Real-Time Predictive Control for SI Engines Using Linear Parameter-Varying Models

Pawel Majecki*, Gerrit M. van der Molen*, Michael J. Grimble*, Ibrahim Haskara, Yiran Hu**, Chen-Fang Chang****

**Industrial Systems and Control, Glasgow, UK (e-mail: pawel@isc-ltd.com)*
***General Motors R&D, Propulsion Systems Research Lab, Warren, MI 48090 (e-mail: ibrahim.haskara@gm.com)*

Abstract: As a response to the ever more stringent emission standards, automotive engines have become more complex with more actuators. The traditional approach of using many single-input single output controllers has become more difficult to design, due to complex system interactions and constraints. Model predictive control offers an attractive solution to this problem because of its ability to handle multi-input multi-output systems with constraints on inputs and outputs. The application of model based predictive control to automotive engines is explored below and a multivariable engine torque and air-fuel ratio controller is described using a quasi-LPV model predictive control methodology. Compared with the traditional approach of using SISO controllers to control air fuel ratio and torque separately, an advantage is that the interactions between the air and fuel paths are handled explicitly. Furthermore, the quasi-LPV model-based approach is capable of capturing the model nonlinearities within a tractable linear structure, and it has the potential of handling hard actuator constraints. The control design approach was applied to a 2010 Chevy Equinox with a 2.4L gasoline engine and simulation results are presented. Since computational complexity has been the main limiting factor for fast real time applications of MPC, we present various simplifications to reduce computational requirements. A benchmark comparison of estimated computational speed is included.

Keywords: SI engines, model predictive control, nonlinear control, LPV models

I. INTRODUCTION

In order to meet more stringent future emissions standards as well as the desire for better engine performance, engine control systems have become more complex. Dual independent cam phasing, that was rare, is now widely used. Each actuator needs to be controlled to a setpoint, such that the overall engine performance, which is assessed based on the opposing objectives of drivability, emissions and fuel economy, is optimized. Coordination of the various engine actuators has always been difficult. Traditionally, each actuator was controlled to its own setpoint based on engine operating condition. A large effort is required to calibrate the set-points offline so that the real-time control of the actuators does not result in undesirable behaviour. Moreover, ad-hoc patches are also often needed so that good performance is achieved under transient conditions. To avoid having too many patches, setpoints are selected to be conservative, which makes the engine controls difficult to calibrate.

The difficulty with the engine control problem is that the system is nonlinear, multi-input multi-output (MIMO), and has many actuator and state/output constraints. The shortcomings of the traditional single-input single-output (SISO) design philosophy are therefore becoming more evident. Model based control design can potentially provide a solution, which is truly multivariable, more flexible, and easier to upgrade when engine configurations change. Of the many advanced control design methodologies available, model predictive control (MPC) is a popular control strategy because of its ability to tackle multivariable processes, handle constraints, deal with long time-delays, and utilize future reference knowledge. The main disadvantage of MPC controllers is that they can be computationally intensive due to the online optimization process used to compute the current control. Quadratic programming (QP) usually provides the most efficient optimization algorithm, but this in principle only applies to linear models. Early work on MPC focused on linear time-invariant (LTI) systems. Popular predictive algorithms are Dynamic Matrix Control (Cutler and Raemaker, 1979), Generalized Predictive Control (GPC), due to Clarke et al. (1989), and those due to Richalet (1978). For useful review papers on MPC see Bemporad and Morari (2004), Qin and Badgwell (2003), and the references therein.

Unfortunately, few practical systems can be modelled accurately by a linear time-invariant system, across the full operating range. Moreover, there is no generally accepted process for solving an MPC problem involving a general nonlinear (NL) model. Nonlinear MPC (NMPC) has proven successful in some applications based upon simple scheduling and anti-windup methods. This mainly applies to the chemical and process industries, where sampling times are usually of the order of a few seconds or minutes, and the operating points of large complex systems can be moved across operating regions relatively slowly. However, servo-systems and combustion engines have highly nonlinear

behaviour and require sampling times of a few milliseconds. This poses a challenging problem requiring tailored NL predictive control methods. With advances in computing power, it has been possible to apply MPC in high bandwidth control applications, including automotive systems. Nonlinear MPC for automotive engines was considered by Herceg et. al. (2006) and Vermillion et. al. (2010).

Another area of active research is control design based on Linear Parameter Varying (LPV) models. This class of models provides can approximate nonlinear systems whose nonlinearity enters via parametric changes. The application of LPV models to MPC provides a great middle ground between traditional LTI model-based MPC and the less-attainable full nonlinear MPC. The QP optimization methods can be used, because LPV models are quasi-linear, providing an efficient solution method. Due to this improved modelling approach, MPC may now be used on some applications, where it was previously unsuitable (see for example Casavola et al 2002, 2003; Chisci 2003; Besselmann 2012; Li 2010, Duan 2013). The main contribution in the following lies in the formulation of the Nonlinear Generalized Predictive Control (NGPC) problem in a useful form for the engine control application.

In the following, an engine control that uses an LPV model-based MPC solution is proposed. The problem of MIMO torque and air-fuel-ratio (AFR) control is considered. These are two of the most critical variables in the engine control system, which have a direct influence on drivability, emissions and fuel economy. The desired engine torque depends upon the driver's pedal position. The intake throttle is the main actuator to control the intake manifold pressure and thus the inducted air charge. This in turn controls the engine torque. In addition to torque delivery, engine control systems also need to address other objectives such as improved fuel economy, reduced engine-out and tailpipe emissions. For a SI engine, AFR must be regulated, to achieve good emissions through torque transients.

