

On the Tradeoffs between Static and Dynamic Adaptive Optimization for an Automotive Application

Anthony D'Amato, Yan Wang, Dimitar Filev, and Enrique Remes
 Ford Motor Company

ABSTRACT

Government regulations for fuel economy and emission standards have driven the development of technologies that improve engine performance and efficiency. These technologies are enabled by an increased number of actuators and increasingly sophisticated control algorithms. As a consequence, engine control calibration time, which entails sweeping all actuators at each speed-load point to determine the actuator combination that meets constraints and delivers ideal performance, has increased significantly. In this work we present two adaptive optimization methods, both based on an indirect adaptive control framework, which improve calibration efficiency by searching for the optimal process inputs without visiting all input combinations explicitly. The difference between the methods is implementation of the algorithm in steady-state vs dynamic operating conditions. The goal of this work is to study the optimization performance tradeoffs between robustness to sensor noise and required optimization time. We demonstrate both methods on an engine installed in a dynamometer where multiple actuators were calibrated to minimize Break Specific Fuel Consumption (BSFC), while satisfying input and output constraints.

CITATION: D'Amato, A., Wang, Y., Filev, D., and Remes, E., "On the Tradeoffs between Static and Dynamic Adaptive Optimization for an Automotive Application," *SAE Int. J. Commer. Veh.* 10(1):2017, doi:10.4271/2017-01-0605.

I. INTRODUCTION

Automotive engine controls are developed using extensive engine mapping processes, that is, the characterization of engine performance in response to various actuator combinations. From the mapping process, optimal actuator settings that produce desired engine characteristics are extracted for use in control calibrations. The optimal actuator settings are stored in look-up tables or incorporated into regression models. The conventional engine mapping and calibration process, which until recently only included a limited number of engine control inputs such as throttle and spark, involved testing each engine speed/load point while sweeping all actuators through their full range. In recent years, fast-paced industry-wide development of electronics, motor, and computer technologies have introduced new actuators and sensors that add flexibility to the engine calibration process. This increased degree of freedom provides the opportunity to improve emissions, fuel consumption and/or peak torque, while the increased number of actuators, such as External Exhaust Gas Recirculation (EGR), Twin Independent Variable Valve Timing (TiVCT), wastegate, fuel rail pressure, various fuel injection modes, increases the required mapping and calibration time exponentially. The conventional mapping and calibration processes has become expensive in terms of dynamometer, data collection and post processing resources. The mapping and calibration process has become a bottleneck in the production implementation of new technologies.

Past research [3] considered optimizing BSFC while searching for optimal values of spark and cam timings for TiVCT engines. The use of extremum seeking methods have been tested with and without constraints on combustion stability. The three parameters (spark timing, intake valve opening, exhaust valve closing) need to be tuned together given the significant effect cam timing has on burn rate while in-cylinder dilution and turbulence occurs. Other work [10] [11] [12] [13] also exploited extremum seeking for engine calibration purposes. More recently [4] have proposed a least squares support vector machine model that serves as an objective function to optimize idle speed control based on direct search methods such as genetic algorithms to handle the multi-variable constrained optimization problem. Finally, commercial tools, specifically, Cameo, have been developed in which transient data is collected to identify a dynamic model. This model is then used for static engine calibration [16].

In this work, we examine two indirect adaptive optimization methods that produce steady-state actuator settings that simultaneously meet targets and constraints with the minimum required fuel. As shown in [Figure 1](#), the two methods use a common architecture, that is, iteratively identifying a local plant model, which characterizes the engine response around the current operating point, and then using this model to estimate the the required actuator inputs to meet target and minimization objectives.

The difference between the methods is implementation in steady-state vs transient operating conditions. In the static variant, the plant model is estimated as a matrix of partial derivatives or Jacobian between the

change in actuator inputs and the change in engine outputs. The Jacobian is then inverted at each steady-state interval to determine the optimal actuator settings. In the dynamic case, the method is implemented in real-time, a linear dynamic model is estimated and inverted in place of the Jacobian. We study the tradeoff of noise robustness in the static case versus optimization time in the dynamic case. The methods are demonstrated on a test engine where data from the current calibration method is used to benchmark the methods. This paper is organized as follows: In the first section the adaptive control problem is formulated, we present the parameter estimation and model inversion methods required to implement the proposed adaptive optimization strategies. In section two, we present the method specific problem formulation and process flow charts. Finally, in section three we compare optimization results using a current production engine on a production dynamometer and present the bench marking results.

