Skip to Main Content

Major Project Report

by George Jones

FILE	5210_GEORGE_JONES_MAJOR_PROJECT_REPORT_193356_1848512229.PDF (6.08M)		
TIME SUBMITTED	04-MAY-2020 05:23AM (UTC+0100)	WORD COUNT	31330
SUBMISSION ID	126053465	CHARACTER COUNT	181908

Supervisor: **Malcolm McDonald**
Major Project Report
M6X00417

April 2020

A COMBINED MATHEMATICAL & EXPERIMENTAL APPROACH TO VALVETRAIN DYNAMICS BASED ON AN MCR DURATEC ENGINE

George Jones

University of Wales Trinity Saint David
BEng Motorsport Engineering

MATLAB

SPORTS 2000 DuraTEC
SRCC DURATEC CHAMPIONSHIP

Abstract

Mathematical and experimental methods will be employed to determine the characteristics of a Ford Duratec HE420 valvetrain for use in an MCR Race Car. The investigation is inspired by the SRCC Sports 2000 Championship technical regulations, where limited elements of the valvetrain are permitted for development.

Experimental procedures for converting between camshaft lobe and valve lift profiles were built, using a refined matrix method and coordinate geometry (rotation of axes). It was found that curve fitting and interpolation techniques could be used to simulate the resolution size of advanced Coordinate Measuring Machines (CMM). A good improvement was made over the original 360 cam lobe measurements.

A mathematical model was then produced to approximate the cam-to-tappet friction properties, where a 20% reduction in friction was attained by reducing valve spring stiffness. The Fourier series enabled interactions between valve harmonics and spring surge frequency to be analysed. Finally, all findings from experimental procedures were validated against commercially acceptable valve lift profiles.

Declaration of Authenticity

"This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree."

Signed: George Jones Date: 30/04/2020

Acknowledgements

I would like to thank my supervisor Malcolm McDonald for providing exceptional guidance and technical support through this project.

I would also like to thank my girlfriend, Sarah Hardwick, for being so incredibly supportive throughout the last 4 years, and to Gary Jones, Teresa Jones & Louise Powell for always believing in me.

Contents

Abstract 1

Declaration of Authenticity 2

Acknowledgements 2

Nomenclature 6

1. Introduction 1

1.1 Project Aims 2

1.2 Objectives 2

1.3 Initial Project Flow Chart 1

2. Literature Review 1

2.1 Camshaft Measuring Techniques 1

2.2 Camshaft Profile Design 2

2.3 Valve Event Length (fixed valve timing) 4

2.4 Valve Lift 4

2.5 Valvetrain Forces 5

2.6 Natural Resonant Frequencies 6

2.7 Valve Spring 7

2.7.1 Defining Fourier Analysis 8

2.8 Valvetrain Friction 11

3. Physical Measurement Procedure 13

3.1 Camshaft 13

3.2 Valve Spring 15

3.3 Inlet Valvetrain 17

4. Valvetrain Kinematics 18

4.1 Valve Lift Derived from Cam Measurements 18

4.2 Lift Properties and Valve Motion 22

5. Valve Lift Matrix Study 24

6. Cam Measurement Post Processing 27

6.1 Measurement Analysis 27

6.2 Smoothing Methods 29

6.3 Curve Fitting Methods 30

6.3.1 Gaussian Curve 30

6.3.2 Improved Gaussian Curve 32

7. Valvetrain Dynamics 37

7.1 Friction Model 38

8. Commercial Camshaft Manufacturers at Autosport International 2020 42
9. Transformation of Valve Lift to Cam Lift 43
9.1 Rotation of Axes 45
10. Validation Study 49
10.1 Comparison of Commercial Valve motion 50
10.2 Validation of Valvetrain Characteristics 53
10.3 Valve Float Speed Study 58
10.4 Comparison Summary 60
11. Harmonic Analysis 61
11.1 Spring Surge Frequency 66
11.1.1 Spring Surge with Reduced Valve Spring Stiffness 67
11.2 3D Plot for Spring Surge Analysis 68
12. Discussion of Findings 69
12.1 Measuring & Post Processing of Camshaft Lift Profile 69
12.2 Conversions Between Camshaft and Valve Profiles 70
12.3 Valvetrain Friction 75
12.4 Harmonic Analysis and Valve Spring Surge 77
13. Conclusion 79
14. Recommended Further Work 80
15. References 82
16. Appendix 86
16.1 Matrix Sine Study 86
16.2 MCR Maximum Engine Speed & Working Range 88
16.3 Valve Motion 90
16.4 Fourier Analysis (Kent Cams Profile) 92
16.5 Fourier Analysis (Valve Acceleration) 97
16.6 Engine Dynamometer Preparation 98
16.7 Duratec (DHE420) Crate Engine Procedure for Competition Use 101
16.7.1 Introduction 101
16.7.2 Crate Engine Adaptation 101
16.7.3 Engine Dynamometer Application 101
16.7.4 MCR Chassis Application 101
16.7.5 Initial Test 102
16.7.6 Dynamometer Test 102
16.8 Modelled Valvetrain Components 103
16.9 Initial Flow Bench Results 106

16.10 Camshaft Contact Stress 109
16.11 Square Wave Fourier Study 112
16.12 Matlab Scripts 114
16.12.1 Constants Script 114
16.12.2 Example Curve Fitting Procedure 116
16.12.3 Example Matrix Script (3,600 x 3,600) 120
16.12.4 Example of Calculation Script Including Valve Derivatives, Friction Model, Rotation of Axes 122
16.12.5 Spring Surge Calculator with 3D Output 126
16.12.6 Fourier Analysis Script 127

Nomenclature

Acronym	description	Unit
θ	Camshaft Rotation Angle (nose to contact point)	°
φ	Camshaft Rotation Angle (from vertical)	°
L_v	Valve Lift	mm
V_v	Valve Velocity	mm/°
A_v	Valve Acceleration	mm/°/°
Vel_v	Valve Velocity	m/s
Acc_v	Valve Acceleration	m/s^2
R	Base Circle Radius	mm
RPM	Engine Speed	Rpm
F_c	Contact Force	F
MM	Moving Mass	Kg
K	Spring Constant	N/m
S_{cd}	Spring Compression Distance	m
F_f	Friction Force	N
μ	Coefficient of Friction	-
T_f	Friction Torque	Nm
h	Distance (centre of base circle to contact point)	mm
WD_f	Friction Work Done	J
P	Power	W
ω	Angular velocity	rad/s
C_t	Tappet Clearance	mm
RPM_f	Valve Float Speed (engine speed)	rpm
SL_{max}	Maximum Spring Load	N
A_{fl}	Valve Acceleration at Full Lift	mm/°/°
S_s	Valve Spring Surge Frequency	Hz
M_s	Valve Spring Mass	kg
$Order_n$	Harmonic Number (order)	-
V_c	Camshaft Velocity	rpm

1. Introduction

The Valvetrain configuration for the Ford Duratec HE420 comprises 16 valves and double overhead camshafts (DOHC), with direct acting lobes to control valve actuation. Whilst these types are usually more expensive and can carry a weight penalty, the benefits include greater valve lift (and accelerations), simplified cylinder block layouts, a higher natural frequency and lower vibration amplitudes when compared with other common valvetrain formats. Another advantage is the capacity for an increased valve size, where the DOHC Ford engine includes a 20° valve angle.

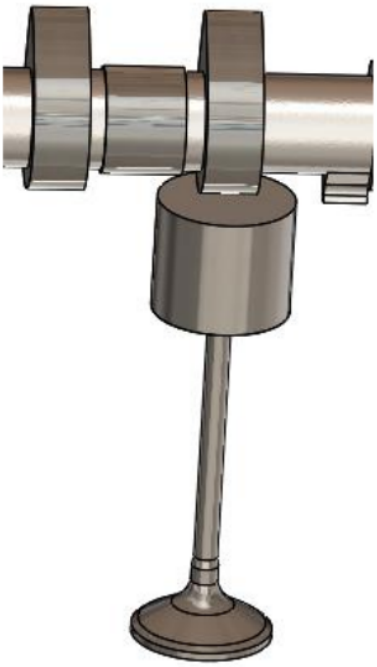


Figure 1.1 Ford Duratec HE420 valvetrain configuration.

The engine in question was taken out of an MCR Race Car before the cylinder head was removed, to gain access for physical measurements and inspection. The valvetrain (including camshafts) are believed to be original equipment (OE) specification. Whilst considerations to the exhaust side exist, the focus of this report will be on the inlet side of the valvetrain to streamline the processes involved.

The MCR Race Car complies with the SRCC Sports 2000 technical regulations (2019), which have been designed to prohibit extensive vehicle development, to reduce the ability for competitors to gain significant performance advantages. Whilst this is the case, some areas of the regulations are either open for interpretation or entirely unregulated. These policies were closely examined, leading to inspiration for this project based on the following technical requirement.

5.7.4 d) *"All valve train components, other than the **valve springs and simple shims** under valve springs, may not be modified or replaced. The replacement tappets from Ford are permitted to accommodate valve train wear but cannot be modified in any way unless to adjust lash length within the standard Ford range. **Valve springs are free**."* (Morris, 2019)

This initial motivation grew into a full mathematical valvetrain study, with additional considerations to camshaft design, including experimental refinements in Matlab.

1.1 Project Aims

- To remove the cylinder head from a Duratec HE420 and gather physical measurement data for all valvetrain components. This includes a suitable inlet cam lobe profile and comprehensive valve spring force measurements.
- To determine valve motion profiles (lift, velocity, acceleration) using the matrix method and create a refined model to increase accuracy of results.
- To predict the friction characteristics at the cam-to-tappet interface based on the physical measurements taken (and developed).
- To implement coordinate geometry for converting valve displacement profiles to cam lobe profiles.
- To validate and compare all findings with commercially accepted valve motion profiles from (PiperCams, 2020) and (KentCams, 2020). This includes conducting a valve float speed study for estimating an optimised spring stiffness, to reduce power loss due to friction.
- To conduct harmonic analysis on the inlet side of the valvetrain using the Fourier series, considering the possibility of causing resonant excitation at valve spring surge frequency.

1.2 Objectives

Matlab will be used to process and combine all measurement data and externally sourced variables into a single script. This will take the form of a reference document for subsequent calculations that can easily be modified or later changed for another engine using direct acting valvetrain.

Additional independent scripts will be written to compute each of the project aims. The results will be output numerically to the workspace or through 2D & 3D plots for further analysis. Supplementary Matlab features such as the curve fitting tool, loops and unique functions will be exploited to aid in more complex or iterative tasks.

1.3 Initial Project Flow Chart

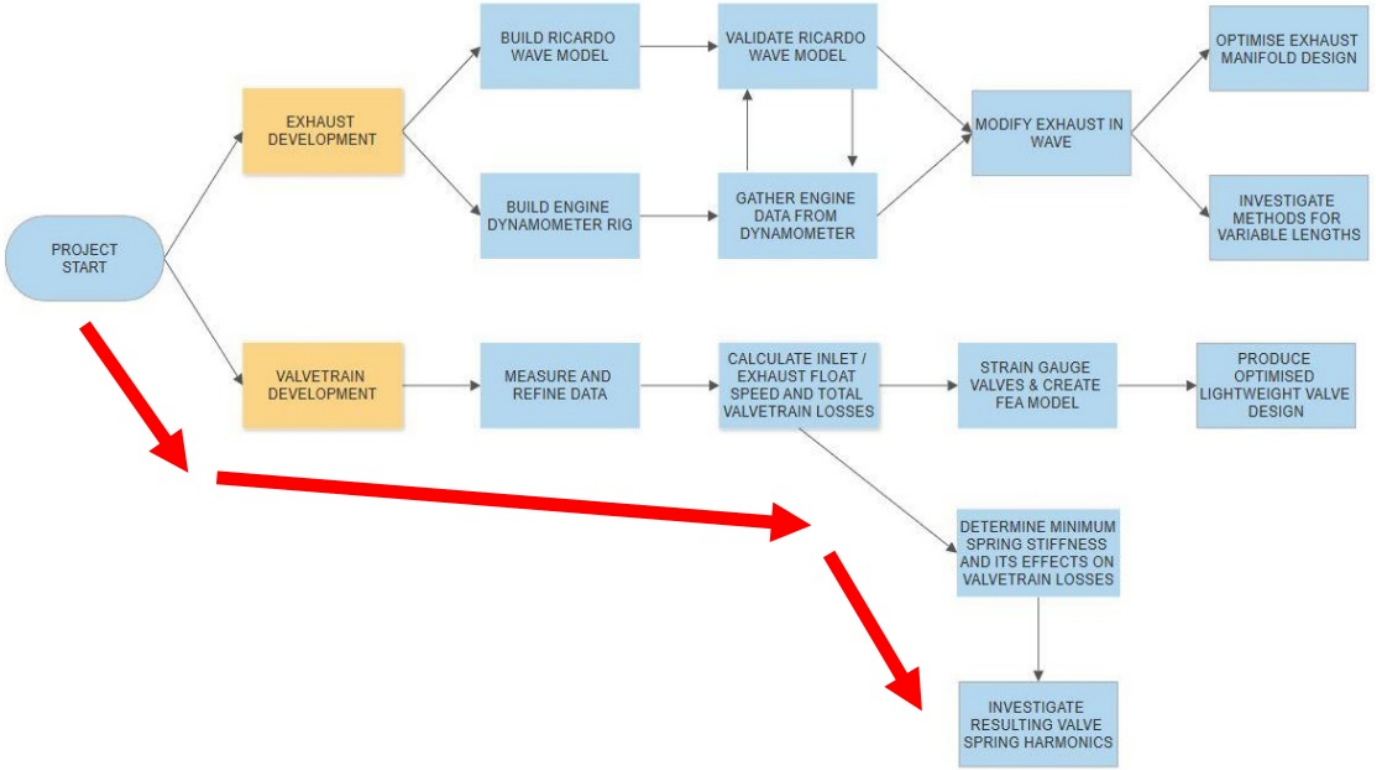


Figure 1.2 The initial project concept to explore the Duratec engine, with final outcome..

2. Literature Review

2.1 Camshaft Measuring Techniques

The physical measurement of cam lobe profiles has always been common practice within machine design, particularly for quality control purposes. To attain the greatest precision across the curved profile, a measuring procedure must track both angular displacement and linear displacement to greater accuracy than the manufacturing tolerance.

The paper by (Nourse, 1965) proposes a statistical approach to refining the accuracy of cam lobe measurements. This work considers systematic errors induced from camshaft misalignment (bend/strain, true centre of base circle, live centre of measuring instrument) and random errors such as surface imperfections, operator error (including rounding errors) and displacements within the measuring instrumentation that do not follow the identical path for every measured data point. The statistical analysis is processed electronically. This software relies on raw data sets that have been gathered with calibrated measuring instrumentation of verified tolerance.

(CGKGage, 2020) and (Adcole, 2020) produce advanced cylindrical coordinate measuring machines (CCMM) for commercial camshaft analysis. These modern solutions include post processing and evaluation software that surpasses the earlier work completed by (Nourse, 1965) and (Turkish, 1950). The Adcole 1302 records 3,600 data points per revolution to provide actionable data, with sub-micron accuracy (Adcole, 2020).



Figure 2.1 Adcole 1302 camshaft measuring machine records to 0.1˚ cam angle (3,600 measurements) and is priced at approximately $260,000.00 (Adcole, 2020).

2.2 Camshaft Profile Design

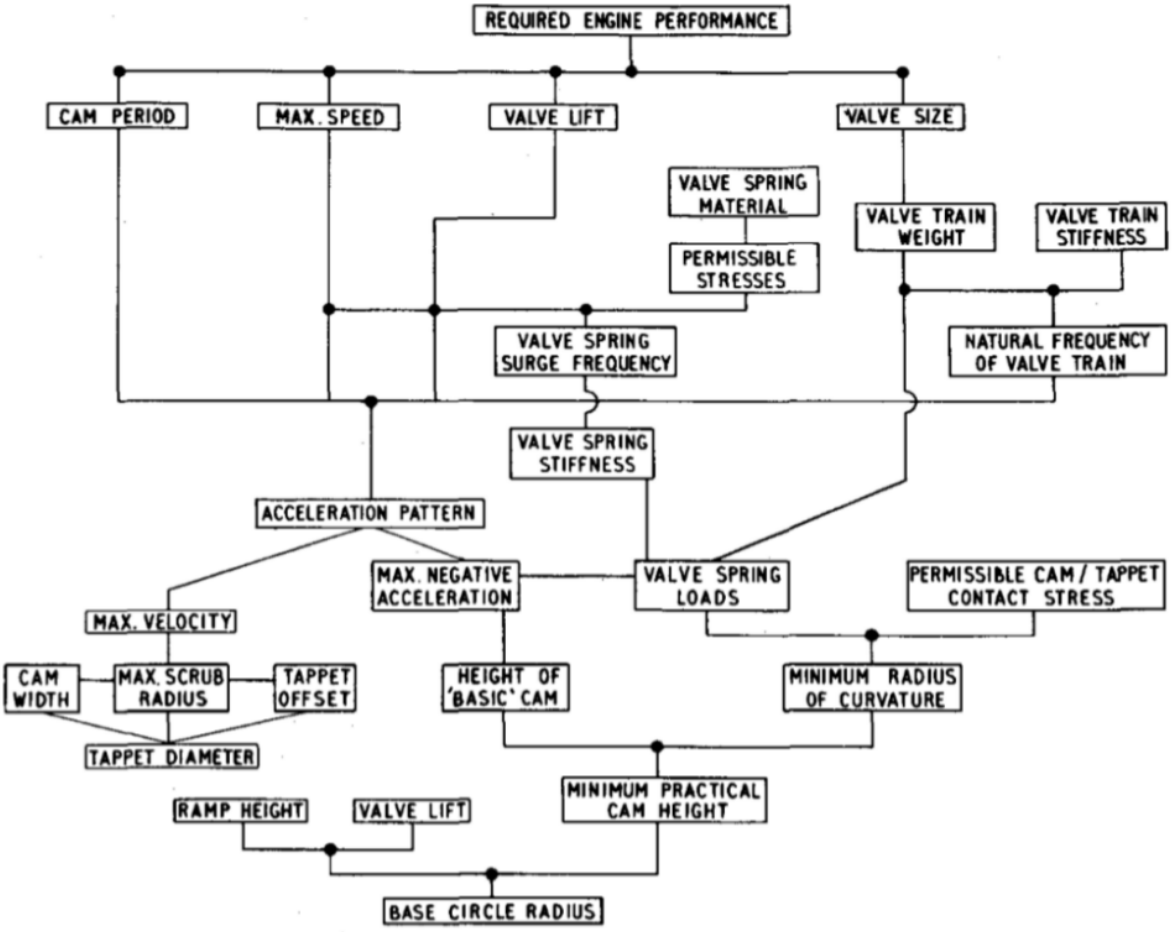


Figure 2.2 Factors to be considered in valve gear design (Beard and Hempson, 1962).

(Beard and Hempson, 1962) characterise the valvetrain design process through a series of interdependent variables. *Figure 2.2* illustrates these conflicting requirements and the many compromises that must be successfully balanced as a valvetrain design engineer. All considerations are affected by the shape of the cam lobe, as this is responsible for valve motion at any engine speed.

Historically, variants of basic curves were used to define the lobe profile. These were predominantly created around trigonometric properties, such as sinusoidal accelerations as described by (Zigo, 1967). Early methods were developed to include polynomial curves where contributions by (M Chew, 1993) and (B Fabien, 1994) involve optimal control theory. These mathematical optimisation models obeyed a set of design criterion for parameters such as varying engine speeds. This resulted in lower contact

stresses, forces and power loss over prior methods. Modern high-speed cam lobes are designed using a selection of blended spline curves (such as Bezier) and algorithm-based models to solve a comprehensive set of operating conditions. (Qiu, 2005) Gave example to this design approach by including kinematic and dynamic optimisation.

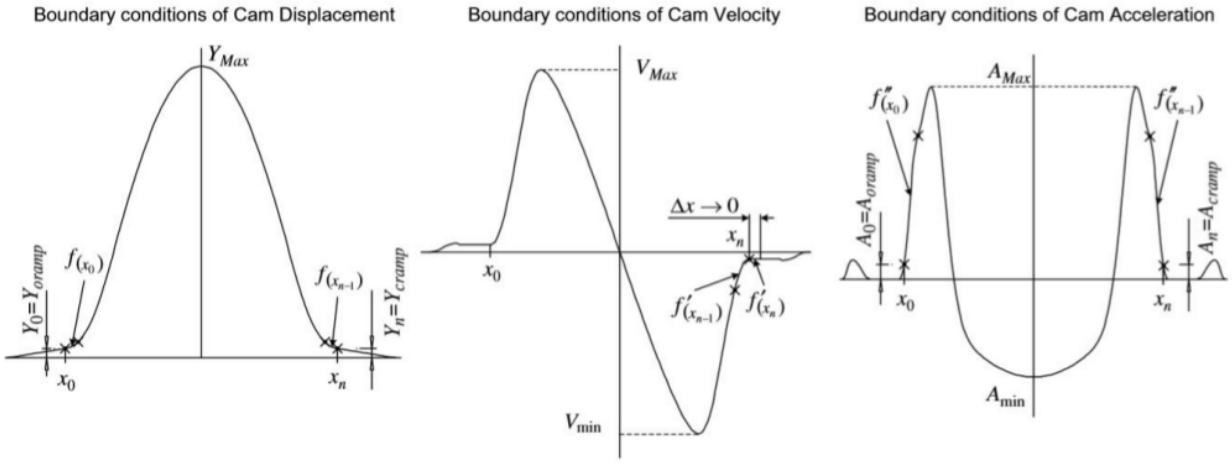


Figure 2.3 Camshaft profile design considerations through motion characteristics (Qiu, 2005).

(Nourse, Dennis and Wood, 1961) suggest the primary constraints for cam profile design include valve duration and maximum valve lift, with (Nguyen and Kim, 2007) expanding these initial boundary conditions to include motion properties up to the 2nd derivative of displacement, using algorithm based splines, as outlined in *figure 2.3*.

This investigation does not include a new camshaft lobe design for the Duratec engine, although awareness of the construction of such contours is advantageous for interpreting results relating to valve motion.

2.3 Valve Event Length (fixed valve timing)

Valve duration forms the initial constraints of cam design and is defined through the specific gas flow requirements of the engine. A study by (Johansson, 1997) gives example to the effects of changing inlet valve durations and the paper by (Tauschek, 1958) outlines fundamental valve and port design considerations. Simulation and empirical data are combined to establish a suitable solution. The valve duration will be found for the Duratec engine and used as a validation parameter for the measured camshaft profile.

2.4 Valve Lift

The effective air flow through the cylinder head is severely limited where valve lift exceeds 0.25 (lift/diameter) (Tauschek, 1958). However, increasing the valve lift past this point can improve the volume of air being flowed through opening and closing phases, as shown in *figure 2.4*.

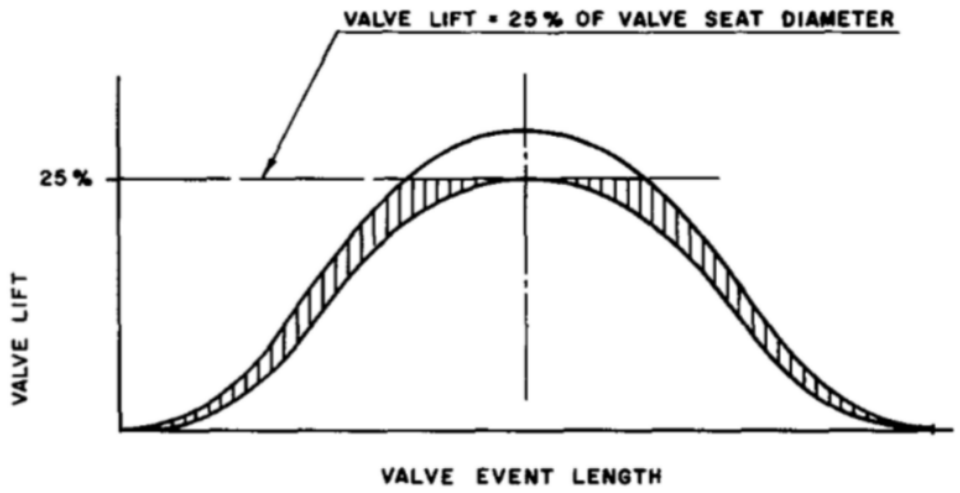


Figure 2.4 A graph to show the gain in effective valve lift area with optimised lift (Nourse, Dennis and Wood, 1961).

(Heath, 1988) demonstrates how direct acting valvetrains can benefit from lift/diameter values in excess of 0.4. With a fixed valve duration, an increase in valve displacement translates to a proportional increase in acceleration. Naturally, this conforms to Newtons 2nd law of motion, exerting a larger force to the system that must be considered. (Turkish, 1953) suggests how the smoothness of the acceleration profile can be beneficial in producing good valvetrain dynamics.

2.5 Valvetrain Forces

(Phlips, Schamel and Meyer, 1989) present a model for simulating valvetrain behaviour for direct acting bucket tappets. This paper demonstrates the effects of changing the rate of acceleration through opening and closing phases. The alternating inertia forces demand a suitable response from the valve spring, which must control contact forces acting at the valve-to-seat and cam-to-tappet interfaces as reviewed by (Clauberg and Huber, 2013). In addition, consequential events such as valve float, loft and bounce must be accounted for across the engines designed speed range. In many applications these are undesirable phenomena, as a loss in control usually translates to unpredictable behaviour that also has potential to cause component failure. However, in some restrictive instances (most notably motorsport), such events can be exploited to gain a performance advantage by extending valve opening times (Heeps, 2018).

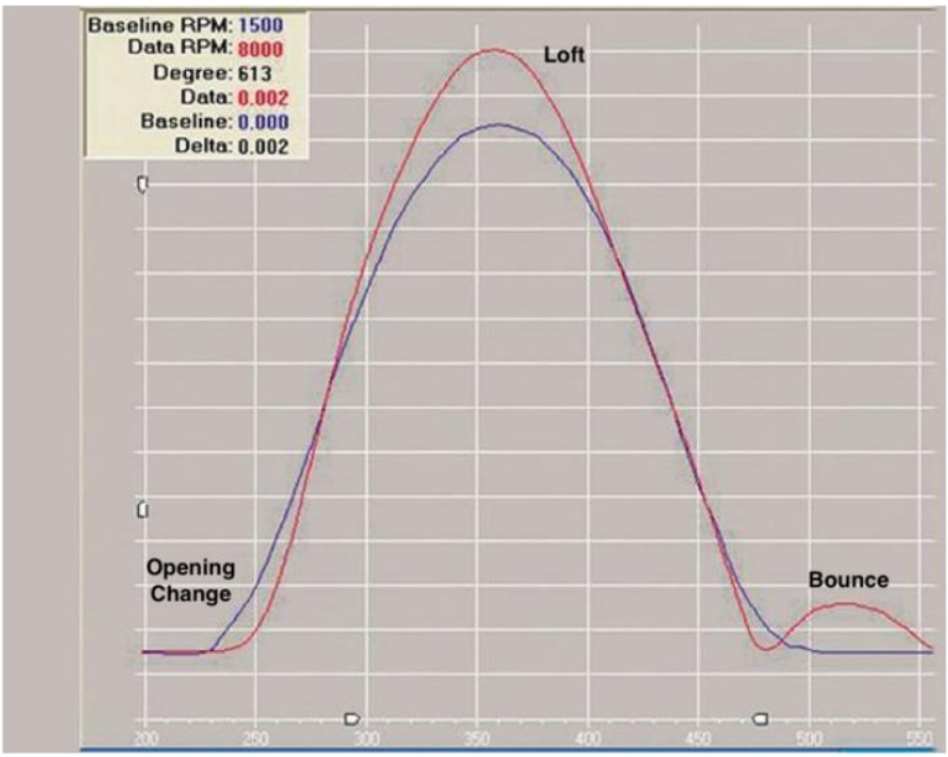


Figure 2.5 Depiction of valve loft and valve bounce in terms of valve displacement (Heeps, 2018).

2.6 Natural Resonant Frequencies

The cyclic loading associated with valve motion can incite mechanical resonance if the frequency of the applied force is proportional to the frequency of the 1st harmonic, of any component in the system. It is well documented that excitation of a components resonant frequency can cause catastrophic failure, with examples given by (Britannica, 2019). This phenomenon is considered in camshaft (and extended valvetrain) design by means of modal analysis. Such analysis can be accomplished through mathematical, computational, and experimental models (Lengvarský and Bocko, 2013). This investigation will focus on the mathematical approach.

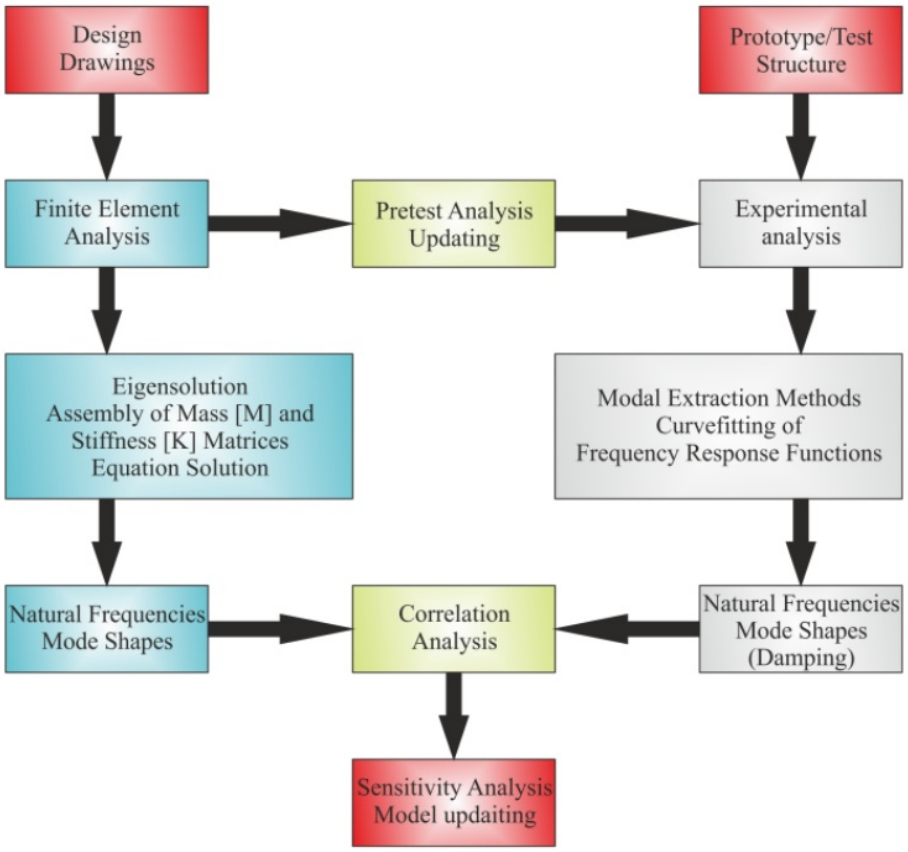


Figure 2.6 A typical flowchart for commercial modal analysis combining theoretical Finite Element Analysis (FEA) and Experimental Modal Analysis (EMA) (Lengvarský and Bocko, 2013).

2.7 Valve Spring

The valve spring design must balance two primary factors.

1. Spring stiffness. Influenced by valve spring loads and valve accelerations (inertia) to provide restoring forces to valves.
2. Valve spring surge frequency. Characterised from physical valve properties and influenced by harmonic excitation.

A linear valve spring vibrates internally when excited by the valvetrain. This can be expressed through the wave propagation equation, with the first mode being an in-phase motion and maximum amplitude occurring in the middle of the spring. The end conditions of the spring are fixed by the cylinder head and valve retaining cap which influences the natural frequency of the system (Liu and Kim, 2009).

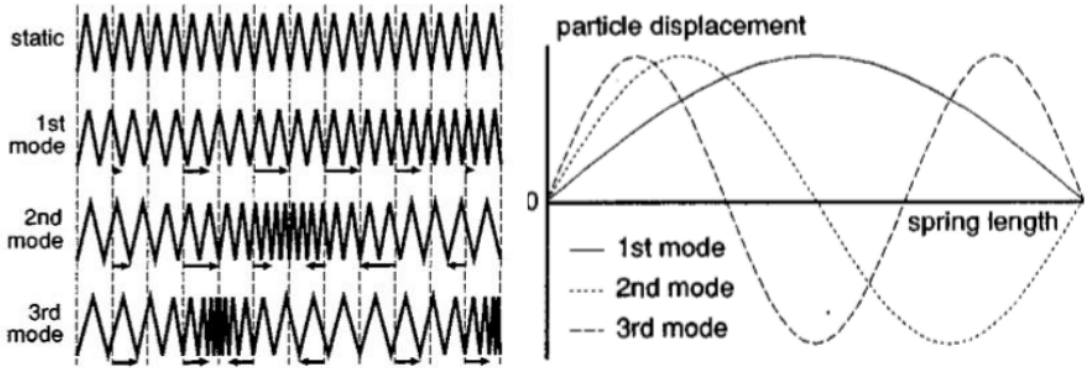


Figure 2.7 Sketch of the first three mode shapes in the valve spring with corresponding particle displacement against spring length (Schamel et al., 1993).

(Dudley, 1948) began development of an early valvetrain model in 1948 that only considered a single mass oscillator. The evolution of valvetrain vibration models has led to important contributions such as those by (Schamel *et al.*, 1993), where Fourier methods were used to characterise spring surge frequencies in high resolution. A total of 150 masses were considered for the proposed simulation and the results were compared to measured frequency characteristics from a purposely developed force transducer, as illustrated in *figure 2.8*. It was found that using 50 cam harmonics provided enough

accuracy and the (valve lift depending) frequencies of the spring were approximated by a maximum of 12-15 orders of the Fourier series.

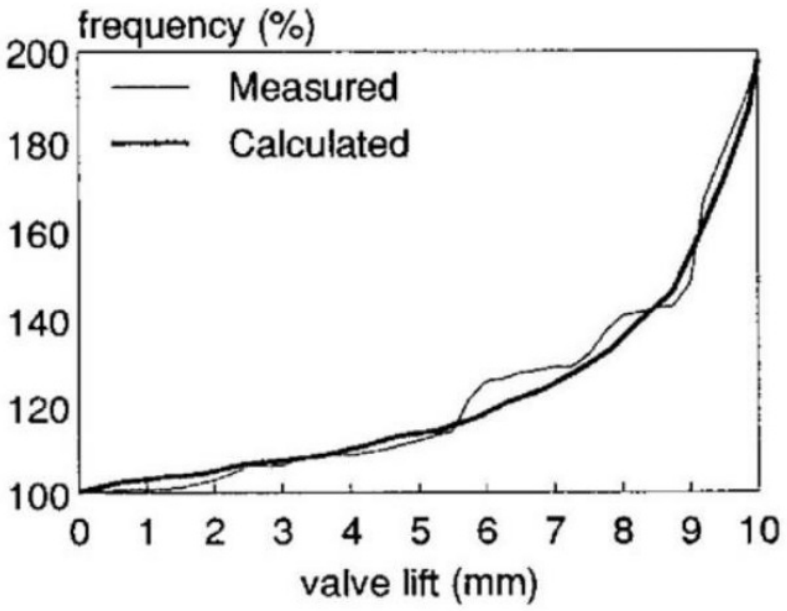


Figure 2.8 Comparison of measured and calculated spring frequency characteristics (Schamel et al., 1993).

2.7.1 Defining Fourier Analysis

Fourier Analysis is a field of mathematics in which general functions can be approximated using the sums of sinusoids (produced by simple trigonometric functions). Joseph Fourier established this method during the 19th century, enabling the premise of harmonic analysis. Peter Dirichlet made further contributions to this theorem by forming restrictive boundary conditions. Any function that satisfies these conditions can be represented by its Fourier series.

- $f(x)$ must be *absolutely integrable* over a period.
- $f(x)$ Must be *bounded.*
- $f(x)$ must have a *finite number of extrema* in any given interval.
- $f(x)$ must have a *finite number of discontinuities* in any given interval.

A function is uniquely determined by its' Fourier coefficients. These change with n^{th} order of the series as demonstrated by the relationships,

$$a_0 = \frac{1}{T}\int_0^T f(x)\ \ dx$$

$$a_n = \frac{2}{T}\int_0^T f(x)\cos(nx)\ \ dx$$

$$b_n = \frac{2}{T}\int_0^T f(x)\sin(nx)\ \ dx$$

$$Where\ a\ and\ b\ are\ Fourier\ coefficients\ of\ order\ n$$

The weighted coefficients of each order can be expressed as a series of summated sine or cosine waves. The facilitation of more complicated functions is achieved through the combination of the sine Fourier series and cosine Fourier series.

$$f(x) = a_0 + \sum_{n=1}^{\infty} a_n \cos(x) + b_n \sin(x)$$

The final Fourier series equations are used to decompose any signal with uniform convergence as order $n \to \infty$, if the original function and its derivative are continuous and piecewise smooth. Limits for the periodicity of the function are also applied to conform to Dirichlet's conditions.

For trigonometric Fourier Series (TFS),

$$\int_0^T f(x) \quad dx = \int_0^T \left(a_0 + \sum_{n=1}^{\infty} a_n \cos(x) + b_n \sin(x) \right) dx$$

Equation 1 Trigonometric Fourier series.

For a complex function, the Exponential Fourier Series (EFS) coefficients are found by,

$$F_n = \frac{1}{T} \int_0^T f(x) e^{-jn\omega_0 x} \; dx$$

With EFS,

$$F(x) = \sum_{N=-\infty}^{\infty} f_n e^{jn\omega_0 x} \; dx$$

The trigonometric Fourier series (*equation 1*) will be used in this investigation to carry out harmonic analysis on the Duratec valvetrain.

2.8 Valvetrain Friction

Valvetrain friction is a significant source of mechanical friction loss in an engine and can be considered to act through three primary modes in direct acting systems.

1. Cam-to-tappet interface
2. Tappet-to-bore interface
3. Camshaft bearings

All three interactions rely on an element of lubrication delivered by the same oil properties that are also responsible for the lubrication of the crankshaft assembly and other systems. For this reason, significant technological advances have been made in both surface finishing and lubricant quality to reduce power loss and improve engine efficiency. (A Gangopadhyay, 2004) observed how engine oil formulation contributes significantly to friction reductions.

Experimental analysis by (Yasuda *et al.*, 2003) proves how a Diamond-Like Carbon Coating (DLC) valve tappet can reduce friction loss by as much as 45% when compared to conventional steel. The experiments on different DLC types were carried out by use of a pin-on-disk tester with motored valvetrain friction apparatus. It was found that arc-ion DLC coatings were the best performers under lubricated conditions. The DLC friction coefficient was 70% lower than uncoated steel. (Amp Gangopadhyay, D Watt, 2007) conducted a similar study with Silicon Nitride tappet inserts. (Al-jeboori *et al.*, 1990) investigated the effects of changing tappet clearance, where a smaller clearance resulted in higher friction and wear.

A study by (J. M. Pieprzak, P. A. Willermet, 1989) demonstrates how tappet rotation can reduce instantaneously measured friction by up to 18% when compared to non-rotating tappets and (Pieprzak, Willerrnet and Dailey, 2020) concluded tappet/bore friction to peak at ±40˚ (mid stroke) where the contact patch offset was the greatest. This contrasts with piston/bore behaviour, where highest friction is seen at top and bottom dead centre.

A comprehensive friction model was proposed by (Okarmus and Keribar, 2015) and considers most of the aforementioned factors. This was correlated against a “V” and an “inline” engine, both being used on a motorised cylinder head test rig with considerations to various oil temperatures, component surface finishes and engine lubricants. The investigation gave encouraging results and it was found that most of the friction was formed from cam-to-tappet contact, with camshaft bearings making an important contribution at low engine speeds (typically below 1500rpm). The boundary friction coefficient for the cam-to-tappet interface was discovered experimentally using a dedicated machine.

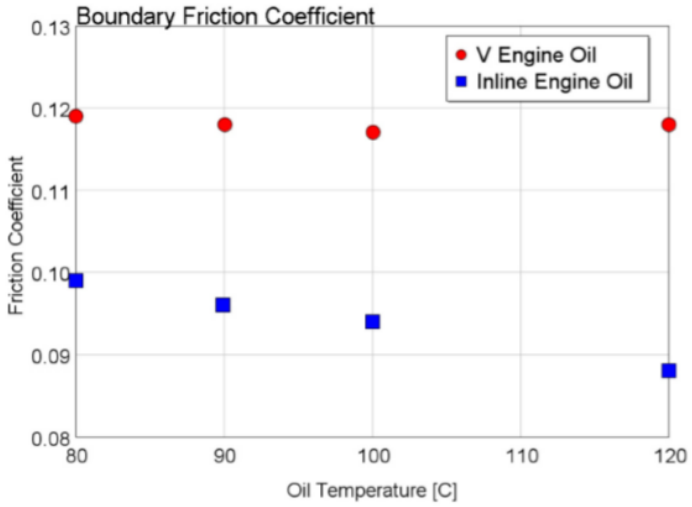


Figure 2.9 Cam-to-tappet boundary friction coefficient for "V" and "Inline" configurations, using two different variants of SAE 5W-20 oil (Okarmus and Keribar, 2015).

(Calabretta and Cacciatore, 2010) proves cam-to-tappet friction to be dynamic, depending on a range of boundary conditions. Asperity contact through boundary lubrication and the effects of oil shear contribute most to friction in this application. *Figure 2.10* provides evidence of asperity contact friction coefficients changing through engine speeds, as well as the effects of different contact surface finishes in a high-performance engine.

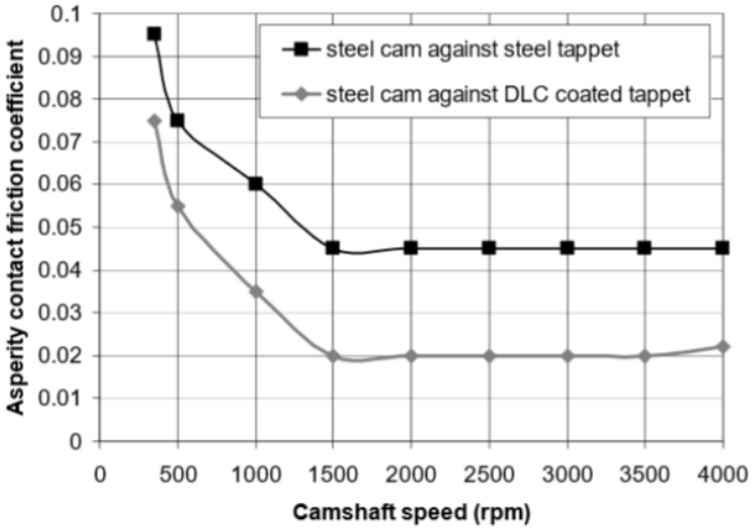


Figure 2.10 Change in friction coefficient (asperity contact) with different surface finishes (Calabretta and Cacciatore, 2010).

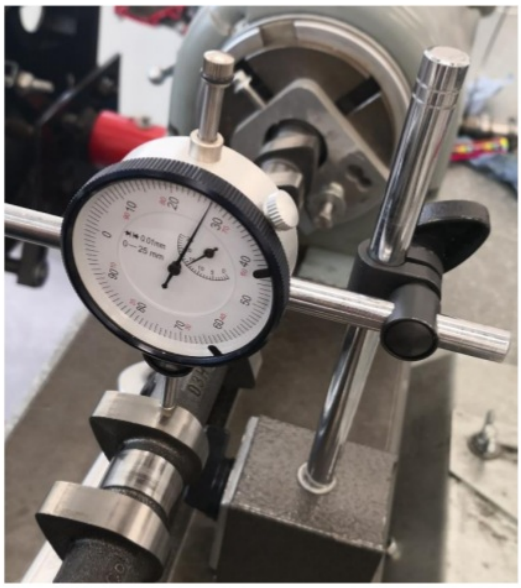
3. Physical Measurement Procedure

3.1 Camshaft

The most accurate measuring apparatus available for this investigation was a dividing head, used for measuring angular displacement and a plunger type dial test indicator (DTI) for linear translation. The dividing head was the driving mechanism for advancing the cam to the next measuring point. This device is considered accurate to 1 degree but relies on visual judgement to align each increment marker. The DTI can display linear displacement accurate to 0.01mm.

