Published 03/28/2017
Copyright © 2017 SAE International
doi:10.4271/2017-01-0605
saecomveh.saejournals.org

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.

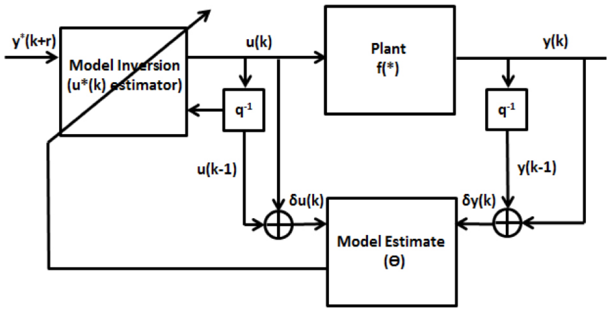

Fig. 1. Incremental adaptive model predictive control architecture

II. ADAPTIVE CONTROL PROBLEM FORMULATION

Consider the discrete-time nonlinear system

$$y(k) = f(u(k-r), \dots, u(k-n), y(k-1), \dots, y(k-n)), \tag{1}$$

where n is the system order, $r \le n$ 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, $f : \mathbb{R}^{l_u} \to \mathbb{R}^{l_y}$ 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 $0 \le \mu_i^- \le u_i(k) \le \mu_i^+ \le 1$ for $i = 1,\dots, l_u$ and the transition rate is bounded such that $\|u_i(k) - u_i(k-1)\| \le \xi_i$ for $i = 1,\dots, l_u$. Figure 1 shows the architecture of the incremental adaptive model predictive control framework.

Next, consider the perturbations

$$\delta y(k) \triangleq y(k) - y(k-1), \tag{2}$$

$$\delta u(k) \triangleq u(k) - u(k-1), \tag{3}$$

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

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

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

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

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

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

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

$$y(k-1), \dots, y(k-n)), \tag{7}$$

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

where

$$\delta y^*(k) = y^*(k) - y(k-1), \tag{9}$$

$$\delta u^*(k) = u^*(k) - u(k-1). \tag{10}$$

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 $\mu_i^- \le \hat u_i(k) \le \mu_i^+$, for $i = 1,\dots, l_u$, $\|\hat u_i(k) - u_i(k-1)\| \le \xi_i$, for $i = 1,\dots, l_u$, and

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

where

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

III. MODEL ESTIMATION: RECURSIVE LEAST SQUARES UPDATE

To estimate the model parameters we write (4) as

$$\delta y(k) = \Theta(k)\Phi(k), \tag{13}$$

where

$$\hat{\Theta}(k) \triangleq \left[\begin{array}{cc} \hat{\beta}_r(k) & \hat{\theta}(k) \end{array} \right], \tag{14}$$

and

$$\Phi(k) = \left[\begin{array}{c} \delta u(k-r) \\ \phi(k) \end{array} \right], \tag{15}$$

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

$$\begin{aligned} J_{\mathrm{R}}(\hat{\Theta}(k)) \triangleq & \sum_{i=r+1}^{k} \lambda^{k-i} \|\hat{\Theta}(k)\Phi(i) - \delta y(i)\|^2 \\ & + \lambda^k \mathrm{Tr}(\hat{\Theta}(k) - \hat{\Theta}(0))P^{-1}(0)(\hat{\Theta}(k) - \hat{\Theta}(0))^{\mathrm{T}}, \end{aligned} \tag{16}$$

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

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

and $P(k)$ is updated by

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

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 $w \le l_y$ and let $\delta y_{w+1,l_y}$ 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

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

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

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

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

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

subject to

$$C(k)\delta\hat{u}(k) \le D(k), \tag{22}$$

where

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

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

where $\rho \in \mathrm{R}^{w\times w}$ is a diagonal matrix with entries $\sigma_{i,i} \in (0, 1]$, for $i = 1,\ldots, w$ which are the target following tolerance boundaries. Substituting (20) into (21) yields

