Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2016

Improving Misfire Detection Using Gaussian Processes and Flywheel Error Compensation

Gustav Romeling

LINKÖPING UNIVERSITY

Master of Science Thesis in Electrical Engineering

Improving Misfire Detection Using Gaussian Processes and Flywheel Error Compensation

Gustav Romeling

LiTH-ISY-EX--16/5007--SE

Supervisor: **Daniel Jung**
ISY, Linköpings universitet

Examiner: **Erik Frisk**
ISY, Linköpings universitet

Division of Vehicular Systems
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Copyright © 2016 Gustav Romeling

Abstract

The area of misfire detection is important because of the effects of misfires on both the environment and the exhaust system. Increasing requirements on the detection performance means that improvements are always of interest. In this thesis, potential improvements to an existing misfire detection algorithm are evaluated.

The improvements evaluated are: using Gaussian processes to model the classifier, alternative signal treatments for detection of multiple misfires, and effects of where flywheel tooth angle error estimation is performed. The improvements are also evaluated for their suitability for use on-line.

Both the use of Gaussian processes and the detection of multiple misfires are hard problems to solve while maintaining detection performance. Gaussian processes most likely loses performance due to loss of dependence between the weights of the classifier. It can give performance similar to the original classifier, but with greatly increased complexity. For multiple misfires, the performance can be slightly improved without loss of single misfire performance. Greater improvements are possible, but at the cost of single misfire performance. The decision is in the end down to the desired trade-off.

The flywheel tooth angle error compensation gives nearly identical performance regardless of where it is estimated. Consequently the error estimation can be separated from the signal processing, allowing the implementation to be modular. Using an EKF for estimating the flywheel errors on-line is found to be both feasible and give good performance. Combining the separation of the error estimation from the signal treatment with a, after initial convergence, heavily restricted EKF gives a vastly reduced computational load for only a moderate loss of performance.

Acknowledgements

I would like to thank my supervisor Daniel Jung for the rewarding discussions and guidance during my work, but also for the work and ideas this thesis is built upon. I also like to thank my examiner Erik Frisk for the opportunity to write this thesis and for making me realise what is possible to achieve in the short time available. I would also like to thank them both for introducing me to the area of diagnosis and supervision, it is what led to me writing this thesis.

My family deserves an immense thank you. I am grateful to my parents for a great upbringing, and their support and encouragement. I am grateful to my sister for plenty of healthy disagreement, amidst countless memorable times. And to my extended family, too numerous to mention, always interested in my endeavours.

A thank you also to my classmates, rugby friends and many more, you made my time at university what it is, you made it great.

Linköping, November 2016
Gustav Romeling

Contents

Notation

Abbreviations

Abbreviation	Meaning
TQ	Test quantity
NF	No fault
MF	Misfire
MD	Missed detection
FA	False alarm
SVM	Support vector machine
GP	Gaussian process
MA	Moving average
ARX	Auto-regression with exogenous input
EKF	Extended Kalman filter
CG-EKF	Constant gain extended Kalman filter
OBD-II	On-board diagnostics II
FIR	Finite impulse response
IIR	Infinite impulse response

1 Introduction

1.1 Background

The subject of engine misfire detection is important for a few different reasons. One of the more obvious ones is the fact that in many parts of the world there is legislation in place that specifies detection rates for misfires. The requirements for detection have become incrementally more stringent. Starting with model year 2010 the complete failure of a cylinder must be detected, and that increased to include detection of intermittent misfires from model year 2013 [16]. Legislation is however only the reaction to the underlying reason for misfire detection. Likely one of the driving reasons is the impact on the environment from misfires, as well as other emissions. There is the obvious emission of unburned fuel when a misfire occurs. Apart from the immediate impact, misfire also risks degrading the catalytic converters, with the consequence of increasing the overall levels of emissions from a vehicle. OBD-II is a commonly used standard for the requirements of emission control that was introduced in the 1990s and has its origin in California, where the first emission requirements was introduced in the 1960s to combat the smog problem, and they have become increasingly more stringent as time has progressed [17].

For the manufacturers and owners of the vehicles, detection of misfires is also of interest since it may be the indication of some fault in the engine, and early detection and mitigation may potentially save on costs for repairs that escalated problems could incur.

1.2 Goals and purpose

The overall purpose of this thesis is to evaluate possible improvements to the misfire detection algorithm proposed in [10] as well as to the flywheel tooth error

compensation added in [11]. The possible ways of improvement are all concerned with a specific part of the original algorithm, and they are summarised in the following research questions:

1. Can a Gaussian process be used to describe the weights of the classifier?
2. Can the algorithm be adapted to handle multiple misfires?
3. Is it possible to improve the estimation of the flywheel tooth angle error?
4. Can the current algorithm be implemented efficiently for use on-line, and what can be done to improve its performance?

The work presented in [10, 11] was performed using data from vehicles with six-cylinder engines, but also validated on data from a four-cylinder engine to show the algorithm to be generalisable. In this thesis the data used is from four-cylinder engines, and thus it also serves as a further validation that the algorithm works for other engines than the six-cylinder one.

1.2.1 Gaussian processes

One idea for improving the algorithm is to model the weights of the classifier using Gaussian processes. Currently, the algorithm divides the operating range into segments, and a classifier is trained for each of them. This has the consequence that the weights are constant over each segment. Thus, the effect is that very similar measurements may be classified by two different classifiers, while two more distant points are classified using the same classifier, simply due to the segmentation of the algorithm. The rationale behind using Gaussian processes is that it would allow for more accurate weights locally. It is also a potential way of having a much higher resolution classifier while at the same time limiting the memory needed for storage.

1.2.2 Multiple misfires

The algorithm proposed in [10] performed well for single misfires. The performance degrades when there are multiple misfires present during one cycle. This is most likely due to the fact that the removal of the mean is no longer as suitable. Removing the mean is intended to remove the effects of other torques acting on the power train, but at the same time, misfires are still visible in the magnitude of the torque signal. The problem that arises when there are multiple misfires present is that the mean falls enough for the misfires to no longer be visible in the magnitude. Logically the variance of the torque signal would still be affected by the misfires, but that is outside of the scope of the algorithm studied here. Looking at the variance would also likely be too computationally intensive to be feasible on-line. The desire here is to find an alternative quantity to remove, one that is not subject to the same problem as the mean in the presence of multiple misfires.

1.2.3 Flywheel tooth angle errors

To compensate between vehicles for the flywheel tooth angle errors the calculations are currently performed on the normalised torque estimate. This has good performance, but may not be the most suitable point of estimation for a few reasons. Firstly, the idea of implementing the estimation with an Extended Kalman Filter (EKF) for use on-line may mean that efficient execution is hampered by the additional operations needed to perform the data treatment in the EKF. To facilitate the use of the flywheel tooth angle error compensation it is desirable to estimate the errors at a different point in the treatment. This would also have the added benefit of, depending on the point of estimation, separating the estimation of the errors from the processing of the data and the detection of misfires, making an on-line implementation more modular.

1.2.4 On-line adaptation

In its current implementation, the algorithm works well when run on desktop PCs. There is a substantial difference in performance between a desktop PC and the embedded computers used in cars. This means that the algorithm needs to be optimised for use on limited hardware, and likely under hard real-time requirements. This also means that there is always the need to balance detection performance with computational requirements. Here the purpose is to adapt the current implementation of the algorithm to perform faster, however as this will still be done on a desktop PC, the adaptations cannot be verified as actually being sufficient to arrive at a feasible on-line implementation. It is however a first step.

1.3 Method

The work is performed on data available from the work presented in [10, 11] that this thesis builds upon. Actual work is performed using the software Matlab to manipulate the data, and to implement and evaluate the proposed improvements.

As the original work was performed mainly for vehicles with 6-cylinder engines, but the work presented here is performed for 4-cylinder engines, a baseline of the algorithms performance is needed. This is done by simply running the algorithm on the new data. In addition to giving a baseline performance, it also verifies that the algorithm at least gives reasonable performance on the new data.

Next, the theoretical underpinnings of the ideas for improvement are investigate and adapted for implementation into code.

From this point, the work mainly consists of implementing and tuning the solutions to maximise their potential effect on the performance of the algorithm.

The practical work with the data and performing the misfire detection uses the convention in modelling to split the data into two parts. One is used to train the classifier, while the other part is used to validate the performance. This is so that over-fitting is not performed, which could result in a classifier that can

describe the behaviour of known data well, while it performs much worse on new data.

The central performance metric for evaluating the changes implemented will be the rates of missed detection (MD) and false alarms (FA). It is also natural to compare performance achieved here to what [10, 11] found, as it is the foundation this work seeks to improve upon.

1.4 Related research

For the work performed in this thesis there are a few areas of research that are of interest. Firstly, the whole area of misfire detection is highly relevant. Furthermore, Gaussian processes are of interest, in terms of statistics as well as in terms of the estimation of them.

1.4.1 Misfire detection

There are a few ways of detecting misfires. They can be divided into two main categories: in-cylinder detection and post-cylinder detection. [1]

In-cylinder detection is often complicated by the fact that the environment is hostile to most sensors. It is also not common to have sensors already in place in the cylinder, thus detection in-cylinder means there is the need of adding sensors. This increases both the complexity of the diagnostics system and the cost of hardware [1].

Post-cylinder detection has the advantage of it being a less hostile environment for sensors, and this is also where a lot of sensors used for other purposes are found. These sensors can also be used to detect misfires in the engine. Such methods either use measurements on the exhaust gases, or the mechanical behaviour of the engine to detect the misfires. The work this thesis builds on uses speed measurements from the flywheel for the detection. This method has the advantage that it is possible to directly attribute the measurements to a cylinder. If measurements from the exhaust system are used the position of the sensor affects how they relate to the cylinders, and the relationship is not necessarily constant with varying operating points.

The research into misfire detection spans all of these fields. In [18] the crankshaft signal is used in an attempt to estimate the vibrations in the engine and detect misfires from these. Another post-cylinder approach is to use the exhaust temperature [23] to perform the detection. For the in-cylinder approach ion-current is one possible way of measuring the combustion, in [3] shown to be able to detect misfires and used in conjunction with post-combustion measurements to perform the detection.

1.4.2 Gaussian processes

"A Gaussian process is a generalisation of the Gaussian probability distribution" [19], and as such much of the general theory for Gaussian variables apply to Gaussian processes, with the appropriate adaptations. The use of Gaussian processes

in this thesis is only to describe the behaviour of the weights in the classifier, and this area has already been thoroughly studied. In [8] Gaussian process regression is shown to be suitable for modelling engine behaviours, and it is done with general limitations of ECUs in mind. For system identification where there is quantisation present, [6] offers insight into how this can be done when statistical measures are desired. However not directly applicable to this thesis, the prospect of managing the quantisation is relevant if sampling with higher angular resolution is to be done.

However, on the topic of Gaussian processes, one area that is not investigated in this thesis, but could be an interesting area for future focus is the use of Gaussian process classifiers. One area in particular that could be well suited for the problem at hand is a one class classifier, as presented in [12], although used for image recognition it could have applications in misfire detection since only data from normal operations would be required.

1.5 Characteristics of the data

The data used in this work was collected from vehicles with four-cylinder engines of the same model. Focus will be on performance on data from driving on roads, there is however also data collected on dynamometers available if testing under more controlled circumstances is to be performed.

The vast majority of the data is collected under conditions where the cold start mode of the engine control never engaged. In total, less than 2600 combustions under cold start conditions are available. This corresponds to roughly 600 to 700 cycles. The small amount of cold start data coupled with the fact that the work in [10] found the performance to be satisfying, as well as the limited time available to perform the work means that cold starts will not be treated.

There are a lot of signals available in the raw data files. However only a few are of interest for the algorithm, as well as a few more for training the classifier and compute the performance. The table below lists the signals of interest for the algorithm after extraction and adaptation.

Table 1.1: *Available signals.*

Signal	Description
T	Estimated torque signal
flywheel	Time for each 30° rotation.
load	Air mass flow.
speed	Engine speed.
cylID	ID of the crank positions.
catwrm	Flag for warming of the catalytic converter.
misfire	Flag for misfire injection.
exe	Flag for diagnosable operations.

If the data is studied in bivariate histograms, a few observation can immedi-

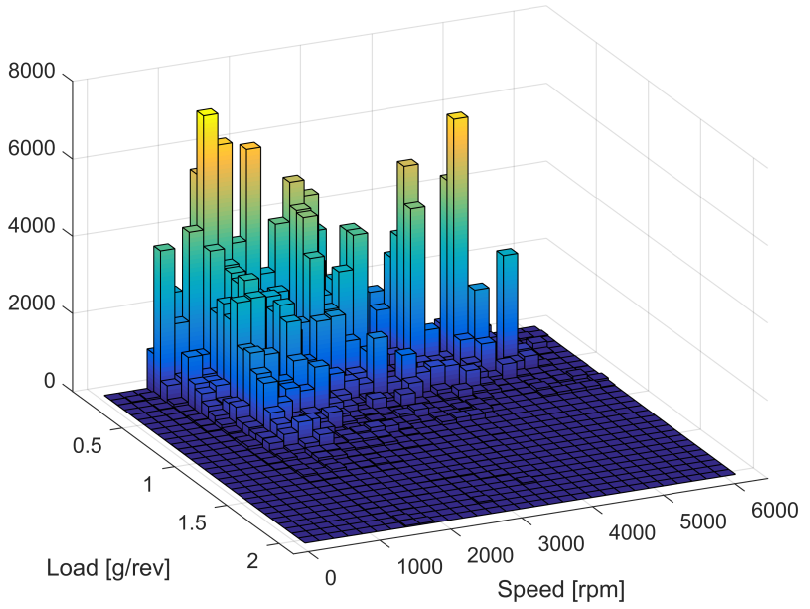


Figure 1.1: *Histogram of the data used for training the classifier.*

ately be made:

- Most of the data is for low loads.
- There is clear clustering in speed of the data.
- There is more variation in load for low speeds.

As the data used here was collected from vehicles on the road, the clustering in speed can potentially be attributed to different gears and speed limits. The variation in load for low speeds can possibly be due to the greater range in operational situations.

To give a feeling for how the data is distributed the multivariate histograms in Figures 1.1 and 1.2 displays the distribution for the data used for training the classifier and estimating the flywheel tooth angle errors.

1.6 Outline

The thesis consists of ten chapters, briefly described below.

Chapter 1 introduces the area of misfire detection, what the goals of the thesis are and how they will be achieved. It continues with an overview of the current

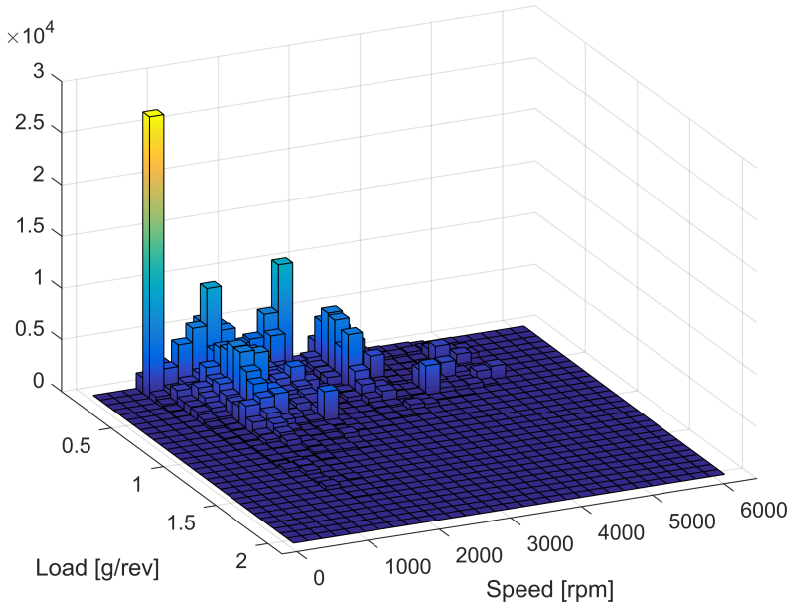


Figure 1.2: *Histogram of the data used for flywheel tooth angle errors.*

research and then looks at the data that will be used, before giving this outline of the thesis.

Chapter 2 describes the work previously done to develop the misfire detection algorithm that this work builds upon. It aims to give the reader an overall understanding of the basic components in the algorithm.

Chapter 3 introduces the concept of Gaussian processes, as well as bootstrap. It also goes into the possible problems in this particular application, as well as the restrictions inherent to it.

Chapter 4 first gives the reason for the original algorithm's data treatment, and then explores possible replacements, their definitions and possible implementations, and the necessary considerations associated with each one.