II. ENGINE CONTROL PROBLEM

The problem of engine torque and air-to-fuel ratio control is considered in this paper. The engine is the 2.4L engine used on a 2010 Chevy Equinox. Amongst the main characteristics of this particular engine are dual independent cam phasers and a direct fuel injection system. The general block diagram of the engine torque and AFR production process is shown in Fig. 1. The throttle is used to maintain intake manifold air pressure. As cylinders go through an induction cycle, air is drawn into the cylinders through the intake valves. Cam phasing changes the intake/exhaust valve opening and closing timing, which varies the amount of trapped residual and the fresh charge in the cylinder. The FPW command, applied to the injectors determines the amount of fuel injected into the cylinders. The injected fuel is mixed with the air charge and is then ignited during the compression cycle. Spark timing controls the ignition time, and this determines the combustion phasing and the final torque generated during that combustion event.

The desired engine torque is determined by the accelerator pedal position interpreting the driver request. The desired AFR is a function of the type of fuel used, as well as operating conditions. For stoichiometric SI engine, the desired AFR is the stoichiometric AFR for the recommended fuel. In some cases, a richer or leaner AFR may be required for other reasons (such as piston protection). Note that there are non-stoichiometric operating SI engines, where the desired AFR will vary significantly depending on the load.

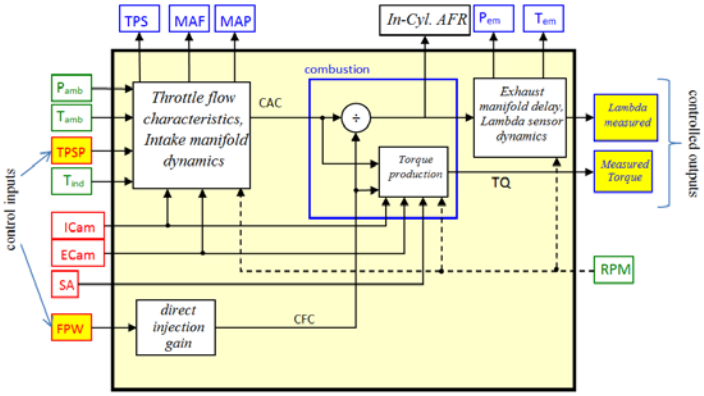

Fig. 1: **Block diagram of the SI Engine**

There are primarily five actuators that control the engine torque and AFR, namely throttle position, fuel pulse width (FPW), intake and exhaust cam phaser angles, and the spark advance. With the direct fuel injection system, there is a simple relationship between the FPW and the cylinder fuel charge. The throttle position and CFC are the two manipulated inputs, resulting in a square system. The ICAM and ECAM phaser position set-points, and the spark advance are obtained from operating-point dependent tables optimized to produce the best torque (MBT), and a desired trade-off between fuel economy and drivability. In this work these are treated as known disturbance inputs. The set of measurements available include Manifold Absolute Pressure (MAP) in the intake manifold, Mass Air Flow (MAF), air charge temperature (MAT), throttle position (TPS), exhaust AFR, cam phaser positions, engine speed, ambient pressure and ambient temperature. The sensors can be used for feedback control and to update model parameters.

III. ENGINE LPV MODEL

One of the objectives was to derive an LPV model suitable for QP-based implicit MPC that also captures the nonlinear and operating-point dependent nature of the engine dynamics. The model needs to relate the control inputs (i.e. fuel and TPS) to system outputs (lambda and torque) with the state-space model matrices A, B, C, and D being dependent on measurable parameters. Two modelling techniques were considered. The first was to start with a physics-based model, based on governing dynamics of engine air path, and then transform the resulting NL model to an LPV format. The second was the direct identification of the model from data. The first method was selected where a physics based model was rewritten in a quasi-LPV form. The major model components needed are the intake manifold dynamics, volumetric efficiency, torque output, and lambda sensor model. Intake manifold dynamics can be modelled based on

the physics-based filling and emptying model. Regression models are available for static components, such as volumetric efficiency and engine torque output. A first-order order lag was used to model the lambda sensor. Transforming a physics based model into an LPV form is therefore more advantageous than the data identification approach.

During the model development, throttle and fuel injector dynamics were ignored, a one-state intake manifold model was used and engine rotational dynamics were not included (engine speed was considered an external parameter). Model parameters were estimated using driving cycle data, and the model was transformed to a quasi-LPV form by exploiting the natural physical structure of the solution. The engine hybrid model is an interconnection of the continuous-time intake manifold and lambda sensor dynamics, and the event-based mean-value models for the volumetric efficiency, torque production and exhaust manifold dynamics. The model for the control design was defined in an event-based time-frame, and consists of the Euler-discretized intake manifold, and lambda sensor dynamics in combination with explicit discrete-time delays. Fig. 2 shows the block diagram of the engine used for control design.

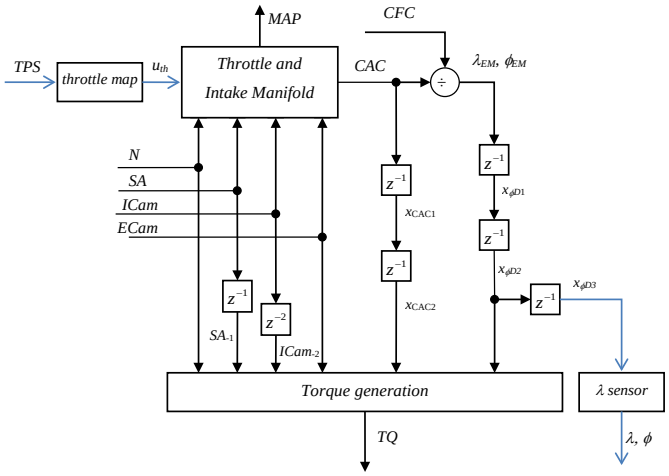

Fig. 2: **Structure of Engine Model for Control Design** (delay blocks represent discrete engine event time steps)