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

where

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

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

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

Next, let $\mathcal{A}^\dagger(k)$ be the generalized inverse of $\mathcal{A}(k)$, then

$$\delta\hat{u}_1(k) = -\frac{1}{2}\mathcal{A}^\dagger(k)\mathcal{B}(k). \tag{25}$$

is the unconstrained minimizer of (21). Next, the constrained solution of (21) subject to (22) is found by solving the linear system

$$\begin{bmatrix} \mathcal{A}(k) & C^{\mathrm{T}}(k)R(k) \\ R(k)C(k) & 0 \end{bmatrix} \begin{bmatrix} \delta u_2(k) \\ \lambda_L \end{bmatrix} = \begin{bmatrix} \mathcal{B}(k) \\ D(k) \end{bmatrix}, \tag{26}$$

where $\lambda^L \in \mathbb{R}^{l_u}$ are the lagrange multipliers, $\delta u_2(k)$ are the constrained controls, $R(k) \in \mathbb{R}^{2(2l_u+w)\times 2(2l_u+w)}$ is a positive semi-definite control penalty and

$$\begin{bmatrix} \delta u_2(k) \\ \lambda_L \end{bmatrix} = \begin{bmatrix} \mathcal{A}(k) & C^{\mathrm{T}}R(k) \\ R(k)C & 0 \end{bmatrix}^\dagger \begin{bmatrix} \mathcal{B}(k) \\ D \end{bmatrix}. \tag{27}$$

The matrix $R(k)$ is chosen using the algorithm:

```
Let $x(k) = C(k)\delta\hat{u}_1(k)$.
Step 1: Compute (25)
Step 2:For $i = 1, \ldots, 2(2l_u + w)$,
  if $x_i(k) > D_i(k)$, then $R_{i,i}(k) = 1$,
Step 3: Compute (27)
Step 4: if $x_i(k) < D_i(k)$ for $i = 1, \ldots, l_u$,, then $\delta u(k) = \delta u_2(k)$
  else goto Step 2.
```

In place of the presented method, other equality constrained optimization methods may be used to solve (21) subject to (22), such as LSQLIN or QUADPROG.

V. STATIC VERSUS DYNAMIC ADAPTIVE OPTIMIZATION

In this section we discuss the operational differences and parameter choices for both the static and dynamic variations of the methodology. In order to distinguish between the algorithms, which have different control and update rates, we introduce $t(\epsilon)$, which is the real-world time sampled at the maximum rate allowable by the test hardware, and define $\tilde{t}_s = t(\epsilon) - t(\epsilon - 1)$ as the hardware sampling rate. Next, $t(k)$ is the real-world time sampled at the rate used by adaptive optimization, where $t_s = t(k) - t(k - 1)$ depends on the method used. Furthermore, let

$$\hat{\tilde{y}}(\epsilon) = \tilde{y}(\epsilon) + d(\epsilon), \tag{28}$$

be the dynamometer outputs reading sampled at $\tilde{t}_s$, where $\tilde{y}(\epsilon)$ are the true outputs and $d(\epsilon)$ is zero-mean white noise. Finally, the relationship between the maximum sampling rate data index ϵ, and k used by the optimization is

$$t(k) = \gamma\epsilon, \tag{29}$$

where γ is a positive integer.

The methods were validated on a GTDI engine, where the number of inputs is $l_u = 5$, which are throttle/wastegate, spark, Start Of Injection (SOI), Fuel Rail Pressure (FRP) and normalized cam position. The number of outputs is $l_y = 4$, that is, load, Crank Angle at 50 percent Burn (CA50), Coefficient of Variation (COV) of Indicated Mean Effective Pressure (IMEP), and Break Specific Fuel Consumption (BSFC). The goal of the testing was to determine the inputs that provide the minimum BSFC, with constraints on the load and CA50 tracking, and an upper bound on COV of IMEP.

A. Static Optimization

For the static implementation, we choose $n = 0$, and $r = 0$. Specifically, (4), (14) and (15) become $\delta y(k) = \beta_0 \delta u(k)$, $\delta y(k) = \beta_0\delta u(k), \hat{\Theta}(k) = \hat{\beta}_0(k)$ and $\Phi(k) = \delta u(k)$, respectively. Next, the control and adaptation rate t_s is variable since the required time to reach steady-state is dependent on the magnitude of actuator perturbation and hardware nonlinearities. We define steady-state by monitoring the standard deviation of a data window once below a user selected threshold, the instantaneous outputs $\hat{\tilde{y}}(\epsilon)$ are averaged such that

$$y(k) = \frac{1}{m}\sum_{i=1}^{m}\hat{\tilde{y}}(t(k) - i) \approx \tilde{y}(t(k) - 1). \tag{30}$$

Next, while implementing the optimization process, certain actuator and speed load combinations produce undesirable engine operation that may result in termination of the test due to test facility protection, or damage to the engine. These cases include operating points that produce, for example, high exhaust temperatures and/or engine knock. To mitigate these undesirable operating conditions, a supervisory control is implemented in real-time and in parallel to the adaptive optimization. For example, exhaust temperature can be lowered to an acceptable limit by enriching the fuel mixture. For knock, spark is retarded from the algorithm specific value that is meant to deliver the best fuel. When the supervisory control manipulates an actuator used in the optimization, the manipulated value is used in the plant modeling steps, (17) and (18).