Chapter 5 explores the different points of estimation and how the choice can potentially affect the estimated flywheel tooth angle errors. It finishes with a look into how the problem can be solved off-line through optimisation.

Chapter 6 looks at the ideas presented over Chapters 3-5 and the original classifier, discussing how they can be adapted for use on-line.

Chapter 7 goes through all the performed evaluations of the proposed improvements, with the focus being on the best performing version of each implementation.

Chapter 8 discusses the results from Chapter 7 and the likely reasons behind them.

Chapter 9 summarises the main results, both improvements and the best options for the proposed improvements.

Chapter 10 presents areas that would be of interest for continued work. Both areas from this thesis to look deeper into, as well as areas that are related but outside the scope here, are suggested.

2

Misfire detection algorithm

The algorithm below is the one presented in [10] and further developed in [11]. This chapter aims at familiarising the reader with the specific algorithm to the degree needed for a general understanding of this thesis. The aim is however not to give as detailed a description as in [10, 11].

The requirement in misfire detection of distinguishing between cylinders, together with the chosen approach means that each cylinder can be considered on its own when performing the misfire detection. The only shared part between the cylinders is the mean.

The engine considered is a four-stroke, and thus for each two revolutions of the crankshaft, each cylinder goes through one full cycle. The detection is being performed on a per-cycle basis.

2.1 Torque model

Central to the misfire detection algorithm is the estimation of torque from the flywheel-timing signal. The time it takes the flywheel to rotate through 30° is measured continuously, and it can be converted into fluctuations in the torque of the power train.

Firstly, for each time measurement it is easy to convert into the angular velocity of the flywheel using equation (2.1a). From mechanics it is well known that for a rotating disc the torque is defined as equation (2.1b). The moment of inertia J does not need to be known, as long as it is kept constant for all calculations it is only a scaling of the magnitude of the estimate. To connect these two equations, the relationship between angular velocity and angular acceleration is needed, (2.1c)

$$\omega[k] = \frac{\theta}{t[k]} \tag{2.1a}$$

$$T = \alpha \cdot J \tag{2.1b}$$

$$\dot{\omega}[k] = \alpha[k] \tag{2.1c}$$

From the equations above it is possible to go from time measurement to torque. However, this would require sampling that is time synchronous to directly work due to the time derivative in (2.1c). The available data is sampled synchronous with the angle of the flywheels rotation. Therefore, a substitution is needed. Using partial derivatives and the product rule of derivatives the results in (2.2) can be used to perform the substitution.

$$\alpha = \dot{\omega} = \frac{d\omega}{dt} = \frac{d\omega}{d\theta} \cdot \frac{d\theta}{dt} = \frac{d\omega}{d\theta} \cdot \omega \tag{2.2a}$$

$$\frac{d(\omega^2)}{d\theta} = \frac{d(\omega \cdot \omega)}{d\theta} = \frac{d\omega}{d\theta} \cdot \omega + \omega \cdot \frac{d\omega}{d\theta} = 2 \cdot \frac{d\omega}{d\theta} \cdot \omega \leftrightarrow \frac{d\omega}{d\theta} \cdot \omega = \frac{1}{2} \frac{d(\omega^2)}{d\theta} \tag{2.2b}$$

From the equations listed above an expression relating angular velocity to torque is now available (2.3a). However, since the signal available from the flywheel corresponds to the angular velocity, and the equation uses its derivative to calculate the torque, the derivative needs to be estimated. This is done using Euler forward approximation (2.3b). Combining all but the conversion from time to angular velocity (2.1a) yields the torque equation in (2.3c).

$$T = \frac{J}{2} \frac{d(\omega^2[k])}{d\theta} \tag{2.3a}$$

$$\frac{d(\omega^2[k])}{d\theta} = \frac{\omega^2[k+1] - \omega^2[k]}{\Delta\theta} \tag{2.3b}$$

$$T = \frac{J}{2} \frac{\omega^2[k+1] - \omega^2[k]}{\Delta\theta} \tag{2.3c}$$

There are additional torques present in the power train other than the torque generated by the combustions (2.4b). Variations in these torques do not depend on the misfires, and thus it is of interest to remove them. By realising that these torques vary slower than the torque generated by the combustions, they can be assumed constant for each cycle, and thus only bias the torque estimates. To account for this the algorithm removes the mean torque over each cycle (2.4c).

$$T = T_{cyl} + T_{load} \tag{2.4a}$$

$$T_{cyl} = \sum_{i=1}^{n_{cyl}} T_{cyl,i} \tag{2.4b}$$

$$T_0[k] = T[k] - \frac{1}{24}\sum_{l=1}^{24} T[l] \tag{2.4c}$$

In addition to the variations of other torques, the torque estimates are also subject to influences from the load on the engine. For this, the intake manifold pressure is a good representation. This signal is however not available, but rather the algorithm uses the fact that it is proportional to the air mass induced per revolution in the six-cylinder engines. It also mostly holds for the four-cylinder engine. This proportionality means that the air mass can be used to perform normalisation with the load. The resulting estimate will not have the correct magnitude, nor is all the contributing physical properties available to calculate the actual pressure. However, due to the proportionality and since it is sufficient to capture the variations of the torque, the magnitude of the resulting estimate is not important. The torque estimate after the removal of the mean is normalised with the load, and the resulting torque estimate is given in (2.5).

$$T_{m_a}[k] = \frac{1}{m_a}\left(T[k] - \frac{1}{24}\sum_{l=1}^{24} T[l]\right) \tag{2.5}$$

The normalised torque estimate arrived at above typically behaves as in Figure 2.1 for fault-free data over one cycle.

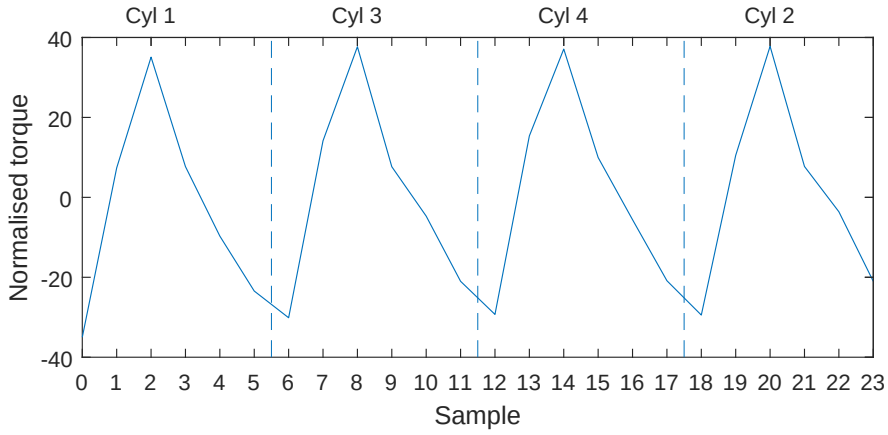


Figure 2.1: *Normalised estimated torque for one cycle without misfire.*

2.2 Classifier

To detect misfires, a classifier of the type Support Vector Machine (SVM) is used. For the misfire detection a binary classifier is used, that is misfire or normal operation. Trying to find the hyperplane that best separates the two classes arrives at a

SVM classifier. [9] The data is made up of two parts. The first one is observations from each combustion stroke, that is: six torque estimates per cylinder. The second one is the class to which the observation belongs. If the data is separable, the goal is also to maximise the margin, i.e. the distance from the hyperplane to the closest point in either class, with the hyperplane chosen so that the closest points are the same distance away. If the data is not separable, the goal is instead to minimise some misclassification cost. The implementation of SVM used for the algorithm uses a cost that tries to keep the relative number of misclassifications as low as possible.

When designing a SVM classifier, one of the most impactful decisions can be the choice of which predictors to use. A predictor is a characteristic in the data used to describe the different classes. Due to the restrictions on computational resources in vehicles, only the direct torque estimates were used, rather than higher order combinations of them. The general reason for using higher order combinations of signals as predictors is that training the classifier remains a problem of finding a linear hyperplane, only in higher dimensions of the predictors [9]. Transformed back in to the original dimensions, the limits would then be of a higher order. Computationally it is more attractive to find the higher order linear hyperplanes, rather than estimating higher order surfaces directly.

With y being the classification, β a column vector with the weights for each of the torque estimates, x a column vector with the actual torque estimates and b the bias, the general form for the separating hyperplane (2.6a) and classifier (2.6b) for a linear SVM are obtained. In Figure 2.2 a two-dimensional example of the classifier is given. It also illustrates how the cost being relative to the number of samples in the class effects the classification. All misclassifications in the example are of normal operations, i.e. false alarms.

$$\beta \cdot x' + b = 0 \tag{2.6a}$$

$$y = \beta \cdot x' + b \tag{2.6b}$$

Due to the behaviour of the torque estimates over the operating ranges in speed and load of the engine, the original algorithm segments the data by speed. A separate classifier is trained for each range and cylinder. This does increase the required storage in the vehicles, but results in increased detection performance.

When using the classifier the test quantity (TQ) is typically distributed similarly to Figure 2.3. For positive values of the TQ a misfire is assumed to have occurred.

2.3 Flywheel tooth angle error compensation

Thus far, all of the steps in the algorithm have been performed on the same vehicle, and variations due to manufacturing are not a problem when considering only one vehicle. However, if the algorithm is evaluated on another vehicle without adaptation, with the same set-up in terms of its power train, the performance

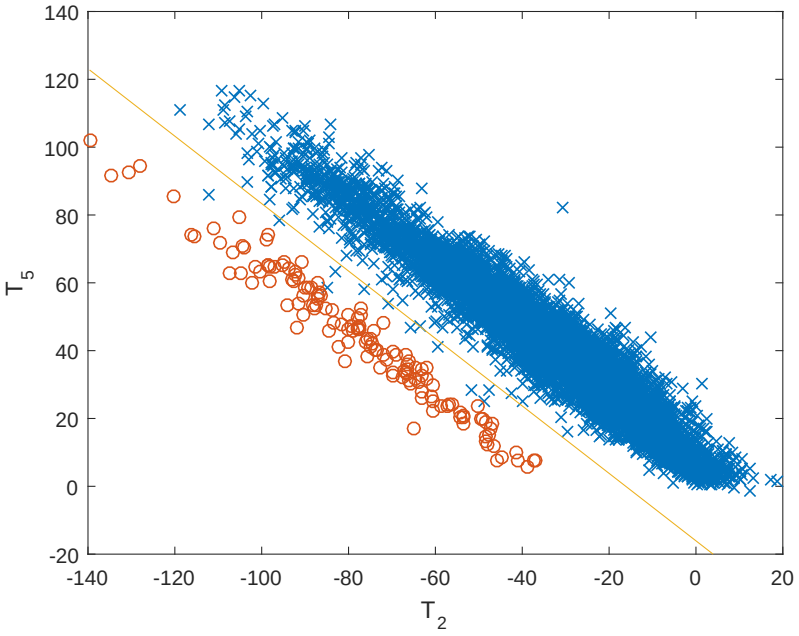


Figure 2.2: *Example of a 2-D hyperplane for two crank positions from cylinder one. Data from two classes, normal combustions (x) and misfires (o), are used and the separating hyperplane is the illustrated with the line.*

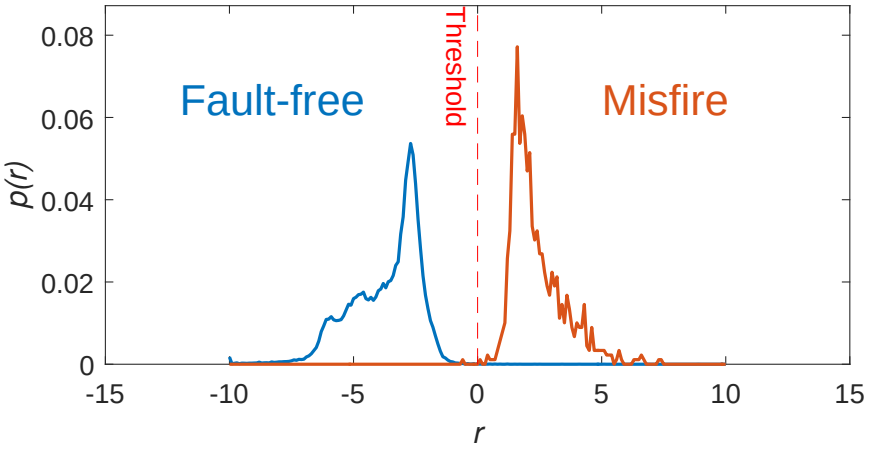


Figure 2.3: *Distribution of the test quantity for fault-free and misfire data.*

will likely diminish. It has been shown that a small deviation in the tooth angles of the flywheel can drastically impact the performance of the misfire detection algorithm. This is due to small differences in the angles between flywheels. By estimating the errors of each vehicle relative to the test vehicle a compensation

can be arrived at. Since it would be impractical to collect the same amount of data for production vehicles as for test vehicles to compute the flywheel errors from before delivery, the errors are estimated on-line.

For the vehicle used for training, data is already available. Data for the torque, load and rpm is used to estimate torque curves for each of the 24 crank positions in a cycle. For points where there are multiple torque estimates available the average is used. To compensate for differences between flywheels the torque curves can be used in other vehicles to estimate the flywheel tooth angle error using an extended Kalman filter (EKF). One added benefit this may also yield is to compensate for slow changes in the error over time.

To compensate for the tooth angle errors, denoted $\mathbf{\Psi}$, the errors are modelled as additive and are included into the calculation for both the angular velocity as well as the torque. The updated equations are then:

$$\omega[k] = \frac{\Delta\theta + \Psi_i}{\Delta t[k]} \tag{2.7a}$$

$$T[k|\mathbf{\Psi}] = \frac{J}{2}\frac{\omega^2[k+1] - \omega^2[k]}{\Delta\theta + \Psi_i} \tag{2.7b}$$

The indexing used here is $k = 1, \ldots, 24$ for the crank positions and $i = 1, \ldots, 11$ for the tooth angle errors, which repeats again for $k = 13, \ldots, 23$ since two revolutions are considered. The last tooth angle error, for $k = 12, 24$ is not estimated directly. Since the errors are deviations from the angles on the training vehicle, the total deviation should sum to zero. Then it follows that the last error on each revolution is given by equation (2.8). This indexing will be used throughout the thesis.

$$\Psi_{12} = -\sum_{l=1}^{11} \Psi_l \tag{2.8}$$

3

Gaussian processes

To improve the misfire classification algorithm, one way may be to use Gaussian processes (GPs) to model the weights of the classifier. As the GP does not store the weights for each operating point, but rather parameters that describe its behaviour over the full range, it could potentially reduce the storage requirements.

3.1 Theory of Gaussian processes

The Gaussian process is a generalisation from the Gaussian, or Normal, distribution in statistics. The Gaussian process is assumed to be continuous, and for each point there is an associated normally distributed random variable [19]. In the misfire detection problem at hand, the Gaussian process will be used to model the weights and bias of the classifier.

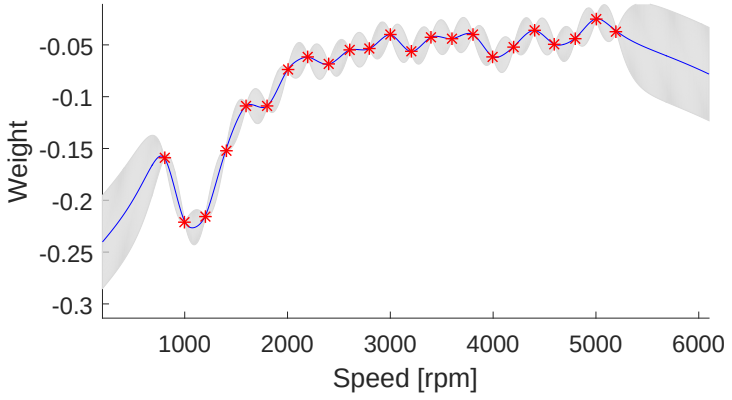
As for structuring the GP mathematically, there are two ways. The first assumes zero mean, and uses basis functions outside the GP to model what would be done by the mean in the second. When assuming the mean to be zero, that changes many of the covariance functions due to how variance is commonly defined. However, as long as the orders of the models used for the basis functions and covariances are not changed the result is functionally the same, even if the numeric values differ. In this work the first representation will be used as it is consistent the Matlab implementation that will be used.