The model has 7 states: intake manifold pressure P_{im}, two delayed CAC states $x_{CAC1\text{-}2}$, three delayed in-cylinder equivalence ratio states $x_{\phi D1\text{-}3}$ and the state ϕ representing the output of lambda sensor. The outputs are the generated torque TQ and the in-cylinder fuel-air ratio ϕ_{IC}. The manipulated control variables are the Throttle Position (TPS) and Cylinder Fuel Charge (CFC). The *LPV* model used in the *MPC* controller suggested use of the quadratic function of the throttle area A_{th} as the effective control variable u_{th}:

$$u_{th} = A_{th} \cdot p_{CdA1} \cdot (p_{CdAx} A_{th} + 1) \tag{1}$$

The parameter p_{CdAx} is defined such that the function has a maximum at $A_{th,max}$, i.e. for *TPS* = 100%. The throttle position *TPS* can be retrieved from u_{th} using a one-to-one mapping. The equivalence ratio ϕ_{IC}, rather than the air-fuel ratio λ_{IC}, was chosen as the output to be controlled, exploiting its proportionality to the control input CFC. The state, input, and the controlled and measured output vectors:

$$x_0 = \begin{bmatrix} P_{im} & \phi & x_{\phi D1} & x_{\phi D2} & x_{\phi D3} & x_{CAC1} & x_{CAC2} \end{bmatrix}^T$$

$$u = \begin{bmatrix} u_{th} \\ CFC \end{bmatrix}, \; y_c = \begin{bmatrix} TQ \\ \phi_{IC} \end{bmatrix}, \; y_m = \begin{bmatrix} MAP \\ TQ \\ \phi \end{bmatrix} \tag{2}$$

State Equation Matrices: The discrete-time *LPV* model of the engine with both measured (y_m) and controlled (y_c) outputs is written in general form as:

$$\begin{cases} x_{0,t+1} = A_{0t} x_{0,t} + B_{0t} u_{t-k} + d_{x,t} \\ y_{m,t} = C_{0mt} x_{0,t} + E_{mt} u_{t-k} + d_{ym,t} \\ y_{c,t} = C_{ct} x_{0,t} + E_{ct} u_{t-k} + d_{yc,t} \end{cases} \tag{3}$$

where k is the common input delay and the notation X_t denotes $X(p_t)$, i.e. an LPV matrix evaluated based on the values of parameters at time t. The terms d_x, d_{ym} and d_{yc} represent known input signals other than the control u. The parameter vector in this problem contains the following variables p = (N, *MAP*, *SA*, *ICAM*, *ECAM*, T_{im}, P_{amb}, T_{amb}). Apart from the exogenous signals it also contains a system state (*MAP*), making the model quasi-*LPV*. The state matrices $A_0(p_t)$ and $B_0(p_t)$ can be constructed as:

$$A_0 = \begin{bmatrix} 1-\frac{V_{cyl}\eta}{V_{im}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1-\frac{t_s}{\tau_\lambda} & 0 & 0 & \frac{t_s}{\tau_\lambda} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{V_{cyl}\eta}{R_{air}T_{im}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad B_0 = \begin{bmatrix} \Omega & 0 \\ 0 & 0 \\ 0 & \frac{Stoich}{CAC} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

where $\Omega = t_s \sqrt{R_{air}} T_{im} P_{amb} \cdot \Psi \cdot (1 + p_{CdA2} P_{im}) / \sqrt{T_{amb}} V_{im}$.

These matrices, that are measured, or estimated, at time t, contain both constant and varying engine parameters. The volumetric efficiency $\eta(\cdot)$, throttle function $\Psi(P_{im}/P_{amb})$ and the cylinder air charge *CAC* expressions are given as:

$$\begin{aligned} \eta(P_{im}, N, ICam, ECam) &= p_{VE1} N^3 + p_{VE2} P_{im}^3 + p_{VE3} N^2 + \\ & p_{VE4} P_{im}^2 + p_{VE5} N + p_{VE6} P_{im} + p_{VE7} + p_{VE8} ICam + p_{VE9} ICam^2 \\ & + p_{VE10} ECam + p_{VE11} ECam^2 + p_{VE12} N^2 P_{im} + p_{VE13} N P_{im} \end{aligned} \tag{4}$$

$$\Psi(pr) = \begin{cases} pr^{\frac{1}{\gamma}} \sqrt{\frac{2\gamma}{\gamma-1}\left(1 - pr^{\frac{\gamma-1}{\gamma}}\right)}, & pr > \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma}{\gamma-1}} \\ \sqrt{\gamma}\left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{2(\gamma-1)}} & pr \le \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma}{\gamma-1}} \end{cases} \tag{5}$$

$$CAC = \dot{m}_{ac} \cdot t_s = (V_{cyl} / R_{air} T_{im}) \cdot \eta \cdot P_{im} \quad (6)$$

$$t_s = 0.5[rev] \cdot 60[s/\min] / N[rev/\min] \quad (7)$$

The effective disturbance input $d_{x,t}$ is zero in this LPV model. Indeed, all the disturbance inputs form elements of the *LPV* state matrices. From the definition of the measured and controlled outputs in (2), the output LPV matrices are:

$$C_{0m} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{TQ3} & 0 & 0 & p_{TQ2} \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \; E_{0m} = 0, \; d_{ym} = \begin{bmatrix} 0 \\ d_{TQ,t} \\ 0 \end{bmatrix}$$

$$y_{c,t} = \begin{bmatrix} TQ_t \\ \phi_{EM,t} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & p_{TQ3} & 0 & 0 & p_{TQ2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} x_t + \begin{bmatrix} 0 & 0 \\ 0 & \dfrac{Stoich}{CAC_t} \end{bmatrix} u_t + \begin{bmatrix} d_{TQ,t} \\ 0 \end{bmatrix}$$

The output torque components due to disturbance inputs:

$$d_{TQ} = p_{TQ1} + p_{TQ4} SA_{-1} + p_{TQ5} SA_{-1}^2 + p_{TQ6} N + p_{TQ7} N^2 + p_{TQ8} SA_{-1} \cdot N + p_{TQ9} SA_{-1} \cdot N^2 + p_{TQ10} ICam_{-2} + p_{TQ11} ICam_{-2}^2 + p_{TQ12} ECam + p_{TQ13} ECam^2$$

The volumetric efficiency and cam angles appear as LPV parameters; RPM appears indirectly through sample time; dependence of the discharge coefficient on the *MAP* state is taken into account by a term in the B_0 matrix (demonstrates *quasi*-LPV nature of the model). The control inputs u_{th} and *CFC* appear linearly in the equations, with the control signal u_{th} depending uniquely on the throttle position *TPS* and not on the state *MAP*. The terms in the torque model that are not model states appear as measured disturbance terms.