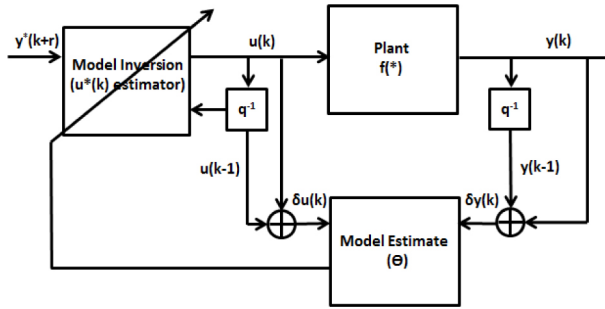


Fig. 1. Incremental adaptive model predictive control architecture

II. ADAPTIVE CONTROL PROBLEM FORMULATION

Consider the discrete-time nonlinear system

$$y(k) = f(u(k-r), \dots, u(k-n), y(k-1), \dots, y(k-n)), \quad (1)$$

where n is the system order, $r \leq n$ is the relative degree, $y(k) \in \mathbb{R}^{l_y}$ is the output, $u(k) \in \mathbb{R}^{l_u}$ are the normalized inputs, $f: \mathbb{R}^{l_u} \rightarrow \mathbb{R}^{l_y}$ is the plant, and $y^*(k) \in \mathbb{R}^{l_y}$ is the desired output. Furthermore, the magnitude of $u(k)$ is bounded such that $0 \leq \mu_i^- \leq u_i(k) \leq \mu_i^+ \leq 1$ for $i = 1, \dots, l_u$ and the transition rate is bounded such that $\|u_i(k) - u_i(k-1)\| \leq \zeta_i$ for $i = 1, \dots, l_u$. Figure 1 shows the architecture of the incremental adaptive model predictive control framework.

Next, consider the perturbations

$$\delta y(k) \triangleq y(k) - y(k-1), \quad (2)$$

$$\delta u(k) \triangleq u(k) - u(k-1), \quad (3)$$

about the operating point $u(k), y(k)$.

We assume that for small perturbations (1) can be written as

$$\delta y(k) = \theta \phi(k) + \beta_r \delta u(k-r), \quad (4)$$

where $\theta \in \mathbb{R}^{l_y \times \ell[(n-r)l_u + nl_y]}$ are model parameters and

$$\phi(k) = \begin{bmatrix} \delta u(k-r-1) \\ \vdots \\ \delta u(k-n) \\ \delta y(k-1) \\ \vdots \\ \delta y(k-n) \end{bmatrix} \in \mathbb{R}^{(n-r)l_u + nl_y}. \quad (5)$$

The goal of this work is to determine the ideal control $u^*(k-r)$, which yields the ideal outputs

$$y^*(k) = f(u^*(k-r), u(k-r-1), \dots, u(k-n), \quad (6)$$

$$y(k-1), \dots, y(k-n)), \quad (7)$$

$$\delta y^*(k) = \theta^* \phi(k) + \beta_r^* \delta u^*(k-r), \quad (8)$$

where

$$\delta y^*(k) = y^*(k) - y(k-1), \quad (9)$$

$$\delta u^*(k) = u^*(k) - u(k-1). \quad (10)$$

To solve for the ideal controls we use the known desired output $y^*(k)$, a model estimate $\hat{\theta}(k)$ and $\hat{\beta}_r(k)$, we introduce an estimate $\hat{u}(k)$ of the ideal controls $u^*(k)$, such that $\mu_i^- \leq \hat{u}_i(k) \leq \mu_i^+$, for $i = 1, \dots, l_u$, $\|\hat{u}_i(k) - u_i(k-1)\| \leq \zeta_i$ for $i = 1, \dots, l_u$ and

$$\delta y^*(k) = \hat{\theta}(k) \phi(k) + \hat{\beta}_r(k) \delta \hat{u}(k-r), \quad (11)$$

where

$$\delta \hat{u}(k) \triangleq \hat{u}(k) - u(k-1). \quad (12)$$

III. MODEL ESTIMATION: RECURSIVE LEAST SQUARES UPDATE

To estimate the model parameters we write (4) as

$$\delta y(k) = \Theta(k) \Phi(k), \quad (13)$$

where

$$\hat{\Theta}(k) \triangleq \begin{bmatrix} \hat{\beta}_r(k) & \hat{\theta}(k) \end{bmatrix}, \quad (14)$$

and

$$\Phi(k) = \begin{bmatrix} \delta u(k-r) \\ \phi(k) \end{bmatrix}, \quad (15)$$

Next, we define the *recursive least-squares cost function*

$$J_R(\hat{\Theta}(k)) \triangleq \sum_{i=r+1}^k \lambda^{k-i} \|\hat{\Theta}(k)\Phi(i) - \delta y(i)\|^2 + \lambda^k \text{Tr}(\hat{\Theta}(k) - \hat{\Theta}(0))P^{-1}(0)(\hat{\Theta}(k) - \hat{\Theta}(0))^T, \quad (16)$$

where $P(0) \in \mathbb{R}^{[(n-r)l_u + nl_y] \times [(n-r)l_u + nl_y]}$ is positive definite and $\lambda \in (0, 1]$ is the forgetting factor. The minimizer of (16) is

$$\begin{aligned} \hat{\Theta}(k) &\triangleq \hat{\Theta}(k-1) + [\hat{\Theta}(k-1)\Phi(k-1) - \delta y(k-1)] \\ &\quad \cdot [\Phi^T(k-1)P(k-1)\Phi(k-1) + \lambda]^{-1} \\ &\quad \cdot \Phi^T(k-1)P(k-1), \end{aligned} \quad (17)$$

and $P(k)$ is updated by

$$\begin{aligned} P(k) &\triangleq \lambda^{-1}P(k-1) - \lambda^{-1}P(k-1)\phi(k-1) \\ &\quad \cdot [\Phi^T(k-1)P(k-1)\Phi(k-1) + \lambda]^{-1} \\ &\quad \cdot \Phi^T(k-1)P(k-1). \end{aligned} \quad (18)$$

We initialize $P(0) = \beta_1 I$, where $\beta_1 > 0$.