The use of the GP here is in the form of a regression. Using the zero-mean GP notation, the regression is given by equation (3.1). The term $h(x)^T\beta$ consists of the basis functions in $h(x)$ and their corresponding coefficients β. This part of the regression models the behaviour that is due to the actual position in speed of an observation. The order of the basis functions determines how complex behaviours the model is capable of capturing. The second part of the regression is the covariance of the GP, $K(x, x')$, also known as a kernel. In addition to these

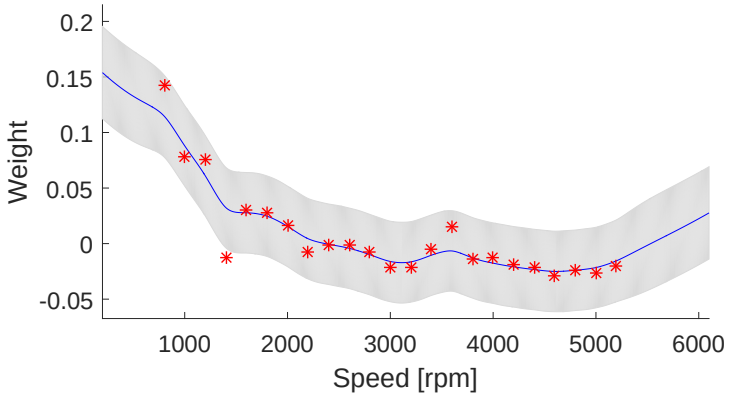
parts there is also an assumption of some zero-mean error, modelled as random noise, which cannot be explained by the GP. The lower the noise, the better is the fit to the data used to estimate the GP, which makes the noise an indicator that gives a rough sense of how well suited a particular GP is.

$$y = h(x)^T \beta + f(x), \; f \sim GP(0, K(x, x')) \tag{3.1}$$

Figure 3.1 (a) illustrates one potential behaviour of a GP, following the actual weights well, but having some uncertainty in-between points with data used for training. In Figure 3.1 (b) the GP does not model the actual weights as accurately, and as a consequence, the prediction error is greater and does not shrink as much close to points where training data was available.



(a)



(b)

Figure 3.1: *Example of a Gaussian process. The estimated GP in (a) follows the training data (∗) nicely, while the shaded area illustrates uncertainty with the 95 % prediction interval. In (b) the GP does not model the behaviour as accurately, and as a consequence the prediction error is greater.*

3.2 Bootstrap

One idea for improving the detection performance is to use bootstrap, most commonly used as a tool for evaluating statistical accuracy [9]. The primary application in this work is to estimate the mean weights, with the idea being to lower the effects of the weights from the SVM that are due to outliers in the data used for training. For the SVM, outliers are particularly troublesome as it uses only the data-points that directly give the classifier, not their distribution in load and speed.

Bootstrap sampling is the idea to select N values with replacement from an original dataset of the same size, N, and do this B times. Each of these B sets is called a bootstrap sample.[9] For each bootstrap sample a classifier can be trained. As this results in B classifiers there needs to be a way to reduce this down to only one classifier, which in this case will use the mean weights over all bootstrap samples. The concept of bootstrap sampling can be illustrated by Figure 3.2, which is a slight adaptation of the illustration used in [9] to explain the concept. It goes from the set X, to the bootstrap samples X^*, and finally the weights of the classifiers estimated from those samples, here denoted by $S(X^*)$.

If the number of bootstrap samples, B, is sufficiently large, the means of the weights and bias should reasonably well classify the data. This is since over a large number of bootstrap samples the effects of the outliers are reduced. Thus, if most of the data is homogeneous, most of the torque estimates in the bootstrap samples for each crank position should be similar, giving similar weights when training the classifiers. If the weights are similar, then the mean is sufficiently robust to be used for arriving at a representative weight. The effects of the outliers are not fully removed with this approach, as they are as likely as any other sample to be included in a bootstrap sample.

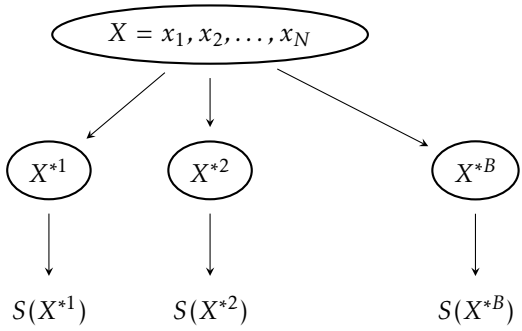


Figure 3.2: *Structure of the concept of Bootstrap, with the original set of size N at the top, followed by the B bootstrap samples and finally the bootstrap realisations, which are in this application the weights of the classifier.*

3.3 Estimation of Gaussian processes

When using Gaussian processes to model the weights of the classifier two different approaches to data partitioning will be evaluated. The first is to estimate weights for non-overlapping ranges and associate each observation to its range centre-point. If each observation was associated with its actual values in speed, the fact that all observations on a range have the same weights would result in an approximately piece-wise constant behaviour. The second way is to estimate weights over a sliding range, thus using overlap between the ranges and potentially smoothing out the behaviour of the weights as part of the data would remain the same over multiple neighbouring ranges. The use of overlapping ranges can also be motivated by the fact that when using non-overlapping ranges there is a limit to how narrow they can be made while still guaranteeing data from both fault-free operations and misfires. This is necessary for training a binary classifier. The limiting factor here is the misfire data since it is scarcer.

Using overlapping ranges allows estimation of weights closer together, which could in turn mean a smoother behaviour of the weights. For a smoother behaviour, a set of lower order basis functions would be able to describe it. There is however a drawback inherent to the SVM that could make a too dense grid problematic: the high influence afforded to outliers. If there are outliers, they can give drastically different weights for a small change in speed, which would asymptotically introduce steps into the weights used to estimate the GP as the distance between range-centres approaches zero. If the behaviour of the weights becomes smoother for a dense grid, that makes it more feasible to use the actual speed of each observation to predict the weights, rather than the range-centre value.

Of interest here is to model the weights as functions of the speed. The basic approach to estimation is to use speed as the predictor and weight as the output. This means that each weight is predicted from speed. One potential problem with this approach is the dependence between the weights in relation to the classification. The six weights and the bias are all related such that the misclassification cost in the SVM is minimized. If the GP is not estimated with a very good fit for all actual weights, the consequence can be a loss of detection performance since the dependence between the weights is lost.

The estimation of the GPs is implemented with `fitrgp` in Matlab. It estimates a Gaussian regression on the form given in equation (3.1). The implementation results in a model that for the same grid requires more storage than the basic implementation, which means that for the GP to be feasible in on-line use it has to use a sparser grid than the original implementation. This will be treated in further detail in Chapter 6.

4

Multiple misfires

The algorithm from [10], as mentioned in Section 1.2.2, works well for a single misfire in a cycle. However, when evaluated on data with multiple misfires in the same cycle, it performs much worse. Also mentioned in 1.2.2, the reason is likely to be the fact that the effect of multiple misfires on the mean results in the misfires no longer being clearly visible in the treated signal.

4.1 Data treatment

Data treatment is essential for the algorithm since the main purpose is to remove the effects of any torque not due to combustion, making the misfire detection easier [10]. If the treatment is not performed, then the estimation of classifiers can become harder since there is potentially more variation in the observations, and misfires may then be less visible in the data.

4.2 Alternatives to mean

The original algorithm removes the mean as its way of treating the data. The mean is potentially not the best measurement of the general behaviour of the torque in the presence of misfires, but the effects of one misfire are not enough to throw the algorithm off. That is not the case for multiple misfires in one cycle.

There are a few ways of dealing with the mean not working for multiple misfires. Firstly, one can entirely replace the mean with the median. Alternatively, the mean can be calculated over multiple cycles, a moving average, such that the effects of the misfires are diminished. For the moving average, there are also multiple choices in terms of how to implement it.

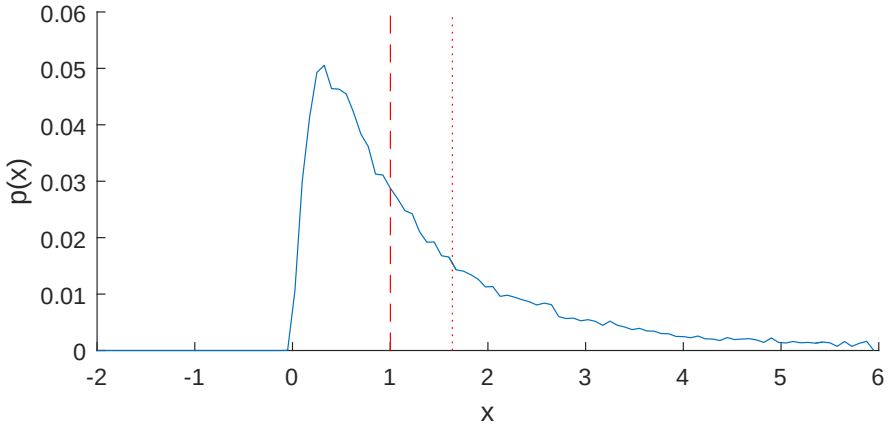


Figure 4.1: *Comparison between mean and median for the log-normal distribution. It illustrates the reaction of the mean (dotted) and the median (dashed).*

4.3 Median

Just like the mean is a measurement of central tendency, so is the median. It is a robust alternative to the mean when working with skewed data [14]. If for example the lower end of the data suddenly became substantially further removed from the other data, making the data negatively skewed, the mean would reflect that with a sudden drop. The median on the other hand would not show any indication of this change as long as the middle sample remained the same. To illustrate this the log-normal distribution is used in Figure 4.1. Here it is evident that the mean has a more pronounced reaction to a skewed distribution.

The median is defined as the middle value for an ordered set with an odd number of data points, whereas if the number of data points is even, it is the mean of the two most central ones. If $X_1, \ldots, X_n$ is a set of n ordered observations, then the median can be expressed as equations (4.1a) and (4.1b). [20]

$$m = X_{k+1},\ n = 2k + 1\ odd. \tag{4.1a}$$

$$m = \frac{X_k + X_{k+1}}{2},\ n = 2k\ even. \tag{4.1b}$$

If the structure of the original algorithm is followed, then the median is computed over one cycle. To increase the usefulness of the median it can alternatively be computed over multiple cycles. With this approach, the increased number of samples reduces the effect of any one cycle, making the median less sensitive to cycles substantially deviating from expectation.

4.4 Moving average

To combat the drop in mean that comes with multiple misfires, a moving average (MA), a mean over multiple cycles, can be used instead. This would mean that the effects of multiple misfires in one cycle have less impact on the moving average since it constitutes a smaller part of the total number of torque estimates used. This approach will work as long as there are not multiple misfires in too many of the cycles used for computing the average. If the number of cycles with multiple misfires becomes too high, it will have the same problem as the original algorithm.

The first choice in implementing the MA is between using a finite impulse response (FIR) or an infinite impulse response (IIR). In this work, implementations of both types will be evaluated. A note on nomenclature is that in [15] and [7] moving average is used to describe how noise is modelled. Moving average is in this work instead used to refer to a treatment method for the torque estimates that takes the mean over multiple cycles.

When designing the moving averages there is also the need to consider the update rate used. This is since it can affect the resulting MA. First, there will be a short section on the possible update rates, and then in the respective sections for FIR and IIR implementations there will be a more thorough look at the consequences for them.

4.4.1 Update rate

Concerning the rate at which to update the MA, there are a few limitations due to the misfire detection algorithm. The slowest possible rate at which to update the MA is once per cycle, this is since the combustions in each cycle are currently treated at the same time. If updating were performed any slower it would mean that multiple cycles would have to use the same MA, effectively varying the position of the current cycle relative to the ones used for computing the MA. The fastest rate at which it is practical to update the MA is once per combustion. It is possible to update more frequently, but then there may be effects from older cycles making classification harder since the same quantity is not removed from all torque estimates in the same cycle. From what was said above about the rates at which to update the MA, there are only two real options, once per cycle or once per observation. Each of the approaches can have an effect on the implementation, and that will be treated in their respective sections below.

4.4.2 FIR implementation

The MA implemented here draws inspiration from the MA-*process* in [7] and the MA-*model* in [15]. Neither has any external inputs, but by using the ARMAX-model [15] that is added, and through selection of the coefficients it is possible to construct what could loosely be called a MAX-model, or following the nomenclature in literature: an X-model, as it does not model the noise. It results in the treatment model as given in (4.2).

The FIR implementation of the moving average requires a few things. Firstly, the length of the average, the number of cycles to include, k, needs to be chosen. Secondly, the weights w_i needs to be set for all k cycles. In addition to these parameters the means m_i for the k last cycles needs to be stored for computing the moving average. In equation (4.2) the expression is given, with M_n denoting the moving average and m_n the mean for the n^{th} cycle. Here M_n is the output, with the input being m_n.

$$M_n = w_n \cdot m_n + w_{n-1} \cdot m_{n-1} + ... + w_{n-k+1} \cdot m_{n-k+1} \tag{4.2}$$

Concerning which rate to use to update the MA can be somewhat simplified for a FIR implementation. Equation (4.3a), with N being the number of crank positions used, is the expression for updating for every torque estimate, which is then simplified to the expression for updating every cycle. That it actually is every cycle can be realised by substituting in (4.3b), which is the number of crank positions in each cycle, and then (4.3c), which is the mean over one cycle. This holds regardless of where a cycle is defined to start in the rotation of the engine, even mid observation. That would however not be practical for the purposes here. One thing that is not included here, but a fully possible intermediate step is the expression for updating for every combustion. This means that if the same weights are used the expressions will be equivalent each time the per-cycle one updates.

$$\begin{aligned} &\frac{1}{N}\left(w_N \cdot t_N + w_{N-1} \cdot t_{N-1} + \ldots + w_{N-24k+1} \cdot t_{N-24k+1}\right) = \\ &\frac{1}{k}\left(\frac{k}{N}\sum_{n=N-3}^{N}(w_n \cdot t_n) + \frac{k}{N}\sum_{n=N-7}^{N-4}(w_n \cdot t_n) + \ldots + \frac{k}{N}\sum_{n=N-24\cdot k+1}^{N-24\cdot k+4}(w_n \cdot t_n)\right) \end{aligned} \tag{4.3a}$$

$$N = 24 \cdot k \Leftrightarrow \frac{k}{N} = \frac{1}{24} \tag{4.3b}$$

$$m_c = \frac{1}{24}\sum_{n=1}^{24}(w_n \cdot t_n) \tag{4.3c}$$

4.4.3 IIR implementation

Similarly to the FIR implementation having likenesses with literature's MA, the IIR is instead related to the AR-*process* and AR-*model* [15, 7]. When adding the input to the model and assuming no noise it coincides with the ARX-model [15].

The implementation of the moving average as an IIR requires three things. The mean of the current cycle, m_n, the moving average for the previous cycle, M_{n-1}, which contains information about all the previous cycles, as well as some factor for weighing together new and old information, α. This factor determines how much the new cycle and the previous MA influences the new MA. A large factor allows the mean of the current cycle to highly influence the MA, while the

effect of previous MAs die away quickly. If α is chosen to be small, each new mean has little impact on the MA, while the influence of old MAs persist longer. The resulting expression is found in equation (4.4).

$$M_n = \alpha \cdot m_n + (1 - \alpha) \cdot M_{n-1} \tag{4.4}$$

As was mentioned in the beginning of this section, the IIR implementation can be considered an ARX-model. This is since the input, the mean of the current cycle, m_n, is put in relation to one or multiple outputs, M, for a cycle. This is evident if equation (4.4) is rewritten as (4.5).

$$\frac{1}{\alpha} M_n - \frac{1 - \alpha}{\alpha} M_{n-1} = m_n \tag{4.5}$$

5

Flywheel tooth angle error

The difference in flywheels between vehicles, be it from variations in the teeth or off-centre mounting, means that the classifier is not performing as well on other vehicles than the one used for training. This has been shown in [11] to be possible to compensate for by estimating the flywheel tooth angle errors.

5.1 Points of estimation

There are multiple possible points in the processing of the torque where the tooth angle errors can be estimated. The one used in the original algorithm is the normalised torque estimate, i.e. after the removal of the mean and normalisation with load, see equation (2.5).

Apart from the original point of estimation, there are two natural alternatives from the structure of the algorithm. Those are the torque signal after removing the mean and the raw torque estimate before any manipulations have been applied. The different points of estimation are illustrated in Figure 5.1.