The above *LPV* model formulation is not unique. In fact, pointwise controllability and achievable performance depend on the choice of the model, even though the open-loop characteristics remain unaffected (Huang and Jadbabaie, 1999). This is a feature of *quasi-LPV* models. In some cases the model formulation follows naturally from the system structure, but in general the "best" formulation may not be obvious. The model does not rely on Jacobian linearization, i.e. it is valid for the whole range of engine operating conditions, limited only by the validity of the NL model.

IV. CONTROLLER DESIGN

There are many possible formulations and variations of the predictive control problem. The *Nonlinear Generalized Predictive Controller* (*NGPC*) algorithm used here seems well suited to real-time engine control applications. At each time step, the controller aims to minimize the sum of squares of predicted performance variables, with or without constraints on the control signal changes. The traditional method of introducing integral action in predictive control is to augment the system input by adding an integrator:

$$\begin{cases} x_{i,t+1} = \beta x_{i,t} + \Delta u_{t-k} \\ u_{t-k} = \beta x_{i,t} + \Delta u_{t-k} \end{cases} \quad (8)$$

$$u_{t-k} = (1/(1-\beta z^{-1}))\Delta u_{t-k} \quad (9)$$

The MPC cost function normally contains a penalty on the incremental change in control action Δu_t . The error weighting matrices can in this case just be constant matrices (no extra states). It is useful to define a generalized operator in unit-delay terms $\Delta_\beta = (1-\beta z^{-1})$ so that if β = 0 then $\Delta_\beta u_{t-k} = u_{t-k}$. The results therefore apply to both systems using control input and rate of change of control input, respectively. If β =1 equation (9) defines an integrator without additional delay. If the actual control input u_t is used, an alternative way of including integral action is to use a dynamically weighted error signal involving a high gain in low frequencies. The steady-state error should then be removed, otherwise the cost would increase indefinitely.

Performance variables and combined model: The output variables of interest are contained in the vector y_c:

$$y_{c,t} = C_{ct} x_t + E_{ct} \Delta_\beta u_{t-k} + d_{yc,t} + c_t \quad (10)$$

The controlled output y_c can be different from the measured output y_m. A 'robustness' signal c_t is included in an output disturbance model, to compensate for modelling mismatch:

$$\begin{aligned} x_{d,t+1} &= A_d x_{d,t} + B_d \omega_t \\ c_t &= C_{dc} x_{d,t} \end{aligned} \quad (11)$$

It is often desirable to consider dynamically weighted performance variables, to penalize the signals in different frequency ranges. In fact, it may also be useful to penalize actuator movements to prevent too aggressive (high-frequency) control actions. This motivates an introduction of dynamic weighting functions acting on the control error $(r_t - y_{c,t})$ and possibly on the control action u_t:

$$e_{p,t} = W_e(z^{-1})(r_t - y_{c,t}) : \begin{cases} x_{p,t+1} = A_p x_{p,t} + B_p (r_t - y_{c,t}) \\ e_{p,t} = C_p x_{p,t} + E_p (r_t - y_{c,t}) \end{cases} \quad (12)$$

The augmented state $x_t = \left[x_{0t}^T, x_{dt}^T, x_{it}^T, x_{pt}^T \right]^T$ for the combined model and vector of performance variables, including weighted error $z = e_p$ follows:

$$\begin{bmatrix} x_{0,t+1} \\ x_{d,t+1} \\ x_{i,t+1} \\ x_{p,t+1} \end{bmatrix} = \begin{bmatrix} A_0 & 0 & \beta B_0 & 0 \\ 0 & A_d & 0 & 0 \\ 0 & 0 & \beta I & 0 \\ -B_p C_c & -B_p C_{dc} & -\beta B_p E_c & A_p \end{bmatrix} \begin{bmatrix} x_{0t} \\ x_{dt} \\ x_{it} \\ x_{pt} \end{bmatrix} + \begin{bmatrix} B_0 \\ 0 \\ I \\ -B_p E_c \end{bmatrix} \Delta_\beta u_{t-k} + \begin{bmatrix} 0 & G_0 \\ 0 & 0 \\ 0 & 0 \\ B_p & -B_p H_c \end{bmatrix} \begin{bmatrix} r_t \\ d_t \end{bmatrix} + \begin{bmatrix} D_0 & 0 \\ 0 & B_d \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \xi_t \\ \omega_t \end{bmatrix}$$

$$e_{pt} = \begin{bmatrix} -E_p C_c & -E_p C_{dc} & -\beta E_p E_c & C_p \end{bmatrix} x_t$$

$$+\left[-E_p E_c\right]\Delta_\beta u_{t-k}+\left[E_p \quad -E_p H_c\right]\begin{bmatrix} r_t \\ d_t \end{bmatrix}$$

More concisely: $$\begin{cases} x_{t+1} = A_t x_t + B_t \Delta_\beta u_{t-k} + g_{x,t} + D\zeta_t \\ z_t = C_t x_t + E_t \Delta_\beta u_{t-k} + g_{y,t} \end{cases} \quad (13)$$

The model (13) defines the variables to be minimized. The dynamic control weighting is not used here, i.e. $W_u = I$. Limiting the controller bandwidth can also be achieved by using the constrained solution. The control law derivation summarized below is in *Grimble and Majecki* (2010).