A small female taper was centre drilled in the head of the camshaft (sprocket) bolt as the dividing head and tailstock include male tapered spindles for concentric mounting. A clamping mechanism was fabricated to lock the position of the shaft in relation to the dividing head. The final configuration can be seen in *figure 3.1*.

The DTI gauge was secured with a magnetic base and positioned vertically above the shaft. Concentricity of the camshaft was tested by measuring the DTI runout across various carrier bearing faces. A maximum runout of *±0.06mm* was recorded through one revolution. Finally, the base circle for the lobe was measured with a Vernier caliper.





Figures 3.1 Camshaft measuring equipment and final mounting solution.

The raw measurements were imported to Matlab, where a graph was generated for the inlet lobe profile, as shown in *figure 3.2*. The DTI runout introduces noticeable errors through the base circle measurements which should otherwise be a constant radius section. It is anticipated that some small runout errors are also carried through the duration of the lobe. However, this flaw is neglected outside of the base circle.

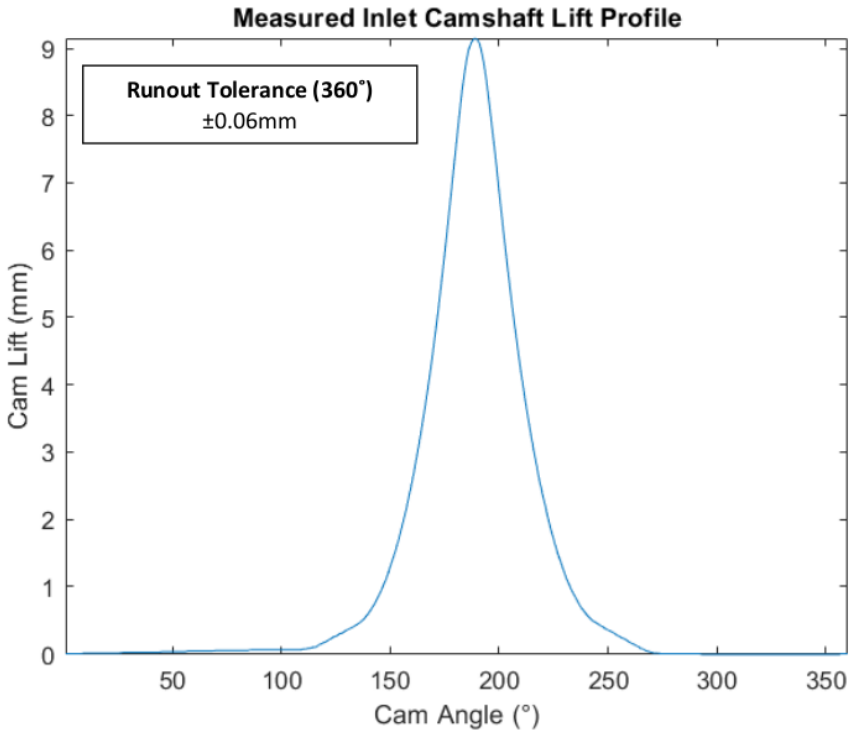


Figure 3.2 Cam lift profile from raw measurement data, using the optical dividing head and DTI.

The measured data is processed by taking the points from 180˚ to 360˚ and correcting the base circle to read zero until the first noticeable change in lift takes place. This half of the profile is then mirrored in place of the first 180 data points as the Ford camshaft lobes are considered symmetrical. The total measured runout effecting the final cam profile through lift was estimated to be lower than ±0.02mm.

3.2 Valve Spring

The valve spring deflections were measured using a Vernier caliper, with the spring fitted to the cylinder head. This determined the seated length and compression distance, revealing the working range of the spring.

Further physical measurements were obtained with the valve spring removed, including natural spring height and weight.

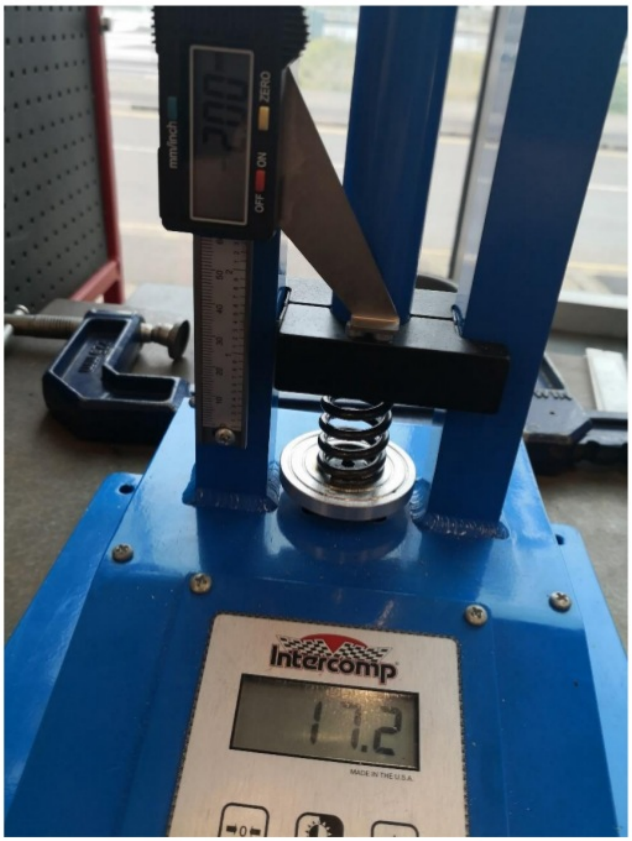


Figure 3.3 Measuring force and displacement properties - Ford Duratec inlet valve spring (Intercomp Spring tester).

The force displacement properties were obtained using the Intercomp Spring Rate Tester in *figure 3.3*. A spring load was recorded for every millimetre of displacement from *1-15mm*. The results from the spring rate test were converted from *lb/mm* to *N/m* and plotted in Matlab.

Figure 3.4 shows a good linear fit, with an R-Squared value of *0.9997*. The 1st term in the equation $y = mx + c$ reveals the gradient of the curve, distinguishing the valve spring rate.

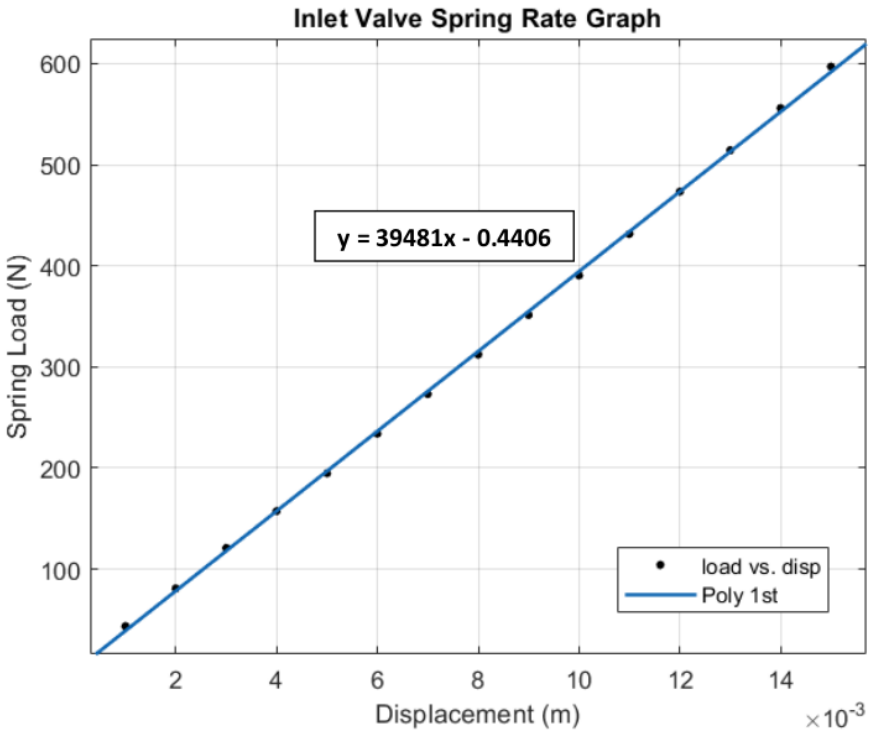


Figure 3.4 Force Displacement graph for the OEM Ford Duratec inlet valve spring.

The linear relationship is an accurate model of the Ford Duratec valve spring as the distance between each coil is equally spaced, resulting in a consistent reaction through the valve opening and closing events. A progressive spring would become stiffer as it is compressed and can usually be found in high speed internal combustion engines as supported by (Sequenzia *et al.*, 2011).

3.3 Inlet Valvetrain

The weight of each component was recorded and the recommended tappet clearance was extracted from the engine technical specifications provided by (Autodata, 2020). The stipulated clearance is an average of the recommended hot and cold values.

To summarise, *table 1* combines the results from the previous measurement methods. The Complete Valvetrain reflects half of the spring weight due to one end of the spring being fixed against the cylinder head.

Table 1 Valvetrain (Measured) constants

	Property	**Value**	**Unit**
Camshaft	Lobe height	9.15	mm
	Base circle radius	16.5	mm
Valve Spring	Free height	42.6	mm
	Seated height	38.5	mm
	Seated compression distance	4.1	mm
	Spring rate	39,481	N/m
	Weight	34.7	g
Tappet Bucket	Tappet clearance	0.25	mm
	Weight	33	g
Collet (pair)	Weight	1.8	g
Spring Retaining Cap	Weight	9.8	g
Valve	Weight	47.2	g
Complete Valvetrain	Total Weight (1/2 spring)	109.15	g

4. Valvetrain Kinematics

4.1 Valve Lift Derived from Cam Measurements

The kinematics of the valvetrain are determined by the interaction between the cam and the valve. Ford have implemented a directing acting configuration, by choosing a tappet bucket design. This results in an explicit relationship between the cam and valve lift which can be mathematically modelled as the contact point slides along the bucket.

The diagram in *figure 4.1* reveals the trigonometric relationship between the contact point ***B,*** the centre point of the cam base circle ***O*** and the vertical distance to the tappet bucket ***A***. In this instance, ***OA*** will be used to obtain valve lift.

Length ***OB*** is the base circle radius plus additional eccentric cam lift (*section 3*). This can also be denoted as ***S***.

An imaginary axis can be placed through centreline of the lobe, intersecting the maximum lift point (cam nose) and another axis placed through the vertical. From this, φ represents the angle that the nose has turned from the vertical axis, θ is the angle from the nose to the contact point and α is the angle between the contact point and ***OA***.

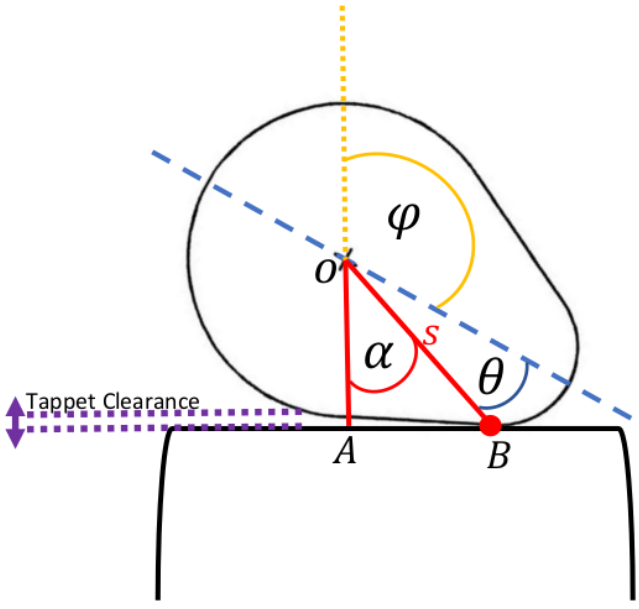


Figure 4.1 Diagram of cam-to-tappet interface with reference to contact point and valve lift.

As a result of these relationships, valve lift (derived from ***OA***) can be determined by executing the following process.

Since,

$$Cos(\alpha) = \frac{Adjacent}{Hypotenuse}$$

Then,

$$Cos(\alpha) = \frac{OA}{S}$$

$$\therefore\ OA = S \times Cos(\alpha)$$

From *figure 4.1*, α can be expressed as,

$$\alpha = 180 - (\theta + \varphi)$$

Substituting α yields the valve lift equation,

$$OA = S \times Cos(180 - (\theta + \varphi))$$

Equation 2 Valve lift (inside matrix).

*It is necessary to rearrange the values in **S;** so that maximum camshaft lift occurs when $\theta = 0$ as depicted in figure 4.1.*

This method can be solved in Matlab by creating a column vector of θ and a row vector of φ, per cam lobe measurement (every 1 degree), resulting in a 360 x 360 matrix as demonstrated in *figure 4.2*.

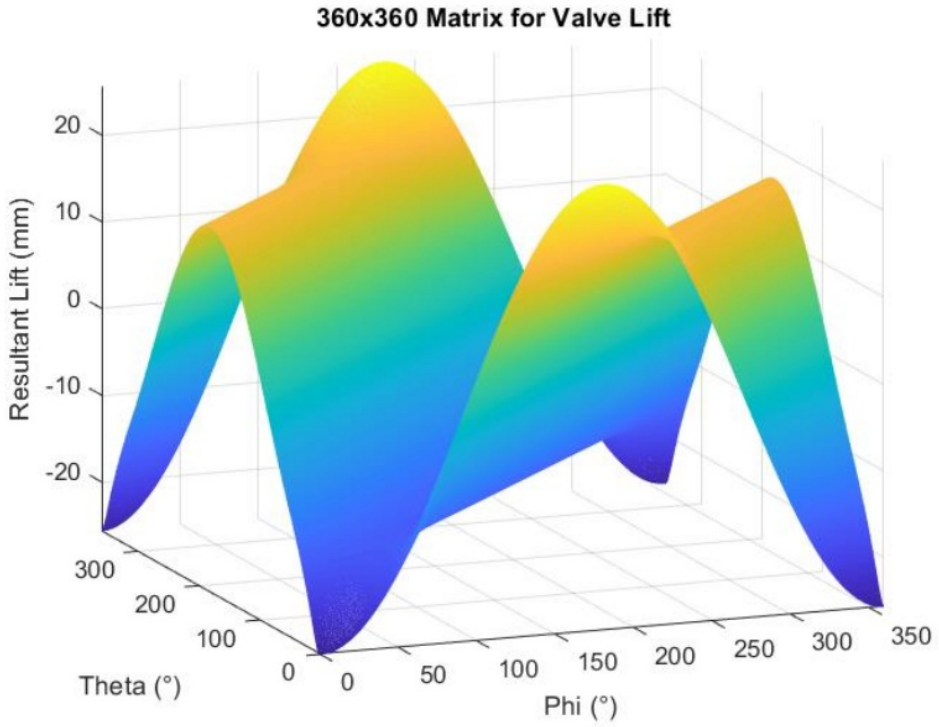


Figure 4.2 3D representation of 360 x 360 valve lift matrix.

The maximum values for ***OA*** are taken from each column of φ and the final valve displacement per degree cam angle is found by,

$$L_v = OA_{max} - R - C_t$$

Equation 3 Valve lift (resultant)

The tappet clearance (C_t) is only deducted from values of the resultant valve lift that are greater than the magnitude of the clearance gap. In the case of the Duratec, inlet valve lift begins when

$OA_{max} - R => 0.25mm.$

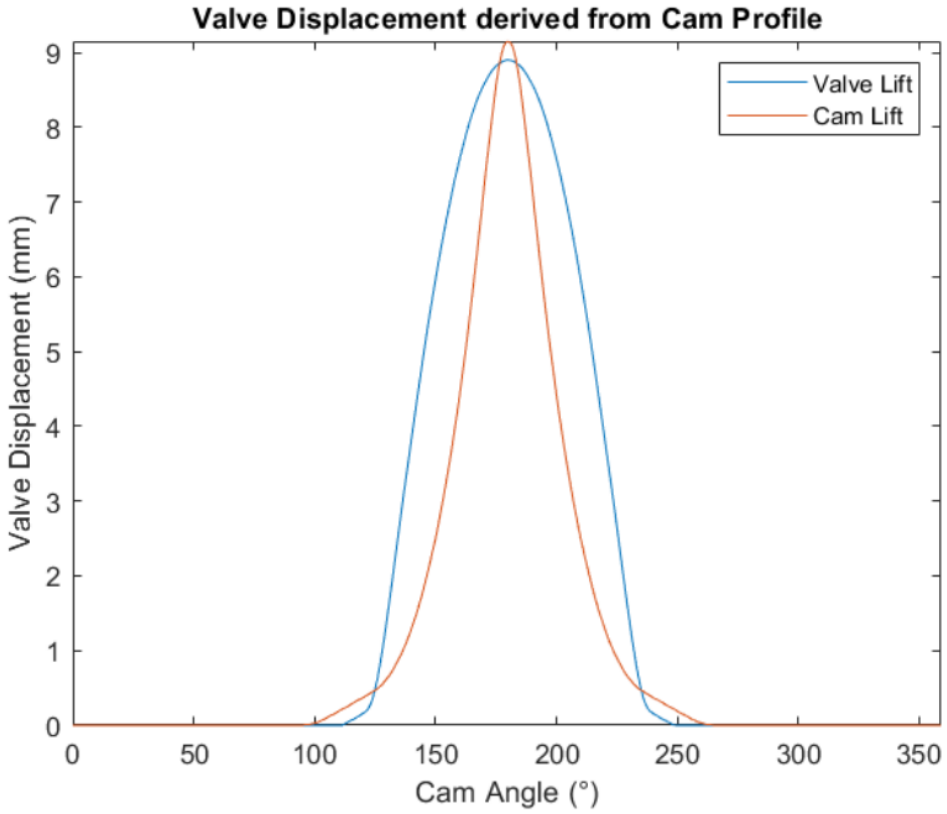


Figure 4.3 Resultant valve lift from matrix, derived from measured cam profile.

Peak valve lift is reduced from the original cam lift owing to the tappet clearance (valve lash). The difference in profile shape is a product of the sliding action through the cam-to-tappet interface. Valve event duration and peak lift are typically used as triangulation points in valve motion synthesis. Therefore, it was necessary to confirm these fundamental values before continuing with further investigations.

4.2 Lift Properties and Valve Motion

Table 2 Inlet valve event properties.

	Measured Value	Actual Value	Units
Peak cam lift	9.15	9.11	mm
Valve Lash	0.25	0.25	mm
Peak Valve Lift	8.9	8.86	mm
Valve Duration	208	208	˚ Crank angle
Duration results follow recognised convention for onset of lift and fall, 0.05" (approx. 1mm)			

The matrix method is appropriate for verifying basic valve displacement data as shown in *figure 4.3* and *table 2*. Only minor discrepancies occur between the original (actual) cam data provided by (KentCams, 2020) and the measured results from this study. The measurement procedure yields accurate valve duration prediction, which is an important constraint in valvetrain design.

To establish the design considerations for maximum valve lift it is necessary to calculate the lift/diameter ratio. The Duratec inlet valve diameter was measured at *35.15mm*. This translates to a ratio of 0.25. the study by (Xu *et al.*, 2019) demonstrates how a ratio of 0.25 is the typical value for the mass flow rate to plateau (at maximum relative pressure difference). This suggests the Ford Duratec inlet flow characteristics are likely to be good to optimal.

Higher order valve derivatives were calculated to establish the smoothness of the recorded cam data. If excessive noise or inaccurate data is present through these orders it is necessary to post process the original measured cam data until appropriate smoothness is achieved.

The valve acceleration profile is a primary focus of study, as this forms the basis for force related equations in the valvetrain system.

The 1st derivative of valve lift represents valve velocity,

$$V_v = \frac{dL_v}{d\varphi}$$

And 2nd derivative represents valve acceleration,

$$A_v = \frac{d^2L_v}{d\varphi^2}$$

Traces for each vector can be observed over one engine cycle in *figure 4.4*. Noise is visible with an increased order, indicating an issue within the measurement procedure.

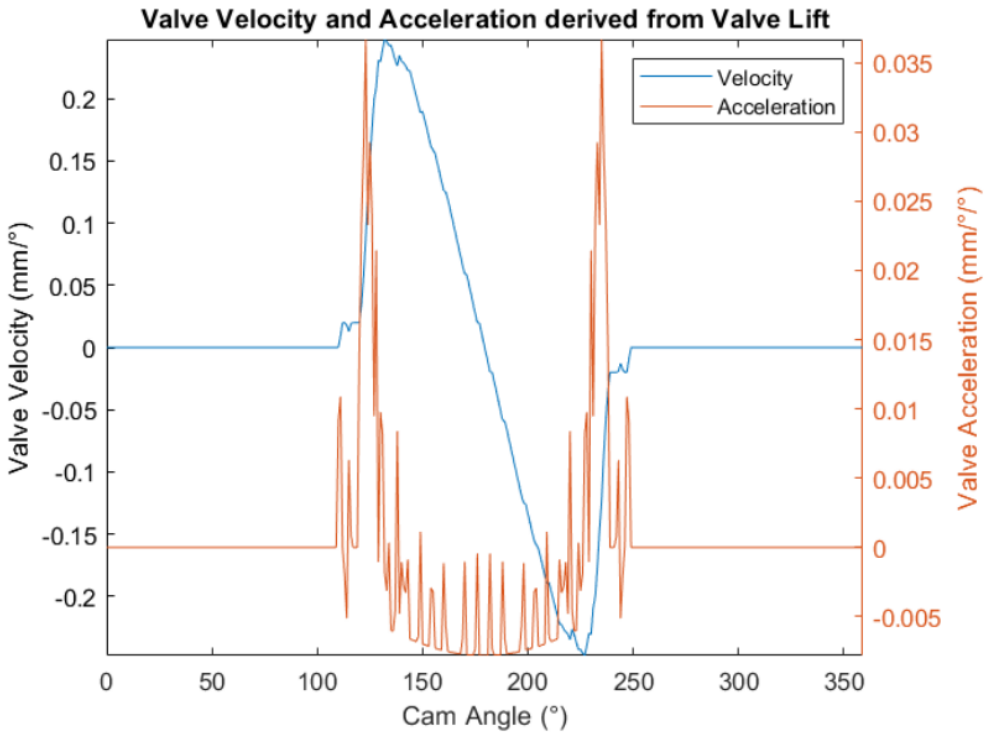


Figure 4.4 Valve velocity and valve acceleration derived from measured cam profile.

5. Valve Lift Matrix Study

It is hypothesised that the precision of the valve lift matrix can be further developed by increasing the matrix size, to improve the probability of obtaining the maximum values from each column of φ. This is achieved by expanding the original cam measurement count from 360 measurements to account for additional points between each whole degree. The proposed matrix test plan can be observed in *table 3*.

Table 3 Tested Matrices.

Matrix Size	Data Points per Degree	Decimal Size per Data Point	Total Data Points
360 x 360	1	1˚	129,600
720 x 720	2	0.5˚	518,400
3,600 x 3,600	10	0.1˚	12,960,000
36,000 x 36,000	100	0.01˚	1,296,000,000

This study was undertaken in Matlab by generating a loop that inserted the stipulated quantity of blank elements, before applying a shape preserving piecewise cubic polynomial (PCHIP) function to compute an interpolation command on the resultant vector.

An example of the Matlab script can be seen in *figures 5.1-5.2*.

```
%% VALVE LIFT MATRIX (36,000 x 36,000) [0.01 DEGREE INCREMENTS]
% Phi vector in 0.01 deg steps
PhiD = (0 :0.01: 359.99); % ADJUST FOR 0.01 deg steps

% Extend 360 values to 36,000 values

Vi = RLI'; % Original 360 ROW VECTOR
[ri,ci] = size(Vi);
p = 99; % number of NaN to be inserted between each values of V
Vf36k = NaN(1,ci*(p+1)-p);
[rf,cf] = size(Vf36k);
k = 1;
for i = 1:p+1:cf
    Vf36k(i) = Vi(k);
    k = k+1;
end
Vf36k; % RESULTANT EXTENDED VECTOR

Vector36k = fillmissing(Vf36k, 'pchip'); % INTERPOLATION COMMAND
```

Figure 5.1 Matlab script to show the valve lift matrix "for loop".

A second loop was run to select and compile the maximum values for the specified matrix size. This condensed vector enables subsequent calculations and comparisons to be made with respect to each degree of cam angle.

```
%% [ ONLY MATRICES >360 ] Find maximum values per degree (from max vector)

% check ====> [ADD 16.5 to VLift (1,1) AFTER run]

VLift36k = zeros(360,1);
i = 0;
l = 1;
for t = 1:0.01:359.99   % change this counter depending on matrix size
    y = t+99;            % change this counter depending on matrix size
    l = l+1;

    VLift36k(l,:) = max(Max36k(t:y,:));
end
```

Figure 5.2 Matlab script to show the valve lift matrix "maximum phi for loop".

The robustness of the matrix script was analysed using a perfectly smooth (sine) function of period 2π, replicating the vector dimensions of the measured data. The details of this validation study can be found in *appendix*.

Figure 5.3 demonstrates the effect of increasing the valve lift matrix size through each iteration of the test. A reduction in noise can be observed as the size is increased up to *3,600 x 3,600* where the results appear to converge. The additional data points from the *36,000 x 36,000* matrix yields no further improvement. For this application, it can be said that interpolating the measurement values to *0.1mm* is the most effective method for matrix error reduction. The dedicated CCMM produced by (Adcole, 2020) also uses *3,600* measurements, further supporting the findings in this study.

The smoothness of the lowest points across the duration of the valve lift are lost as the matrix size is increased. This suggests further data processing is necessary to clean the raw measured camshaft data to a higher standard before the optimised valve lift matrix is applied.

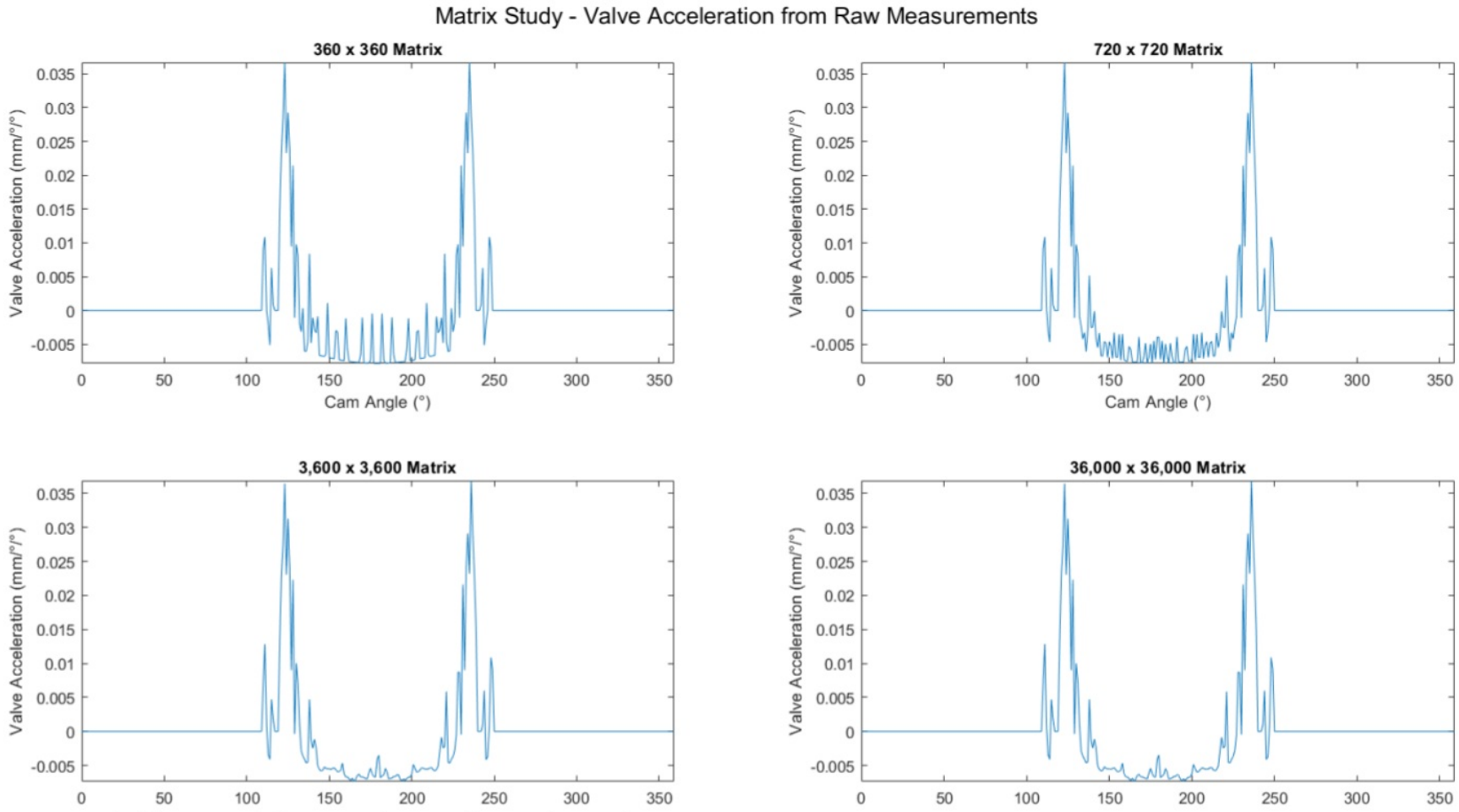


Figure 5.3 Valve lift matrix study for measured cam profile – acceleration plots.

6. Cam Measurement Post Processing

6.1 Measurement Analysis

The raw cam measurement data was analysed to find the cause of the errors observed through velocity and acceleration (*section 4.2*). The errors (being exaggerated with an increased order of differentiation) indicate an inconsistent rate of change through each recorded measurement.

This evidence suggests either one, or a combination of following phenomenon are in effect.

1. The DTI stylus may be pushed to an offset angle from the vertical (if any free play exists), due to the reactive force from the curved surface of the camshaft lobe as it rotates on the dividing head. This tipping action may result in minor measurement deviations.
2. The surface finish of the camshaft lobe may influence the results. As the measured camshaft was removed from an engine with reasonable road mileage, it is possible that some undulations due to wear may be transferred to the recorded data.
3. Error may exist where the tip of the DTI migrates excessively from a perpendicular position with respect to the measuring surface. Although this type of error is more consistent with Dial test indicators that operate in an arc (from a pivot), the aggressive and constant curvature of the cam lobe may affect the true readings in some form.
4. The DTI and dividing head (accuracies of *0.01mm* and *1˚* respectively) may not be the most consistent devices for this repetitive operation. User oversights (such as rounding to the nearest increment), may trigger errors in measurement resolution. This can cause successive measurements to either undershoot or overshoot from the actual amount, thus affecting the rate of change through the entire operation.

It is conceivable to refine the measuring technique and increase the number of measurement samples to improve the quality of the raw data, but this approach ultimately relies on the use of more sensitive measuring apparatus. The magnified valve lift plots in *figure 6.1* illustrate the form of errors causing the noise. From this evidence, the most appropriate solution was found within post processing methods.

The 2nd order derivative of cam lift was used to compare noise through post processing, to eliminate the possibility of errors being introduced through the valve lift calculations.

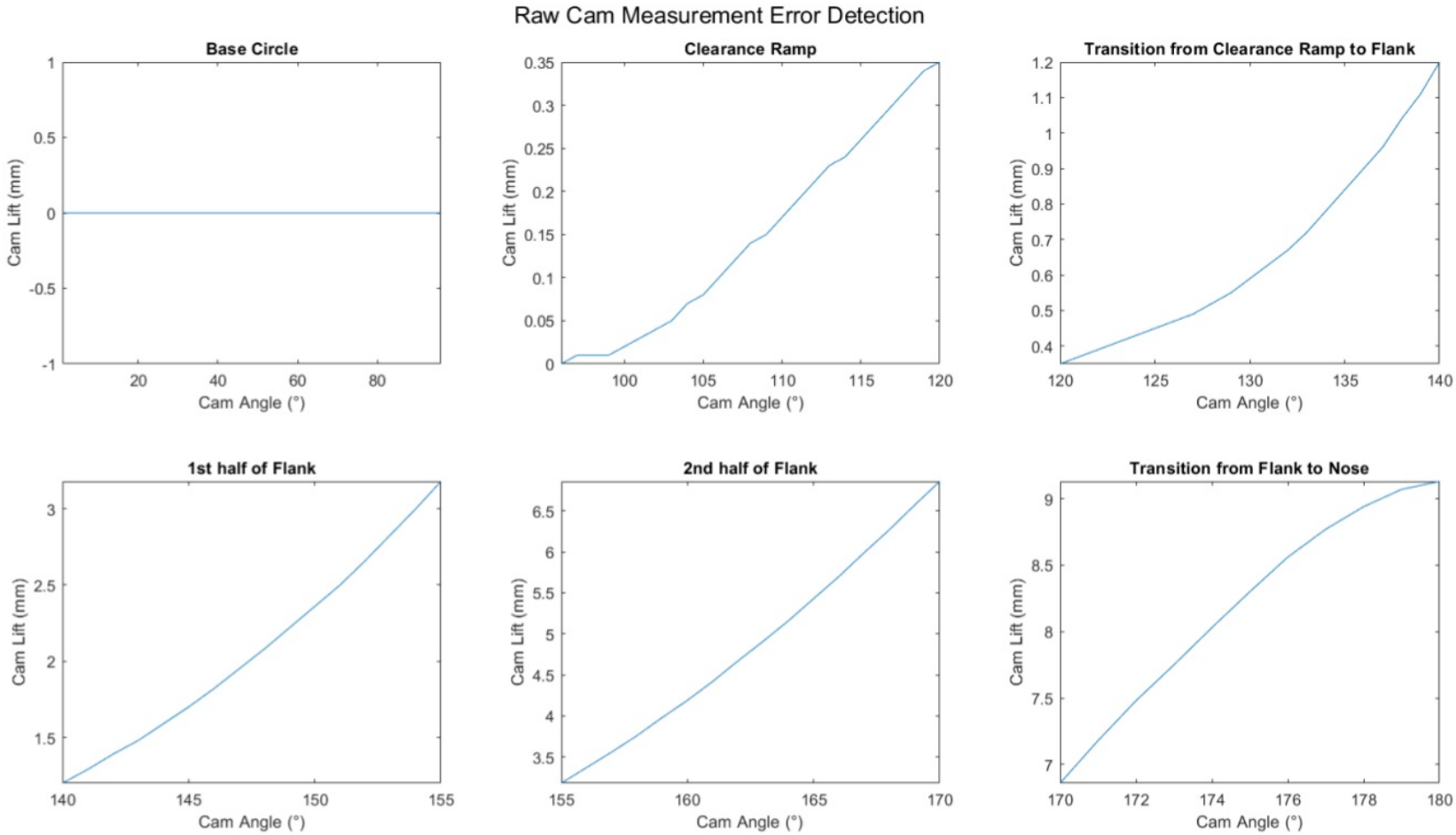


Figure 6.1 Cam lobe measurement error detection.

6.2 Smoothing Methods

A smoothing function was applied to the raw cam measurement vector as an attempt to improve the quality of the results. Of all smoothing attempts, the most suitable function used a moving average method to process the data through a lowpass filter. A user defined span was input, with filter coefficients being equal to the reciprocal of the span. A range of span values were investigated and compared against the raw measurement data. *Table 4* highlights the results.

Table 4 Smoothing analysis.

Span Percentage	Peak Cam Lobe Lift	Accuracy
Raw Measured Data	*9.13*	-
1 %	9.1167	99.85 %
3 %	8.8936	97.41 %
5 %	8.5871	94.05 %

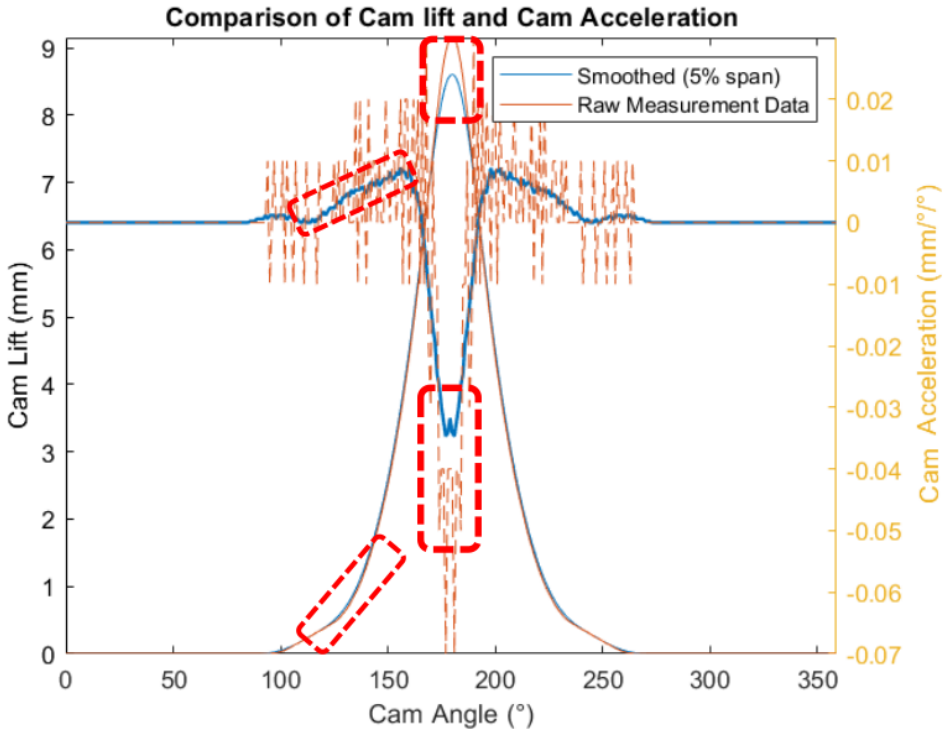


Figure 6.2 Comparison of 5% Smoothed model to measured data, highlighting inaccuracies.

Figure 6.2 compares the raw measurement data to the 5% smoothed results for both camshaft lobe lift and resultant cam acceleration. The smoothing function achieves a reduction in acceleration noise through the duration of the cam. However, this is at the cost of inaccuracies through peak lift as well as the transition between the clearance ramp and flank. For this reason, the smoothing function was considered too aggressive for improving the measured cam lift profile to an acceptable standard for higher order valve motion.

6.3 Curve Fitting Methods

6.3.1 Gaussian Curve

A Gaussian curve was chosen to replicate the valve lift function due to its inherent bell-shaped profile.

Gaussian Equation,

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}}$$

```
% Resulting Curve Equation
x = Theta360;
y = a1*exp(-((x-b1)/c1).^2)...
    + a2*exp(-((x-b2)/c2).^2)...
    + a3*exp(-((x-b3)/c3).^2)...
    + a4*exp(-((x-b4)/c4).^2)...
```

Figure 6.3 Properties of the Gaussian curve embedded into the Matlab script. R-square value of 0.9999 @ 4 terms.

Figure 6.4 compares the basic Gaussian fit with the raw measured data. Unlike the smoothing process, the bell curve closely matches the peak cam lift (within 0.*02mm*)**.** The fitted curve breaks down in modelling the clearance ramps due to the linearity of these sections, corrupting the acceleration trend. Noise is significantly improved when compared to the raw data.

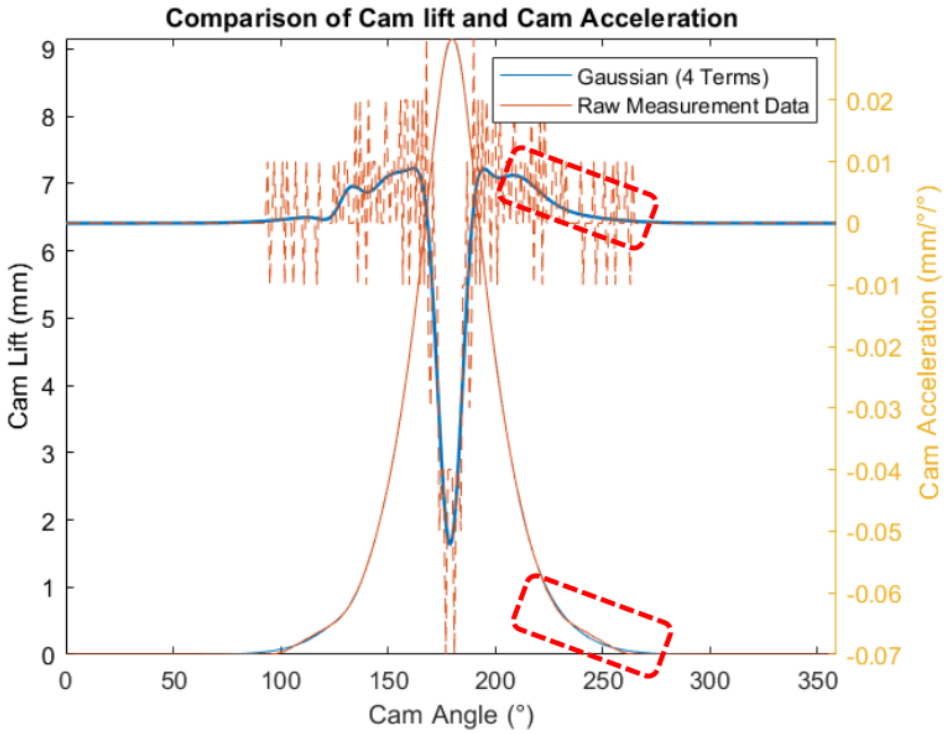


Figure 6.4 Comparison of 4 term Gaussian model to measured data, highlighting divergence.

6.3.2 Improved Gaussian Curve

The 4 term Gaussian model was further developed by integrating a polynomial function across each clearance ramp. This was achieved by splitting the raw measurement cam profile into appropriate sections for individual curve fitting procedures. *Table 5* details this.

Table 5 Blend of multiple curve fits to improve quality of measured cam profile.

Cam Rotation Angle	Curve Type	Order Count
0 ˚ → 95 ˚	Linear	0
96 ˚ → 141 ˚	Polynomial	9th Degree
142 ˚ → 222 ˚	Gaussian	3 Terms
223 ˚ → 267 ˚	Polynomial	9th Degree
268 ˚ → 359 ˚	Linear	0

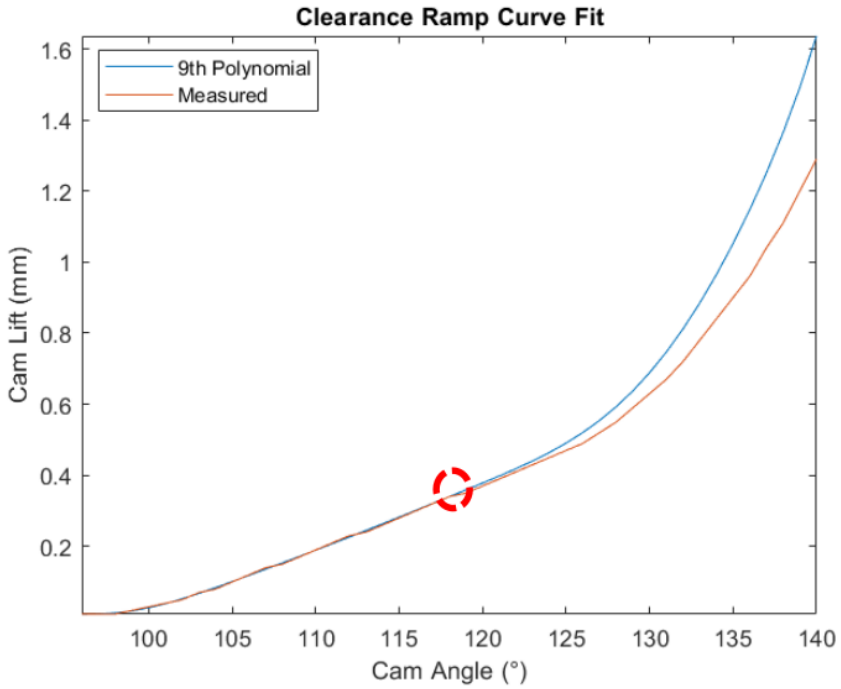


Figure 6.5 Clearance ramp of the cam profile. The 9th degree polynomial can be seen to converge until approximately 118˚.

The Gaussian curve was fitted to the *79˚* range defined in *table 5*. The exclusion of both clearance ramps yields an R-square value of *1,* with a decrease in the number of terms required to precisely fit (from 4 to 3). *Figure 6.6* illustrates the convergence of this fit against the measured cam profile.