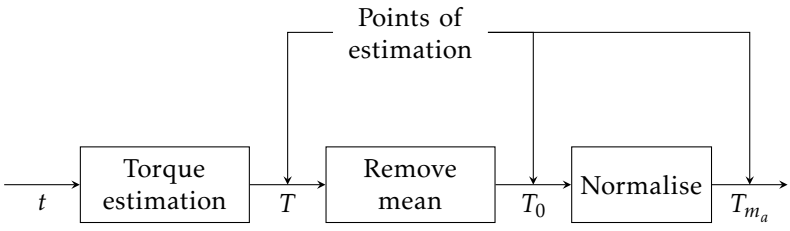


Figure 5.1: *The signal processing with the possible points of estimation indicated.*

5.1.1 After removal of the mean

Estimating the torque after only removing the mean, but before normalising with the load means that the torque estimate being used is:

$$T_0[k] = T[k|\boldsymbol{\Psi}] - \frac{1}{24}\sum_{l=1}^{24} T[l|\boldsymbol{\Psi}] \tag{5.1}$$

This alternative retains the benefits of removing torques not due to combustion, as it is simply a version of the original treatment, but scaled by the load. Although, as each cycle varies around zero, the average torque estimate for each crank position should be a good representation for most of the estimated torques associated with it. Since this treatment is only scaled by load compared to the normalised treatment, the estimated errors should be rather close. As each observation is scaled by its actual load, there may however be a slight skewing of the torques within the same range, and the consequences would depend on how the torques vary with changing load.

5.1.2 On raw torque estimate

In the processing of the data from measured time to estimated torque, first the angular velocity is computed through equation (2.1a), followed by the torque estimate in equation (2.3c). While misfires are visible directly in the angular velocity, it is easier to perform the detection on the torque estimates [10]. As the torque and angular acceleration are directly related through (2.1b), with J assumed constant [10], it would be just as feasible to use the latter to perform the misfire detection if no other treatment was to be used. As torque is more commonly used when talking about engines, it seems more intuitive and practical to use it. If it is possible to achieve an accurate torque estimate through using the correct moment of inertia, then it becomes even more practical to use torque, as it could be useful for other computations on-line.

The obvious risk with finding the errors directly from the raw torque estimates is that variations in the other torques, the very reason for removing the mean, can affect the estimates of the errors. Even if the torque map uses the mean torque for each point to represent the behaviour this may not be enough.

5.2 Off-line optimisation

For comparing the performance of the difference points of estimation, it is possible to estimate the errors that minimise the difference between compensated data and the torque map. This is done with all data already collected and can thus be performed off-line through optimisation. The result from the optimisation can then also be used to evaluate the accuracy of the on-line implementations after their estimates have converged.

From equations (2.7a) and (2.7b) the model from time at the flywheel to torque estimate is already known. It is parameterised by $\boldsymbol{\Psi}$ which can be estimated

through minimisation of the mean square error by solving equation (5.2). This approach to finding the errors is only practical on collected data and performed off-line due to the high computational demands of finding an optimum.

$$\arg\min_{\boldsymbol{\Psi}} \frac{1}{N} \sum_{k=1}^{N} \left(T_{map} - \hat{T}(k|\boldsymbol{\Psi}) \right)^2 \tag{5.2}$$

This is a non-linear programming problem, and it is possible to formulate it as a constrained or unconstrained problem depending on how Ψ_{12} is treated. The problem is constrained if Ψ_{12} is used in the cost function and the constraint given by equation (2.8). It is possible to reformulate the problem to be unconstrained by substituting Ψ_{12} by its definition in the objective function. The set of errors, $\boldsymbol{\Psi}$, is convex as each element can take any value on $\mathbb{R}$ and should remain small as that creates the best fit with the map. Values of more than one revolution, or even $\Delta\theta$ indicates that something in the optimisation did not work properly, or that the data collected is bad, as the expectation is $\Psi_l \ll \Delta\theta,\ l = 1, ..., 11$. The objective function is also convex, as it will grow with any change to an element in $\boldsymbol{\Psi}$ from its optimal value, regardless of the direction of change. The objective function is also infinitely differentiable in $\boldsymbol{\Psi}$, making it a problem to which there are multiple algorithms for solving [22].

6

On-line considerations

One thing that the different parts making up this work have in common is that they, in addition to improving performance, need to be realisable on limited hardware in production vehicles to be considered practical improvements. While the more demanding computations, such as training the SVM and estimating the GP are performed off-line and can benefit from substantially more powerful hardware and no real-time constraints, many of the computations must be performed on-line to have any real-world value. The two main limiting factors on-line are memory and processor power. There are also real-time requirements for the system to be feasible.

6.1 Classifier

The implementation of the classifier used in the existing code for the work upon which this thesis is based used Matlab's built in functions for SVMs both when training the classifier and performing the classification itself. Training is done off-line and does not need to be as restrictive with computations. However, the classification needs to be done with on-line performance in mind, and it is then not practical to use the `predict` function for computing the test quantity.

One thing that can impact training performance is if the SVM normalises the training data or not. This work uses normalisation as it can, as a general practice, be shown to significantly improve the performance both when classifying and training [4]. The particular normalisation used here is zero-mean normalisation. Using the notation x_{old} for the original data, x_{new} for the normalised data, m_x for the mean of the predictor and σ_x for its standard deviation, normalisation is performed by equation (6.1) for each of the predictors, resulting in each having zero mean and unit variance.

$$x_{new} = \frac{x_{old} - m_x}{\sigma_x} \tag{6.1}$$

When training the SVM in Matlab the result is returned as its own type of object, which contains more data than necessary for use on-line. The SVM used here is linear and the separating hyperplane is defined by six coefficients, β, and one intersect, b. When using Matlab's built in function for classifying a new observation, the normalisation is performed automatically. It is possible to remove the need to store the means and standard deviations used for normalisation by incorporating them into the coefficients and intersect of the SVM. Equation (6.2) gives the relationship between new (β^*,b^*) and old (β,b) weights and intersect. μ is the mean and σ the standard deviation for each crank position of the data used to train the SVM. This reduces the storage need by twelve values for each cylinder and range that would otherwise be needed to perform the normalisation.

$$\beta^*_{crnk} = \frac{\beta_{crnk}}{\sigma_{crnk}} \tag{6.2a}$$

$$b^*_{crnk} = -\mu_{crnk} \cdot \beta^*_{crnk} + b_{crnk} \tag{6.2b}$$

6.2 Gaussian processes

Much like for the SVM in the previous section the Gaussian process can be implemented by built-in functions in Matlab. This also presents the same drawbacks in terms of being overly complex for an effective implementation on-line. It does however not simplify as well as the SVM. There are parts of the object in Matlab that can be removed, but there is still the need to store all coefficients for the basis function and the parameters for the covariance function. The Matlab implementation of the GP stores at least as many parameters for the covariance alone as the default classifier does for the same split. The result of the estimation is a discrete function only defined for the points at which data is provided. In addition, there is also the need to store the coefficients for the basis functions, but its storage need is only determined by its order, contributing much less. Therefore, the GP needs to use fewer points for its prediction, or describe the behaviour with a continuous function, to rival the basic algorithm in terms of storage.

The representation used for the regression, assuming the GP to have zero-mean, means that there is a set of basis functions with associated coefficients. Depending on how the regression is designed, the order of the basis functions may be higher than one. This can rapidly increases their contribution to the computational load. In the case at hand, with the predictor being speed, the number of combinations is manageable, and can from a computational standpoint be feasible on-line, but if the number of predictors was to increase it would also mean an rapidly increasing number of coefficients. Even if it may be desirable to keep the basis functions linear, it may not actually be possible from a performance standpoint. The combination of the basis functions and the covariance function aims to accurately describe the weight as it varies with speed. This means that

the simpler the covariance function is, the more complex the basis functions have to be to account for that. Since the covariance depends on distance to other values for the predictor and the basis functions commonly depends on the values of the predictors, not their differences, the consequence of trying to capture certain behaviours with the wrong one may result in an overly complicated regression.

6.3 Multiple misfires

The impact of the treatment of multiple misfires depends on which method is used. This is due to the fact that the averaging methods can use the torque estimates directly and also often simplifies well, whereas the median requires sorting of the data, effectively needing to do comparisons between many different values.

6.3.1 Moving average

The alternative treatments proposed with MAs work directly on the torque data, in the case with the IIR only α and the previous value for the MA needs to be stored. The FIR needs to save the mean for each cycle to be included in the moving average. Were the coefficients to be different between cycles, or within cycles, the needs may change somewhat. If the coefficients are constant within each cycle, the coefficient for each cycle needs to be stored once and applied for each computation of the MA. In fact, any coefficient that is not varying within its corresponding cycle can be put directly into the code without the need to store it, making the means the only values needing to be stored. If the mean for one cycle has the same weight applied for all crank positions, only the mean, not all torque estimates, needs to be stored. Much of this also holds for different coefficients within a cycle, only with the added requirement of storing all torque estimates, rather than the sums for each cycle.

6.3.2 Median

For using the median as the replacement, the requirements will be higher since it requires knowing the middle one or two torque values from a list sorted by magnitude when using an integer number of cycles. This sorting will impact the complexity of the computation. The sorting introduces the need to perform a lot of comparisons between the torque estimates. In a simple form, the sorting could be implemented with a bubble sort, with not so great performance. Requiring in the best case as many comparisons as the number of torque estimates minus one, i.e. starting with the data sorted. If the data starts sorted in reverse, the number of comparisons needed is then instead squared and corresponds to the worst-case scenario. There are more effective algorithms for sorting the torque estimates, and there may be merit to using them if the size of the data set is appropriate. However, assuming only a few cycles are used for computing the median, the burden of sorting the torque estimates remains, if not low, at least manageable.

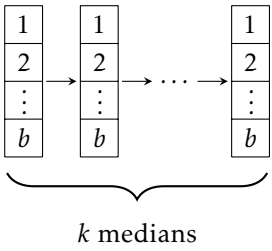


Figure 6.1: *Illustration of a remedian. It consists of k medians with b values in each.*

One possible way of reducing the computational load and the storage needs for computing the median is to implement the remedian. The remedian is an estimator of the median that can reduce the need for both storage and computational time. This is done by taking the median over a small number of measurements, b, and then taking the median for b such medians and repeating k times [21]. The concept is illustrated in Figure 6.1, which is adapted from [21]. Assuming $n = b^k$, the storage needed for the torque estimates to compute the remedian is $b \cdot k$, rather than n. Depending on the number of cycles the median is computed over, the savings in needed storage can be substantial. The time needed to compute the remedian is asymptotically the same as for the median, but for the number of samples used here the remedian requires less time to compute.

The use of remedian is further discussed in [2], where a more appropriate implementation for continuously arriving (streaming) data is studied. Also mentioned is the fact that the remedian performs better on bigger sets of data, which is possibly why it would not be a viable alternative to the median for this particular use. The error between remedian and median for different b and n evaluated in [2] reinforces that, where for a small n the relative error is rather large, and diminishes with growing n regardless of b. The choice of b is best made to be small, $b < 10$, as it is not simple to optimally find the median from more samples [2].

As the intended use of the median in this work is over a limited number of cycles, while the remedian arrives at one estimate for n samples, some modification is needed if the remedian is to be implemented. One straightforward way of achieving this is slightly alter the last layer: let the median of the previous layer be pushed to the top of the highest layers list of medians, forcing out the oldest one. Each of the other layers is emptied when it is full. This would mean that there is always an estimate of the median for the n last torque estimates.

6.4 Flywheel tooth angle error compensation using Extended Kalman Filters

While evaluating and comparing the feasibility of different points for where to estimate the flywheel tooth angle error is possible off-line, that particular approach

is not practical for implementation on-line due to the time required and the fact that it needs all data to be available before working, and takes multiple passes. As was mentioned in Chapter 1 the intention is to instead use the EKF to estimate the errors on-line. This makes it possible to estimate the errors every time data from a new cycle becomes available. The basic EKF presented below is based upon the presentation in [5], and uses its notation.

The EKF is an extension of the Kalman Filter (KF) to work for non-linear systems. The EKF used here approximates the non-linear system through first order Taylor expansion. Like the KF, the EKF assumes zero-mean Gaussian noise. The state-space model of the dynamic system that the EKF requires is:

$$x[k+1] = f(x[k], u[k], v[k]), \;\; v[k] \sim N(0, Q) \tag{6.3a}$$

$$y[k] = h(x[k], u[k], e[k]), \;\; e[k] \sim N(0, R) \tag{6.3b}$$

Using the state-space model the EKF can be expressed as two parts, a prediction step (6.4.1) and an update step (6.4.2).

6.4.1 Prediction update

The prediction update gives the state vector that would result from a certain input, given the current state vector (6.4a). The fact that prediction is performed from old data means that the state covariance increases (6.4b).

$$\hat{x}[k+1|k] = f(\hat{x}[k|k]) \tag{6.4a}$$

$$P[k+1|k] = Q[k] + f'(\hat{x}[k|k])P[k|k](f'(\hat{x}[k|k])^T \tag{6.4b}$$

For the problem of estimating the flywheel tooth angle errors, the errors are assumed constant as any change from normal wear would be over a very substantial amount of time. With this assumption the state equation is then constant (6.5a). It then follows that its Jacobian is an identity matrix (6.5b). This allows some simplification of the prediction update. The predicted state vector will be the current state vector (6.5a), and predicting the state covariance is reduced to a simple addition (6.5c).

$$\hat{x}[k+1] = \hat{x}[k] \tag{6.5a}$$

$$f'(\cdot) = I_{11} \tag{6.5b}$$

$$P[k+1|k] = Q[k] + P[k|k] \tag{6.5c}$$

This simplification could also have been done by realising that the state equation is linear, and then use the KF prediction update instead. This would yield the same results.

6.4.2 Measurement update

When new information becomes available about the system, the EKF can be updated to use this information, see equations (6.6a)-(6.6e). Since there is already a prediction of the state vector available, the measurement update corrects for any error between the new information and the predicted state.

From a computational standpoint, it would be of interest to simplify the measurement equation, as a linear equation would mean an easier update. However, as the measurement equation (2.7b) divides by the state, simplification is not possible. The most computationally demanding part of the measurement update is the inversion of (6.6b). If the errors can be estimated from the raw torque estimates with negligible performance loss the matrix $h'(\cdot)$ would become considerably more sparse as each row not using Ψ_{12} would go from having no zero-entries to only having two non-zero entries. Unfortunately this does not translate to an as sparse $S[k]$, but it is still more sparse than what the original algorithm would have resulted in.

$$\epsilon[k] = y[k] - h(\hat{x}[k|k-1]) \tag{6.6a}$$

$$S[k] = R[k] + h'(\hat{x}[k|k-1])P[k|k-1]h'(\hat{x}[k|k-1])^T \tag{6.6b}$$

$$K[k] = P[k|k-1](h'(\hat{x}[k|k-1]))^T S[k]^{-1} \tag{6.6c}$$

$$\hat{x}[k|k] = \hat{x}[k|k-1] + K[k]\epsilon[k] \tag{6.6d}$$

$$P[k|k] = P[k|k-1] - P[k|k-1](h'(\hat{x}[k|k-1]))^T S[k]^{-1} h'(\hat{x}[k|k-1])P[k|k-1] \tag{6.6e}$$

6.4.3 Constant gain

One proposed way to lessen the computational burden of finding the flywheel tooth angle errors is to use the constant gain EKF (CG-EKF) [13], rather than the standard EKF. What the CG-EKF does is it removes the need to recompute the gain for each measurement update. The gain, K, is instead calculated using a fixed covariance matrix for the measurement noise. This also removes the most demanding part of the computations, the matrix inversion for computing K, at least all but one time off-line.

The usefulness of the CG-EKF can possibly be increased by using multiple gains depending on how close the estimated torque is to the torque curve. Using two gains would lessen the need to balance the speed of the filter with its variance, allowing for higher gain initially and a more conservative one when the estimates have converged. This means it could be possible to have fast convergence initially, while achieving low variance when the estimates have stabilised. To determine when to perform the switch the square sum error between the prediction from the filter and the torque map can be used.

6.4.4 Reduced update rate

Much like how the potential usefulness of the CG-EKF can be increased by having multiple gains, it and the EKF can be less computationally intensive if they are updated much more sparsely after converging. This can be done much in the same way as the multiple gain approach for the CG-EKF, using the difference between torque map and predicted torque to determine when the filter has converged enough to no longer need to be run as frequently. The reduction in computational load for the EKF with this approach would mean that if it becomes a feasible alternative, it would likely rule out the CG-EKF as an interesting alternative. Just like for the multiple gain CG-EKF the square sum error can be used to switch the update-rate.

7

Evaluation

7.1 Classifier

Using the design method for the classifier proposed in [10] results in a classifier that performs similarly for the four-cylinder engine as it did for the six-cylinder one.

The classifier was initially only segmented in speed, using both five and four ranges, where the four-range one merged the two highest speed ranges into one. The five ranges had midpoints [750 1650 2500 3700 5000] rpm with limits symmetrically between each pair of neighbouring midpoints. For the four-range classifier the 3700 rpm and 5000 rpm ranges where merged into one yielding the midpoints [750 1650 2500 3700] rpm. The two alternative segmentations have similar performance as seen in Table 7.1.