Cost Function and Prediction Equations: At each sample instant, the predictive controller minimizes a criterion over a cost horizon N, involving a weighted combination of the predictions of performance variables and the control effort:

$$J_t(N) = \sum_{i=0}^{N-1}\left\{z_{t+k+i}^T z_{t+k+i} + \left(\Delta_\beta u_{t+i}\right)^T \Lambda_u \left(\Delta_\beta u_{t+i}\right)\right\} \quad (14)$$

Note the time shift of k sample time steps between the two quadratic terms in the cost, reflecting the fact that the control at the current time t can only affect the output with a k-steps delay. For simplicity, it will be assumed that $k = 1$. Consequently, the solution involves the initial step of computing the k-step state prediction. The cost (14) can then be represented in a vector-matrix form as:

$$J_N = Z_{t+k,N}^T Z_{t+k,N} + \left(\Delta_\beta U_{t,N}\right)^T \Lambda_U \left(\Delta_\beta U_{t,N}\right) \quad (15)$$

where $$Z_{t+k,N} = \begin{bmatrix} z_{t+k} \\ z_{t+k+1} \\ \vdots \\ z_{t+k+N-1} \end{bmatrix}, \quad \Delta_\beta U_{t,N} = \begin{bmatrix} \Delta_\beta u_t \\ \Delta_\beta u_{t+1} \\ \vdots \\ \Delta_\beta u_{t+N-1} \end{bmatrix} \quad (16)$$

The future values of the performance variables z are estimates given the information available at time t. They are based on the state estimates (provided by the *Kalman Filter*), current measured disturbances (assumed to stay constant within the prediction horizon), the reference signal (with or without future knowledge), previous control actions and the control input sequence $\Delta_\beta U_{t-1,N}$ (computed in the previous iteration). The aim is to find the control sequence $\Delta_\beta U_{t,N}$ that minimizes the cost (15). Invoking the receding horizon strategy, only the first element of the sequence is applied.

To minimize the cost index equations are needed to predict future values of performance variables. Based on (13), setting the stochastic noise inputs to zero, the state predictions:

$$\begin{aligned} x_{t+j} &= \left[A_{t+j-1}A_{t+j-2}\dots A_t\right]x_t + \left[A_{t+j-1}A_{t+j-2}\dots A_{t+1}\right]B_t\Delta_\beta u_t \\ &+ \left[A_{t+j-1}A_{t+j-2}\dots A_{t+2}\right]B_{t+1}\Delta_\beta u_{t+1} + \left[A_{t+j-1}\right]B_{t+j-2}\Delta_\beta u_{t+j-2} \\ &+ B_{t+j-1}\Delta_\beta u_{t+j-1} + \left[A_{t+j-1}A_{t+j-2}\dots A_{t+1}\right]g_{x,t} \\ &+ \left[A_{t+j-1}A_{t+j-2}\dots A_{t+2}\right]g_{x,t} + A_{t+j-1}g_{x,t} + g_{x,t} \end{aligned}$$

or $$x_{t+j} = x_{t+j}^0 + x_{t+j}^d + s_{t,j}^x \Delta_\beta U_{t,N} \quad (17)$$

Note that $x_{t+j}^0 = \left[A_{t+j-1}A_{t+j-2}\dots A_t\right]x_t$ represents the free response, x_{t+j}^d the forced response and $\Delta_\beta U_{t,N}$ the optimization variables. The *j-step-ahead* prediction follows:

$$\begin{aligned} z_{t+j} &= C_{t+j}x_{t+j} + E_{t+j}\Delta_\beta u_{t+j-k} + g_{y,t} \\ &= C_{t+j}s_{t,j}^x\Delta_\beta U_{t,N} + E_{t+j}\Delta_\beta u_{t+j-k} + C_{t+j}\left(x_{t+j}^0 + x_{t+j}^d\right) + g_{y,t} \\ &= s_{t,j}\Delta_\beta U_{t,N} + f_{t+j} \end{aligned} \quad (18)$$

where $$s_{t,j} = C_{t+j}s_{t,j}^x + E_{t+j} \quad (19)$$

$$f_{t+j} = C_{t+j}\left(x_{t+j}^0 + x_{t+j}^d\right) + g_{y,t} \quad (20)$$

Finally, the vector $Z_{t+k,N}$ can be written as:

$$Z_{t+k,N} = S_{t,N}\Delta_\beta U_{t,N} + F_{t,N} \quad (21)$$

with $S_{t,N} = \left[s_{t,1}^T \quad s_{t,2}^T \quad \cdots \quad s_{t,N}^T\right]^T$, $F_{t,N} = \left[f_{t+1}^T \quad f_{t+2}^T \quad \cdots \quad f_{t+N}^T\right]^T$

Due to the k-steps delay, the prediction equations (17) are shifted by (k-1) steps. The starting point for predictions is not the current $\hat{x}_t$ but $\hat{x}_{t+k-1}$. This can be computed from the previous controls assuming no future disturbance changes.

The parameter-dependent matrices in the above expressions are computed based on the future predicted states in (17). Alternatively, they can be found, at each control calculation, using the current state estimate x_t (frozen model). This results in a less computationally demanding algorithm but there is a loss of model accuracy. The future control sequence is needed to compute the future state estimates. An iterative "successive approximation" procedure can be used to improve predictions but adding to the computational burden.