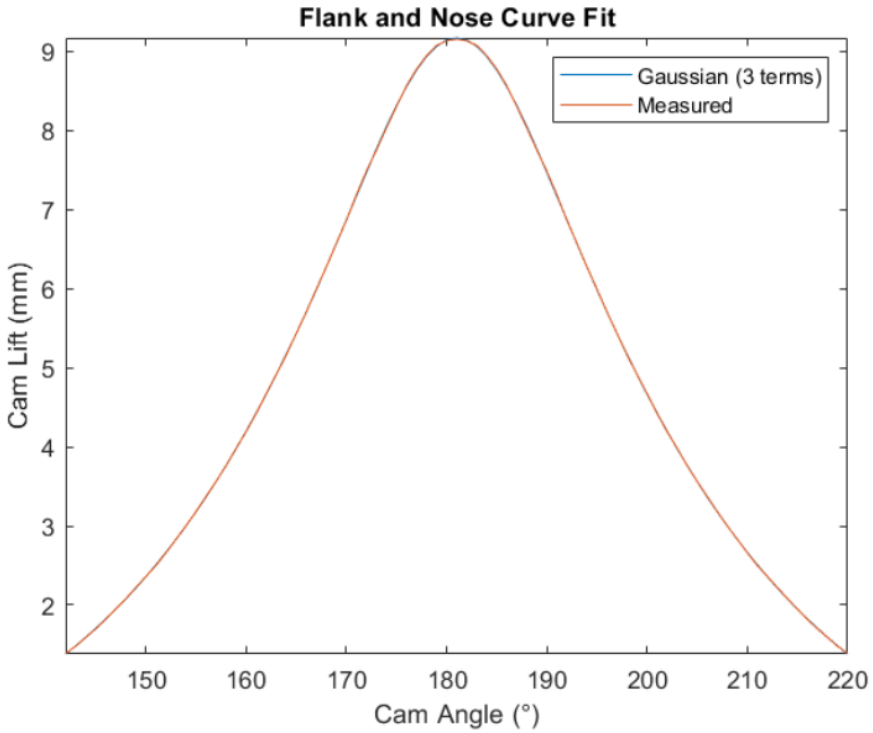


Figure 6.6 The flanks and nose of the cam profile. 3 term Gaussian curve-fit, fully converged.

The combined curve fits are displayed in *figure 6.7*. As each curve is driven by a unique equation it is necessary to ensure the transition is smooth. For this reason, the observed divergence of the 9^{th} degree polynomial (highlighted) is not a direct concern. A local interpolation (PCHIP) method was applied to produce a data patch, guiding a consistent curve around the joint by referencing original points through the measured lift data.

Figure 6.8 depicts the final curve fitted model (including data patch) and resultant cam acceleration. A significant improvement in cam lift correlation is achieved over the previous processing methods. The cam acceleration profile is smooth, with good definition through the clearance ramp and flank sections.

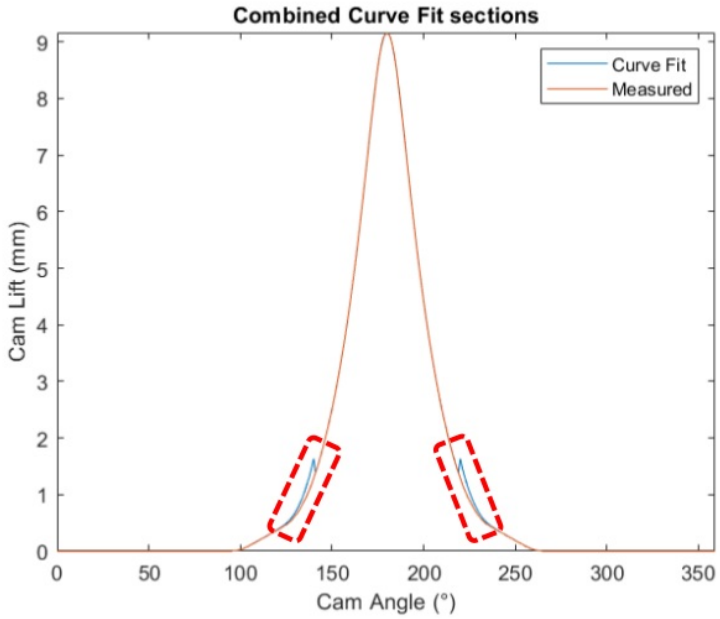


Figure 6.7 Combined curve fits before interpolation "data patch" was applied to the highlighted sections.

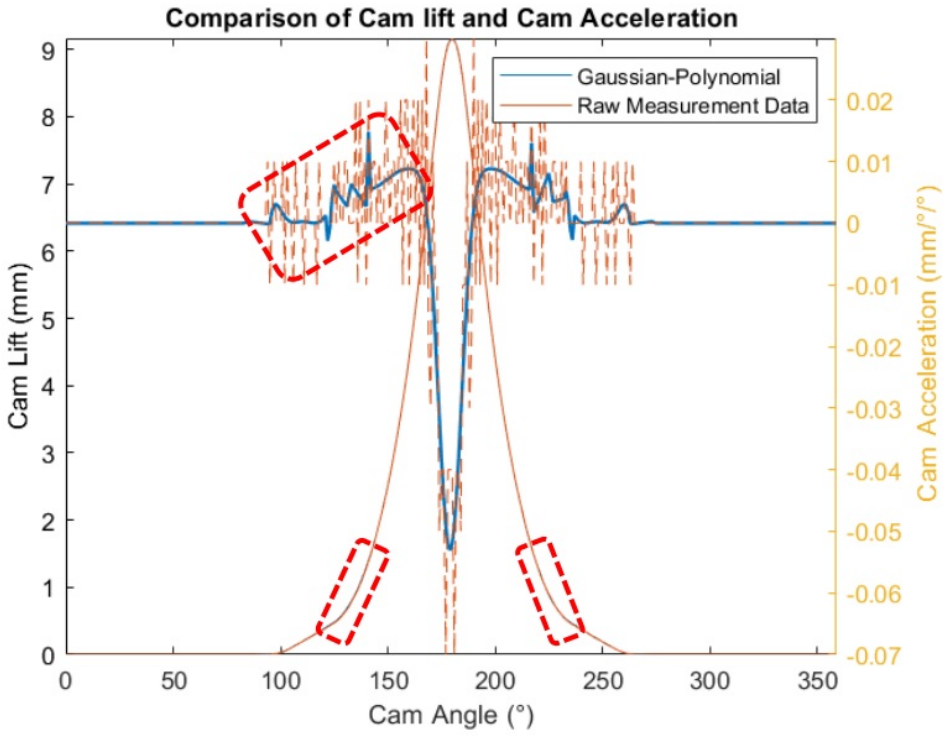


Figure 6.8 Final curve fit with good definition after "data patch" was applied to the highlighted sections.

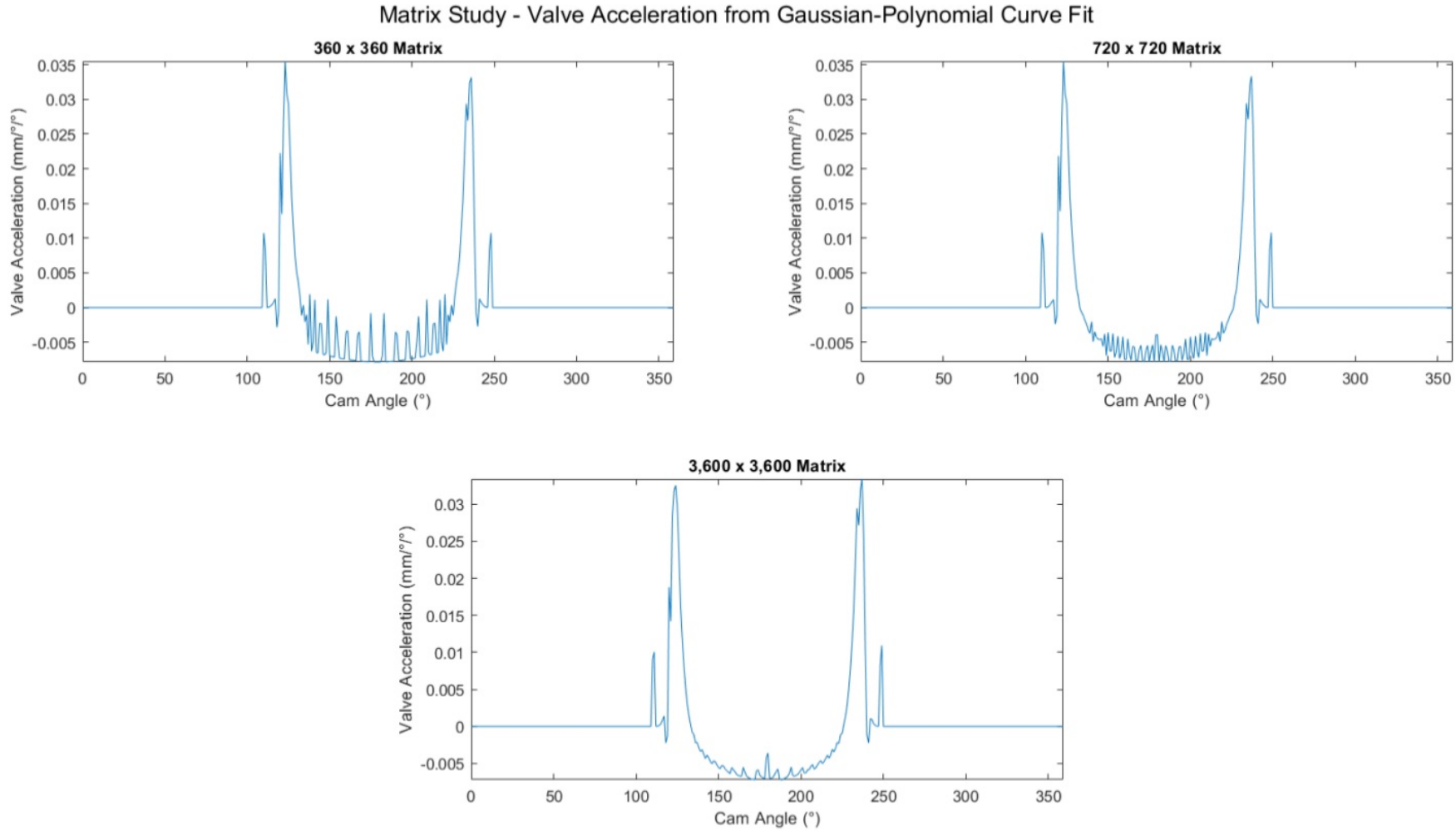


Figure 6.9 Valve lift matrix study for the hybrid curve fit model (Gaussian-Polynomial) – acceleration plots.

A significant improvement to the overall valve acceleration profile can be seen through the Gaussian-Polynomial matrix study (*figure 6.9*), particularly when compared with the previous raw measurement results. Peak valve accelerations through the flank to clearance ramp transition is improved with matrix resolution as well as a reduction in noise magnitudes. A consistent profile is now visible through the whole engine cycle, giving confidence for future calculations.

Table 6 includes some comparitive analysis between the raw and processed profiles. The peak values through all matrices (both studies) are within *10%* of the original measured values.

Table 6 Valve acceleration comparison for all valve lift matrix variations.

Profile	**Maximum Valve Acceleration (mm/˚/˚)**	**Minimum Valve Acceleration (mm/˚/˚)**
Raw Measured		
360 x 360	0.03668	-0.007808
720 x 720	0.03638	-0.007804
3,600 x 3,600	0.03646	-0.007293
Gaussian-Polynomial		
360 x 360	0.03548	-0.007818
720 x 720	0.03548	-0.007817
3,600 x 3,600	0.03334	-0.007184

7. Valvetrain Dynamics

This section will introduce forces acting through the valvetrain as a function of engine speed.

The Duratec package (MCR race car application) is subject to a (Sports 2000 Championship) locked engine map, run through an Omex 600 ECU. This raises the maximum engine speed from the OEM limit of *6800rpm*. The MCR race car logs engine parameters to a MoTeC dashboard for data analysis.

Appendix shows the Silverstone session where peak engine speed was recorded at *7512rpm* during an aggressive downshift. The data also reveals an average working range for the engine to be between *5000rpm* and *7200rpm*. From this, a weighted engine speed vector was produced in Matlab to evaluate the valvetrain for competition use.

The valve velocity and acceleration found through *section 4.2* was converted from angular displacement to distance over time by the following methods.

For valve velocity per engine speed,

$$Vel_v\ (\mathrm{m/s}) = V_v \times \frac{(6 \times RPM)}{1000}$$

For valve acceleration per engine speed,

$$Acc_v\ (m/s)^2 = A_v \times \frac{(6 \times RPM)^2}{1000}$$

3D interpretations of the valve motion through the engine speed range can be observed in appendix.

7.1 Friction Model

Some energy being transferred through the valvetrain is lost to resistive forces through the cam-to-tappet interface. This directly impacts the total energy available from the crankshaft. The following procedure will provide insight to the extent of power loss experienced in the system using the physical measurement data collected through *section 3*.

The contact force (cam-to-tappet) combines resistive forces due to inertia from the moving components and the effect of spring stiffness through the engine cycle. *Figure 7.1* demonstrates how the spring force component dominates contact forces through the lower engine speeds (valve lift profile), with inertia forces opposing the spring force as engine speed increases (acceleration profile).

$$F_c = (MM \times Acc_v) + \left[k \times \left(S_{cd} + \frac{L_v}{1000}\right)\right]$$

Equation 4 Contact force (cam-to-tappet interface).

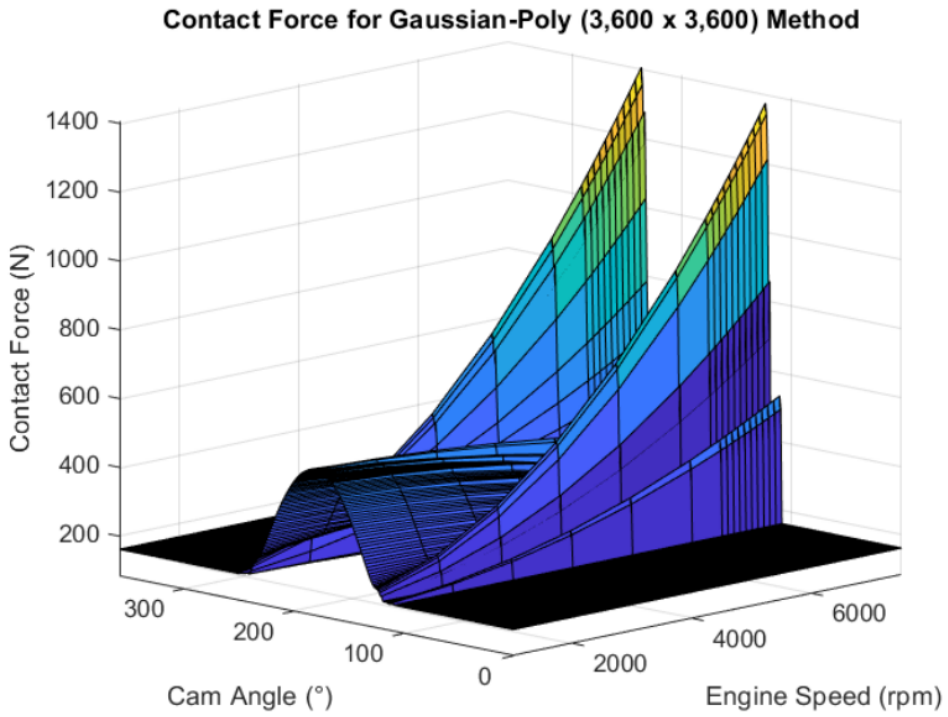


Figure 7.1 Contact force per cam angle, per engine speed for the Gaussian-Polynomial method.

The friction force per degree, per engine speed was found by,

$$F_f = F_c \times \mu$$

Equation 5 - Friction force

For this engine, the coefficient of friction was assumed to be constant at 0.1μ. This fixed value is likely to overpredict the effects of friction at higher engine speeds and oil temperatures, based on the work completed by (Calabretta and Cacciatore, 2010) and (Okarmus and Keribar, 2015).

Force due to friction was multiplied by the distance from the centre of the camshaft to the contact point (h). The inclusion of h yields an exaggeration of spring force at low engine speeds. Comparing friction torque (*figure 7.2*) to contact force (*figure 7.1*) highlights this.

$$T_f = F_f \times \frac{h}{1000}$$

Equation 6 - Friction torque

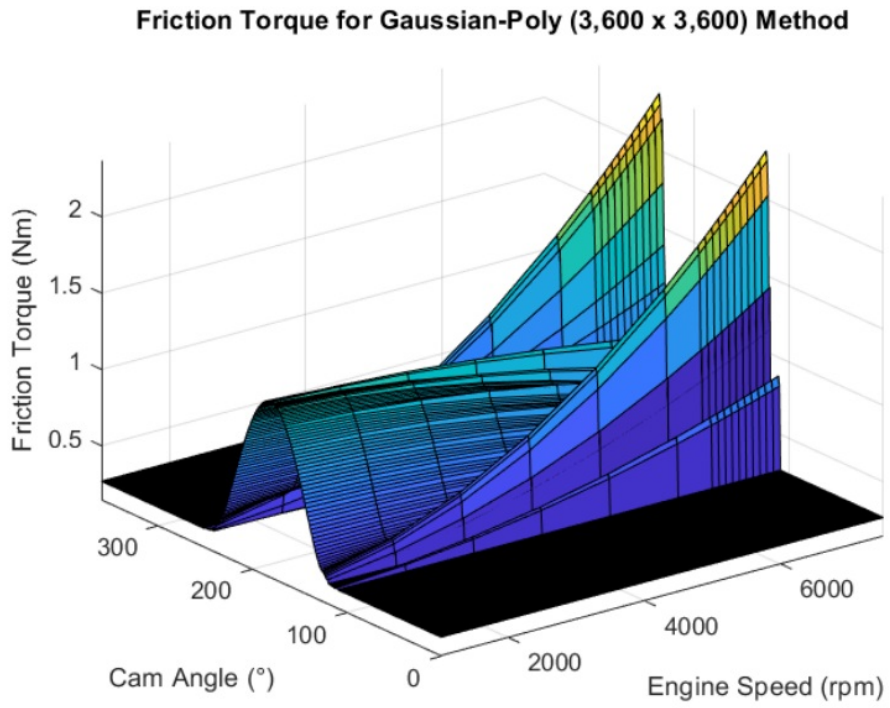


Figure 7.2 Friction torque per cam angle, per engine speed for the Gaussian-Polynomial method.

The total work done to overcome the frictional resistance was calculated by,

$$WD_f = \sum_{\theta=0}^{359} \frac{T_{f1} + T_{f2}}{2} \times \frac{2\pi}{360}$$

Equation 7 Friction work done (total from cycle, 1 valve)

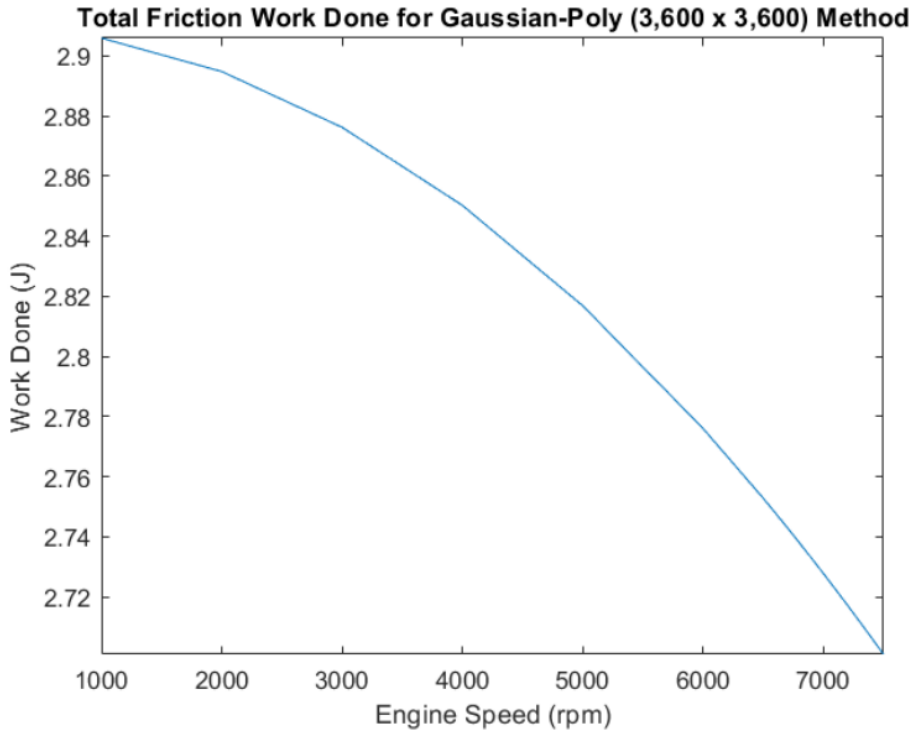


Figure 7.3 Total friction work done through the engine speed range.

Figure 7.3 represents the energy lost through the cam-to-tappet interface, for one inlet valve through the engine speed range. This equates to a *6% decrease* from *1000rpm* to *7500rpm*. The inertia force reduces the impact of valve spring force at the interface at increased engine speeds.

The total power loss suffered through the resistive forces at the cam-to-tappet interface, across all 8 inlet valves is given by,

$$P = (8 \times WD_f) \times \frac{RPM}{120}$$

Equation 8 Friction power loss at inlet (total from cycle, 8 valves)

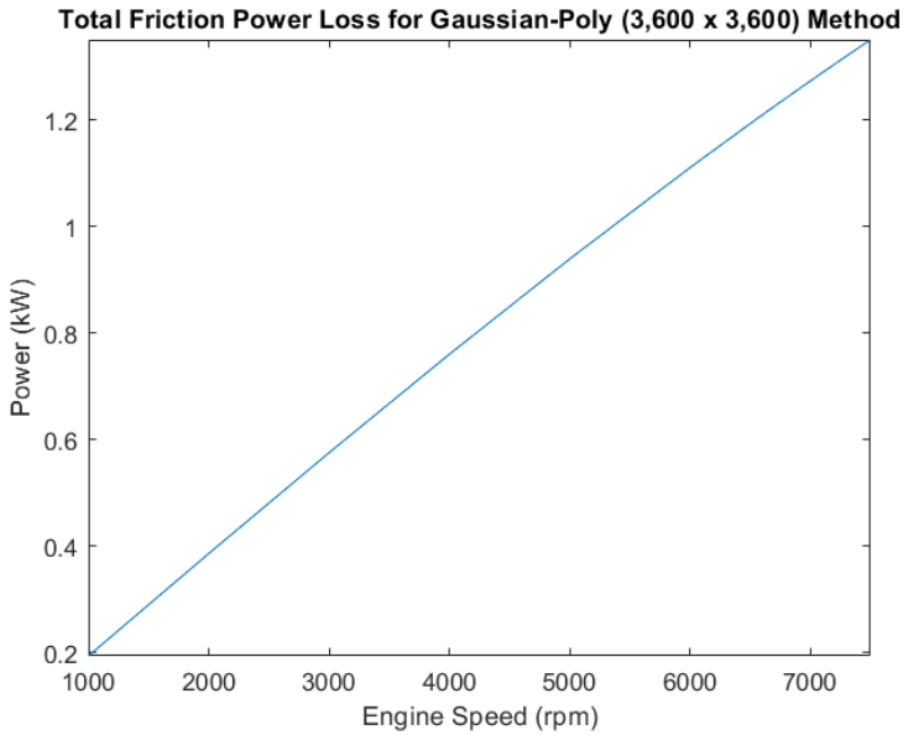


Figure 7.4 Total friction power loss through the engine speed range.

Peak power loss was calculated to be *1.35kW @ 7,500rpm*. This translates to an approximate total engine power loss of *2.5kW*, with the exhaust valve lift and mass being slightly lower than the inlet.

As the total friction power loss is driven by the friction coefficient, developments in surface finishing could yield improved engine efficiency. A DLC coated tappet is capable of reducing cam friction by 0.03μ, per (Calabretta and Cacciatore, 2010), translating to a *0.75kW* saving across the full valvetrain. Reducing the spring rate by *10%* saves an additional *0.18kW*.

8. Commercial Camshaft Manufacturers at Autosport International 2020

The author attended a trade day at Autosport International 2020, to network with industry professionals regarding this project. Three performance camshaft manufacturers were approached, with an ambition to obtain OEM cam lift data, for validation purposes.

In response, Piper Cams and Kent Cams each issued a copy of their version of the standard valve lift profiles.

It was evident that both manufacturers shared preference in measuring the vertical translation of the valve tappet bucket (flat tappet) as opposed to measuring the camshaft lobe.

Figure 8.1 shows Kent Cams valve displacement profiles.

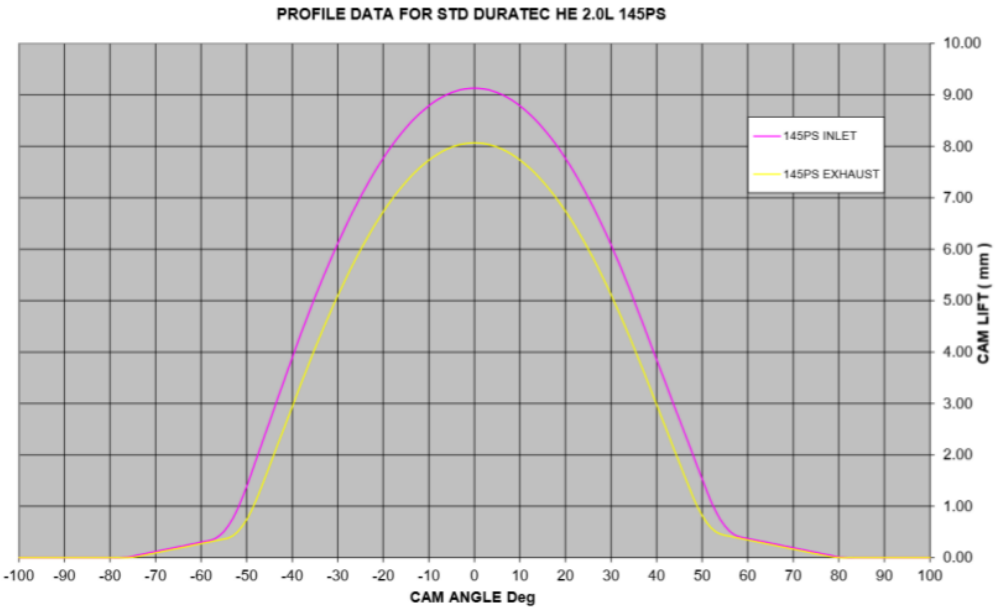


Figure 8.1 Kent Cams interpretation of original Ford Duratec inlet and exhaust valve profiles.

Note: The industry professionals refer to these valve lift plots as cam lift (y axis label). This suggests cam lobe profiles are generated directly from valve displacement data.

Detailed comparisons between the manufacturers data and the methods used through this investigation are drawn in *section 10*.

9. Transformation of Valve Lift to Cam Lift

The cam lobe and valve lift profiles are intrinsically linked when direct contact flat tappets are used. *Section 4.1* features the procedure for obtaining the valve lift profile directly from the cam lobe profile, by use of the matrix method. This chapter will study the inverse process through the application of coordinate geometry.

It is possible to define the valve lift (x, y) polar coordinates as follows, with reference to *figure 9.1*.

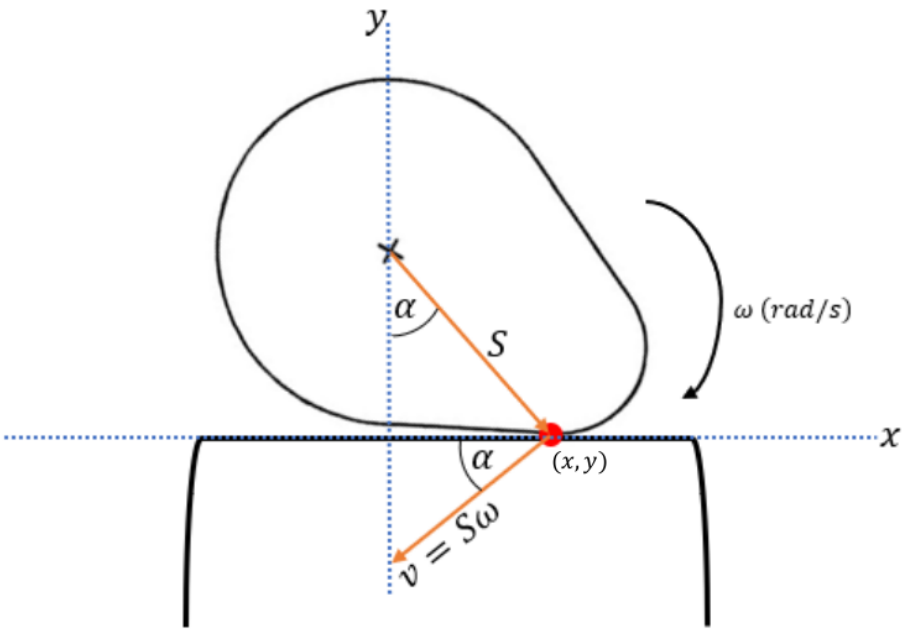


Figure 9.1 Diagram to show the relationship between cam and tappet.

Using trigonometry,

$$x = S\ sin(\alpha)$$

And,

$$V_v = v\ sin(\alpha)$$

$$\therefore$$

$$V_v = \omega x$$

To convert valve velocity into mm/°,

$$V_v = \frac{2\pi}{360} x$$

Since x is unknown,

$$x = \frac{360}{2\pi} V_v$$

Equation 9 Finding local coordinate (x)

To find y (inverse operations to finding valve lift from cam lift),

$$y = L_v + R + C_t$$

Equation 10 Finding local coordinate (y)

9.1 Rotation of Axes

Context for the proposed coordinate systems are presented in *figure 9.2*.

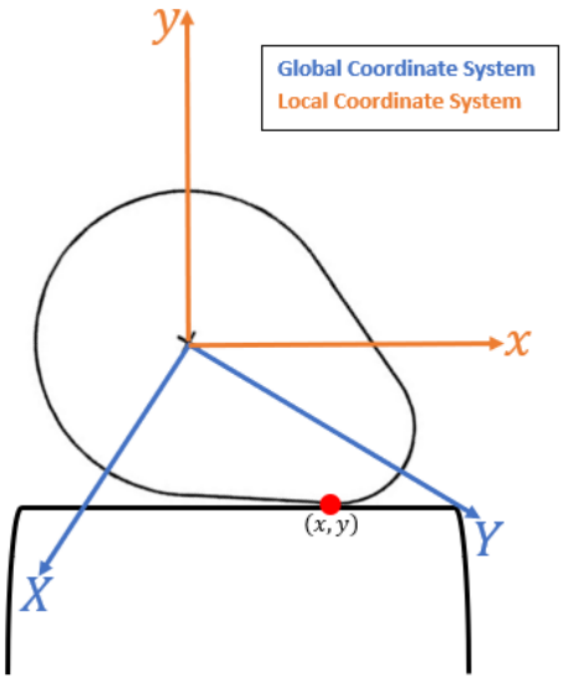


Figure 9.2 Relative coordinate system for rotation of axis. Applied to cam lobe and tappet (valve lift).

The geometric relationship is somewhat counterintuitive through this application. In principle, the position of the camshaft lobe is locked (global coordinate system) and the tappet bucket (local coordinate system) can be thought to rotate about the cam, with respect to the contact point coordinates (x, y).

A refined derivation for the rotation of axes technique was created, where the following observations were made.

- Hypotenuse of each triangle is formed parallel to local axes.
- Adjacent and opposite sides are formed parallel to global axes.

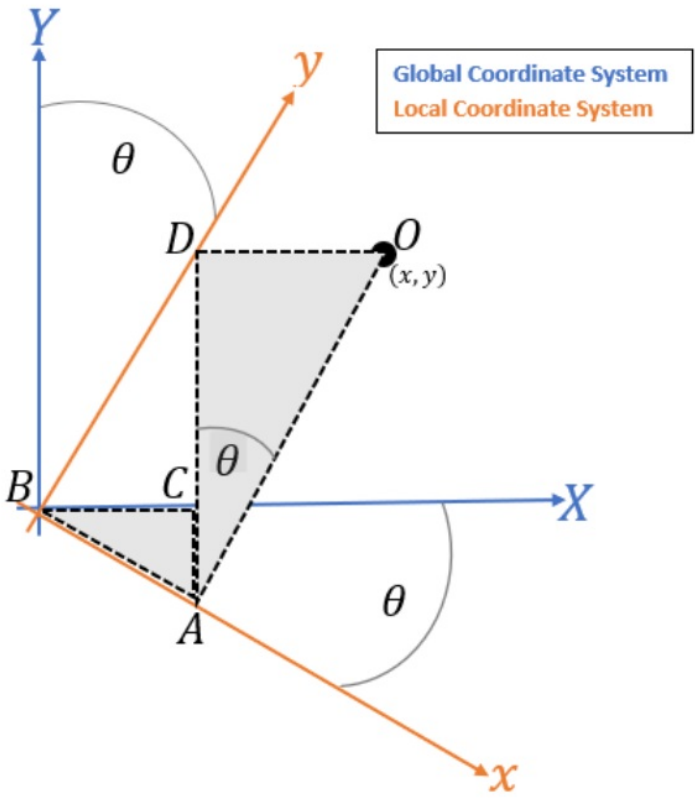


Figure 9.3 Diagram for finding global coordinate X.

To find global coordinate X,

From Triangle ABC,

$$X_1 = BC = x\cos(\theta)$$

From Triangle ADO,

$$X_2 = DO = y\sin(\theta)$$

$$\therefore$$

$$X = X_1 + X_2$$

$$\therefore$$

$$X = x\cos(\theta) + y\sin(\theta)$$

Equation 11 Finding global coordinate (X)

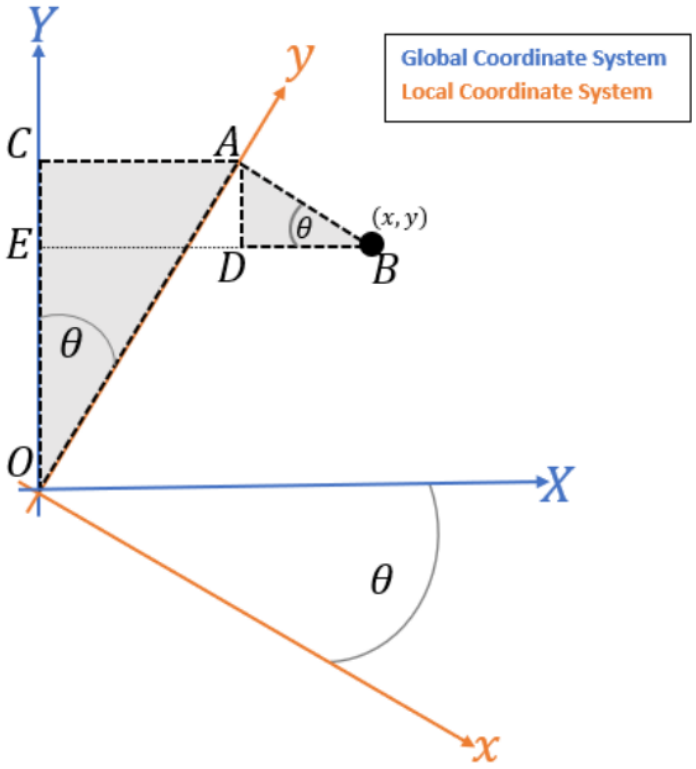


Figure 9.4 Diagram for finding global coordinate Y.

To find global coordinate Y,

From Triangle ADB,

$$Y_1 = AD = x\ sin(\theta)$$

From Triangle ABC,

$$Y_2 = CO = y\ cos(\theta)$$

$$\therefore$$

$$Y = EO = Y_2 - Y_1$$

$$\therefore$$

$$Y = y\ cos(\theta) - x\ \sin\ (\theta)$$

Equation 12 - Finding global coordinate (Y)

A matrix multiplication method was used in Matlab to resolve the final equations,

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \cos(\theta) & -\sin(\theta) \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix}$$

Equation 13 Matrix multiplication (rotation of axes)

Figure 9.5 shows an example of the resulting cam lobe profile being produced in polar form, an ideal output for camshaft design and manufacturing.

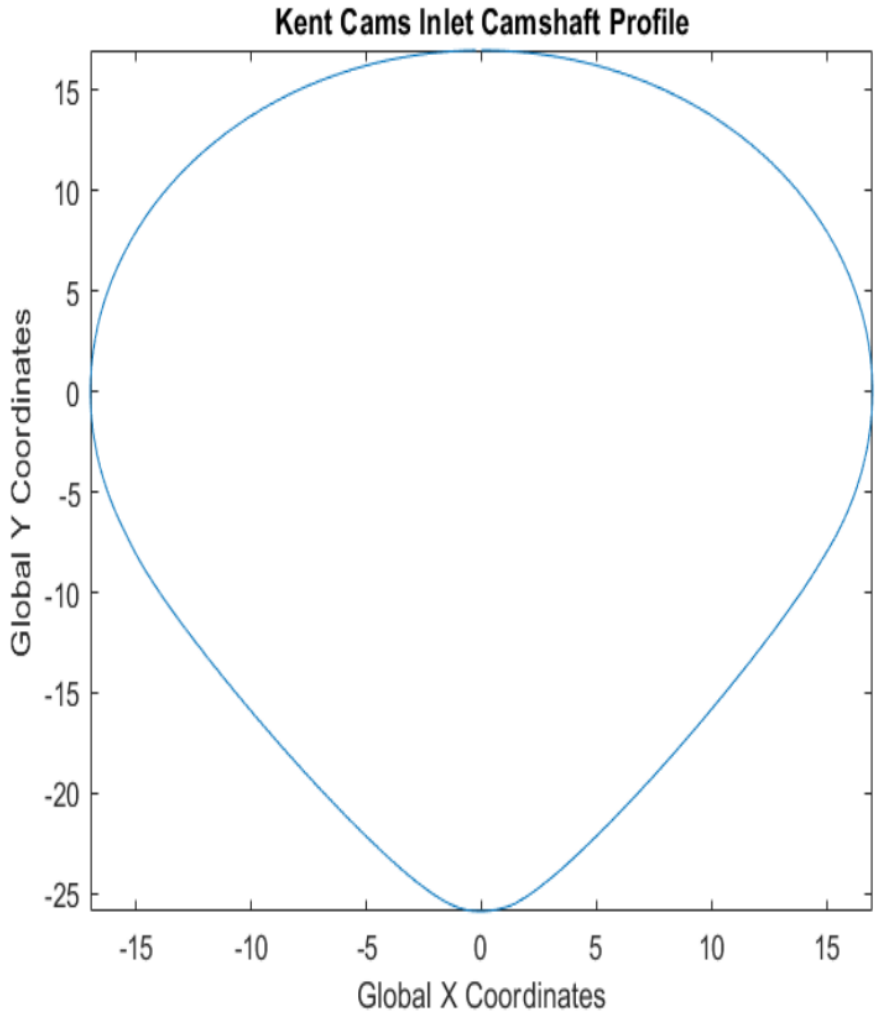


Figure 9.5 Cam lobe profile in polar form, given from the rotation of axes method.

10. Validation Study

The valvetrain characteristics that have been calculated through this investigation indicate some disparity in results, depending on the methods used to obtain valve lift profiles. This report has demonstrated how the size of the valve lift matrix can influence calculations that include terms relating to valve motion. In addition, the results generated by the smoothed data shows contrast to those produced by the raw cam measurement data.

One advantage of acquiring the Duratec valve lift profile from independent camshaft manufacturers is having opportunity to compare commercial interpretations of the original Ford valvetrain design. Whilst there may be some variation in design intricacies for each company, both versions of valve lift and resulting camshaft lobe design by Piper Cams and Kent Cams have been proven acceptable for intended use.

An added benefit to accessing these commercial profiles is for validating valvetrain characteristics derived from initial camshaft measurements and smoothing methods. The validation process will also determine the impact of increasing valve lift matrix size, and whether this improves the reliability of the valve displacement derivatives when compared to OEM data.

10.1 Comparison of Commercial Valve motion

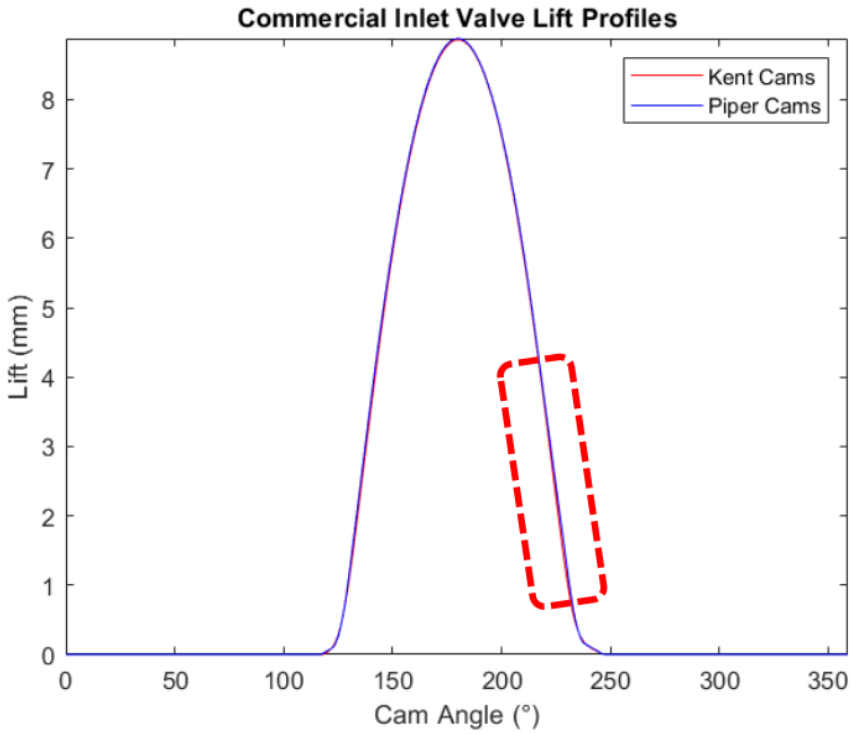


Figure 10.1 Comparison between Kent Cams and Piper Cams inlet valve lift.

Figure 10.1 presents the commercial valve lift profiles, normalised for valve lash, over one engine cycle. Kent Cams and Piper Cams measure valve lift to 6 decimal places, with maximum lift occurring at *8.864592 mm* and *8.881106 mm,* respectively. This sensitivity is necessary to generate an acceptably smooth profile for derivatives of higher order, such as acceleration and jerk. The difference of *0.016514 mm* in peak values is within the accuracy range of the DTI used to measure the cam lobe. Subtle variation can be observed through the profile, most notably in the closing flank as outlined. This suggests both companies use unique measurement/design techniques.