Table 7.1: *Performance of basic classifier.*

	Five ranges		**Four ranges**	
NF	60366	-	60364	-
MF	893	-	893	-
MD	1	0.1119 %	1	0.1119 %
FA	29	0.0480 %	31	0.0513 %

Looking at the behaviour in Figure 7.1 of the weights of the classifier in both speed and load shows that the weights varies greatly with speed, but also with load for one speed. Based on this the classifier was also divided by load, and looking at how the data is distributed in load and speed, Figure 1.1, the midpoints of the ranges were initially chosen as [0.1 0.3 0.5 0.8] g/revolution, but removing the 0.8 g/rev one yields nearly identical performance with centres of the ranges as [0.1 0.3 0.5] g/rev. Similarly to the speed ranges the limits for the ranges were

chosen centred between the midpoints.

The resulting classifier delivers higher performance than if segmentation was only performed with regards to speed. However, when combining the five range segmentation of speed with segmentation in load, there was not enough data to perform detection for the 5000 rpm and 0.1 g/rev range on the default data. The data from that range was ignored when performing the classification. The two alternatives still have very similar performance seen in Table 7.2. To mitigate the problem of not classifying over the full range the classifier for the 3700 rpm and 0.1 g/rev range was tried on the 5000 rpm and 0.1 g/rev range as well. The results was then the same as for the four range classifier.

Table 7.2: *Performance for classifier divided in both speed and load.*

	Five ranges		**Four ranges**	
NF	60356	-	60372	-
MF	894	-	894	-
MD	0	0 %	0	0 %
FA	21	0.0348 %	23	0.0381 %

7.2 Gaussian processes

When estimating a Gaussian process from the weights for the initial classifier in Section 7.1 the performance was lower than for the basic classifier, with MD at 0.3356 % and FA at 0.6772 %. The classifier weights as well as the weights predicted from the GP for cylinder one are plotted in Figure 7.2.

The results above only use the split in speed from the classifier, meaning that each observation uses the weights predicted for its range centre-point. A more realistic use of the GP is to use predictions that are more local to each observation, predicting the weights from its actual speed rather than its speed-range. From Table 7.3 it can be seen that the performance improves slightly.

Table 7.3: *Results for GP using actual speed for prediction.*

	GP	
NF	60257	-
MF	891	-
MD	3	0.3356 %
FA	138	0.2285 %

Since part of the idea behind using a GP to model the weights was higher resolution in speed between the weights, the next logical step in the evaluation is to estimate the GP from much closer spaced range-centres. This is however not without problem since there is not enough data to properly perform detection over the whole speed range when the step-length becomes too short. This was mitigated by using overlapping ranges, allowing for close centres and essentially

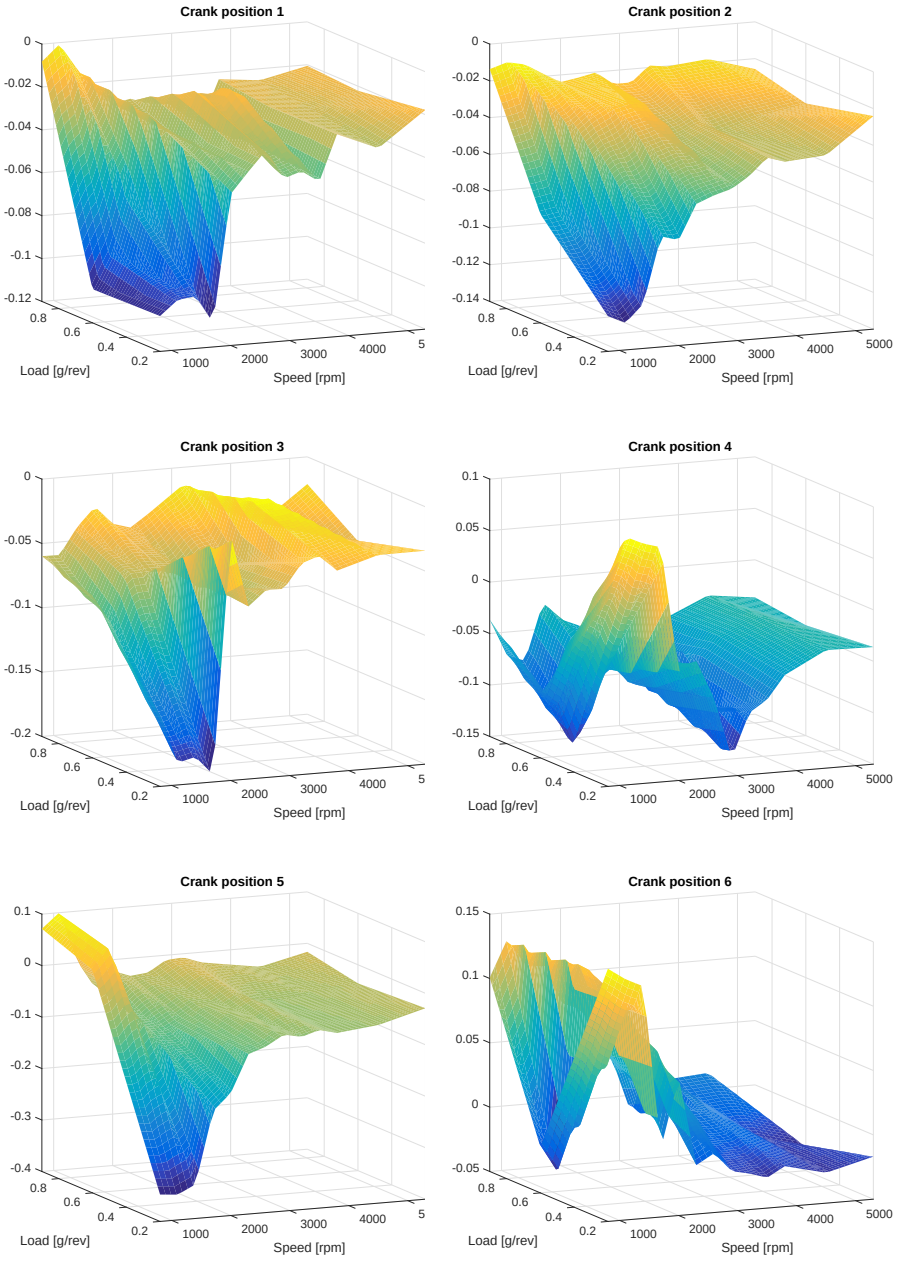


Figure 7.1: *Weights for the basic classifier.*

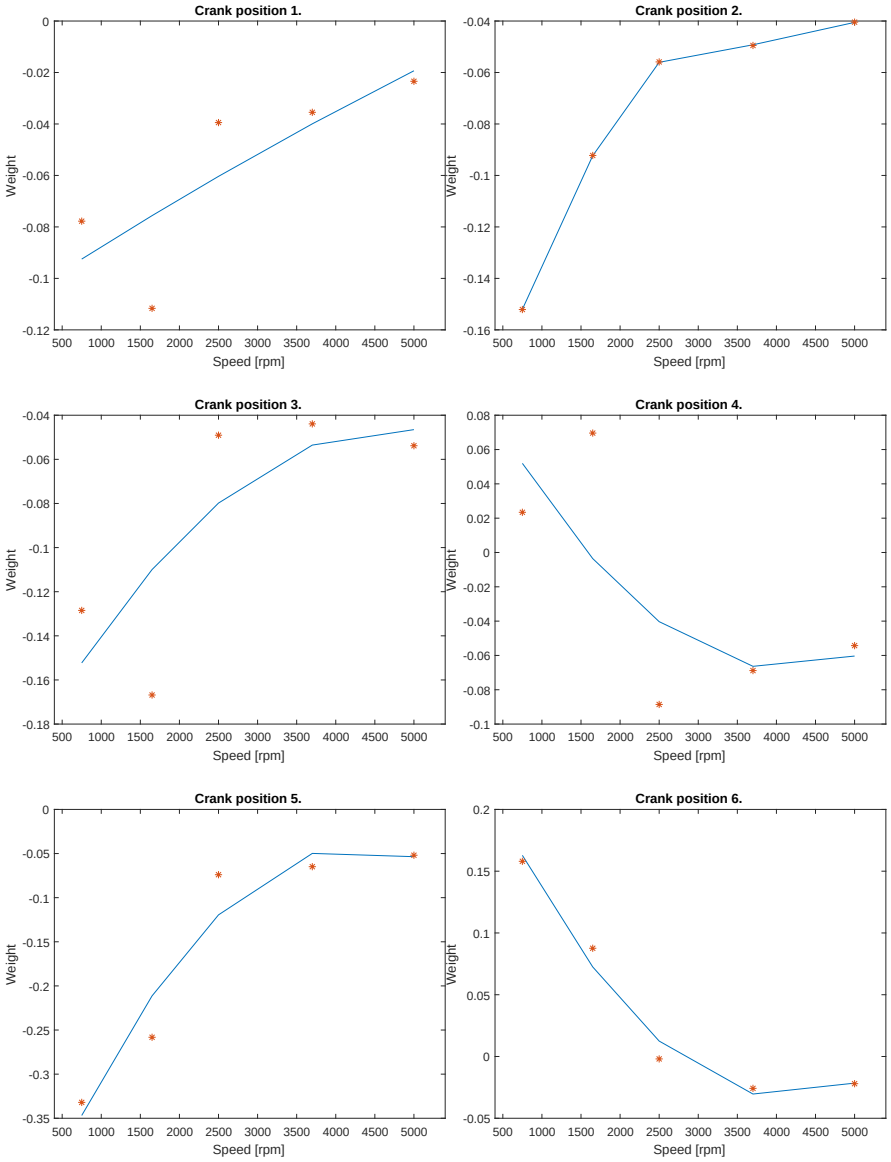


Figure 7.2: *Plots of weights from classifier (red) and GP (blue) for cylinder one.*

performing smoothing of the weights in speed. Performance for this approach using centres from 800 to 5300 rpm every 100 rpm with width 500 rpm, is listed in Table 7.4.

Table 7.4: *Performance for GP on overlapping ranges.*

	GP			
	Range centres		**Interpolated weights**	
NF	60350	-	60220	-
MF	891	-	890	-
MD	3	0.3356 %	4	0.4474 %
FA	45	0.0745 %	175	0.2898 %

Using higher resolution to estimate the GP does not give higher performance, but rather misclassifies more observations than the original classifier. The behaviour of the GP compared to the weights used to estimate it are found in Figure 7.3.

One way of improving the performance when using a GP as proposed in Chapter 3 is to use bootstrapping. First the impact of bootstrapping was evaluated on the basic classifier, and then on the GP. For the basic classifier with high density overlapping ranges the performance, Table 7.5, improved slightly for a low number of bootstraps, while a higher number did not have the same impact. The corresponding results for the GP can be seen in Table 7.6.

Table 7.5: *Performance using only bootstrap.*

	Number of bootstraps					
	5		**10**		**100**	
NF	60227	-	60225	-	60225	-
MF	946	-	946	-	946	-
MD	1	0.1056 %	1	0.1056 %	1	0.1056 %
FA	29	0.0481 %	31	0.0514 %	31	0.0514 %

Table 7.6: *Performance for GP using bootstrap weights.*

	Number of bootstraps					
	5		**10**		**100**	
NF	60362	-	60371	-	60371	-
MF	893	-	893	-	893	-
MD	1	0.1119 %	1	0.1119 %	1	0.1119 %
FA	33	0.0546 %	24	0.0397 %	24	0.0397 %

As modelling the weights using GP and bootstrap gives good performance while only partitioning the data in speed, it is of interest to see what results can be achieved by partitioning the data both in speed and load. The GPs will however

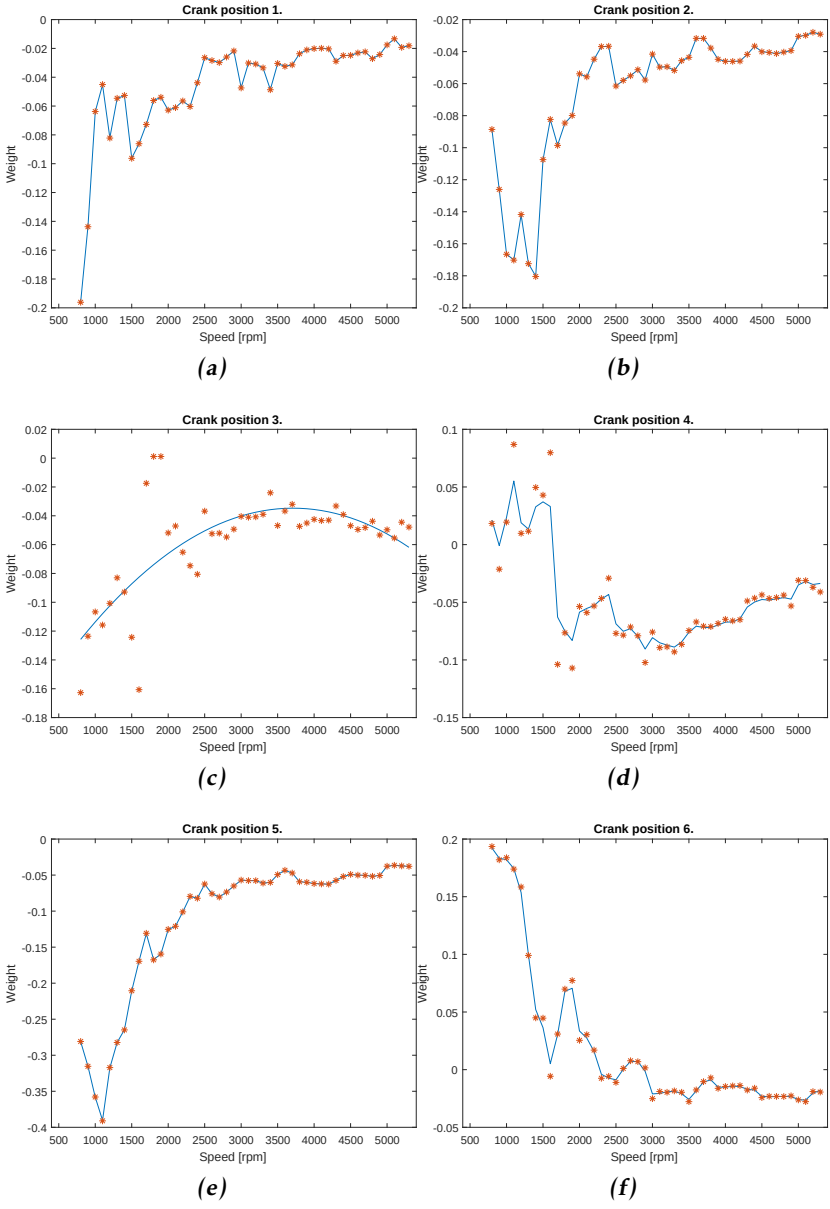


Figure 7.3: *Plots of weights from classifier (red) and high resolution GP (blue) for cylinder one.*

still only use the speed as their predictors. The results from this addition are listed in Table 7.7.

Table 7.7: *GP performance for data partitioning in both load and speed.*

	GP			
	Range centres		**Interpolated weights**	
NF	60368	-	60233	-
MF	893	-	890	-
MD	1	0.1119 %	4	0.4474 %
FA	27	0.0447 %	162	0.2682 %

7.3 Multiple misfires

From Section 7.1 the performance when dividing the data by both load and speed outperformed splitting only by speed. The evaluation performed below divides the data by both load and speed since it is the most relevant to practical applications with its higher performance. The division that will be used is the one with three load and four speed ranges arrived at in Section 7.1. For comparison with the alternative treatments, the first evaluation to be performed is using the basic algorithm on multiple misfires, giving a baseline against which to compare the alternatives, Table 7.8.

Table 7.8: *Performance of the original algorithm on data with multiple misfires.*

	Number of observations	**Misclassification rates**
NF	25823	-
MF	302	-
MD	3	0.9836 %
FA	72	0.2780 %

The goal with the alternatives to removing the mean is to have a quantity that reacts less to misfires than the mean. To this end, the first result of interest is the quantity to be removed, in Figure 7.4 plotted together with the estimated torque signal.

With an initial visualisation of the treatments reactions, the next step is actual performance on data. Since the single misfire performance of the algorithm is already established the first evaluation is of how the alternative treatments perform for single misfires, Table 7.9. This so that no improvement in multiple misfire detection is at the cost of single misfire performance, or if some degradation of single misfire performance is allowed, it can easily be compared to the gain in performance for multiple misfires, Table 7.10. The versions of the median and MA used here are over 100 cycles and with $\alpha = 0.9$, respectively. This follows from the investigations presented in Sections 7.3.2 and 7.3.1.