Unconstrained and Constrained Solutions: The unconstrained minimum of the cost-function (15), given the prediction equation (21), is computed from a simple gradient calculation, or by completing the squares, to obtain:

$$\Delta_\beta U_{t,N} = -\left(S_{t,N}^T S_{t,N} + \Lambda_U\right)^{-1} S_{t,N}^T F_{t,N} \quad (22)$$

The same solution applies to both the absolute and incremental control formulation, with the corresponding changes in the system model and the resulting matrices $S_{t,N}$ and $F_{t,N}$. In accordance with the receding horizon principle, only the first element of this control sequence is applied as the input $u_t = \left[I\,0\dots0\right]\cdot U_{t,N}$. The whole sequence $U_{t,N}$ can be used in the next sample instant to perform the predictions. The unconstrained solution is faster to compute, and is acceptable when the normal operation is within the operational constraints. When optimal performance is demanded, the system will often operate close to the constraints imposed on actuator magnitude and rate, and outputs $U_{\min} \le U_{t,N} \le U_{\max}$, $\Delta U_{\min} \le \Delta U_{t,N} \le \Delta U_{\max}$ and

$Y_{\min} \le Y_{t,N} \le Y_{\max}$. They can be written in the linear matrix inequality form: $M_{\Delta_\beta U}\Delta_\beta U_{t,N} \le b_{\Delta_\beta U}$ (23)

The control magnitude and rate are related as:

$$\Delta U_t = U_t - U_{t-1} = \underbrace{\begin{bmatrix} -I \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{C_1} u_{t-1} + \underbrace{\begin{bmatrix} I & \dots & 0 & 0 \\ -I & I & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & -I & I \end{bmatrix}}_{C_2} U_t \quad (24)$$

$$U_t = \underbrace{\begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix}}_{C_3} u_{t-1} + \underbrace{\begin{bmatrix} I & \dots & 0 & 0 \\ I & I & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ I & \dots & I & I \end{bmatrix}}_{C_4} \Delta U_t \quad (25)$$

The output constraints can be represented in the standard form by writing the outputs in a form similar to (21):

$$Y_{t+1,N} = S^Y_{t,N} U_{t,N} + F^Y_{t,N} \quad (26)$$

with appropriate definition of $S^Y_{t,N}$ and $F^Y_{t,N}$. The minimum of (15), subject to the constraints, can be obtained by quadratic programming, using the Hildreth algorithm (Wang, 2009).

Time-Varying Kalman Filter: To construct the prediction matrices $S_{t,N}$ and $F_{t,N}$ the *NGPC* controller relies on the system states to be available at the current time *t*. These may be measured but a Kalman filter is used in practice to estimate them. The filter makes uses of the *LPV* model of the engine, the measurable disturbances and outputs. The process model and sensor noise covariances can be used as filter tuning parameters. The filter provides the state estimates including disturbance and the dynamic weighting states. The system model (13) includes stochastic white noise sources $\varsigma_t = \begin{bmatrix} \xi_t^T & \omega_t^T \end{bmatrix}^T$ and v_t, representing process and measurement noise and can be written as:

$$\begin{cases} x_{t+1} = A_t x_t + B_t \Delta_\beta u_{t-k} + g_{x,t} + D\zeta_t \\ y_{mt} = C_{mt} x_t + E_{mt} \Delta_\beta u_{t-k} + d_{ym,t} + v_t \end{cases} \quad (27)$$

The covariance matrices of the signals ς_t and v_t are denoted Q_N and R_N, respectively. These can be treated as tuning parameters, used to define the relative 'confidence' in the model and the measurements. The *D* matrix is used when there is more detailed knowledge about the system stochastic properties. When there is little information the *D* can be set to identity and then process noise directly represents the uncertainty associated with the estimation of the particular state. The part of the *D* matrix associated with the white noise input ω_t corresponds to the output disturbance model, including mismatch states. The signal u_{t-k} is fed to the *Kalman Filter*, as in Fig. 3. At time *t*, the time-varying Kalman Filter equations involve two steps:

Step 1: Computation of the Kalman gain, state estimation and update of the estimation error covariance (system matrices evaluated with the state prediction $\hat{x}_{t|t-1}$):

$$K_{t|t-1} = P_{t|t-1} C^T_{t|t-1} \left(C_{t|t-1} P_{t|t-1} C^T_{t|t-1} + R_N \right)^{-1} \quad (28)$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_{t|t-1} \left(y_{m,t} - C_{t|t-1} \hat{x}_{t|t-1} - E_{m,t} u_{t-k} - d_{ym,t} \right) \quad (29)$$

$$P_{t|t} = P_{t|t-1} - K_{t|t-1} C_{t|t-1} P_{t|t-1} \quad (30)$$

Step 2: State and covariance matrix predictions (system matrices evaluated with the state estimate $\hat{x}_{t|t}$):

$$\hat{x}_{t+1|t} = A_t \hat{x}_{t|t} + B_t u_{t-k} + g_{x,t} \quad (31)$$

$$P_{t+1|t} = A_t P_{t|t} A_t^T + D Q_N D^T \quad (32)$$

The model mismatch may result in a steady-state offset in the system output estimate which can sometimes be compensated by representing the mismatch by an output disturbance.

MPC Controller Structure: The overall NMPC controller structure in Fig. 3 is a form of separation principle and consists of the Kalman filter, state predictor and optimizer. The RHC block represents the receding horizon control strategy, whereby only the first element of the computed control sequence is used for actual control. The remaining elements are utilized for future state predictions.

MPC Design Issues and Weightings: The cost-function utilized is general using dynamic cost-function weightings. The sample time and prediction horizon *N* should be set so that the dominant transient behaviour is captured. There is a choice between the absolute and incremental control formulations. The 'Δu' approach is the classical approach. Integral action can also be added (when penalizing the control *u*) by including an integrator on the error weighting. The dynamic error and control weightings $W_e(z^{-1})$ and $W_u(z^{-1})$ are usually chosen to have low pass and high pass characteristics, respectively, and a constant control weighting matrix Λ_u may be used. The actuator and operating constraints (for QP computations) and the Q_N and R_N covariance's for the Kalman Filter are also design variables.

