Adaptive Machine Learning for Modeling and Control of Non-Stationary, Near Chaotic Combustion in Real-Time

by

Adam Vaughan

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2015

Doctoral Committee:

Associate Research Scientist Stanislav V. Bohac, Co-Chair
Associate Professor Claus Borgnakke, Co-Chair
Provost Dionissios N. Assanis, Stony Brook University
Professor André L. Boehman
Associate Research Scientist John W. Hoard
Associate Professor Clayton D. Scott

"All models are wrong, but some are useful."

— George E. P. Box

"Chaos: When the present determines the future, but the approximate present does not approximately determine the future."

— Edward N. Lorenz

"Popper claimed that, if a theory is falsifiable (i.e. it can be contradicted by an observation or the outcome of a physical experiment.), then it is scientific. Since prediction is the most falsifiable aspect of science it is also the most scientific one."

— Gianluca Bontempi

© Adam Vaughan

All Rights Reserved

2015

This work is dedicated to the star-stuff from which we all have come.

Acknowledgments

I would like express my gratitude to Jeff Sterniak for pulling the all-nighter that convinced the Bosch® ECU to respond to my engine control messages early in December, 2014, and to Dr. Vijay Janakiraman for providing the raw, unprocessed data that I initially analyzed.

I'm also thankful for my committee members Dr. Stani Bohac, Prof. Claus Borgnakke, Provost Dennis Assanis, Prof. André Boehman, John Hoard, and Prof. Clayton Scott for their time, guidance, and support. Provost Assanis brought me to the University of Michigan, and both Dr. Bohac and Prof. Borgnakke gave me the freedom to explore an interesting thesis topic.

I am pleased to warmly acknowledge all who supported me through thoughtful discussions about my work and my future: Richard and Janis Meyer, Dr. Joshua Lacey, Dr. Joel Forman, Vas Triantopoulos, Prasad Shingne, Prof. Nisar Ahmed, Patrick Gorzelic, Julien Vanier, Alan Mond, Donald Horkheimer, Dr. Luke Hagen, Peter Andruskiewicz, Dr. Ashwin Salvi, Sakthish Sathasivam, Dimitris Assanis, Sam and Laura Olesky, Alex Burnap, Dennis Robertson, Prof. Sotirios Mamalis, Prof. Ben Lawler, Vickey Kalaskar, Jerry Fuschetto, Dr. Orgun Güralp, Dr. Shyam Jade, Dr. Jacob Larimore, Dr. Rob Middleton, Jay Ellis, Dr. Marek Wlodarczyk, Dr. Ilan Gur, and Keith Hughes. I also would like to express appreciation to Tom Marino, William Lim, Bill Kirkpatrick, Melissa McGeorge, Laurie Stoianowski, and Kathie Wolney for their project support, and to gratefully acknowledge the financial support of the United States Department of Energy and Robert Bosch LLC.

Finally, a special thanks to my family, friends, and girlfriend Jess Alper for the love and support that sustained me through this long journey. After all these years of graduate school, I still don't have the words to express my gratitude.

Table of Contents

List of Figures

Figure

List of Tables

Table

List of Symbols and Abbreviations

$\boldsymbol{\beta}$	Output layer weight vector
γ	Adiabatic index
ϵ-SVR	ϵ-Support Vector Regression
θ_{50}	The °CA where 50% of Q_{net} is achieved
θ_a	Start of heat release integration in °CA
θ_b	End of heat release integration in °CA
μ	Mean
σ	Standard deviation
$\sigma(x)$	Logistic function
τ	Time delay until autoignition
Φ	Equivalence ratio
Φ'	Fuel-to-charge ratio
°ATDC	°CA After TDC
°BTDC	°CA Before TDC
°CA	Crank angle degree
$^\top$	Matrix transpose
$\mathbf{o}$	Zero matrix
a	Arrhenius-like activation energy
$\mathbf{a}$	Random input weight matrix (collected from all neurons)
$\mathbf{a}_i$	Neuron i's input weight vector
ADC	Analog to Digital Converter
b	Polytropic exponent
c	Q(x) parameter or (lumped) constant
CA10	The °CA where 10% of Q_{net} is achieved
CA50	The °CA where 50% of Q_{net} is achieved
CA90	The °CA where 90% of Q_{net} is achieved
CAN	Controller Area Network
CV	Cylic Variability
DoE	Design of Experiments
ECU	Engine Control Unit
ELM	Extreme Learning Machine
EVC	Exhaust Valve Close
EVO	Exhaust Valve Open
FIQ	Fast Interrupt reQuest
FLOPS	Floating Point Operations Per Second (double precision)
G	Neuron activation function (logistic function)
GEMM	GEneric Matrix Multiply

GPIO	General Purpose Input-Output (pins)
$\mathbf{H}$	Hidden layer output matrix
HCCI	Homogeneous Charge Compression Ignition
i	Neuron number
IO	Input Output
$\mathbf{I}$	Identity matrix
IMEP	Indicated Mean Effective Pressure
IRQ	Interrupt ReQuest
IVC	Intake Valve Close
IVO	Intake Valve Open
LVDS	Low Voltage Differential Signaling
n	Number of moles of gas, number of input-output data pairs, or time step
$\widetilde{N}$	Number of neurons
$\mathcal{N}(0,1)$	Gaussian distribution with $\mu = 0$ and $\sigma = 1$
NOx	Total nitrogen oxides (NO and NO_2)
NVO	Negative Valve Overlap
$[O_2]$	Oxygen mole fraction
$p(x)$	Padé 3,3 approximant of $exp(x)$
P	In-cylinder pressure
$Q(x)$	Quadratic function
Q_{net}	Mean heat release integral value between θ_b-10 °CA and θ_b
R	Ideal gas constant
R^2	Coefficient of determination
RMSE	Root Mean Squared Error
$s(x)$	Logistic function using the Padé 3,3 $exp(x)$ approximant
SI	Spark Ignition
SOC	Start of Combustion
SOI	Start of Injection
SPI	Serial Peripheral Interface
T	Absolute temperature
$\mathbf{T}$	Target vector
TDC	Top Dead Center
TI	Injection pulse width
u	Mole-specific internal energy
USB	Universal Serial Bus
V	Cylinder volume
VE	Volumetric Efficiency
$\mathbf{W}_0$	Offline data weight matrix or scalar
$\mathbf{W}_1$	Online data weight matrix or scalar
WR-ELM	Weighted Ring - ELM
$\mathbf{x}$	Input matrix
$\mathbf{x}_n$	Input vector at time step n
x_r	Residual gas fraction
z	Number of input variables

Abstract

Fuel efficient Homogeneous Charge Compression Ignition (HCCI) engine combustion phasing predictions must contend with non-linear chemistry, non-linear physics, near chaotic period doubling bifurcation(s), turbulent mixing, model parameters that can drift day-to-day, and air-fuel mixture state information that cannot typically be resolved on a cycle-to-cycle basis, especially during transients.

Unlike many contemporary modeling approaches, this work does not attempt to solve for the myriad of combustion processes that are in practice unobservable in a metal engine. Instead, this work treads closely to physically measurable quantities within the framework of an abstract discrete dynamical system that is explicitly designed to capture many known combustion relationships, without ever explicitly solving for them.

This abstract dynamical system is realized with an Extreme Learning Machine (ELM) that is extended to adapt to the combustion process from cycle-to-cycle with a new Weighted Ring-ELM algorithm. Combined, the above techniques are shown to provide unprecedented cycle-to-cycle predictive capability during transients, near chaotic combustion, and at steady-state, right up to complete misfire. These predictions only require adding an in-cylinder pressure sensor to production engines, which could cost as little as ~$13 per cylinder.

By design, the framework is computationally efficient, and the approach is shown to predict combustion in sub-millisecond real-time using only an iPhone® generation 1 processor (the $35 Raspberry Pi®). This is in stark contrast to supercomputer approaches that model down to the minutiae of individual reactions but have yet to demonstrate such fidelity against cycle-to-cycle experiments. Finally, the feasibility of cycle-to-cycle model predictive control with this real-time framework is demonstrated.

Chapter 1
Introduction

1.1 Preliminaries

Since the 1800s, gasoline engines have largely been operated by (1) controlling power output with a throttle that restricts airflow, (2) using a simple spark to control burn timing, and (3) operating close to fuel-air stoichiometry for reliable spark ignition and so catalysts can reduce NOx emissions (a toxic, smog and acid rain precursor). The throttle hurts fuel efficiency with pumping losses (especially at low-load), and the stoichiometric mixtures used are thermodynamically less fuel efficient than mixtures diluted with air or exhaust gases.

With the broad availability of enabling technologies (e.g. variable valve timing), a relatively new type of combustion called Homogeneous Charge Compression Ignition (HCCI) has received increased research interest over the past decade. HCCI uses autoignition to burn lean (excess air) mixtures and can produce ultra-low NOx quantities that do not require expensive catalyst aftertreatment. Instead of a spark, combustion timing is controlled by the thermodynamic trajectory of the mixture and complex chemical kinetics. With both ultra-low NOx production and freedom from the stoichiometric shackles of spark ignition, HCCI achieves greater fuel efficiency (up to ~20% better [1]) through thermodynamically ideal lean mixtures and unthrottled operation. This improved fuel economy, has real-world relevance to near term sustainability, national oil independence, and greenhouse gas initiatives that seek to curb petroleum usage (see Fig 1.1).

The primary challenge of HCCI autoignition is to ensure that the burn timing is synchronized against the motion of the piston. This is important for efficient extraction of

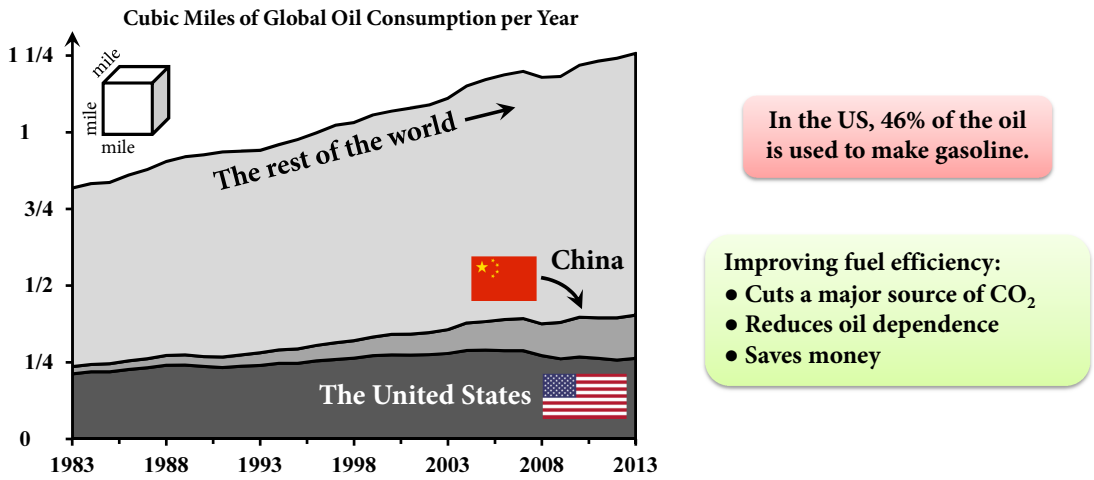


Figure 1.1 Higher fuel efficiency (up to ~20% better [1]) with gasoline HCCI can help reduce the world's increasing petroleum usage. This figure was generated using the raw data available at [2].

mechanical work from the fuel-air mixture and to avoid unsafe, noisy combustion or unstable (near chaotic) combustion oscillations. This synchronization is so important that combustion researchers do not use normal units of time. Instead, they use the angle of the crank, which (1) describes the position of the piston, and (2) represents time because each crank angle takes a certain amount of time at a fixed crank rotation speed.

The thermodynamic trajectory of the air-fuel mixture is driven by the piston varying the cylinder volume as function of crank angle (Fig. 1.2). These angles are measured relative to when the piston is at the top of the cylinder, or Top Dead Center (TDC). In a four-stroke engine, TDC occurs twice per cycle. In different regions, the piston may be compressing or expanding the mixture, or, if a valve is open, moving the mixture into or out of the intake or exhaust manifolds.

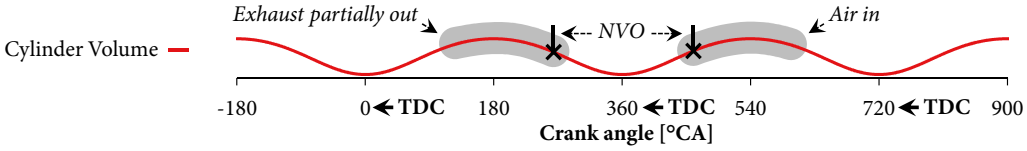


Figure 1.2 Cylinder volume as a function of crank angle.

Highlighted on the curve are two regions, one for when the exhaust valve is open and the other for when the intake valve is open. Note that the two valve events are separated by a

number of crank angle degrees, termed Negative Valve Overlap or NVO. Unlike conventional engines, NVO prevents some of the hot exhaust gases from leaving the cylinder. This stores "residual gases" for the next cycle, offering a practical way to raise the mixture temperature to ensure HCCI autoignition [3]. By changing the amount of NVO, one can affect the mixture temperature and dilution and ultimately control the chemical kinetics behind combustion timing. Temperature and dilution work in opposite directions, but typically temperature dominates [4]. NVO is not instantly adjustable with common variable valve timing systems, and the reader is cautioned that many researchers publish results with fully variable (lift and timing) electric or hydraulic valve actuation systems that are expensive to implement in production engines.

1.2 Primary contribution and existing models

The primary contribution of this dissertation is the ability to experimentally predict large, near chaotic HCCI oscillations from one combustion cycle to the next (also referred to as high Cyclic Variability, or high CV). This is important because these oscillations are one of two main constraints that prevent HCCI from being used over a sizable load range that is practical for production cars [5]. (The other constraint is combustion that is too rapid for safe, low-noise engine operation.)

To understand the significance of high CV predictions, it's important to understand the limitations of existing modeling approaches. To begin, a key fact is that HCCI is very sensitive to the quantity and temperature of residual gases carried over with NVO, and at the same time these quantities are not directly observable with common sensors (e.g. in-cylinder pressure). To see this, consider the ideal gas law $PV = nRT$, where P is pressure, V is cylinder volume, n is molar gas quantity, R is the ideal gas constant, and T is absolute temperature. If we know V from the measured crank angle and P from a sensor, we are limited to just knowing the product of n and T. In general, in-cylinder temperature cannot be directly measured.

It varies spatially (the cylinder wall temperature is different from the core of the gas) and sufficiently fast thermocouple sensors are not physically robust with bead sizes 1/10th the diameter of a human hair.

As of October, 2014, the state-of-the-art, physics-based solution to quantifying residual gases is to use various assumptions about mixture and heat transfer with a robust (but many cycles slow) temperature measurement in the exhaust manifold to back compute the approximate conditions in the cylinder [6, 7]. The validity of this approach across a large range of engine conditions and large transients has yet to be demonstrated, and the solution itself has easily excited oscillatory convergence behavior with high sensitivity to air mass estimation errors.

This difficulty in estimating the mixture state and composition (the temperature and quantities of air and residual gases) explains why low-order physics-based models and even Computational Fluid Dynamics (CFD) with chemical kinetics have yet to demonstrate cycle-to-cycle predictions of near chaotic combustion in experiments. To see how mixture uncertainties impact predictions, Fig. 1.3 shows a state-of-the-art, physics-based model of near chaotic HCCI at steady-state (no transients) intended for feedback control [8]. Specifically, Fig. 1.3 shows θ_{50} (the time when 50% of the air-fuel-residual mixture has burned measured as the angle from TDC, more commonly known as CA50) on a type of plot known as a return map. A return map shows the temporal relationship between the current combustion timing k along the x-axis (abscissa) and the next timing $k+1$ on the y-axis (ordinate). Points near the diagonal are stable (current value is near next value), whereas off diagonal points are oscillatory, or unstable and near chaotic.

The original figure's caption "predictions" is somewhat misleading, because this is in fact a tuned simulation where the cloud of possible combustion timing points is generated by simulating residual gas fraction x_r as a random variable with mean 44.3% and a standard deviation of 0.8%. (Residual gas fraction is the mass fraction of residual in the mixture.) This standard deviation of 0.8% is a very small variation of x_r and is essentially

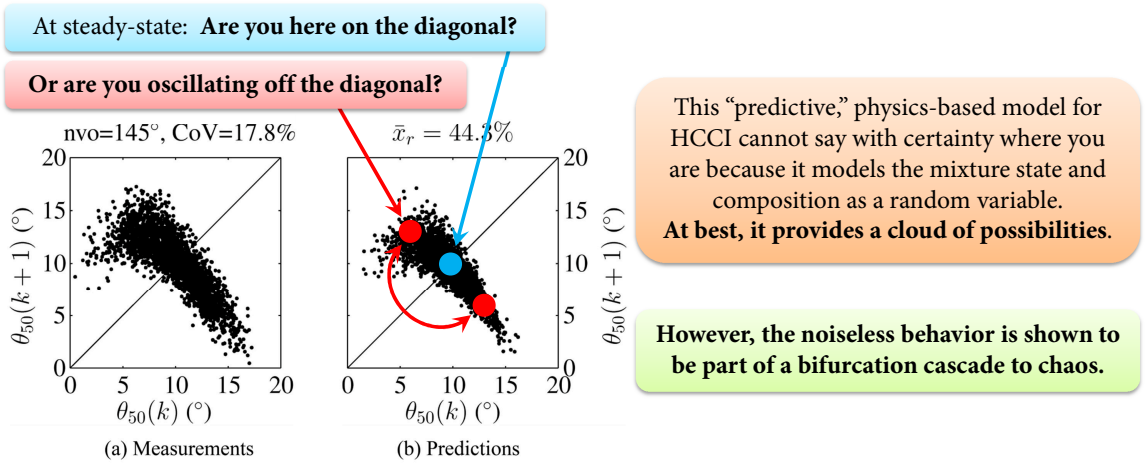


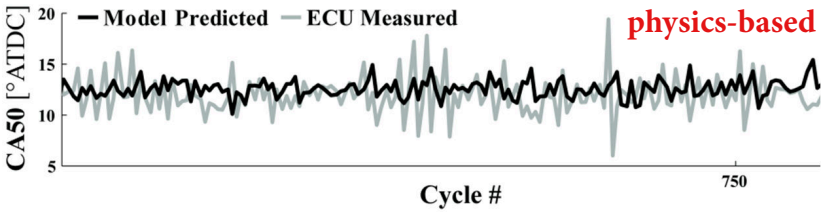
Figure 1.3 Steady-state simulations of combustion phasing at the combustion stability limit with residual fraction noise to capture CV. Adapted from [8] under fair use.

unobservable. Different residual estimation methods can disagree about the actual x_r value by 6% experimentally (Fig. 11 in [6]), and even when tested under tuned, steady-state, noiseless simulation, individual methods disagree about the actual value by up to 2% [9]. As an aside, the ~3.5 crank angle exhaust valve timing step shown for transient validation in Fig. 11 of [6] is small compared to the steps used for the algorithm in this dissertation, which were up to 10x larger (Fig. 5.6).

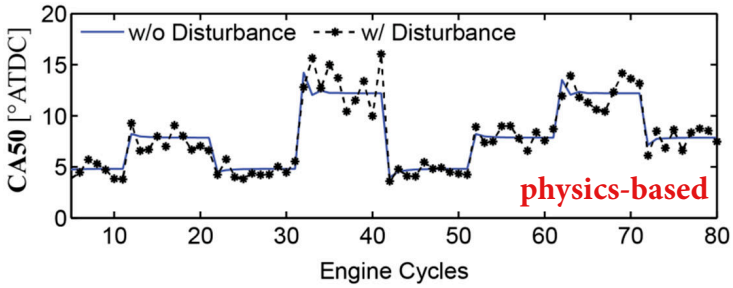
These experimental realities place an upper limit on what's possible with physics-based approaches, and this is evidenced by the fact that even when the model of Fig. 1.3 is combined with a state-of-the-art residual estimation model of [6, 7] unstable HCCI oscillations are not predicted (subfigure ❶ in Fig. 1.4). In fact, this is even after an adaptation routine has been implemented to correct for uncertainties in residual estimation. Finally, while these models are based on physics, in the end they're still curve fits, which has implications for practical engine calibration. For example, for a single engine operating point, Fig. 1.3 requires the tuning of 10 parameters and ultimately only provides a cloud of possibilities. The more advanced, adaptive algorithm (subfigure ❶ in Fig. 1.4) requires tuning of 22 parameters [10] and still doesn't track cycle-to-cycle oscillations beyond the mean value.

One might argue that higher-order approaches that use Computational Fluid Dynamics

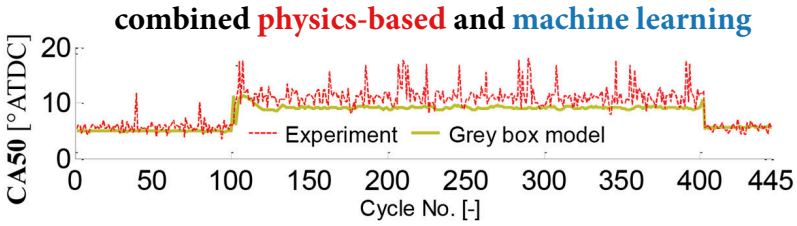
(CFD) and chemical kinetics may be able to predict unstable HCCI oscillations. However, these models are typically only validated to provide *trend-wise* correct average combustion behavior [16], not cycle-to-cycle behavior. They also take day(s) to simulate a single combustion event (which is completely impractical for real-time control). As of September, 2014, the state-of-the-art for understanding for CV with CFD and chemical kinetics (for Reactivity Controlled Compression Ignition (RCCI), which has similarities to HCCI) is to argue that varying the initial conditions will reproduce experimental variability [15] (shown



❶ 2013-2014 control-oriented HCCI model at steady-state with both residual gas estimation and online adaptation.



2013-2014 control-oriented HCCI model simulation where the author adds noise to model the oscillations.



2013-2014 neural network combined with a physics-based, control-oriented model of RCCI with over an octane number step.

Figure 1.4 While these models may reproduce trends and track the mean value, none actually predict cycle-to-cycle oscillations. Adapted from [7, 11, 12] under fair use. Subfigure ❶ is a zoomed view of the vector graphics for cylinder 2 in Fig. 3 of [7] created using Adobe Illustrator®.

in subfigure ❷ in Fig. 1.5). Various reasons are given, but the practice of adjusting boundary conditions to values other than those measured in the experiment so that the model fits experimental data is very common, and is decidedly not a method for a priori predictions. For example, a 2014 Ph.D. thesis states, "In order to replicate the combustion phasing of [HCCI] experiments, the simulations required a constant 10 K increase in intake manifold temperature." [16]

Figs. 1.4 and 1.5 also show that existing machine learning approaches cannot reproduce combustion oscillations, even when combined with physics-based models. The author of subfigure ❸ in Fig. 1.5 calls the oscillations "noise" and adds noise to his model reproduce the

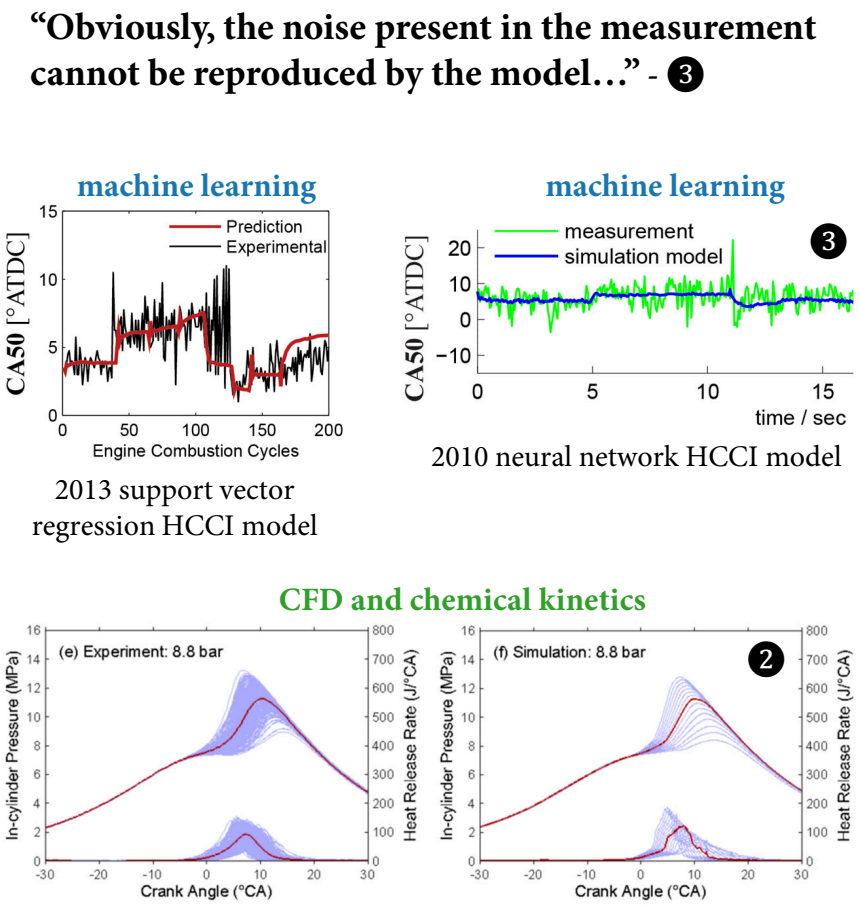


Figure 1.5 While these models may reproduce trends and track the mean value, none actually predict cycle-to-cycle oscillations. Adapted from [13, 14, 15] under fair use.

experimental behavior later in his Ph.D. dissertation. Finally, the only remaining approach is to use joint probability distributions from symbolic statistics [17]. This data-driven method has not been shown to generalize beyond the engine condition for which the data was acquired, and all publications related to this topic do not show cycle-to-cycle predictions of CA50 (although, curiously, [17] does show cycle-to-cycle model error instead).

1.3 Chemistry and chaos

While the previous discussion might lead one to think it is impossible to predict HCCI autoignition, that is actually not the case. Autoignition is very predictable when the full system state is known. In well-controlled rapid compression machine experiments [18, 19], the ignition delay of iso-octane under typical, unboosted HCCI engine conditions can be well defined* using the global mixture state:

$$\tau \propto P^{-1.05}\, \Phi^{-0.77}\, [O_2]^{-1.41}\, EXP\left(\frac{a}{RT}\right) \tag{1.1}$$

where τ is the time delay until the mixture autoignites, P is pressure, Φ is the equivalence ratio, $[O_2]$ is the oxygen mole fraction, R is the ideal gas constant, T is absolute temperature and a is an Arrhenius-like activation energy.†

Yet despite the relatively simple functional form for individual iso-octane autoignition events given by Eq. 1.1,‡ it is well known that complex combustion stability issues pervade practical HCCI implementations using iso-octane and other fuels [21, 22]. For iso-octane and gasoline-like fuel blends, the stability limit behavior is characterized by large cycle-to-cycle

---

* He et al. achieved a 0.98 coefficient of determination (R^2) for Eq. 1.1 [18].

† The value a is termed "Arrhenius-like" because it is for an ignition delay curve fit and not for an isolated reaction rate. Additionally, a can vary significantly depending on the region of interest [19].

‡ Even when combined with an autoignition integral to account for the full trajectory of thermodynamic states imposed by the piston's motion [20].

oscillations or high CV prior to misfire [23].[§] While a number of possible explanations have been suggested to understand high CV oscillations (cf. [26]), one of the most fascinating observations has been that high CV can be viewed as part of a cascade of near chaotic system bifurcations that have sensitive dependence on the engine set point [27], similar to what is seen in lean limit and high residual Spark Ignition (SI) engines [28, 29]. A brief introduction to bifurcation theory as it relates to HCCI combustion is provided for the reader in Sec. 2.1.

1.4 Motivation and goals

In light of the above, the author contends that one of the primary challenges facing practical, gasoline based HCCI implementations is not so much the existence of complex bifurcation behavior with sensitive dependence on state and parameters, but that new methods are needed to capture the global mixture state and autoignition characteristics on a cycle-to-cycle basis so that near chaotic combustion instabilities can be predicted and corrected for a priori. In particular, the ability to accurately control and reduce combustion phasing oscillations can potentially enable the use of late phased combustion to mitigate the excessive pressure rise rates that currently constrain HCCI's high-load operation [30], while also addressing the high CV experienced at low-load. The first step in this endeavor is to accurately predict combustion phasing across a wide variety of transients, operating points with high CV and at steady-state from cycle-to-cycle, and the final step is to use these predictions to intelligently control the engine on a cycle-to-cycle basis. The above views are supported by the facts that individual autoignition events can be fairly deterministic (e.g. Eq. 1.1) and even simple deterministic

[§] As shown in [23], the exact HCCI stability limit behavior does vary between different fuel blends. That said, the general behavior of iso-octane can be considered representative of the typical, unboosted behavior of gasoline [24, 21, 23]. Additionally, while Eq. 1.1 captures the behavior of iso-octane under typical, unboosted HCCI conditions, it is worth highlighting that the behavior across a larger range of conditions is substantially more complex than Eq. 1.1 suggests (see [25]). In fact, as noted earlier, the behaviors of both iso-octane and gasoline-like mixtures are sufficiently complex that the full chemical kinetics (cf. [25]) are computationally infeasible on modern engine controllers. Additionally, even if one could perform the chemical kinetics cycle-to-cycle on a real-time engine controller, there remains the subproblem of determining the exact mixture state and composition.

functions can experience the sudden onset of bifurcation behavior when iterated (see Sec. 2.1).

1.5 A new approach to engine combustion

The following chapters will detail a new machine learning approach to HCCI combustion phasing predictions that not only works with steady-state data, but also works with transient, near chaotic experimental (not simulation) data (Fig. 1.6). The approach is able to adapt to new engine conditions in real-time on low-cost hardware, and it differs substantially from existing models that have not been shown to track near chaotic high CV outside of simulated mean behavior (see comparison in Sec. 1.2). This new methodology enables a new class of predictive control strategies for complex, high CV HCCI combustion (Fig. 1.7), and offers an alternative to the common ways we understand and control engine combustion today, possibly beyond HCCI (Fig. 1.8).