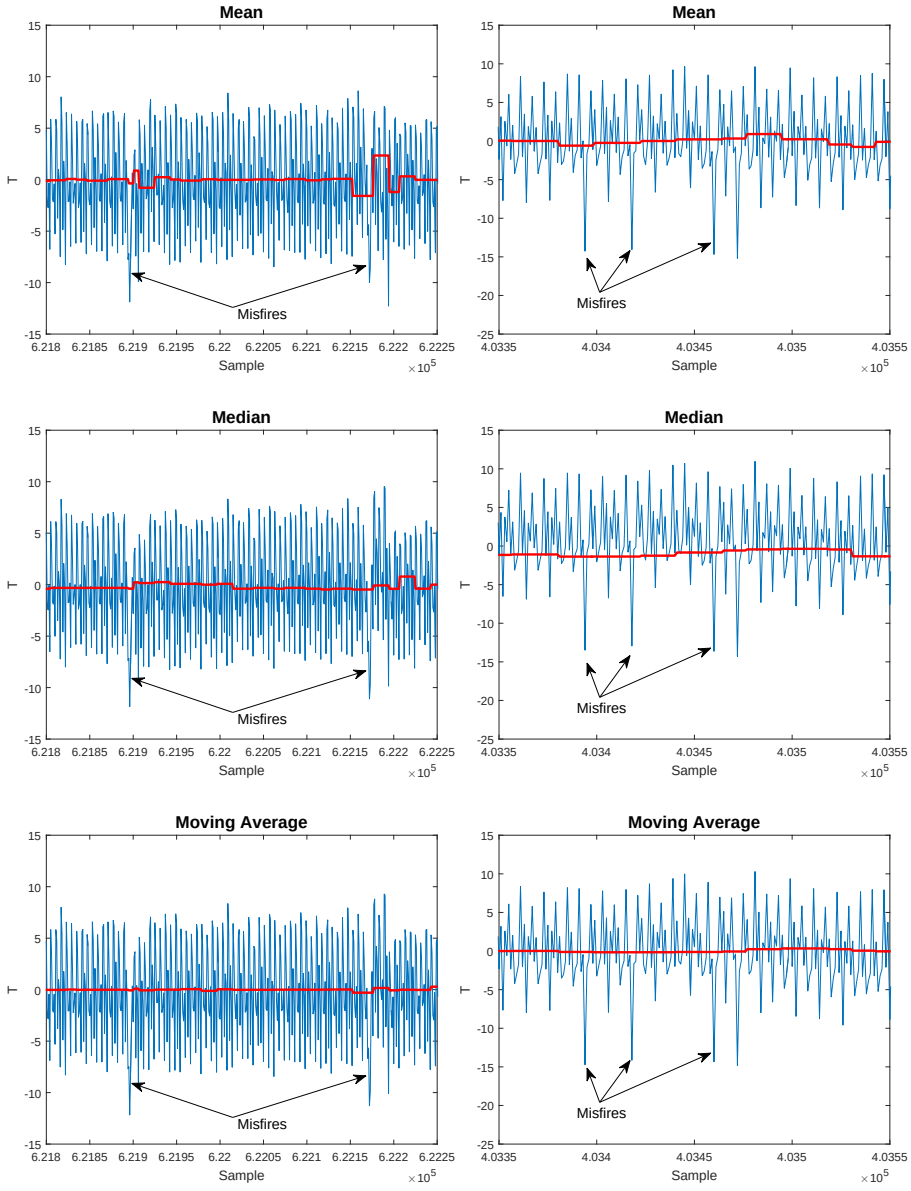


Figure 7.4: *Effects of signal treatments. Left column is single misfires, right multiple misfires. Treatment to subtract in red, treated torque in blue.*

Table 7.9: *Performance of alternative treatments on data with single misfires.*

	Untreated		Mean	
NF	60293	-	60372	-
MF	892	-	894	-
MD	2	0.2237 %	0	0 %
FA	102	0.1689 %	23	0.0381 %

	Median		MA	
NF	60113	-	60371	-
MF	888	-	893	-
MD	6	0.6711 %	1	0.1119 %
FA	282	0.4669%	24	0.0397 %

Table 7.10: *Performance of alternative treatments on data with multiple misfires.*

	Untreated		Mean	
NF	25811	-	25823	-
MF	301	-	302	-
MD	4	1.3115 %	3	0.9836 %
FA	78	0.3013 %	72	0.2780 %

	Median		MA	
NF	25806	-	25826	-
MF	302	-	301	-
MD	3	0.9836 %	4	1.3115 %
FA	83	0.3206 %	69	0.2665 %

7.3.1 Comparison of implementations of moving average

As was mentioned in Section 4.4 there are two ways of implementing the MA, as a FIR or an IIR. Presented below are some of the results pertaining to the implementations.

In Table 7.11 the performance of the FIR implementation is listed for averages over one and 50 cycles. They were selected for giving the highest performance for data with single and multiple misfires respectively. There were a number of intermediary number of cycles evaluated, but none of them gave as good performance, chosen as firstly giving a low number of MDs and then a low total number of misclassifications for multiple misfires. For single misfires the behaviour was a general worsening of performance when the number of cycles increased, which is illustrated by the listed results, even if somewhat limited. The behaviour of performance for multiple misfires is an initial worsening turning in to a slight improvement.

Table 7.11: *Performance for FIR MA on data with single and multiple misfires for averages over 1 and 50 cycles.*

	1			
	Single		**Multiple**	
NF	60362	-	25819	-
MF	893	-	302	-
MD	1	0.1186 %	3	0.9836 %
FA	33	0.0546 %	76	0.2935 %

	50			
	Single		**Multiple**	
NF	60296	-	25826	-
MF	892	-	302	-
MD	2	0.2237 %	3	0.9836 %
FA	99	0.1639 %	69	0.2665 %

Using the IIR-implementation of the MA yielded the results in Table 7.12. Just as for the FIR the results here were chosen such that the number of MDs was as low as possible, and then the lowest number of total misclassifications. Here α = 0.4 gave the best performance, followed by α = 0.9. The behaviour of the performance for single misfires was principally that as α grew the performance improved, consistent with the results in Table 7.12. For data with multiple misfires the number of MDs rose slightly with growing α before coming down slightly as α neared 1, while the number of FAs instead dropped slightly before rising again, although not reaching as high. In general, the number of misfires is low, and a change by one is a significant jump in the rates.

Table 7.12: *Performance for IIR MA on data with single and multiple misfires for α = 0.4 and 0.9.*

	$\alpha = 0.4$			
	Single		**Multiple**	
NF	60357	-	25830	-
MF	892	-	301	-
MD	2	0.2237 %	4	1.3115 %
FA	38	0.0629 %	65	0.2510 %

	$\alpha = 0.9$			
	Single		**Multiple**	
NF	60371	-	25826	-
MF	893	-	301	-
MD	1	0.1119 %	4	1.3115 %
FA	24	0.0397 %	69	0.2665 %

7.3.2 Effects of number of cycles included in median

As presented in Section 4.3, the design decision for implementing the median is the number of cycles to include. Looking at results for using from one up to 100 cycles in the median at best resulted in the performance in Table 7.13. This performance was selected as in Section 7.3.1, with the added constraint that one should be for a low number of cycles. The general behaviour of the single misfire performance is a sharp initial drop in the number of misclassifications, followed by a slow fall and subsequent rise to about the same level as for two cycles. For data with multiple misfires the behaviour is similar, with an initial sharp drop, but then a slight decline as the number of cycles increases. For both cases the number of MDs varies quite a lot, but has a trend of a slight decrease.

Table 7.13: *Performance for median on data with single and multiple misfires for 2 and 100 cycles.*

	2			
	Single		**Multiple**	
NF	60120	-	25788	-
MF	888	-	299	-
MD	6	0.6714 %	6	1.9672 %
FA	275	0.4553 %	101	0.3901 %

	100			
	Single		**Multiple**	
NF	60113	-	25806	-
MF	888	-	302	-
MD	6	0.6711 %	3	0.9836 %
FA	282	0.4669 %	83	0.3206 %

7.4 Flywheel tooth angle error compensation

The rational for estimating the flywheel tooth angle errors elsewhere than on the normalised torque signal is to reduce the required computational resources for on-line use. The errors have been estimated for each point along the treatment. That is: normalised torque, after removal of the mean and for the raw torque estimate.

Initially a baseline was established by using the uncompensated algorithm on the calibration set. The performance from this evaluation is listed in 7.14, alongside the performance for the validation data set from the vehicle used for training. The normalised torque estimate is used here, and the classifier uses the four-speed and three-load range split presented in Section 7.1.

As the first step in the compensation, a reference torque map is estimated for the vehicle used for training. The results for the three points of estimation for a

Table 7.14: *Performance of the original algorithm on another vehicle without compensation.*

	Training vehicle		Calibration vehicle	
NF	60372	-	306403	-
MF	894	-	3383	-
MD	0	0 %	15	0.4414 %
FA	23	0.0381 %	1602	0.5201 %

couple of crank positions associated with cylinder one are shown in Figure 7.5 below.

Next, the errors are estimated for the full calibration set. This is done by optimising the flywheel tooth angle errors such that the quadratic error between compensated torque and torque map is minimized. The errors for the different points of estimation are listed in Table 7.15 below.

Table 7.15: *Errors from off-line optimisation.*

Error #	Treatment		
	Normalised	**Mean removed**	**Raw estimate**
1	0.0123	0.0118	0.0120
2	-0.0014	-0.0017	-0.0014
3	0.0035	0.0027	0.0028
4	-0.0178	-0.0175	-0.0175
5	0.0277	0.0296	0.0296
6	0.0034	0.0042	0.0042
7	-0.0102	-0.0107	-0.0108
8	-0.0157	-0.0176	-0.0178
9	-0.0158	-0.0174	-0.0177
10	-0.0009	0.0000	-0.0002
11	-0.0295	-0.0287	-0.0287
12	0.0443	0.0453	0.0455

For each point of estimation a classifier was trained using the four speed and three load midpoints as described in Section 7.1, and the performance tested using the corresponding errors. In Table 7.16 below the performance is listed.

Table 7.16: *Performance using compensation on same point of error estimation and classification.*

	Normalised		Mean removed		Raw estimate	
NF	307863	-	307782	-	307672	-
MF	3385	-	3387	-	3382	-
MD	13	0.3826 %	11	0.3237 %	16	0.4709 %
FA	142	0.0461 %	223	0.0724 %	333	0.1081 %

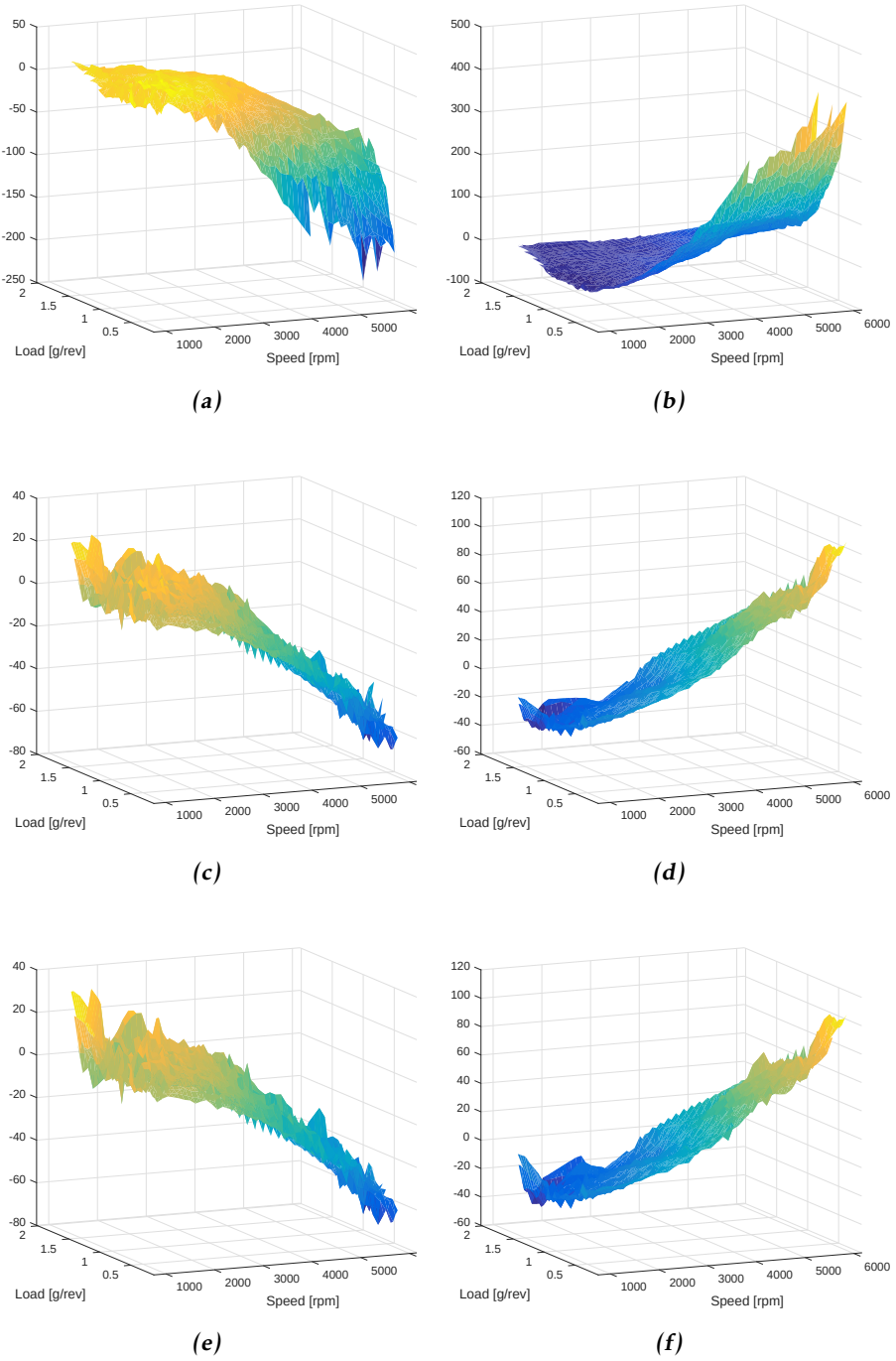


Figure 7.5: *Torque maps for the different points of estimation. Normalised torque (a) and (b), mean removed (c) and (d) and raw estimate (e) and (f). (a), (c) and (e) are from crank position 1, and (b), (d) and (f) from crank position 5.*

From Table 7.16 the best classifier looks to be the one using the normalised torque estimate. As this classifier is consistent with the one arrived at in [10, 11], this is the type of classifier that will be used in the next evaluation. Using this classifier with the different points of error estimation yield the performance listed in Table 7.17.

Table 7.17: *Performance for best classifier for different points of error estimation.*

	Normalised		**Mean removed**		**Raw estimate**	
NF	307863	-	307867	-	307865	-
MF	3385	-	3385	-	3385	-
MD	13	0.3826 %	13	0.3826 %	13	0.3826 %
FA	142	0.0461 %	138	0.0448 %	140	0.0455 %

7.5 Online

7.5.1 Classifier

The performance when using the predict function, compared to using pre-computed weights that takes the necessary normalisation in to account, is found in Table 7.18.

Table 7.18: *Time for performing detection with predict-function or pre-calculated weights.*

Method	Predict	Weights
Time	14.320 s	3.692 s

7.5.2 Extended Kalman filter

From Section 7.4 it follows that estimation of the flywheel tooth angle errors can be performed already at the raw torque estimate without loss of performance. This leads to the subsequent question of doing this efficiently on-line. The next part of the investigation is to use the EKF and CG-EKF to estimate the errors and classify the combustions on a per-cycle basis, simulating how the treatment would be performed on-line.

Initially, the full ranges of load and speed will be evaluated. As no information is available about the errors, the filter is initialised with all errors assumed to be zero. The results of running the filter from that initialisation are given in Table 7.19. The behaviour of the estimated errors from the EKF is plotted in Figure 7.6, with (a) being the initial behaviour, (b) an unexpected jump in all errors, and (c) after convergence with the optimised errors as dashed lines.