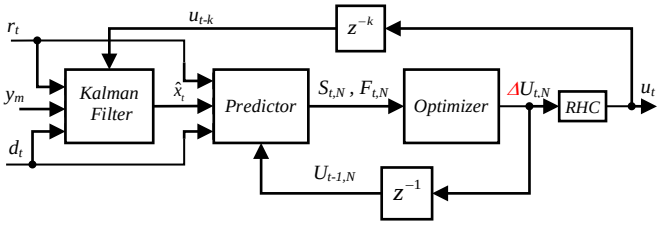

Fig. 3: **MPC Block Diagram Including Kalman Filter**

V. NGPC CONTROL SIMPLIFICATIONS

The predictive control algorithm as detailed in the previous section may result in an excessive computational burden on the production control processing unit, particularly when both

the sampling frequency and prediction horizon are high, and the constrained solution is used. For practical implementation the controller code must execute in real time in a deterministic way. The additional caveat in engine control is that the sample period varies with engine cycle and is dependent on the engine speed. Since all the code is executed at each trigger, it must have sufficient time to complete for the highest expected engine speed. Alternatively, the number of flops required may need to be reduced. The simplest way to reduce the numerical load is to reduce the horizon N. From (22), the *MPC* solution involves inverting at each step a matrix of size N, which has complexity $O(n^3)$. Even a small reduction of N can then bring significant computational savings, resulting in some degradation of performance. It is therefore important to strike the right balance between the algorithm complexity and the performance demanded.

Freezing Prediction Model: The state and output predictions performed at each step in (17) and (18) involve *LPV* model matrices, which in general vary with external parameters and/or system states. Consequently, these matrices need to be evaluated N times at each step. A simple way to reduce the number of computations is to fix the prediction model based on the parameters and states available at the current time t and use them for prediction as in the linear *GPC* case.

Connection Matrix and Control Profile: At each step, the *MPC* optimization problem with prediction horizon N involves the computation of N decision variables (control moves). With a large N, this may lead to over-parameterization and an impractical solution. One commonly used solution is to introduce a separate control horizon N_u which uses the first N_u controls as the decision variables, and fixes the remaining (N–N_u) ones. Here we generalize this idea by defining a control profile P_u of the form:

$$row\{P_u\} = [number\ of\ samples\ held,\ repetitions] \qquad (33)$$

For example, letting P_u = [1 3; 2 2; 3 1] represents 3 different initial controls, then 2 samples with the same control used but this is repeated again, and finally 3 samples with the same control used. Based on a control profile, it is possible to specify the transformation matrix T_u, relating the control moves to be optimized (say vector V) to the full control vector (U). For incremental control, the connection matrix:

$$\begin{bmatrix} \Delta u_t \\ \Delta u_{t+1} \\ \Delta u_{t+2} \\ \Delta u_{t+3} \\ \Delta u_{t+4} = 0 \\ \Delta u_{t+5} \\ \Delta u_{t+6} = 0 \\ \Delta u_{t+7} \\ \Delta u_{t+8} = 0 \\ \Delta u_{t+9} = 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta v_1 \\ \Delta v_2 \\ \Delta v_3 \\ \Delta v_4 \\ \Delta v_5 \\ \Delta v_6 \end{bmatrix} \Rightarrow \quad \Delta U = T_{\Delta u} \Delta V$$

This represents a situation with $N_u = 3+2+1 = 6$ independent control moves and $N = 3\times1+2\times2+1\times3 = 10$ sample points. The computation of 4 control moves has been avoided, representing a substantial computational saving. This approach, often termed 'move blocking' in the MPC literature, can provide the solution to the *GPC* class of problems with different control and error horizons. The changes to the algorithm are minimal. It also solves the problem where the horizons are the same but the control changes are not allowed at each sample instant. The usual approach is to assume future control changes are null after the control horizon N_u and in this case the connection matrix can be defined to have N_u+1 rows and N+1 columns. For example, the form of the T_u matrix, when N = 6 and N_u = 2:

$$T_u = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

$$\text{and } \Delta U_{t,N} = T_u \Delta U_{t,Nu} = \begin{bmatrix} \Delta u_t^T & \Delta u_{t+1}^T & \Delta u_{t+2}^T & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

VI. RESULTS

Two simulation scenarios are presented here. Fig. 4 shows the comparison between the nominal MPC design and the conventional controller for a fragment of an FTP drive cycle. Extended prediction horizon and handling system interactions by the multivariable controller generally lead to improved tracking for both torque and lambda output signals. The more aggressive MPC causes overshoots at low/negative torques, however in reality this operating range would be handled by a separate idle speed controller.

The responses to torque reference and speed disturbance step changes for different MPC designs are shown in Fig. 5. The nominal case DC0 was based on the absolute control formulation (β = 0), constrained solution and equal horizons $N = N_u$ = 15. The other design cases represent various design simplifications and are listed in Table 1, where their relative speed-up factors are also benchmarked against the nominal. Analysis of the figures and table leads to several conclusions. First, freezing the prediction model (neglecting the time variation of state matrices) gives almost 20% speed-up without changing the results noticeably (this case was not included in the figure). As expected, reducing the prediction horizon significantly reduces the computational load, however the dynamic performance also changes substantially. On the other hand, the use of separate control horizon or control profile/connection matrix allows reducing the load without degrading performance.