Figure 2 is a logical flow chart of the static adaptive optimization method. The communication link with the dynamometer controller and outer-loop supervisory control are shown.

B. Dynamic Optimization

For dynamic optimization the controller parameters are $n = 2$, and $r = 0$. The control and adaptation rate is increased compared to the static version. Furthermore,

$$y(k) = \hat{\tilde{y}}(t(k) - 1) = \tilde{y}(t(k) - 1) + d(t(k) - 1), \tag{31}$$

that is, sensor noise is not mitigated in the dynamic optimization version of the algorithm.

In the dynamic case, the same supervisory mitigation is required for knock and exhaust temperature. However, the key difference as demonstrated in Figure 3 is operation of the safety controllers in series with the adaptive optimization as opposed to in parallel.

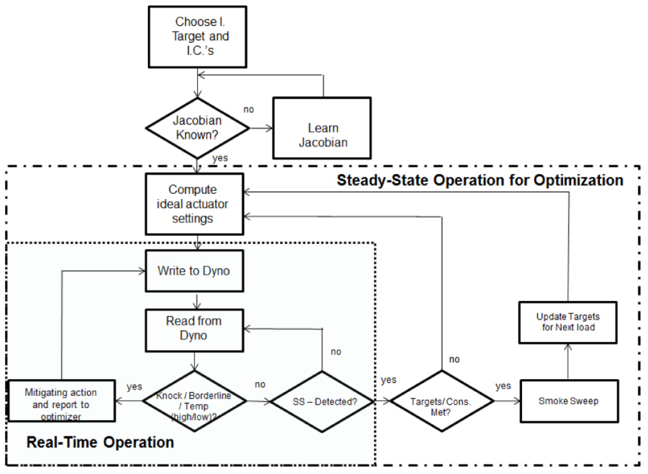
Choose I. Target and I.C.'s
Jacobian Known?
no
Learn Jacobian
yes
Steady-State Operation for Optimization
Compute ideal actuator settings
Write to Dyno
Read from Dyno
no
Update Targets for Next load
Mitigating action and report to optimizer
yes
Knock / Borderline / Temp (high/low)?
no
SS – Detected?
yes
Targets/ Cons. Met?
no
yes
Smoke Sweep
Real-Time Operation

Fig. 2. Logical flow chart for static indirect adaptive optimization. Note that while the Jacobian learning step and computation of actuator settings takes place at steady-state intervals, the supervisory control for knock and exhaust temperature mitigation occur in real-time and in parallel to the optimization.

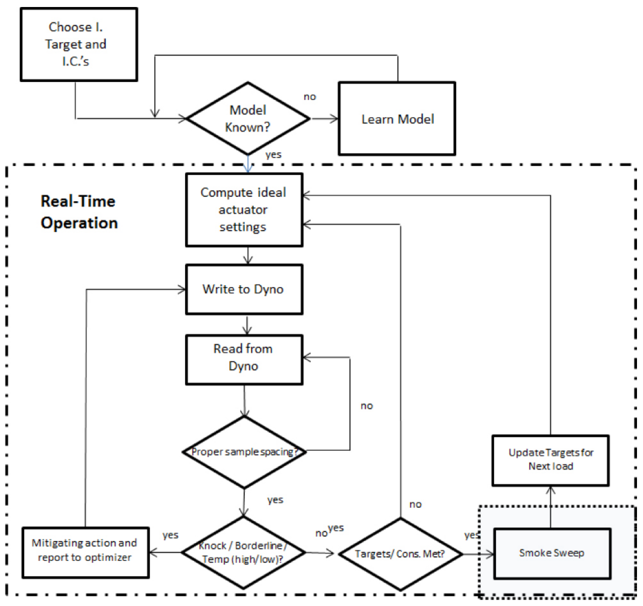
Choose I. Target and I.C.'s
Model Known?
no
Learn Model
yes
Real-Time Operation
Compute ideal actuator settings
Write to Dyno
Read from Dyno
no
Proper sample spacing?
yes
Update Targets for Next load
no
Mitigating action and report to optimizer
yes
Knock / Borderline / Temp (high/low)?
no
yes
Targets/ Cons. Met?
yes
Smoke Sweep

Fig. 3. Logical flow chart for dynamic indirect adaptive optimization. In this architecture, the model and actuator updates occur in near real-time. The supervisory control operates in series with the adaptive algorithm.