The estimated errors are quite different for the two filters after an initial pass. Due to the nature of the CG-EKF it cannot be expected to perform as well as the

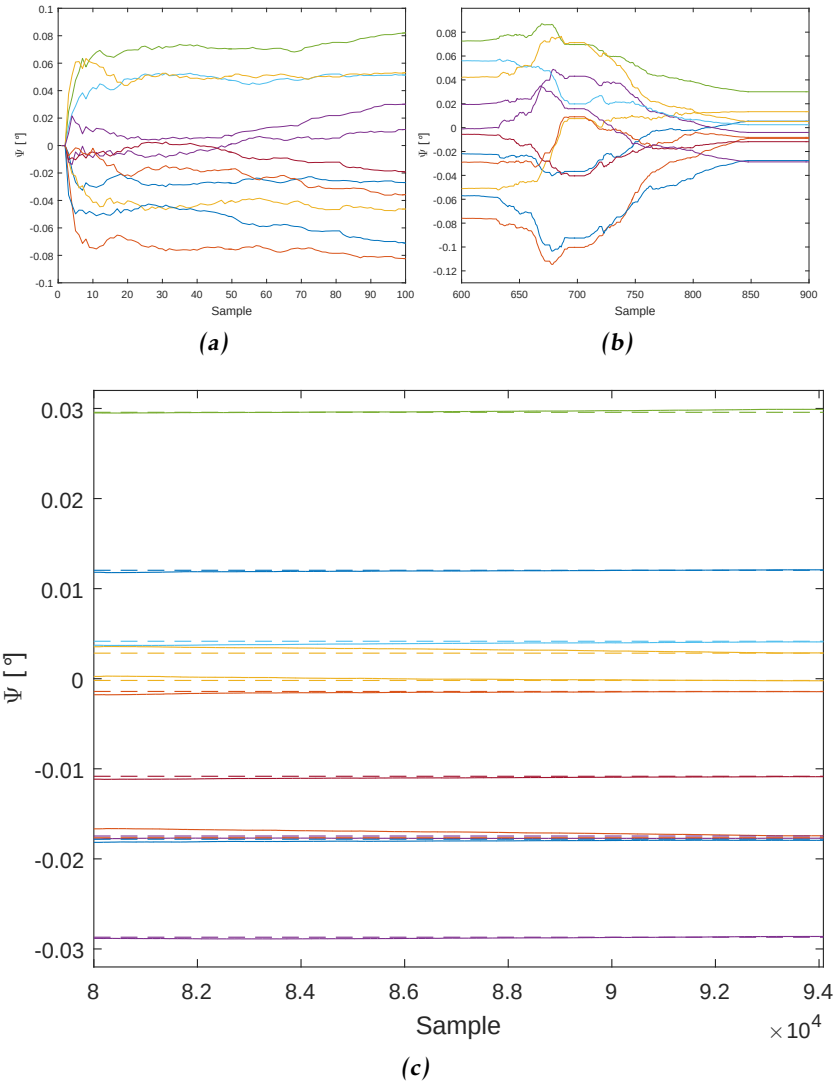


Figure 7.6: *Behaviour of* Ψ *for the EKF initialised from zero.*

Table 7.19: *Results for EKF initialised with all errors zero.*

	EKF		CG-EKF	
NF	307869	-	307866	-
MF	3384	-	3375	-
MD	14	0.4120 %	23	0.6769 %
FA	136	0.0442 %	139	0.0451 %

EKF over short amounts of time. Doing another pass with the CG-EKF, and using the last state estimate from the previous run as the starting state, improves the results. Detection performance is now on par with the EKF, having fewer FAs but more MDs. The results are listed in Table 7.20 together with the results for the EKF on a second run.

Table 7.20: *Results for EKF initialised with errors from the previous run.*

	EKF		CG-EKF	
NF	307870	-	307876	-
MF	3385	-	3382	-
MD	13	0.3826 %	16	0.4709 %
FA	135	0.0438 %	129	0.0419 %

Thus far the CG-EKF has been run with one gain for each full run. In [11] it is proposed that one way of making the CG-EKF more competitive with the EKF is to use different gains initially and after the state estimates have converged. Performance when using this approach is listed in Table 7.21.

Table 7.21: *Results for the CG-EKF using adaptive gain.*

	Adaptive gain	
NF	307872	-
MF	3374	-
MD	24	0.7063 %
FA	133	0.0432 %

As mentioned above the estimated errors are from the full ranges in load and speed. Reducing the computational load by only updating the EKF during certain operational conditions has been evaluated using loads $m_a \in [0.1, 0.7]$ and speeds $\omega \in [1500, 2500]$, the results of which can be seen in Table 7.22. The reduction in evaluated range did not remove the initial behaviour with the subsequent jump in estimates.

To limit the computational demands when using the filters on-line, another idea is to, after an initial period of the filter converging, only update for certain samples, reducing the number of updates of the filter. This approach was evaluated and the results are given in Table 7.23. The reduction was down to updating ever 200^{th} cycle.

Table 7.22: *Results for EKFs using data from a limited range.*

	EKF		CG-EKF	
NF	307845	-	307848	-
MF	3385	-	3381	-
MD	13	0.3826 %	17	0.5003 %
FA	160	0.0519%	157	0.0510 %

Table 7.23: *Results for EKFs updating less frequently.*

	EKF		CG-EKF	
NF	307862	-	307828	-
MF	3381	-	3380	-
MD	17	0.5003 %	18	0.5297 %
FA	143	0.0464 %	177	0.0575 %

8 Discussion

8.1 Classifier

The misfire detection algorithm presented in [10] performed similarly for the four-cylinder engine compared to the six-cylinder one used in the initial work. The performance using only five ranges in speed for the four-cylinder engine was slightly worse than the nine ranges used for the six-cylinder engine in [10], which indicates that the algorithm is also valid for engines other than the six-cylinder one. The four-cylinder engine has the advantage that since the combustions have greater angular separation, there is more information available to perform the detection on for each combustion. It is possible that this increase in information leads to a decreased difference in performance between the engines, even with the difference in the number of ranges.

From Figure 7.1 it is clear that there are substantial variations in the optimal weights of the classifier depending on load, but also that most of the variations are found at low speeds. Dividing the data also in load gives a better performing classifier, which should reasonably follow since the tighter grid restricts variations in load and speed, and in turn possibly the behaviour of the torques. If the two highest speed ranges are merged into one the performance drops slightly, as seen in Table 7.1, but when combined with the split in load the performance was the same as for the five-speed three-load classifier. Although for the latter, it is important to note that there was no misfire data for the 5000 rpm speed and 0.1 g/rev load range, leading to data on it not being classified. When extending the classifiers for the 3700 rpm range to cover for the lack, they both performed the same. From this, there are a few things to note. As the performance became on par after this adjustment, it implies that the performance for the other speeds could be the same. It also implies, as the classifier used was not trained on data from the classified range, but performed the same as the smaller classifier, that

classification at higher speeds is easier than at low speeds, substantiating the observations previously made from Figure 7.1. Together these two notes implies that the classifiers also performs identically for the 3700 rpm range, despite differences in their data. In addition, as the three lowest speed ranges are identical, and thus should perform identically, the two classifiers perform identically here. It is possible that this would change with more data at high speed and low load, in particular misfires.

The greater variations of the weights by load for lower speeds are potentially due to the fact that more behaviours of a vehicle are possible there than at higher speeds, where the vehicle is more likely to only be operating during acceleration.

Something not listed as a result in Section 7.1, but rather pertaining to all evaluations throughout Chapter 7, is how the size of the dataset affects the detection statistics. Based on how the rates for MD and FA are defined, both are their own problem. For all datasets the resolution of the FA rate is high enough since most of the data is without misfires. The problem arises when the number of misfires is low. This is exemplified by the multiple misfires validation data set. It contains 305 injected misfires. This means that one missed detection gives a MD rate of 0.329 %. When put in contrast with what is considered a good rate, 0.1 %, the lowest possible rate of MD, bar no missed detections, is more than three times the limit. This could call the reliability of the performance in to question as the MD may not be due to the chosen structure for the classifier but possibly the data not being collected in a way that gives enough information for certain ranges. In terms of desirable resolution, with the limit for good performance at 0.1 %, it would mean at least 1000 injected misfires are needed to give a resolution that is high enough to not have greater steps than the limit. It could even be argued that the number of injected misfires should be even higher, as to get even higher resolution and thus possibly increase the accuracy of the evaluated performance. If the approach used here with training and validation data, then the amount of data needs to be scaled to allow for that split as well.

8.2 Gaussian processes

Using a Gaussian process to model the weights of the classifier shows performance that is worse than for the basic classifier. The main difference between the two is an increase in the false alarm rate. The missed detection rate also increases, but not to the same extent. The increase is possibly due to the fact that the GP allows for noise in the model, which in turn means that the weights can be smoothed out by not perfectly reproducing each weight used for training.

Using interpolated weights is clearly a worse alternative than the basic classifier, both when using the basic classifier to train the GP, as well as when using a higher density partitioning in speed. For the interpolated weights the performance actually becomes worse when using the higher density partitioning, which is not the case for when using the range centre speeds to predict the weights. Then the higher density partitioning actually improves the FA-rate. The denser speed partitioning should intuitively result in higher performance since the nar-

rower ranges give data more similar in load and speed, potentially resulting in data that are more homogeneous. This in turn can result in a classifier that does not have to separate as complicated sets of data, giving better performance. Even with this improvement, the GP is not performing as well as the basic classifier.

Introducing bootstrap into the classifier gives a shift in performance for the classifier using the high density partitioning. The MD rate lowers slightly, while the FA rate displays a higher increase, corresponding to an increase in the total number of misclassifications. When the bootstrap samples are used as the base for a GP it actually gives an increase in performance, but still falls short of the basic classifier with partitioning in load. This increase is potentially achieved by diminishing the effects of outliers that otherwise overly influences the weights.

When partitioning the GP also by load, but still only using speed as its predictor, the performance increased slightly, but can still not reach the same performance as the bootstrap GP or the basic classifier when it is partitioned in load.

Overall, the use of GPs to model the weights has not shown any improvement in performance that would motivate to use it for practical applications. It is possible to achieve performance that is good in absolute terms, but not when compared to simple improvements to the basic classifier. As the GPs are also a more complex solution, it would need to perform better to even be considered in any on-line application.

As was mentioned about the GP in Section 3.3, the weights are inherently dependent from the training of the classifier, but this dependence is lost when the GP is estimated. This may account for some of the loss in performance when using the GPs. The use of bootstrap can be seen as a potential way to improve the likelihood of maintaining the weights closer to their actual values by reducing the influence of outliers, which could explain the increase in performance. The fact that a GP based on bootstrapped weights gives better performance than using just the bootstrap does however contradict this as the expected result would then rather be to see a slight decrease in performance when using a GP based on bootstrap.

Another thing to note is that the GP can fall short in the modelling of the weights. Looking at Figure 7.3 (c) it is obvious that the GP is not performing nearly as well as any of its neighbours. This could be down to the available parameters in the basis and covariance functions of the GP not being able to capture the behaviour of the weights, and treating it as a substantial amount of noise. This is also present in (d) in the same figure, but to a lesser extent. What this could indicate is that the weights do not have a Gaussian behaviour for these particular crank positions.

8.3 Multiple misfires

As previously mentioned in Section 1.2.2 the algorithm presented in [10] performs well on single misfires, but the performance degrades when there are multiple misfires present. When looking for a method to improve the performance by introducing new alternative treatments, the problem remains a hard one to

solve by only changing the treatment of the torque estimates.

Looking first at how the alternatives perform on multiple misfires, all but the median gives a similar result, even if the variations in number of MDs correspond to a substantial relative difference.

The median is the method with the worst performance, even no treatment of the estimates performs better. As the reasoning behind using the median was that it is a more robust alternative to the mean, it can be argued that it does not hold true for the problem at hand, at least not well enough to be used for the treatment of the signal. It does fulfil the requirement of reacting less to misfires than the mean, but instead appears to have a delayed and prolonged reaction, which could possibly be why its performance is not as good as the other methods. For data with multiple misfires, the median clearly distinguish itself from the mean and MA, both of which are close to and varying around zero, by being further removed from zero, seemingly moving the torque estimate away from varying around zero. This would indicate that there are more negative torque estimates, but with smaller magnitudes, which in turn translates to the median being lower than the mean, and not a sufficiently good approximation.

Looking at the mean and MA, the first reacts the most to misfires, while the latter has the least reaction. As the MA was proposed as a way to mitigate the reaction to misfires, it seems to have worked as intended as far as the actual values of the MA. If the actual performance is considered it does to some degree substantiate that observation. The MA has a lower number of total misclassifications, but as the improvement is in number of FAs, and at the cost of MDs, it comes down to how the two are prioritised. In relative terms the change in performance from the mean to the MA is a more substantial rise in MD than fall in FA for multiple misfires.

Even if the primary reason for introducing alternative ways of performing the treatment of the torque estimates was to increase performance for multiple misfires, the performance for single misfires cannot be ignored. Here, just like for multiple misfires, the median performs worse than the other methods, including no treatment, practically ruling it out as a solution of interest. When comparing the performance of the mean and MA to that of not treating the estimate at all, they both show substantially better performance, and for both MD and FA. This should not be surprising for the mean as its introduction was to remove the effect of torques not due to combustions, thus making the data more homogeneous as the torque developed at similar speeds and loads should be close, regardless of other torques. The MA should then, as a related solution with the possibility of tuning, reasonably have similar performance, which is also the case.

Evaluating the FIR and IIR implementation of the MA yielded the best performance when using the IIR implementation with $\alpha = 0.9$. Best is here a balance between performance for single and multiple misfires. There are other settings that perform better for multiple misfires, but at the cost of single misfire performance.

The FIR and IIR share a similar general behaviour, displaying roughly the same one with changes in their parameters. The FIRs performance for single misfires shows that a low number of cycles performs the best. The increasing number

of cycles means that each cycle is given less and less influence over the MA, and it would seem that the most recent cycle contains the most information about the current behaviour of the torque, which is only reasonable. When looking at the behaviour for multiple misfires it is initially consistent with that for single misfires, but when increasing the number of cycles towards a high number, the performance improves again. This can be interpreted as the mean over a long time has more explanatory power than over an intermediate number of cycles.

As the IIR displays two somewhat different behaviours, they have slightly different implications. The single misfire performance indicates that the more significant the current cycles is allowed to be, and the quicker old cycles lose their influence, the better the performance possible to achieve. When there are multiple misfires present, the performance indicates that either giving the current cycle low importance, and allowing the influence of old cycles to remain higher for longer, or mainly looking at the current cycle with the old ones contributing little, gives the highest performance.

The observations from both FIR and IIR indicate the same behaviour for the performance. It would seem that for single misfires, most of the pertinent information is available in the current cycle, and no increase in the number of cycles will improve the performance. Unfortunately, this does not hold true for multiple misfires as either a low or a high number of cycles yields the best performance.

Tuning of the median is done by selecting the number of cycles to include in it. From this, the selection of the best one was fairly trivial as three of the four rates shared the same behaviour. This was improving performance with a rising number of cycles, making the choice 100 cycles. Using 100 cycles to compute the median means that the introduction of one cycle with multiple misfires has the smallest possible effect on the median of the evaluated settings. Despite tuning of the median it would appear that it is not a good enough approximation of the mean.

8.4 Flywheel tooth angle error compensation

The torque maps in Figure 7.5 are fairly similar. The biggest visual difference comes from the normalisation with the load, and since the difference is observed in the load dimension it is to be expected that the behaviour becomes more exponential, rather than linear, as is the case for the two other treatments. What difference there is in behaviour in the speed dimension is mostly an effect of the different scales of the axes as the magnitudes remain similar. One last thing of note regarding the effects of the load normalisation is that it seemingly smooths out the map, but that may only be an illusion created by the greater magnitudes of much of the maps after the normalisation.

The torque maps differ only slightly between the raw estimate and after removing the mean. Removing the mean should mainly constitute a displacement in magnitude of the torque map, which is also the principal behaviour that can be observed in Figure 7.5 (c)-(f).

Combining the observations above means that a reasonable assumption is that

the error estimation is similar for the raw torque signal and after removing the mean. Normalising with the load seems to reduce the variations in the map, which, together with the more exponential behaviour, could be the reason why it results in different estimations of the flywheel tooth angle errors.

The flywheel tooth angle errors in Table 7.15 indicate that there is only a slight difference between the raw estimate and after removing the mean. Some of the errors do differ slightly, but they are only relative errors, and as they have very similar differences between errors, it is not a concern. As the twelfth error is derived from all the others, and only differs 0.0002, it can be seen as further verification that the two points of estimation should give comparable results when classifying.

Estimating the errors from the normalised torque estimate gives a somewhat different result, but with a similar behaviour of the differences between the errors, although not as close as for the two other points of estimation. Here the twelfth error is within 0.0012 of the other two estimates. Even though it is substantially greater, it is still an order of magnitude smaller than most of the estimated errors.