Table 1 **MPC Design Cases and speed-up factors due to the algorithmic simplifications used**

DC	Configuration	Speed-Up (%)
0	Nominal design case (N = 15)	0.0
1	Prediction model frozen	19.3
2	Unconstrained solution	48.5
3	N = 12	3.1
4	N = 9	18.4
5	N = 6	39.7
6	N = 15, Pu = [1 6; 9 1]	27.2

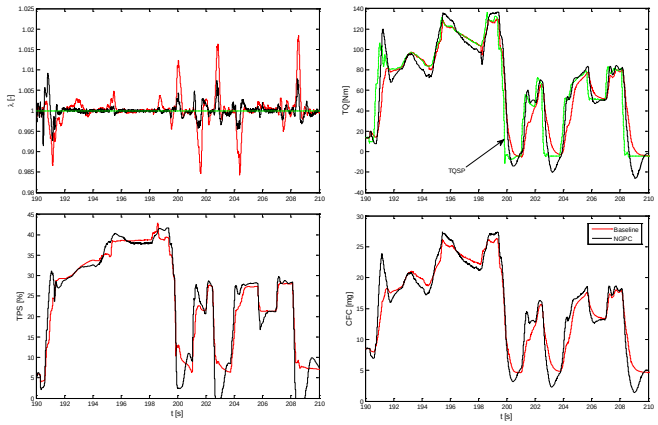

Fig. 4: **Comparison of the classical (red) and MPC (black) control for a fragment of FTP cycle**

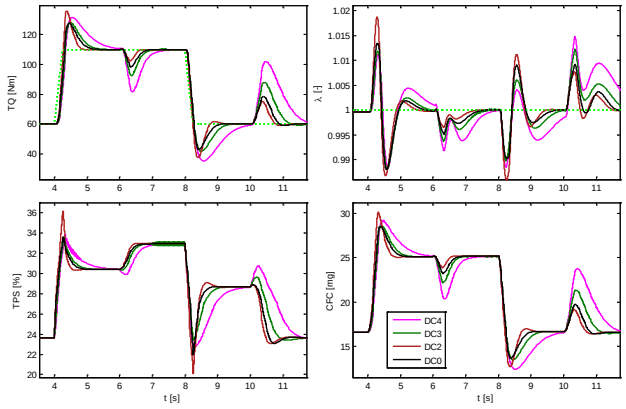

Fig. 5: **Responses to torque reference and speed disturbance step changes for MPC design (**cases: DC0 (black), DC2 (red), DC3 (green) and DC4 (magenta))

In fact, down-sampling the control action may lead to performance improvement. The unconstrained solution is also less computationally expensive and is always considered an option. For the constrained version, computational savings can also be obtained by reducing the number of the active-set iterations in the QP algorithm. As this is related to convergence of the algorithm, this step must be taken carefully by gradually reducing the number of iterations.

VII. CONCLUSIONS

The LPV-based version of the predictive control law is computationally intensive, although simpler than most nonlinear predictive algorithms. Attention therefore turned to simplifications to the controller, to reduce the computational load. We considered the application to the engine control problem and benchmarked the performance. The impact of code optimization was assessed, and the results show significant savings are possible through a combination of measures to modify the standard design process and simplify the algorithm. These were validated during the trials of embedded engine control code.

The value of this work also lies in the systematic framework for NL MPC based on quasi-LPV models, including absolute and incremental control. The use of an output disturbance model to reduce the effects of plant-model mismatch is useful, as is the way of limiting the control moves to gain computational savings. The simulation results for the torque tracking and air-fuel ratio regulation problem were shown. On-going work involves the use of variable cam phasing to improve fuel economy and real-time implementation.

REFERENCES

Bemporad, A and M Morari, (2004), Robust model predictive control: A survey, in Proc. European Control Conference, Porto.

Clarke, D.W., and C. Montadi, (1989), Properties of generalised predictive control, Automatica, Vol. 25, No. 6, pp. 859-875.

Cutler C.R. and Ramaker B.L. (1979), Dynamic matrix control - A computer control algorithm, A.I.C.H.E, 86th National Meeting

Grimble M.J. and P. Majecki (2010), State-space approach to nonlinear predictive generalized minimum variance control, International Journal of Control, Vol. 83, pp.1529-1547.

Herceg M., T. Raff, R. Findeisen and F. Allgöwer, (2006), "Nonlinear model predictive control of a turbocharged engine", Proc. 2006 IEEE Int. Conf. on Control Algs, Munich, pp. 2766-71

Huang Y. and A. Jadbabaie, (1999) Nonlinear H-infinity control: an enhanced quasi-LPV approach, IFAC World congress, Beijing.

Jankovic M., (2002) "Nonlinear control in automotive engine applications", Proceedings of 15th MTNS Conf., South Bend, IN.

Qin, S and T Badgwell, (2003), A survey of industrial model predictive control technology, Con. Eng. Prac., Vol. 11, pp. 733-764

Richalet J., A. Rault, J.L. Testud, J. Papon (1978), Model predictive heuristic control applications to industrial processes, Automatica, 14, pp. 413-428.

Vermillion C, K Butts, and K Reidy, (2010), Model Predictive Engine Torque Control, ACC, Baltimore.

Wang L., (2009) Model Predictive Control System Design and Implementation Using Matlab, Springer.

Casavola, A, Famularo, D and G Franzè, 2003, Predictive control of constrained nonlinear systems via LPV linear embeddings, Int. Jour of Robust and Nonlinear Control, Vol.13; Issue 3-4; pp. 281–294.

Chisci, L., F, Paola and G Zappa, 2003, Gain-scheduling MPC of nonlinear systems, Int Jour. of Robust and Nonlin Cont., 13; pp.3-4.

Casavola, A., Famularo, D. and G Franze, 2002, A feedback min-max MPC algorithm for LPV systems subject to bounded rates of change of parameters, IEEE Trans. A.C, ,47, No.7, pp.1147-1153.

Besselmann, T., J. Lofberg. and M Morari, 2012, Explicit MPC for LPV Systems: Stability and Optimality, IEEE Transactions on Automatic Control, vol.57, No.9, pp.2322-2332.

Li D., and Y. Xi, 2010, The Feedback Robust MPC for LPV Systems With Bounded Rates of Parameter Changes, IEEE Trans. on Automatic Control, vol.55, No.2, pp.503-507, Feb., pp.503-507.

Duan G-R and H-H Yu, 2013, LMIs in Control Systems: Analysis, Design and Applications, CRC Press, ISBN 9781466582996.