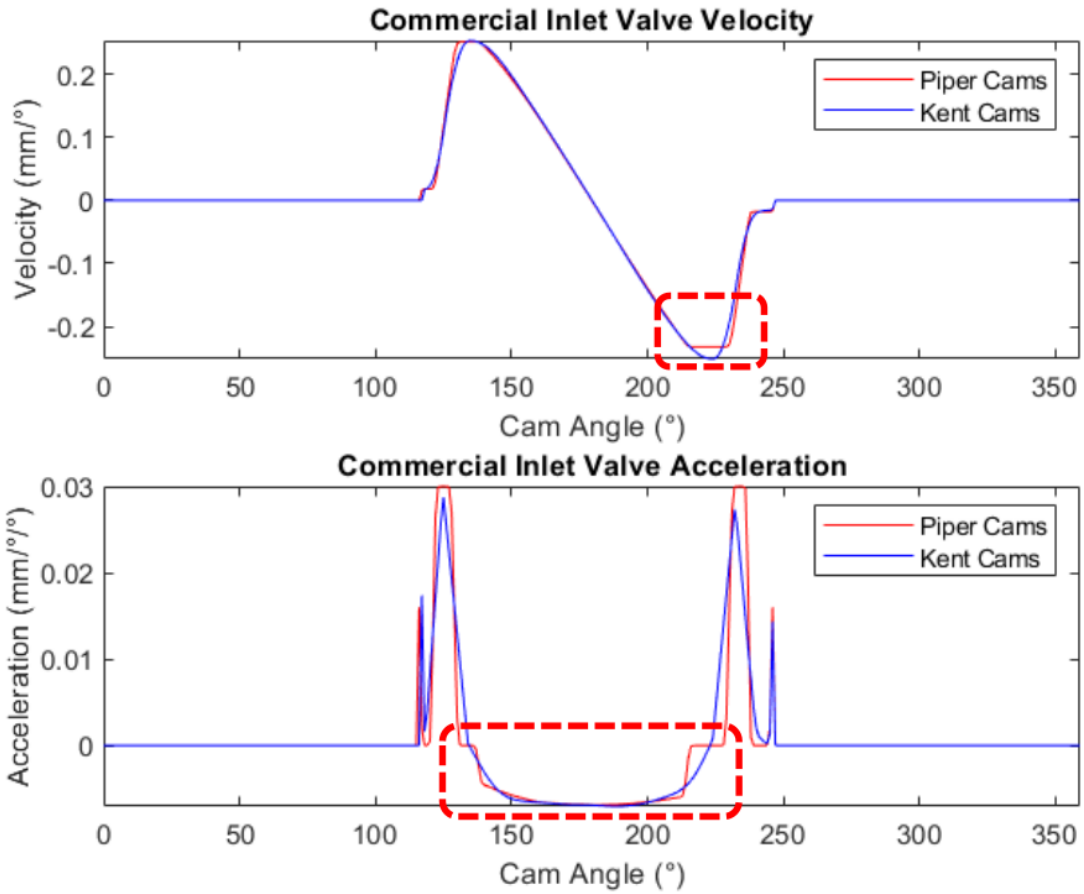


Figure 10.2 Comparison between Kent Cams and Piper Cams inlet valve velocity and acceleration.

The most noticeable difference between the two commercial profiles is the method in which they are processed for smoothness. The Piper Cams valve displacement demonstrates localised constant velocities and accelerations through the duration of the cam. This is pronounced asymmetrically, on the closing flank of the cam lobe. The evidence suggests this could have been engineered to encourage specific valvetrain behaviour, or that a noise suppressing filter was used to process original measurement data.

Kent cams appear to maintain a smooth profile, that closely resembles the valve motion generated through this investigation.

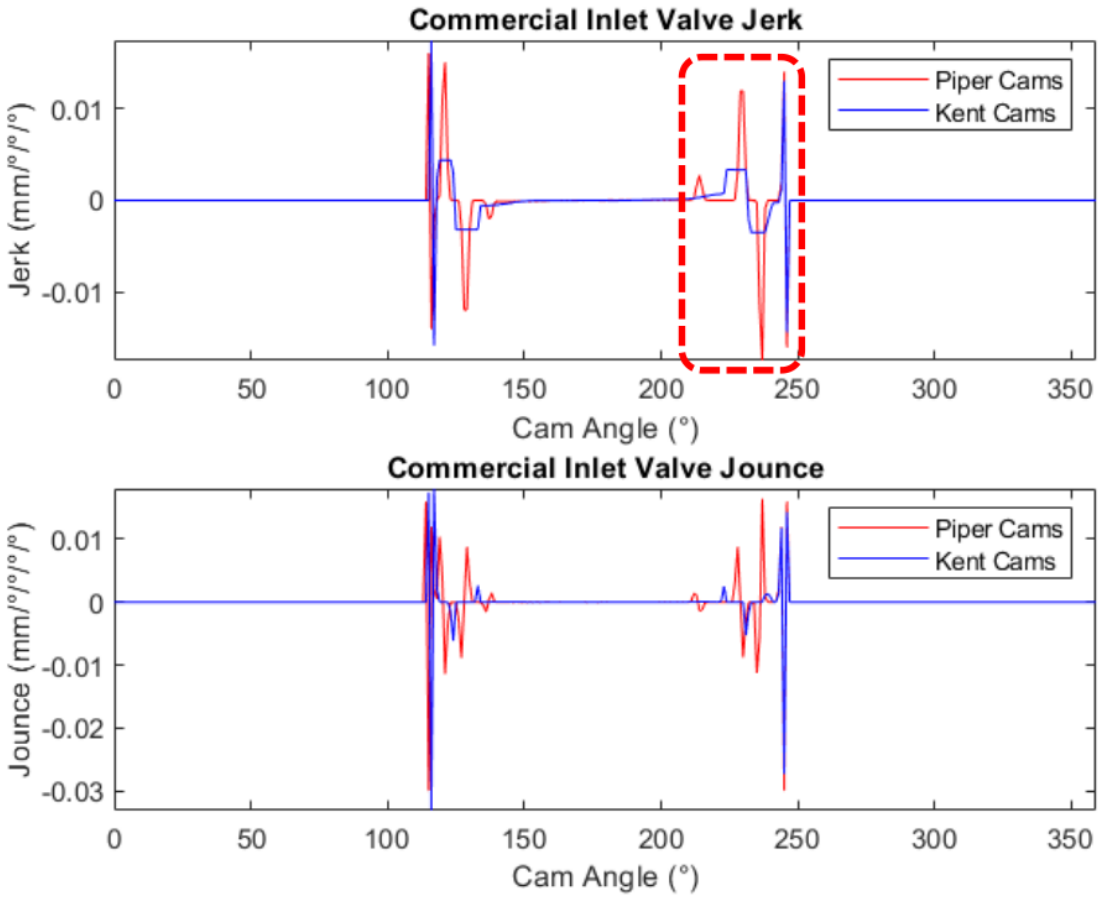


Figure 10.3 Comparison between Kent Cams and Piper Cams inlet valve jerk and jounce.

Piper cams can be seen to generate greater jerk and jounce responses as a result of the aggressive changes in valve velocity and acceleration. However, these do remain finite in magnitude, proving no jump discontinuities exist in the earlier derivatives. Despite these additional peaks, Kent Cams achieves the greatest in magnitude overall.

(Hollingsworth, P. and Hodges, 1991) suggests an acceptable jerk value for a direct contact configuration is between *0.01016* and *0.02032 mm/˚/˚/˚*. **Table 7** presents the jerk data from both manufacturers.

Table 7 Comparison of commercial jerk values.

	Piper Cams (mm/˚/˚/˚)	**Kent Cams (mm/˚/˚/˚)**	**Percentage Difference (%)**
Peak Jerk	0.0160	0.0174	8.7

10.2 Validation of Valvetrain Characteristics

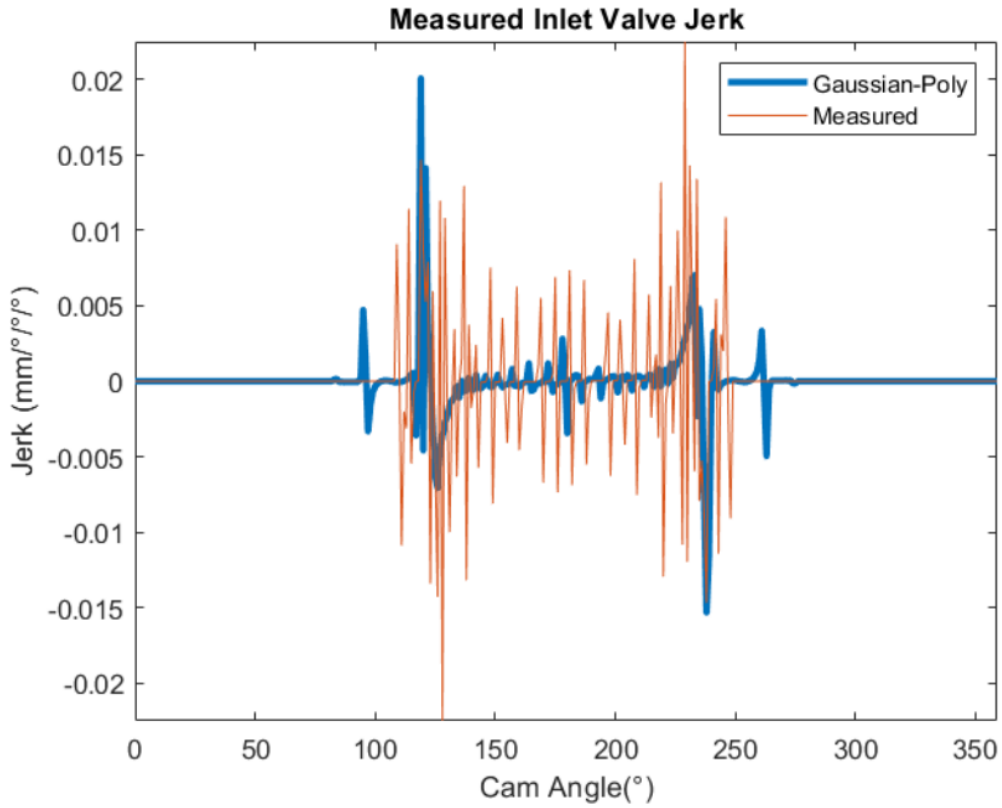


Figure 10.4 Comparison between Kent Cams and Piper Cams inlet valve jerk and jounce.

The processed peak jerk is *0.0201 mm/˚/˚/˚*, translating to *16%* higher than the averaged commercial value. Whilst this appears somewhat inaccurate, it still falls within the sensible range proposed by (Hollingsworth, P. and Hodges, 1991). The significant reduction in noise over the raw measurement data can be further appreciated here, yet a distinct difference still exists between the improved and commercial trends.

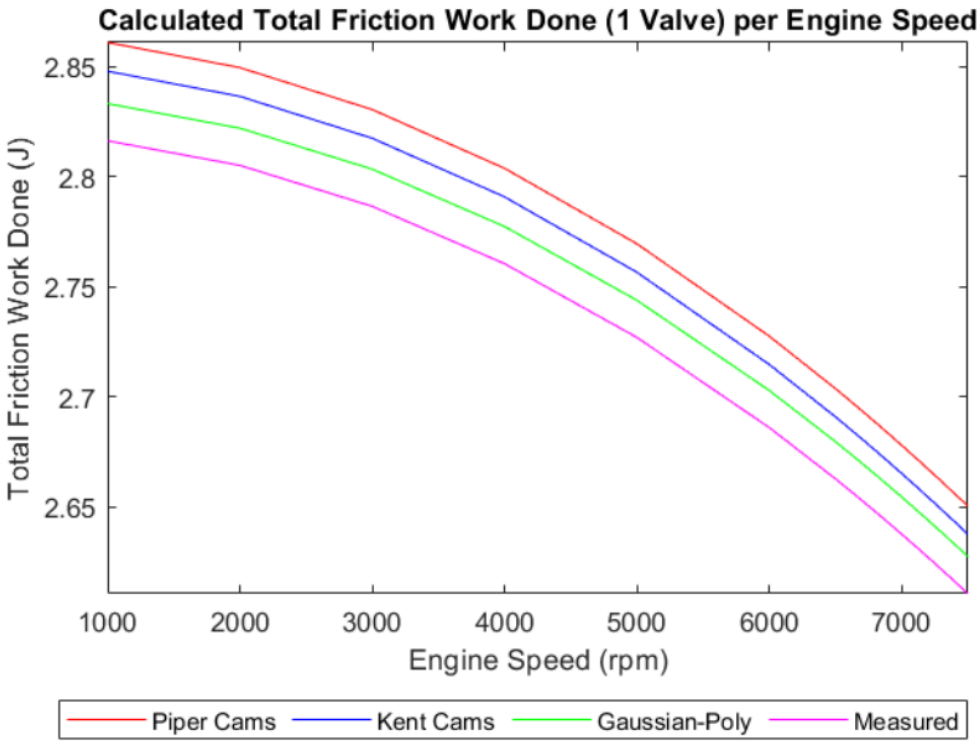


Figure 10.5 Comparison of total friction work done, between commercial and measured data.

The varying motion profiles can be seen to influence the friction work done by one valve across the engine speed range. However, the magnitude of change is relatively small when compared to those found in valve float speeds. The Gaussian-Poly method remains the closest matched to the commercial data.

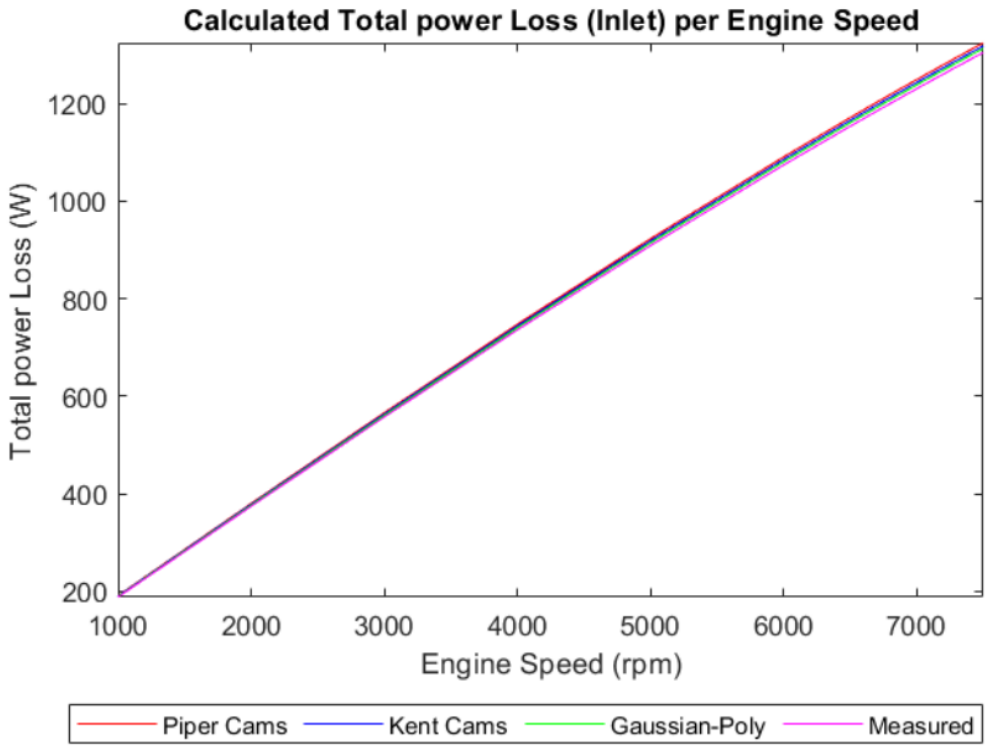


Figure 10.6 Comparison of total power loss, between commercial and measured data.

The total power loss due to friction for the inlet side of the valvetrain is approximately *1.3kW @ 7500rpm,* for all tested valve profiles. *Figure 10.6* concludes the increased accuracy of the Gaussian-Poly has no major effect on the total power loss when motion profiles are the only changing variables applied through the friction model.

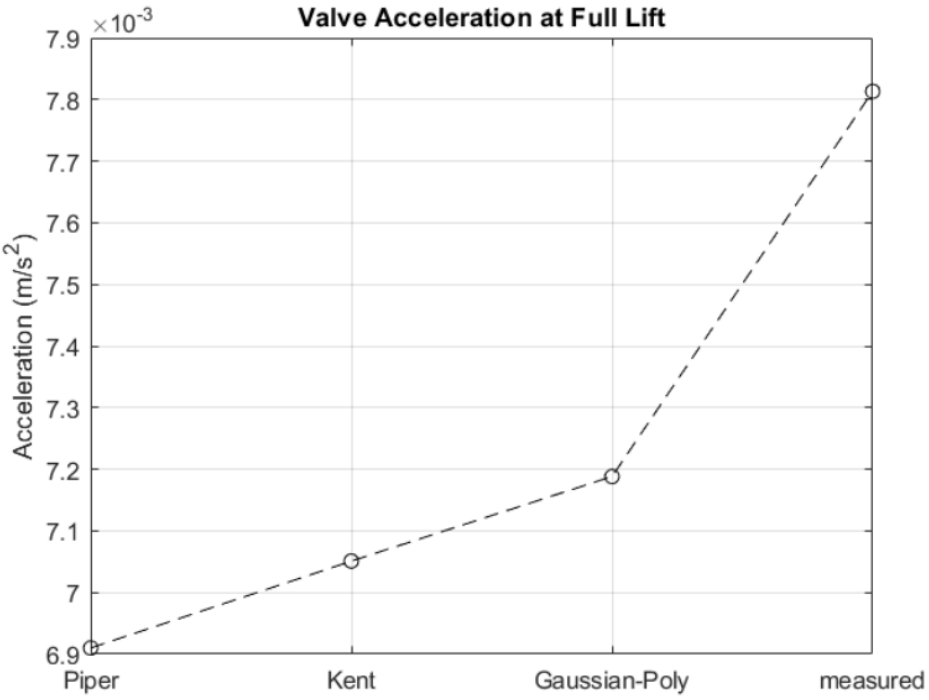


Figure 10.7 Acceleration at full lift for commercial and Gaussian-Poly (3600 x 3600), measured (360x360).

The combination of smoothing (Gaussian-Polynomial method) and increasing the valve lift matrix size can be seen to significantly improve the accuracy of predicting valve acceleration at full lift, over the original measurement data. The Gaussian-Poly method is within *2%* of the Kent Cams result, whereas the measured data is *10.8%* out.

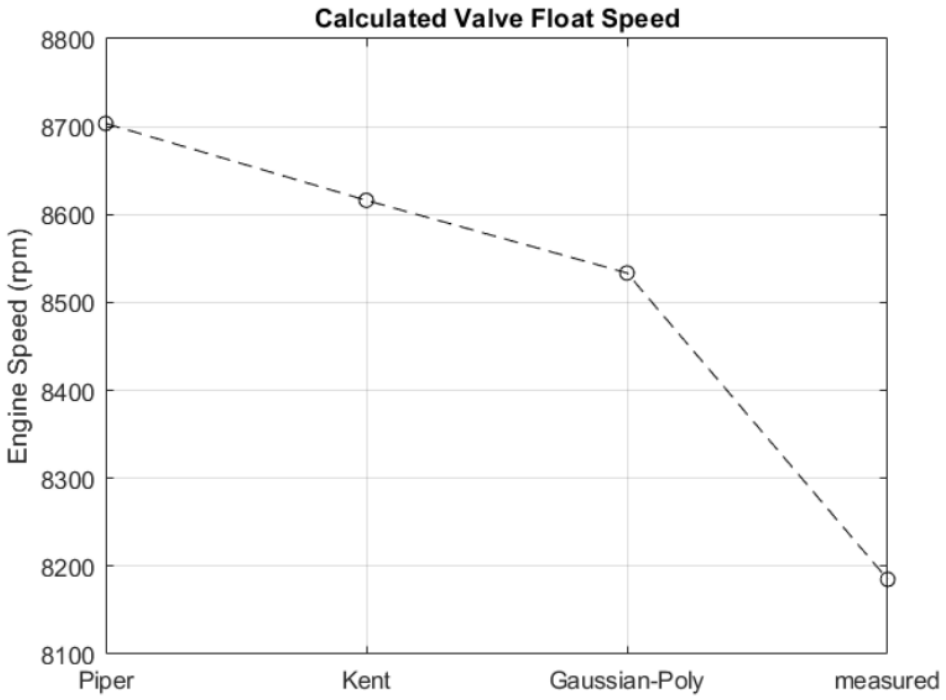


Figure 10.8 Valve float speed for commercial and Gaussian-Poly (3600 x 3600), measured (360x360).

The valve float speeds were compared for each method with the corresponding valve acceleration at maximum lift being the only changing variable. **FIGURE 50** shows a *431rpm* difference between the results from the original measurement and Kent Cams. The Gaussian-Poly predicted valve float speed falls within *82rpm (0.95%)* of the commercially produced data.

10.3 Valve Float Speed Study

This subsection will establish a prediction for the engine speed in which the valve loses contact with the closing phase of the cam lobe profile, followed by a comparison of optimised float speeds derived from the raw measured valve lift (360 x 360 matrix) and the Gaussian-Poly (3,600 x 3,600 matrix).

A visual guide to the working range of the linear valve spring can be seen in *table 8*, where spring preload and maximum load can be found.

Table 8 Spring test data including seated and full lift compression distances.

Spring Test Data

Distance		Load		
mm	*m*	*Lb*	*N*	
1	0.001	9.8	43.59257	
2	0.002	18.2	80.95763	
3	0.003	27.2	120.9916	
4	0.004	35.4	157.467	*Seated Comp Dist*
5	0.005	43.8	194.8321	4.1
6	0.006	52.6	233.9765	
7	0.007	61.4	273.1208	
8	0.008	70.2	312.2652	
9	0.009	79	351.4095	
10	0.01	87.8	390.5539	
11	0.011	97	431.4775	
12	0.012	106.4	473.2908	
13	0.013	115.6	514.2144	*Max Valve Lift*
14	0.014	125	556.0277	8.9
15	0.015	134.2	596.9513	

Previously determined values were substituted into the following equation to find the estimated engine speed that valve float will occur.

$$RPM_f = \sqrt{\frac{Sl_{max} \times 1000}{MM \times A_{fl} \times 36}} \times 2$$

Equation 14 Valve float speed.

Table 9 compares the resultant float speeds to the MCR race car. The findings suggest potential for lowering the valve spring stiffness to reduce resistive forces in the valvetrain, for all attempted methods. The size of the valve lift matrix has the greatest effect on the float speed calculations, with the smoothed data also contributing to change.

The Gaussian-Poly (3600 x 3600) method gives over *1000rpm* clearance before valve float occurs.

Table 9 Comparison of valve float speeds calculated from all variations of valve lift matrix.

Method	Calculated Float Speed (rpm)	Maximum Engine Speed [MCR] (rpm)	Engine Speed Difference (rpm)	Percentage Difference (%)
Measured				
360 x 360	8,185	7512	673	8.22
720 x 720	8,190		678	8.28
3,600 x 3,600	8,427		915	10.86
Gaussian-Polynomial				
360 x 360	8,182	7512	670	8.19
720 x 720	8,183		671	8.2
3,600 x 3,600	8,534		1,021	11.97

The results from *figure 10.8* were converted in terms of a theoretically optimised valve spring stiffness, to give real context for the potential reduction in cam-to-tappet power loss (friction). This was achieved by transposing *equation 14* for maximum compressive force then back calculating the equivalent spring rate to include into a revised friction model.

Table 10 shows a comparison for all results.

Table 10 Optimised valve float speeds and resultant power losses for commercial and measured data.

Method	Calculated Float Speed (rpm)	Optimised Float Speed (rpm)	Percentage Difference (%)	Original Spring Rate (N/m)	Optimised Spring Rate (N/m)	Percentage Difference (%)	Original Power Loss (kW)	Optimised Power Loss (kW)	Percentage Difference (%)
Piper Cams	8,704	7,600	12.68	39,481	30,154	23.62	1.35	0.99	26.67
Kent Cams	8,616		11.79		30,770	22.06		1.00	25.93
Gaussian-Poly	8,534		10.94		31,385	20.51		1.05	22.22
Measured	8,185		7.15		34,077	13.69		1.15	14.81

Optimised float speed was determined by the maximum engine speed (aggressive downshift) plus approximately 100 rpm for an element of safety. The calculations prove the value of smooth cam measurement data when developing the efficiency of the valvetrain. A theoretical power saving of 20% can be made by a similar reduction in spring stiffness for the inlet side.

10.4 Comparison Summary

Of the two commercial profiles, Kent Cams is most representative of the measured cam. Piper's interpretation of the OEM lobe includes areas of constant velocity and constant acceleration (*figure 10.2*). This is a very subtle difference in cam profile, but the more aggressive acceleration can be advantageous over smooth motion in some applications (Hollingsworth, P. and Hodges, 1991).

Adjusting the friction model based on the optimised valve float calculation has a significant effect on the resultant power loss (due to changing valve spring properties accordingly) when compared to only changing the motion profiles.

11. Harmonic Analysis

Valve motion naturally conforms to Dirichlet's conditions (*Section1.10.1*) through camshaft rotation. This phenomenon is considered a repeating function of period 2π, containing 360 continuous data samples (each degree of camshaft cycle, θ) and can be represented by the following integral equation.

$$\int_0^{2\pi} f(\theta) \ d\theta$$

A Discrete Fourier Series (DFS) of trigonometric equations can be employed to synthesise the valve displacement through one full rotation of camshaft.

Valve Lift Function as trigonometric DFS (derived from *equation 1*),

$$\int_0^{2\pi} f(\theta) \ d\theta = \int_0^{2\pi} \left(a_0 + \sum_{n=1}^{36} a_n \cos(\theta) + b_n \sin(\theta) \right) d\theta$$

A Matlab script (*appendix*) was produced to compute and present the Fourier series for the Duratec valve displacement. The Kent Cams profile was used as a commercial benchmark for the process.

Figure 11.1 demonstrates a good match across the main duration of the valve lift as low as the 11^{th} order. As the corruptions caused by constant velocity ramps can be seen to still have a large effect on overall accuracy at this point, it was necessary to continue expanding the Fourier series until full convergence was attained.

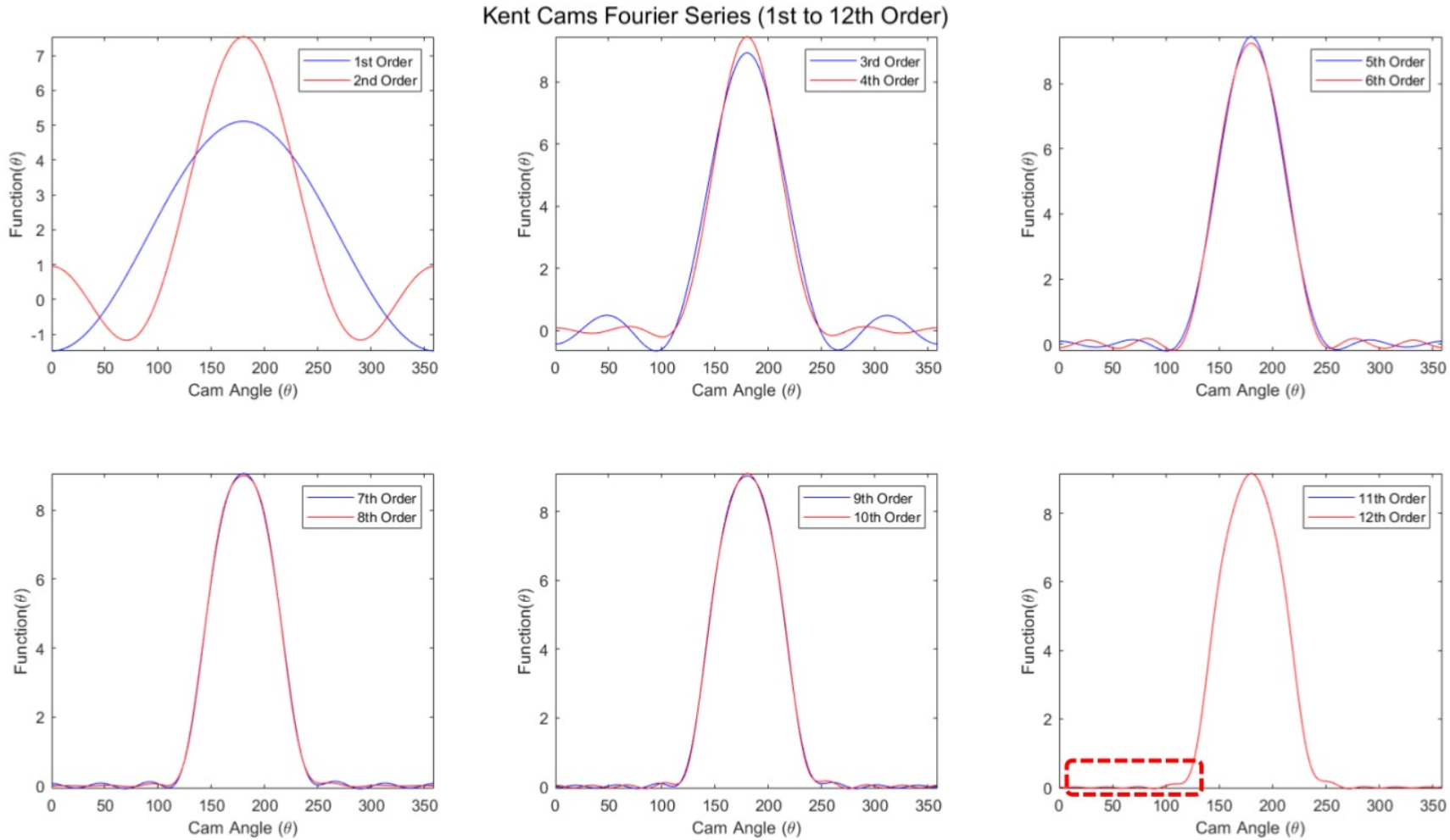


Figure 11.1 - Kent Cams Fourier series (1st to 12th order)

This study satisfies the observation made by (Blair, 2020), where full convergence tends to occur around the 36th order for valve lift in a Fourier series. *Figure 11.2* proves this, with further evidence displayed in *appendix.*

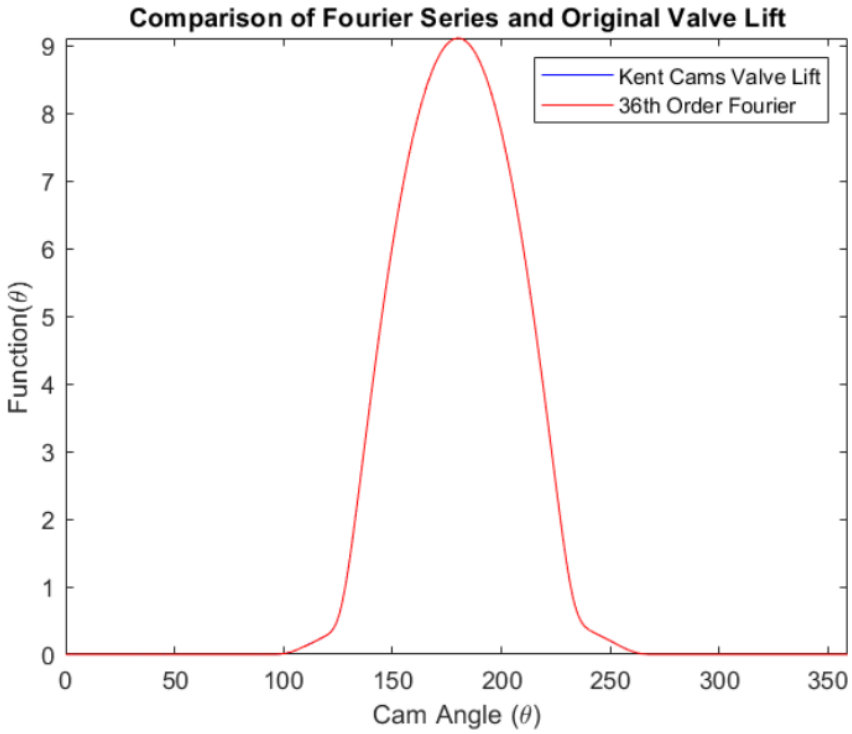


Figure 11.2 36th order Fourier series from Kent Cams valve lift data.

A closer look at the constituent parts of the synthesised function shows how the sine and cosine waveforms are used to construct the lift profile. The natural phase induced by valve lift being symmetrical about π *radians* means cosine signals are dominant in this application. A Fourier Series (of 36th order) was generated from the cosine (a coefficients) alone to prove this as can be seen in *figure 11.3*. The divergence (highlighted) around the constant velocity ramps show where the sine (b coefficients) take effect.

Figure 11.4 includes the sine series of the function. The inverse peaks (highlighted) can be seen to influence the region (cosine divergence), with a magnitude significantly less than the dominant cosine series.

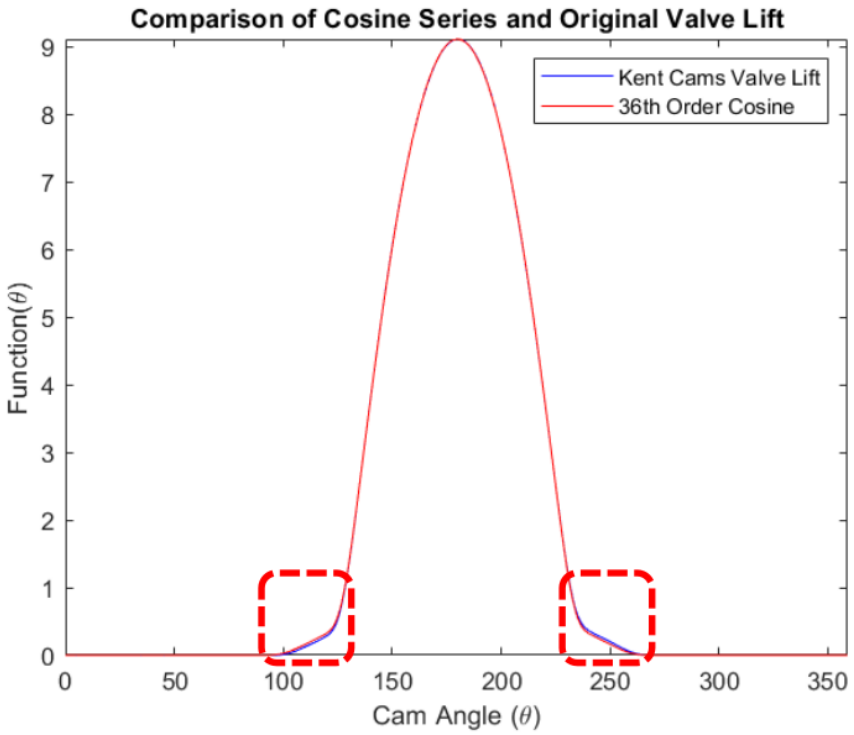


Figure 11.3 36th order Cosine series from Kent Cams valve lift data.

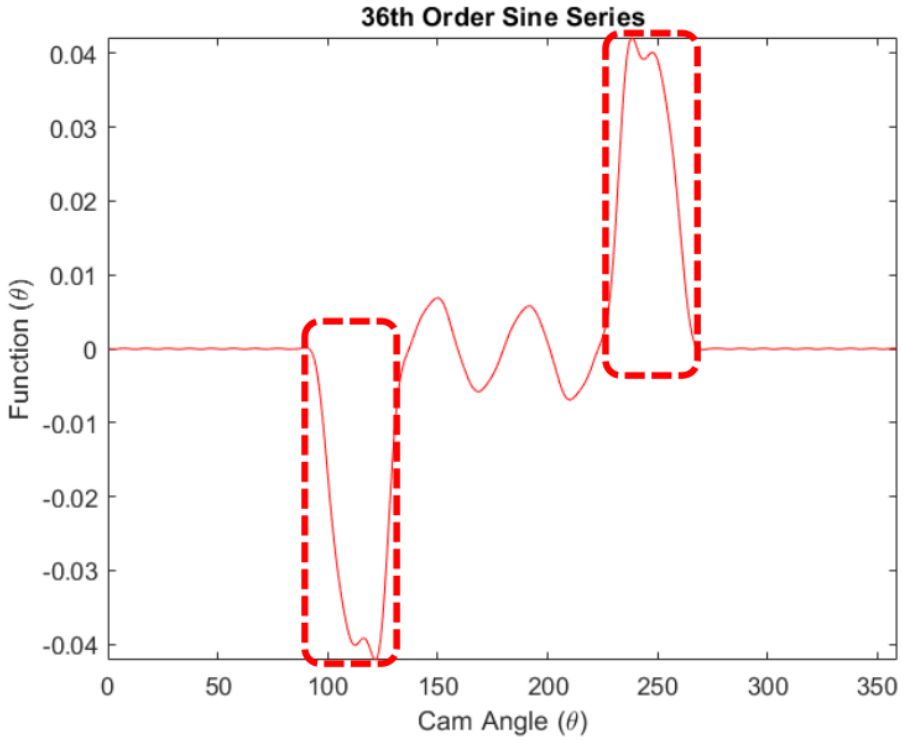


Figure 11.4 36th order Cosine series from Kent Cams valve lift data.

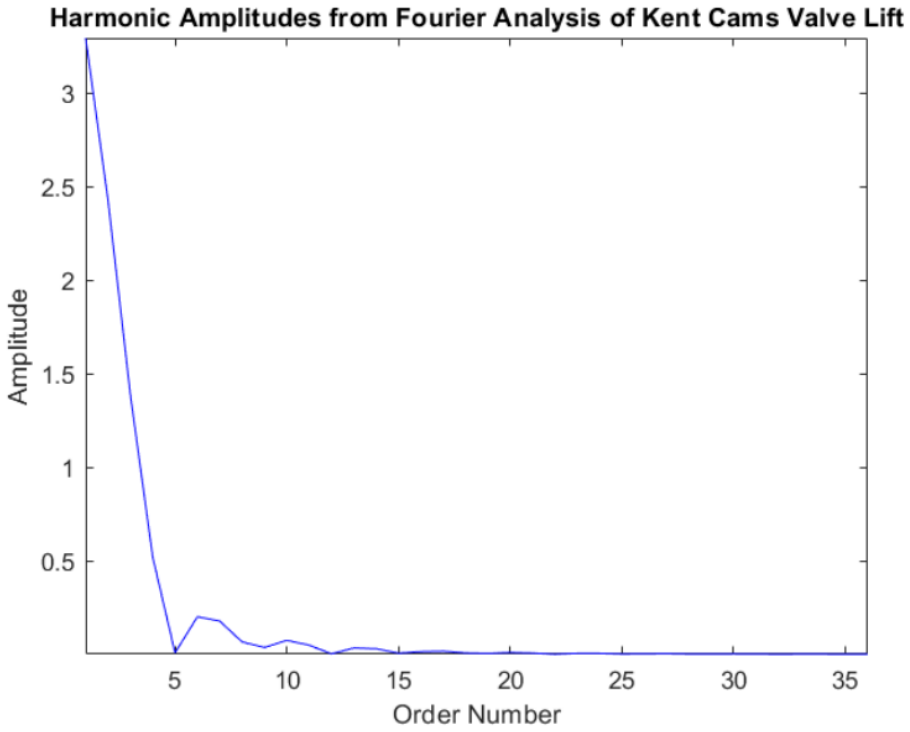


Figure 11.5 Harmonic amplitudes from Fourier analysis of Kent cams valve lift.

Figure 11.5 demonstrates how fundamental frequencies (1st order) can severely increase the excitation of resonance. The amplitude tends to decrease with harmonics of higher order.

The amplitude is influenced by the rate of convergence through the Fourier series, as lower harmonics have an increased impact from the weight of the Fourier coefficients.

This aspect of valvetrain analysis is critical for considerations to valve spring design. The impact of such phenomenon is discussed in *section 11.1*.

11.1 Spring Surge Frequency

The method for obtaining spring surge frequency is determined by variables relating to physical spring properties and spring end conditions. For this application, the spring is fixed at both ends by the cylinder head and valve spring retaining cap. The cap end of the spring does translate with valve motion but is not free to move in harmonic terms. For these conditions, (R G Budynas, 2012) and (GUPTA, 2005) support the following equation in helical compression springs.

$$S_s = \frac{1}{2} \times \sqrt{\frac{k}{M_s}}$$

Equation 15 Spring surge frequency

The calculated 1st resonant frequency for the valve spring is *533.33 Hz*. (GUPTA, 2005) recommends the natural frequency of the spring should be at least twenty times the frequency of application to prevent resonance with harmonic frequencies up to 20th order. Conversely, engine specific valvetrain papers by (R C Juvinall, 2017) and (Schamel *et al.*, 1993) indicate using a factor of 13 is sufficient.

The normal engine speed operating range for the MCR race car is between *5,500-7,200 rpm*.

$$Order_n = \frac{S_s}{V_c/60}$$

Equation 16 Harmonic order

Solving for the limits of engine speed range gives an order window of *9th to 12th* harmonic. From this, it is possible to cross examine the harmonic amplitudes produced by the valve motion, to determine the order in which greatest excitation is present. A comparative version of *figure 11.5* was magnified to display this.

Figure 11.6 shows a close correlation between the commercial camshaft manufacturers. Both exhibit a drop amplitude at 9th order. The Gaussian-Polynomial function shares a similar trend owing to the closely matched valve lift profile but has some inconsistencies.

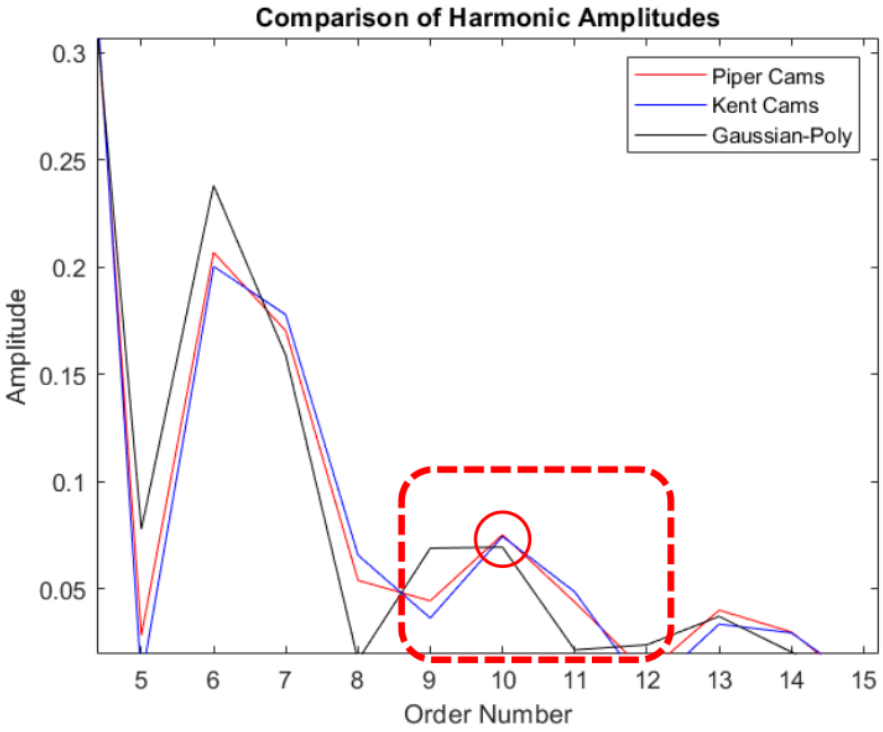


Figure 11.6 Comparison of harmonic amplitudes from all methods, highlighted engine speed range.

Of the working engine speed range, 10th order expresses greatest amplitude for all profiles. The engine speed at this point is found by transposing *equation 16 – 6,396 rpm*.

The decline in amplitudes at maximum engine speed (9th order) will be factored into the design of the commercial valve lift profiles. The exact engine speed that falls at 9th order is *7,106 rpm.*

11.1.1 Spring Surge with Reduced Valve Spring Stiffness

The valve float speed study conducted in *Section 10.3* proved a reduction in cam-to-tappet friction (power loss) could be achieved with a drop in spring stiffness. It was found that lowering the spring constant from *39,481* to *31,000 N/m* could see a power saving of approximately 20%. This translates to a lower 1st mode of spring surge, calculated at *473 Hz*. A consequence of the softer spring would be a shift in the harmonic window, from 9th order to 8th order, for maximum engine speed.

11.2 3D Plot for Spring Surge Analysis

A Matlab script was used to generate a 3D plot to show the relationship between engine speed, order number and resultant frequency. *Figure 11.7* shows the OEM Duratec spring surge frequency (red band) for each harmonic order through the specified camshaft speed range.

