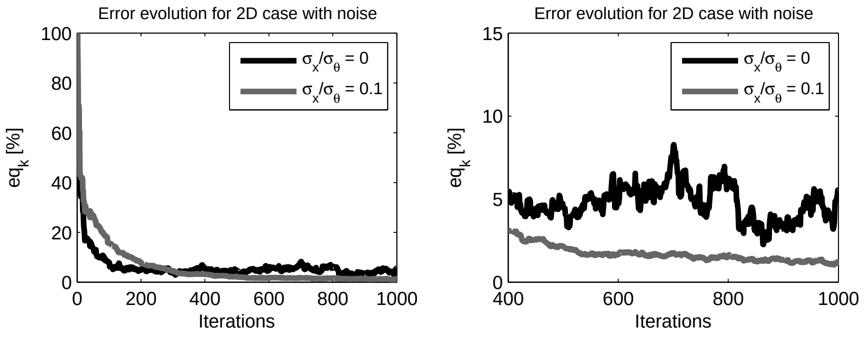

Figure 9: Evolution of the quadratic error in 2D case applying a uniform distribution of error with an amplitude of 0.4 in the reference.

5.3. Noise rejection

In previous calculations, x_k has been considered as a reliable reference, and all cases suggest to select the biggest correction possible, where σ_x tends to zero. Noise in real examples complicates the identification. For this, an uniform noise distribution ζ_k with zero mean and amplitude of 0.2 is applied to the reference for seeing the influence in the error as in (5). Figure 9 shows the results. Although case with $\sigma_x = 0$ is faster in the first iterations until 300, then the grey line has a higher level of convergence than the black one, with a slower but more reliable updating. Right plot of this figure shows the noise transmission from the input to the output of the table, showing how in the black line, this is much higher. That is because higher corrections lead to higher oscillations, compromising the stability and robustness of the method. It is evident that user must tune noises taking into account these considerations.

To sum up, all cases have shown not only the correction speed of the method for different noise sets but convergence, noise rejection and bias cancellation properties, where the final tuning of the algorithm depends on the function or system to be analysed and the reliability of the reference. Anyway, dynamical effects will affect the final tuning of the method, and this makes necessary to check the method on engine.

6. Method performance on real engine

Sections 2 to 5 have been devoted to the Learning block of figure 1. This section recovers the complete system of the figure 1 for checking the validity of the method under real engine conditions. For this, experimental data is obtained from a 2.2 common rail turbocharged diesel engine coupled to a variable frequency eddy current dynamometer that allowed carrying out dynamical tests (more information about test set-up can be found in [9]). External bypass of the ECU is used for varying injection parameters. A real time hardware system was connected via CAN with a rapid prototyping system connected via ETK with the engine. A commercial ZrO_2 [18] sensor installed downstream of the turbine was used for getting NO_x and λ^{-1} measurements.

An sportive driving profile in a mountain road (already used in [19, 20]) is selected as far as it presents fast transients of both n and m_f, causing sharp variations of exhaust variables, such as λ^{-1} and NO_x. This cycle is a good scenery for checking the method under severe dynamical conditions. Objective is mapping NO_x and λ^{-1}, using a 2D look-up table being $u_1 = n$ and $u_2 = m_f$. Initial values for table are zeros $T_1 = [0]$. For n, an uniform grid between 500 and 5000 with values every 500 rpm is chosen, whereas for m_f an uniform grid between 0 and 80 mg/str every 5 mg/str is selected; covering all possible range of values (see [19]). Then, (3) can be set out as:

$$\hat{x}_k = f(T_k, n_k, m_{f,k}) \tag{20}$$

where n_k and $m_{f,k}$ are both inferred by the engine inductive sensor and the ECU estimation respectively. Although both inputs can be conveniently delayed or filtered if necessary, for the current example, no dynamical treatment is made, as far as values are considered sufficiently fast, at least comparing with NO_x and λ^{-1} variables.