When the errors are used to classify data from the calibration vehicle, the performance for the different points of estimation are very similar. Since the difference in performance is only a few observation it is possible that it is down to the split of data into training and validation skewing it to one estimates advantage, rather than an actual difference in performance between the points of estimation. As the detection performance increases when the treatment progresses further, it would stand to reason that the performance when using the errors should also improve further along the treatment, as the data should become more homogeneous. The fact that the performance first drops and then rises again, while all being very similar, indicates that the points all give good estimations of the errors, and that any difference could be due to how the data used for training and validation is split. If the difference in performance was due to the point of estimation, then looking at the errors, the expectation would be a more substantial difference between the normalised torque and the other two. Now they are all evenly spaced, and the normalised torque does not stand out compared to the other ones. Its performance is the worst, and it not being in between the other two's performance is consistent with the difference in errors. All this together strengthens the argument that the split of the data into training and validation could be the reason for the difference.

The MD rate is the same for all the points of estimation, and this may very well be due to the classifier not being able to perform better for the data evaluated. As the classifier was trained on another vehicle than the one classified, there is the possibility that there are differences in how they were driven to collect the data. This could result in the data being different enough for the classifier to not transfer properly between vehicles. This would mean that the restriction on performance is not the compensation, but rather the classifier. Looking at Figures 1.1 and 1.2 it is obvious that there is a substantial difference between the data sets.

If the performance when compensating for flywheel tooth angle errors is compared to that of the validation data, it is worse. The rate of FAs only increases

slightly, but the rate of MDs goes from zero up to 0.3826 %. Not only is this a worse rate, but also over what can be considered a good performance for the data used. However, as was discussed in the previous paragraph, the reason is not necessarily the compensation falling short, but the data not being suited for the specific task. The torque map is based on only a few observations, which in a larger data set may be considered outliers, and used to find errors for a substantial part of the calibration data. If the data used for the map is not representative, the risk is finding errors that are not the actual ones.

8.5 On-line

8.5.1 Classifier

Optimising the computation of the test quantity by replacing Matlab's built in function with just weights, and introducing the normalisation into the weights, reduces the time required to roughly a 13^{th}. Even though this is evaluated on a desktop PC, and thus the saving may not compare directly to an on-line scenario. The fact that computing the weights and including the normalisation in them give the same detection performance, while requiring the least storage on-line possible for any given partitioning of data, means that storing the pre-calculated weights is optimal in terms of memory use.

8.5.2 Extended Kalman filters

Using Kalman filters to estimate the flywheel tooth angle errors appears to work well. The performance when using the Extended Kalman filter is slightly better than the performance achieved using the errors from the optimisation. This may seem counter-intuitive as the EKF starts with all errors being zero and the optimisation uses the errors that minimises the deviation from the torque maps. The reason for this difference in performance is likely the fact that the EKF is implemented with continuous classification of the data, one cycle at a time, while the optimisation classifies all cycles at once. The optimisation uses the same errors to classify all cycles, while errors from the EKF can change as new observations become available. This means that the performance when using the EKF can be higher than for the optimal errors, not because it estimates the actual errors better, but because it can adapt to the perceived errors at any time.

The actual error of a flywheel can be seen as the difference relative to the one on the vehicle used for training. It is here found through optimisation against a torque map, but it could just as well be measured physically from the flywheels. The perceived error is the error that for one cycle results in the smallest difference compared to the torque map, and can thus potentially vary during operation. If the engine control can vary for the same load and speed, effectively shifting the torque behaviour in time, it can potentially be a reason behind the errors changing.

The difference in perceived error at different times can potentially be connected to the data used for the evaluation. As was mentioned in Section 8.4,

there is a significant difference in the distribution of the data. In Figure 7.6 (b) the estimated errors quickly change and most of the error estimations switches signs. This is possibly an effect of the difference in the data, and even if that is not the reason, something with the data used to estimate the errors is different from the training data as the initial behaviour of the errors is not towards their final converged behaviour, but rather quite different. The final estimated errors are however close to their corresponding results when optimising, seen in Figure 7.6 (c).

The CG-EKF was proposed as a computationally lighter alternative, and it showed performance that was almost on par with the EKF, at least when given sufficient time to converge. The slightly lower performance on the initial run can potentially be attributed to the fact that the gain is fixed, and in order to achieve a higher gain, the state estimate was assumed to have a higher noise level. This higher noise level can in turn result in some of the TQs changing sign, and thus change the classification.

Using multiple gains and switching between them to make the CG-EKF behave more like an EKF worked to some degree. It missed one more detection, while the number of FAs dropped by six, making the total number of misclassifications lower. The use of multiple gains gave little improvement for the CG-EKF if not using it, and comparing the results to the EKF, the reduced number of FAs put it under the level of the EKF, but the MDs are still more than 50 % higher when compared.

As the computational load of the EKF is rather high, reducing it by only updating the filter with measurements from a certain range in load and speed was evaluated. Using the smaller range did not give worse MD-rates for either of the filters, but the FA-rates rose for both. The odd initial behaviour of the error estimates did not disappear when the range was limited, which would indicate that it is either present in the evaluated range, or that it is some transient behaviour present for all speeds and loads.

Lastly it was evaluated if reducing the rate at which the filters update after convergence would be a viable solution, and from the performance numbers in Table 7.23 it did give worse performance for all rates but the MDs of the CG-EKF. For the CG-EKF the number of FAs rose substantially, while the overall change in performance of the EKF was much smaller. As the change evaluated reduced the update rates of the filters to $\frac{1}{200}^{th}$ of the data rate, this would mean a substantially lower computational load due to the EKF, negating part of the logic behind using the CG-EKF. During initial convergence, this approach would still mean a high computational load, but after that the computational load would be much lower. Combined with performance that was not much worse than for the full-time EKF, the reduced update frequency approach is viable, and comes down to a decision about permissible levels of MD and FA-rates compared to computational load.

9 Conclusions

9.1 Classifier

The classifier proposed in [10] can be re-confirmed to work well on the four-cylinder engine. The weights of the classifier from training the SVM varies heavily with both speed and load. Thus dividing the data not only by speed, but also by load, increases the performance of the classifier substantially.

It was also shown that merging the two highest speed ranges for low load did not impact the performance. The lack of misfire data for the highest range did influence the implementation, but further study would be required to substantiate any actual difference. What remains true regardless is that the classifier resulting from combining the two highest speed ranges is a good choice. It reduces the storage need by 20 % while the performance changes less than 5 % in a worst-case scenario.

9.2 Gaussian processes

The use of GPs as a way to improve the detection performance did not result in an actual improvement of the performance. There are ways to get the performance of the GP close to the basic classifier, but never exceeding it. The GP as used here has the drawback of not maintaining the dependence between the weights that follows from the SVM, ultimately making it an impractical solution.

The use of GP here did however restrict itself to a limited set of basis and covariance functions, and without further study it cannot be ruled out that there are customised alternatives for these that would be a possible improvement over the basic classifier.

9.3 Multiple misfires

Using alternative methods to treat the torque estimates showed that the problem is a hard one to solve, and that only altering the treatment is not enough to fully solve it. From the combined performance on single and multiple misfires, the mean is still a viable option next to the alternatives proposed. The median was not a good choice, being outperformed by all other methods evaluated, including doing nothing. The MA was the best alternative to the mean, yielding performance that was objectively better for single misfires, and depending on how the MD and FA rates are prioritised, also for multiple misfires. For the MA, the best performing implementation was the IIR with $\alpha = 0.9$.

9.4 Flywheel tooth angle error compensation

The estimation of the flywheel tooth angle errors was previously performed after all the treatment of the torque estimates had been performed. It was shown in this work that the point at which the estimation is performed only affects the performance marginally, making it possible to estimate it as early as on the raw torque estimate. This reduces the number of computations necessary to find the errors, useful for on-line implementations that are to run on limited hardware. It also decouples the estimation of the errors from the processing of the torque estimates, allowing higher flexibility when changing any one part by increased modularity.

9.5 On-line

Adapting the classifier from its model based form to the minimal number of coefficients necessary works well; it drops the computational needs of classifying to about one thirteenth of the original.

Using an EKF to estimate the flywheel errors is possible. It can perform better than finding the globally minimizing errors and then classifying all data. This highlights the need to have enough data when estimating the torque maps. Enough is subjective and is the compromise between performance and time spent collecting data. What can be said is that there needs to be data from all possible operational behaviour used to estimate the errors. This is exemplified by the difference in training and calibration data seen in Figures 1.1 and 1.2.

The results for the CG-EKF show that it is competitive when given sufficient time to converge. Using an adaptive gain slightly increased its performance, and with more data and tuning would close the gap even further.

Of the other ways proposed to lessen the computational burden while maintaining performance, the one showing most promise was reducing the rate of update of the filter. Tuning the switching less aggressively allows the filter to converged further, and would give better performance. Even if the performance numbers show it below the basic EKF implementation, it is the best option in

terms of combining the power of the EKF with a reduction in computational load.

10 Future work

The work performed in this thesis is limited in that it works of the classifier presented in [10]. Paired with the limited time and multiple research questions, there are some thing that cannot be covered, but that would be of interest for future work.

10.1 Data collection

The data collected can have an impact on the results, and for this work Figure 1.2 shows an example of data that is likely not the most appropriate, which the results for the EKF in Figure 7.6 possibly illustrates.

It could then be of interest to look into how to best collect data from vehicles for the general type of work done here. From a more practical point of view it may be beneficial to determine a driving pattern that gives good initial estimation of the flywheel errors.

10.2 Classifier

The classifier that is used was chosen under the assumption that the computations necessary to use classifiers of higher orders than linear are too demanding for use on-line. It would be interesting to look into how using a quadratic classifier affects the performance. To do this while being cognoscente of the impact on on-line applications, it would also be relevant to evaluate different subsets of the possible combinations making up the quadratic and multiplicative terms. One simple approach would be to only use the pure quadratic terms, leaving out the multiplicative terms.

In a wider sense it would also likely be beneficial to evaluate different types of

classifiers. Most other classifiers require more computations and are more complex in terms of their structure, which can be realised already for the quadratic one. It would be of interest to study the performance of different types of classifiers for the misfire detection problem, and this could in turn feed into a discussion on how to strike the necessary balance between detection and on-line performance.

10.3 Gaussian processes

From the findings about Gaussian processes it is evident that there are certain problems with achieving the necessary performance, and doing so while managing to reduce the storage need. From that, there are two possible avenues of continued investigation.

As the functionality in Matlab used for the implementation in this work is limited by the available basis functions and kernel, the first avenue is modifications to the GP. One option would be to evaluate the performance when using bespoke functions, modelling behaviours observed from the weights. If the functions can be expressed continuously it could reduce the storage need. Another possible solution is to limit the maximum distance used for the covariance, depending on its behaviour.

Even if the idea of modelling the weights with some function is valid, the use of GPs may not be the best approach. Therefore, it would be relevant to investigate if the use of other function estimations may perform better. As the storage need was mainly due to the covariance in the GPs it should result in a reduced storage need when it is removed. Judging from the general behaviour of the weights observed during the work with this thesis, it appears that this would be a feasible approach.

10.4 Multiple misfires

During the whole of this thesis, the mean that has been used can be considered a true mean in the sense that it uses the same weight for all included values. One possible extension to this would be to look in to how using different weights for different cycles, or even different weights within the same cycle impacts the detection performance. The latter would essentially determine if certain crank positions are more indicative of the general behaviour of the torque than others.

Bibliography

[1] M. Boudaghi, M. Shahbakhti, and S. A. Jazayeri. Misfire detection of spark ignition engines using a new technique based on mean output power. *Journal of Engineering for Gas Turbines & Power*, 137(9):091509–1 – 091509–9, 2015. ISSN 07424795. Cited on page 4.

[2] Domenico Cantone and Micha Hofri. Further analysis of the remedian algorithm. *Theoretical Computer Science*, 495:1–16, 2013. ISSN 0304-3975. doi: 10.1016/j.tcs.2013.05.039. Cited on page 32.

[3] Nicolo Cavina, Luca Poggio, and Giovanni Sartoni. Misfire and partial burn detection based on ion current measurement. *SAE International Journal of Engines*, (2):2451, 2011. ISSN 1946-3944. Cited on page 4.

[4] Arnulf B. A. Graf, Alexander J. Smola, and Silvio Borer. Classification in a normalized feature space using support vector machines. *IEEE Transactions on Neural Networks*, 14(3):597–605, 2003. doi: http://dx.doi.org/10.1109/TNN.2003.811708. Cited on page 29.

[5] Fredrik Gustafsson. *Statistical sensor fusion*. Studentlitteratur, 2nd edition, 2012. ISBN 9789144077321. Cited on page 33.

[6] Fredrik Gustafsson and Rickard Karlsson. Statistical results for system identification based on quantized observations. *Automatica*, 45(12):2794 – 2801, 2009. ISSN 0005-1098. doi: http://dx.doi.org/10.1016/j.automatica.2009.09.014. Cited on page 5.

[7] Fredrik Gustafsson, Mille Millnert, and Lennart Ljung. *Signal Processing*. Studentlitteratur, 1st edition, 2010. ISBN 9789144058351. Cited on pages 21 and 22.

[8] Tobias Gutjahr, Holger Kleinegraeber, Thorsten Huber, and Thomas Kruse. Advanced statistical system identification in ecu-development and optimization. Technical Report 2015-01-2796, SAE Technical Paper, 2015. Cited on page 5.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer series in statistics. Springer, 2nd edition, 2009. ISBN 9780387848570. Cited on pages 12 and 17.

[10] Daniel Jung, Lars Eriksson, Erik Frisk, and Mattias Krysander. Development of misfire detection algorithm using quantitative FDI performance analysis. *Control Engineering Practice*, 34(0):49–60, 2015. Cited on pages 1, 2, 3, 4, 5, 9, 19, 26, 37, 50, 55, 57, 63, and 67.

[11] Daniel Jung, Erik Frisk, and Mattias Krysander. A flywheel error compensation algorithm for engine misfire detection. *Control Engineering Practice*, 47:37–47, 2016. Cited on pages 2, 3, 4, 9, 25, 50, and 52.

[12] Michael Kemmler, Erik Rodner, Esther-Sabrina Wacker, and Joachim Denzler. One-class classification with gaussian processes. *Pattern Recognition*, 46:3507 – 3518, 2013. ISSN 0031-3203. Cited on page 5.

[13] Takahisa Kobayashi, Donald L Simon, and Jonathan S Litt. Application of a constant gain extended kalman filter for in-flight estimation of aircraft engine performance parameters. In *ASME turbo expo 2005: Power for land, sea and air*, pages 617–628, Reno, Nevada, 2005. Cited on page 34.

[14] Lawrence L. Lapin. *Probability and Statistics for Modern Engineering*. PWS-KENT Publishing Company, 2nd edition, 1990. ISBN 0534981658. Cited on page 20.

[15] Lennart Ljung and Torkel Glad. *Modellbygge och simulering*. Studentlitteratur, 2nd edition, 2004. ISBN 9789144024431. Cited on pages 21 and 22.

[16] J Mohammadpour, M Franchek, and K Grigoriadis. A survey on diagnostic methods for automotive engines. *International Journal of Engine Research*, 13(1):41 – 64, 2012. ISSN 14680874. Cited on page 1.

[17] Tschanen Niederkohr. Carbing out a piece of history. *Aftermarket Business*, 117:1–107, 2007. ISSN 08921121. Cited on page 1.

[18] Andrzej Puchalski. A technique for the vibration signal analysis in vehicle diagnostics. *Mechanical Systems and Signal Processing*, 56-57:173 – 180, 2015. ISSN 0888-3270. Cited on page 4.

[19] Carl Edward Rasmussen and Christopher K I Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. ISBN 026218253X. Cited on pages 4 and 15.

[20] von Mises Richard. *Mathematical theory of probability and statistics*. Academic Press, 1964. Cited on page 20.

[21] Peter J Rousseeuw and Gilbert W Bassett. The remedian: A robust averaging method for large data sets. *Journal of the American Statistical Association*, 85(409):97–104, 1990. ISSN 01621459. Cited on page 32.

[22] Andrzej Ruszczynski. *Nonlinear Optimization*. Princeton University Press, 2006. ISBN 9780691119151. Cited on page 27.

[23] Masayuki Tamura, Hitoshi Saito, Yukimaro Murata, Kunihiro Kokubu, and Satoshi Morimoto. Misfire detection on internal combustion engines using exhaust gas temperature with low sampling rate. *Applied Thermal Engineering*, 31(SET 2010 Special Issue):4125 – 4131, 2011. ISSN 1359-4311. Cited on page 4.