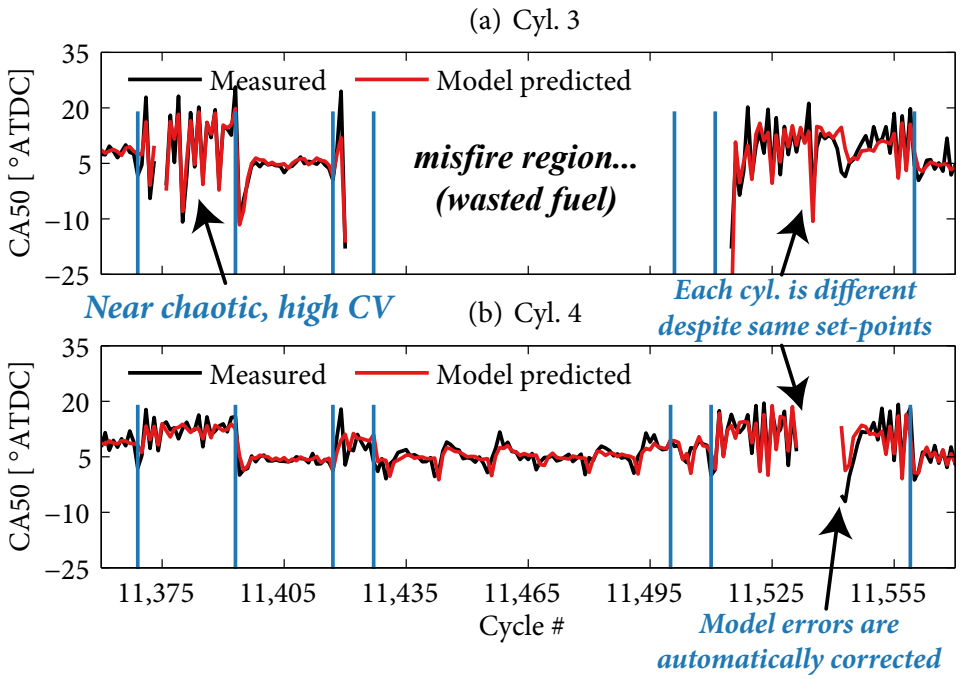


Figure 1.6 Multi-cylinder predictions with the techniques developed in this dissertation. Each blue vertical line is a harsh, simultaneous transient of four different engine actuators (see Fig. 5.6 for detailed information about the actuator settings).

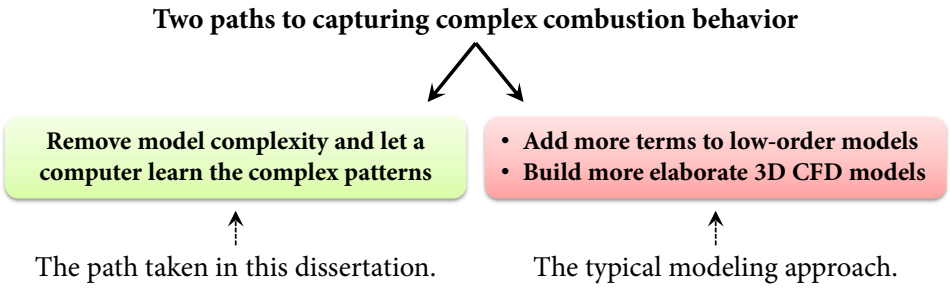
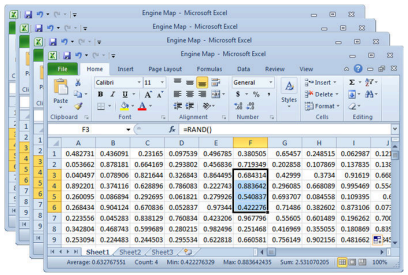


Figure 1.7 A high-level, conceptual view of how this dissertation differs from typical approaches to modeling complex combustion behavior.

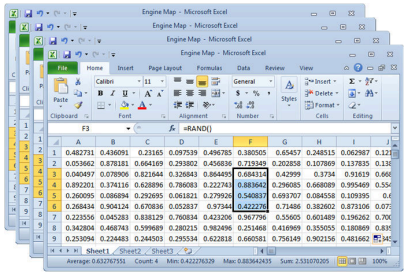
❶ Typical Engine Control

Microsoft Excel®-like lookup tables

- Steady-state tuned maps
- Varying PID loop gains



- Tables for tables (e.g. trims)



❷ Two-State, Physics-Based Model for HCCI Control

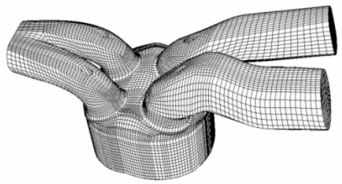
White-box curve-fitting, physical states are not directly measured, 22 parameters, needs adaptation

$$K_{th}(\theta_{soi}) = k_0 - u_{soi}$$
$$= \int_{\theta_{ivc}}^{\theta_{ivc}(k-1)} \frac{A}{\omega} p_c(\theta)^{n_{ht}} \exp\left(\frac{B}{T_c(\theta)}\right) d\theta$$
$$p_c(\theta) = p_{ivc}(k-1)\left(\frac{V_{ivc}}{V(\theta)}\right)^{n}$$
$$T_c(\theta) = T_{ivc}(k-1)\left(\frac{V_{ivc}}{V(\theta)}\right)^{n-1}$$
$$\theta_{50}(k-1) = b_1\theta_{soi}(k-1) + b_0.$$
$$\theta_{50}(k-1) = \alpha_{q1}(soi)T_{ivc}^2(k-1) + \alpha_{q2}(soi)T_{ivc}(k-1) + \alpha_{q3}(soi)$$
$$T_{ac}(k-1) = T_{ivc}(k-1)\left(\frac{V_{ivc}}{V_c}\right)^{n-1} + \Delta T(k-1)$$
$$\Delta T(k-1) = \eta_{ht}(k-1)\frac{q_{hv}m_f(k-1)}{c_v(k-1)m_c(k-1)}$$
$$= \frac{\eta_{ht}(k-1)q_{hv}R}{c_v(k-1)p_{ivc}V_{ivc}} m_f(k-1)T_{ivc}(k-1)$$
$$\eta_{ht}(k-1) = \frac{a_0}{1+\exp\left\{\frac{\theta_{50}(k-1)-a_3}{a_2}\right\}}(1+a_5\omega(k-1))$$
$$c_v(k-1) = 1 + a_5 b_v(k-1)$$
$$T_{exc}(k-1) = T_{ac}(k-1)\left(\frac{V_c}{V_{exc}}\right)^{n-1}$$
$$= T_{ivc}(k-1)\left(\frac{V_{ivc}}{V_{exc}}\right)^{n-1}$$
$$\left[1 + \frac{\eta_{ht}(k-1)q_{hv}RV_c^{n-1}}{c_v p_{ivc}V_{ivc}^{\,n}} m_f(k-1)\right]$$
$$p_{exc}(k-1) = p_{ivc}(k-1)\frac{V_{ivc}}{V_{exc}}\frac{T_{exc}(k-1)}{T_{ivc}(k-1)}$$
$$= p_{ivc}(k-1)\left(\frac{V_{ivc}}{V_{exc}}\right)^{n}$$
$$\left[1 + \frac{\eta_{ht}(k-1)q_{hv}RV_c^{n-1}}{c_v p_{ivc}V_{ivc}^{\,n}} m_f(k-1)\right]$$
$$x_r(k-1) = 1 - (c_0 + c_1\theta_{exc})$$
$$\Pi^{c_2}T_{ad}(k-1)^{c_3}\omega(k-1)^{c_4}$$
where $\Pi = \frac{p_{ivc}(k-1)}{p_{exc}(k-1)}$
$$T_{exc}(k) = c_v T_{ad}(k)$$
$$T_{soc}(k) = T_{exc}(k)\left(\frac{V_{exc}}{V_{soi}}\right)^{n-1} - a_4 m_f(k-1)$$
$$T_{res}(k) = T_{soi}(k)\left(\frac{V_{soi}}{V_{exc}}\right)^{n-1}$$
$$T_{exc}(k) = \left[c_v T_{ad}(k)\left(\frac{V_{exc}}{V_{soi}}\right)^{n-1} - a_4 m_f(k-1)\right]\left(\frac{V_{soi}}{V_{exc}}\right)^{n-1}$$
$$T_{ivc}(k-1) = x_r(k-1)T_{exc}(k-1) + (1-x_r(k-1))T_{im}.$$
$$b_v(k-1) = x_r(k-1)b_{ad}(k-1)$$
$$b_{ad}(k+1) = \frac{(AFR_s+1)m_f(k-1)}{m_c(k-1)} + b_v(k-1)$$
$$= \frac{(AFR_s+1)R}{p_{exc}V_{exc}} T_{ivc}(k-1)m_f(k-1) + b_v(k-1)$$

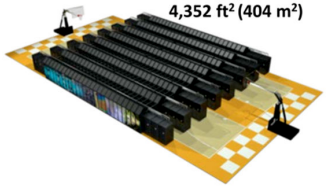
❸ CFD + Chemical Kinetics

Discretize Navier-Stokes eqs. for up to ~700k mesh elements, select a favorite turbulence model, choose ~100 to ~6,000 reactions

2014 PhD thesis: "...model was capable of predicting **trend-wise agreement** with metal engine..."



2014, 8.2 megawatt supercomputer for dilute SI combustion stability: "**Changing the random number [generator] seed produces... significant variation in predicted performance.**"



Machine Learning

"Black-box curve-fitting"

This dissertation:

- **0.000001 megawatts**
- **0.0005 seconds** to calculate on iPhone® gen. 1 processor
- **~20 min. to learn near chaos**
- Can **adapt and control** an engine in **real-time**
  - 1 main parameter, 2 sensors
  - Robust across multiple engines
  - Analytically differentiable

New possibilities:

Previous machine learning and ❶ ❷ ❸ approaches have only been shown to track the average behavior of experiments. **None** predict near-chaotic HCCI in cycle-to-cycle experiments.

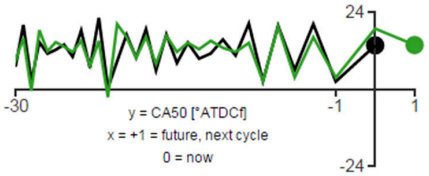


Figure 1.8 The model shown under ❷ is from [7, 10]. The quotes and images under ❸ are from [16, 31] under fair use.

1.6 Document organization

- Chapter 2 provides a short introduction to both bifurcation theory and machine learning.
- Chapter 3 proposes an abstract mapping function abstraction for the cycle-to-cycle time evolution of HCCI combustion.
- Chapter 4 derives a new online adaptive machine learning algorithm and provides details of its application to fitting the abstract mapping function.
- Chapter 5 describes the experimental setup, details extensive cycle-to-cycle experimental observations, and demonstrates the model performance on a subset of those datasets.
- Chapter 6 explores model parameter sensitivity and extrapolation information with existing datasets.
- Chapter 7 details a custom low-cost, platform that implements the dissertation framework in real-time.
- Chapter 8 demonstrates that the framework is feasible for real-time engine control.
- Chapter 9 summarizes the dissertation and provides directions for future work.

Chapter 2
Bifurcation Theory and Machine Learning

2.1 Concepts from bifurcation theory

To understand what a bifurcation entails, we begin by defining a dynamical system. A dynamical system describes the time evolution of a system using some fixed rule. Dynamical systems are typically encountered in the form of differential equations wherein a fixed relation amongst derivatives describes how a system evolves in time. A discrete dynamical system can sometimes be constructed to provide a simpler understanding of a periodic continuous system. When such a discrete system realization is possible, the time evolution of the continuous system can be reduced to an iterative function that takes a periodic point in the continuous case and maps it to a subsequent period (similar to a Poincaré map, although a Poincaré map deals with the evolution on phase space sections without mention of time explicitly). [32, 33]

In the context of an HCCI engine that uses NVO to carry a sizable fraction of residual gases (typically 20-60% [34, 9]) from the previous cycle to promote the next cycle's combustion, this fixed iterative rule can be thought of as a function that maps some measure(s) of the state of one combustion event to the next:

$$\text{current combustion} = \text{function}\big(\text{ previous combustion, parameters}\big) \tag{2.1}$$

While this hypothetical mapping function compactly summarizes how one might view NVO HCCI within the framework of a discrete dynamical system, the function itself must be specified, parameters may vary cycle-to-cycle, and there are stochastic components that are likely to be present.

As an example of a simple mapping function that exhibits many common behaviors of discrete dynamical systems, consider the quadratic function $Q(x) = c - x^2$ with c as a parameter [32]. In this case, $Q(x)$ is the fixed rule used to map the current state x_n to the next state x_{n+1}:

$$x_{n+1} = Q(x_n) = c - (x_n)^2 \tag{2.2}$$

If one starts to iterate Eq. 2.2 with $x_0 = 0.0$ and $c = 0.25$:

$$\begin{aligned} 0.25 &= Q(0.00) \\ 0.19 &= Q(0.25) \\ 0.21 &= Q(0.19) \\ 0.20 &= Q(0.21) \\ 0.21 &= Q(0.20) \\ 0.21 &= Q(0.21) \\ &\vdots \end{aligned} \tag{2.3}$$

the function value converges to 0.21, which is an attracting fixed point of the system for the chosen parameter. If the c parameter is gradually increased to 1.0, the fixed point moves and ultimately bifurcates into two recurring values:

$$\begin{aligned} 1.0 &= Q(0.0) \\ 0.0 &= Q(1.0) \\ &\vdots \end{aligned} \tag{2.4}$$

This oscillatory behavior is called a period 2 cycle. To illustrate this behavior, Fig. 2.1 shows the final function value(s) in the limit of a large number of iterations as c is varied. It's clear that the period doubling bifurcation that yielded the period 2 cycle occurred abruptly as c was increased past $3/4$. The bifurcation is thus a significant change in the solution's behavior for

a small change in the system's parameter. It is also the first of an entire cascade of bifurcations that ultimately lead to chaos as c moves towards the value 2. This cascade is a characteristic of many mapping functions (including low-order models of HCCI [8]), although the particulars of the cascade typically vary from function-to-function [35]. Additionally, it must be stressed that while the same fixed rule is still operating from iteration-to-iteration in the chaotic region of Fig. 2.1, the system has become very sensitive to both the parameter and the starting point. This extreme sensitivity despite completely deterministic behavior is the hallmark of a chaotic system. As noted earlier, HCCI autoignition can be very deterministic* and, as will be discussed later, HCCI has the visible markings of a bifurcating system.

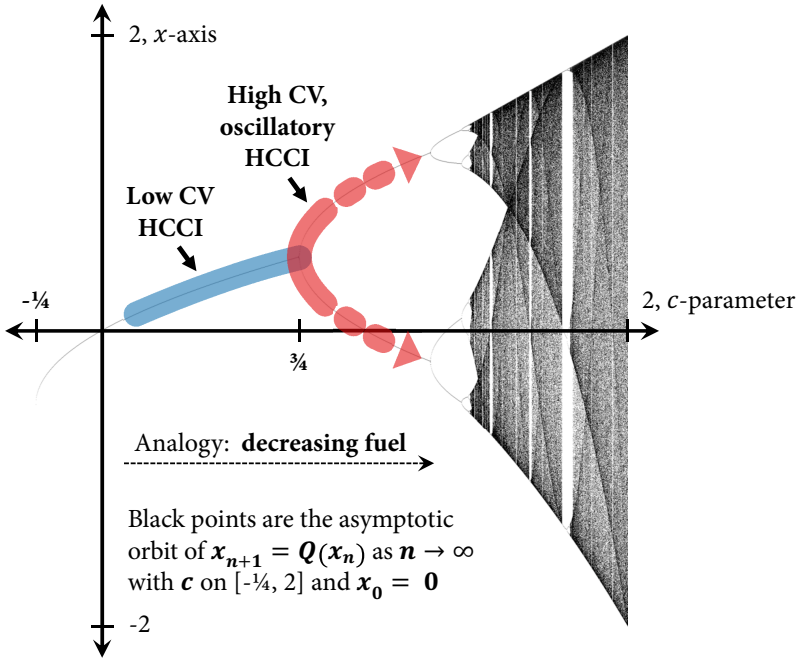


Figure 2.1 The bifurcation diagram of $Q(x)$ as c is varied, along with a qualitative representation of how one might view high CV HCCI.

Despite the oscillatory behavior for $c = 1.0$, a fixed point is still given by the solution of $x = Q(x)$, with one of the roots being $\frac{1}{2}(\sqrt{5}-1)$. However, this fixed point cannot be "seen" because it no longer attracts nearby points. Thus, even if one started with a point close to the

* He et al. achieved a 0.98 coefficient of determination (R^2) for Eq. 1.1 [18].

exact root $x = 0.61803...$, the system would still be attracted to the period 2 cycle:

$$\begin{aligned} 0.64 &= Q(0.60) \\ 0.59 &= Q(0.64) \\ 0.65 &= Q(0.59) \\ &\vdots \\ 0.00 &= Q(1.00) \\ 1.00 &= Q(0.00) \end{aligned} \tag{2.5}$$

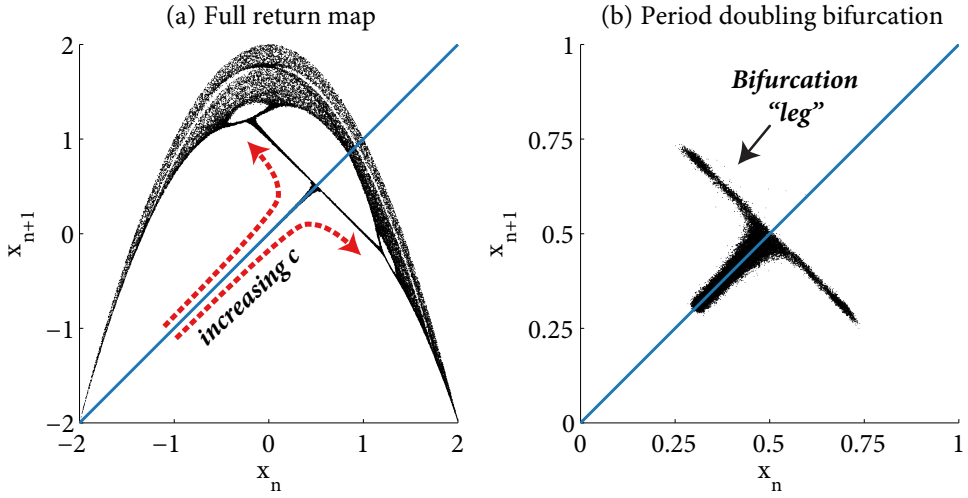


Figure 2.2 (a) The return map of $Q(x)$ as c is increased. (b) A zoomed view of the bifurcation that occurs as c is swept past $3/4$.

Finally, return maps provide another useful tool to visualize iteration-to-iteration behavior, however they can do so without explicit knowledge of a system's parameters. A return map is generated by plotting a function's current value along the x-axis and the function's next value on the y-axis. Fig. 2.2 shows $Q(x)$'s return map as c is varied with added Gaussian noise that might be present in an experiment. If a point is near the blue diagonal $y = x$, the next value is approximately equal to its current value. If a point is off the diagonal, it means the system has some additional periodic structure. For brevity, the reader is referred to [32, 33, 36] for more rigorous discussions on the behaviors seen in bifurcating dynamical systems.

2.2 Machine learning

Over the past few decades there has been considerable progress in the use of machine learning to perform non-linear regression of unknown functions [37]. There are various algorithms used in machine learning regression, but in general they take data and attempt to find a functional mapping between inputs and outputs without a predefined analytical relationship between those inputs and outputs. In the context of a discrete dynamical system, one might use one of these algorithms to find the mapping function between iterations in Eqs. 2.1 or 2.2.

This machine learning approach stands in contrast to existing CFD with chemical kinetics by being very computationally efficient. While CFD and chemical kinetics can capture a great deal of combustion chemistry (see [38] and gasoline mechanism validation [39]), their simulation time of a ~50 millisecond engine cycle is typically measured in day(s) for a single core of a modern computer.

At the other computational complexity extreme, low-order approximation models of HCCI for control have been developed since at least the early 2000's [40] based off spark ignition engine knock models developed in the 1950's [20]. Recently, efforts have been made to extend such autoignition models to high CV regions of HCCI by injecting random residual noise in simulation and observing similar cycle-to-cycle dynamics as experiments [8]. Such CV models are tuned to a handful of engine conditions and are only "predictive" in the sense that when tuned with random noise they simulate the general return map shape seen in actual engine experiments (see Sec. 1.2).

That said, for a restricted set of conditions, these methods have been useful for theoretically showing that a period doubling cascade to chaos underlies the observed high CV behavior [8]. It should be noted that an extension of the model described in [8] has also been developed with online residual estimation and online adaptation [7], however it has not been demonstrated to capture CV beyond the mean value (see subfigure ❶ in Fig. 1.4).

Thus, the machine learning approach is intended to fill the prediction gap between high fidelity chemical kinetics and low-order models, ***and*** avoid the air-fuel-residual mixture state

estimation needed for both of those methods. Additionally, the approach in this thesis avoids the need to calibrate sub-models that explicitly capture the non-linear chemistry, non-linear physics, the high engine parameter sensitivity during period doubling bifurcation(s), etc.

While there are some clear benefits to the machine learning approach, a key issue is that machine learning is data driven, and relatively large quantities of data are needed to adequately cover large dimensional spaces. As shown conceptually in Fig. 2.3, these high dimensional data might be viewed as a "porcupine" [37]. Each engine operating condition might be viewed as a "quill" of Eq. 3.14's six-dimensional "porcupine," and machine learning algorithms know nothing about the ideal gas law or chemical kinetics, so their ability to extrapolate between "quills" is limited, especially when provided sparse data. The author's previous work in [41] used a random sampling of cycle time series for training to ensure the data driven model had data to fit the "quills," and then assessed the model's ability to predict on the remaining (randomly chosen) cycles. Thus, the training dataset was partially acausual and that the model itself wasn't shown to adapt to new conditions.

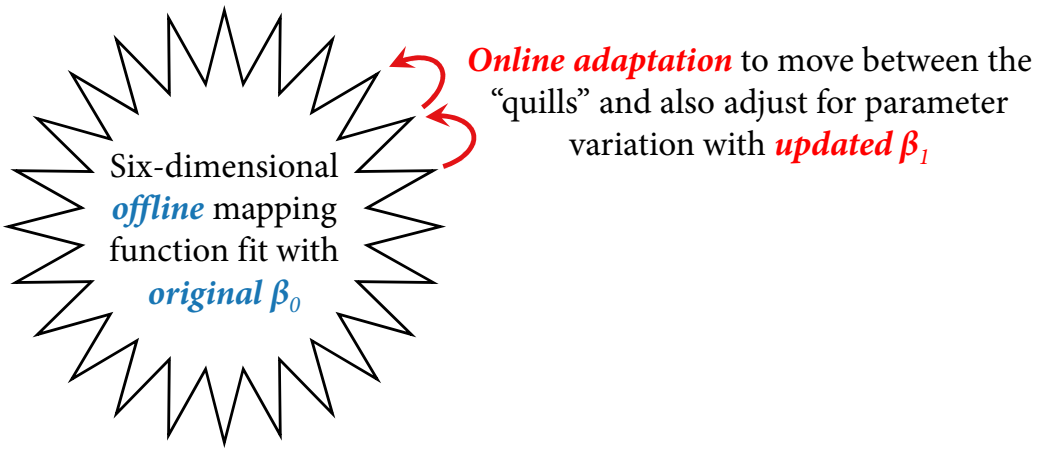


Figure 2.3 High dimensional data (such as the six dimensional input to Eq. 3.14) might be viewed conceptually as a "porcupine" [37]. A primary goal of this work is to design an adaptive algorithm to fit online data between the "quills" while also maintaining good generalization by using a weighted balance of online data to offline data "quills."

A contribution of this work is the development of a new online learning method to provide real-time adaptive, fully causal predictions of HCCI combustion phasing. This method, called

Weighted Ring - Extreme Learning Machine (WR-ELM), enables online extrapolation of an offline Extreme Learning Machine (ELM) model that is trained to fit the "quills" of offline data. WR-ELM was originally proposed by the author in the October, 2013 paper [42].

Chapter 3
Mapping Function

3.1 Selection of an HCCI mapping variable

In this chapter, we consider the selection of a suitable mapping variable for NVO HCCI engine combustion. Ideally, this mapping variable would be a scalar that quantifies the previous cycle's influence on the next cycle. This variable should also be experimentally measurable during transients and at operating points with high CV. A number of variables have been studied to understand cycle-to-cycle behavior in HCCI in prior work, such as net heat release [27], CA50 [17], and in-cylinder temperature [28]. Of these, temperature is perhaps the most meaningful because of its exponential influence in Eq. 1.1.

Unfortunately, temperature is difficult to measure with in-cylinder pressure transducers because it requires knowledge of the total number of moles in the cylinder (to use the ideal gas law), which in turn requires a model for both the temperature and quantity of residual carried over from the previous cycle along with an engine breathing model for Volumetric Efficiency (VE). Thus the measurement is both indirect and recursively based on the previous cycle's temperature. Moreover, if the models for residual and VE are parameterized at steady-state, they may not be valid during transients and operating points with high CV. Any errors in temperature are also exponentially amplified in equations such as Eq. 1.1, and those exponential errors are then integrated with additional exponential relations for polytropic compression in an autoignition integral. In fact, a $\pm 1\%$ error in absolute temperature at IVC can cause approximately ± 2.5 °CA of error in a start of combustion for a state-of-the-art physics based, control-oriented combustion model at a single operating point with steady-

state late phasing (see Fig. 2.3 in [43]). Finally, there are also wall temperature effects that can significantly affect autoignition [44, 30] and the fact that unburned fuel and trace species in residual gases such as formaldehyde influence autoignition in addition to temperature [45, 23, 46].

As a way around these issues with temperature, the author has chosen to focus on directly measurable characteristics of the burn itself. In particular, the timing of the burn is the most important variable when using late phased combustion as a method to control excessive pressure rise rates [30]. Although CA50 is typically used to quantify combustion phasing, CA90 is used in this work because it is a stronger indicator of bifurcation dynamics on a return map (see Fig. 5.2 in Sec. 5.3). CA90 and CA50 are related quantities, and in terms of model fit statistics there was no significant benefit of using one over the other. That said, a few isolated instances were observed where CA90 did a better job predicting large oscillations at the high CV stability limit.

CA90 represents the timing of the tail end of the burn in relation to the window of high temperatures around Top Dead Center (TDC) before any remaining reactions that could potentially influence residual gas species are frozen by expansion from the piston's motion. It captures both late phased burns that are often associated with high CV and early phased burns that have issues with excessive pressure rise rates. Thus, the author views CA90 as both an indirect, inverse measure of temperature (since high temperature burns will go to completion faster) and a measure of the trace species present because late phased burns typically have poor combustion efficiency. Finally, the 90% burn point is chosen because it is typically the largest heat release fraction that is well defined against experimental noise and pressure fluctuations in the nonlinear tail of a heat release curve [47].

3.2 Heat release model

To compute CA50 and CA90 on a cycle-to-cycle basis, a single zone net heat release (neglecting heat transfer) model was used:

$$Q_{net} = \int_{\theta_a}^{\theta_b} \frac{\gamma}{\gamma - 1} P \, dV + \int_{\theta_a}^{\theta_b} \frac{1}{\gamma - 1} V \, dP \tag{3.1}$$

where Q_{net} is the net heat release, P is either 3-point smoothed in-cylinder pressure or zero phase, 8th order Butterworth 3.5 kHz filtered P,[*] V is cylinder volume from crank slider calculations, $\gamma = 1.305$, dP is computed using a 5-point central difference, dV is from the analytical derivative, and integration is performed using the trapezoidal method. The crank angles θ_a and θ_b and other key variables are shown in Fig. 3.1. Because of experimental noise and pressure fluctuations, the final value of Q_{net} was taken to be the mean integral value between 50 and 60 degrees After TDC (°ATDC). Based on a separate heat release sensitivity study, a compression ratio of 11.0 (instead of the geometric compression ratio of 11.2) was selected for volume calculations.

Although heat transfer can account for ~20% of gross heat release under typical HCCI conditions [48], it is neglected here because it requires measurements that may not be valid for transients and operating points with high CV. These measurements include in-cylinder temperature of both the wall and gas along with heat transfer correlations that were developed for steady-state engine operation. A constant γ is used in Eq. 3.1 because it does not require tracking the in-cylinder temperature and composition, and because allowing γ to float with the measured polytropic exponents cycle-to-cycle can make heat release calculations sensitive to pressure pegging errors [47]. This particular constant γ value was chosen in the spirit of the Rassweiler and Withrow model described in [47] after observing the similarity of the cycle-to-cycle polytropic exponent means before and after combustion. Specifically, the

[*] Both filtering approaches produced similar combustion phasing results for the data described in this work. To reduce computational complexity, the final real-time implementation uses 3-point smoothed pressure.

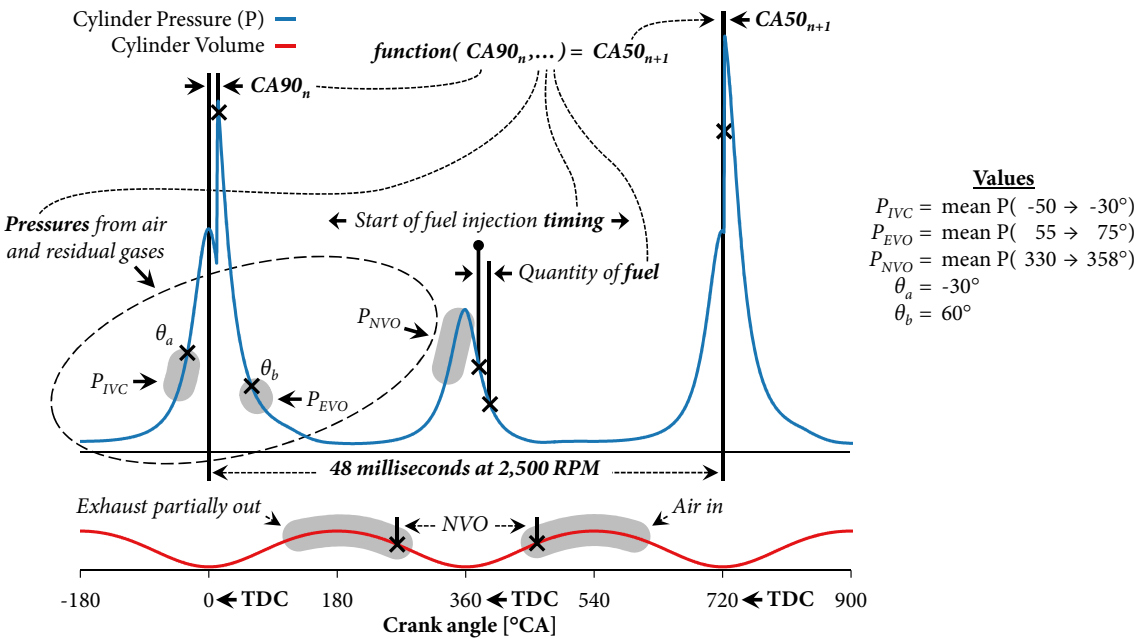


Figure 3.1 A schematic of key engine cycle variables. The timing of both P_{IVC} and P_{NVO} have been modified from the original version in [41] to meet real-time engine controller timing requirements. P_{IVC} was moved to the previous cycle and P_{NVO}'s crank angle range was shortened. P_{IVC} has also been moved closer to TDC to take advantage of the inherent signal amplification provided by the compression process.

polytropic exponent 50 to 30 degrees Before TDC (°BTDC) had a mean (μ) of 1.300 with a standard deviation (σ) of 0.017, and the polytropic exponent 60 to 80 °ATDC had a μ of 1.308 with a σ of 0.033 across all four cylinders over the full range of randomized actuator set points detailed in Sec. 5.1.