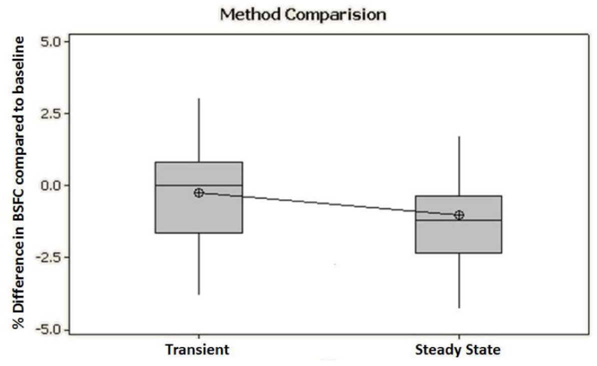

Fig. 4. This plot is a statistical comparison of the minimum break specific fuel consumption found by each of the methods compared to the conventional calibration method. Both data sets are normally distributed. While both methods are comparable to the conventional technique, the static method yields approx. 0.77% better BSFC.

VI. COMPARISON OF METHODS

Using the methodologies described in the previous section both methods were used to map a common set of speed load points on a GTDI engine for which calibration tables generated using the existing sweep based methods are available. For comparison we treat the baseline calibration tables and the resulting break specific fuel consumption (BSFC) as the baseline case, the objective was to minimize BSFC, while maintaining a target load, and constraints on CA50 and combustion stability. We noted that both methods were able to track the targets and maintain CA50, and combustion stability within desired boundaries. Figure VI is a statistical comparison of the found minimum break specific fuel consumption. The zero percent line indicates the specific method has found the same value as the baseline calibration, while a negative percentage indicates that the fuel consumption is lower than baseline. The 95% confidence interval for the difference between the minimum BSFC found by the dynamic and static version of the optimization compared to the baseline is −0.82% to 0.35% and −1.71% to −0.32%, respectively. Specifically, both methods yield acceptable performance in terms of minimum BSFC found, however the static optimization yields approx. 0.77% better minimum BSFC on average than the dynamic variant.

Next, we compare the found actuator settings that resulted in the minimum BSFC for a given speed and load point. Figures 5 and 6, are the required throttle and spark positions found by each optimization algorithm while maintaining the given constraints. The actuator values and trends were similar, since the these actuators are the primary drivers of the load value, CA50 and combustion constraints.

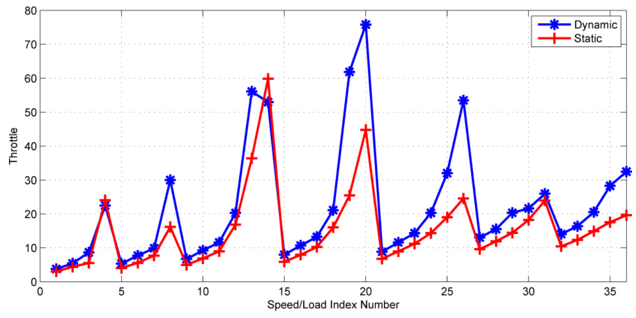

Fig. 5. This plot shows a comparison of found throttle position for each speed load point. Throttle is primarily used to achieve load.

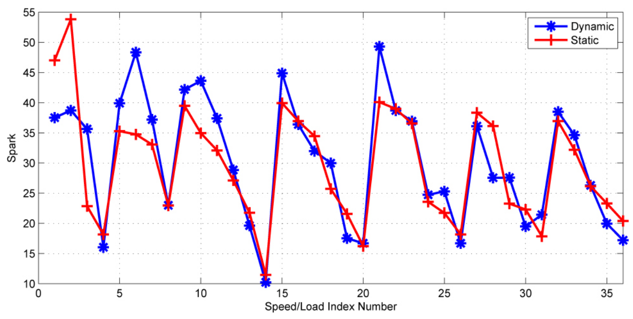

Fig. 6. This plot shows a comparison of found spark position for each speed load point. Spark can also be used to achieve load change, but is primarily used to manage constraints on combustion stability and CA50.