An standard Kalman filter for the Kalman filter block in figure 1 is built [1] for observing directly the sensor: including a parameter as a first order filter to model dynamics; x_k the observed state that will update table and $\hat{x}_k$ the table output; v_k and w_k white noises with a selected constant variance of 1 for both:

$$\begin{aligned} x_k &= (1-a)x_{k-1} + a\hat{x}_k + w_k \\ y_k &= x_k + v_k \end{aligned} \tag{21}$$

Learning algorithm presented in section 5 is applied jointly with system (21), with $\sigma_x^2/\sigma_\theta^2 = 1$ and choosing $a = 1$ for the example just for checking the method performance. If the sensor dynamics are known, a Kalman filter considering the system (21) can be used for inferring the actual value of the variable at each time iteration. This value will be then the input x_k for the table update process, as defined in (4). However, in this case, the engine output variables have been directly mapped without considering any dynamics, as far as measured quantity y_k has been directly considered as the reference. Even with this simplification the table is able to behave quite well according to Figures 10 and 11.

Figure 10 shows the results for λ^{-1}. Left plot shows the adaptation of the table output: during first iterations, the table is learning, and finally gets sensor output; right plot shows the simulation of the learnt model (using last updated table in the whole cycle) in a segment of the cycle, clearly showing the good agreement between sensor and model lines. Figure 11 shows same plots for the NO_x output of the sensor, again with a good agreement. In both cases, engine output has been mapped, being the method also a powerful tool for characterising engine behaviour. Bias that appears in figures is not a problem, as far as table is being updated during the whole cycle

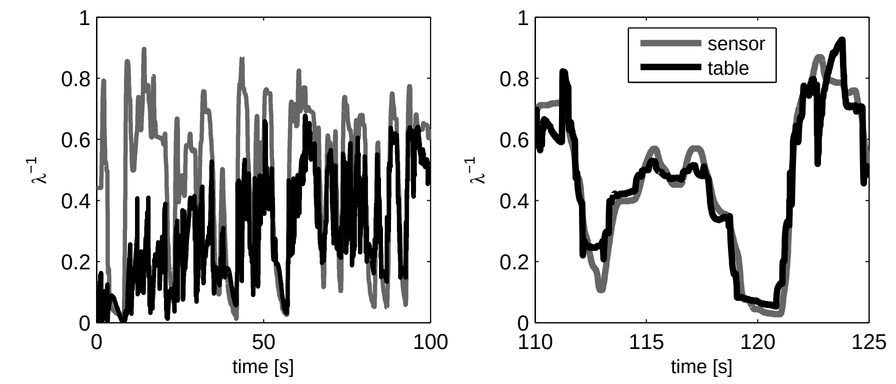

Figure 10: Left: Table adaptation; grey line is sensor y_k, while black line is $\hat{x}_k$. Right: λ^{-1} identification from table in engine tests.

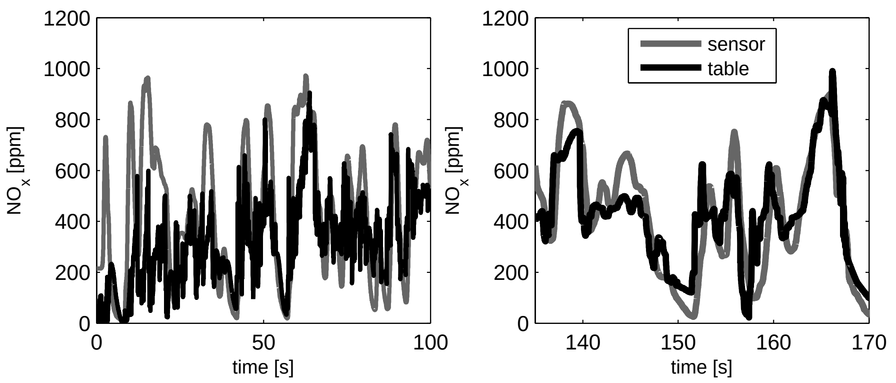

Figure 11: Left: Table adaptation; grey line is sensor y_k, while black line is $\hat{x}_k$. Right: NO_x identification from table in engine tests.

coping with these variations. Right plot shows again the application of the last updated table to a segment of the cycle for the NO_x case.