While inaccuracy is introduced with the above heat release simplifications, it is felt that uniformly applying the above method still provides relative combustion timing information without introducing parameterizations that may not be valid for transients and operating points with high CV. Additionally, outlier criteria are provided in Sec 5.2 to ensure that the net heat release has a achieved a stable value over the range the Q_{net} mean is evaluated to avoid the experimental uncertainty in determining the end of combustion. These criteria are fairly permissive, with only 3% of the cycles removed. The reader is referred to [49, 48, 47] for more thorough discussions on the difficulties of computing heat release at steady-state, without the added complexity of transients or high CV.

3.3 Abstract mapping function

The goal of this section is not to develop explicit relations, but rather to derive an abstract, skeletal functional form for CA90 that is based on known models such as Eq. 1.1. This functional dependence will be used later in conjunction with the machine learning technique described in Chapter 4 to compute a cycle-to-cycle mapping function. Please refer to Fig. 3.1 for a schematic overview of the various measurements along with the specific crank angles referred to by valve timing subscripts.

Cycle-to-cycle air measurements

Consider an idealized, adiabatic intake process between Intake Valve Open (IVO) and Intake Valve Close (IVC) where work effects are neglected, n is the moles of gas in the cylinder, and $u(T)$ is the molar specific heat as a function of temperature:

$$\begin{aligned} n_{residual} \cdot [\, u_{residual}(T_{IVO}) - u_{residual}(T_{IVC}) \,] \\ = n_{air} \cdot [\, u_{air}(T_{IVC}) - u_{air}(T_{intake}) \,] \end{aligned} \tag{3.2}$$

If one assumes equal specific heats, collects the terms, notes that $n_{IVC} = n_{residual} + n_{air}$, redefines $n_{IVO} = n_{residual}$ and incorporates ideal gas law to solve for n_{air}, one has:

$$n_{air} = \frac{P_{IVC} V_{IVC} - P_{IVO} V_{IVO}}{R T_{intake}} \tag{3.3}$$

While equal mean specific heats are used here to further simplify the equation, the reader should note that this is not truly valid given the CO_2 and H_2O content in residual. Still, assuming T_{intake}, V_{IVO} and V_{IVC} are known along with the other assumptions above, the equation can be generalized to:

$$n_{air} = \text{function}(\, P_{IVO},\, P_{IVC}) \tag{3.4}$$

Cycle-to-cycle residual measurements

Unfortunately, directly computing residual using in-cylinder pressure is not possible without a second intensive thermodynamic property. There are steady-state parameterized methods that use exhaust temperature to provide approximate in-cylinder temperature for this second state (e.g. [50]), but it is difficult to get exhaust temperature measurements that can resolve the blowdown process cycle-to-cycle [51]. This difficulty is highlighted in the recent residual estimation paper [6] that claims that reducing the observed model errors requires a faster exhaust temperature measurement (along with better air mass estimation and parameter adjustments to lessen the model's inherent oscillatory convergence behavior). There are also methods that use in-cylinder sampling with a high speed lambda meter [52] and methods that use the gross heat release and the lower heating values of the fuel [53]. The former adds additional complexity to the experimental apparatus, and may also have spatial sensitivity to the sampling point. The latter is a posteriori method that uses a gross heat release model that may not be parameterized well for transients and operating points with high CV.

In an effort to avoid these difficulties, only the Mirsky method [9] is considered here. This method is both simple and robust against experimental errors [9]. The method is derived by assuming the residual gases undergo a polytropic process through the exhaust event, and it can gauge the degree of coupling to the previous cycle with a residual ratio:

$$\frac{n_{residual}}{n_{total,\,previous\,cycle}} = \frac{V_{EVC}}{V_{EVO}} \left(\frac{P_{EVC}}{P_{EVO}} \right)^{1/b} \tag{3.5}$$

where b is the polytropic exponent, EVO is Exhaust Valve Open and EVC is Exhaust Valve Close. Assuming b, V_{EVC} and V_{EVO} are known and given the assumptions above, the equation can be generalized to:

$$\frac{n_{residual}}{n_{total,\,previous\,cycle}} = \text{function}(\, P_{EVO},\, P_{EVC}) \tag{3.6}$$

The combined functional form

Eq. 1.1 can be rewritten using the terminology developed in previous sections:

$$\tau \propto P^{-1.05} \left(\frac{n_{fuel}}{n_{air}}\right)^{-0.77} \left(\frac{n_{air}}{n_{total}}\right)^{-1.41} EXP\left(\frac{a}{R \cdot T}\right) \tag{3.7}$$

where $n_{total} = n_{residual+air+fuel} \approx n_{residual+air}$ because $n_{fuel} \ll n_{air}$. Note that O_2 carried over in the residual is neglected. The ideal gas law can then be incorporated with (θ) representing crank angle dependence and the tick mark as a reminder of the unit conversion between time and °CA which is engine speed dependent:

$$\tau' \propto P(\theta)^{-1.05}\, n_{fuel}^{-0.77}\, n_{air}^{-0.64}\, n_{total}^{+1.41}\, EXP\left(\frac{a \cdot n_{total}}{P(\theta) \cdot V(\theta)}\right) \tag{3.8}$$

To get the Start of Combustion θ_{SOC}, an autoignition integral [20] starting from θ_{IVC} can be used:

$$1 = \int_{\theta_{IVC}}^{\theta_{SOC}} \frac{1}{\tau'}\, d\theta \tag{3.9}$$

If a polytropic process is assumed during compression to define an exponential relation between $P(\theta)$ and the known $V(\theta)$, the integration to unity (Eq. 3.9) can be performed to implicitly define θ_{SOC} as:

$$\theta_{SOC} = \text{function}(\, n_{fuel},\, n_{air},\, n_{total},\, P_{IVC}) \tag{3.10}$$

From [34], CA90 can be computed as:

$$CA90 = \text{function}(\, \theta_{SOC},\, engine\ speed,\, \Phi') \tag{3.11}$$

Dropping the speed dependence[†] and noting that the definition of Φ' is $\frac{n_{fuel}}{n_{air}}\left(1 - \frac{n_{residual}}{n_{total}}\right)$, CA90 can be given by:

$$CA90 = \text{function}(\, n_{fuel},\, n_{air},\, n_{total},\, P_{IVC}) \tag{3.12}$$

At this point, even if the heat losses, polytropic exponents, specific heats, etc. were known for all the previous equations on a cycle-to-cycle basis, the residual quantity would not be known beyond the ratio of residual retained from the previous cycle using the Mirsky method. Despite this shortcoming, the author hypothesizes that enough information about the residual's influence on the next cycle is encoded between the residual ratio and the previous cycle's CA90 (based on the CA90 discussion in Sec. 3.1):

$$CA90_{n+1} = \text{function}(\, CA90_n,\, n_{fuel},\, n_{air},\, n_{total},\, P_{IVC}) \tag{3.13}$$

This equation can then be combined with previous relations for the residual ratio and air quantity, and reduced in dimensionality by assuming that an average NVO pressure P_{NVO} can be used instead of both P_{EVC} and P_{IVO} (the P_{NVO} range is shown in Fig. 3.1). Additionally, n_{fuel} is typically not able to be measured directly cycle-to-cycle, so injector pulse width (TI) is used here under the assumption that they are linearly related with P_{NVO} specified to capture the pressure drop from a constant fuel rail pressure. Since the heretofore equations do not include a Start of Injection (SOI) effect [46], the dependence must be added. Finally, while the larger "legs" of CA90 show that CA90 is a more sensitive indicator of what appears to be a single period doubling bifurcation in Fig 5.2, CA50 is used as the output of the mapping function instead of CA90 because CA50 is the standard metric of combustion

[†] While engine speed sensitivity is necessary for any practical engine model, the dependence is dropped here because the data used in this work were taken at a single engine speed. Additionally, the engine speed exponent in [34] is 0.3 over the fairly wide range of 750 to 4000 rpm. This indicates that phasing sensitivity to perturbations in engine speed is "small" relative to factors such as fueling (which has an exponent of -1.5). Sec. 8.4 will show that adaptation allows the model to tolerate speed variations.

phasing and it has the same functional dependence as CA90 [34]. Thus, after a considerable set of assumptions, we have:

$$CA50_{n+1} = \text{function}(\, CA90_n, \text{TI}, \text{SOI}, P_{EVO}, P_{NVO}, P_{IVC}) \tag{3.14}$$

While there is an argument for using CA90 instead of CA50 (see Sec. 3.1), it was ultimately found that all four combinations of CA50 and CA90 as inputs and outputs in Eq. 3.14 produced statistically similar phasing prediction results with the machine learning algorithm.

The reader will note that CA50 or CA90 is being iterated by the function along with a handful of variables to capture the global mixture state and autoignition characteristics in Eq 3.14. While this function clearly has more parameters that are not specified, CA50 or CA90 remains analogous to $Q(x)$'s mapping variable x and everything else basically quantifies a set of c parameters. Going further, if this mapping function can truly be considered a continuous, one-dimensional mapping of CA50 or CA90, there are implications with regard to the ordering and stability of periodic cycles. Specifically, Sarkovskii's theorem guarantees the existence of an (ostensibly unstable) fixed point when a period 2 or greater cycle is present [32, 36]. As the return maps will later show, at least a period 2 cycle is visible across a wide variety of conditions with late phased HCCI combustion. That said, Sarkovskii's theorem and other mathematical aspects are outside the scope of this work.

Chapter 4

Weighted Ring - Extreme Learning Machine

4.1 Overview

Near chaotic combustion is very sensitive to parameters, and it was found that model adaptation is essential to make the original mapping function approach with ϵ-SVR machine learning proposed by the author in [41] provide good generalization on out-of-sample measurements. To achieve adaptation in real-time, the following algorithm is designed to use an Extreme Learning Machine (ELM).

An ELM is a single-hidden layer feed forward network that uses randomly configured neurons in lieu of iteratively tuning them. The primary benefits of an ELM approach over the ϵ-SVR method originally proposed by the author in [41] are that an ELM is more easily adapted to online adaptation, provides good model generalization when the data are noisy, and is extremely fast to execute [54, 55].

Fig. 4.1 shows the algorithm layout graphically, with more details to be provided in subsequent sections.

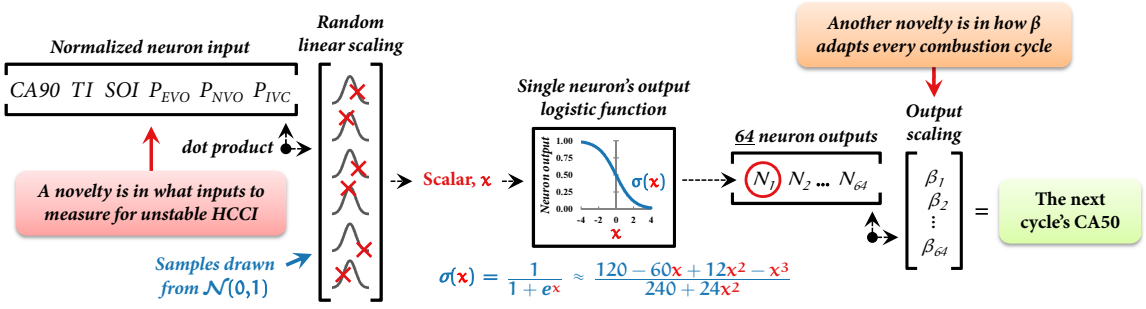


Figure 4.1 A graphical schematic of the Extreme Learning Machine used in this work.

4.2 The weighted ring

The Weighted Ring - Extreme Learning Machine (WR-ELM) is developed in this work as a weighted least squares extension to Online Sequential - ELM [55]. While developed independently, a similar derivation is available in [56]. The difference between this work and the classification application in [56] is the use of a ring buffer data structure "chunk" for online updates to an offline trained regression model.

The data in this WR-ELM ring buffer are weighted more heavily than the data originally used to fit the offline model. This allows extra emphasis to be placed on recent measurements that might be between the "quills" of Fig. 2.3's offline trained model or the result of day-to-day engine parameter variation. The ring buffer weights form a kind of "memory model" because values eventually exit the buffer and are "forgotten." Thus, this approach allows one to prescribe a balance between the least squares residuals of the offline trained model against the need to adapt to the most recent conditions. It explicitly avoids over adaptation to the local conditions (that could compromise global generality) by "forgetting" old ring buffer data. Fig. 4.2 gives a schematic representation of this approach, and Fig. 4.3 shows how this adaptation approach differs from typical adaptation approaches.

Other differences from [55, 56] are that the derivation below lacks a bias vector b, uses the Gaussian distribution for **a**, and drops the unnecessary logistic function exponential negative. It was found empirically that the computation of the bias b addition step could be removed with no loss of fitting performance if **a**'s elements were drawn from the Gaussian distribution $\mathcal{N}(0,1)$. ELM theory only requires the distribution to be continuous [54], although the ability

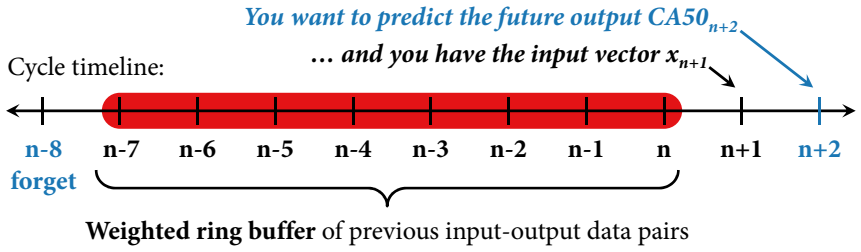


Figure 4.2 A schematic overview of WR-ELM ring buffer approach.

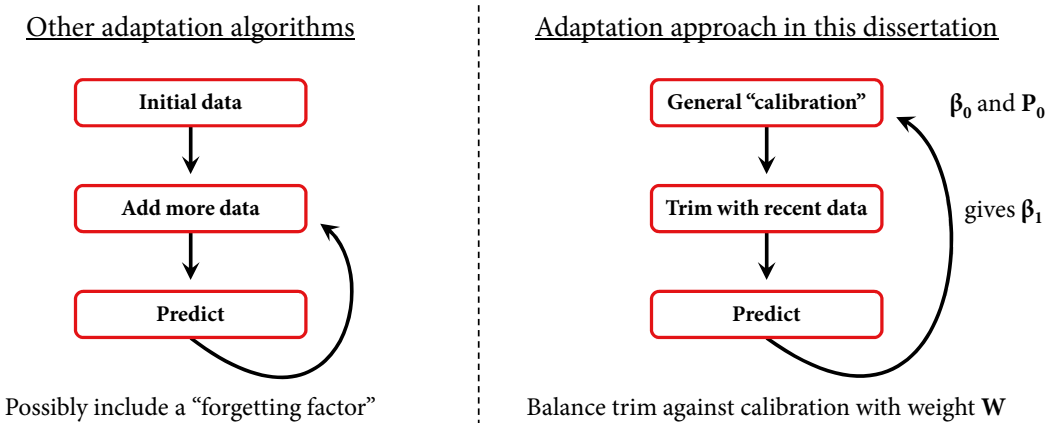


Figure 4.3 A high-level view of how the adaptation algorithm in this dissertation differs from typical machine learning adaptation approaches.

to remove the bias is likely problem specific.

4.3 Core algorithm

The basic goal of an Extreme Learning Machine (ELM) is to solve for the output layer weight vector $\boldsymbol{\beta}$ that scales the transformed input $\mathbf{H}$ to output $\mathbf{T}$:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{4.1}$$

where $\mathbf{H}$ is the hidden layer output matrix of a given input matrix and $\mathbf{T}$ is the target vector. $\mathbf{H}$ can be thought of as a non-linear front-end to the linear $\boldsymbol{\beta}$ back-end, with $\boldsymbol{\beta}$ being used to scale the front-end to fit the target vector $\mathbf{T}$.

For a set of n input-output data pairs and $\widetilde{N}$ neurons at the nth cycle timestep, these variables are given by

$$\mathbf{H}(\mathbf{a},\mathbf{x}) = \begin{bmatrix} G(\mathbf{a}_1,\mathbf{x}_1) & \dots & G(\mathbf{a}_{\widetilde{N}},\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1,\mathbf{x}_n) & \dots & G(\mathbf{a}_{\widetilde{N}},\mathbf{x}_n) \end{bmatrix}_{n\times\widetilde{N}} \tag{4.2}$$

where $G(\mathbf{a}_i, \mathbf{x})$ is the neuron activation function, chosen to be a commonly used logistic function, but without the unnecessary negative:

$$G(\mathbf{a}_i, \mathbf{x}) = \frac{1}{1 + exp(\mathbf{x} \cdot \mathbf{a}_i)} \,. \tag{4.3}$$

Using a random input weight vector $\mathbf{a}_i$ that is composed of random variable (r.v.) samples from a Gaussian distribution for each of the z input variables gives

$$\mathbf{a}_i = \begin{bmatrix} r.v. \sim \mathcal{N}(0,1) \\ \vdots \\ r.v. \sim \mathcal{N}(0,1) \end{bmatrix}_{z \times 1} \,. \tag{4.4}$$

The use of a random $\mathbf{a}_i$ that is re-sampled for each of the $\widetilde{N}$ individual neurons during initialization is the main difference of an Extreme Learning Machine vs. conventional neural networks that iteratively train each $\mathbf{a}_i$ [54]. These $\mathbf{a}_i$ vectors can then be collected into a single input weight matrix $\mathbf{a}$, which is held fixed across all n input row vectors $\mathbf{x}$

$$\mathbf{x} = \begin{bmatrix} CA90_n & TI & SOI & P_{IVC} & P_{EVO} & P_{NVO} \end{bmatrix}_{n \times z} \tag{4.5}$$

and n output values

$$\mathbf{T} = \begin{bmatrix} CA50_{n+1} \end{bmatrix}_{n \times 1} \,. \tag{4.6}$$

The normal equations can then be used to solve for the least squares solution $\boldsymbol{\beta}$ of Eq. 4.1 with

$$\boldsymbol{\beta} = (\mathbf{H}^\top \mathbf{H})^{-1} \mathbf{H}^\top \mathbf{T} \,. \tag{4.7}$$

To extend this to a weighted least squares solution, one can incorporate a diagonal weight

matrix $\mathbf{W}$ to the normal equations [57]:

$$\boldsymbol{\beta} = (\mathbf{H}^\top \mathbf{W}\mathbf{H})^{-1}\mathbf{H}^\top \mathbf{W}\mathbf{T}\,. \tag{4.8}$$

The solution can then be split between an offline and online "chunk" of recent input-output data pairs to avoid both the computational and storage burden of using offline data directly. To do this, the matrices are partitioned with subscript 0 and 1 denoting the offline and online updated components, respectively:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}_{n\times\widetilde{N}} \quad \mathbf{W} = \begin{bmatrix} \mathbf{W}_0 & \mathbf{o} \\ \mathbf{o} & \mathbf{W}_1 \end{bmatrix}_{n\times N} \quad \mathbf{T} = \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix}_{n\times 1}\,. \tag{4.9}$$

Then, following a similar derivation in [55] for recursive least squares but adding the weight matrix, the inversion portion $\mathbf{H}^\top\mathbf{W}\mathbf{H}$ of the weighted normal equations Eq. 4.8 can be re-written in terms of $\mathbf{K}_0$ and $\mathbf{K}_1$:

$$\begin{aligned} \mathbf{K}_1 = \mathbf{H}^\top\mathbf{W}\mathbf{H} &= \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{W}_0 & \mathbf{o} \\ \mathbf{o} & \mathbf{W}_1 \end{bmatrix} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{H}_0^\top & \mathbf{H}_1^\top \end{bmatrix} \begin{bmatrix} \mathbf{W}_0 & \mathbf{o} \\ \mathbf{o} & \mathbf{W}_1 \end{bmatrix} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{H}_0^\top\mathbf{W}_0 & \mathbf{H}_1^\top\mathbf{W}_1 \end{bmatrix} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \\ &= \mathbf{H}_0^\top\mathbf{W}_0\mathbf{H}_0 + \mathbf{H}_1^\top\mathbf{W}_1\mathbf{H}_1 \\ &= \mathbf{K}_0 + \mathbf{H}_1^\top\mathbf{W}_1\mathbf{H}_1\,. \end{aligned} \tag{4.10}$$

The non-inverted portion of the normal equations can similarly be re-written using existing

relations:

$$
\begin{aligned}
\mathbf{H}^\top \mathbf{W}\mathbf{T} &= \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{W}_0 & \mathbf{o} \\ \mathbf{o} & \mathbf{W}_1 \end{bmatrix} \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{H}_0^\top & \mathbf{H}_1^\top \end{bmatrix} \begin{bmatrix} \mathbf{W}_0 & \mathbf{o} \\ \mathbf{o} & \mathbf{W}_1 \end{bmatrix} \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{H}_0^\top \mathbf{W}_0 & \mathbf{H}_1^\top \mathbf{W}_1 \end{bmatrix} \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \\
&= \mathbf{H}_0^\top \mathbf{W}_0 \mathbf{T}_0 + \mathbf{H}_1^\top \mathbf{W}_1 \mathbf{T}_1 \\
&= \mathbf{K}_0 \mathbf{K}_0^{-1} \mathbf{H}_0^\top \mathbf{W}_0 \mathbf{T}_0 + \mathbf{H}_1^\top \mathbf{W}_1 \mathbf{T}_1 \\
&= \mathbf{K}_0 \boldsymbol{\beta}_0 + \mathbf{H}_1^\top \mathbf{W}_1 \mathbf{T}_1 \\
&= \left(\mathbf{K}_1 - \mathbf{H}_1^\top \mathbf{W}_1 \mathbf{H}_1 \right) \boldsymbol{\beta}_0 + \mathbf{H}_1^\top \mathbf{W}_1 \mathbf{T}_1 \\
&= \mathbf{K}_1 \boldsymbol{\beta}_0 - \mathbf{H}_1^\top \mathbf{W}_1 \mathbf{H}_1 \boldsymbol{\beta}_0 + \mathbf{H}_1^\top \mathbf{W}_1 \mathbf{T}_1 \, .
\end{aligned} \tag{4.11}
$$

Substituting Eq. 4.11 into the full online solution

$$
\begin{aligned}
\boldsymbol{\beta}_1 &= \left(\mathbf{K}_1^{-1} \right) \left(\mathbf{H}^\top \mathbf{W}\mathbf{T} \right) \\
&= \boldsymbol{\beta}_0 - \mathbf{K}_1^{-1} \mathbf{H}_1^\top \mathbf{W}_1 \mathbf{H}_1 \boldsymbol{\beta}_0 + \mathbf{K}_1^{-1} \mathbf{H}_1^\top \mathbf{W}_1 \mathbf{T}_1 \\
&= \boldsymbol{\beta}_0 + \mathbf{K}_1^{-1} \mathbf{H}_1^\top \mathbf{W}_1 \left(\mathbf{T}_1 - \mathbf{H}_1 \boldsymbol{\beta}_0 \right)
\end{aligned} \tag{4.12}
$$

yields the online solution without the need for the offline dataset. To trade the computational burden of the $\widetilde{\mathrm{N}} \times \widetilde{\mathrm{N}}$ sized $\mathbf{K}$ inverse for an inverse that scales with smaller sized ring buffer, one can let $\mathbf{P} = \mathbf{K}^{-1}$

$$
\mathbf{P}_0 = \mathbf{K}_0^{-1} = \left(\mathbf{H}_0^\top \mathbf{W}_0 \mathbf{H}_0 \right)^{-1} \tag{4.13}
$$

$$
\mathbf{P}_1 = \mathbf{K}_1^{-1} = \left(\mathbf{P}_0^{-1} + \mathbf{H}_1^\top \mathbf{W}_1 \mathbf{H}_1 \right)^{-1} \tag{4.14}
$$

and use the matrix inversion lemma on Eq. 4.14 to yield:

$$\mathbf{P}_1 = \mathbf{P}_0 - \mathbf{P}_0\mathbf{H}_1^\top \left(\mathbf{W}_1^{-1} + \mathbf{H}_1\mathbf{P}_0\mathbf{H}_1^\top\right)^{-1} \mathbf{H}_1\mathbf{P}_0 \,. \tag{4.15}$$

Unlike OS-ELM and WOS-ELM [55, 56], additional simplification is possible because the WR-ELM algorithm does not propagate $\mathbf{P}_1$ in time. To begin, append the $\mathbf{H}_1^\top\mathbf{W}_1$ portion of Eq. 4.12 to Eq. 4.15 and distribute $\mathbf{H}_1^\top$ to give:

$$\begin{gathered}\mathbf{P}_1\mathbf{H}_1^\top\mathbf{W}_1 = \\ \left[\mathbf{P}_0\mathbf{H}_1^\top - \mathbf{P}_0\mathbf{H}_1^\top \left(\mathbf{W}_1^{-1} + \mathbf{H}_1\mathbf{P}_0\mathbf{H}_1^\top\right)^{-1} \mathbf{H}_1\mathbf{P}_0\mathbf{H}_1^\top\right]\mathbf{W}_1 \,.\end{gathered} \tag{4.16}$$

Eq. 4.16 can then be simplified with the substitutions $\mathbf{A} = \mathbf{P}_0\mathbf{H}_1^\top$, $\mathbf{B} = \mathbf{H}_1\mathbf{A}$ and then distributing $\mathbf{W}_1$ to provide:

$$\mathbf{P}_1\mathbf{H}_1^\top\mathbf{W}_1 = \mathbf{A}\left[\mathbf{W}_1 - \left(\mathbf{W}_1^{-1} + \mathbf{B}\right)^{-1}\mathbf{B}\mathbf{W}_1\right] \,. \tag{4.17}$$

Transforming Eq. 4.17 with the identity $(\mathbf{X}+\mathbf{Y})^{-1}\mathbf{Y} = \mathbf{X}^{-1}(\mathbf{X}^{-1}+\mathbf{Y}^{-1})^{-1}$ [58] gives:

$$\mathbf{P}_1\mathbf{H}_1^\top\mathbf{W}_1 = \mathbf{A}\left[\mathbf{W}_1 - \mathbf{W}_1\left(\mathbf{W}_1 + \mathbf{B}^{-1}\right)^{-1}\mathbf{W}_1\right] \,. \tag{4.18}$$

Eq. 4.18 is then in a form where the identity $\mathbf{X} - \mathbf{X}(\mathbf{X}+\mathbf{Y})^{-1}\mathbf{X} = (\mathbf{X}^{-1}+\mathbf{Y}^{-1})^{-1}$ [58] can be applied to yield a substantially simpler form with a ring buffer sized inverse:

$$\mathbf{P}_1\mathbf{H}_1^\top\mathbf{W}_1 = \mathbf{A}\left(\mathbf{W}_1^{-1} + \mathbf{B}\right)^{-1} \,. \tag{4.19}$$

Finally, noting that $\mathbf{P}_1\mathbf{H}_1^\top\mathbf{W}_1 = \mathbf{K}_1^{-1}\mathbf{H}_1^\top\mathbf{W}_1$, one can then substitute Eq. 4.19 into Eq. 4.12 and arrive at the following algorithm summary:

OFFLINE TRAINING

$$\begin{aligned}\mathbf{P}_0 &= \left[\left(\mathbf{H}_0^\top \mathbf{W}_0 \mathbf{H}_0\right)^{-1}\right]_{\widetilde{\mathrm{N}}\times\widetilde{\mathrm{N}}} \\ \boldsymbol{\beta}_0 &= \left[\mathbf{P}_0 \mathbf{H}_0^\top \mathbf{W}_0 \mathbf{T}_0\right]_{\widetilde{\mathrm{N}}\times 1}\end{aligned} \tag{4.20}$$

ONLINE ADAPTATION

$$\begin{aligned}\mathbf{A} = \mathbf{P}_0\mathbf{H}_1^\top,\ \mathbf{B} &= \mathbf{H}_1\mathbf{A} \\ \boldsymbol{\beta}_1 = \boldsymbol{\beta}_0 + \mathbf{A}\left(\mathbf{W}_1^{-1} + \mathbf{B}\right)^{-1}&(\mathbf{T}_1 - \mathbf{H}_1\boldsymbol{\beta}_0)\end{aligned} \tag{4.21}$$

ONLINE PREDICTIONS

$$\mathrm{CA50}_{n+2} = \mathbf{T}_{n+1} = \mathbf{H}(\mathbf{a}, \mathbf{x}_{n+1})\boldsymbol{\beta}_1 \tag{4.22}$$

The reader should note that only $\mathbf{P}_0$ and $\boldsymbol{\beta}_0$ are needed for online adaptation, and the size of these matrices scales only with an increasing number of neurons $\widetilde{\mathrm{N}}$. **None of the original offline data are needed**. Additionally, note that Eq. 4.22 is simply the reverse of Eq. 4.1 with the most recent $\mathbf{x}_{n+1}$ cycle vector and $\boldsymbol{\beta}_1$ updated from the weighted ring buffer. Finally, it should mentioned that the resulting update law Eq. 4.21 is structurally similar that of the steady-state Kalman filter [57], which also uses recursive least squares. Future work should look at applying Kalman filtering algorithm improvements (e.g. square root filtering) to WR-ELM.