Next, we compare the found CAM index values, each index maps to a physical intake and exhaust cam position. The the CAM index has the greatest impact on BSFC. The found normalized CAM value for each method compared to the baseline calibration is shown in Figure 7.

Next, the found SOI and FRP values are shown in Figures 8 and 9, respectively. Although the sensitivity of BSFC to SOI and FRP are small in comparison to CAM, it was observed that the found that SOI and FRO follow the similar trends to the baseline values. We note that the baseline calibration is not necessarily optimal in a BSFC sense, is typically manually smoothed for vehicle implementation.

Finally, while both algorithms obtain calibration tables much faster than the convectional method, the dynamic variant is approximately 65% faster than its static counterpart, as shown in Figure VI. This leads to a significant reduction of dynamometer resources and faster turnaround time of calibration tables.

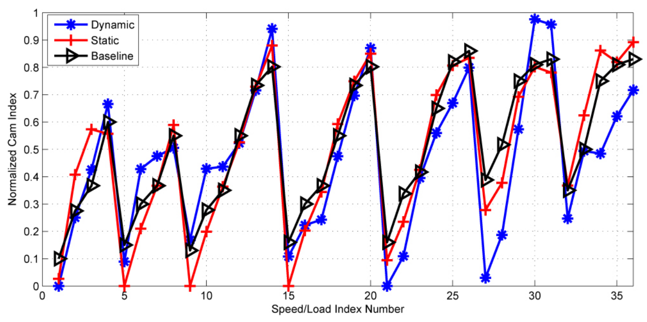

Fig. 7. This chart shows the found normalized CAM index for both the dynamic and static variants of the algorithm. The baseline calibration is also shown for comparison.

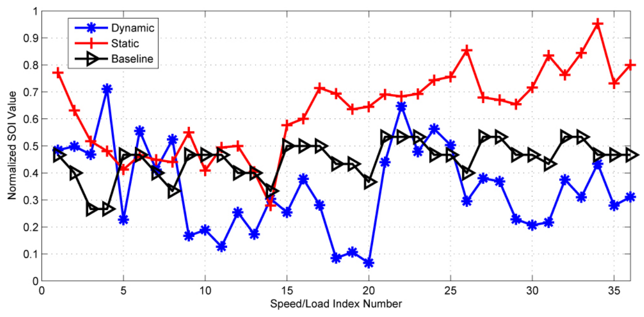

Fig. 8. This plot shows a comparison of found normalized start of injection for each speed load point.

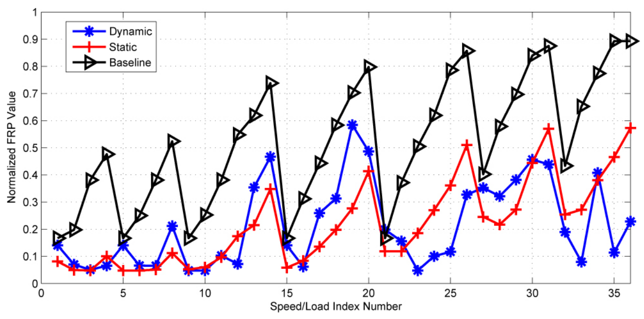

Fig. 9. This plot shows a comparison of found normalized fuel rail pressure for each speed load point.

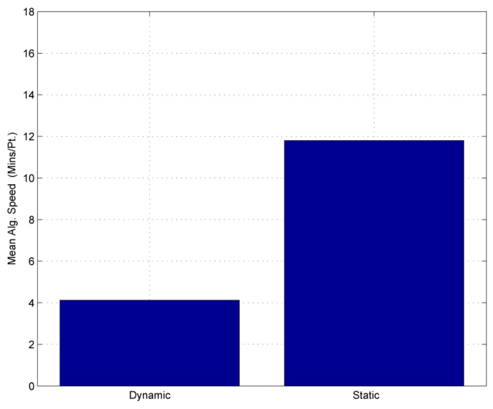

Fig. 10. This plot shows the required dynamometer time in terms of average time required to calibrate each speed/load point. The dynamic variant is 65 % faster than its static counterpart.