7. Conclusion

The current paper proposes a novel learning algorithm for updating look-up tables using pre-calculated membership functions based on an off-line use of Kalman filter gains. Table elements are updated in a fast way ensuring the convergence, noise rejection and bias cancellation of the system. The method has been checked under different conditions in simulation, and under hard dynamic conditions in real tests, where the shown example is mapping λ^{-1} and NO_x outputs of a ZrO_2 sensor.

The only requirement for the full table adaptation is running the engine in all the matrix areas. Stability and convergence is ensured if an enough level of excitation similar to other adaptation methods is presented. The memory saving and computational reduction that the method offers, against other based on Kalman filter, make this algorithm suitable for being implemented on commercial ECUs and open the possibility of using the method for online control purposes.

References

[1] C. Guardiola, B. Pla, D. Blanco-Rodriguez, A. Mazer, and O. Hayat, "Bias compensation of λ estimator for internal combustion engines," *International Journal of Engine Research*, submitted 2011.

[2] A. Schilling, A. Amstutz, C. Onder, and L. Guzzella, "A real-time model for the prediction of the NO_x emissions in DI diesel engines," in *Proceedings of the 2006 IEEE International Conference on Control Applications*, Munich, Germany, October 4-6, 2006, 2006.

[3] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 35-45, 1960.

[4] E. Höckerdal, E. Frisk, and L. Eriksson, "EKF-based adaptation of look-up tables with an air mass-flow sensor application," *Control Engineering Practice*, vol. 19, pp. 442–453, 2011.

[5] J. M. Desantes, J. M. Luján, C. Guardiola, and D. Blanco-Rodriguez, "Development of NO_x fast estimate using NO_x sensors," in *EAEC 13 Congress*, S. I. 978-84-615-1794-7, Ed., June 2011.

[6] L. del Re and D. Alberer, "Fast oxygen based transient diesel engine operation," *SAE paper 01-0622-2009*, 2009.

[7] E. Grünbacher, P. Kefer, and L. del Re, "Estimation of the mean value engine torque using an extended kalman filter," *SAE paper 2005-01-0063*, 2005.

[8] E. Höckerdal, "Model error compensation in ode and dae estimators with automotive engine applications," Ph.D. dissertation, Linköping University Institute of Technology, 2011.

[9] J. Galindo, J. Serrano, C. Guardiola, D. Blanco-Rodriguez, and I. Cuadrado, "An on-engine method for dynamic characterisation of NO_x concentration sensors," *Experimental Thermal and Fluid Science*, vol. 35, p. 470476, 2011.

[10] S. Regitz and N. Collings, "Fast response air-to-fuel ratio measurements using a novel device based on a wide band lambda sensor," *Meas. Sci and Technol.*, vol. 19, no. 075201, 2008.

[11] A. Schilling, A. Amstutz, and L. Guzzella, "Model-based detection and isolation of faults due to ageing in the air and fuel paths of common-rail direct injection diesel engines equipped with a λ and a nitrogen oxides sensor," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 222, pp. 101–117, 2008.

[12] T. S. Chen and R. Z. You, "A novel fault tolerant sensor system for sensor drift compensation," *Sensors and Actuators A: Physical*, vol. 147, pp. 623–632, 2008.

[13] J. Peyton Jones and K. Muske, "Identification and adaptation of linear look-up table parameters using an efficient recursive least-squares technique," *ISA Transactions*, vol. 48, pp. 476–483, 2009.

[14] E. Höckerdal, E. Frisk, and L. Eriksson, "Model based engine map adaptation using EKF," in *6th IFAC Symposium on Advances in Automotive Control*, Munich, Germany, 2010.

[15] F. Payri, C. Guardiola, D. Blanco-Rodriguez, A. Mazer, and A. Cornette, "Methodology for design and calibration of a drift compensation method for fuel-to-air ratio estimation," *SAE paper 2012-01-0717*, 2012.

[16] G. Wu, "A table update method for adaptive knock control," *SAE paper 2006-01-0607*, 2006.

[17] H. Shen, J. Fu, and Y. Fan, "A new adaptive interpolation scheme of NURBS based on axis dynamics," *Int J. Adv. Manuf. Technol.*, vol. 56, p. 215 221, 2011.