4.4 Usage procedure

1. Scale **x** and **T** columns between zero and unity for each variable. For the combustion implementation, column variable values below the 0.1% and above the 99.9% percentile were saturated at the respective percentile value, and then normalized between zero and unity between these percentile based saturation limits. This was done to both adequately represent the distribution tails and avoid scaling issues.

2. The random non-linear front-end transformation that enables the low computational complexity of the WR-ELM algorithm may result in ill-conditioned matrices. All numerical implementations should use double precision. Additionally, one should consider using Singular Value Decomposition for ill-conditioned matrix inversions.

3. Using the $\mathcal{N}(0,1)$ Gaussian distribution, initialize the $z \times \widetilde{N}$ ELM input weights $\mathbf{a}$ and hold them fixed for all training / predictions. For the combustion implementation, this was done with MATLAB®'s built-in `randn()` function and the Mersenne Twister pseudo random number generator with seed 7,898,198. An $\widetilde{N}$ of 64 was used based on initial trials, and each cylinder's individually computed WR-ELM model used an identical input weight matrix $\mathbf{a}$.

4. Build $\mathbf{H}_0(\mathbf{a}, \mathbf{x}_0)$ from previously acquired samples that cover a wide range of conditions with Eq. 4.2 using an input matrix $\mathbf{x}_0$ and output target vector $\mathbf{T}_0$ (the formats of these are given in Eqs. 4.5 & 4.6, respectively). For the combustion implementation, the initial training data were ~40 minutes of random engine set points covering 53,884 cycles and 1,179 random engine set points at a single engine speed; however, it appears that only ~20 minutes of data may be sufficient. Pruning the training data to only include ~6 cycles before and ~9 cycles after a transient set point step provided a small model fitting performance improvement.

5. Specify a weight matrix $\mathbf{W}_0$ for offline measurements. For the combustion implementation, a simple scalar value $\mathbf{W}_0 = 3.5 \times 10^{-3}$ was chosen using a design of experiments (see Chapter 6). While this weight works well as a proof of concept, future work should more rigorously determine the weight(s), perhaps with optimization techniques. Note that $\mathbf{W}_0$ allows weighting to be applied offline and that a small offline weighting is equivalent to a large online weighting.

6. Solve for the offline solution $\mathbf{P}_0$ and $\boldsymbol{\beta}_0$ using Eqs. 4.20 and hold these values constant

for all future predictions.

7. Populate a ring buffer of size r with recently completed input-output pairs using:

$$\begin{aligned} \textbf{input ring buffer} = \mathbf{x}_1 &= \begin{bmatrix} \mathbf{x}_{n-r+1} \\ \vdots \\ \mathbf{x}_n \end{bmatrix}_{r\times z} \\ \textbf{output ring buffer} = \mathbf{T}_1 &= \begin{bmatrix} CA50_{n-r+2} \\ \vdots \\ CA50_{n+1} \end{bmatrix}_{r\times 1} \end{aligned} \tag{4.23}$$

and execute the WR-ELM update algorithm between combustion cycle $n+1$ and $n+2$ as shown in Fig. 4.2. For the combustion implementation, r was taken to be 8 cycles after tuning with existing datasets (see Chapter 6). If desired, r can vary cycle-to-cycle.

8. As with the offline data, build $\mathbf{H}_1(\mathbf{a}, \mathbf{x}_1)$ with Eq. 4.2 using an input matrix $\mathbf{x}_1$ and output target vector $\mathbf{T}_1$. Specify a weight matrix $\mathbf{W}_1$. For the combustion implementation the identity matrix ($\mathbf{W}_1 = \mathbf{I}$) was chosen since weighting was already applied to the offline data in step 5. Gradually increased weighting on the most recent time steps in the ring buffer was explored, however, it did not net a significant improvement to model fitting performance over a simple scalar value on offline data. Although not explored in the current implementation, $\mathbf{W}_1$ can vary cycle-to-cycle.

9. Solve for the updated $\boldsymbol{\beta}_1$ solution using Eqs. 4.21.

10. After cycle $n+1$'s input vector $\mathbf{x}_{n+1}$ is fully populated, transform vector into $\mathbf{H}_{n+1}$ using Eq. 4.2 and solve for a predicted target value $\mathbf{T}_{n+1}$ or $CA50_{n+2}$ using Eq. 4.22.

11. Repeat steps 7-10 for each new time step, caching results (e.g. hidden layer outputs) from previous time steps to reduce computational requirements.

4.5 Real-time exponential

While the above algorithm works well, one modification improves the computational efficiency on processors without a dedicated $exp(x)$ instruction, such as the Raspberry Pi®. The modification is to replace the exponential with the following Padé approximant:

$$exp(x) \approx p(x) = \frac{120 + 60x + 12x^2 + x^3}{120 - 60x + 12x^2 - x^3}\,, \tag{4.24}$$

which has the following simple relations for the logistic of Eq. 4.3:

$$\frac{1}{1 + exp(x)} = \sigma(x) \approx \frac{1}{1 + p(x)} = s(x) = \frac{(120 + 12x^2) - 60x - x^3}{2 \cdot (120 + 12x^2)} \tag{4.25}$$

$$\frac{d}{dx} \cdot \sigma(x) = \sigma(x)(\sigma(x) - 1) \approx s(x)(s(x) - 1) \tag{4.26}$$

The small number of floating point operations used, reused intermediate terms, known boundedness of the normalized inputs, and known **a** weights make this approximant work well in this application. No significant degradation in model performance was found, and as such it is used in all implementations described hereafter. Fig. 4.4 shows the expected errors of using the approximant. The analytical derivative (which will be discussed in later chapters) is also shown. It is worth mentioning that this approximant was found to provide better prediction results than Schraudolph's bit manipulation algorithm [59], ostensibly because it is a smooth function without a stairstep in the least significant bits.*

4.6 Analytical partial derivative

Finally, below is the ELM prediction dot product (Eq. 4.22) presented as an explicit sum to clearly show how one might compute a cycle-to-cycle analytical partial derivative of the

* There are other formulations of Schraudolph's algorithm that avoid the least significant bit stairstep, but they were not tried due to project time constraints.

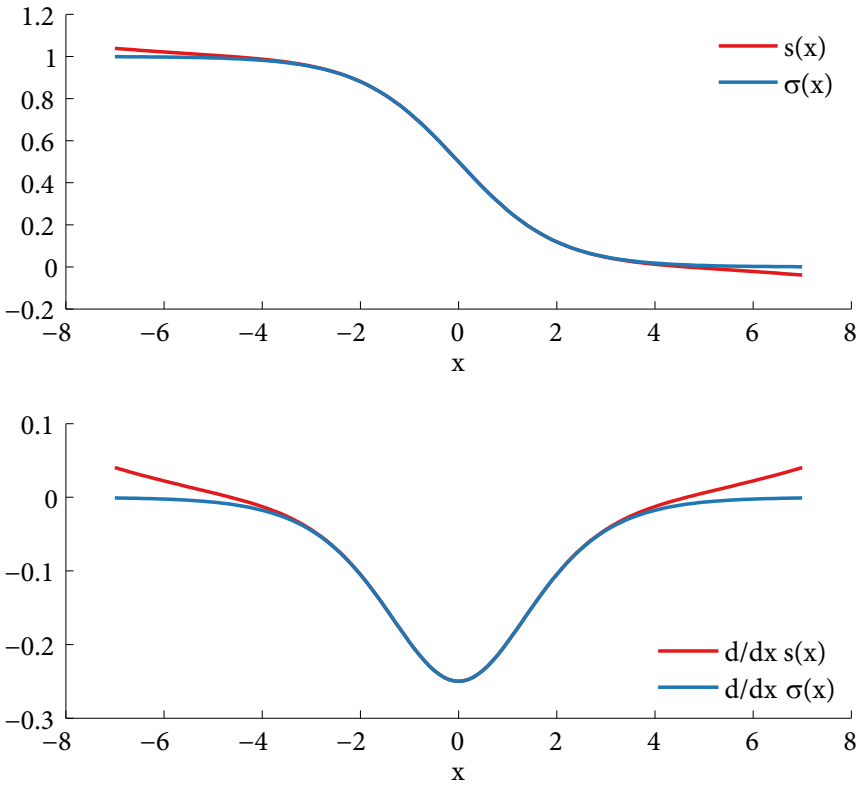


Figure 4.4 The logistic function, its derivative, and associated Padé approximations.

algorithm for a given input variable x while holding the other inputs constant (shown as c). The notation $a_{x,i}$ refers to the random input weighting in **a** for a given neuron number i's input variable x and s is from the Padé logistic approximants of Eqs. 4.25 and 4.26.

$$t_i = s(a_{x,i} \cdot x_{n+1} + c) \tag{4.27}$$

$$CA50_{n+2}(x) = \sum_{i=1}^{\widetilde{N}} \beta_i \cdot t_i \tag{4.28}$$

$$\frac{\partial CA50_{n+2}}{\partial x}(x) = \sum_{i=1}^{\widetilde{N}} a_{x,i} \cdot \beta_i \cdot t_i \cdot (t_i - 1) \tag{4.29}$$

Chapter 5

Initial Predictions

5.1 Test conditions

As part of this work, two experimental test cells at the University of Michigan were set up by the author with other collaborators (see Fig. 5.1). Test cell number ❶ was used to provide initial model training data and used again three years later for the real-time control feasibility experiments described in Chapter 8. Test cell number ❷ was used in a separate study to briefly confirm that WR-ELM adaptation can allow the model to work on an engine with a different head, different injector orientation (central mount), and slightly different compression ratio.

Table 5.1 provides a summary of the experimental setup and conditions visited for the initial data with test cell number ❶. The reference cam lift for timing and duration in Table 5.1 is 0.5 mm. The full collection of 129,964 cycles is comprised of five ~20 minute random test

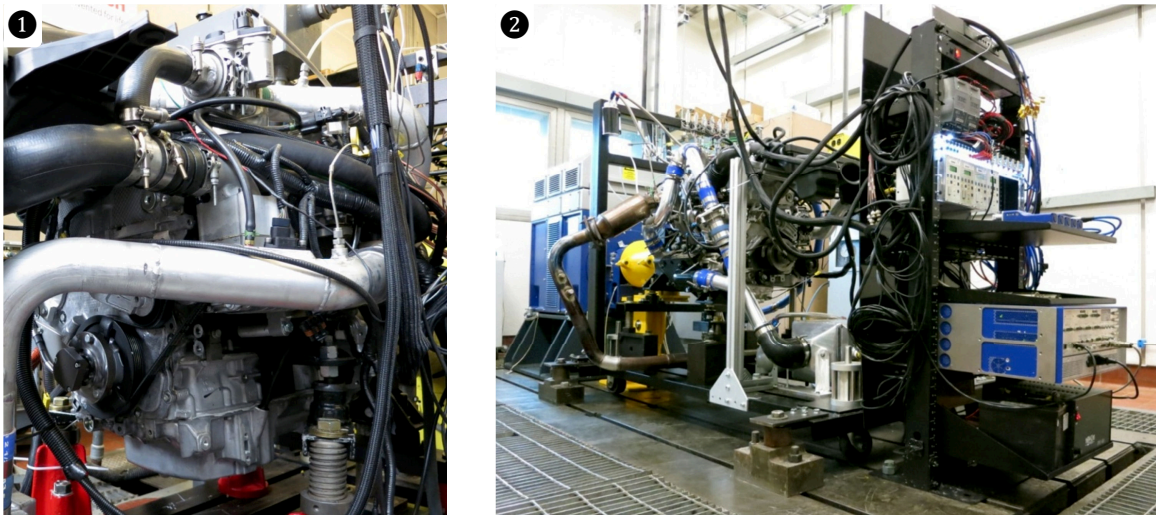


Figure 5.1 Two experimental test cells at the University of Michigan.

subsequences. Each random subsequence covers the same nominal ranges listed in Table 5.1; however, one subsequence holds SOI fixed. The sequence with SOI fixed is only used as part of Fig. 5.2, and not during the model training and testing presented here.

Pressure data for each cylinder were sampled at a resolution of 1.0 °CA and pegged thermodynamically for each cycle after IVC using a polytropic exponent of 1.35. This exponent was chosen to most closely match the pegging results achieved using the single intake runner high speed pressure sensor on cylinder 1. For the purpose of computing cycle-to-cycle net Indicated Mean Effective Pressure (IMEP), a cycle was defined as starting at 360 °BTDC firing and ending at 359 °ATDC firing. The reference cam lift for timing and duration in Table 5.1 is 0.5 mm. The air-fuel ratio range indicated in Table 5.1 was measured post-turbine, and represents a mixture from all four cylinders. Fuel mass per cycle was *estimated* using the fuel's lower heating value, assuming that the gross heat release was 20% greater than Q_{net} and that the combustion efficiency was 100%.

Table 5.1 Experimental setup and test conditions.

Engine		
Make / model		GM / LNF Ecotec
Cylinder layout		in-line 4
Overall displacement		2.0 L
Bore / stroke		86 / 86 mm
Geometric compression ratio	(*modified from stock engine*)	11.2 : 1
Cam lift	(*modified from stock engine*)	3.5 mm
Cam duration	(*modified from stock engine*)	93 °CA
Cam phaser type		hydraulic
Fuel injector type		direct, side mounted, wall guided
Fuel		
Designation		Haltermann HF0437, EPA Tier II EEE
Description		U.S. Federal Emission Certification Gasoline
Research Octane Number		97.0
Motor Octane Number		88.1
ASTM D240 heating value		42.8 MJ / kg
Aromatic / olefin / saturate fractions		28 / 1 / 71 % volume
Test conditions		
Throttle position		wide open
Turbocharger		wastegate open
Supercharger		bypassed
Residual retention strategy		negative valve overlap
IVO set point range	(1^{st} / 99^{th} *percentile*)	78.6 / 128 °ATDC
EVC set point range	(1^{st} / 99^{th} *percentile*)	-118 / -83.0 °ATDC
SOI set point range	(1^{st} / 99^{th} *percentile*)	272 / 378 °BTDC
TI set point range	(1^{st} / 99^{th} *percentile*)	0.582 / 1.01 ms
Net IMEP values visited	(1^{st} / 99^{th} *percentile*)	1.85 / 3.62 bar
Air-fuel ratios visited	(1^{st} / 99^{th} *percentile*)	0.90 / 1.6
Estimated fuel per cycle	(1^{st} / 99^{th} *percentile*)	6.0 / 11 mg
Intake runner temps. across all four cyls.		$\mu = 52.6$ °C, $\sigma = 1.6$ °C
Fuel injection pressure		$\mu = 70.0$ bar, $\sigma = 0.85$ bar
Coolant temperature		$\mu = 89.5$ °C, $\sigma = 3.4$ °C
Engine speed		$\mu = 2500$ rpm, $\sigma = 6$ rpm

5.2 Outlier criteria

The transitions between random set points were frequently harsh, and thus a robust method was needed to systematically remove cycles with very poor or no combustion that do not have well defined combustion phasing. To this end, cycles were classified as outliers if they met any of the following criteria:

1. A misfire had occurred with a $Q_{net} < 50$ J. Note that this definition is very permissive, and it does not distinguish between a misfire and a partial burn.

2. A linear, least squares fit of the tail of the cumulative heat release curve between 50 and 60 °ATDC had an absolute slope that was greater than 3 J / °CA. This criterion is also very permissive, and it is needed to remove very late burns that do not have a stable Q_{net} at the end of the heat release evaluation window.

3. A cycle was preceded by a cycle that met either of the two criteria above. This is a necessary constraint because the model structure depends on reliably knowing the previous cycle's combustion phasing.

Outlier cycles that met the above criteria totaled only ~3% of the data acquired.

5.3 General observations

The entire 129,964 cycle dataset (excluding outliers as defined in Sec. 5.2) was used to generate the return maps seen in Fig. 5.2. The noisy "legs" that grow from the diagonal show the system's oscillatory behavior. While CA10 (the time in °CA where the integral in Eq. 3.1 achieves 10% of its mean final value Q_{net}) in Fig. 5.2(a), CA50 in Fig. 5.2(b) and CA90 in Fig. 5.2(c) all show similar structure, the larger "legs" of CA90 show that CA90 is a more sensitive indicator of what appears to be a single period doubling bifurcation. For comparison, note the similarities between the period doubling bifurcation in Fig. 2.2(b) and the experimental

results of Fig. 5.2. It is also clear that this bifurcation structure in cycle-to-cycle combustion phasing has remained surprisingly well defined despite the large number of operating points explored in the dataset. Additionally, it is shown that the extent of the bifurcation can be different depending on the cylinder that one is observing, despite all the cylinders receiving the same input. In particular, cylinder 4 in Fig. 5.2(f) shows significantly less oscillatory behavior when compared to the other three cylinders in Figs. 5.2(c-e).

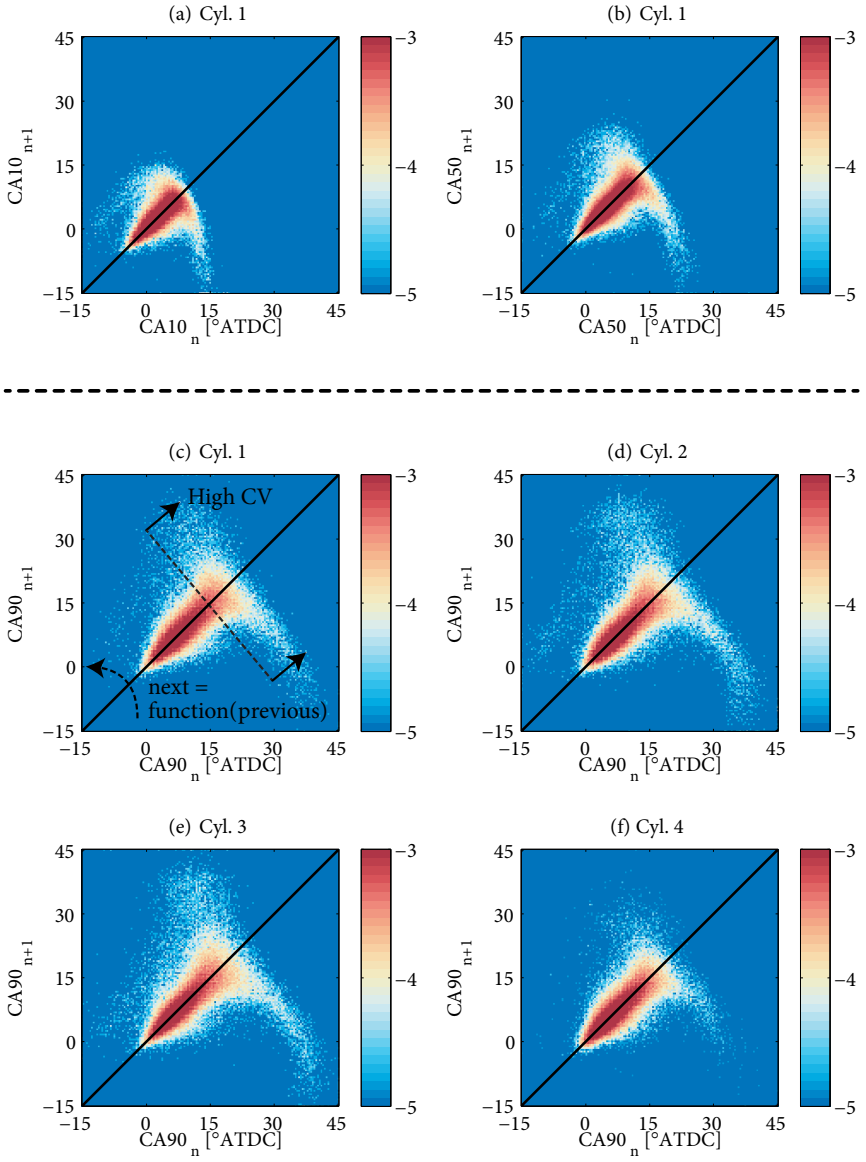


Figure 5.2 Experimentally acquired return map probability histograms of CA10, CA50 and CA90 using the 129,964 cycle dataset with 2,221 actuator set points. The colormap is $\log_{10}$ to show order of magnitude differences.

Fig. 5.3 shows how CA90 relates to injector pulse width and net heat release when projected onto the IMEP axis. The author does not consider IMEP to be a fundamental parameter like the c for $Q(x)$, but it is a convenient variable to separate out different operating regions. The reader will note that while there is considerable noise, Figs. 5.3(a, d, g and j) show what appear to be oscillatory "legs" developing as one moves to IMEP values below 2.5 bar, similar to $Q(x)$'s bifurcation diagram in Fig. 2.1. For the inner cylinders (2 and 3), it appears as though there is only a single large bifurcation in CA90 that causes the bifurcation "legs" in Fig. 5.2. For the outer cylinders (1 and 4), the low-load behavior is clearly more complex. Upon examining the IMEP time series for the full dataset, it was also observed that the outer cylinders typically move together to IMEP values that differ from the inner cylinders, despite receiving the same actuator inputs. One can speculate that this behavior is result of dynamic effects in the manifolds affecting VE; however, it is left to future work with cylinder-to-cylinder emissions data and computational fluid dynamics to confirm this hypothesis.

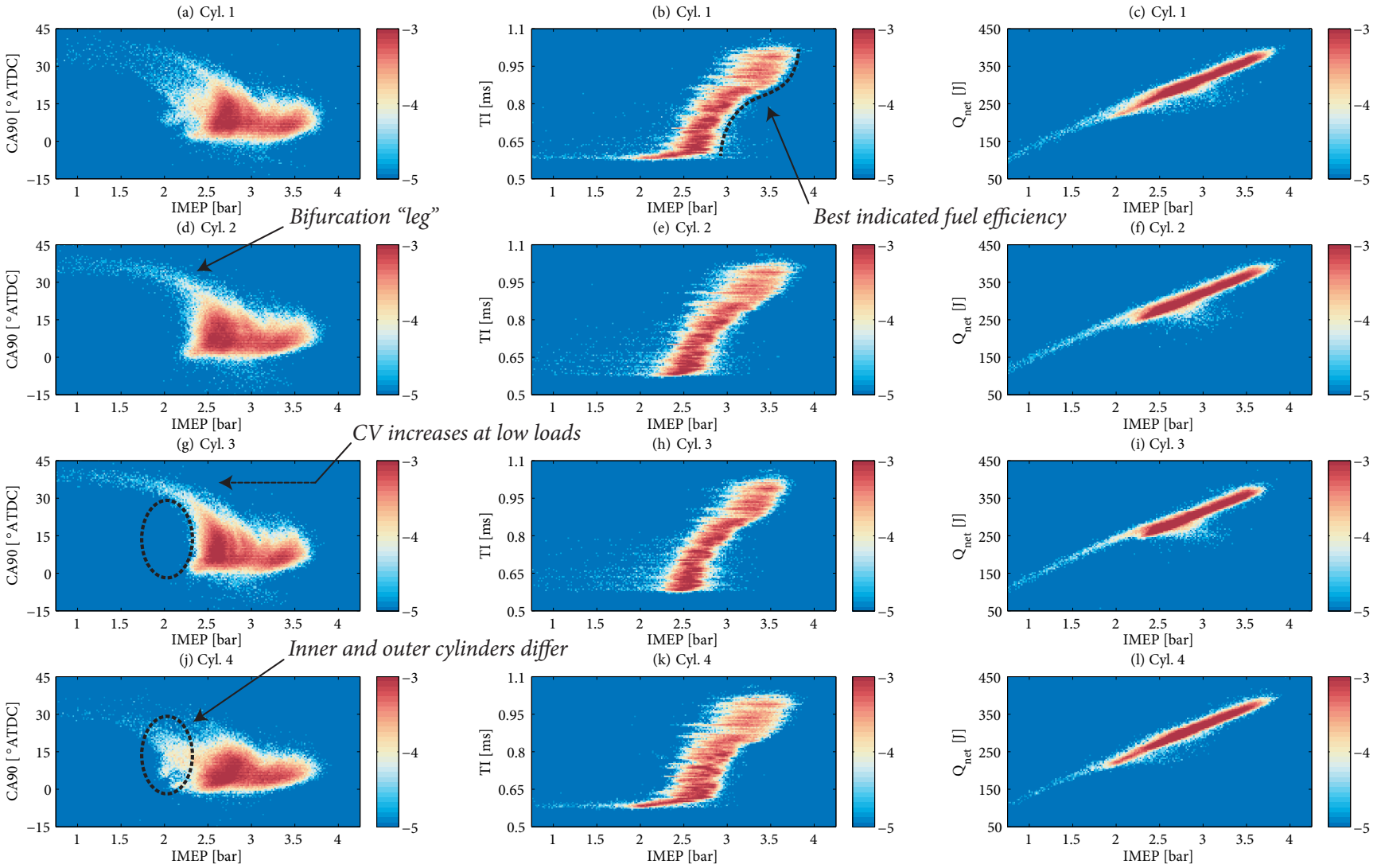


Figure 5.3 Experimentally acquired probability histograms of CA90, TI and Q_{net} vs. IMEP across 129,964 cycles and 2,221 actuator set points. The colormap is $\log_{10}$.

5.4 Model training procedure

The offline solution was trained using ~40 minutes of test cell time covering 53,884 cycles and 1,179 random engine set points at 2,500 rpm (two random subsequences). These subsequences are comprised of random, transient set point steps occurring approximately every 0.5 - 10 sec. with occasional misfires covering the nominal variable ranges given in Table 5.1. The training data were pruned to only include 6 cycles before and 9 cycles after a transient set point step for a small model fitting performance improvement. The online solution was run with a separate random subsequence and fed unseen cycles one-by-one, similar to what would be experienced in a real-time implementation. This online dataset is comprised of 25,323 consecutive cycles with 521 random engine set points. Longer online sequences were also tested, and achieved similar results.

5.5 Model performance

The WR-ELM model's fitting performance on a 25,323 cycle dataset (excluding outliers as defined in Sec. 5.2) is shown in Table 5.2 and in Figs. 5.4, 5.5, and 5.6. The minimum coefficient of determination (R^2) given in Table 5.2 shows that at least 80% of the cycle-to-cycle variance can be explained by the model as it is currently defined for a dataset with random transient

Table 5.2 WR-ELM model of Eq. 3.14 error statistics.

Cylinder #	Overall† R^2	Overall RMSE [°CA]	Steady-State RMSE [°CA]
1	0.81	1.85	0.84
2	0.81	2.06	0.97
3	0.80	2.17	0.97
4	0.83	1.64	0.86

† 25,323 consecutive cycles with random transient steps occurring approximately every 0.5 - 10 sec. and occasional misfires.

steps occurring approximately every 0.5 - 10 sec. and occasional misfires. This is better than the 76% achieved with ϵ-Support Vector Regression (ϵ-SVR) on the same dataset in [41]. However, this is not a 1:1 comparison because WR-ELM is fully predicting the entire 25,323 cycle dataset, whereas ϵ-SVR's training strategy ensured the data-driven model had partially seen the operating points it was trying to predict. Steady-state Root Mean Squared Error (RMSE) in Table 5.2 was assessed at a single set point with a mean CA50 of 3.9 °ATDC and a net Indicated Mean Effective Pressure (IMEP) of 2.8 bar before the transient sequence started.

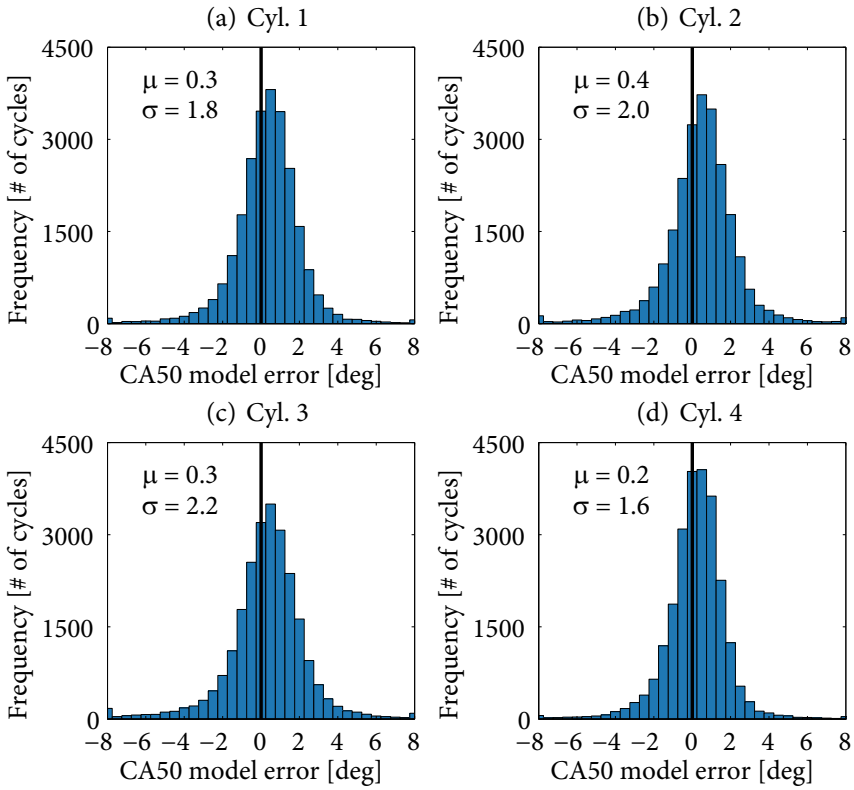


Figure 5.4 Error histograms for WR-ELM model of Eq. 3.14 across 25,323 consecutive cycles with random transient steps occurring approximately every 0.5-10 sec. and occasional misfires.