VII. CONCLUSION

In this work we demonstrated two indirect adaptive control based optimization strategies for engine mapping and calibration. While the methods share an identical architecture, the key difference is the implementation of the algorithms in steady-state versus dynamic operating conditions. The methods were demonstrated on a GTDI engine with the goal of load and CA50 target following, while minimizing break specific fuel consumption with constraints on combustion stability, exhaust temperature and knock. Both methods were compared against the conventional calibration techniques and showed adequate calibration performance with significantly improved efficiency compared to the conventional calibration technique. The dynamic variant was demonstrated to be approximatively 65% faster than its static counterpart. However, the static algorithm yielded 0.77% on average better break specific fuel economy compared to the dynamic method. Future testing may include a hybridization of the methods, such that while transiting between speed load points, the dynamic variant is used to minimize time, while in the vicinity of the desired speed load point the static method is used to mitigate noise and improve optimization accuracy.

REFERENCES

1. Filev D.P.,Larsson T.,Lixing M. A., “Intelligent Control for Automotive Manufacturing- Rule based Guided Adaptation,” *Proc. of 26th IECON*, pp 283–288,Nagoya, Japan,October2000.
2. Larsson T.,Lixing M. A.,Filev D. P.,“Adaptive Control of a static multiple input multiple output system,”*Proc. of the 2000 American Automatic Control Council*.pp. 2573–2577,Chicago, IL,June2000.

3. Popovic D.,Jankovic M.,Magner S.,Teel A.,"Extremum Seeking Methods for Optimization of Variable Cam Timing Engine Operation," *IEEE Transactions on Control Systems Technology*, Vol 14. No.3, May2006.
4. Wong P.K.,Tam L. M.,Li K.,Wong H. C.,"Automotive engine idle speed control optimization using least squares support vector machine and genetic algorithm,"*International Journal of Intelligent Computing and Cybernetics*, Vol. 1 Issue 4, pp. 598–615,2008.
5. Ljung L., "System Identification Theory for the user," 2nd edition, PTR Prentice Hall2007.
6. D'Amato A. M.,Sumer E. D. andBernstein D. S.,"Frequency-Domain Stability Analysis of Retrospective-Cost Adaptive Control for Systems with Unknown Nonminimum-Phase Zeros," *Proc. Conf. Dec. Contr.*, Orlando, FL,December2011.
7. D'Amato andA. M.,Bernstein D. S.,"Adaptive Forward-Propagation Input Reconstruction for Nonminimum-Phase Systems," *Proc. Amer. Conf. Contr.*,pp.598–603, Montreal, Canada,June2012.
8. Filev D. P.,Larsson T.,Lixing M. A.,"Adaptive Control of Nonlinear MIMO Systems with transport Delay: Conventional, Rule Based or Neural?,"*The Ninith IEEE Inter. Conf. on Fuzzy Systems*.,pp. 587–592,San Antonio, TX, 2000.
9. Siouris G.,"An Engineering Approach to Optimal Control and Estimation Theory,"Wiley Publishing,1996.
10. Zhang C., andOrdonez R.,"Non-gradient extremum seeking control of feedback linearizable systems with application to ABS design," *Proceedings of the 45th IEEE Conference on Decision and Control*, San, Diego, Ca, pp. 6666–6671, 2006.
11. Haskara I.,Ozguner U. andWinkelman J., "Extremum control for optimal operating point determination and set-point optimization via sliding modes," *ASME Journal of Dynamic Systems, Measurement and Control*, pp.719–724,December, 2000.
12. Pan Y., and Ozguner U.,"Extremum seeking control of a variable valve timing engine,"*IFAC Symposium on Advances in Automotive Control*,pp. 173–178,Salerno, Italy,2004.
13. Jankovic M., andMagner S., "Optimization and scheduling for automotive powertrains," *Proceedings of the 2004 American Control Conference*, pp. 4054–4059, 2004.
14. Malikopoulos A., Papalambros P.Y., and Assanis D.N., "A learning algorithm for optimal internal combustion engine calibration in real-time," Proceedings of the ASME 2007 International Design Engineering Technical Conference and Computers and Information in Engineering Conference, IDETC/CIE, Las Vegas, Nevada,2007.
15. Malikopoulos A., Papalambros P.Y., and Assanis D.N., "Real-time, self-learning optimization of diesel engine calibration," Proceedings of 2007 Fall Technical Conference of the ASME Internal Combustion Engine Division, October 14–17, 2007, Charleston, South Carolina, USA, ICEF2007-1603.
16. Cameo Powertrain Calibration Tool, https://www.avl.com/calibration.

ACKNOWLEDGEMENTS AND CONTACT INFORMATION

This work was supported by Ford Motor Company.

A. M. D'Amato, D. P. Filev, Y. Wang and E. Remes are with the Modern Control Methods and Computational Intelligence Group at Ford Motor Company,

adamato2@ford.com