[18] N. Kato, K. Nakagaki, and N. Ina, "Thick film ZrO_2 NO_x sensor," *SAE paper 960334*, 1996.

[19] J. Galindo, H. Climent, C. Guardiola, A. Tiseira, and J. Portalier, "Assessment of a sequentially turbocharged diesel engine on real-life driving cycles," *Int. J. Vehicle Design*, vol. 49, no. Nos. 1/2/3, 2009.

[20] E. Lughofer, V. Macian, C. Guardiola, and E. P. Klement, "Identifying static and dynamic prediction models for NO_x emissions with evolving fuzzy systems," *Applied Soft Computing*, vol. 11, pp. 2487–2500, 2011.

Appendix A. Steady-state derivation of kalman gain for table updating

Given the system (12)-(17) and a noise set $\sigma_x^2/\sigma_\theta^2$, the value of $P_{k|k}$ does not converge under normal operation, making that value K_k varies. This is because D_k matrix is not constant and depends on the engine operating conditions.

But if the operating point condition is constant, system runs in a constant point ($D_k = D$) with $\sigma_x^2/\sigma_\theta^2$, then it is possible to derive a K_k constant value, although $P_{k|k}$ value does not converge. This fact can be observed in simulations and then it can be set out:

$$\lim_{k\to+\infty} K_{k-1} = \lim_{k\to+\infty} K_k \tag{A.1}$$

Taking into account the symmetric property of the problem, the simplest 1D table system is considered simplifying matrices (14)-(15) to:

$$A = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \; ; \; D_k = \begin{bmatrix} (1-\eta) \\ \eta \end{bmatrix}' \; ; \; \vec{\theta} = \begin{bmatrix} \theta_i \\ \theta_{i+1} \end{bmatrix} \tag{A.2}$$

$$Q = \begin{bmatrix} q(\sigma_\theta^2) & \\ & q(\sigma_\theta^2) \end{bmatrix} \; ; \; R = r(\sigma_x^2) \tag{A.3}$$

Using (17), matrices (A.2) and (A.3) and imposing (A.1) condition, the value of Kalman gain k_i related with element θ_i can be derived analytically with a certain effort, where η relative position is defined as in left plot of figure 3:

$$k_i(\eta, \sigma_x^2/\sigma_\theta^2) = \frac{0.5(1-\eta)(1+s)}{0.5(1+s)(1-2\eta+2\eta^2) + \sigma_x^2/\sigma_\theta^2} \tag{A.4}$$

where

$$s = \sqrt{1 + \frac{4}{(1-2\eta+2\eta^2)} \frac{\sigma_x^2}{\sigma_\theta^2}} \tag{A.5}$$

showing clearly the dependance of k_i value only with noise trade-off $\sigma_x^2/\sigma_\theta^2$ and relative position η of the current operating point.

This k_i function is used for calculating gain correction for node i, when observable region is between i and $i+1$ in the 1D case. Using symmetric property:

$$k_{i+1} = k_i(1-\eta, \sigma_x^2/\sigma_\theta^2) \tag{A.6}$$

For the 2D case, calculations of gains is direct from (A.5). Inputs η_1 and η_2 are defined in right plot of 3:

$$\begin{array}{rcl} k_{i,j} & = & k_i(\eta_1, \sigma_x^2/\sigma_\theta^2) \cdot k_i(\eta_2, \sigma_x^2/\sigma_\theta^2) \\ k_{i,j+1} & = & k_i(\eta_1, \sigma_x^2/\sigma_\theta^2) \cdot k_i(1-\eta_2, \sigma_x^2/\sigma_\theta^2) \\ k_{i+1,j} & = & k_i(1-\eta_1, \sigma_x^2/\sigma_\theta^2) \cdot k_i(\eta_2, \sigma_x^2/\sigma_\theta^2) \\ k_{i+1,j+1} & = & k_i(1-\eta_1, \sigma_x^2/\sigma_\theta^2) \cdot k_i(1-\eta_2, \sigma_x^2/\sigma_\theta^2) \end{array} \tag{A.7}$$

This can be generalised for the nD case.