Fig. 5.4 shows the distribution of model errors, and it is clear that there is a slight positive bias to the predictions. Fig. 5.5 provides insight into the tails of Fig. 5.4 and shows that model errors still generally capture the correct directionality. Fig. 5.5 also shows that late phasing is somewhat under predicted, and that the positive bias is largely from the midrange values of CA50. The cycle-to-cycle model fit presented in Fig. 5.6 shows good agreement, with

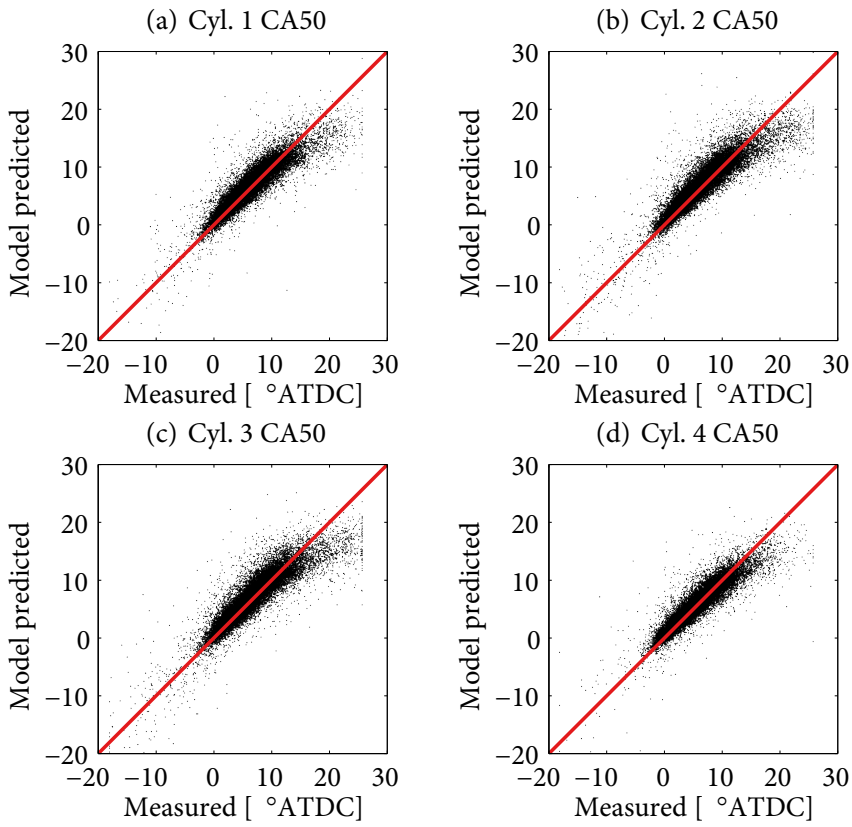


Figure 5.5 Predicted vs. measured WR-ELM model of Eq. 3.14 across 25,323 consecutive cycles with random transient steps occurring approximately every 0.5 - 10 sec. and occasional misfires. Late phasing is somewhat under predicted, but almost all prediction outliers still capture the correct directionality.

occasional tracking errors that are quickly corrected by online adaption. Missing segments are outliers (as defined in Sec. 5.2) and total only 3% of the data recorded from the experiment. The reader is encouraged to compare and contrast this fit against the partially acausal fit in Fig. 6 of [41] for the same exact experimental subsequence. Fig. 5.6 also includes colormaps to provide qualitative insight into the model weights and online adaptation behavior. Note that the β weights for next cycle predictions are computed on the previous cycle. Also note that the same non-linear front-end specified by $\mathbf{a}$ is used for each cylinder, and any cylinder-to-cylinder differences in the cycle-to-cycle $\hat{\beta}_1$ are due to different characteristics of each cylinder. The neurons are sorted by the 2-norm of their respective input weight vector $\mathbf{a}_i$.

Overall, the author believes the level of fit shown in Table 5.2 and in Figs. 5.4, 5.5, and 5.6

is very good considering that the dataset includes both transients and operating points with high CV, right up to complete misfire. Finally, some of the model tracking errors appear to be correlated with misfires on other cylinders. Each cylinder model is independent, so such the cross dependence is not captured, but it does appear that adaptation is robust against these outside influences. In future work, it would be of interest to explore the influence of calibration ranges and pressure sampling resolution on model fit statistics.

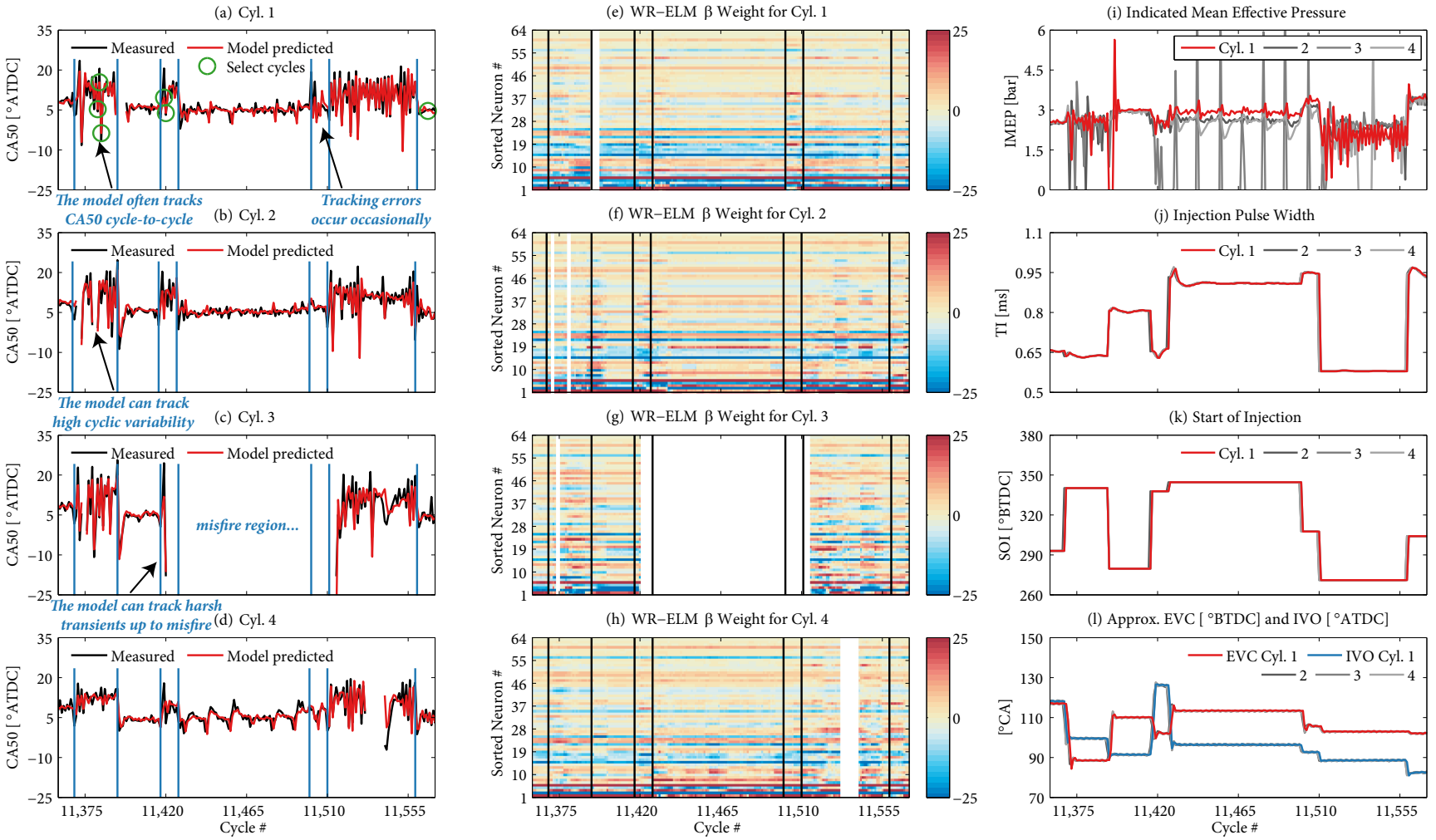


Figure 5.6 The WR-ELM CA50 model of Eq. 3.14 can track CA50 through transients every 0.5 - 10 sec., operating points with high CV, and at steady-state during a particularly harsh region of the 25,323 cycle dataset that includes misfires. The colormaps (linearly scaled) provide qualitative insight into the level of cycle-to-cycle adaptation and into cylinder-to-cylinder model differences. The blue and black vertical lines mark random engine set point steps in sub-figures a-h.

Chapter 6
Sensitivity and Cycle-to-Cycle Extrapolation

6.1 Model parameter and input sensitivity

A series of parameter sensitivity studies were performed using the initial datasets and model described in Chapters 4 and 5. The first study varied the offline data weight $\mathbf{W}_0$ while setting the online ring buffer weight $\mathbf{W}_1$ to the identity matrix. Note that a small $\mathbf{W}_0$ is the same as having a large $\mathbf{W}_1$ on the ring buffer measurements. From the figure, it is clear that additional emphasis on the recent measurements can boost model fitting performance up to a peak R^2 at $\sim 3.5 \times 10^{-3}$. Qualitative observations of the model behavior above and below this value are what one would expect, with smaller $\mathbf{W}_0$ values over-adapting to current conditions

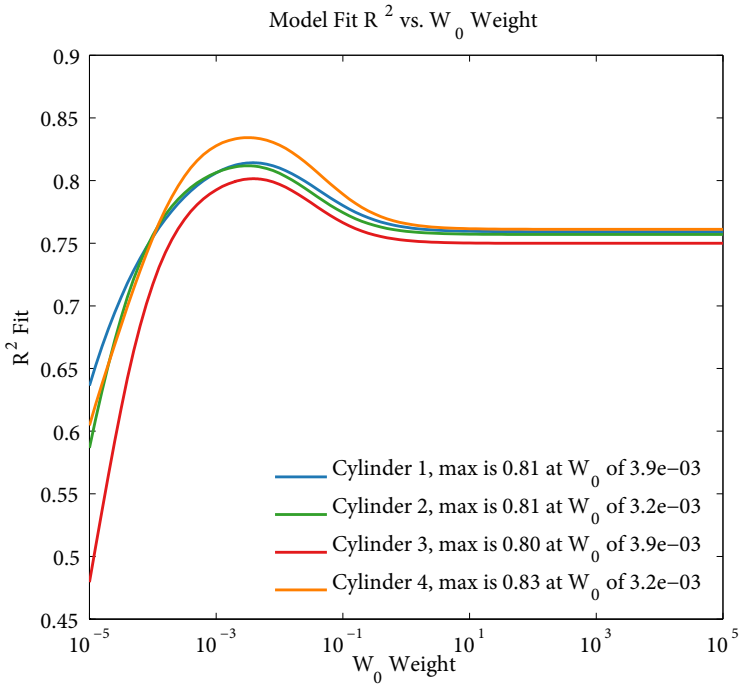


Figure 6.1 $\mathbf{W}_0$ sensitivity with 64 neuron WR-ELM and ring buffer size of 8.

(Fig. 6.2) and larger values not adapting enough (Fig. 6.3) to effectively track combustion behavior (occasionally producing an observed offset in the predicted value).

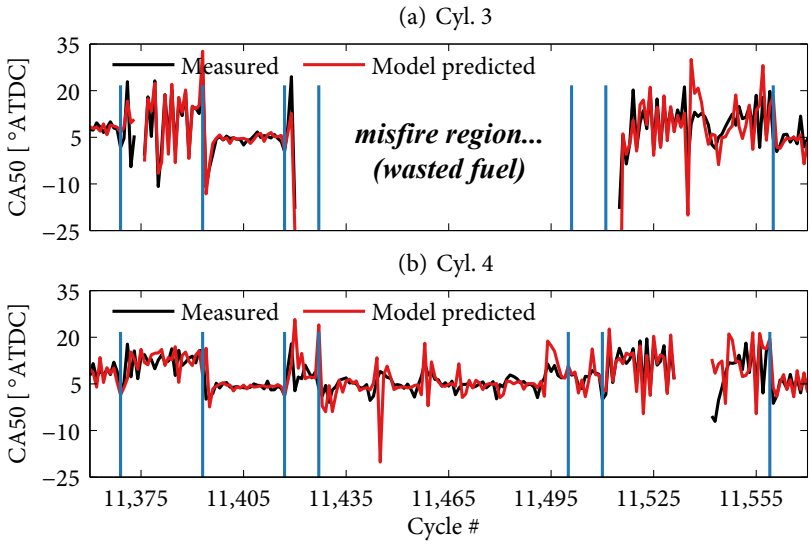


Figure 6.2 Model performance with $\mathbf{W}_0 = 3.5 \times 10^{-6}$ and a ring buffer size of 8.

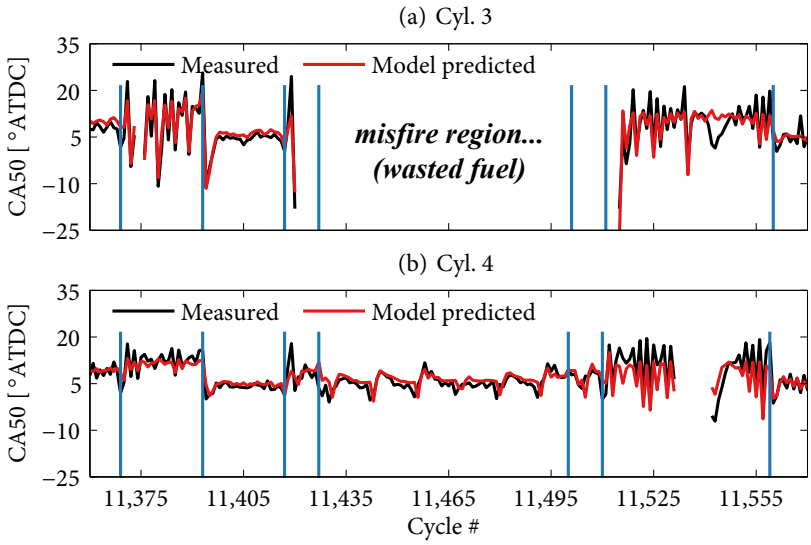


Figure 6.3 Model performance with $\mathbf{W}_0 = 3.5 \times 10^{0}$ and a ring buffer size of 8.

The model sensitivity to the number of neurons is shown in Fig. 6.4. It was found ~64 neurons provided good tracking of high cyclic variability with no significant advantage to using a greater number of neurons. For computational efficiency it is desirable to go with a smaller number of neurons. Below ~64 worked, but at a steep performance penalty. The 64

neurons used in this work was felt to be a good compromise. The value 64 is also generally faster for matrix operations as it is a multiple of the Raspberry Pi® cache line size, a multiple of the Raspberry Pi® floating point vector register bank size, and a power of two (which allows fast bit shifts and masks for memory address calculations).

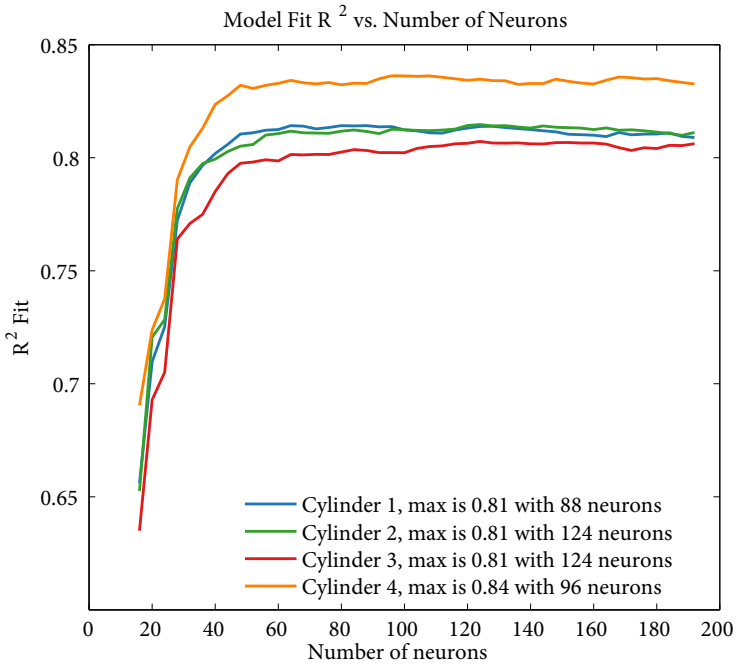


Figure 6.4 Model sensitivity to the number of neurons with $\mathbf{W}_0 = 3.5 \times 10^{-3}$ and a ring buffer size of 8

Fig. 6.5 shows the model sensitivity to the ring buffer size. It is clear that a larger ring buffer can help improve the model fit; however, at the stability limit where misfires occurred, it was found that too large of a ring buffer was detrimental to model tracking performance. This is ostensibly due to the rapidly varying in-cylinder transient conditions (e.g. unmodeled cylinder wall temperature [44]) as combustion is re-established after misfire that produces ring buffer samples not reflective of non-misfiring combustion behavior. Another situation that seemed to be occurring was a cylinder misfiring causing mis-prediction on an adjacent cylinder. Regardless of the source of measurement errors, a ring buffer size of ~8 was found to be a good compromise between better steady-state fitting performance and tracking at the stability limit with occasional misfires. The value 8 is also generally faster for matrix

operations as it is a multiple of the Raspberry Pi® cache line size, a multiple of the Raspberry Pi® floating point vector register bank size, and a power of two (which allows fast bit shifts and masks for memory address calculations).

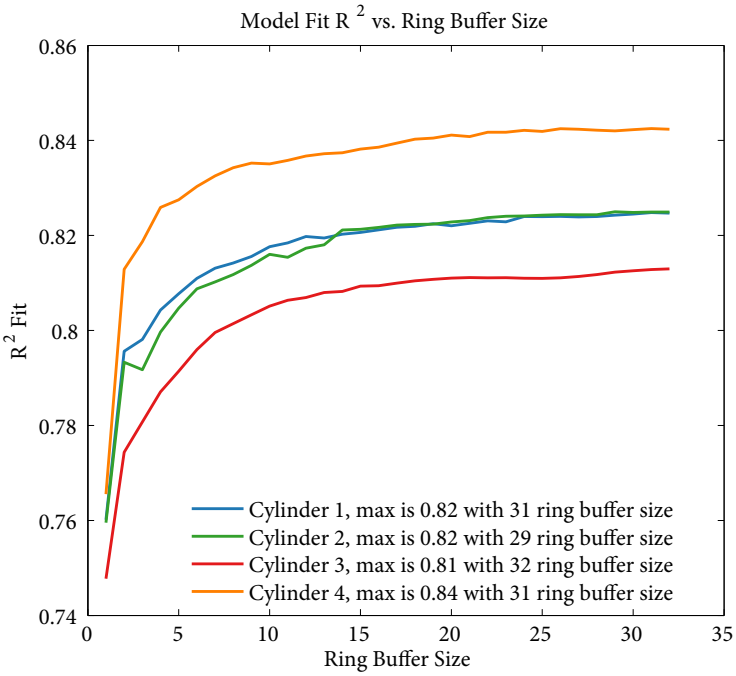


Figure 6.5 Ring buffer size sensitivity with 64 neuron WR-ELM and $\mathbf{W}_0 = 3.5 \times 10^{-3}$.

Fig. 6.6 shows Leave-One-Out (LOO) model sensitivity to removing one of the model inputs. This LOO study is computed by running the model with the same **a** input weight matrix, but selectively zeroing the LOO model input. The figure shows that enabling WR-ELM reduces sensitivity to the loss of a model input, and that CA90 and P_{nvo} are the most significant model inputs. When WR-ELM is disabled (that is, $\beta_1 = \beta_0$) the sensitivity changes dramatically. The most significant input in that situation is P_{nvo} followed by SOI, with the remaining variables in the same order of importance as when WR-ELM is enabled. Evidentially, adaptation makes the model more robust to the loss of an input. Curiously, the model is least sensitive to the loss of fuel quantity (fuel injection pulse width TI) as an input. It is mostly concerned with SOI, combustion timing, and the pressures intended to capture residual and air mixture state.

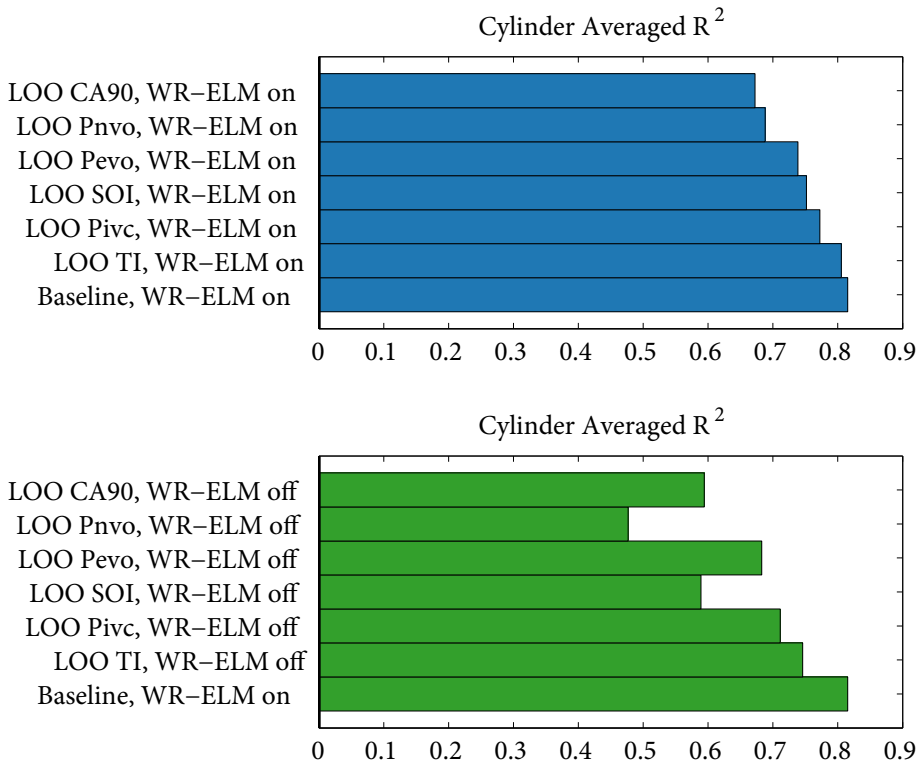


Figure 6.6 Model sensitivity to zeroing the LOO model input.

6.2 Cycle-to-cycle extrapolation

While cycle-to-cycle predictions of combustion phasing have been the main focus thus far, it is particularly interesting to explore the model's ability to extrapolate beyond its current operating point. This is useful to understand what the model "sees" in its inputs along with assessing control authority for quantities such as SOI which can be quickly varied cycle-to-cycle. To do this, one can select cycles (highlighted in Fig. 5.6(a)) and observe the model behavior. To help understand the more complicated high CV behavior, Fig. 6.7 shows the pressure traces for the high CV cycles in Fig. 5.6(a). Note that some apparent NVO heat release is observed on cycle $11382+1$.

Figs. 6.8-6.10 show extrapolation plots for all model inputs for the select cycles highlighted in Fig. 5.6(a). These curves are calculated by setting all but one of the model inputs to the actual measured cycle-to-cycle value, and the remaining variable (x-axis, x) is swept from the minimum to maximum normalized value. The analytical partial derivative against a given

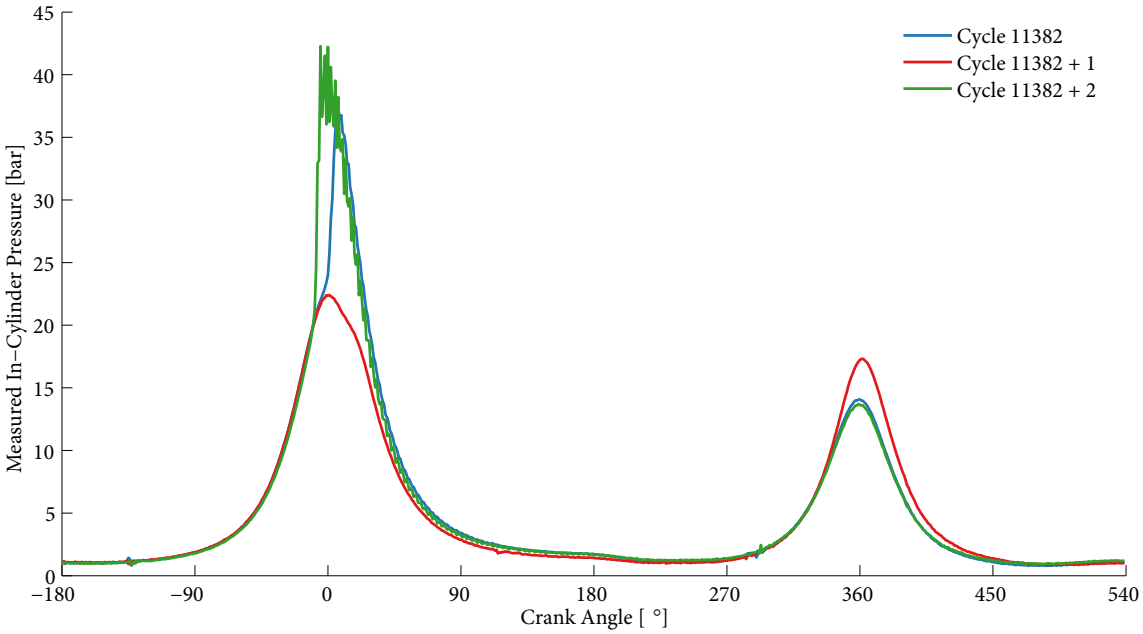


Figure 6.7 Pressure traces for the high CV cycles in Fig. 5.6(a).

input variable x (see Sec. 4.6) through this sweep is also shown for select points along the curve as the red arrows. Fig. 6.8 shows features known to existing physics-based models such as steady-state sensitivity for SOI [45] and a non-minimum phase "charge cooling" relationship for fuel quantity steps (TI) [11]. It's important to stress that this model only needs ~20 minutes engine test cell time to capture known physics for the transient datasets discussed in Chapter 5, whereas traditional steady-state sweeps (cf. [45]) to gather these trends could easily take days of test cell time.

The peak in CA90 in Fig. 6.9 shows the boundary to instability moving at different operating points. Past the peak combustion starts oscillating which can be seen in the high CV cycles. Note that SOI sensitivity in Fig. 6.8 remains the same during high CV cycles, and that the general shapes of all curves do not drastically change from cycle-to-cycle. This supports the hypothesis that it's a single mapping function moving from cycle-to-cycle, and this seems to hold even after model mispredicts (cycle $11419 + 1$). The remaining quantities P_{IVC}, P_{EVO}, P_{NVO} in Figs. 6.9 and 6.10 are "black-box" quantities related to mixture state and composition (see Chapter 3), and are open to interpretation. Of these, P_{EVO}'s extrapolation

shows the highest variability. This is ostensibly due to this measurement's proximity to the end of the combustion process.

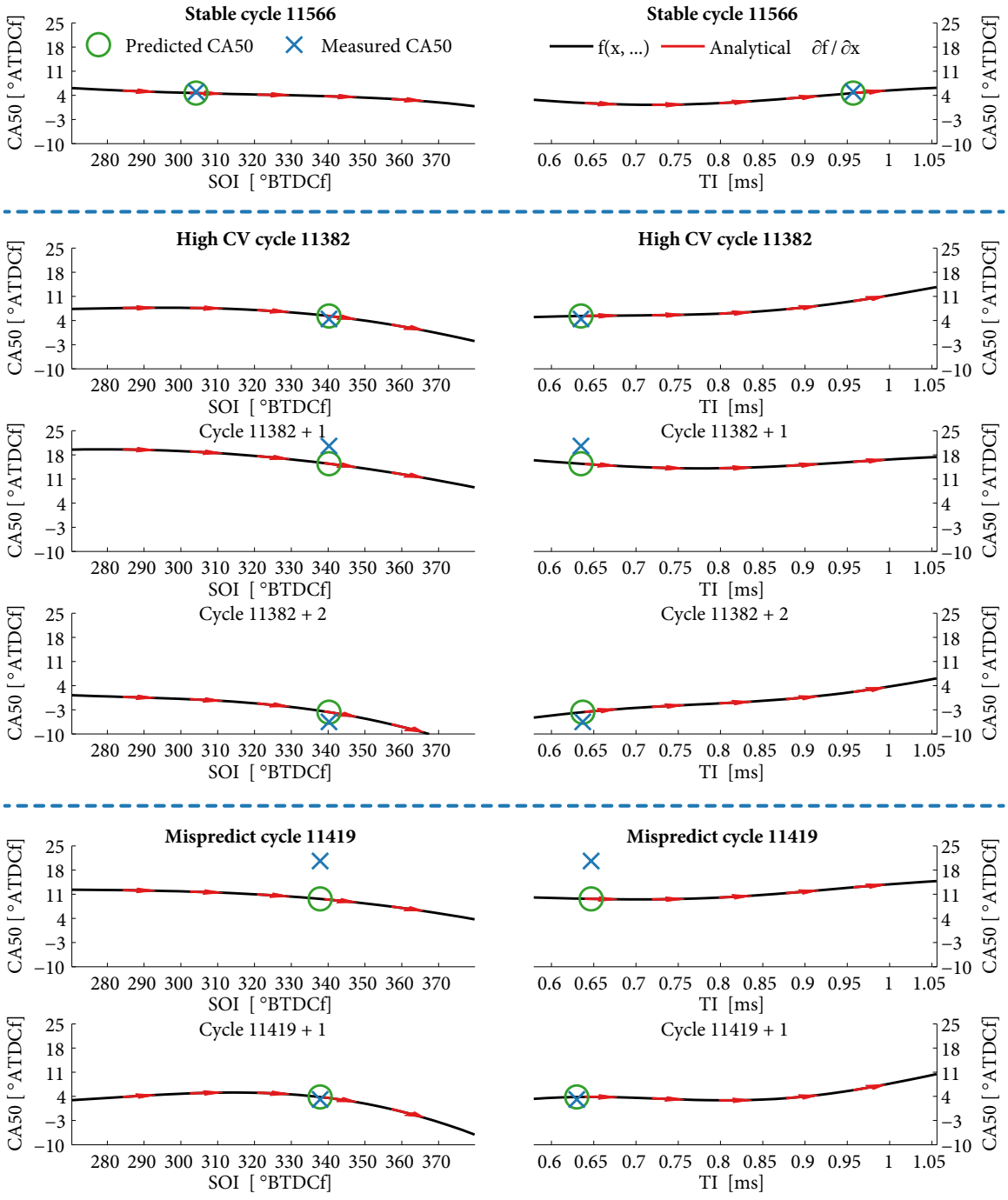


Figure 6.8 SOI and TI extrapolation for select cycles in Fig. 5.6(a).