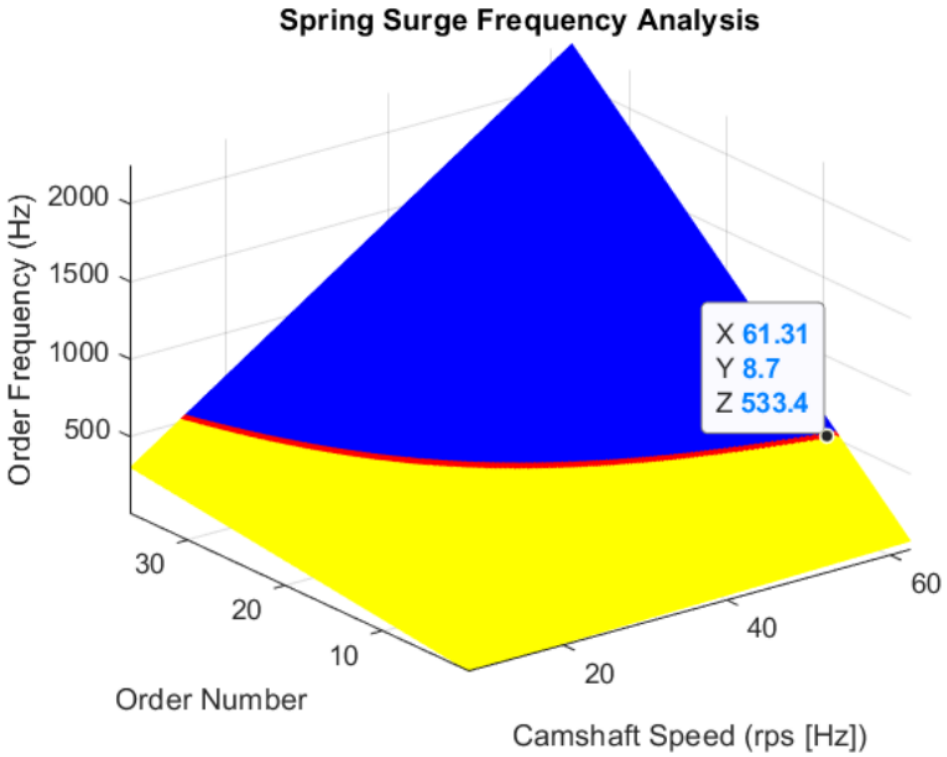


Figure 11.7 Spring surge frequency analysis.

This visualisation of spring surge frequency proves how lower engine speeds are unlikely to excite resonance, as higher order harmonics contain weaker amplitudes.

12. Discussion of Findings

12.1 Measuring & Post Processing of Camshaft Lift Profile

Whilst best efforts were made to ensure every measurement sample was meticulously collected (free of rounding errors and misalignments), the resulting profile showed fundamental flaws in the elected technique.

At first glance, the raw lift data appeared entirely legible (minus the inherent run-out error which was minimised by resetting the base circle and mirroring the second half of the data set). No counts of outlying samples or deviations between each side of the profile could be detected by viewing the overall cam lift plot. The faults manifested with an increased order of derivatives (velocity, acceleration, jerk), creating uncertainty for the accuracy of dependant calculations.

Section 6 encompasses the primary concerns around the selected measuring procedure, with magnified plots for each lobe section. The evidence suggested limitations of physical test apparatus and a level of incompatibility between the measuring resolution of the dividing head and DTI. It was deemed impractical to obtain devices of higher fidelity, such as those made available by (CGKGage, 2020) and (Adcole, 2020), so a solution was sought to repair the undesirable noise through post processing methods. Failed attempts of filtering and smoothing functions within Matlab led to the use of the programs powerful curve fitting tool. The Gaussian (bell) curve was found to most closely replicate the trend of the lift profile, but failed to model the linear clearance ramps, even with an increased number of terms.

It was understood that camshaft lobes can take the form of “blended curves”, where more than one curve or curve type can be employed to form the final contour. Research into this area of cam design revealed how these curves are driven by specific design criteria and these engine-specific details were unobtainable for the Duratec. Although the original contour design process could not be replicated easily, the research informed a decision to include a polynomial curve by using local interpolation to blend between each curve type. The outcome was a significant reduction in 2^{nd} derivative cam noise.

12.2 Conversions Between Camshaft and Valve Profiles

The cam lift data was passed through a valve lift matrix, where values were obtained for each degree of cam rotation angle, as detailed in *Section 4*. The resultant valve velocity and acceleration data contained consistent noise across the duration of lift. The accuracy of subsequent calculations that relied on aspects of valve motion became uncertain from this phenomenon, prompting an effort to reduce the noise by refining the matrix method to a higher resolution. The processing power of Matlab enabled matrices of considerable size to be investigated. This was realised by creating a loop to expand the original vector (360 points) with user defined spacing, then interpolating to find estimated values between each degree of original cam measurement.

A Brief description for each considered interpolation method:

- *Nearest* – Nearest non-missing value.
- *Linear* – Interpolation of neighbouring, non-missing values.
- *PCHIP* – Shape-preserving piecewise cubic Hermite spline interpolation.
- *Spline* – Piecewise cubic spline interpolation.

Both the nearest and linear functions were rejected due to incompatibility (interpolating with existing values). The PCHIP interpolation method was selected with the expectation of preserving the shape of the curve-fitted input through to the resolved valve lift vector. (Borboni *et al.*, 2020) discusses the use of interpolation to reconstruct continuous motion profiles from a discrete set of data and gives context to cam mechanisms. The authors of the research paper explain how Hermite interpolation assures continuity of a function and its derivative by adopting cubic splines. A smoother profile was eventually achieved using the Matlab spline (non-shape-preserving) function, which can be seen in *appendix*.

The largest matrix in the study contained approximately 1.3 billion data points, with the most effective matrix using around 13 million. Both yielded identical acceleration trends, implying the model had fully converged somewhere between "*720 x 720*" and "*3,600 x 3,600* ". The previously mentioned cam measuring machine by (Adcole, 2020) was advertised to measure at every 0.1˚ cam angle, the equivalent resolution of the fully converged matrix. Whether this is significant to all cam profiles (due to such possibilities as typical manufacturing tolerances or sensitivity limits from valve motion design parameters) remained undetermined in this investigation.

A peak can be seen to occur in acceleration at the nose of the valve, reducing with increased matrix size. However, this feature persisted, even with the curve fitted models. This phenomenon was comparable to that found in *figure 12.1*. The study by (Turkish, 1950) shows a similar occurrence, where an accurately ground cam lobe was measured in a lathe (approx. 200rpm) and the tappet follower was held against the cam by an oscilloscope to simulate actual valve motion. Whilst this is a dated example, it does suggest larger jumps in measurement values can exist about the tip of the nose, causing this local spike in rate-of-change. The curve fitted model carried this fault through, due to converging with the symmetry that existed in the original data at this section.

The significance of this phenomenon is the potential impact on estimated valve float speed, which uses the acceleration at peak valve lift to calculate. This investigation took the "absolute minimum" acceleration value, which agreed with the nose by $\pm 10^{\circ}$ cam angle for all profiles.

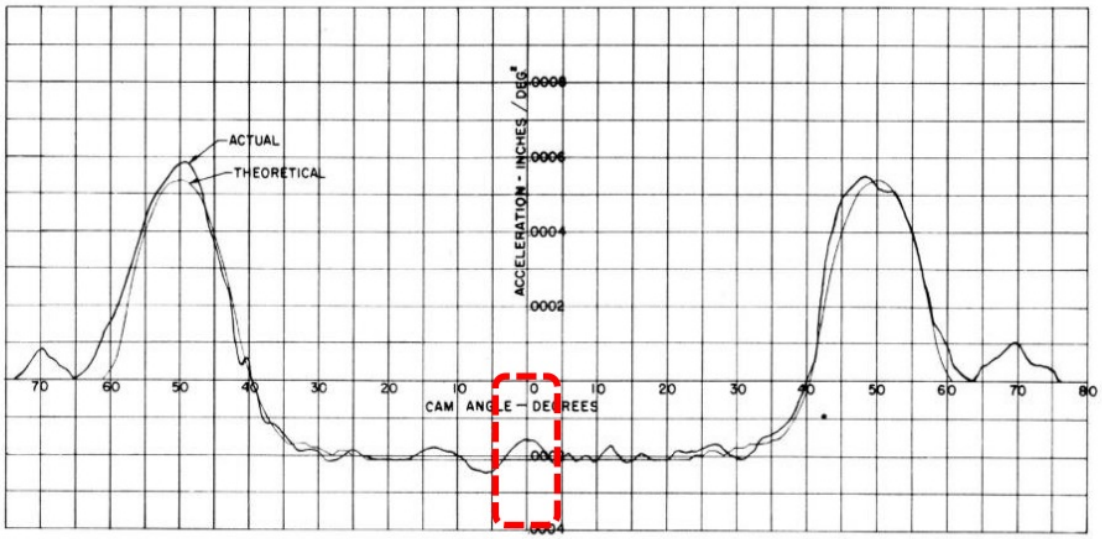


Figure 12.1 Example of valve acceleration, derived from measuring cam in lathe (Turkish, 1950).

Overlaying the commercial data in *figure 12.2* shows how an asymmetric trend is present around the nose of the Kent cam profile, with "absolute minimum" acceleration occurring just past the nose of the cam lobe. Piper cams demonstrates a more symmetrical approach at this point. However, the actual acceleration values at peak lift converge between commercial profiles, supporting a match in principle design parameters, as proposed by (Qiu, 2005).

The subtleties in OEM specification profiles provided by both commercial camshaft companies show how interpretations in measuring and processing methods can influence the resulting valvetrain motion. For this reason, the methods of valve lift approximation in this study were deemed sufficiently accurate, given the quality of the foundation camshaft measurement data.

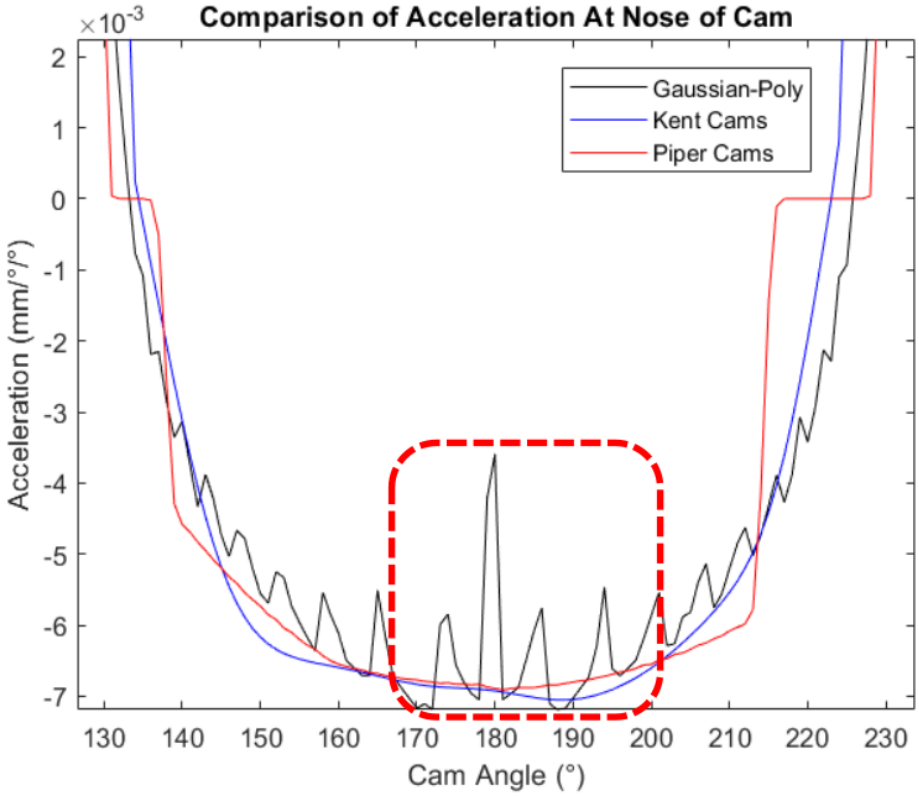


Figure 12.2 Comparison of valve accelerations for each profile, magnified at the nose of the cam.

It became clear through researching the papers by (Qiu, 2005), (Sahu, Kedia and Sahu, 2016), (Turkish, 1950) and conversations with professional representatives at Autosport 2020 that valve lift is interchangeably referred to as cam lift. This link suggests the camshaft lobe is a consequence of valve motion design, rather than valve motion being a function of pure cam design. This approach led to the use of the rotation of axes method, where the valve displacement profile was converted to a cam profile, giving opportunity to compare the impact of subtle valve motion properties on lobe geometry. The adopted method is considered accurate for direct acting cams that use a flat follower. A close inspection of the resultant cam lobes revealed no indications of noise being introduced through the conversion process like those found through the matrix method. This technique was verified by converting the original lobe measurement data to polar form and comparing against the rotation of axes output.

Figure 12.3 highlights the difference between both commercial profiles and the measurement data. This contextualises the errors introduced by the elected DTI measurement procedure. The largest difference in flank offset (between commercial and measured) was calculated to be approximately *0.8mm,* with the difference in peak lift being confirmed as *0.02mm*.

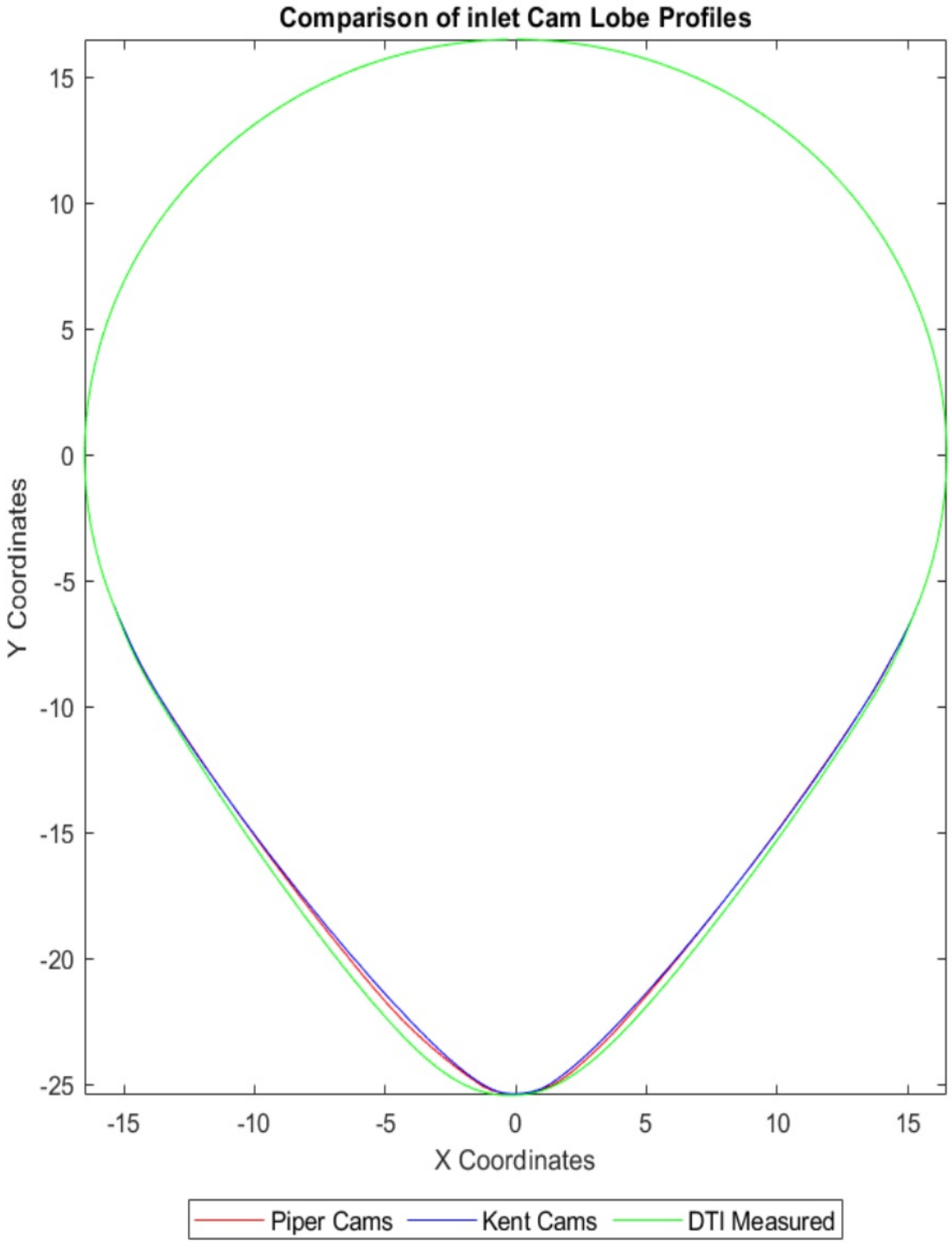


Figure 12.3 Inlet cam lobe profiles generated from rotation of axes conversion method.

12.3 Valvetrain Friction

A study was conducted to investigate the greatest contributor to valvetrain friction, the cam-to-tappet interface (*section 7*). It includes reference to the coefficient of friction (μ) changing with engine speed and oil temperature (Okarmus and Keribar, 2015). These are dynamic variables that are best found through empirical methods, proving difficult to predict mathematically. The paper by (Okarmus and Keribar, 2015) includes further complications for predictive models. Values for minimum and central film thickness are required for elastohydrodynamic conditions and for more accurate predictions. The flash temperatures of sliding friction would also have to be considered for temperature rise due to frictional heating. In addition, suitable asperity calculations (such as Greenwood-Tripp model) rely on assumptions of asperity peaks, which can deviate significantly from actual surface roughness.

For these justifications, it was not considered viable to accurately replicate the specific tribological conditions of the Duratec engine mathematically.

Instead, an approximate calculation was formed to predict contact forces present, by considering the combination of valvetrain inertia and valve spring force. The additional pressure exerted on the valve face from latent cylinder pressures through each valve event have not been included to this calculation. The resultant contact force was combined with a constant coefficient of friction to establish the friction torque, friction force, total work done and total power loss. Whilst this is a simplified model, it provided sensible results that represent the system type. Papers by (Okarmus and Keribar, 2015) and (Calabretta and Cacciatore, 2010) show similar trends. The main advantage of using this simplified method was to provide a foundation for comparative analysis between the commercial and measured data sets. This proved particularly useful when conducting the valve float speed analysis, to determine potential reductions in frictional power loss.

Disparity in acceleration through the valve closing event existed between the measured and commercial data as shown in *figure 12.4*. This took the form of a minor deceleration after initial opening point which also persisted through the attempted smoothing methods, and a slight offset in initial acceleration points. These were not accounted for due to only having a minor effect the calculations of friction values through contact force. The anomalies would have been more carefully considered if the correlation between both commercial profiles was more tightly formed.

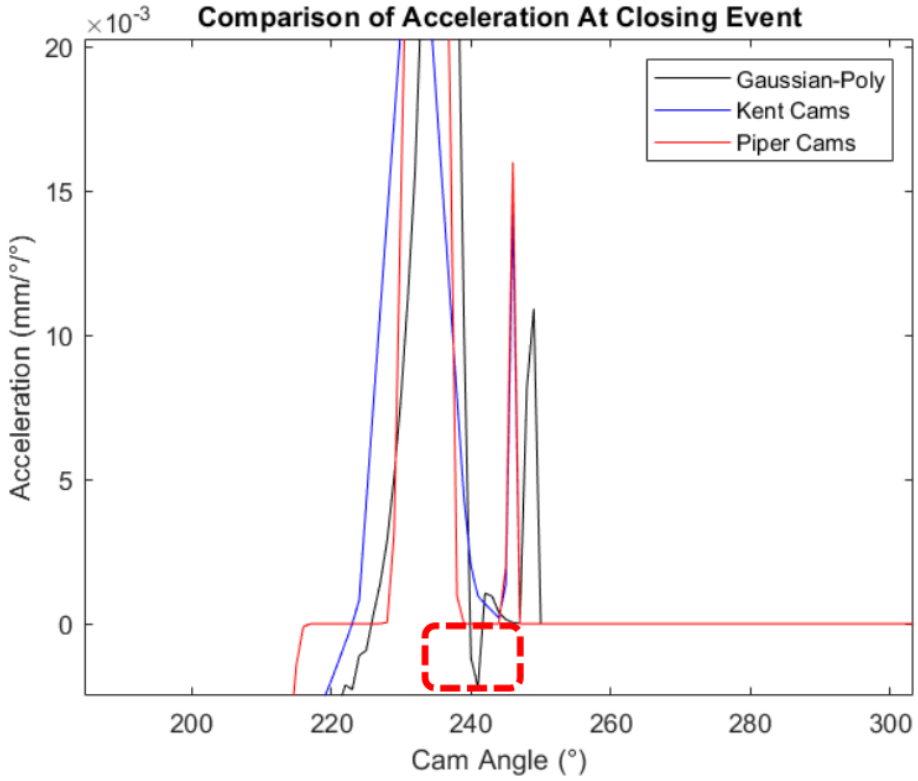


Figure 12.4 Comparison of acceleration profiles magnified to valve closing point.

The two remaining valvetrain friction interfaces (tappet-to-bore and camshaft bearings) were not considered for this investigation, as they are significantly less involved in the overall contribution to power loss, particularly at the higher speeds of the MCR Race Car.

The Hertzian contact stress calculations for the cam-to-tappet conjunction are in *appendix*. (Dobrenizki *et al.*, 2016) recommends a maximum contact pressure between *400-1200 GPa* for a direct acting valvetrain, with peaks predicted to form at ±20˚ cam angle. (Beard and Hempson, 1962) considers the effects of surface finish, hardness and boundary condition, although, the Hertzian stress calculation can support camshaft engineers with initial designs (e.g. choosing lobe length). Contact relationships have traditionally been developed experimentally, but more recent methods include the Fourier transform to establish the elastic response from each harmonic of contact pressure (M. Teodorescu, 2004). Finally, it is conceivable to engineer cams with varying layer thickness (layer thickness changes with cycle) to optimise performance, as proved by (Rahnejat, 2007).

12.4 Harmonic Analysis and Valve Spring Surge

Section 11 contains the methodology and findings from applying the Fourier series to assess valvetrain harmonics. The spring surge characteristics were calculated as it is the valvetrain component with the lowest stiffness (greatest elasticity) by far, consequently, it is most susceptible to reach its lowest mode of surge. The spring constant was used to establish a stiffness ratio for the appropriate equation. The linearity of the Duratec valve spring enables the rate to be used as opposed to a fixed value, which is more precise as the natural frequencies of the valve spring change significantly with valve displacement (spring compression). (Liu and Kim, 2011) present experimental methods for determining the damping ratio by the half power bandwidth method. *Figure 12.5* demonstrates a typical constant pitch valve spring, such as those fitted to the Duratec, with a damping ratio remaining stable until a sudden increase at high compression. The damping ratio was not established for this investigation, but intuition suggests the Duratec valve spring is more vulnerable to resonance caused by externally applied excitations when the valve is close to its seat (low compression state), where more active coils and less damping occurs.

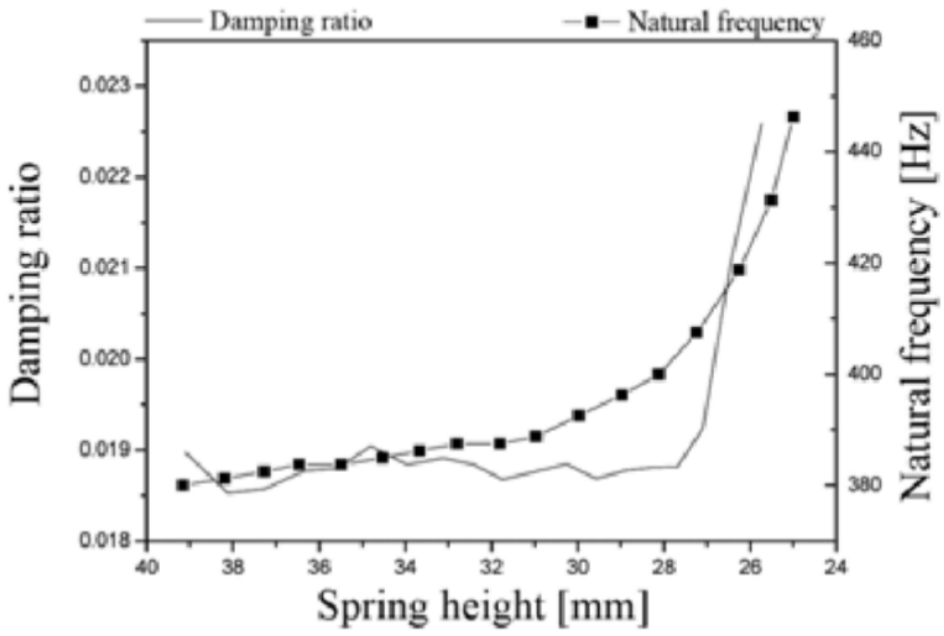


Figure 12.5 Compression spring natural frequency and damping ratio comparison (Liu and Kim, 2011).

(Phlips, Schamel and Meyer, 1989) Prove the frequency of the camshaft (controlling valve displacement) is responsible for the excitation of the valve spring rather than other components in the valvetrain system. Typically, the DOHC configuration is considered stiff, when compared to more traditional formats. This coupled with higher engine speeds results in valvetrain natural frequencies being significantly higher than the first mode of spring surge.

Papers by (Phlips, Schamel and Meyer, 1989), (Liu and Kim, 2011), (Zuck, 2020) and (Turkish, 1953) all mention the importance of valve acceleration on valvetrain natural frequencies. Results of the Fourier analysis for valve acceleration can be seen in *appendix*. Although, the amplitudes were found to be significantly smaller than those of the periodic valve displacement.

Section 11.1.1 revealed a revised spring rate being considered (to reduce friction) for spring surge analysis. This disclosed a shift in harmonic window, with an 8^{th} order harmonic estimated to excite the resonant frequency (*473 Hz)*. A lower harmonic amplitude is seen to act at this point, but a comparably large jump can be seen to occur between 7^{th} and 8^{th} order. Further considerations should be made to validate the findings, such as physical testing and FEA based modal analysis. Exploring the possibility for a change in spring type (from linear to progressive) may also prove valuable.

(Turkish, 1953) explains how the use of dual springs can be beneficial to neutralize the effects of spring surge, and how they also offer the additional safety factor if one of the springs fail. This could be used to improve the frictional qualities of the valvetrain without compromising resonance. Although this solution may incur a weight penalty depending on material selection. (Turkish, 1953) also states how using springs with a variable pitch at either end are another consideration, as these are typically useful between the *9^{th} and 11^{th}* harmonics, the estimated window for the Duratec MCR. These are usually lighter, offering potential for reducing total valvetrain moving mass. All spring types can be developed further by such methods as cyclo-pelting (shot peening) which produces compressive stresses on the surface of the spring, increasing the strength of surface material where fatigue tends to initiate. Another method is “pre-setting” the spring which improves endurance and load-set. This is achieved by compressing the spring beyond yield during the manufacturing process.

13. Conclusion

This report has presented a Matlab based mathematical model for defining critical design characteristics of the Ford Duratec valvetrain. An experimental approach was used for deriving sufficiently accurate valve motion profiles from erroneous measurement data, in the form of a hybrid curve fitting procedure. The final solution (Gaussian-Polynomial) realised the elimination of 2nd derivative camshaft noise whilst maintaining a peak valve lift within *0.02mm* of the original measurement (*figure 6.8*).

The solution comprises,

- 3rd order Gaussian curve for primary structure.
- 9th degree Polynomial curve for clearance ramp definition.
- Piecewise cubic interpolation (PCHIP) method for blending.

The Duratec valve motion was also found experimentally, by investigating the effects of increasing the valve lift matrix size. A study to simulate data points at 0.5˚,0.1˚ and 0.01˚ intervals was conducted by interpolating the original 360 samples. The results clearly demonstrated an improvement in profile resolution, with full convergence at 1.3 million data points (0.1˚) (*figure 5.3*). However, the reintroduction of some 2nd derivative noise was a consequence of the valve lift matrix and attempts to eliminate this were unsuccessful. Analytical evidence suggested the interpolation method had some effect on this phenomenon, as a slight improvement was made by tweaking the piecewise cubic method from a shape-preserving to non-shape-preserving function (*appendix*). The conversion from valve lift to cam lift was found to be precise, as it relies on a system of geometry. This would be the preferred method for obtaining cam lift profiles in future investigations.

A cam-to-tappet friction model was built from the forces acting at the interface (inertia and spring). An assumed constant coefficient of friction of 0.1μ yielded a total power loss of *1.35kW @ 7,500rpm* across the inlet, with the generalisation that total system power loss (including exhaust side) would be approximately *2.5kW*. A theoretical valve float speed study concluded a potential power saving of *20%* with an equivalent reduction in spring stiffness. All measured and experimentally collected results were compared with commercially accepted valve motion profiles obtained from company representatives of Kent Cams and Piper Cams, at Autosport International 2020. A correlation could be seen between the measured and commercial data, being further improved with the Gaussian-Polynomial cam lift profile, achieving a calculated valve float speed within 1% accuracy of the Kent Cams equivalent.

A successful approximation of valve displacement was made using the Fourier series, comprising *36* cam harmonics. The Fourier coefficients were employed to ascertain the amplitude of each harmonic, with close consideration to the MCR engine speed range. It was found that a harmonic window of 9^{th} to 12^{th} order represented these conditions, with the 10^{th} harmonic possessing greatest amplitude (*0.0737*) and occurring at *6,396 rpm*. The Gaussian-Polynomial trend correlated against the commercial profiles, but harmonic amplitudes at 6^{th} 8^{th} 9^{th} and 11^{th} order were proved inaccurate (*figure 11.6*). A visual aid in the form of a 3D plot was created, for relating the spring surge to harmonic number and camshaft frequency (*figure 11.7*). Valve spring analysis revealed the 1^{st} mode of spring surge occurs at a frequency of *533 Hz (OEM)*.

14. Recommended Further Work

- Further analysis into the cause of induced noise through the valve lift matrix. Investigate alternatives to the interpolation methods used and attempt alternative curve fitting methods. Explore the prospect of using Fourier series for smoothing the valve motion profiles as this appears to be an option from analysing velocity and acceleration plots (profiles in *appendix*).

- Use of digital linear potentiometers (or CMM) to measure vertical valve translations to gain more accurate valve lift data. This would bypass the requirement for the cam measuring procedure due to the rotation of axes method outlined in this report. A softer valve spring could be used for ease of rotation and successive samples could be recorded more efficiently; this would improve the accuracy of data by averaging the results over several cycles.

- Explore the creation of basic and synthetic curves such as polynomial, polydyne, N harmonic and splines for replicating valve/cam profiles. This would prove beneficial for valve motion optimisation, where a specific motion criterion could be engineered for performance development.

- Implementation of a powered valvetrain test rig. A comprehensive validation tool for studying motion, forces and tribology. Working examples of this experimentation type can be seen in papers by:

1. (Hanaoka and Fukumura, 1993), where valvetrain noise were analysed using a sound level meter and oscilloscope.
2. (J. M. Pieprzak, P. A. Willermet, 1989), where measurement of tappet rotation and its effects were performed, using light reflectance.
3. (Norton *et al.*, 1998), where displacement and velocity transducers were used with a piezo electric accelerometer to measure valvetrain vibrations.
4. (Pieprzak, Willerrnet and Dailey, 2020), where friction acting at the tappet-to-bore and cam-to-tappet interfaces were measured using a torque meter and a temperature-controlled enclosure for valvetrain components, with strain gauges to measure spring forces.
5. (P Purmer, 1985), where camshaft wear and kinematics were investigated by use of transducers.
6. (Calabretta and Cacciatore, 2010) accurately measured valvetrain friction effects with changing temperatures and engine speeds, with an integrated oil pump and heater. Measurement was taken by a laser system for motion and the electric motor was used to calculate drive torques and (Okarmus and Keribar, 2015) used similar apparatus to measure friction with different oil grades, engine configurations and temperatures.
7. (Liu and Kim, 2011) used a rig that included an oil pump and rotation encoder for the camshaft, this experiment used a strain gauge fixed at 0.5 turns from the top of the valve spring to measure internal spring vibration (from torsional deformation caused by longitudinal motion).

- Surface roughness should be considered for asperity calculations.

- Produce FEA to support any experimental methods, to correlate results and analyse interactions from the proposed accurate CAD model (*appendix*).

- Expand mathematical model to include empirical and FEA predicted values. This could also take the form of an improved Matlab script or application, where a user can input cam or valve data and the software will automatically post process and calculate all required information to consider for valvetrain design and development. Comparable systems include (Lotus, 2020), (Ricardo, 2020) and (CDS, 2020).

15. References

A Gangopadhyay, E. S. (2004) 'Valvetrain friction and wear: Influence of surface engineering and lubricants', *Proceedings of the Institution of Mechanical Engineers, Part J: Journal of Engineering Tribology*, 218(3). doi: https://doi.o.

Adcole (2020) *Crankshaft & Camshaft Measuring Gages.* Available at: https://www.adcole.com/products/gage-types/?Crankshaft-Camshaft-Measuring-Gages (Accessed: 19 April 2020).

Al-jeboori, Y. *et al.* (1990) 'The effect of clearance between tappet insert and camlobe on the tribological and tribochemical performance of cam / follower surfaces', *International Fuels & Lubricants Meeting & Exposition*. Elsevier Ltd, (June). doi: 10.1016/j.triboint.2018.09.021.

Amp Gangopadhyay, D Watt, et al (2007) *Effects of Composition and Surface Finish of Silicon Nitride Tappet Inserts on Valvetrain Friction*. Dearborn, Miami. Available at: https://doi.org/10.1016/S0167-8922(99)80083-2%0A.

Autodata (2020) 'Solera Autodata (Cars & Vans)'. Bracknell: AUTODATA LTD.

B Fabien, R. L. (1994) 'The Design of High-Speed Dwell-Rise-Dwell Tuned Cam Displacement Curves Using Linear Quadratic Optimal Control Theory', *Transactions of the ASME, Journal of Mechanical Design*, 116, pp. 867–874.

Bakonyi, S. M. (1968) *The Advantages of Overhead Camshafts - What Are They?*

Beard, C. A. and Hempson, J. G. G. (1962) 'Problems in valve gear design and instrumentation', *SAE Technical Papers*. doi: 10.4271/620288.

Blair, P. (2020) *Information on the HMB Method of Valve Lift Design*, *Prof. Blair & Associates*. Available at: http://www.profblairandassociates.com/GPB_4stHEAD_ValveLiftDesign_hmb.html (Accessed: 20 April 2020).

Borboni, A. *et al.* (2020) 'Effects of profile interpolation in cam mechanisms', *Mechanism and Machine Theory*. Elsevier Ltd, 144, p. 103652. doi: 10.1016/j.mechmachtheory.2019.103652.

Britannica (2019) 'Resonance (vibration)', *Encyclopaedia Britannica*. Encyclopaedia Britannica, inc. Available at: https://www.britannica.com/science/resonance-vibration#ref188053.

Calabretta, M. and Cacciatore (2010) 'Valvetrain friction - Modeling, analysis and measurement of a high performance engine valvetrain system', *SAE Technical Papers*, (May), pp. 72–84. doi:

10.4271/2010-01-1492.

CDS (2020) 'Camshaft Design System -CDS'. Available at: http://www.cds-valvetrain.com/.

CGKGage (2020) *EZcam Camshaft Measuring Machine*. Available at: http://cgkgage.com/ezcam-camshaft-measuring-machine/ (Accessed: 19 April 2020).

Clauberg, J. and Huber, R. (2013) *Using Non-Smooth Mechanics and Parallelization Techniques for the Efficient Simulation of Different Types of Valve Springs*. doi: 10.4271/2013-01-1119.

Dobrenizki, L. *et al.* (2016) 'Surface & Coatings Technology Ef fi ciency improvement in automobile bucket tappet / camshaft contacts by DLC coatings', *Surface & Coatings Technology*. Elsevier B.V., 308, pp. 360–373. doi: 10.1016/j.surfcoat.2016.09.041.

Dudley, W. . (1948) 'New Methods in Valve Cam Design', *SAE Quarterly Transactions*, 2(1), p. 1948.

GUPTA, R. S. K. J. K. (2005) *A textbook of Machine Design*.

Hanaoka, M. and Fukumura, S. (1993) 'A Study of Valve Train Noises and a Method of Cam Design to Reduce the Noises', pp. 986–996.

Heath, A. R. (1988) *Valve Train Design For Multivalve Automotive Gasoline Engines*, *Simulation*.

Heeps, G. (2018) *SpinTron Secrets: Valve Float, Bounce, and Loft Explained!* Available at: https://blog.trendperform.com/spintron-secrets-valve-float-bounce-and-loft-explained.

Hollingsworth, P. and Hodges, R. . (1991) *The History and Mathematical Development of Cam Profile Design in Rover*.

J. M. Pieprzak, P. A. Willermet, and D. K. (1989) *A Tappet Rotation Monitor Based on Light Reflectance - Development and Functional Testing*.

Johansson, F. S. and B. (1997) *Fluid Flow, Combustion and Efficiency with Early or Late Inlet Valve Closing*. Available at: https://doi-org.ezproxy.uwtsd.ac.uk/10.4271/972937.

KentCams (2020) 'Kent Cams Duratec Valve Lift'. Available at: https://www.kentcams.com/.

Lengvarský, P. and Bocko, J. (2013) 'Theoretical Basis of Modal Analysis', 1(7), pp. 173–179. doi: 10.12691/ajme-1-7-4.

Liu, H. and Kim, D. (2009) 'EFFECTS OF END COILS ON THE NATURAL FREQUENCY OF AUTOMOTIVE ENGINE VALVE SPRINGS', 10(4), pp. 413–420. doi: 10.1007/s12239.

Liu, H. and Kim, D. (2011) 'ESTIMATION OF VALVE SPRING SURGE AMPLITUDE USING THE VARIABLE

NATURAL FREQUENCY AND THE DAMPING RATIO', 12(5), pp. 631–638. doi: 10.1007/s12239.

Lotus (2020) *Lotus Concept Valve Train*. Available at: https://www.lotuscars.com/content/media/LCVT_insert.pdf.

M. Teodorescu, et all (2004) 'Harmonic Analysis to Determine Contact Characteristics of Concentrated Counterformal Contacts', in *6th Biennial ASME Conference Engineering Systems Design and Analysis*. Manchester.

M Chew, F. F. and R. L. (1993) 'Application of Optimal Control Theory to the Synthesis of High-Speed Cam-Follower Systems', *System Optimization Transactions, Journal of Mechanisms*, 105, pp. 576–591.

Morris, S. (2019) *SRCC SPORTS 2000 DURATEC CHAMPIONSHIP SPORTING & TECHNICAL REGULATIONS 2019*.

Nguyen, V. and Kim, D. (2007) 'Mechanism and Machine Theory Flexible cam profile synthesis method using smoothing spline curves', 42, pp. 825–838. doi: 10.1016/j.mechmachtheory.2006.07.005.

Norton, R. L. *et al.* (1998) 'SAE TECHNICAL Analyzing Vibrations in an IC Engine Valve Train', (724).

Nourse, J. H. (1965) *Recent Developments In Cam Profile Measurement and Evaluation*. doi: 10.4324/9780429280696-8.

Nourse, J. H., Dennis, R. C. and Wood, W. M. (1961) 'Recent developments in cam design', *SAE Technical Papers*. doi: 10.4271/610048.

Okarmus, M. M. and Keribar, R. (2015) 'Methodology for Predictive Friction Modeling in Direct-Acting Mechanical Bucket Valvetrain System'. doi: 10.4271/2015-01-0677.

P Purmer, W. van den B. (1985) 'Measurement of Camshaft Wear Wear and Kinematics of Overhead Camshafts'.

Phlips, P. J., Schamel, A. R. and Meyer, J. (1989) 'An efficient model for valvetrain and spring dynamics', *SAE International Congress and Exposition*. doi: 10.4271/890619.

Pieprzak, J. M., Willerrnet, P. A. and Dailey, D. P. (2020) 'Experimental Evaluation of Tappet/Bore and Cam/Tappet Friction for a Direct Acting Bucket Tappet Valvetrain'.

PiperCams (2020) 'Piper Cams Duratec Valve Lift'. Available at: https://pipercams.co.uk/camshafts-car.

Qiu, H. (2005) 'Mechanism and Machine Theory A universal optimal approach to cam curve design and its applications', 40, pp. 669–692. doi: 10.1016/j.mechmachtheory.2004.12.005.

R C Juvinall, K. M. M. (2017) *Fundamentals of Machine Component Design*. 6th edn. Hoboken: Wiley.

R G Budynas, J. K. N. (2012) *Mechanical Engineering Design*. 9th edn. McGraw-Hill.

Rahnejat, H. (2007) 'Mathematical modelling of layered contact mechanics of cam – tappet conjunction', 31, pp. 2610–2627. doi: 10.1016/j.apm.2006.10.019.

Ricardo (2020) *VALDYN - Valvetrain Design*. Available at: https://software.ricardo.com/products/valdyn/valdyn-valvetrain-design.

Sahu, L. K., Kedia, V. K. and Sahu, M. (2016) 'Design of Cam and Follower system using Basic and Synthetic curves : A Review', 3(2), pp. 362–372.

Schamel, A. R. *et al.* (1993) 'Modeling and Measurement Techniques for Valve Spring Dynamics in High Revving Internal Combustion Engines', (412).

Sequenzia, G. *et al.* (2011) 'A new methodology for calculating and modelling non-linear springs in the valve train of internal combustion engines', *SAE Technical Papers*.

Tauschek, M. A. X. J. (1958) 'VALVE TRAIN ENGINEERING - FUNDAMENTALS FOR THE E N G I N E D E S I G N E R'. Available at: https://doi-org.ezproxy.uwtsd.ac.uk/10.4271/972937.

Turkish, M. C. (1950) 'Inspection of Cam Contours by the Electronic Method.pdf', *Eaton Engineering Forum*, 11(1).

Turkish, M. C. (1953) 'Relationship of valve-spring design to valve gear dynamics and hydraulic lifter pump-up', *SAE Technical Papers*. doi: 10.4271/530263.

Xu, Y. *et al.* (2019) 'Influence of Intake Valve Lift on Flow Capacity of Intake Port', pp. 1–7. doi: 10.4271/2019-01-0223.Abstract.

Yasuda, Y. *et al.* (2003) 'Research on Diamond-Like Carbon Coatings for Low-Friction Valve Lifters', (724).

Zigo, M. (1967) 'A General Numerical Procedure for the Calculation of Cam Profiles From Arbitrary Specified Acceleration Curves', *Journal of Mechanisms*, 2(4), pp. 407–414.

Zuck, D. (2020) *Valve Train Design and Calculation for High-performance Engines*, *Advanced Engine Technology*. Available at: http://www.cds-valvetrain.com/Doku/Valve Train.pdf (Accessed: 29 April 2020).

16. Appendix

16.1 Matrix Sine Study

```
%% BUILDING SINE WAVE TO REPRESENT VALVE LIFT

t = 0:0.01:3.59; % Period
a = 9.15; % Amplitude
f = 1/(2*3.59); % Frequency
sinlift = a*sin(2*pi*f*t); % SineWave

Phi = (0 :1: 359); % 1 deg steps
```

Figure 16.1 Sine wave script to replicate valve lift and camshaft rotation, designed to test valve lift matrix for noise introduction.

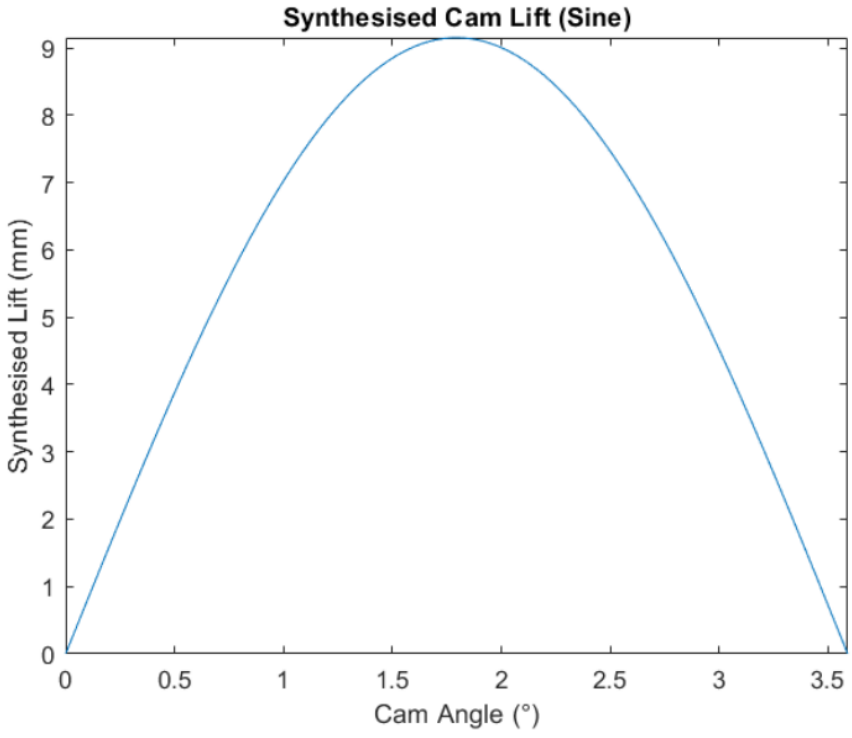


Figure 16.2 Sine Wave Plot for replicating cam Lift (before valve lift matrix).