IV. MODEL INVERSION USING EQUALITY CONSTRAINED QUADRATIC PROGRAMMING

Before attempting to solve the model inversion problem we separate the outputs into two groups, those with explicit targets, and those we wish to minimize, specifically, let $\delta y_{1,w}(k)$ be the components of $\delta y(k)$ with explicit targets, where $w \leq l_u$, and let $\delta y_{w+1,l_y}(k)$ be the outputs we wish to minimize. Next, let $\hat{\beta}_{r,1,w}(k)$ be rows 1 through w of $\hat{\beta}_r(k)$, likewise, let $\hat{\theta}_{1,w}(k)$ be rows 1 through w of $\hat{\theta}(k)$. Finally, at time k we wish to determine $\hat{u}(k)$ such that $\|y(k+r) - y^*(k+r)\|$ is small. Next, we define

$$\begin{aligned} z(k+r) &= \hat{\theta}(k+r)\phi(k+r) + \\ &\quad \hat{\beta}_r(k+r)\delta \hat{u}(k) - \delta y^*(k+r), \end{aligned} \quad (19)$$

which is (11) propagated r steps into the future. Next, we assume that $\Theta(k) - \Theta(k-1)$ is small such that (19) can be written as

$$\begin{aligned} \hat{z}(k+r) &= \hat{\theta}(k)\phi(k+r) + \\ &\quad \hat{\beta}_r(k)\delta \hat{u}(k) - \delta y^*(k+r). \end{aligned} \quad (20)$$

Finally, we determine $\hat{u}(k)$ by minimizing the cost function

$$J(\delta \hat{u}(k)) = \hat{z}_{w+1,l_y}^T(k+r)\hat{z}_{w+1,l_y}(k+r), \quad (21)$$

subject to

$$C(k)\delta \hat{u}(k) \leq D(k), \quad (22)$$

where

$$C(k) = \begin{bmatrix} I_{l_u} \\ -I_{l_u} \\ I_{l_u} \\ -I_{l_u} \\ \hat{\beta}_{r,1,w} \\ -\hat{\beta}_{r,1,w} \end{bmatrix} \in \mathbb{R}^{2(2l_u+w) \times l_u},$$

$$\begin{aligned} D(k) &= \begin{bmatrix} \mu_1^+ - u_1(k-1) & \cdots & \mu_{l_u}^+ - u_{l_u}(k-1) \\ -\mu_1^- + u_1(k-1) & \cdots & -\mu_{l_u}^- + u_{l_u}(k-1) \\ \xi_1 & \cdots & \xi_{l_u} \\ \xi_1 & \cdots & \xi_{l_u} \\ (I + \rho)y_{1,w}^*(k+r) - \hat{\theta}_{1,w}\phi(k) \\ -(I - \rho)y_{1,w}^*(k+r) + \hat{\theta}_{1,w}\phi(k) \end{bmatrix}^T \in \mathbb{R}^{2(2l_u+w)}, \end{aligned}$$

where $\rho \in \mathbb{R}^{w \times w}$ is a diagonal matrix with entries $\sigma_{i,i} \in (0, 1]$, for $i = 1, \dots, w$ which are the target following tolerance boundaries.

Substituting (20) into (21) yields

$$J(\delta \hat{u}(k)) = \delta \hat{u}^T(k)\mathcal{A}(k)\delta \hat{u}(k) + \delta \hat{u}^T(k)\mathcal{B}(k) + \mathcal{C}(k),$$

where

$$\mathcal{A}(k) \triangleq \hat{\beta}_{r,w+1,l_y}^T(k)\hat{\beta}_{r,w+1,l_y}(k), \quad (23)$$

$$\begin{aligned} \mathcal{B}(k) &\triangleq -2\hat{\beta}_{r,w+1,l_y}^T(k)[\hat{\theta}_{w+1,l_y}(k)\phi(k+r) \\ &\quad - \delta y_{w+1,l_y}^*(k+r)], \end{aligned}$$

$$\begin{aligned} \mathcal{C}(k) &\triangleq \phi^T(k+r)\hat{\theta}_{w+1,l_y}^T(k)\hat{\theta}_{w+1,l_y}(k)\phi(k+r) \\ &\quad - \phi^T(k+r)\hat{\theta}_{w+1,l_y}^T(k)\delta y_{w+1,l_y}^*(k+r) \\ &\quad - \delta y_{w+1,l_y}^{*T}(k+r)\hat{\theta}_{w+1,l_y}^T(k) \\ &\quad + \delta y_{w+1,l_y}^{*T}(k+r)\delta y_{w+1,l_y}^*(k+r). \end{aligned} \quad (24)$$