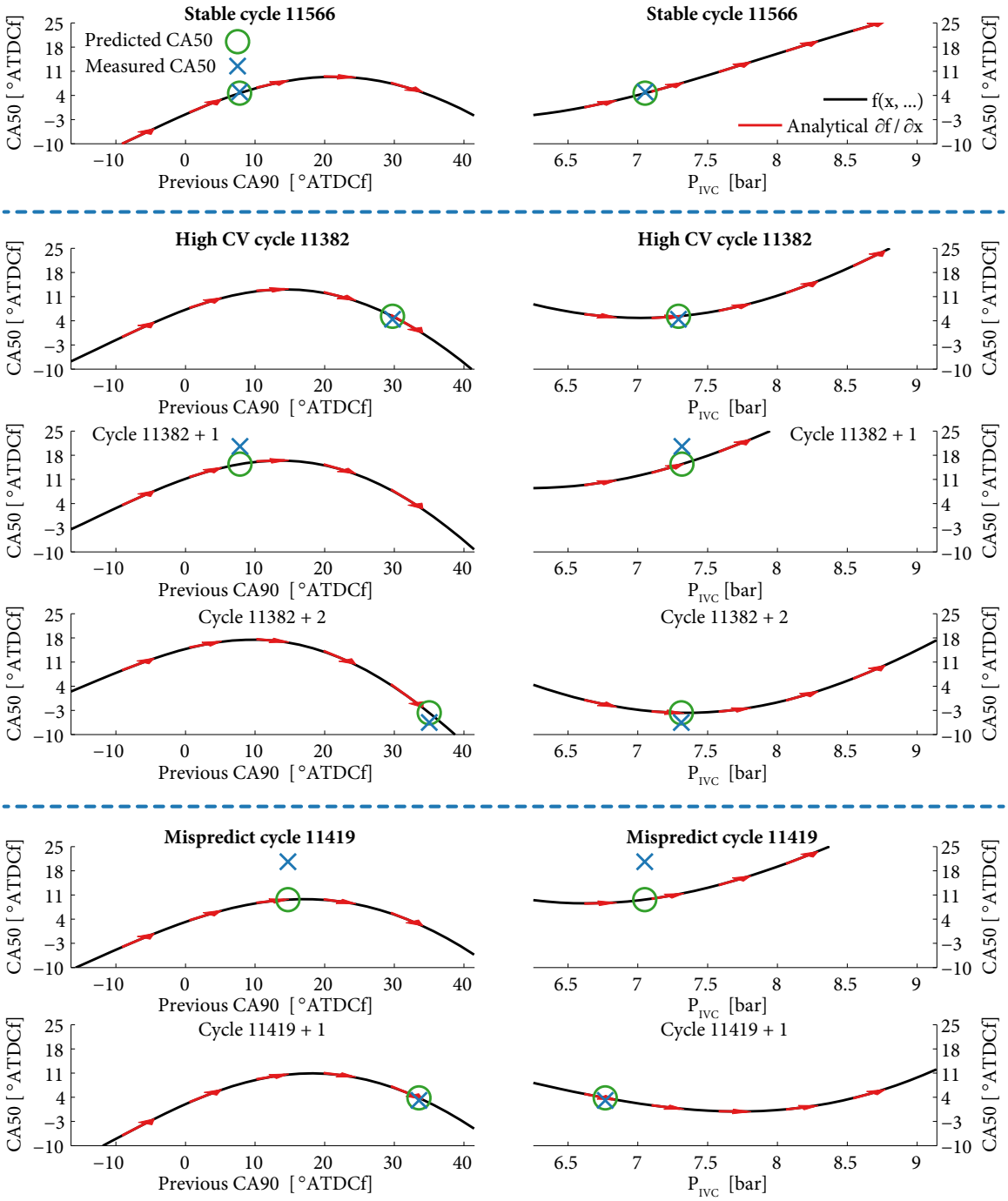


Figure 6.9 CA90 and P_{IVC} extrapolation for select cycles in Fig. 5.6(a).

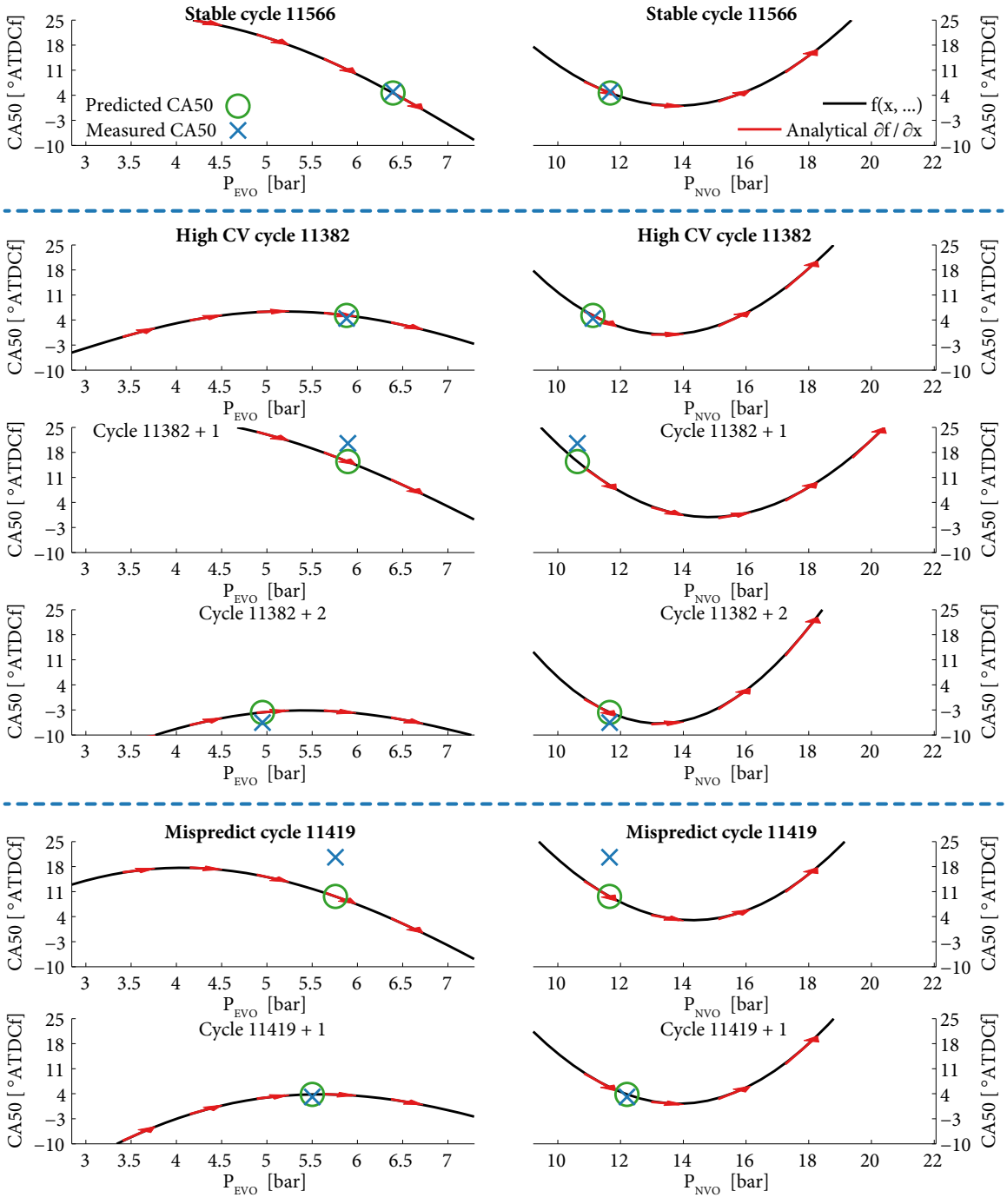


Figure 6.10 P_{EVO} and P_{NVO} extrapolation for select cycles in Fig. 5.6(a).

Chapter 7
Real-Time Implementation

7.1 Overview

The following sections will detail the software and hardware used for a real-time implementation of the mapping function and WR-ELM code. To develop this technology required well over 10,000 lines of custom MATLAB®, C, C++, JavaScript, and assembly code. For quick reference, Fig. 7.1 shows the entire architecture.

7.2 Initial feasibility tests

A collection of *unoptimized* MATLAB® software routines was developed using the techniques described in the previous chapters. The offline solution provided by Eqs. 4.20 was solved at an average rate of 1.1 µs per combustion cycle per cylinder on an Intel® i7 860 2.8 GHz desktop computer running Gentoo GNU/Linux®. The online predictions from Eqs. 4.21, 4.23, & 4.22 were recast into a `parfor` loop that automatically parallelized the code across four worker threads to provide predictions at an average rate of 66 µs per combustion cycle per cylinder. This level of performance is more than adequate for real-time. Additionally, preliminary code experiments showed that the execution time of the core algorithm written in C++ using the Eigen matrix library was fast enough for real-time on a low-cost Raspberry Pi®.

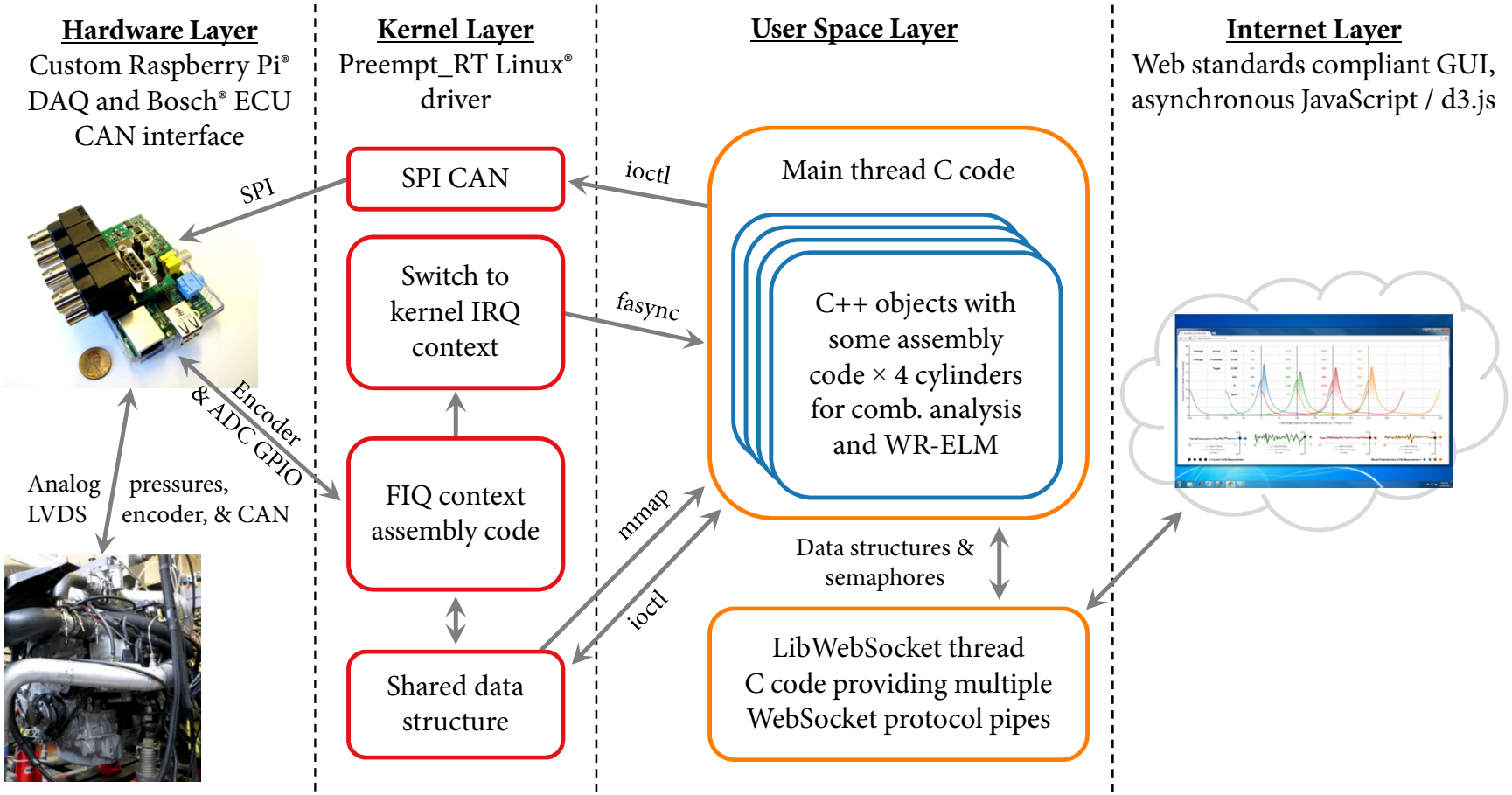


Figure 7.1 The entire hardware and software architecture.

7.3 Real-time response latency

Real-time engine control with the algorithm described in the previous chapters has four main response latency concerns:

1. The time for an operating system to switch tasks.

2. The time needed to read in-cylinder pressure data, process it, and adapt the model.

3. The time to read any remaining in-cylinder pressure data (e.g. P_{nvo}) and to calculate a next engine control command.

4. The time necessary to transmit an engine control command over Controller Area Network (CAN), which is approximately 0.2 milliseconds at a standard speed of 500 kilobits/sec.*

For real-time control, initial experiments were designed to use the Start of Injection (SOI) as a control actuator because it can be quickly varied cycle-to-cycle. As Fig. 7.2 highlights, this creates a short red window which must address latency concerns 1, 3, and 4. This ~500 μs window between the P_{nvo} measurement and the end of SOI command transmission is the most difficult time constraint of the algorithm.† If this window is missed, SOI control over the next combustion degrades substantially (see Fig. 6.8). Efforts to move the P_{nvo} measurement window earlier hurt model tracking performance. Fortunately, while latency concern 2 is the most computationally intensive, it can be computed as soon as the previous cycle's heat release is complete, which is well before the red window (it still needs to be fast as there are four cylinders firing one after another that each need CPU time).

---

* Later versions of the Bosch® ECU code were switched to the maximum rate of 1 megabit/sec.

† While the raw pressure measurements have a must complete in less than ~10 μs, the code for such measurements is not complex and can be easily handled by a special interrupt mode which will be described in later sections.

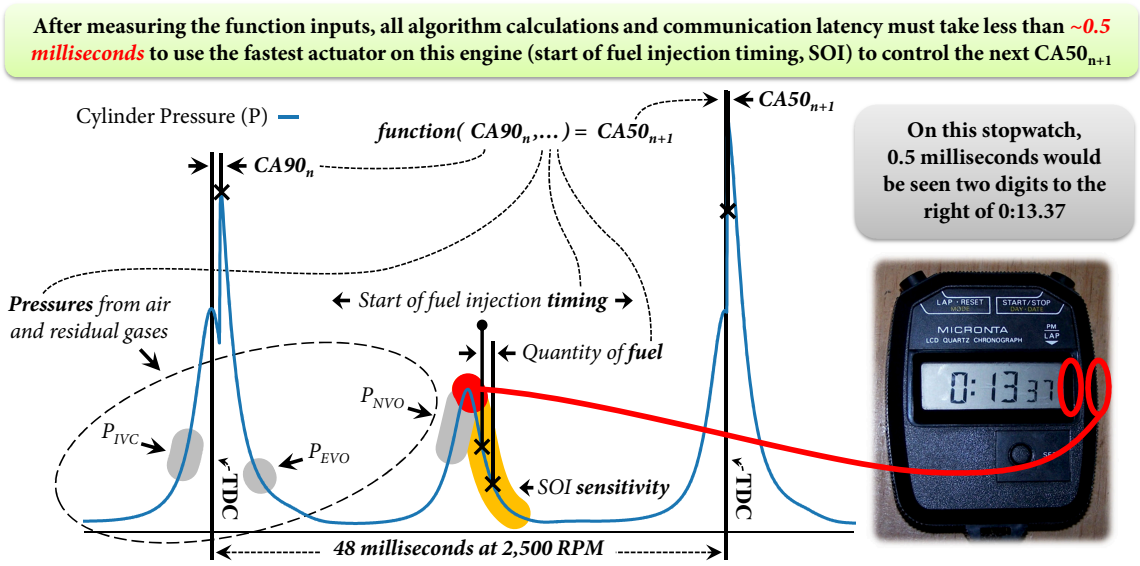


Figure 7.2 Using SOI as a real-time control actuator with the prediction algorithm requires tight computational latency constraints to be satisfied. The stopwatch image has been released into public domain [60].

7.4 Platform choice

Early on in the real-time implementation phase of this project, a consultation was made with an expert in the field of engine control software. He stated that this project's red window (Fig. 7.2) would not be achievable with the engine test cell's existing ETAS® ES910 rapid software prototyping system. As such, it was necessary to look at alternatives. The author ultimately decided that the *guaranteed*-to-work solution would be to implement the system from scratch.‡

Based on the approximate floating point operations per second (FLOPS) needed, a Raspberry Pi® was determined to have sufficient computing horsepower. The Raspberry Pi® is essentially an iPhone® generation 1 class processor (specifically, an ARM 1176JZF-S). There are other, more recent single board computers similar to the Raspberry Pi®, many with greater overall performance; however, these alternatives do not necessarily have faster double

‡ This was not particularly difficult. Prototype pressure data acquisition and CAN circuitry was fabricated and functional less than two months after the real-time project's start.

precision FLOPS over the older ARM VFPv2 floating point co-processor on the Raspberry Pi®. A desktop PC is another alternative, but it unfortunately has more complicated hardware I/O and potential latency issues with system management interrupts (SMI) [61]. Other alternative solutions such as a dSPACE MicroAutoBox® and National Instruments Drivven® systems were considered; however, there was uncertainty as to whether they would meet Fig. 7.2's red window with non-trivial double precision matrix algebra along with concern about lost time and monetary cost if they proved too slow.

Amongst the other single board computers, the Raspberry Pi® was ultimately chosen because it has one of the largest (if not the largest) community of support, with over 3 million units sold. This community greatly reduces the effort to build a real-time engine control system because the community has already solved many implementation challenges.

7.5 Real-time operating system

An important decision was to run the engine control system under the GNU/Linux® operating system. The primary benefit was that GNU/Linux® provides a standard, free UNIX®-like operating system, with existing ecosystem of tools and libraries that greatly simplify software development. The caveat is that the Linux® kernel is designed for throughput, and does not guarantee any real-time latency constraints will be met.

There are various solutions to this latency issue, and two main, free solutions are Xenomai and Preempt_RT. Xenomai can achieve an execution latency of $<\sim$50 µs and Preempt_RT $<\sim$150 µs on the Raspberry Pi®. Xenomai achieves lower latency through a small co-kernel that has higher priority than the Linux® kernel. While slower, Preempt_RT programs are still standard Linux® programs, and this has compatibility advantages. Fig. 7.3 shows the distribution of observed process latencies under Preempt_RT on the Raspberry Pi®.

Xenomai was the real-time Linux® solution used for most of this project's development; however, the patches needed for the Raspberry Pi® were ultimately deemed not production

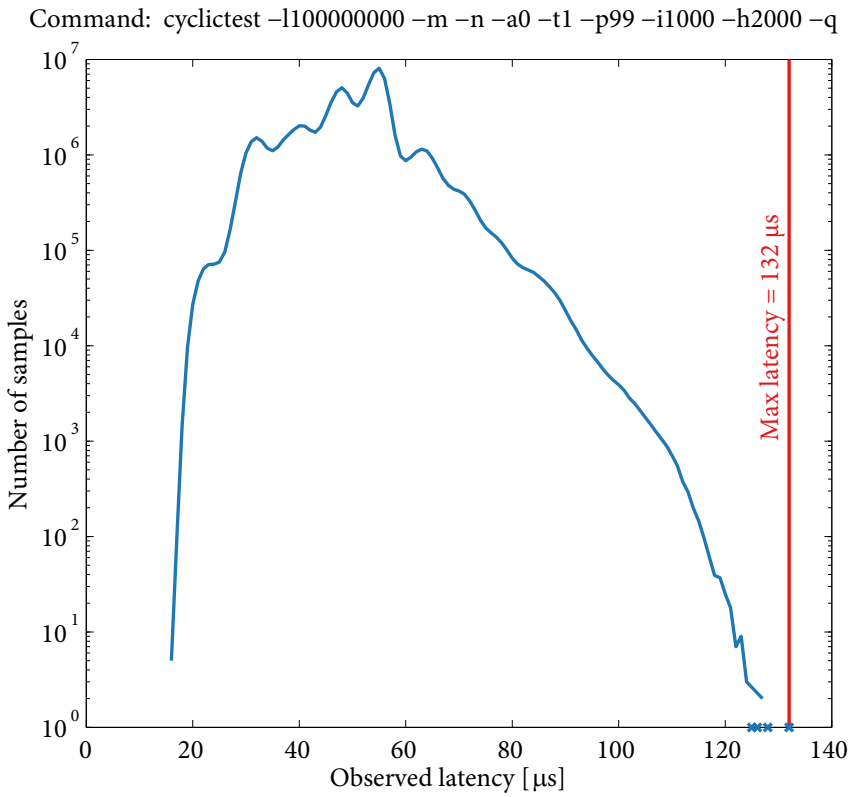


Figure 7.3 A ~28 hour latency test while acquiring 8 simultaneous channels worth of 18 bit pressure data at 32 kHz (256 kilosamples per second, total) with the custom data acquisition hardware developed in this work, under moderate CPU load, with continuous network and SD card storage access.

ready. A primary issue was the Universal Serial Bus (USB) implementation on the Raspberry Pi®. Even when there's nothing to process, the USB hardware will throw an Interrupt ReQest (IRQ) event for every USB frame. This can be very resource intensive (up to ~20% CPU time) when it occurs at 8,000 times a second under USB 2.0 [62]. The Raspberry Pi® community's solution to this hardware "feature" was to rework the USB driver to use a Fast Interrupt reQuest (FIQ), which is an ARM specific higher priority interrupt.

Like an IRQ, FIQ code is event driven code execution (e.g. a pin changing voltage marking a crank angle pulse or a new USB frame); however, it executes independently and with higher priority than the main Linux® kernel. This avoids the overhead that Linux® needs to perform for regular IRQs that can come from any source, with no native ARM support for IRQ priorities. Using the FIQ for the USB driver ostensibly allows the driver to quickly check

whether the USB hardware actually has something to process, or if it's just firing an interrupt for a throw away USB frame.

Unfortunately, the in-cylinder pressure data needs to be measured at a non-trivial data rate.[§] The FIQ offers one of the best ways to get these pressure data into and processed by the Raspberry Pi® with guaranteed real-time determinacy at this data rate. This is because the code path is only the FIQ code itself, and the FIQ has its own dedicated registers that lower the context switch overhead from whatever code the processor was executing before the FIQ. The caveat is that the USB driver cannot easily continue to use the FIQ.

When reverted to a regular IRQ, it was observed that the USB driver would stall the system for upwards of 15 milliseconds under heavy USB Ethernet network usage, which is wholly unacceptable given the ~500 µs time constraints outlined in Fig. 7.2.[¶] The recommended way to resolve this issue would be to run Xenomai with a Preempt_RT kernel and forcing the USB driver thread to a low priority. However, Xenomai for the Raspberry Pi® targets an older, out-of-tree Linux® kernel that the Preeempt_RT patch was not easily applied to. The easier solution was to sacrifice ~100 µs of latency, stop using Xenomai, and use a stable, tested Preempt_RT Linux® kernel with various patches which were previously made available at [63]. To the best of the author's knowledge, there is no similarly well tested Xenomai patch for Raspberry Pi® Linux®. It is also worth mentioning that the Preempt_RT solution still requires patches to the USB driver to correct for USB code assumptions that assume nothing else is using the FIQ, leading it to occasionally disable the FIQ code used for pressure data aquisition.

An alternative that was considered was to use an external microcontroller help buffer the real-time data into larger, lower frequency chunks; however, this would add additional synchronization demands to the software and require managing two separate code bases for both the Raspberry Pi® and the microcontroller. The data also would still have to be streamed

---

§ 30 KHz at 2,500 rpm, 0.5 crank angle resolution, 4 cylinder channels, at 18 bit resolution is 2.2 megabit/s, or more at higher engine speeds, with a greater number of cylinders, or at finer resolution.

¶ The observed delays are the motivation for testing the kernel in Fig. 7.3 continuous network usage.

over the same General Purpose Input Output (GPIO) or Serial Peripheral Interface (SPI) pins as the non-microcontroller solution to maintain real-time, and as such a microcontroller didn't offer much in the way of advantages. It was just an additional middleman. Although the data-rate could be easily handled by USB, the USB standard provides none of the real-time guarantees needed for this project and, as noted earlier, the existing driver was unreliable for real-time (e.g the spurious ~15 millisecond system stalls under heavy USB Ethernet network usage).

A possibility that could improve the situation would be pre-processing the data (e.g. running heat release analysis) on a microcontroller to avoid sending all the pressure data. However, microcontrollers do not typically have native floating point arithmetic, and must emulate it at a significant performance penalty. There was some question as to whether a microcontroller would be fast enough for this task. It would also likely require sacrificing the quality of the graphical display of pressure data to keep the data bandwidth requirements low. Thus, it was ultimately decided to use the FIQ approach because it offered lower complexity on both the hardware and software side.

7.6 Fast Interrupt reQuest

As discussed in the previous section, the FIQ was chosen to read real-time in-cylinder pressure data, streamed from an Analog to Digital Converter (ADC). This ADC (described in later sections) offers two data transmission methods, parallel through GPIO pins or SPI. SPI was initially used because it is simpler to implement, but a later hardware revision switched to the parallel GPIO interface because higher data rates were possible and it completely decoupled the ADC data path from the CAN chip which also used SPI.

Using the parallel interface incurred an additional expense of more FIQ processing overhead. The overhead stems from the need to actively drive the parallel bus read strobes since the Raspberry Pi® does not have a dedicated parallel Input Output (IO) peripheral (found

in some microcontrollers) and because the limited Raspberry Pi® GPIO pins necessitate the use of a multiplexer to convert the ADC's 16 bit bus to 8 bit. Another overhead is from the fact that Raspberry Pi® GPIO pins are not in consecutive order, and as such the code needs to appropriately sort which bit is on which pin (this presently requires two instructions).

The FIQ responds to two events (1) the falling edge of the ADC's BUSY pin and (2) the rising or falling edge of the engine encoder's trigger signal that marks one full rotation. When it executes, it determines which of these two events has occurred and appropriately adjusts a data structure that is used to track crank angle position for all 8 recorded channels. The code and data structure support the ability to offset the start angle of a full cycle measurement by any number of crank angle degrees. For the four-cylinder engine used in this work, the offset is used to start each cylinder's measurement with a 180° offset consistent with the mechanical offset inherent to the crank shaft.

Once the measurement angle is determined, it is used to determine the data array index that the pressure data sample will be stored. Each channel has two arrays, one working array that is being populated with each encoder pulse and another that contains the entire previous engine cycle. Once an array is filled, the FIQ code swaps the working and complete arrays (through a bit field that simply exclusive ors (xors) in a bit to toggle the arrays). This working and complete array logic was needed when all of the machine learning code ran in its entirety at once; however, it is presently unnecessary as the current code spans two separate executions. These two executions are (1) adapt the model and (2) calculate a control command (Fig. 7.2's red window). Each of these two executions operates on a copy of the working array, which is maintained as part of the conversion from signed integers from the ADC to floating point pressure values (which are thermodynamically pegged cycle-to-cycle).

The final bit of functionality that the FIQ code provides is notification when a specific crank angle has been achieved for any of the offset channels. This is used to notify a Linux® user space application that that it is time to process one of the two code executions (adapt or control). To do this, it is necessary to switch from FIQ context to the Linux® kernel context.

One of the simplest ways to do this is to fire an IRQ by enabling the ARM timer (unused by the standard Raspberry Pi® Linux® kernel), which is later processed by Linux® and asynchronously notifies the user space application through Linux®'s standard fasync interface. All of the above FIQ software is coded in assembly code for performance reasons (e.g. to take advantage of the dedicated FIQ registers).

7.7 Linux® user space and web-based user interface

For a variety of reasons, floating point code within a Linux® kernel driver is generally considered bad form, and as such all combustion analysis and machine learning code is within a user space application. Linux®'s standard mmap interface is used to map the FIQ pressure data buffers to user space for performance reasons. Control and coordination with the FIQ and kernel driver is accomplished through the standard Linux® ioctl interface, and (as mentioned earlier) crank angle notifications that the user space application subscribes to from the FIQ arrive over the standard fasync mechanism.

This user space code is structured as two C code threads, one for pushing processed pressure data and results over a standard WebSocket (using the libwebsocket library) to a client computer's JavaScript user interface (coded with the d3.js library) and the main thread. Once an fasync signal arrives, C++ code is called to either (1) run heat release calculations and adapt the model or (2) calculate an engine control command based on the previous adapted model and recently completed input vector which was waiting on the P_{nvo} measurement. Each cylinder's state and model is encapsulated as an individual C++ object. The split in fasync code paths allows the more computationally intensive adaptation routine to be performed before the red region of Fig. 7.2. To further reduce the computational burden of model adaptation, a custom assembly code ARM VFPv2 matrix multiply was developed which was benchmarked to be 29% faster than OpenBLAS's VFPv2 GEneric Matrix Multiply (GEMM) and 126% faster than Eigen C++.

After the engine control command is calculated, it is sent to the Linux® kernel driver over a standard ioctl call. The full calculation results and pressure data are then loaded into a data structure for the libwebsocket thread to process and asynchronously push to a client computer that subscribes to the data stream. Independent WebSocket protocol pipes are used for each cylinder's data, with an additional pipe used for user interface commands transmitted back to the Raspberry Pi®.

7.8 Hardware

There does not exist a Raspberry Pi® add-on board that provides all the necessary features for fast, real-time engine control needed by this project. As such, it was necessary to develop the circuitry (shown in Fig. 7.4). The author's prior experience made this one of the most straightforward aspects of this dissertation.

As has been partially described in the previous sections, the circuitry provides:

- 8 analog inputs with selectable ± 10 V or ± 5 V input ranges.
- A high resolution (18 bit) ADC with both analog and digital filtering for all inputs.
- A precision voltage reference for the ADC.
- A high noise immunity Low-Voltage Differential Signaling (LVDS) encoder interface for individual crank angle marks and a trigger pulse mark every crank rotation.
- A CAN interface to send or receive commands to external equipment (e.g. an existing engine controller).