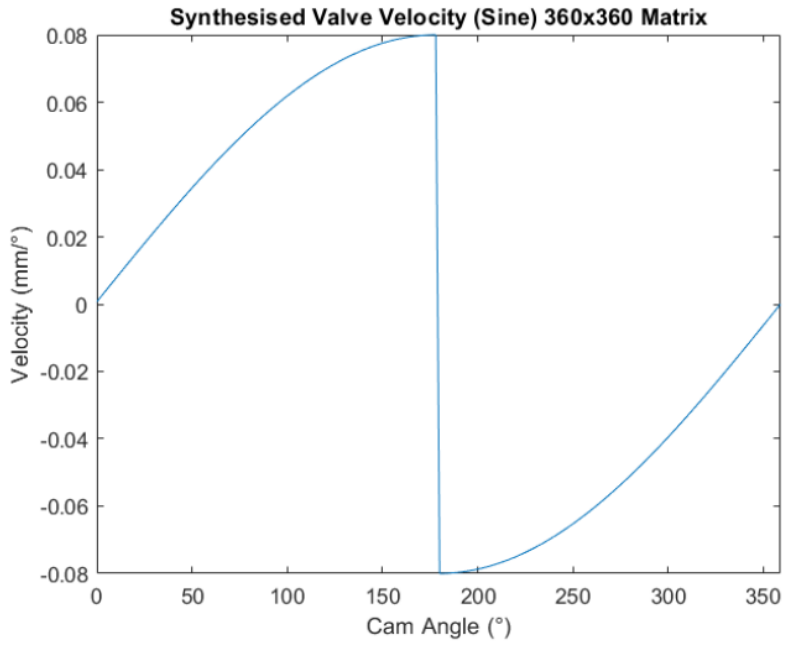


Figure 16.3 Synthesised valve velocity proves the pure valve lift matrix does not introduce noise.

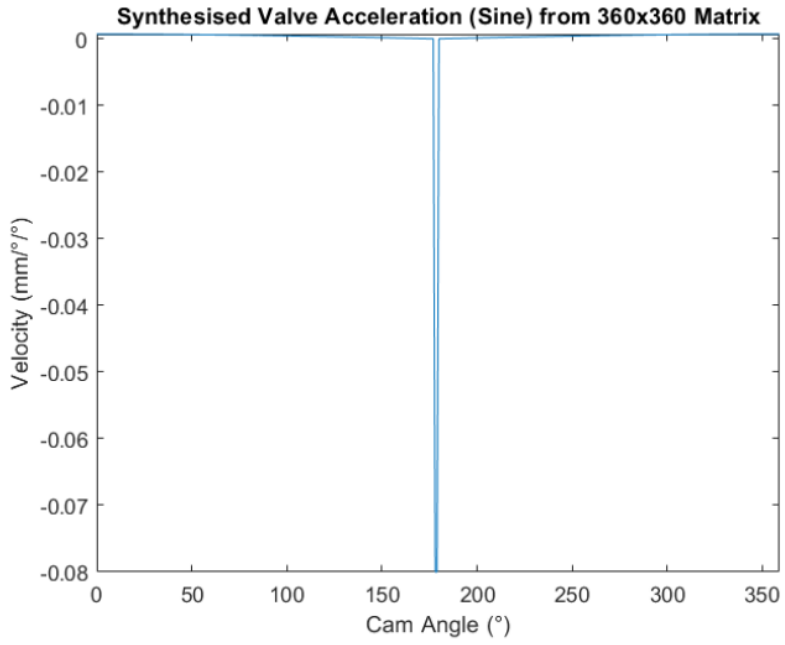


Figure 16.4 Synthesised valve acceleration proves the pure valve lift matrix does not introduce noise.

16.2 MCR Maximum Engine Speed & Working Range

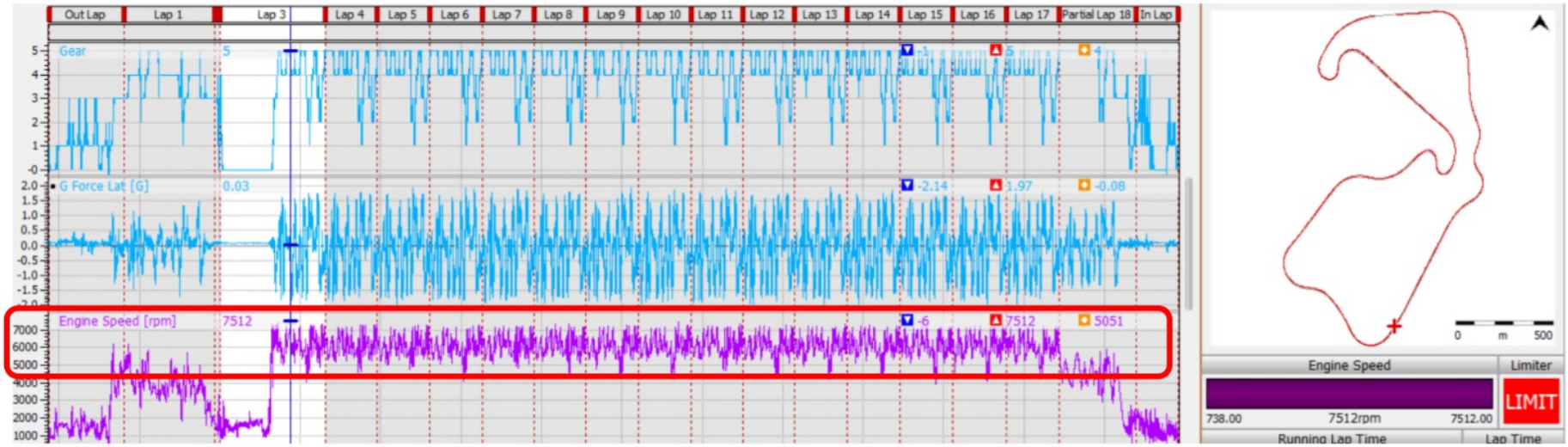


Figure 16.5 MoTeC data acquisition for the UWTSD MCR Race Car at Silverstone GP circuit, shows peak engine speed at 7,512rpm with aggressive downshift into Stowe corner. Also shows average lap engine speed between 5000rpm and 7500rpm.

1000	2000	3000	4000	5000	6000	6500	6600	6700	6800	6900	7000	7100	7200	7300	7400	7500

Figure 16.6 Weighted engine speed vector used for this investigation to focus in most frequent engine speed range of MCR Race Car.

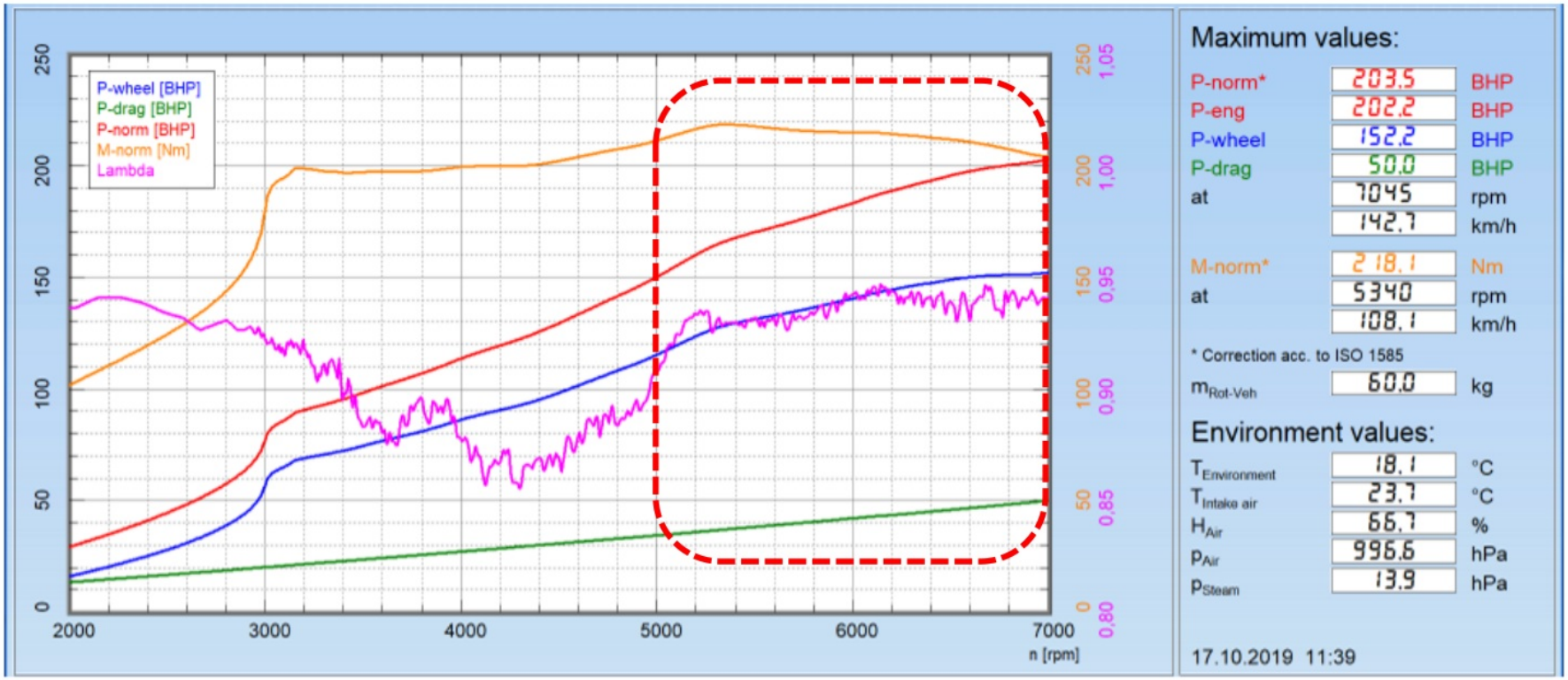


Figure 16.7 Chassis dynamometer print out of UWTSD MCR Race Car, where the lambda switches after 5,000rpm, indicating mapped engine speed range.

16.3 Valve Motion

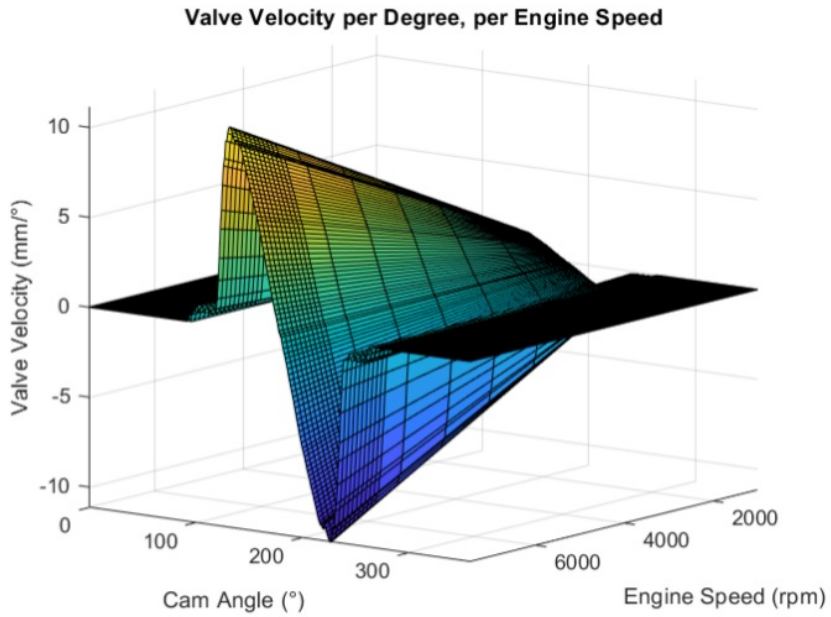


Figure 16.8 Valve velocity from raw measured data, per engine speed.

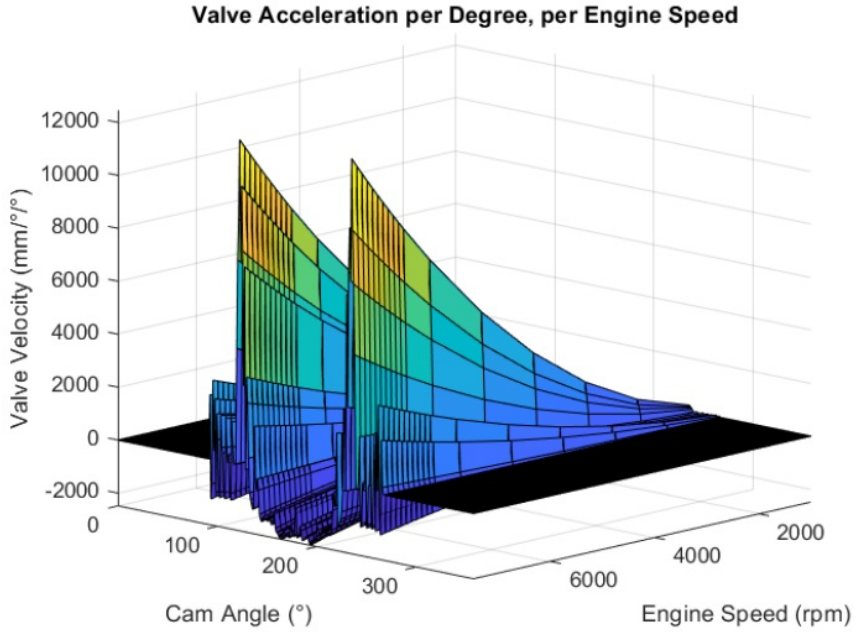


Figure 16.9 Valve velocity from raw measured data, per engine speed.

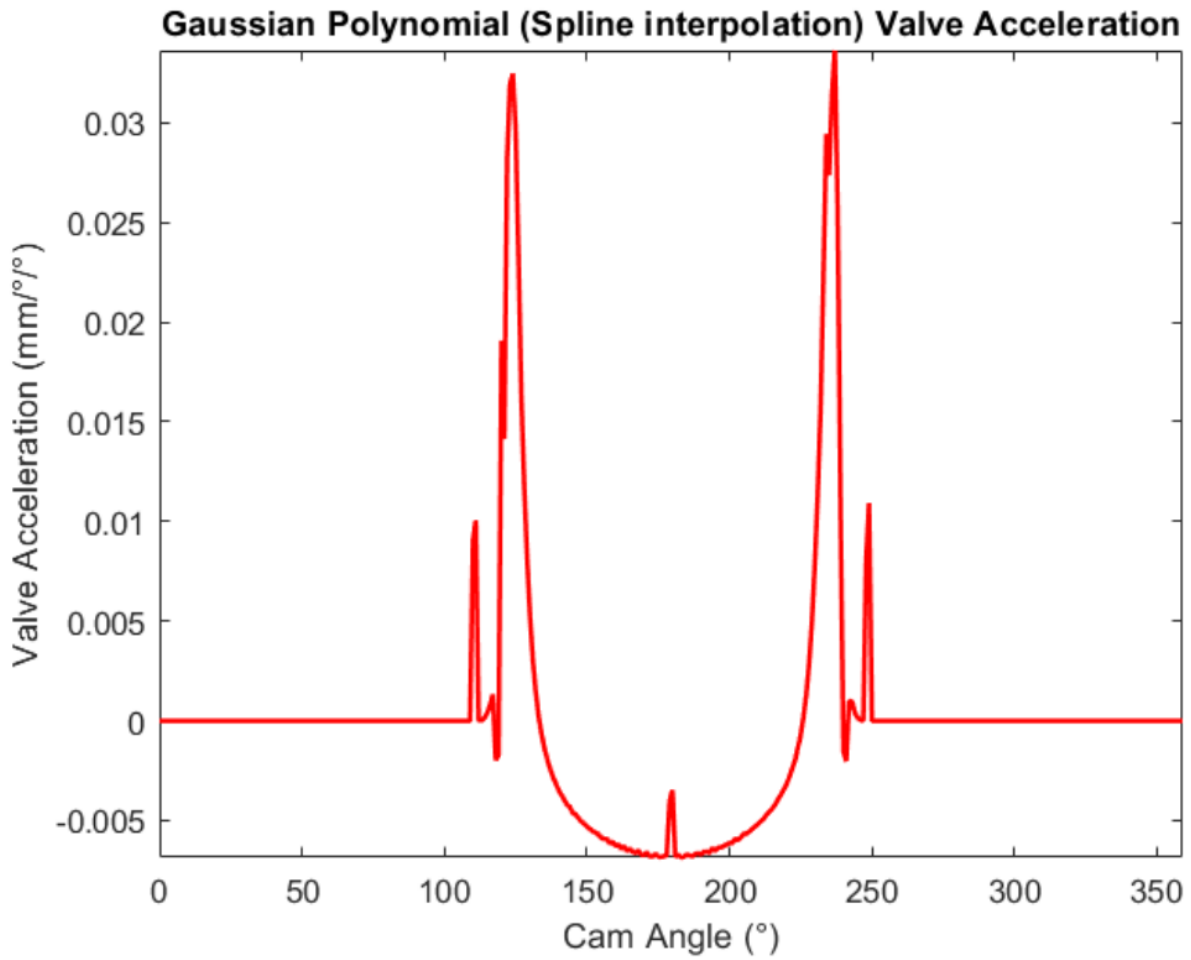


Figure 16.10 Gaussian Polynomial (3,600 x 3,600) with spline interpolation (non-shape-preserving). Yields imprvement in noise over PCHIP.

16.4 Fourier Analysis (Kent Cams Profile)

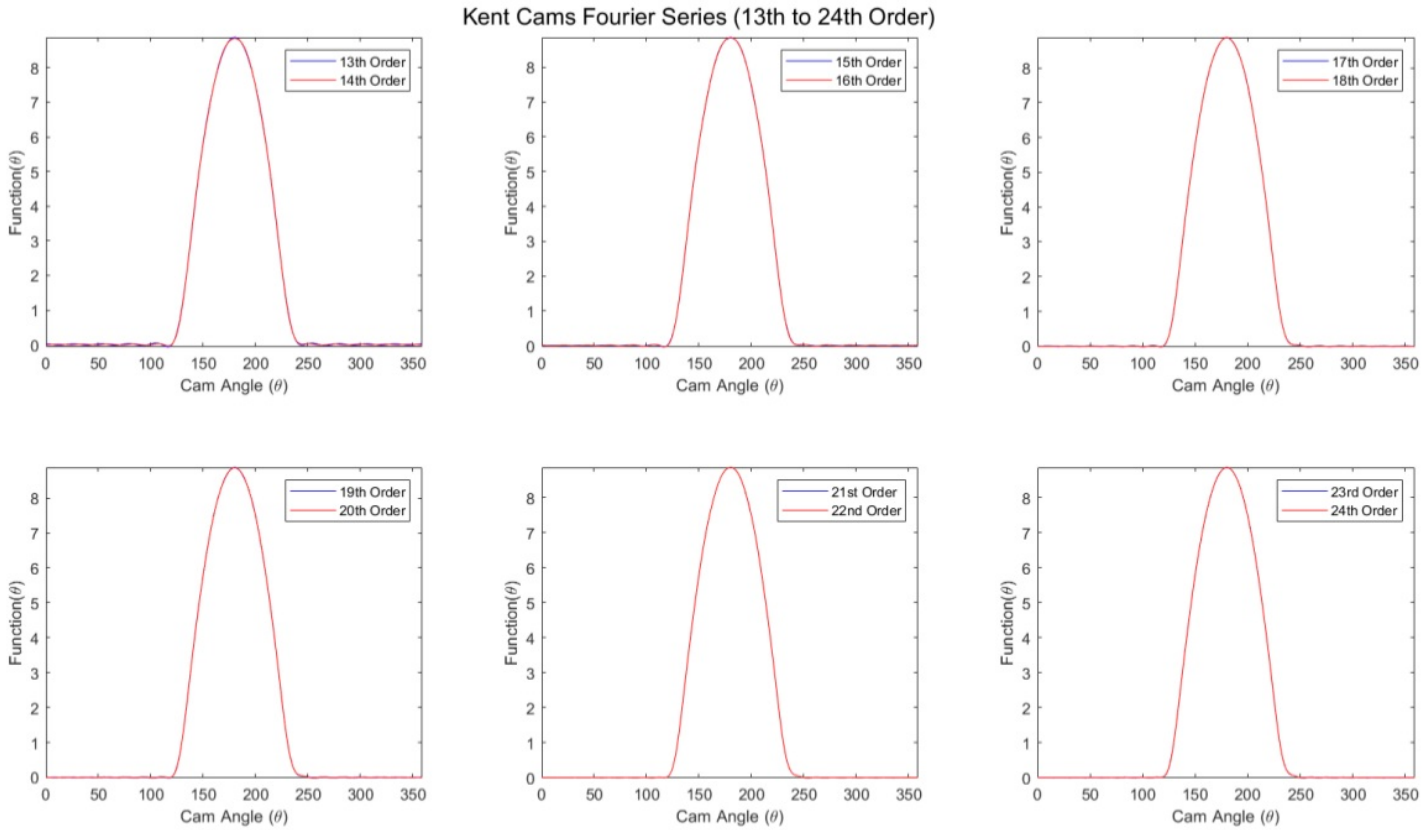


Figure 16.11 Convergence of Fourier series (orders 13 to 24).

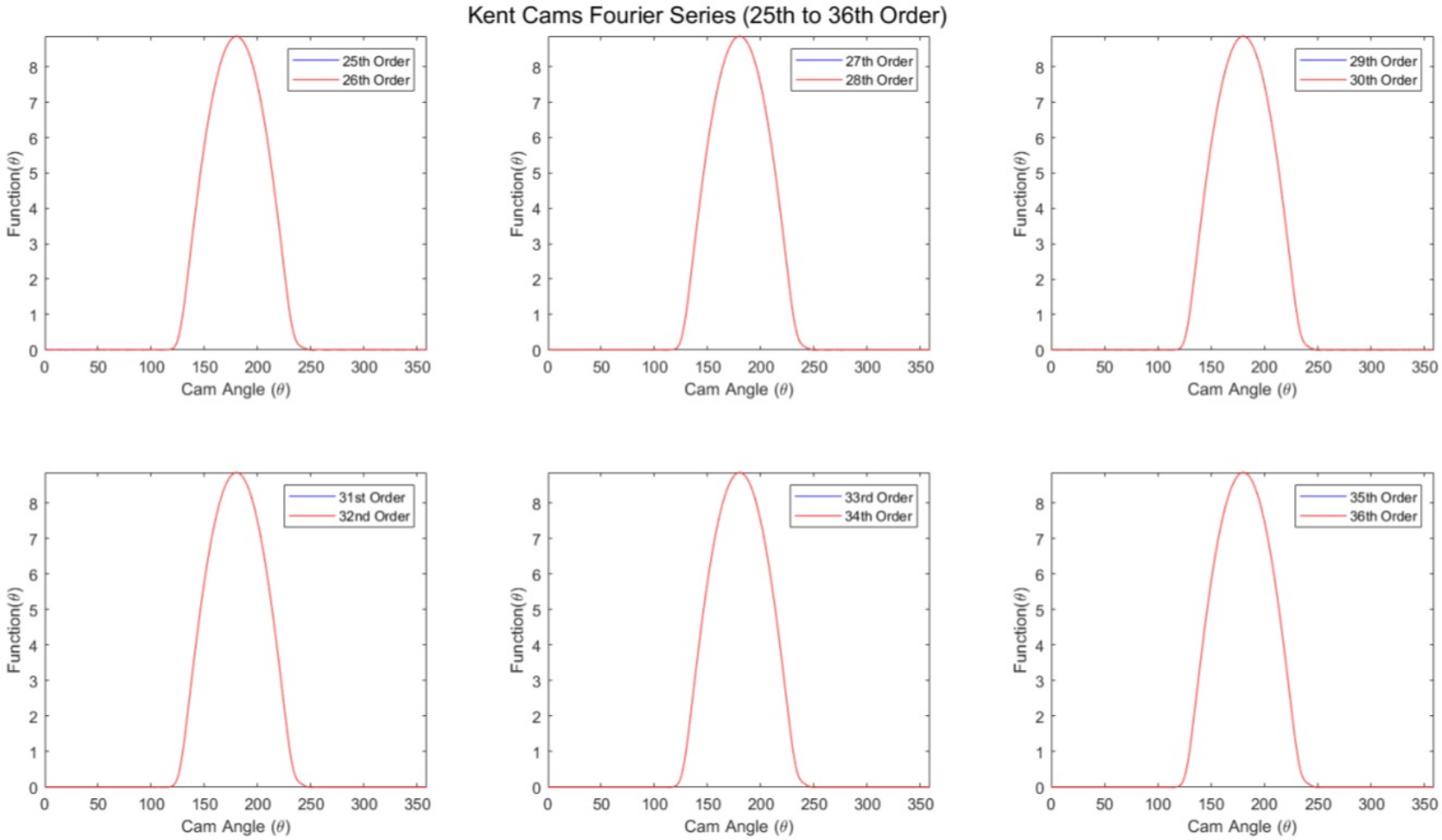


figure 16.12 Convergence of Fourier series (orders 25 to 36).

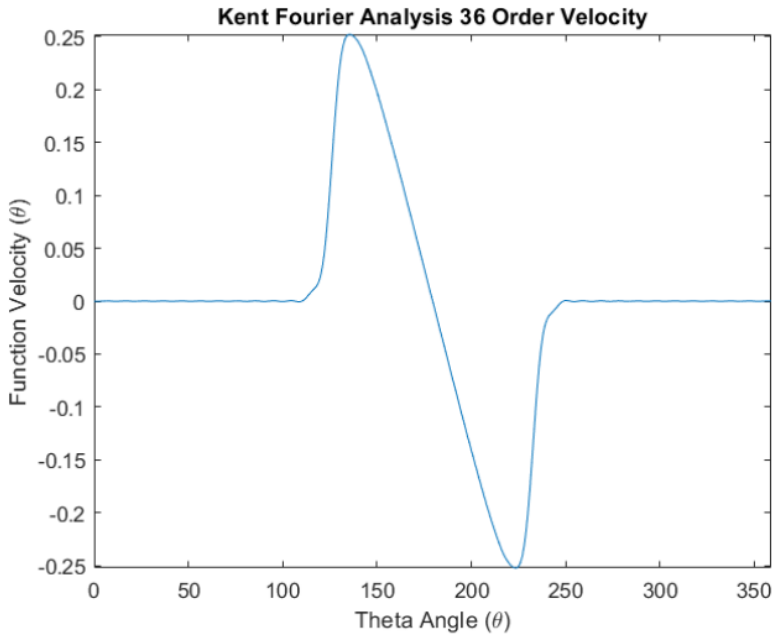


figure 16.13 differential of Fourier series for valve lift.

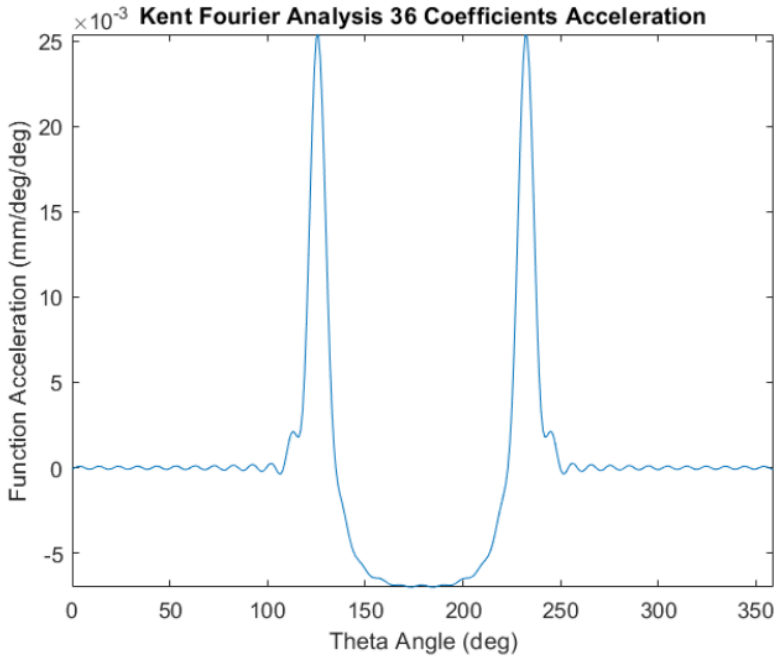


figure 16.14 2nd order differential of Fourier series for valve lift. Shows little noise where measured does.

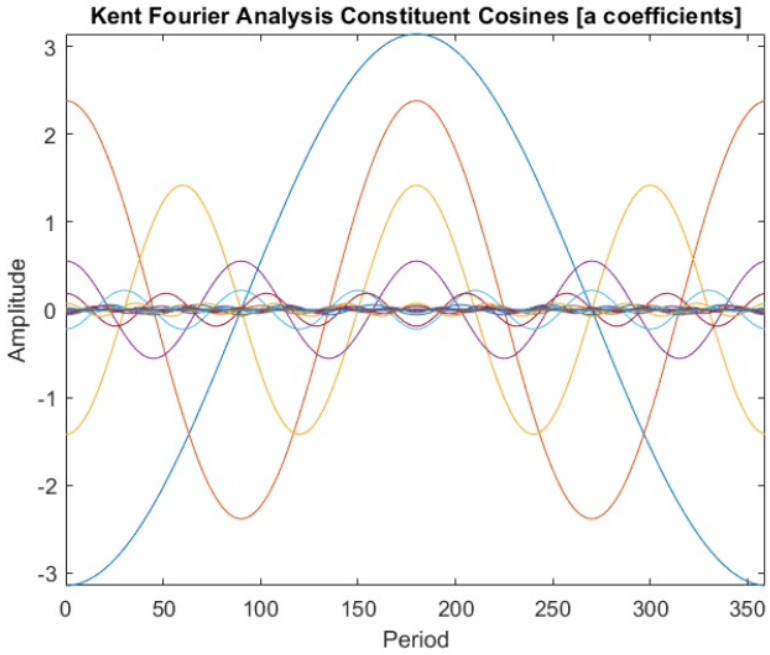


figure 16.15 Constituent cosine waves of the Fourier series.

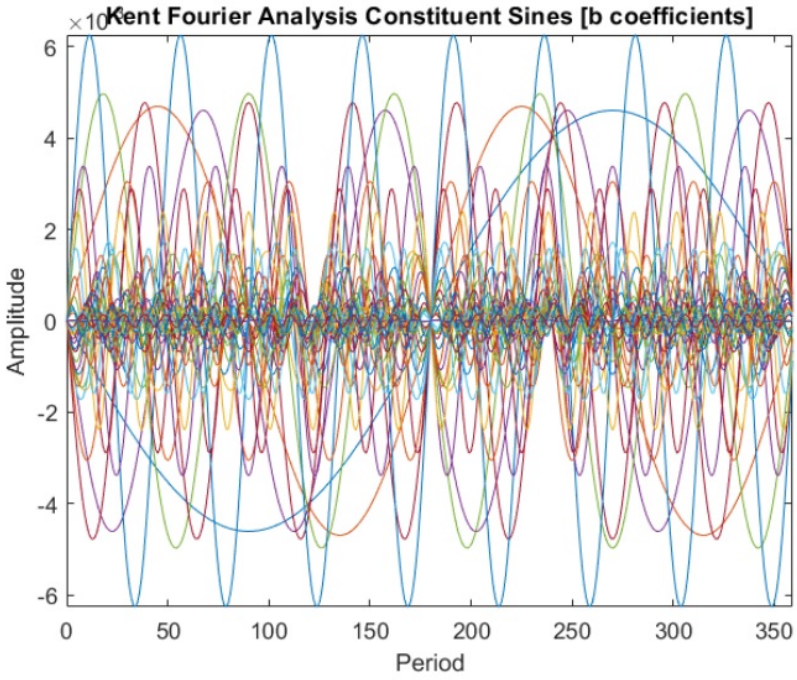


figure 16.16 Constituent sine waves of the Fourier series.

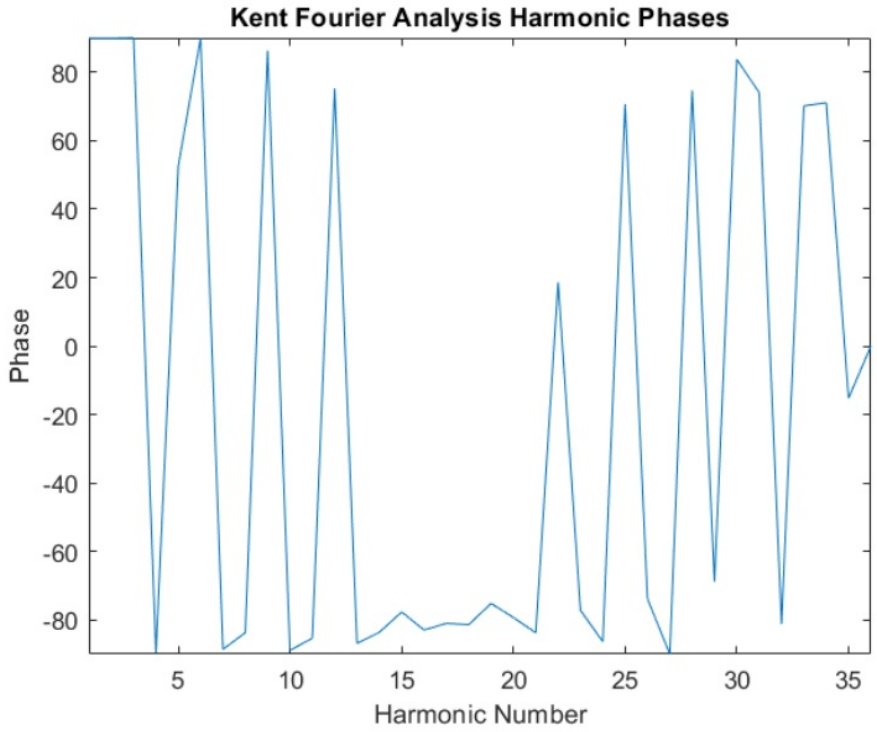


Figure 16.17 Phase plot of Fourier Series.

16.5 Fourier Analysis (Valve Acceleration)

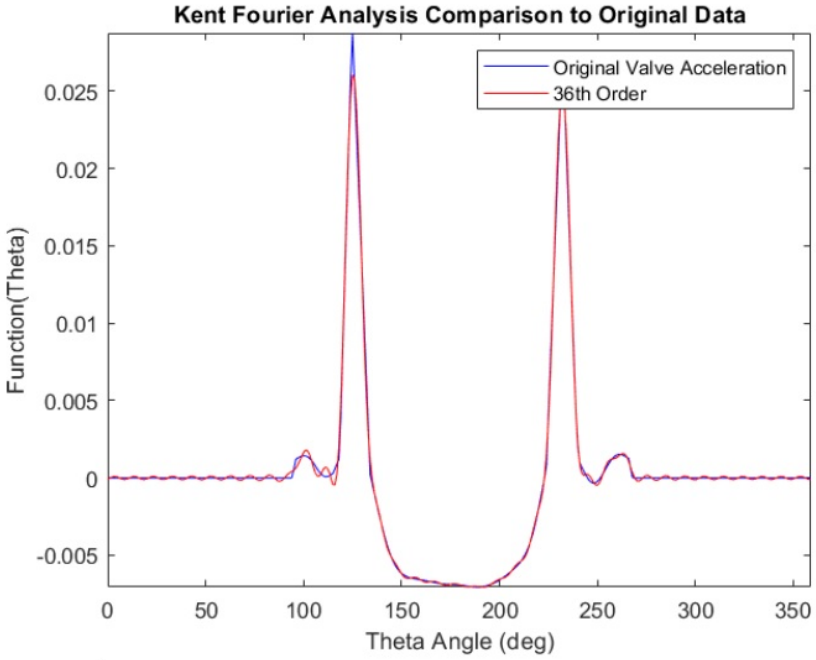


Figure 16.17 36th order of acceleration convergence (requires higher order, more complex function).

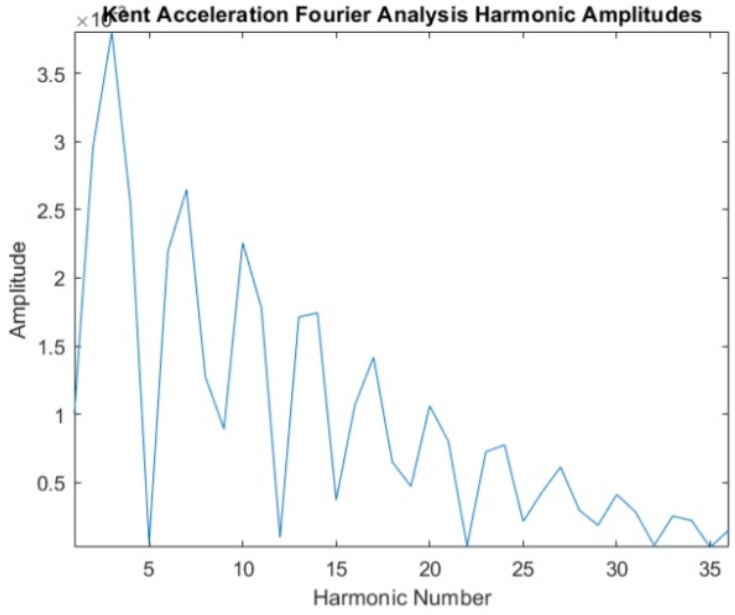


Figure 16.18 36th order of Fourier acceleration harmonic amplitudes.

16.6 Engine Dynamometer Preparation

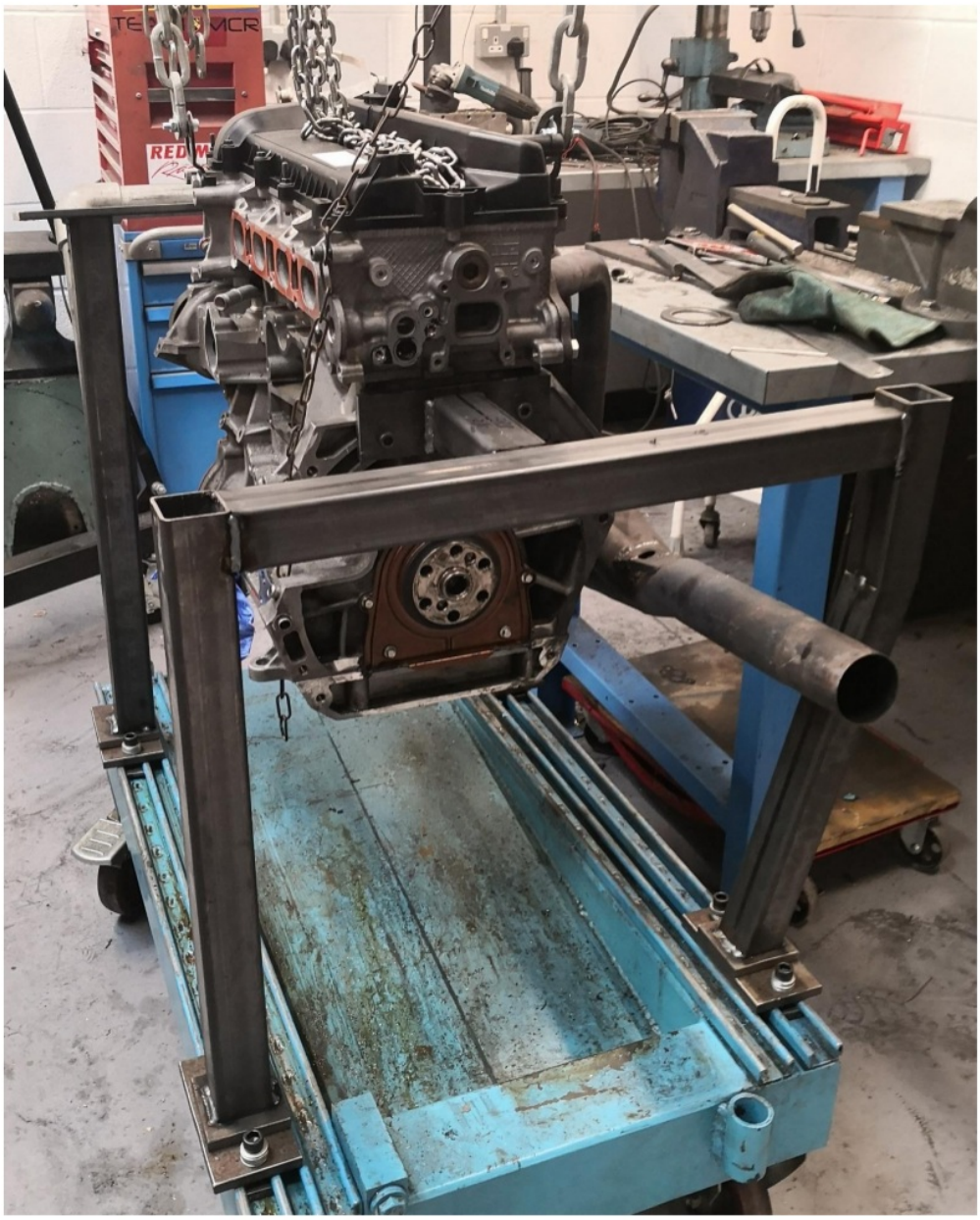


Figure 16.19 preparing the Duratec engine for future use on a dynamometer, rear of engine trolley.



Figure 16.20 preparing the Duratec engine for future use on a dynamometer, front of engine trolley.

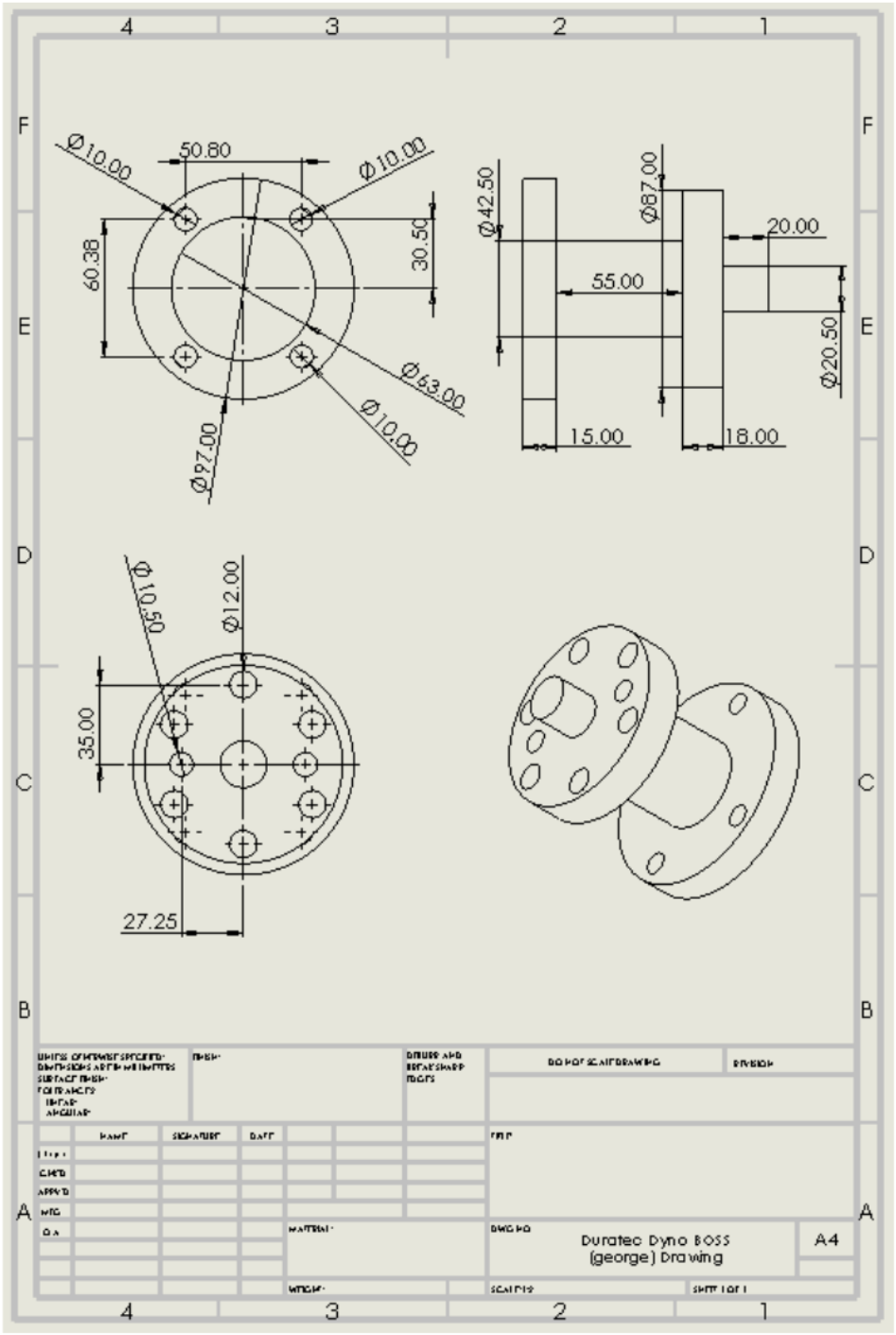


Figure 16.21 Dynamometer to Duratec adaptor boss, created in Solidworks from engine measurements.