The multiple analog input channels allow the device to interface with multiple in-cylinder pressure transducers on a multi-cylinder engine. The device is not limited to common piezoelectric in-cylinder pressure sensors (which are expensive and require an external charge amplifier); it is flexible enough to also read fiber optic pressure sensors (e.g. from Optrand, Inc.), piezoresistive pressure sensors, knock sensors, ion current sensors, etc. Beyond measuring pressure, the analog channels can log injection pulse width, timing, and

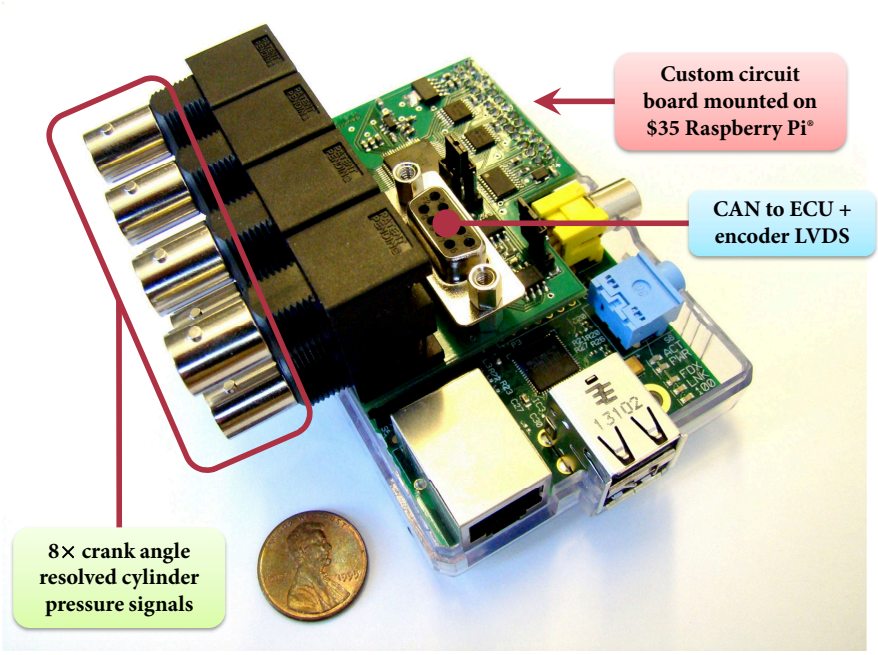


Figure 7.4 Custom real-time circuitry developed for this dissertation.

injector needle valve lift to provide closed loop metering of fuel mass instead of only relying on injection pulse width TI to quantify fuel mass in the abstract cycle-to-cycle mapping function framework.

Finally, in combination with above software latency tests (Fig. 7.3), it was also important to externally verify the software was functioning as expected. This was done with an oscilloscope connected to main input and output connections of the circuity. Fig. 7.5 shows a typical infinite persistence oscilloscope trace triggered on a TDC signal to verify CAN packets were being sent within the time constraints shown in Fig. 7.2. Infinite persistence means that the trace shown is the sum of all observed traces, and it is useful for tracking outliers (e.g. a CAN packet that was delivered too late for real-time).

In Fig. 7.5, a total of 11 model predictive control iterations (from multiple start points detailed in Chapter 8) were preformed between TDC and the start of the CAN packet transmission. In the worst case, the CAN packet is fully transmitted by 0.6 milliseconds after TDC, which is close to the latest the packet can be sent. This worst case latency can be reduced by performing fewer predictions since the control algorithm typically converged

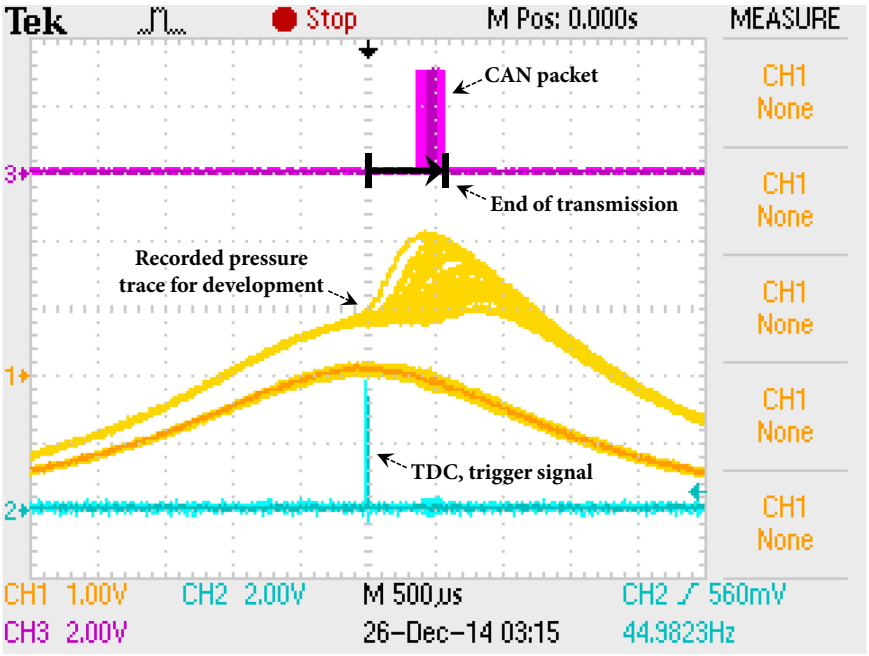


Figure 7.5 This figure shows an infinite persistence oscilloscope trace for verifying CAN packet latency constraints are met.

within less than four iterations, optimizing the code further, or increasing the CPU or RAM speed beyond stock settings. However, this is unnecessary with the current Bosch® ECU software version.

Other situations where the oscilloscope was useful were:

- Identifying that heavy USB Ethernet network usage causes immense (~15 milliseconds) execution delays stemming from the Raspberry Pi® USB driver. This was fixed by manipulating Preeempt_RT thread priorities.
- Finding bugs in the Raspberry Pi® USB IRQ driver that caused the data acquisition FIQ code to be periodically disabled. This was fixed by patching the USB driver code.
- Verifying the FIQ data acquisition code was fast enough to read real-time pressure data.

Chapter 8
Feasibility of Engine Control

8.1 Overview

The previous chapter described in detail the low-level software and hardware developed for this thesis. Fig. 8.1 shows the web-based user interface with the engine set to an operating point that caused high cyclic variability for cylinder 2 (the rightmost, yellow trace). Real-time predictions are shown along the bottom of Fig. 8.1 with the predictions marked by colored dots. Note that all the cylinders behave differently despite the fact that they're receiving the same actuator set points. They all have the same SOI, TI, and valve timings.

Fig. 8.2 shows typical results when cylinder control is enabled. Note that all the cylinders balance to their set point of 9 °ATDCf. Highlighted in Fig. 8.2 is the fact that cylinder 2 is being measured and controlled with an Optrand® fiber optic pressure sensor (the same sensor is used in Fig. 8.1). At production volumes, this sensor could be produced for as little as $13 per cylinder (personal communication, May 21st, 2014). Perhaps with more calibration and control design effort (see Sec. 8.2 and 8.3) the small control tracking errors seen in Fig. 8.2 can be mitigated. For the remainder of this chapter, all measured cylinder data will use the same water-cooled Kistler® 6041A pressure sensors installed in the engine head (not spark plug).

A video that demonstrates cylinder balancing from an operating point that exhibits near chaotic behavior when uncontrolled is available at [64].

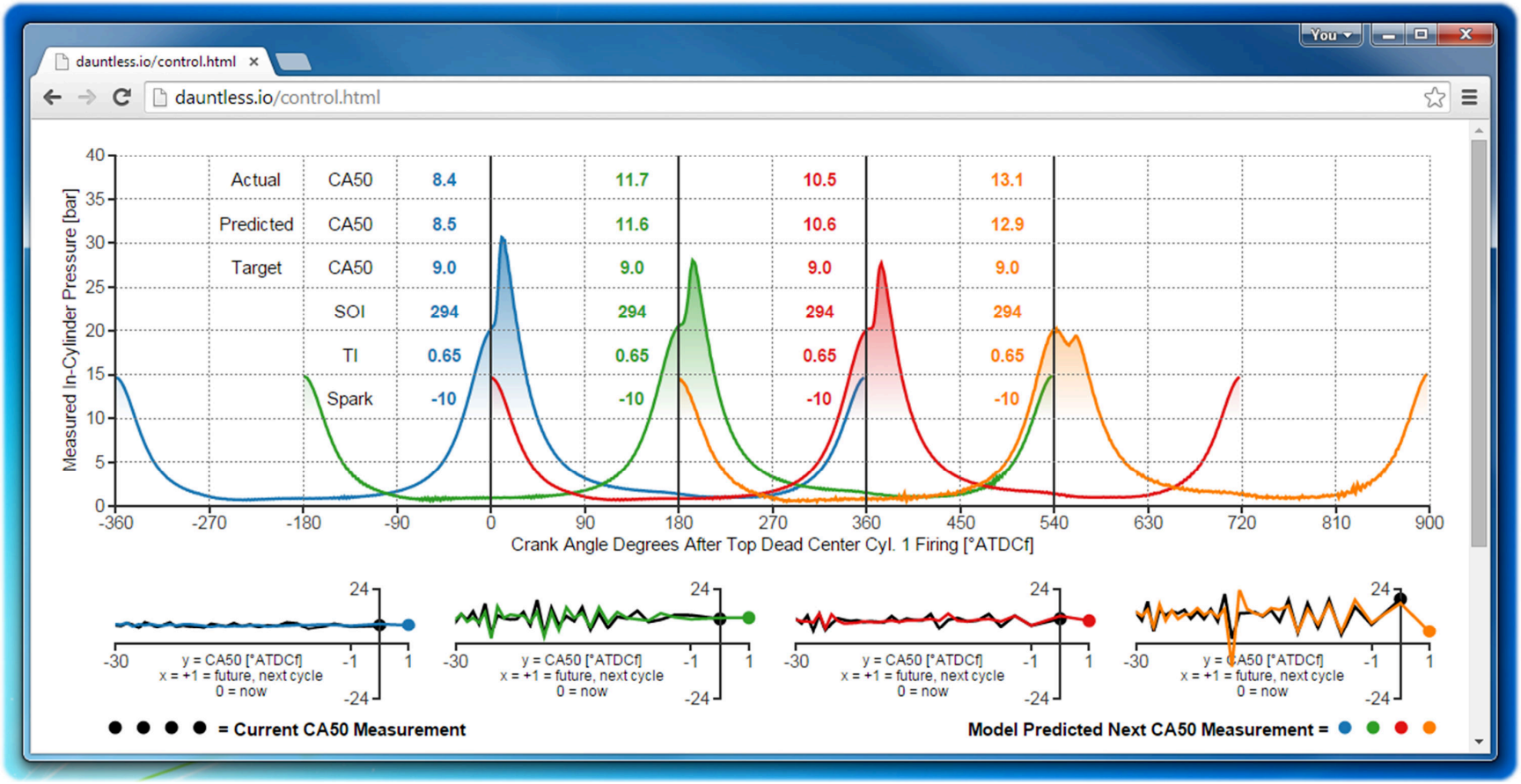


Figure 8.1 A screenshot of the custom software developed for this dissertation. Real-time predictions are shown while engine control is disabled.

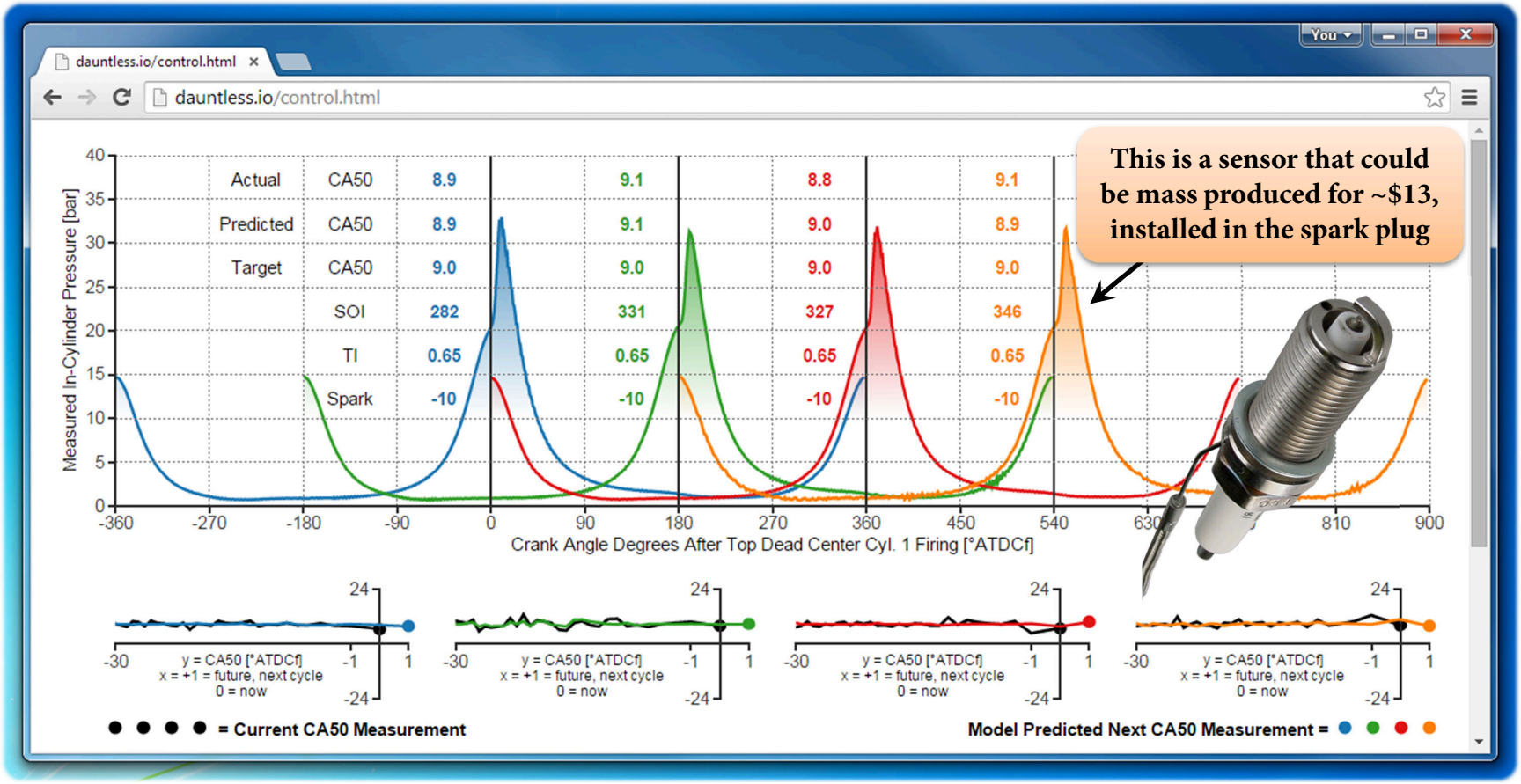


Figure 8.2 Control enabled using Eq. 8.1 for a set point of 9 °ATDC firing. The overlaid spark plug pressure sensor image was provided by Optrand® Inc.

8.2 Calibration

As is the case for all results shown in this chapter (and in the video at [64]), the model was trained using only cylinder 1 data not from the present engine, but from a similar (though not identical*) engine installed at the University of Michigan three years prior to these control experiments. The training is identical to what is described in Sec. 5.4. This was done due to project time constraints and to emphasize that the data-driven WR-ELM mapping function model has generalized actual combustion physics and is not engine *or* cylinder specific. The same model is used for all cylinders.

8.3 Control law

To demonstrate proof of concept control with the real-time implementation, a simple one-step-ahead Model Predictive Control (MPC) law was explored with SOI as a control actuator:

$$f(\mathrm{SOI}) - \text{target } CA50 = 0 \tag{8.1}$$

where $f(\mathrm{SOI})$ is the cycle-to-cycle adapted WR-ELM mapping function CA50 prediction for the current input vector as a function SOI. Eq. 8.1 was solved with Newton's Method:

$$\mathrm{SOI}_{n+1} = \mathrm{SOI}_n - \frac{f(\mathrm{SOI}_n) - \text{target } CA50}{f'(\mathrm{SOI}_n)} \tag{8.2}$$

Out of concern for algorithm convergence within a finite, real-time computation window (see Fig. 7.2), three initial SOI_n values were used. These initial values were the previous SOI value, ¼ the normalized SOI range, and ¾ the normalized SOI range. Also out of concern for convergence, the final selected SOI value was not the final iterated value. The

* The engine under test has a new decked head, block, connecting rods, crank, and an additional Exhaust Gas Recirculation (EGR) loop. EGR was disabled for the experiments of this chapter. The engine parameters for the new engine are similar to those described in Table 5.1, but there has been some question as to whether the compression ratio is identical.

selected SOI value was whichever iteration most closely satisfied Eq. 8.1. Experimentally, these convergence concerns did not appear to be an issue, but the multiple initial values code was active for all experiments. SOI actuator constraints were enforced with simple saturation; however, future work should include the constraints in the MPC optimization. For performance reasons, the above control laws were programmed in C code that manually traversed the WR-ELM matrices in a fused loop that calculated the analytical SOI derivative (Eq. 4.29) and predicted CA50 (Eq. 4.28).

8.4 Control feasibility

As a proof of concept, Figs. 8.3-8.7 demonstrate the feasibility of combustion phasing control with Eq. 8.1. The results are measured by a third party AVL IndiSet® 642 data acquisition system. It should be mentioned that only an SOI of up to 346 °BTDC firing is possible with the current Bosch® ECU firmware, and this limits SOI authority over the next cycle's CA50. Fig. 6.8 shows a gain in control authority if SOI could be moved up to ~357 °BTDC, which is computationally feasible with the present real-time hardware and software. In Figs. 8.4-8.7, cylinder 2 remains in a region with the most SOI control authority (away from the lower SOI limit), and thus its performance is highlighted with the dashed ovals.

Fig. 8.3 shows each controller's ability to balance cylinders to a similar combustion phasing. Fig. 8.4 shows the controller's ability to control combustion phasing under large (29% increase) injection pulse width transient steps. Note that the Air-Fuel-Ratio (AFR) varies from 1.19 to 0.99 during these transient steps and that CA50 moves significantly without control for cylinders 2 and 3. Fig. 8.5 shows the controller's ability to hold CA50 under large (13 °CA) exhaust valve timing transient step. CA50 moves significantly when control is disabled. Cylinder 2 shows the largest variation in CA50 when control is disabled. When control is enabled, cylinder 2 holds its set point, but does show a large oscillation in CA50 around cycle 378 that is quickly corrected. As one would expect, controller performance is degraded

when the SOI actuator signal is saturated during all these trials.

While engine speed dependence in the mapping function model was dropped in Eq. 3.12, it is important to understand whether this model can adapt around engine speed variations. Fig. 8.6 shows the controller's response to a large speed ramp from 1,750 rpm to 2,500 rpm and back. It is clear that the controller can reject the engine speed transient up to the point where SOI becomes saturated (Fig. 8.6g).

Finally, Fig. 8.7 shows the controller's ability to reduce cyclic variability for cylinder 2 with the controller set point near the uncontrolled mean CA50 value. This was the best case with a CA50 standard deviation reduction of ~60%, but the mean has moved earlier 0.7 °CA, which influences the measured reduction. From multiple trials, the smallest observed standard deviation reduction was ~10%. More work needs to go into quantifying cyclic variability reduction with WR-ELM mapping function control beyond this proof of concept (ideally with an engine specific calibration because near chaotic combustion is very sensitive to parameter variations).

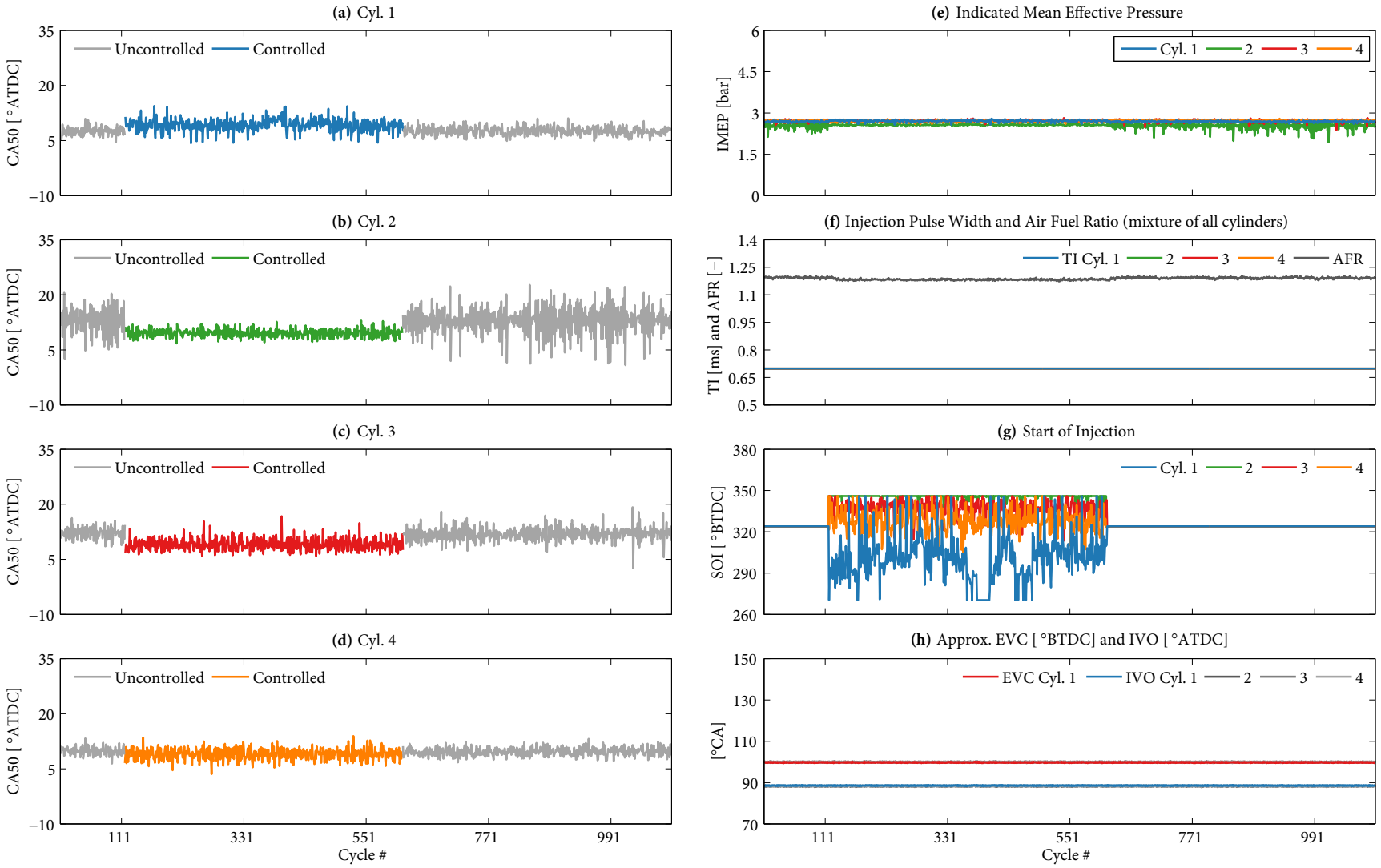


Figure 8.3 Control enabled to balance cylinders to a similar combustion phasing at 2,500 rpm.

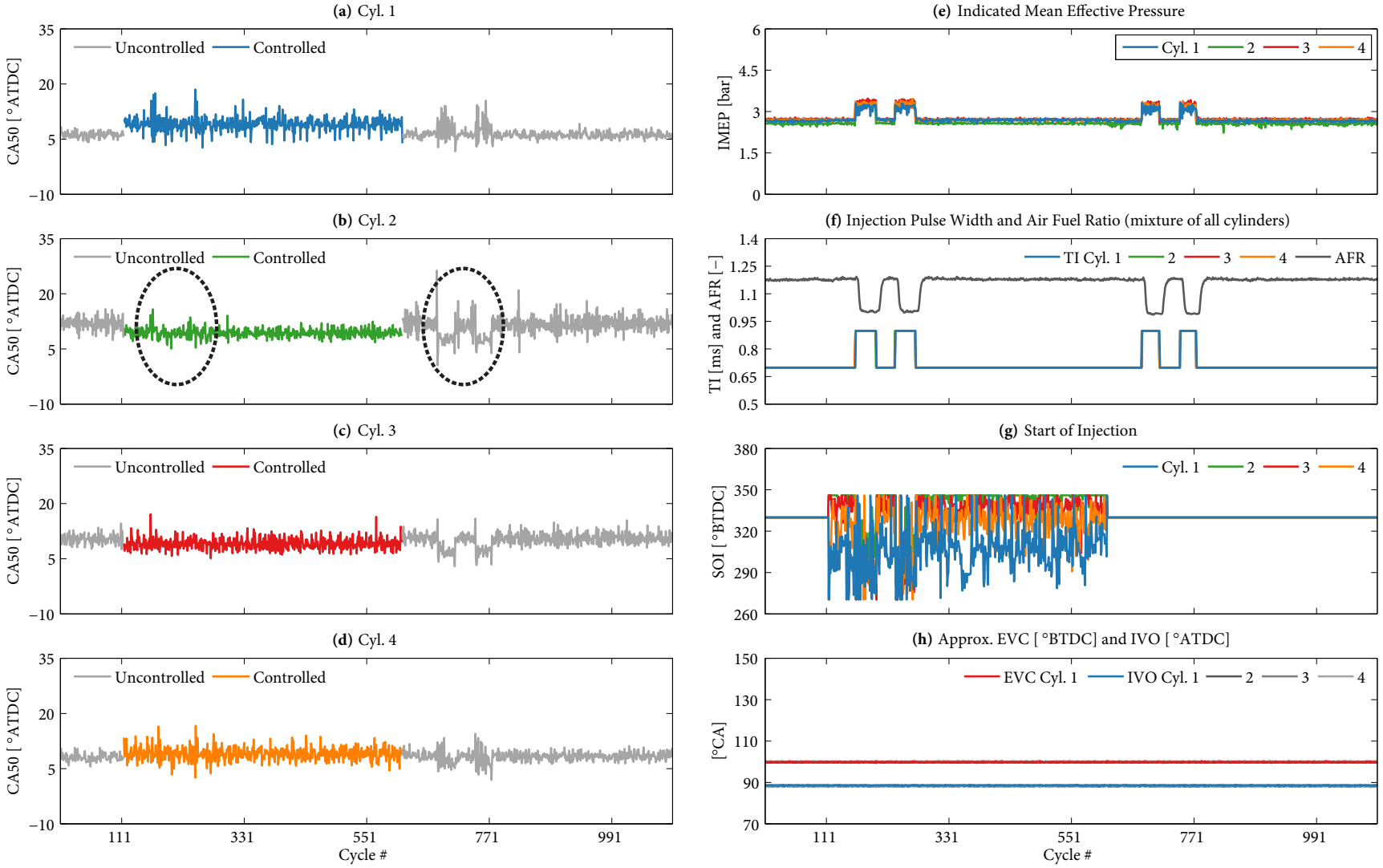


Figure 8.4 Controller performance at 2,500 rpm with a large (29%) increase in injection pulse width (TI), see subplot f. Cylinder 2 remains in a region with the most SOI control authority (away from the lower SOI limit), and thus its performance is highlighted with the dashed ovals.

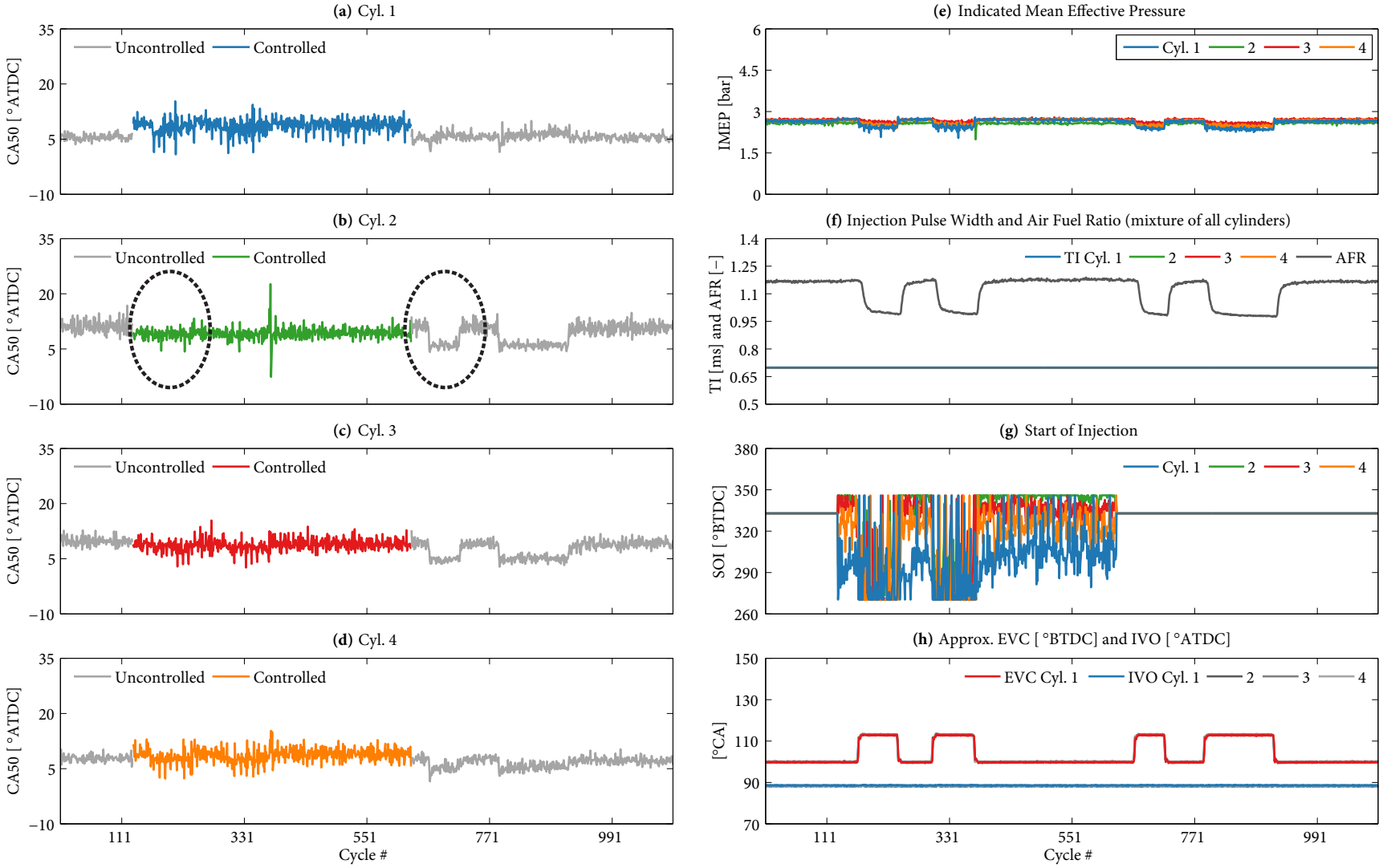


Figure 8.5 Controller performance at 2,500 rpm with large Exhaust Valve Close (EVC) steps from 100 to 113 °BTDC, see subplot h. At this operating point, this was the largest open loop EVC step possible. Cylinder 2 remains in a region with the most SOI control authority (away from the lower SOI limit), and thus its performance is highlighted with the dashed ovals.