16.7 Duratec (DHE420) Crate Engine Procedure for Competition Use

16.7.1 Introduction

The following method is designed to prepare a new crate DHE420 power unit (road spec) for competition use. An initial break in procedure is conducted by the manufacturer prior to shipment, although the intended purpose for these crate engines is typically in road car warranties.

The additional loadings and increased engine speeds of competition use requires a more comprehensive initial running procedure to reduce risk of premature fatigue/failure.

16.7.2 Crate Engine Adaptation

The engine should be visually inspected upon delivery for signs of external damage (from storage or transit), with attention to corners and extruding faces.

The crate engine is supplied wet (used at the factory for initial break-in). This oil has an increased level of Zinc dialkyldithiophosphate (ZDDP) which is a heat activated anti-wear additive. The sump should be drained before any adaptations are carried out.

16.7.3 Engine Dynamometer Application

- Wet sump system
- Dyno adaptor boss - CAD, material, extended bolts to crank, use of existing dowel pins in crank
- Mounting frame – packaging & dimensions, CAD, engine mounting locations, NVH solutions, remote systems (coolant, loom, fuel supply)

16.7.4 MCR Chassis Application

- Dry sump system (including oil pump) – clean/flush all external oil components

Once all relevant adaptations are complete it is necessary to prime the lubrication system and check basic engine operation.

16.7.5 Initial Test

1. Remove spark plugs and disconnect coil pack
2. Remove fuel pump relay
3. Crank engine for 10 second bursts at 30 second intervals (to avoid damaging starter motor)
4. Observe oil pressure increasing after each burst (oil pressure should reach **40-45psi** whilst cranking)
5. Repeat step 3 a further 5 times after reaching 40psi to ensure oil has reached valvetrain.
6. Replace fuel pump relay
7. Replace spark plugs and connect coil pack
8. Start engine and monitor oil pressure (oil pressure should now reach **58-64psi** at idle)
9. Wait 30 seconds before raising engine speed to 2,500rpm and hold for 30 seconds
10. Progressively pulse the engine speed between 2,500rpm and 4,000rpm for a further 15 minutes whilst observing oil pressure remains above 58psi
11. Allow engine to idle for 60 seconds with consistent oil pressure before switching ignition off
12. Check engine for leaks and check oil level is correct

16.7.6 Dynamometer Test

1. Start engine and monitor oil pressure (oil pressure should reach **58-64psi** at idle)
2. Allow engine coolant temperature to reach 55˚C for map to cut in.
3. conduct a constant load (constant RPM) test at 3000rpm for 5 minutes.
4. Increase engine speed in 500rpm increments, holding each for 5 minutes until 5000rpm is reached whilst continually monitoring oil pressure.
5. Allow engine to coast down to idle speed before turning off ignition
6. Allow engine to cool to ambient conditions
7. Check engine for leaks and check oil level is correct
8. Repeat steps 1 & 2
9. Conduct a power run in 4^{th} gear
10. Repeat power run at 5-minute intervals until power output stabilises (piston rings sealed)
11. Drain engine oil and check for contaminants before replacing oil filter and refilling with fresh oil.

16.8 Modelled Valvetrain Components

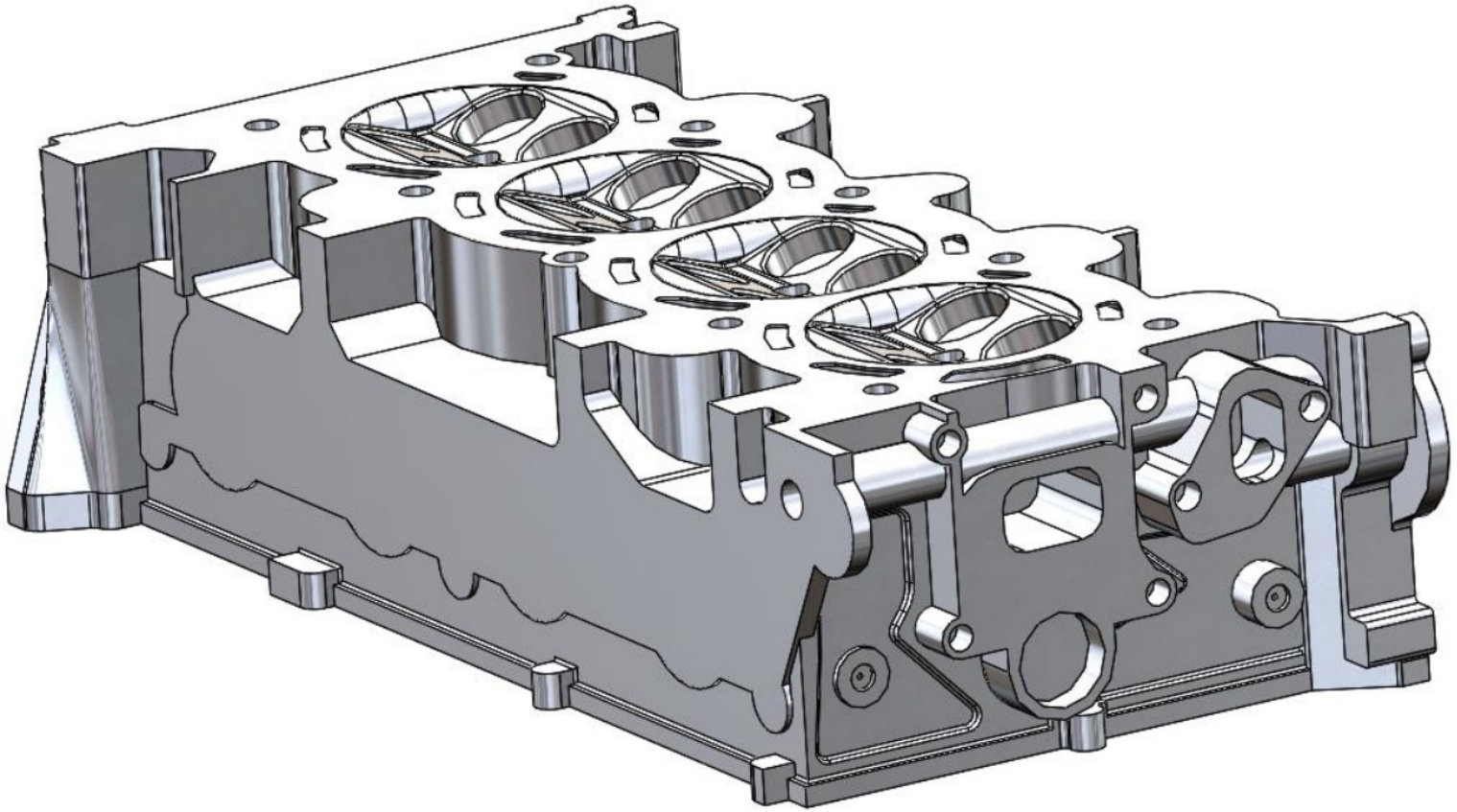


Figure 16.22 Duratec HE420 modelled Cylinder head intended for use in FEA

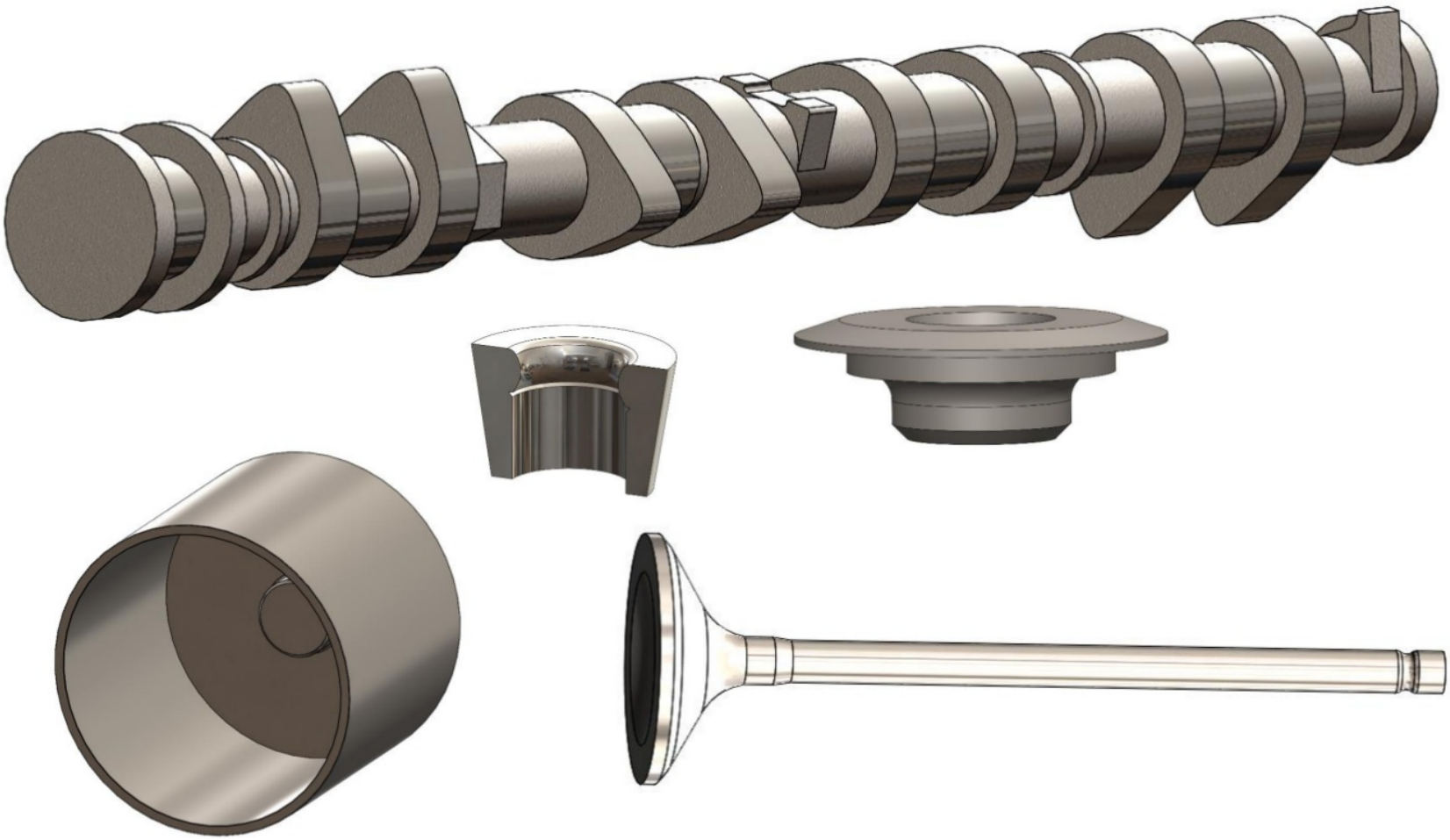


figure 16.23 Duratec camshaft, tappet bucket, collet, valve cap, inlet valve modelled in Solidworks

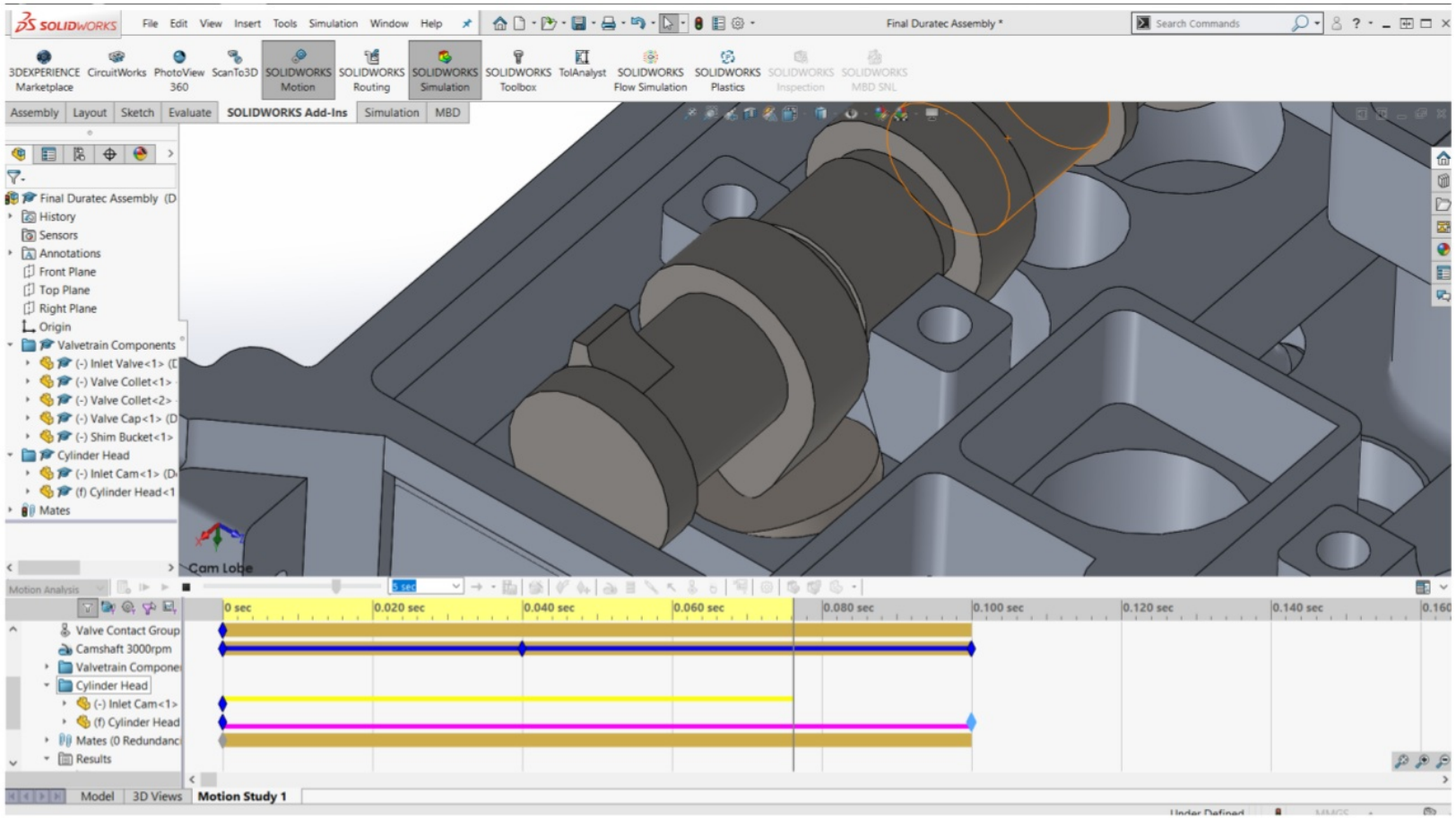


Figure 16.24 Cylinder head motion simulation set up to conduct FEA in Solidworks.

16.9 Initial Flow Bench Results

Flow Range Chart		
Flow Range	Intake	exhaust
1	16.5	19.5
2	33.2	37.1
3	69.7	75
4	138.2	148.5
5	207.8	219.6
6	279.5	284.6

Conversion	
cfm	l/s
7	3.31
2.114803625	

Cylinder Head					
Valve Opening (mm)	Flow Range	Flow Scale (%)	Temp (°C)	Flow (cfm)	Flow (l/s)
0.5	1	44.00%	20	7.26	15.35347
1	1	90.40%	20	14.916	31.54441
1.5	2	68.00%	20	22.576	47.74381
2	3	44.00%	20	30.668	64.8568
2.5	3	54.50%	20	37.9865	80.33399
3	3	63.50%	20	44.2595	93.60015
3.5	3	72.60%	20	50.6022	107.0137
4	4	40.50%	20	55.971	118.3677
4.5	4	44.00%	20	60.808	128.597
5	4	48.00%	21	66.336	140.2876
5.5	5	34.00%	21	70.652	149.4151
6	6	27.00%	21	75.465	159.5937
6.5					
7					
8					
9					
10					
11					
12					

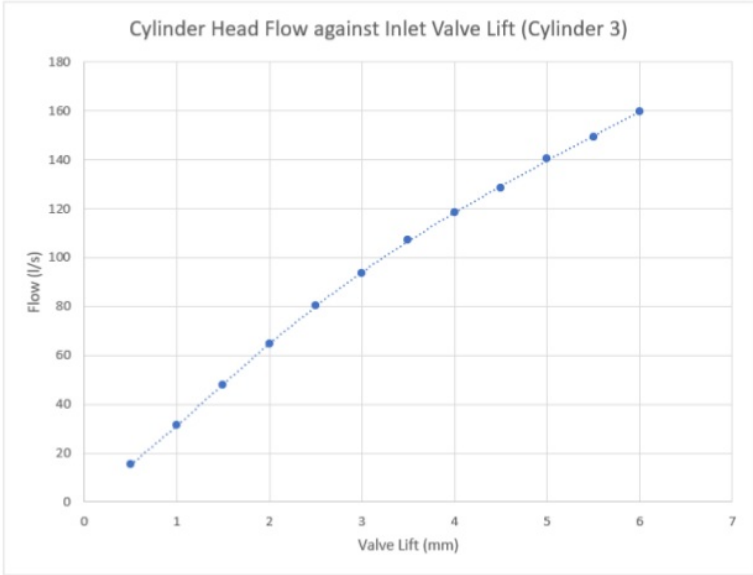


Figure 16.25 Cylinder head flow bench test (flow bench failed through run).

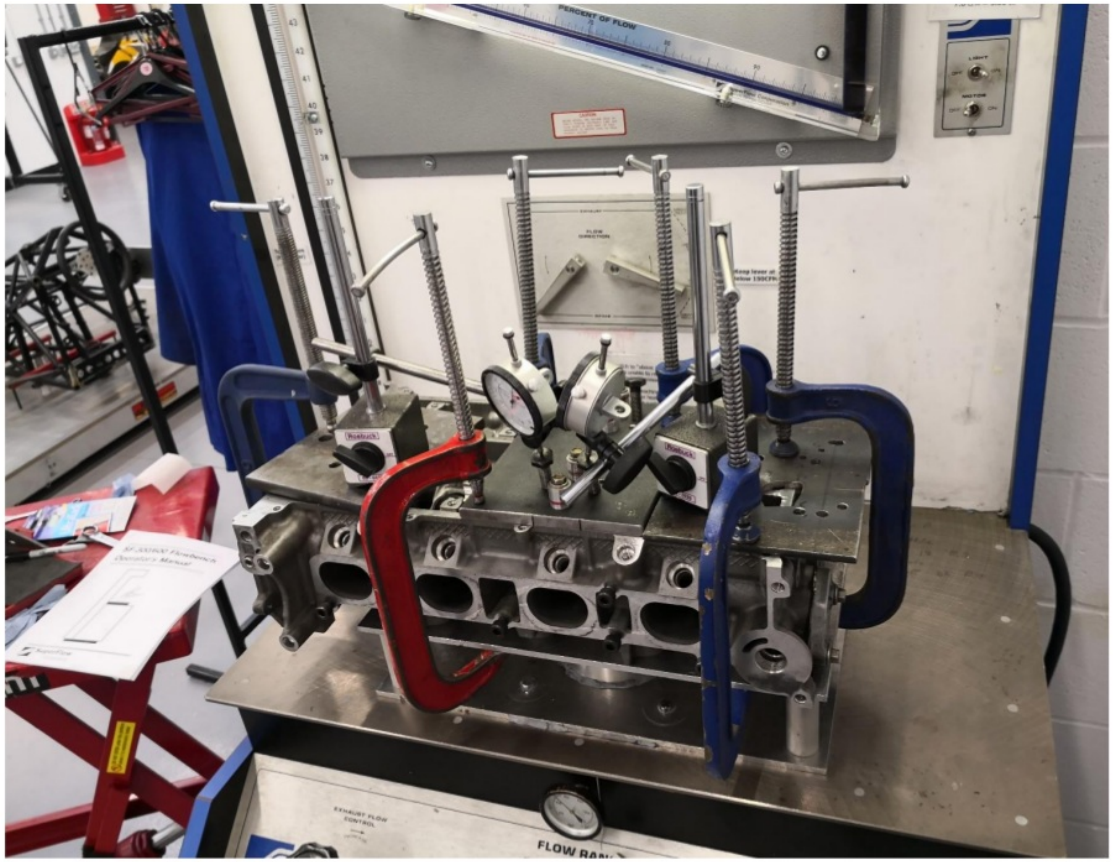


Figure 16.26 Cylinder head flow bench test setup.

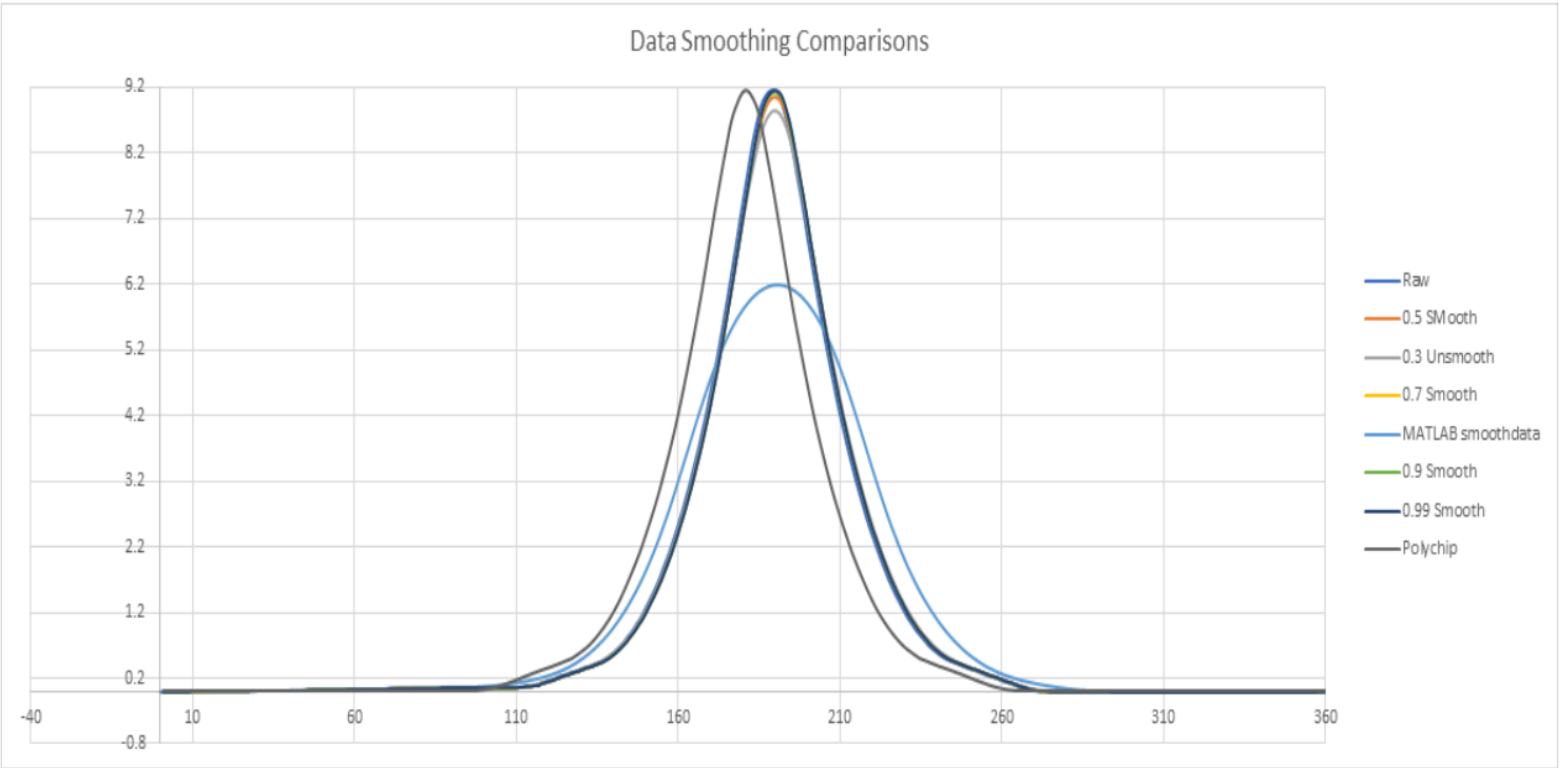


Figure 16.27 Evidence of additional smoothing functions attempted.

16.10 Camshaft Contact Stress

The contact force (due to valve spring and inertia) was used to calculate the Hertzian contact stress at the cam-to-tappet interface (Kent Cams profiles) using the cylinder to flat-plane method.

The exact production materials are unknown, so it is assumed that both camshaft and tappet bucket are made of cast iron, with a Poisson's Ratio of *0.25* and Youngs modulus *160 GPa*.

Radius of cam curvature,

$$R = LRI + L_v + \left(\frac{180^2}{\pi} \times A_v\right)$$

$$where\ R = radius\ of\ curvature\ (lobe)$$

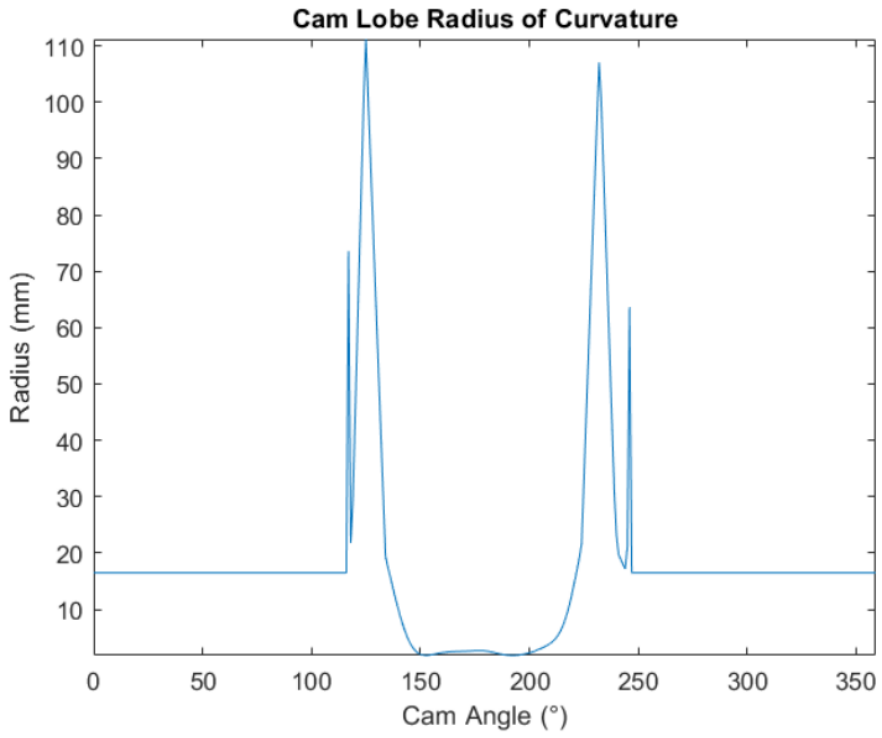


Figure 16.28 Radius of curvature (cam lobe).

For contact area,

$$CA = 2 \times \left[\sqrt[1.076]{\frac{F_c}{L} \times R \times \frac{2}{E}} \right] \times L$$

$where\ L = Length\ of\ contacting\ surface\ (13mm)\ E = Youngs\ Modulus\ (N/mm\text{^}2\ )$

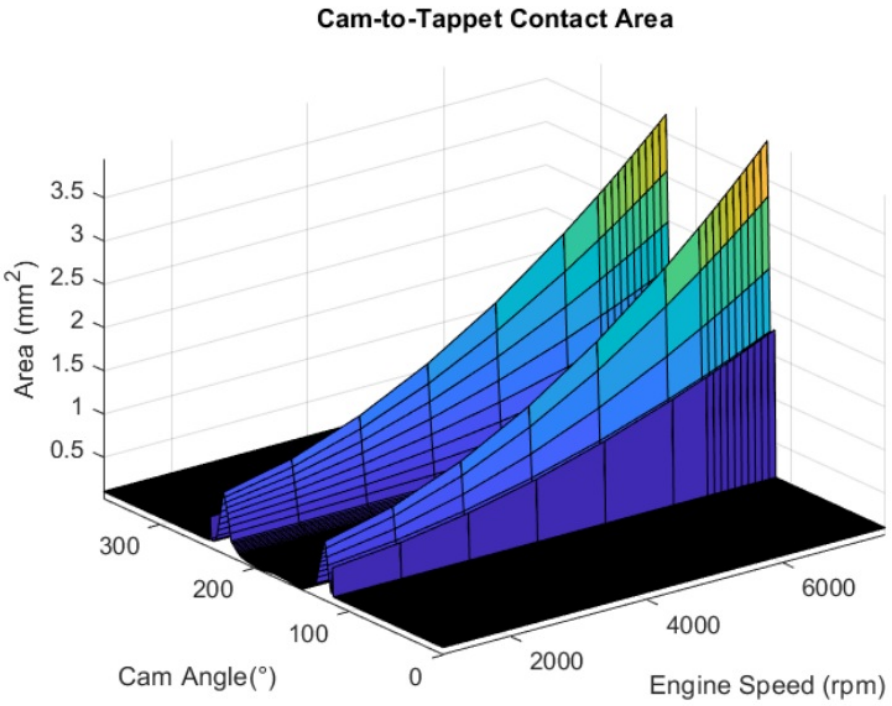


Figure 16.29 Area of contact.

For maximum contact stress,

$$Contact\ Stress = \frac{2}{\pi} \times \frac{F_c}{CA \times L}$$

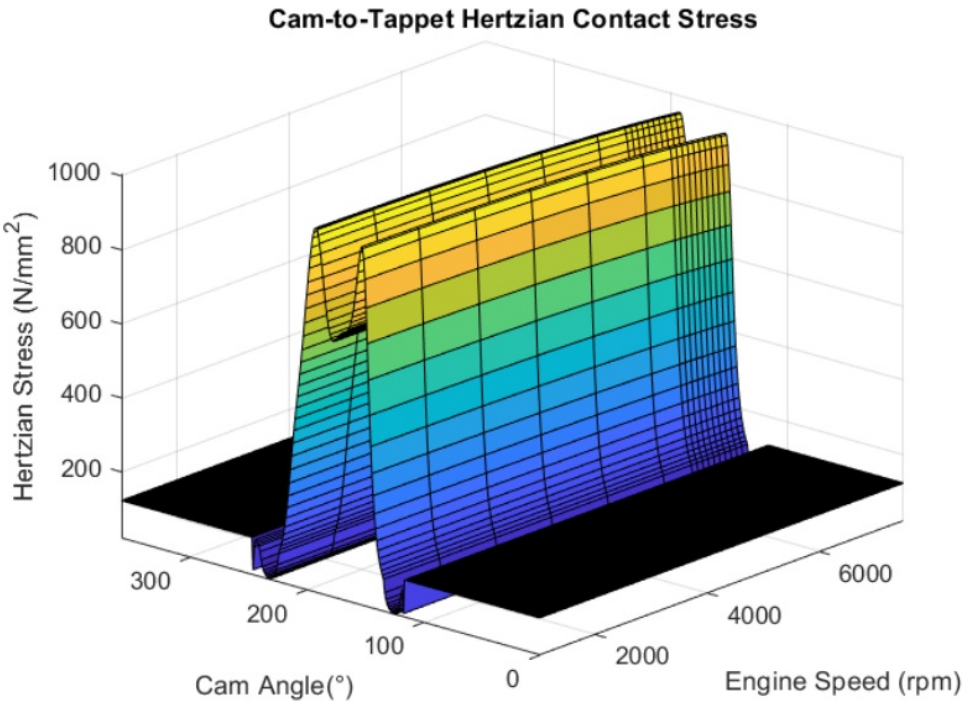


Figure 16.30 Hertzian stress.

16.11 Square Wave Fourier Study

The Fourier hypothesis was tested in MATLAB by simulating a pulse train (square wave) with the summation of sine waves. This example was chosen to test the limitations of the Fourier method, where a continuous wave model is used to represent a discontinuous function.

A weak approximation can be formed as early as the 3rd order in the series.

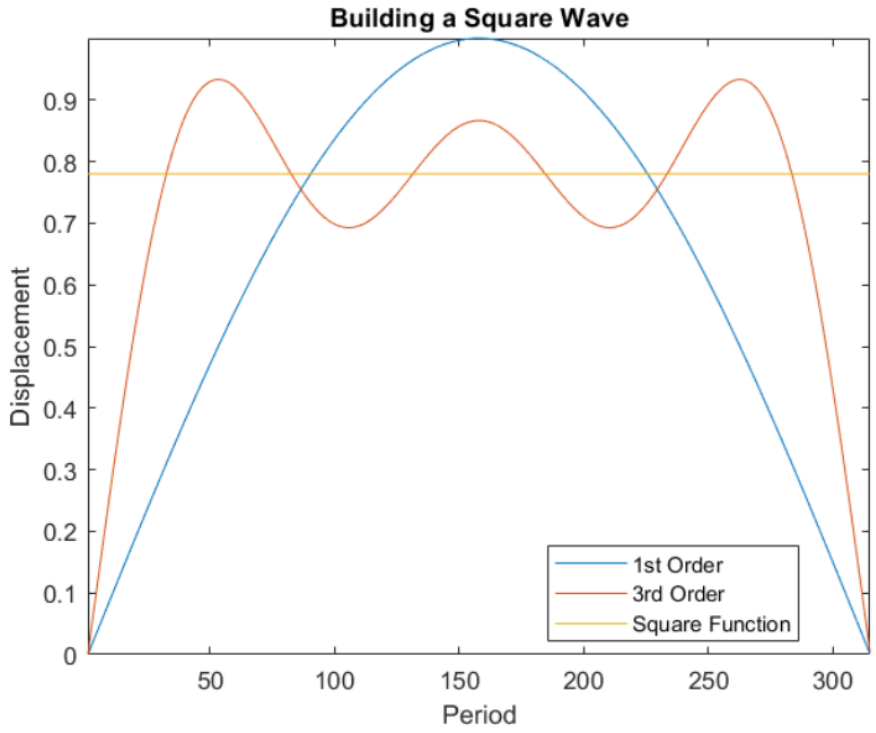


Figure 16.31 Building a square wave.

As the order is increased, correlation against the original square signal is refined. 49th order shows an overshoot at each of the jump discontinuities. This effect is known as *ringing* and is a consequence of the *Gibbs Phenomenon*, highlighting one of the limitations in representing functions with the summation of sinusoids.

The order was increased to 4,999, where the Gibbs effect can still be seen to act. However, convergence across the remainder of the signal is robust.

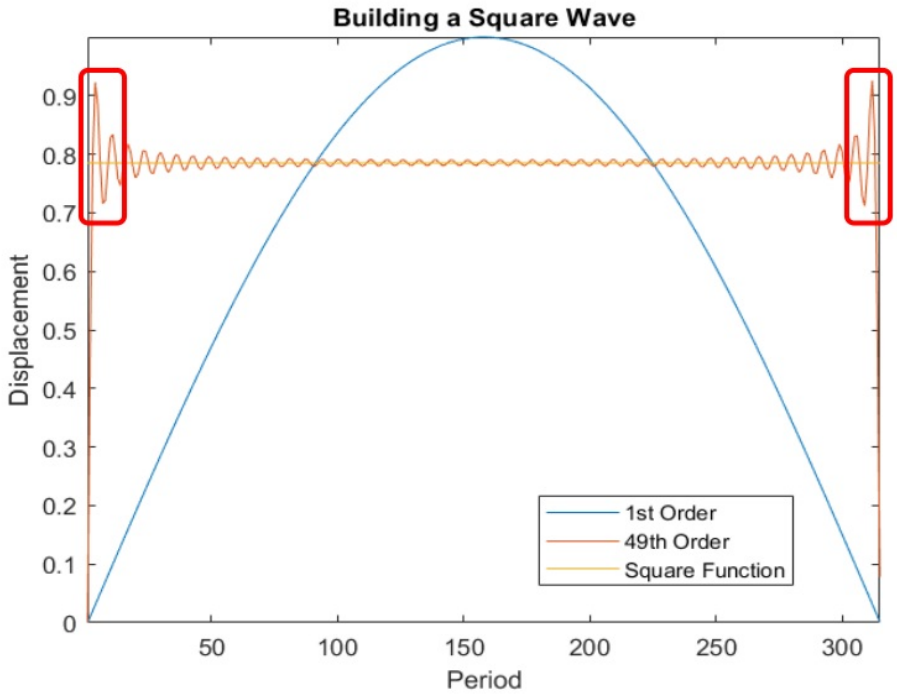


Figure 16.32 Building a square wave (49th order).

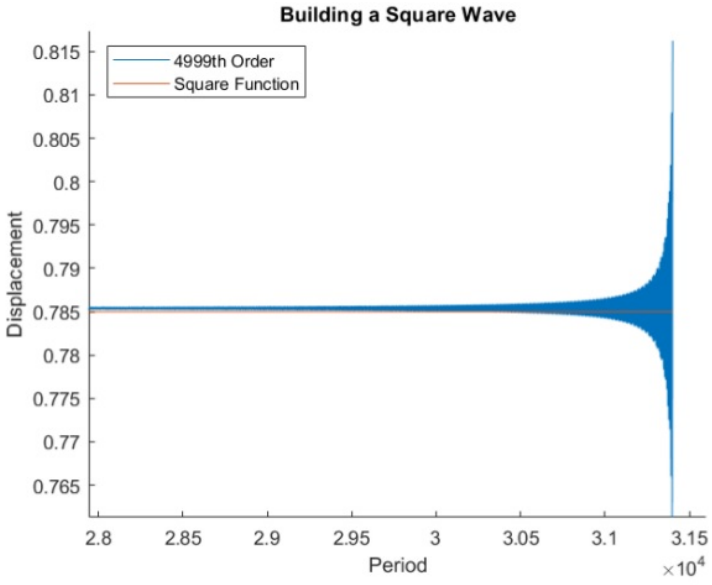


Figure 16.33 Building a square wave (4,999th order).