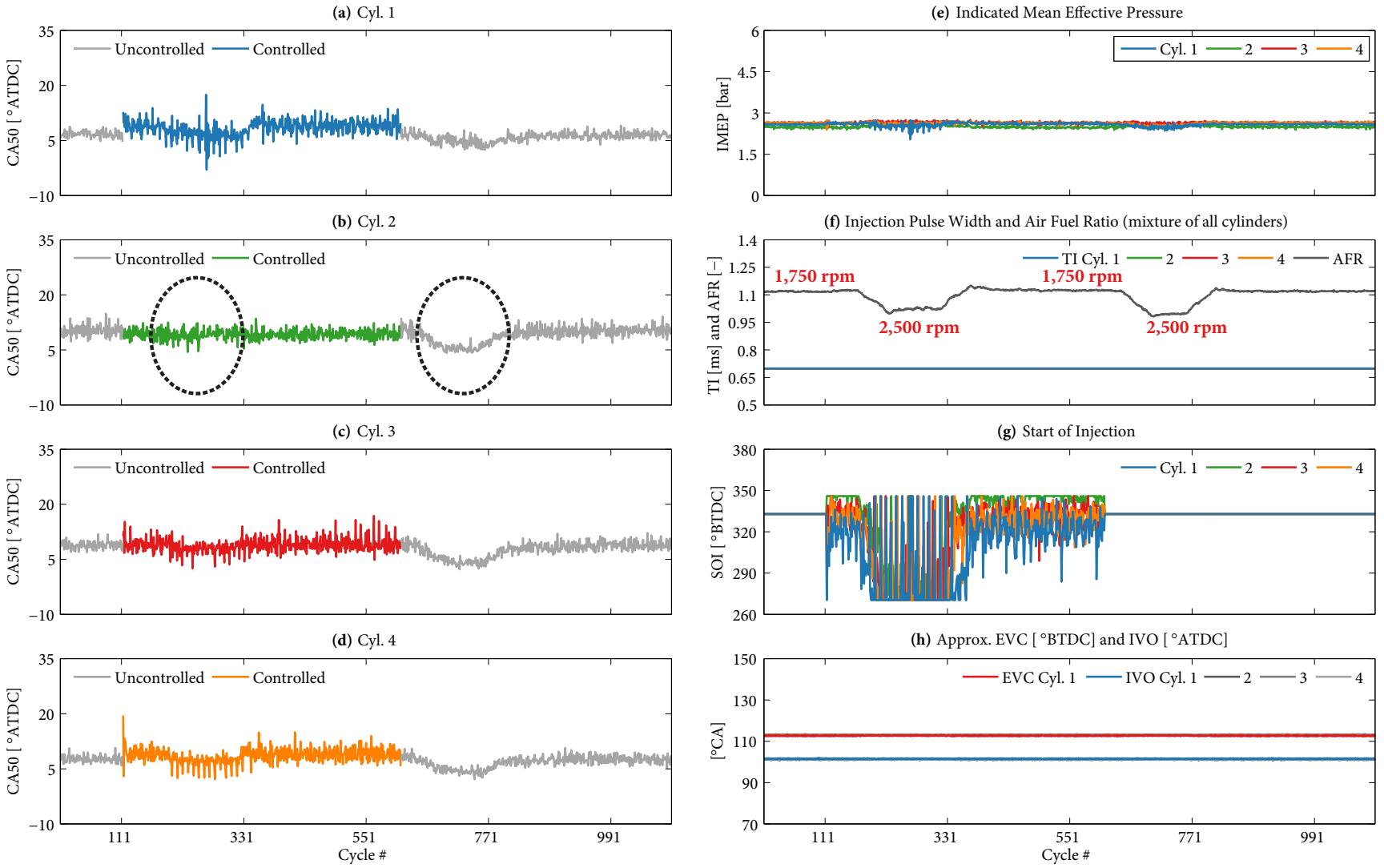


Figure 8.6 Controller performance with large engine speed ramps from 1,750 to 2,500 rpm and back are shown in each segment. Note that there is no engine speed in this model — predictive control outside the model's original design is being achieved with adaptation. Cylinder 2 remains in a region with the most SOI control authority (away from the lower SOI limit), and thus its performance is highlighted with the dashed ovals.

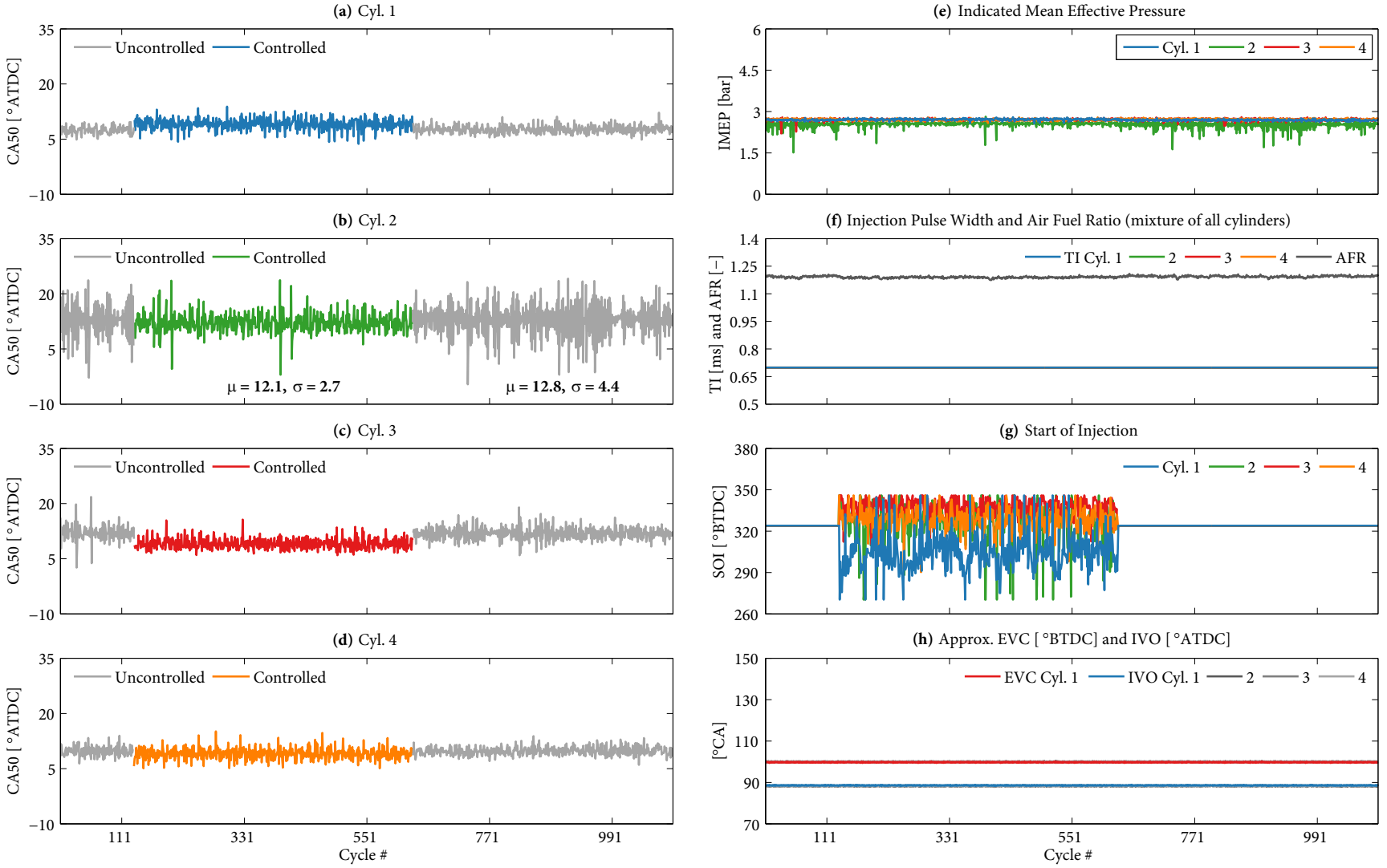


Figure 8.7 Controller performance for reducing cyclic variability on cylinder 2 while maintaining a combustion phasing similar to uncontrolled mean value at 2,500 rpm.

Chapter 9

Contributions and Recommendations

9.1 Insights and findings of this work

9.1.1 A method to predict near chaotic HCCI combustion

The primary contribution of this dissertation is a technique to experimentally predict near chaotic, high CV HCCI combustion phasing in real-time. An example of such predictions is shown in Fig. 9.1.

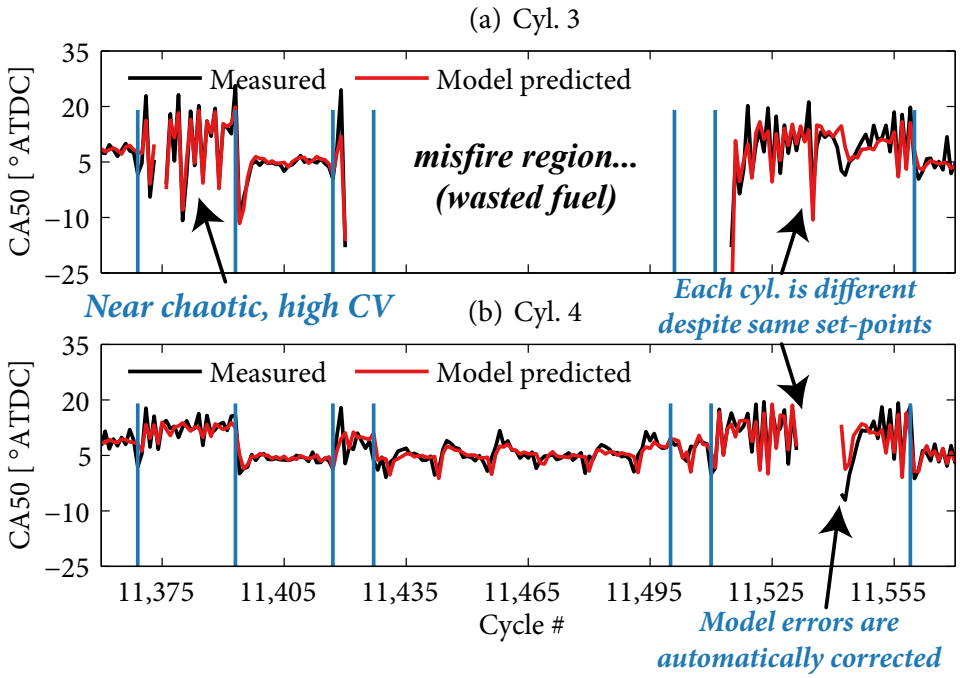


Figure 9.1 Multi-cylinder predictions with the techniques developed in this dissertation. Each blue vertical line is a harsh, simultaneous transient of four different engine actuators (see Fig. 5.6 for detailed information about the actuator settings).

These results confirm the hypothesis that a simple, abstract mapping function can capture

complex combustion behavior when combined with adaptive machine learning. No previous modeling approach has been shown to have such fidelity predicting cycle-to-cycle experimental measurements during transients, operating points with high CV, and at steady-state, right up to complete misfire. Sec. 1.2 shows examples of contemporary models, four of which are from recent Ph.D. dissertations. This new technique opens up new avenues for combustion modeling and control.

9.1.2 Extensive cycle-to-cycle observations from experiments

This work experimentally shows the cycle-to-cycle behavior of NVO HCCI across most of unboosted operating range at 2,500 rpm (129,964 cycles and 2,221 random engine set points). Observations show what appears to be a single period doubling bifurcation in combustion phasing, along with structural differences between the behavior of the inner and outer cylinders (Figs. 5.2 and 5.3). Previous cycle-to-cycle studies have only shown results for a handful of engine set points, whereas this work shows the aggregate behavior across thousands of random set points.

9.1.3 An abstract mapping function

An abstract combustion mapping function is derived that leverages machine learning to accurately predict combustion phasing from cycle-to-cycle across a large number of engine operating points. The technique has validity in a multi-cylinder engine during transients, operating points with high CV, and at steady-state, right up to complete misfire (Fig. 5.6). These predictions can be computed as early as 362 °BTDC firing, or earlier if one is willing to sacrifice model fidelity.

9.1.4 Implicit (not explicit) mixture state and composition

Unlike the typical approach to high fidelity combustion modeling, this work shows that it is not necessary to explicitly compute residual gas fraction, temperatures, air mass, etc. All that is necessary is to implicitly capture a functional dependence for these quantities in an abstract mapping function.

9.1.5 Combustion modeling with just two sensors

Only in-cylinder pressure and crank angle encoder sensors are needed for cycle-to-cycle predictions (no lambda meter, exhaust thermocouples, etc.). Crank angle sensors already exist on production engines, and it is shown that predictions are possible with an in-cylinder pressure sensor that can cost as little as ~$13 per cylinder at production volumes (see Sec. 8.1).

9.1.6 A combustion model that's fast to calibrate

The prediction model can be parameterized without the tedium of performing steady-state actuator sweeps. It can be calibrated with as little as ~20 minutes of test cell time.

9.1.7 A new adaptive machine learning algorithm

A new online adaptation algorithm named Weighted Ring - Extreme Learning Machine (WR-ELM) is proposed. This approach uses a weighted ring buffer data structure of recent measurements to update an offline trained Extreme Learning Machine. At each combustion cycle this strategy applies a single trim to single offline model, whereas previous adaptation approaches apply trims on top of trims continuously ($\mathbf{P}_1$ is not propagated, see Secs 4.2 and 4.3).

It is shown that WR-ELM can provide accurate, *causal* predictions of near chaotic combustion behavior (Fig. 5.6). Analytical partial derivatives against engine sensors and

actuators are possible with this technique, and the model is general enough to provide predictions on an engine it was not originally trained on.

9.1.8 A computationally efficient real-time adaptation algorithm

WR-ELM is very computationally efficient, and it is shown that a ~$7 iPhone® generation 1 processor can run the algorithm in real-time (see Sec. 8.1). Real-time in this context means combustion predictions in well under a millisecond.

9.1.9 Feasibility of engine control and high cyclic variability reduction

Without recalibrating the model for a new engine (described in Sec. 8.2), preliminary model predictive control experiments with the real-time implementation show that engine control with this approach is feasible (see Sec. 8.4). Using SOI as a control knob, it is shown that this method can:

- Balance all the cylinders to a similar combustion phasing in a multi-cylinder engine.
- Tolerate engine speed ramps.*
- Tolerate fuel steps.
- Tolerate exhaust valve closing steps.
- Provide a reduction in near chaotic high cyclic variability.

A video that demonstrates cylinder balancing from an operating point that exhibits near chaotic behavior when uncontrolled is available at [64].

* Even though engine speed dependence isn't included in the model (see Sec. 3.3).

These tests are only intended to show a proof of concept. With the fundamental principles elucidated in this dissertation, calibration and further refinement of combustion control with the WR-ELM mapping function approach should be straightforward for a commercial venture.

9.2 Recommendations for future work

Engine control with the methods of this dissertation are at the proof of concept stage, and future work is needed to refine the approach. Cyclic variability is known to be very sensitive to parameters, and of particular interest is whether model re-calibration for the current engine can provide a greater reduction in cyclic variability and extend the HCCI operating envelope. It is also interesting to explore the use of other actuators for engine combustion (e.g. multiple injections and spark), and to reduce control latency further so that more SOI authority is available. Additionally, it would be of interest to explore whether WR-ELM's adaptation can adapt around small fuel property variations.

Other areas of interest are applying the general WR-ELM mapping function framework developed in this work to other (boosted) combustion modes such as Spark Assisted Compression Ignition (SACI) and Partially Premixed Compression Ignition (PPCI) and Reactivity Controlled Compression Ignition (RCCI) along with conventional Spark Ignition (SI) and Diesel. While these other combustion modes may not be as deterministic or have as strong cycle-to-cycle coupling, it is felt that the general adaptive framework developed in this dissertation can still capture some cycle-to-cycle and mean combustion characteristics. Additionally, the adaptive machine learning framework may provide a reduction in engine calibration effort over traditional lookup table based engine control (Fig. 1.8).

References

[1] H. Yilmaz, O. Miersch-Wiemers, and L. Jiang. "Advanced Combustion Concepts - Enabling Systems and Solutions (ACCESS)". *2013 DOE Vehicle Technologies Office Annual Merit Review*. URL: http://energy.gov/sites/prod/files/2014/03/f13/ace066_yilmaz_2013_0.pdf. 2013.

[2] *U.S. Energy Information Administration*. URL: http://eia.gov. [Online; accessed 9/18/2014]. 2014.

[3] L. M. Olesky, J. Vavra, D. Assanis, and A. Babajimopoulos. "Effects of Charge Preheating Methods on the Combustion Phasing Limitations of an HCCI Engine With Negative Valve Overlap". *Journal of Engineering for Gas Turbines and Power* 134.11, 112801 (2012), p. 112801.

[4] S. Saxena and I. D. Bedoya. "Fundamental phenomena affecting low temperature combustion and HCCI engines, high load limits and strategies for extending these limits". *Progress in Energy and Combustion Science* 39.5 (2013), pp. 457 –488.

[5] L. Manofsky, J. Vavra, D. Assanis, and A. Babajimopoulos. "Bridging the Gap between HCCI and SI: Spark-Assisted Compression Ignition" (2011). SAE Paper 2011-01-1179.

[6] J. Larimore, E. Hellström, S. Jade, A. G. Stefanopoulou, and L. Jiang. "Real-time internal residual mass estimation for combustion with high cyclic variability". *International Journal of Engine Research* (2014).

[7] J. Larimore, S. Jade, E. Hellström, A. G. Stefanopoulou, J. Vanier, and L. Jiang. "Online adaptive residual mass estimation in a multicylinder recompression HCCI engine". *Proc. ASME Dynamic Systems and Control Conference*. Palo Alto, CA, USA, 2013.

[8] E. Hellström, A. G. Stefanopoulou, and L. Jiang. "Cyclic Variability and Dynamical Instabilities in Autoignition Engines with High Residuals". *IEEE Transactions on Control Systems Technology* 21.5 (2013), pp. 1527–1536.

[9] E. A. O. Soto, J. Vavra, and A. Babajimopoulos. "Assessment of Residual Mass Estimation Methods for Cylinder Pressure Heat Release Analysis of HCCI Engines With Negative Valve Overlap". *Journal of Engineering for Gas Turbines and Power* 134.8, 082802 (2012).

[10] J. W. Larimore. "Experimental Analysis and Control of Recompression Homogeneous Charge Compression Ignition Combustion at the High Cyclic Variability Limit". PhD thesis. University of Michigan, 2014.

[11] A. F. Jungkunz. "Actuation strategies for cycle-to-cycle control of homogeneous charge compression ignition combustion engines". PhD thesis. Stanford University, 2013.

[12] M. Bidarvatan and M. Shahbakhti. "Grey-Box Modeling for HCCI Engine Control". *ASME Conference Proceedings* 2013.19097 (2013).

[13] K. Hoffmann. "Non-linear Model-based Predictive Control of a Low-Temperature Gasoline Combustion Engine". PhD thesis. RWTH Aachen University, 2010.

[14] V. M. Janakiraman. "Machine Learning for Identification and Optimal Control of Advanced Automotive Engines". PhD thesis. University of Michigan, 2013.

[15] M. Jia, A. B. Dempsey, H. Wang, Y. Li, and R. D. Reitz. "Numerical simulation of cyclic variability in reactivity-controlled compression ignition combustion with a focus on the initial temperature at intake valve closing". *International Journal of Engine Research* (2014).

[16] R. J. Middleton. "Simulation of Spark Assisted Compression Ignition Combustion Under EGR Dilute Engine Operating Conditions". PhD thesis. University of Michigan, 2014.

[17] A. Ghazimirsaied, M. Shahbakhti, and C. R. Koch. "HCCI Engine Combustion Phasing Prediction Using a Symbolic Statistics Approach". *Journal of Engineering for Gas Turbines and Power* 132.8, 082805 (2010). Paper 082805.

[18] X. He, M. Donovan, B. Zigler, T. Palmer, S. Walton, M. Wooldridge, and A. Atreya. "An experimental and modeling study of isooctane ignition delay times under homogeneous charge compression ignition conditions". *Combustion and Flame* 142.3 (2005), pp. 266–275.

[19] S. Walton, X. He, B. Zigler, M. Wooldridge, and A. Atreya. "An experimental investigation of isooctane ignition phenomena". *Combustion and Flame* 150.3 (2007), pp. 246–262.

[20] J. Livengood and P. Wu. "Correlation of autoignition phenomena in internal combustion engines and rapid compression machines". *Symposium International on Combustion* 5.1 (1955), pp. 347–356.

[21] L. Hagen, L. M. Olesky, S. Bohac, G. Lavoie, and D. Assanis. "Effects of Low Octane Gasoline Blended Fuel on HCCI Load Limit, Combustion Phasing and Burn Duration". *Journal of Engineering for Gas Turbines and Power* 135.7 (2013). 072001-1 to 072001-10.

[22] J. S. Lacey. "The effects of advanced fuels and additives on homogeneous charge compression ignition combustion and deposit formation". PhD thesis. University of Michigan, 2012.

[23] M. Sjöberg and J. E. Dec. "Comparing late-cycle autoignition stability for single- and two-stage ignition fuels in HCCI engines". *Proceedings of the Combustion Institute* 31.2 (2007), pp. 2895 –2902.

[24] M. Sjöberg and J. E. Dec. "Combined Effects of Fuel-Type and Engine Speed on Intake Temperature Requirements and Completeness of Bulk-Gas Reactions for HCCI Combustion" (2003). SAE Paper 2003-01-3173.

[25] M. Mehl, W. J. Pitz, C. K. Westbrook, and H. J. Curran. "Kinetic modeling of gasoline surrogate components and mixtures under engine conditions". *Proceedings of the Combustion Institute* 33.1 (2011), pp. 193 –200.

[26] M. Shahbakhti and C. R. Koch. "Characterizing the cyclic variability of ignition timing in a homogeneous charge compression ignition engine fueled with n-heptane / iso-octane blend fuels". *International Journal of Engine Research* 9.5 (2008), pp. 361–397.

[27] C. S. Daw, R. M. Wagner, K. D. Edwards, and J. B. G. Jr. "Understanding the transition between conventional spark-ignited combustion and HCCI in a gasoline engine". *Proceedings of the Combustion Institute* 31.2 (2007), pp. 2887 –2894.

[28] J. C. Kantor. "A Dynamical Instability of Spark Ignited Engines". English. *Science*. New Series 224.4654 (1984), pp. 1233–1235.

[29] C. S. Daw, M. B. Kennel, C. E. A. Finney, and F. T. Connolly. "Observing and modeling nonlinear dynamics in an internal combustion engine". *Phys. Rev. E* 57 (3 1998), pp. 2811–2819.

[30] M. Sjöberg, J. E. Dec, A. Babajimopoulos, and D. Assanis. "Comparing Enhanced Natural Thermal Stratification Against Retarded Combustion Phasing for Smoothing of HCCI Heat-Release Rates" (2004). SAE Paper 2004-01-2994.

[31] K. D. Edwards, C. S. Daw, W. R. Elwasif, C. E. A. Finney, S. Pannala, M. K. Stoyanov, R. M. Wagner, and C. G. Webster. "Accelerating predictive simulation of IC engines with high performance computing". *2014 DOE Vehicle Technologies Office Annual Merit Review*. URL: http://energy.gov/sites/prod/files/2014/07/f17/ace017_edwards_2014_0.pdf. 2014.

[32] R. Devaney. *A First Course in Chaotic Dynamical Systems: Theory and Experiment*. Studies in nonlinearity. Westview Press, 1992.

[33] M. Hirsch, S. Smale, and R. Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Academic Press. Academic Press, 2012.

[34] A. Babajimopoulos, V. Challa, G. Lavoie, and D. Assanis. "Model-Based Assessment of Two Variable CAM Timing Strategies for HCCI Engines: Recompression vs. Rebreathing". *Proceedings of the ASME Internal Combustion Engine Division 2009 Spring Technical Conference, ICES2009-76103*. 2009.

[35] E. Sander and J. A. Yorke. "Period-doubling cascades galore". *Ergodic Theory and Dynamical Systems* 31 (04 2011), pp. 1249–1267.

[36] A. Sharkovsky. *Dynamics of One-Dimensional Maps*. Mathematics and Its Applications. Springer, 1997.

[37] V. Cherkassky and F. Mulier. *Learning from Data: Concepts, Theory, and Methods*. Wiley, 2007.

[38] M. Mehl, W. J. Pitz, M. Sjöberg, and J. E. Dec. "Detailed Kinetic Modeling of Low-Temperature Heat Release for PRF Fuels in an HCCI Engine" (2009). SAE Paper 2006-01-1087.

[39] G. Kukkadapu, K. Kumar, C.-J. Sung, M. Mehl, and W. J. Pitz. "Experimental and surrogate modeling study of gasoline ignition in a rapid compression machine". *Combustion and Flame* 159.10 (2012), pp. 3066 –3078.

[40] G. M. Shaver. "Physics-based modeling and control of residual-affected HCCI engines using variable valve actuation". PhD thesis. Stanford University, 2005.

[41] A. Vaughan and S. V. Bohac. "A Cycle-to-Cycle Method to Predict HCCI Combustion Phasing". *Proceedings of the ASME Internal Combustion Engine Division 2013 Fall Technical Conference, ICEF2013-19203*. 2013.

[42] A. Vaughan and S. V. Bohac. "An Extreme Learning Machine Approach to Predicting Near Chaotic HCCI Combustion Phasing in Real-Time". *CoRR* abs/1310.3567 (2013).

[43] N. Ravi. "Modeling and control of exhaust recompression HCCI using variable valve actuation and fuel injection". PhD thesis. Stanford University, 2010.

[44] K. Chang, A. Babajimopoulos, G. A. Lavoie, Z. S. Filipi, and D. N. Assanis. "Analysis of Load and Speed Transitions in an HCCI Engine Using 1D Cycle Simulation and Thermal Networks" (2006). SAE Paper 2006-01-1087.

[45] S. D. Jade. "Transient Load-Speed Control in Multi-Cylinder Recompression HCCI Engines". PhD thesis. University of Michigan, 2014.

[46] L. Koopmans, R. Ogink, and I. Denbratt. "Direct Gasoline Injection in the Negative Valve Overlap of a Homogeneous Charge Compression Ignition Engine" (2003). SAE Paper 2003-01-1854.

[47] M. F. J. Brunt and A. L. Emtage. "Evaluation of Burn Rate Routines and Analysis Errors" (1997). SAE Paper 970037.

[48] J. Chang, O. Güralp, Z. Filipi, D. Assanis, T.-W. Kuo, P. Najt, and R. Rask. "New Heat Transfer Correlation for an HCCI Engine Derived from Measurements of Instantaneous Surface Heat Flux" (2004). SAE Paper 2004-01-2996.

[49] C. Depcik, T. Jacobs, J. Hagena, and D. Assanis. "Instructional use of a single zone, premixed charge, spark ignition engine heat release simulation". *International Journal of Mechanical Engineering Education* 35.1 (2007).

[50] R. P. Fitzgerald, R. Steeper, J. Snyder, R. Hanson, and R. Hessel. "Determination of Cycle Temperatures and Residual Gas Fraction for HCCI Negative Valve Overlap Operation". *SAE Int. J. Engines* 3 (2010). SAE Paper 2010-01-0343, pp. 124–141.

[51] K. Kar, S. Roberts, R. Stone, M. Oldfield, and B. French. "Instantaneous Exhaust Temperature Measurements Using Thermocouple Compensation Techniques" (2004). SAE Paper 2004-01-1418.

[52] O. Welling and N. Collings. "UEGO Based Measurement of EGR Rate and Residual Gas Fraction" (2011). SAE Paper 2011-01-1289.

[53] C. S. Daw, K. D. Edwards, R. M. Wagner, and J. Johney B. Green. "Modeling Cyclic Variability in Spark Assisted HCCI". *Journal of Engineering for Gas Turbines and Power* 130.5, 052801 (2008). Paper 052801.

[54] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. "Extreme learning machine: Theory and applications". *Neurocomputing* 70.1-3 (2006), pp. 489 –501.

[55] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan. "A fast and accurate online sequential learning algorithm for feedforward networks". *Neural Networks, IEEE Transactions on* 17.6 (2006), pp. 1411–1423.

[56] B. Mirza, Z. Lin, and K.-A. Toh. "Weighted Online Sequential Extreme Learning Machine for Class Imbalance Learning". *Neural Processing Letters* (2013), pp. 1–22.

[57] D. Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience, 2006.

[58] *John Halleck's Matrix Identities (also Ring Identities)*. URL: http://www.cc.utah.edu/~nahaj/math/matrix-identities.html. [Online; accessed 12/31/2014]. 2014.

[59] N. N. Schraudolph. "A Fast, Compact Approximation of the Exponential Function". *Neural Computation* 11.4 (1999), pp. 853–862.

[60] *Leet stopwatch*. URL: http://commons.wikimedia.org/wiki/File:Leet_stopwatch.JPG. [Online; accessed 12/31/2014]. 2014.

[61] *Dealing with x86 SMI troubles*. URL: https://xenomai.org/2014/06/dealing-with-x86-smi-troubles. [Online; accessed 12/31/2014]. 2014.

[62] *8000 interrupts per second when idle!* URL: http://www.raspberrypi.org/forums/viewtopic.php?f=28&t=7866. [Online; accessed 12/31/2014]. 2014.

[63] Open Source Automation Development Lab. *Profile of system in rack #b, slot #3*. URL: https://www.osadl.org/Profile-of-system-in-rack-b-slot-3.qa-profile-rbs3.0.html.

[Online; accessed 9/1/2014; as of 12/31/2014 this link unfortunately no longer hosts the Preempt_RT kernel patches]. 2014.

[64] *Version 0.1 Raspberry Pi Engine Control with Real-Time Adaptive Extreme Learning Machine*. URL: https://www.youtube.com/watch?v=qQG7ocnE3EA. [Online; accessed 1/8/2015]. 2015.