16.12 Matlab Scripts

16.12.1 Constants Script

```
%% INLET Constants for Cam & Valve Calculations

LRI = 16.5;                         % Lobe Radius - Measured
(mm)
SCI = 0.25;                         % Shim Clearance -
Autodata Average (mm)
CRTI = 0.06;                        % Cam Runout Tolerance
(mm)
MLLI = 9.15;                        % Max Lobe Lift -
Measured (mm)

%% INLET - Raw Lift Data (mm)

RLI = xlsread ('Cam LIFT' , 'raw inlet' , 'G3:G362');
%% Cam Angle Vectors (degrees)

Theta = (0 :1: 359)';               % Theta for 360 points [1
deg steps]
ThetaB = (0 :0.5: 359.5)';          % Theta for 720 points
[0.5 deg steps]
ThetaC = (0 :0.1: 359.9)';          % Theta for 3,600 points
[0.1 deg steps]
ThetaD = (0 :0.01: 359.99)';        % Theta for 36,000 points
[0.01 deg steps]

%% ValveTrain Data

RPM  = [1000:1000:6000 6500:100:7500];     % Engine Speed

% Inlet Moving Masses (kg)
%%%%%%%%%%%%INLET%%%%%%%%%%%%
% Inlet Bucket        = 0.033
% Inlet Valve         = 0.0472
ISM       = 0.0347;  % Inlet Spring Mass
% Inlet Spring Cap    = 0.0098
% Inlet Collets       = 0.0018

ITMM = 0.10915;                     % Total Inlet Moving Mass
(kg)
```

```
ISSH = 0.0385;                  % Inlet Spring Seated
Height (m)
ISFH = 0.0426;                  % Inlet Spring Free
Height (m)
ISCD = 0.0041;                  % Inlet Seated
Compression Distance (m)
IMVL = 514.2144;                % Inlet Maximum Valve
Spring Load (N)
IVSC = 39481;                   % Inlet Valve Spring
Constant (Constant)
ICF  = 0.1;                     % Inlet Coefficient of
Friction
```

16.12.2 Example Curve Fitting Procedure

```
%% Type " cftool " Into Command Window To Open Curve
Fitting Tool

%% SPLIT RAW DATA FOR CURVE FITTING

% 0 DEGREES TO 95 DEGREES IS ZERO
% 97 DEGREES TO 141 DEGREES IS UP RAMP
% 142 DEGREES TO 222 DEGREES IS MAIN
% 223 DEGREES TO 267 DEGREES IS DOWN RAMP
% 268 DEGREES TO 360 DEGREES IS ZERO

%% Up Ramp from data point 97deg (0.01mm) to 141deg
(1.29mm) [45deg range]
UpRampY = RLI(97:141,1);
UpRampX = 1:1:45;


% Curve Fit of Up Ramp - Polynomial 9th order

%Coefficients (with 95% confidence bounds):
p1 =   4.781e-13;  %(1.982e-13, 7.579e-13)
p2 =  -9.374e-11;  %(-1.518e-10, -3.573e-11)
p3 =   7.607e-09;  %(2.541e-09, 1.267e-08)
p4 =  -3.308e-07;  %(-5.73e-07, -8.864e-08)
p5 =   8.352e-06;  %(1.45e-06, 1.525e-05)
p6 =  -0.0001228;  %(-0.0002424, -3.204e-06)
p7 =   0.0009617;  %(-0.0002698, 0.002193)
p8 =    -0.00242;  %(-0.009452, 0.004612)
p9 =    0.002991;  %(-0.01623, 0.02221)
p10 =   0.007949;  %(-0.009889, 0.02579)


%Linear model Poly9
UpRampCF = (p1*UpRampX.^9 + p2*UpRampX.^8 + p3*UpRampX.^7
+ p4*UpRampX.^6 + p5*UpRampX.^5 + p6*UpRampX.^4 +
p7*UpRampX.^3 + p8*UpRampX.^2 + p9*UpRampX + p10)';

% Goodness of fit:
% SSE: 0.0003308
% R-square: 0.9999
% Adjusted R-square: 0.9999
% RMSE: 0.003074

% CURVE ACCURATE FROM 1 [0.0094] TO 23 [0.3399] OF
"UpRampCF"
```

```
STUpRampCF = UpRampCF(1:23,1);
STUpRampX = 1:1:23;

figure(11)
plot(STUpRampX,STUpRampCF)
hold on
plot (UpRampX(1:23),RLI(97:119));
hold off

%up to RLI 119 is PROVEN CORRECT
xpoly = 96:1:140;
% Plot of curve against it's corresponding cam angles
figure (1)
plot(xpoly,UpRampCF)
title 'Clearance Ramp Curve Fit'
xlabel 'Cam Angle (°)'
ylabel 'Cam Lift (mm)'
axis tight
hold on
plot(xpoly,RLI(97:141,1));
legend ({'9th Polynomial', 'Measured'})
hold off
%% MAIN from data point 142deg (1.39mm) to 220deg
(1.39mm) [79deg range]
MainY = RLI(142:220,1);
MainX = 1:1:79;

% Curve Fit of Main Duration - Gaussian 3rd order

% Coefficients (with 95% confidence bounds):
a1 =       1.014;  %(0.7854, 1.244)
b1 =          40;  %(39.93, 40.07)
c1 =       9.269;  %(8.73, 9.808)
a2 =        2.72;  %(2.62, 2.821)
b2 =          40;  %(39.95, 40.05)
c2 =       16.57;  %(15.59, 17.55)
a3 =       5.431;  %(5.214, 5.648)
b3 =          40;  %(39.97, 40.03)
c3 =       33.24;  %(32.78, 33.7)

% General model Gauss3:
MainCF = (a1*exp(-((MainX-b1)/c1).^2) + a2*exp(-((MainX-
b2)/c2).^2) + a3*exp(-((MainX-b3)/c3).^2))';

% Goodness of fit:
% SSE: 0.00359
% R-square: 1
```

```
% Adjusted R-square: 1
% RMSE: 0.007162
xmain = 142:1:220;
% Plot of curve against it's correstponding cam angles
figure (2)
plot(xmain,MainCF)
title 'Flank and Nose Curve Fit'
xlabel 'Cam Angle (°)'
ylabel 'Cam Lift (mm)'
axis tight
hold on
plot(xmain,RLI(142:220,1));
legend ({'Gaussian (3 terms)', 'Measured'})
hold off
%% DownRamp [Inverse of UpRamp]

DownRampX = fliplr(UpRampX);
DownRampY = fliplr(UpRampY);
DownRampCF = flip(UpRampCF);

% Plot of curve against it's correstponding cam angles
figure (3)
plot(DownRampX,DownRampCF)
title 'Down Ramp Curve Fit'
xlabel 'Cam Angle (deg)'
ylabel 'Lift (mm)'
axis tight
%% STITCHED CURVES
BCircleUp = zeros(96,1);
BCircleDown = zeros(95,1);
Stitch = [BCircleUp; UpRampCF; MainCF; DownRampCF;
BCircleDown;];

figure (4)
plot(Theta,Stitch)
title 'Combined Curve Fit Model'
xlabel 'Cam Angle (°)'
ylabel 'Cam Lift (mm)'
axis tight
hold on
plot (Theta, RLI)
legend ({'Curve Fit', 'Measured'})

%% STITCHED FAULTS TO FIX BY REPLACING CORRUPTION WITH
NaN's or Zeros
```

```
% StitchPatch = xlsread ('Cam LIFT' , 'raw inlet' ,
'J3:J362');
%% FIXING STITCH CORRUPTIONS FROM JOINS FROM XLS READ
SCRIPT

StitchComplete = fillmissing(StitchPatch, 'pchip');

figure (5)
plot(Theta,StitchComplete)
title 'Combined Curve Fit Model (Interpolated)'
xlabel 'Cam Angle (°)'
ylabel 'Cam Lift (mm)'
axis tight
hold on
plot (Theta, RLI)
legend ({'Curve Fit', 'Measured'})
```

16.12.3 Example Matrix Script (3,600 x 3,600)

```
%% VALVE LIFT MATRIX (3,600 x 3,600) [0.1 DEGREE
INCREMENTS]
ThetaC = (0 :0.1: 359.9)'; % 0.1 deg steps
PhiC = (0 :0.1: 359.9); % 0.1 deg steps

% Extend 360 values to 3600 values

Vi = RLI'; % Original 360 ROW VECTOR
[ri,ci] = size(Vi);
p = 9; %number of NaN to be inserted between each values
of V
RawVLift3600 = NaN(1,ci*(p+1)-p);
[rf,cf] = size(RawVLift3600);
k = 1;
for i = 1:p+1:cf
    RawVLift3600(i) = Vi(k);
    k = k+1;
end
RawVLift3600; % RESULTANT EXTENDED VECTOR
RawVLift3600(1,3592:3600) = RawVLift3600(1,3591); %
RESULTANT EXTENDED VECTOR

RawVLift3600 = fillmissing(RawVLift3600, 'pchip');
%INTERPOLATION

% Rearranged cam lift so max values are on outside
L = [RawVLift3600(1801:3600) RawVLift3600(1:1800)]';  %
Replaces VALUES for size
S = LRI + L(:,1); % Adds base circle radius to lift

% Create the matrix from above values
RawMat3600 = S .* (cosd((180-(ThetaC-PhiC))));

%% Max from Matrix

% Find maximum values from each column in MATRIX
RawMax3600 = max(RawMat3600)';

%% [ ONLY MATRICES >360 ] Find maximum values per degree
(from max vector)

% check ====> [ADD 16.5 to VLift (1,1) AFTER run]

RawVLift3600 = zeros(360,1);
i = 0
```

```
l = 1
for t = 1:10:3590    % change this counter depending on
matrix size
    y = t+9        % change this counter depending on
matrix size
    l = l+1

    RawVLift3600(l,:) = max(RawMax3600(t:y,:));
end
```

16.12.4 Example of Calculation Script Including Valve Derivatives, Friction Model, Rotation of Axes

```
%% EXCEL READ FOR KENT CAMS ORIGINAL INLET CAMSHAFT
PROFILE

KentVlift = xlsread ('Comparison Spreadsheet INL' , 'Kent
Cams' , 'B1:B360');
load 'Inlet Constants'

KentVlift = KentVlift-SCI;
KentVlift(KentVlift<0) = 0;
%% PHI ANGLE

Phi = 0:1:359; % Degree Cam Angle Vector


%% VALVE VELOCITY

KentVlift2 = circshift(KentVlift,-1);

KentVvel = KentVlift2 - KentVlift;


%% VALVE ACCELERATION

KentVvel2 = circshift(KentVvel,-1);

KentVacc = KentVvel2 - KentVvel;

%% VALVE JERK

KentVacc2 = circshift(KentVacc,-1);

KentVjer = KentVacc2 - KentVacc;


%% VALVE JOUNCE

KentVjer2 = circshift(KentVjer,-1);

KentVjou = KentVjer2 - KentVjer;
```

```
%% *************** ROTATION OF AXIS TO GET CAM PROFILE
***************** %%
% ------------------------------------------------------
---------------- %

%% small x & y from LOCAL axis

Kentlocalx = ((360)/(2*pi)) * KentVvel' ;  % Vertical
Valve Velocity

Kentlocaly = KentVlift' + LRI + SCI + SCI-0.5; % Lift &
Lobe Radius + Shim Clearance

%% large X and Y from GLOBAL axis

KentGlobalX = (Kentlocalx.*cosd(Phi)) +
(Kentlocaly.*sind(Phi)); % Reverse of rotation derivation

KentGlobalY = (Kentlocaly.*cosd(Phi)) -
(Kentlocalx.*sind(Phi)); % Reverse of rotation derivation

%% Camshaft Profile from Rotation of Axis

figure (7)

plot(KentGlobalX,KentGlobalY)
title 'Kent Cams Inlet Camshaft Profile'
xlabel 'Global X Coordinates'
ylabel 'Global Y Coordinates'
axis tight

%% ************************ FRICTION MODEL
************************** %%
% ------------------------------------------------------
---------------- %
%% ENGINE SPEED VECTOR

RPM = [1000 2000 3000 4000 5000 6000 6500 6600 6700 6800
6900 7000 7100 7200 7300 7400 7500];
%%  VELOCITY [m/s]

% Original values are mm/degree

% Velocity Per RPM
KentVvelRPM = (KentVvel*(6*RPM) / 1000);

%% Velocity per engine speed plot
```

```
%% ACCELERATION [m/s^2]

%Original values are mm/degree/degree

% Acceleration Per RPM
KentVaccRPM = (KentVacc*(6*RPM.^2) / 1000);


%% ACCELERATION AT FULL LIFT [m/s^2]

% Acceleration where valve is at maximum opening point
[MOP]
KentFLVacc = abs(min(KentVacc));

%% ENGINE SPEED THAT CONTACT FORCE EQUALS ZERO [rpm]

% Engine Speed That Contact Force Equals Zero (Valve
Float)
KentFLOAT = (sqrt((IMVL*1000)/(ITMM*KentFLVacc*36)))*2;

%% CONTACT FORCE [N]

% Contact Force per Degree, per Engine Speed
KentFCRPM = (ITMM*KentVaccRPM) +
(IVSC*(ISCD+(KentVlift/1000)));

%% CALCULATING LOBE HEIGHT PER DEGREE (notation H) [mm]

% Lobe Height
KentH = KentVlift + LRI;

%% FRICTION FORCE [N]

% Friction Force per Degree, per Engine Speed
KentFFRPM = KentFCRPM * ICF;


%% FRICTION TORQUE [Nm]

% Friction Torque per Degree, per Engine Speed
KentFTRPM = KentFFRPM .* (KentH/1000);


%% Friction WD per degree, per engine speed [J]

% Finding Friction Torque 2
KentFT2RPM = circshift(KentFTRPM,-1);
```

```
% Friction WD per Degree, per Engine Speed
KentFWDRPM = ((KentFTRPM + KentFT2RPM)/2) .* (2*pi/360);

%% Total Friction Work Done [J]

% Total Friction WD per Engine Speed
KentTFWDRPM = abs(sum(KentFWDRPM));

%% Total Power Loss Due To Friction [W]

% Total Power Loss per Engine Speed
KentTPLRPM = (8*KentTFWDRPM) .* (RPM/120);


%% Spring Surge Frequency

% This calculates the 1st resonant frequency for the
spring. For safe
% spring design, the operation frequencies should be 13
to 20 times less
% than this calculated frequency.

SpringSF = 1/2 * sqrt(IVSC/ISM);

SpringSFmin = SpringSF / 13;
```

16.12.5 Spring Surge Calculator with 3D Output

```
%% Spring Surge Frequency

% This calculates the 1st resonant frequency for the
spring. For safe
% spring design, the operation frequencies should be 13
to 20 times less
% than this calculated frequency.

IVSC = 39481; % Inlet Valve Spring Constant
ISM = 0.0347; % Inlet Valve Spring Mass

SpringSF = 1/2 * sqrt(IVSC/ISM);

SpringSFmin = SpringSF / 13; % where Denominator is
SAFETY FACTOR

%% FREQUENCY PER ORDER, PER ENGINE SPEED

RPM = (1000:50:15000); % ADJUSTABLE FOR ENGINE TYPE
**SMALL INCREMENTS**

DegSec = (RPM/2) / 60; % RPM to Deg/sec conversion

Order = (1:0.5:36)'; % ADJUSTABLE FOR ORDER RANGE **SMALL
INCREMENTS**
 OrderFreq = Order * DegSec; % Resultant Frequency

% Make colors change depending on values.
redChannel = OrderFreq <= 543;
greenChannel = OrderFreq <= 523;
blueChannel = OrderFreq > 543;
% Make the RGB image.
colors = double(cat(3, redChannel, greenChannel,
blueChannel));
% Plot the surface with those colors.
h = surf(DegSec,Order,OrderFreq, colors);
% Enlarge figure to full screen.
set(gcf, 'units','normalized','outerposition',[0 0 1 1]);
axis tight
set(h, 'edgecolor','none')
title 'Frequency Analysis with Respect to Spring Surge
Frequency'
xlabel 'Camshaft Speed (rps [Hz])'
ylabel 'Order'
zlabel 'Order Frequency (Hz)'
```

16.12.6 Fourier Analysis Script

```
%% Fourier Analysis

%% lOAD KENT VALVE LIFT

KentVlift = xlsread ('Comparison Spreadsheet INL' , 'Kent
Cams' , 'B1:B360');
Theta = (0:1:359)';

%% a Constants

Kenta0 = KentVlift*(2*pi/360);
Kenta0 = (1/(2*pi)*sum(Kenta0));

Kenta1 = KentVlift.*cosd(Theta).*(2*pi/360);
Kenta1 = (1/pi)*sum(Kenta1);

Kenta2 = KentVlift.*cosd(2*Theta).*(2*pi/360);
Kenta2 = (1/pi)*sum(Kenta2);

Kenta3 = KentVlift.*cosd(3*Theta).*(2*pi/360);
Kenta3 = (1/pi)*sum(Kenta3);

Kenta4 = KentVlift.*cosd(4*Theta).*(2*pi/360);
Kenta4 = (1/pi)*sum(Kenta4);

Kenta5 = KentVlift.*cosd(5*Theta).*(2*pi/360);
Kenta5 = (1/pi)*sum(Kenta5);

Kenta6 = KentVlift.*cosd(6*Theta).*(2*pi/360);
Kenta6 = (1/pi)*sum(Kenta6);

Kenta7 = KentVlift.*cosd(7*Theta).*(2*pi/360);
Kenta7 = (1/pi)*sum(Kenta7);

Kenta8 = KentVlift.*cosd(8*Theta).*(2*pi/360);
Kenta8 = (1/pi)*sum(Kenta8);

Kenta9 = KentVlift.*cosd(9*Theta).*(2*pi/360);
Kenta9 = (1/pi)*sum(Kenta9);

Kenta10 = KentVlift.*cosd(10*Theta).*(2*pi/360);
Kenta10 = (1/pi)*sum(Kenta10);

Kenta11 = KentVlift.*cosd(11*Theta).*(2*pi/360);
Kenta11 = (1/pi)*sum(Kenta11);
```

```
Kenta12 = KentVlift.*cosd(12*Theta).*(2*pi/360);
Kenta12 = (1/pi)*sum(Kenta12);

Kenta13 = KentVlift.*cosd(13*Theta).*(2*pi/360);
Kenta13 = (1/pi)*sum(Kenta13);

Kenta14 = KentVlift.*cosd(14*Theta).*(2*pi/360);
Kenta14 = (1/pi)*sum(Kenta14);

Kenta15 = KentVlift.*cosd(15*Theta).*(2*pi/360);
Kenta15 = (1/pi)*sum(Kenta15);

Kenta16 = KentVlift.*cosd(16*Theta).*(2*pi/360);
Kenta16 = (1/pi)*sum(Kenta16);

Kenta17 = KentVlift.*cosd(17*Theta).*(2*pi/360);
Kenta17 = (1/pi)*sum(Kenta17);

Kenta18 = KentVlift.*cosd(18*Theta).*(2*pi/360);
Kenta18 = (1/pi)*sum(Kenta18);

Kenta19 = KentVlift.*cosd(19*Theta).*(2*pi/360);
Kenta19 = (1/pi)*sum(Kenta19);

Kenta20 = KentVlift.*cosd(20*Theta).*(2*pi/360);
Kenta20 = (1/pi)*sum(Kenta20);

Kenta21= KentVlift.*cosd(21*Theta).*(2*pi/360);
Kenta21 = (1/pi)*sum(Kenta21);

Kenta22 = KentVlift.*cosd(22*Theta).*(2*pi/360);
Kenta22 = (1/pi)*sum(Kenta22);

Kenta23 = KentVlift.*cosd(23*Theta).*(2*pi/360);
Kenta23 = (1/pi)*sum(Kenta23);

Kenta24 = KentVlift.*cosd(24*Theta).*(2*pi/360);
Kenta24 = (1/pi)*sum(Kenta24);

Kenta25 = KentVlift.*cosd(25*Theta).*(2*pi/360);
Kenta25 = (1/pi)*sum(Kenta25);

Kenta26 = KentVlift.*cosd(26*Theta).*(2*pi/360);
Kenta26 = (1/pi)*sum(Kenta26);

Kenta27 = KentVlift.*cosd(27*Theta).*(2*pi/360);
Kenta27 = (1/pi)*sum(Kenta27);
```

```
Kenta28 = KentVlift.*cosd(28*Theta).*(2*pi/360);
Kenta28 = (1/pi)*sum(Kenta28);

Kenta29 = KentVlift.*cosd(29*Theta).*(2*pi/360);
Kenta29 = (1/pi)*sum(Kenta29);

Kenta30 = KentVlift.*cosd(30*Theta).*(2*pi/360);
Kenta30 = (1/pi)*sum(Kenta30);

Kenta31 = KentVlift.*cosd(31*Theta).*(2*pi/360);
Kenta31 = (1/pi)*sum(Kenta31);

Kenta32 = KentVlift.*cosd(32*Theta).*(2*pi/360);
Kenta32 = (1/pi)*sum(Kenta32);

Kenta33 = KentVlift.*cosd(33*Theta).*(2*pi/360);
Kenta33 = (1/pi)*sum(Kenta33);

Kenta34 = KentVlift.*cosd(34*Theta).*(2*pi/360);
Kenta34 = (1/pi)*sum(Kenta34);

Kenta35 = KentVlift.*cosd(35*Theta).*(2*pi/360);
Kenta35 = (1/pi)*sum(Kenta35);

Kenta36 = KentVlift.*cosd(36*Theta).*(2*pi/360);
Kenta36 = (1/pi)*sum(Kenta36);
```

```
%% b constants

Kentb1 = KentVlift.*sind(Theta).*(2*pi/360);
Kentb1 = (1/pi)*sum(Kentb1);

Kentb2 = KentVlift.*sind(2*Theta).*(2*pi/360);
Kentb2 = (1/pi)*sum(Kentb2);

Kentb3 = KentVlift.*sind(3*Theta).*(2*pi/360);
Kentb3 = (1/pi)*sum(Kentb3);

Kentb4 = KentVlift.*sind(4*Theta).*(2*pi/360);
Kentb4 = (1/pi)*sum(Kentb4);

Kentb5 = KentVlift.*sind(5*Theta).*(2*pi/360);
Kentb5 = (1/pi)*sum(Kentb5);

Kentb6 = KentVlift.*sind(6*Theta).*(2*pi/360);
Kentb6 = (1/pi)*sum(Kentb6);

Kentb7 = KentVlift.*sind(7*Theta).*(2*pi/360);
Kentb7 = (1/pi)*sum(Kentb7);

Kentb8 = KentVlift.*sind(8*Theta).*(2*pi/360);
Kentb8 = (1/pi)*sum(Kentb8);

Kentb9 = KentVlift.*sind(9*Theta).*(2*pi/360);
Kentb9 = (1/pi)*sum(Kentb9);

Kentb10 = KentVlift.*sind(10*Theta).*(2*pi/360);
Kentb10 = (1/pi)*sum(Kentb10);

Kentb11 = KentVlift.*sind(11*Theta).*(2*pi/360);
Kentb11 = (1/pi)*sum(Kentb11);

Kentb12 = KentVlift.*sind(12*Theta).*(2*pi/360);
Kentb12 = (1/pi)*sum(Kentb12);

Kentb13 = KentVlift.*sind(13*Theta).*(2*pi/360);
Kentb13 = (1/pi)*sum(Kentb13);

Kentb14 = KentVlift.*sind(14*Theta).*(2*pi/360);
Kentb14 = (1/pi)*sum(Kentb14);

Kentb15 = KentVlift.*sind(15*Theta).*(2*pi/360);
Kentb15 = (1/pi)*sum(Kentb15);

Kentb16 = KentVlift.*sind(16*Theta).*(2*pi/360);
```

```
Kentb16 = (1/pi)*sum(Kentb16);

Kentb17 = KentVlift.*sind(17*Theta).*(2*pi/360);
Kentb17 = (1/pi)*sum(Kentb17);

Kentb18 = KentVlift.*sind(18*Theta).*(2*pi/360);
Kentb18 = (1/pi)*sum(Kentb18);

Kentb19 = KentVlift.*sind(19*Theta).*(2*pi/360);
Kentb19 = (1/pi)*sum(Kentb19);

Kentb20 = KentVlift.*sind(20*Theta).*(2*pi/360);
Kentb20 = (1/pi)*sum(Kentb20);

Kentb21 = KentVlift.*sind(21*Theta).*(2*pi/360);
Kentb21 = (1/pi)*sum(Kentb21);

Kentb22 = KentVlift.*sind(22*Theta).*(2*pi/360);
Kentb22 = (1/pi)*sum(Kentb22);

Kentb23 = KentVlift.*sind(23*Theta).*(2*pi/360);
Kentb23 = (1/pi)*sum(Kentb23);

Kentb24 = KentVlift.*sind(24*Theta).*(2*pi/360);
Kentb24 = (1/pi)*sum(Kentb24);

Kentb25 = KentVlift.*sind(25*Theta).*(2*pi/360);
Kentb25 = (1/pi)*sum(Kentb25);

Kentb26 = KentVlift.*sind(26*Theta).*(2*pi/360);
Kentb26 = (1/pi)*sum(Kentb26);

Kentb27 = KentVlift.*sind(27*Theta).*(2*pi/360);
Kentb27 = (1/pi)*sum(Kentb27);

Kentb28 = KentVlift.*sind(28*Theta).*(2*pi/360);
Kentb28 = (1/pi)*sum(Kentb28);

Kentb29 = KentVlift.*sind(29*Theta).*(2*pi/360);
Kentb29 = (1/pi)*sum(Kentb29);

Kentb30 = KentVlift.*sind(30*Theta).*(2*pi/360);
Kentb30 = (1/pi)*sum(Kentb30);

Kentb31 = KentVlift.*sind(31*Theta).*(2*pi/360);
Kentb31 = (1/pi)*sum(Kentb31);

Kentb32 = KentVlift.*sind(32*Theta).*(2*pi/360);
```

```
Kentb32 = (1/pi)*sum(Kentb32);

Kentb33 = KentVlift.*sind(33*Theta).*(2*pi/360);
Kentb33 = (1/pi)*sum(Kentb33);

Kentb34 = KentVlift.*sind(34*Theta).*(2*pi/360);
Kentb34 = (1/pi)*sum(Kentb34);

Kentb35 = KentVlift.*sind(35*Theta).*(2*pi/360);
Kentb35 = (1/pi)*sum(Kentb35);

Kentb36 = KentVlift.*sind(36*Theta).*(2*pi/360);
Kentb36 = (1/pi)*sum(Kentb36);
```

```
%% function Cumulative Coefficients

KentFun1 = Kenta0 + Kenta1.*cosd(Theta) +
Kentb1.*sind(Theta);
KentFun2 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta);
KentFun3 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta);
KentFun4 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta);
KentFun5 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta);
KentFun6 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta);
KentFun7 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta);
KentFun8 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta);
```

```
KentFun9 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta);
KentFun10 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
Kentb10.*sind(10*Theta);
KentFun11 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta);
KentFun12 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
Kentb10.*sind(10*Theta) + Kentb11.*sind(11*Theta) +
Kentb12.*sind(12*Theta);
```

```
KentFun13 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta);
KentFun14 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
Kentb10.*sind(10*Theta) + Kentb11.*sind(11*Theta) +
Kentb12.*sind(12*Theta) + Kentb13.*sind(13*Theta) +
Kentb14.*sind(14*Theta);
KentFun15 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta);
KentFun16 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
```

```
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
Kentb10.*sind(10*Theta) + Kentb11.*sind(11*Theta) +
Kentb12.*sind(12*Theta) + Kentb13.*sind(13*Theta) +
Kentb14.*sind(14*Theta) + Kentb15.*sind(15*Theta) +
Kentb16.*sind(16*Theta);
KentFun17 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta);
KentFun18 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
```

```
Kentb10.*sind(10*Theta) + Kentb11.*sind(11*Theta) +
Kentb12.*sind(12*Theta) + Kentb13.*sind(13*Theta) +
Kentb14.*sind(14*Theta) + Kentb15.*sind(15*Theta) +
Kentb16.*sind(16*Theta) + Kentb17.*sind(17*Theta) +
Kentb18.*sind(18*Theta);
KentFun19 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta);
KentFun20 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
Kentb10.*sind(10*Theta) + Kentb11.*sind(11*Theta) +
Kentb12.*sind(12*Theta) + Kentb13.*sind(13*Theta) +
Kentb14.*sind(14*Theta) + Kentb15.*sind(15*Theta) +
Kentb16.*sind(16*Theta) + Kentb17.*sind(17*Theta) +
Kentb18.*sind(18*Theta) + Kentb19.*sind(19*Theta) +
Kentb20.*sind(20*Theta);
```

```
KentFun21 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta);
KentFun22 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
Kentb10.*sind(10*Theta) + Kentb11.*sind(11*Theta) +
Kentb12.*sind(12*Theta) + Kentb13.*sind(13*Theta) +
Kentb14.*sind(14*Theta) + Kentb15.*sind(15*Theta) +
Kentb16.*sind(16*Theta) + Kentb17.*sind(17*Theta) +
Kentb18.*sind(18*Theta) + Kentb19.*sind(19*Theta) +
Kentb20.*sind(20*Theta) + Kentb21.*sind(21*Theta) +
Kentb22.*sind(22*Theta);
KentFun23 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
```

```
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta);
KentFun24 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
Kentb10.*sind(10*Theta) + Kentb11.*sind(11*Theta) +
Kentb12.*sind(12*Theta) + Kentb13.*sind(13*Theta) +
Kentb14.*sind(14*Theta) + Kentb15.*sind(15*Theta) +
Kentb16.*sind(16*Theta) + Kentb17.*sind(17*Theta) +
Kentb18.*sind(18*Theta) + Kentb19.*sind(19*Theta) +
Kentb20.*sind(20*Theta) + Kentb21.*sind(21*Theta) +
Kentb22.*sind(22*Theta) + Kentb23.*sind(23*Theta) +
Kentb24.*sind(24*Theta);
```

```
KentFun25 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta);
KentFun26 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
Kentb10.*sind(10*Theta) + Kentb11.*sind(11*Theta) +
Kentb12.*sind(12*Theta) + Kentb13.*sind(13*Theta) +
Kentb14.*sind(14*Theta) + Kentb15.*sind(15*Theta) +
```

```
Kentb16.*sind(16*Theta) + Kentb17.*sind(17*Theta) +
Kentb18.*sind(18*Theta) + Kentb19.*sind(19*Theta) +
Kentb20.*sind(20*Theta) + Kentb21.*sind(21*Theta) +
Kentb22.*sind(22*Theta) + Kentb23.*sind(23*Theta) +
Kentb24.*sind(24*Theta) + Kentb25.*sind(25*Theta) +
Kentb26.*sind(26*Theta);
KentFun27 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta);
KentFun28 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
```

```
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
Kentb10.*sind(10*Theta) + Kentb11.*sind(11*Theta) +
Kentb12.*sind(12*Theta) + Kentb13.*sind(13*Theta) +
Kentb14.*sind(14*Theta) + Kentb15.*sind(15*Theta) +
Kentb16.*sind(16*Theta) + Kentb17.*sind(17*Theta) +
Kentb18.*sind(18*Theta) + Kentb19.*sind(19*Theta) +
Kentb20.*sind(20*Theta) + Kentb21.*sind(21*Theta) +
Kentb22.*sind(22*Theta) + Kentb23.*sind(23*Theta) +
Kentb24.*sind(24*Theta) + Kentb25.*sind(25*Theta) +
Kentb26.*sind(26*Theta) + Kentb27.*sind(27*Theta) +
Kentb28.*sind(28*Theta);
KentFun29 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta) + Kentb28.*sind(28*Theta) +
Kentb29.*sind(29*Theta);
```

```
KentFun30 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
Kentb10.*sind(10*Theta) + Kentb11.*sind(11*Theta) +
Kentb12.*sind(12*Theta) + Kentb13.*sind(13*Theta) +
Kentb14.*sind(14*Theta) + Kentb15.*sind(15*Theta) +
Kentb16.*sind(16*Theta) + Kentb17.*sind(17*Theta) +
Kentb18.*sind(18*Theta) + Kentb19.*sind(19*Theta) +
Kentb20.*sind(20*Theta) + Kentb21.*sind(21*Theta) +
Kentb22.*sind(22*Theta) + Kentb23.*sind(23*Theta) +
Kentb24.*sind(24*Theta) + Kentb25.*sind(25*Theta) +
Kentb26.*sind(26*Theta) + Kentb27.*sind(27*Theta) +
Kentb28.*sind(28*Theta) + Kentb29.*sind(29*Theta) +
Kentb30.*sind(30*Theta);
KentFun31 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta) + Kenta31.*cosd(31*Theta) +
```

```
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta) + Kentb28.*sind(28*Theta) +
Kentb29.*sind(29*Theta) + Kentb30.*sind(30*Theta) +
Kentb31.*sind(31*Theta);
KentFun32 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta) + Kenta31.*cosd(31*Theta) +
Kenta32.*cosd(32*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
Kentb10.*sind(10*Theta) + Kentb11.*sind(11*Theta) +
Kentb12.*sind(12*Theta) + Kentb13.*sind(13*Theta) +
Kentb14.*sind(14*Theta) + Kentb15.*sind(15*Theta) +
Kentb16.*sind(16*Theta) + Kentb17.*sind(17*Theta) +
Kentb18.*sind(18*Theta) + Kentb19.*sind(19*Theta) +
Kentb20.*sind(20*Theta) + Kentb21.*sind(21*Theta) +
Kentb22.*sind(22*Theta) + Kentb23.*sind(23*Theta) +
Kentb24.*sind(24*Theta) + Kentb25.*sind(25*Theta) +
Kentb26.*sind(26*Theta) + Kentb27.*sind(27*Theta) +
Kentb28.*sind(28*Theta) + Kentb29.*sind(29*Theta) +
```

```
Kentb30.*sind(30*Theta) + Kentb31.*sind(31*Theta) +
Kentb32.*sind(32*Theta);
KentFun33 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta) + Kenta31.*cosd(31*Theta) +
Kenta32.*cosd(32*Theta) + Kenta33.*cosd(33*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta) + Kentb28.*sind(28*Theta) +
Kentb29.*sind(29*Theta) + Kentb30.*sind(30*Theta) +
Kentb31.*sind(31*Theta) + Kentb32.*sind(32*Theta) +
Kentb33.*sind(33*Theta);
KentFun34 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
```

```
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta) + Kenta31.*cosd(31*Theta) +
Kenta32.*cosd(32*Theta) + Kenta33.*cosd(33*Theta) +
Kenta34.*cosd(34*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
Kentb10.*sind(10*Theta) + Kentb11.*sind(11*Theta) +
Kentb12.*sind(12*Theta) + Kentb13.*sind(13*Theta) +
Kentb14.*sind(14*Theta) + Kentb15.*sind(15*Theta) +
Kentb16.*sind(16*Theta) + Kentb17.*sind(17*Theta) +
Kentb18.*sind(18*Theta) + Kentb19.*sind(19*Theta) +
Kentb20.*sind(20*Theta) + Kentb21.*sind(21*Theta) +
Kentb22.*sind(22*Theta) + Kentb23.*sind(23*Theta) +
Kentb24.*sind(24*Theta) + Kentb25.*sind(25*Theta) +
Kentb26.*sind(26*Theta) + Kentb27.*sind(27*Theta) +
Kentb28.*sind(28*Theta) + Kentb29.*sind(29*Theta) +
Kentb30.*sind(30*Theta) + Kentb31.*sind(31*Theta) +
Kentb32.*sind(32*Theta) + Kentb33.*sind(33*Theta) +
Kentb34.*sind(34*Theta);
KentFun35 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta) + Kenta31.*cosd(31*Theta) +
Kenta32.*cosd(32*Theta) + Kenta33.*cosd(33*Theta) +
Kenta34.*cosd(34*Theta) + Kenta35.*cosd(35*Theta) +
Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
```

```
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta) + Kentb28.*sind(28*Theta) +
Kentb29.*sind(29*Theta) + Kentb30.*sind(30*Theta) +
Kentb31.*sind(31*Theta) + Kentb32.*sind(32*Theta) +
Kentb33.*sind(33*Theta) + Kentb34.*sind(34*Theta) +
Kentb35.*sind(35*Theta);
KentFun36 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta) + Kenta31.*cosd(31*Theta) +
Kenta32.*cosd(32*Theta) + Kenta33.*cosd(33*Theta) +
Kenta34.*cosd(34*Theta) + Kenta35.*cosd(35*Theta) +
Kenta36.*cosd(36*Theta) + Kentb1.*sind(Theta) +
Kentb2.*sind(2*Theta) + Kentb3.*sind(3*Theta) +
Kentb4.*sind(4*Theta) + Kentb5.*sind(5*Theta) +
Kentb6.*sind(6*Theta) + Kentb7.*sind(7*Theta) +
Kentb8.*sind(8*Theta) + Kentb9.*sind(9*Theta) +
Kentb10.*sind(10*Theta) + Kentb11.*sind(11*Theta) +
Kentb12.*sind(12*Theta) + Kentb13.*sind(13*Theta) +
Kentb14.*sind(14*Theta) + Kentb15.*sind(15*Theta) +
Kentb16.*sind(16*Theta) + Kentb17.*sind(17*Theta) +
Kentb18.*sind(18*Theta) + Kentb19.*sind(19*Theta) +
Kentb20.*sind(20*Theta) + Kentb21.*sind(21*Theta) +
Kentb22.*sind(22*Theta) + Kentb23.*sind(23*Theta) +
Kentb24.*sind(24*Theta) + Kentb25.*sind(25*Theta) +
Kentb26.*sind(26*Theta) + Kentb27.*sind(27*Theta) +
Kentb28.*sind(28*Theta) + Kentb29.*sind(29*Theta) +
Kentb30.*sind(30*Theta) + Kentb31.*sind(31*Theta) +
```

```
Kentb32.*sind(32*Theta) + Kentb33.*sind(33*Theta) +
Kentb34.*sind(34*Theta) + Kentb35.*sind(35*Theta) +
Kentb36.*sind(36*Theta);
```

```
%% function Cosine (even) Coefficients

KentFuna1 = Kenta0 + Kenta1.*cosd(Theta);
KentFuna2 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta);
KentFuna3 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta);
KentFuna4 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta);
KentFuna5 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta);
KentFuna6 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta);
KentFuna7 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta);
KentFuna8 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta);
KentFuna9 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta);
KentFuna10 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta);
KentFuna11 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta);
KentFuna12 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
```

```
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta);
KentFuna13 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta);
KentFuna14 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta);
KentFuna15 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta);
KentFuna16 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta);
KentFuna17 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta);
KentFuna18 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
```

```
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta);
KentFuna19 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta);
KentFuna20 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta);
KentFuna21 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta);
KentFuna22 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
```

```
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta);
KentFuna23 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta);
KentFuna24 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta);
KentFuna25 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta);
KentFuna26 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
```

```
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta);
KentFuna27 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta);
KentFuna28 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta);
KentFuna29 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
```

```
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta);
KentFuna30 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta);
KentFuna31 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta) + Kenta31.*cosd(31*Theta);
KentFuna32 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
```

```
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta) + Kenta31.*cosd(31*Theta) +
Kenta32.*cosd(32*Theta);
KentFuna33 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta) + Kenta31.*cosd(31*Theta) +
Kenta32.*cosd(32*Theta) + Kenta33.*cosd(33*Theta);
KentFuna34 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta) + Kenta31.*cosd(31*Theta) +
Kenta32.*cosd(32*Theta) + Kenta33.*cosd(33*Theta) +
Kenta34.*cosd(34*Theta);
KentFuna35 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
```

```
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta) + Kenta31.*cosd(31*Theta) +
Kenta32.*cosd(32*Theta) + Kenta33.*cosd(33*Theta) +
Kenta34.*cosd(34*Theta) + Kenta35.*cosd(35*Theta);
KentFuna36 = Kenta0 + Kenta1.*cosd(Theta) +
Kenta2.*cosd(2*Theta) + Kenta3.*cosd(3*Theta) +
Kenta4.*cosd(4*Theta) + Kenta5.*cosd(5*Theta) +
Kenta6.*cosd(6*Theta) + Kenta7.*cosd(7*Theta) +
Kenta8.*cosd(8*Theta) + Kenta9.*cosd(9*Theta) +
Kenta10.*cosd(10*Theta) + Kenta11.*cosd(11*Theta) +
Kenta12.*cosd(12*Theta) + Kenta13.*cosd(13*Theta) +
Kenta14.*cosd(14*Theta) + Kenta15.*cosd(15*Theta) +
Kenta16.*cosd(16*Theta) + Kenta17.*cosd(17*Theta) +
Kenta18.*cosd(18*Theta) + Kenta19.*cosd(19*Theta) +
Kenta20.*cosd(20*Theta) + Kenta21.*cosd(21*Theta) +
Kenta22.*cosd(22*Theta) + Kenta23.*cosd(23*Theta) +
Kenta24.*cosd(24*Theta) + Kenta25.*cosd(25*Theta) +
Kenta26.*cosd(26*Theta) + Kenta27.*cosd(27*Theta) +
Kenta28.*cosd(28*Theta) + Kenta29.*cosd(29*Theta) +
Kenta30.*cosd(30*Theta) + Kenta31.*cosd(31*Theta) +
Kenta32.*cosd(32*Theta) + Kenta33.*cosd(33*Theta) +
Kenta34.*cosd(34*Theta) + Kenta35.*cosd(35*Theta) +
Kenta36.*cosd(36*Theta);
```

```
%% function Sine (odd) Coefficients

KentFunb1 = Kentb1.*sind(Theta);
KentFunb2 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta);
KentFunb3 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta);
KentFunb4 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta);
KentFunb5 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta);
KentFunb6 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta) +
Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta);
KentFunb7 =  Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta);
KentFunb8 =  Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta);
KentFunb9 =  Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta);
KentFunb10 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta);
KentFunb11 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta);
KentFunb12 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta);
KentFunb13 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
```

```
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta);
KentFunb14 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta);
KentFunb15 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta);
KentFunb16 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta);
KentFunb17 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta);
KentFunb18 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta);
KentFunb19 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
```

```
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta);
KentFunb20 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta);
KentFunb21 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta);
KentFunb22 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta);
KentFunb23 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
```

```
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta);
KentFunb24 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta);
KentFunb25 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta);
KentFunb26 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta);
KentFunb27 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
```

```
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta);
KentFunb28 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta) + Kentb28.*sind(28*Theta);
KentFunb29 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta) + Kentb28.*sind(28*Theta) +
Kentb29.*sind(29*Theta);
KentFunb30 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
```

```
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta) + Kentb28.*sind(28*Theta) +
Kentb29.*sind(29*Theta) + Kentb30.*sind(30*Theta);
KentFunb31 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta) + Kentb28.*sind(28*Theta) +
Kentb29.*sind(29*Theta) + Kentb30.*sind(30*Theta) +
Kentb31.*sind(31*Theta);
KentFunb32 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta) + Kentb28.*sind(28*Theta) +
Kentb29.*sind(29*Theta) + Kentb30.*sind(30*Theta) +
Kentb31.*sind(31*Theta) + Kentb32.*sind(32*Theta);
KentFunb33 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
```

```
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta) + Kentb28.*sind(28*Theta) +
Kentb29.*sind(29*Theta) + Kentb30.*sind(30*Theta) +
Kentb31.*sind(31*Theta) + Kentb32.*sind(32*Theta) +
Kentb33.*sind(33*Theta);
KentFunb34 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta) + Kentb28.*sind(28*Theta) +
Kentb29.*sind(29*Theta) + Kentb30.*sind(30*Theta) +
Kentb31.*sind(31*Theta) + Kentb32.*sind(32*Theta) +
Kentb33.*sind(33*Theta) + Kentb34.*sind(34*Theta);
KentFunb35 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta) + Kentb28.*sind(28*Theta) +
Kentb29.*sind(29*Theta) + Kentb30.*sind(30*Theta) +
Kentb31.*sind(31*Theta) + Kentb32.*sind(32*Theta) +
Kentb33.*sind(33*Theta) + Kentb34.*sind(34*Theta) +
Kentb35.*sind(35*Theta);
KentFunb36 = Kentb1.*sind(Theta) + Kentb2.*sind(2*Theta)
+ Kentb3.*sind(3*Theta) + Kentb4.*sind(4*Theta) +
Kentb5.*sind(5*Theta) + Kentb6.*sind(6*Theta) +
```

```
Kentb7.*sind(7*Theta) + Kentb8.*sind(8*Theta) +
Kentb9.*sind(9*Theta) + Kentb10.*sind(10*Theta) +
Kentb11.*sind(11*Theta) + Kentb12.*sind(12*Theta) +
Kentb13.*sind(13*Theta) + Kentb14.*sind(14*Theta) +
Kentb15.*sind(15*Theta) + Kentb16.*sind(16*Theta) +
Kentb17.*sind(17*Theta) + Kentb18.*sind(18*Theta) +
Kentb19.*sind(19*Theta) + Kentb20.*sind(20*Theta) +
Kentb21.*sind(21*Theta) + Kentb22.*sind(22*Theta) +
Kentb23.*sind(23*Theta) + Kentb24.*sind(24*Theta) +
Kentb25.*sind(25*Theta) + Kentb26.*sind(26*Theta) +
Kentb27.*sind(27*Theta) + Kentb28.*sind(28*Theta) +
Kentb29.*sind(29*Theta) + Kentb30.*sind(30*Theta) +
Kentb31.*sind(31*Theta) + Kentb32.*sind(32*Theta) +
Kentb33.*sind(33*Theta) + Kentb34.*sind(34*Theta) +
Kentb35.*sind(35*Theta) + Kentb36.*sind(36*Theta);
```

```
% CONSTITUENT a HARMONICS

Kentaa0 = Kenta0;
Kentaa1 = Kenta1.*cosd(Theta);
Kentaa2 = Kenta2.*cosd(2*Theta);
Kentaa3 = Kenta3.*cosd(3*Theta);
Kentaa4 = Kenta4.*cosd(4*Theta);
Kentaa5 = Kenta5.*cosd(5*Theta);
Kentaa6 = Kenta6.*cosd(6*Theta);
Kentaa7 = Kenta7.*cosd(7*Theta);
Kentaa8 = Kenta8.*cosd(8*Theta);
Kentaa9 = Kenta9.*cosd(9*Theta);
Kentaa10 = Kenta10.*cosd(10*Theta);
Kentaa11 = Kenta11.*cosd(11*Theta);
Kentaa12 = Kenta12.*cosd(12*Theta);
Kentaa13 = Kenta13.*cosd(13*Theta);
Kentaa14 = Kenta14.*cosd(14*Theta);
Kentaa15 = Kenta15.*cosd(15*Theta);
Kentaa16 = Kenta16.*cosd(16*Theta);
Kentaa17 = Kenta17.*cosd(17*Theta);
Kentaa18 = Kenta18.*cosd(18*Theta);
Kentaa19 = Kenta19.*cosd(19*Theta);
Kentaa20 = Kenta20.*cosd(20*Theta);
Kentaa21 = Kenta21.*cosd(21*Theta);
Kentaa22 = Kenta22.*cosd(22*Theta);
Kentaa23 = Kenta23.*cosd(23*Theta);
Kentaa24 = Kenta24.*cosd(24*Theta);
Kentaa25 = Kenta25.*cosd(25*Theta);
Kentaa26 = Kenta26.*cosd(26*Theta);
Kentaa27 = Kenta27.*cosd(27*Theta);
Kentaa28 = Kenta28.*cosd(28*Theta);
Kentaa29 = Kenta29.*cosd(29*Theta);
Kentaa30 = Kenta30.*cosd(30*Theta);
Kentaa31 = Kenta31.*cosd(31*Theta);
Kentaa32 = Kenta32.*cosd(32*Theta);
Kentaa33 = Kenta33.*cosd(33*Theta);
Kentaa34 = Kenta34.*cosd(34*Theta);
Kentaa35 = Kenta35.*cosd(35*Theta);
Kentaa36 = Kenta36.*cosd(36*Theta);
```

```
% CONSTITUENT b HARMONICS

Kentbb1 = Kentb1.*sind(Theta);
Kentbb2 = Kentb2.*sind(2*Theta);
Kentbb3 = Kentb3.*sind(3*Theta);
Kentbb4 = Kentb4.*sind(4*Theta);
Kentbb5 = Kentb5.*sind(5*Theta);
Kentbb6 = Kentb6.*sind(6*Theta);
Kentbb7 = Kentb7.*sind(7*Theta);
Kentbb8 = Kentb8.*sind(8*Theta);
Kentbb9 = Kentb9.*sind(9*Theta);
Kentbb10 = Kentb10.*sind(10*Theta);
Kentbb11 = Kentb11.*sind(11*Theta);
Kentbb12 = Kentb12.*sind(12*Theta);
Kentbb13 = Kentb13.*sind(13*Theta);
Kentbb14 = Kentb14.*sind(14*Theta);
Kentbb15 = Kentb15.*sind(15*Theta);
Kentbb16 = Kentb16.*sind(16*Theta);
Kentbb17 = Kentb17.*sind(17*Theta);
Kentbb18 = Kentb18.*sind(18*Theta);
Kentbb19 = Kentb19.*sind(19*Theta);
Kentbb20 = Kentb20.*sind(20*Theta);
Kentbb21 = Kentb21.*sind(21*Theta);
Kentbb22 = Kentb22.*sind(22*Theta);
Kentbb23 = Kentb23.*sind(23*Theta);
Kentbb24 = Kentb24.*sind(24*Theta);
Kentbb25 = Kentb25.*sind(25*Theta);
Kentbb26 = Kentb26.*sind(26*Theta);
Kentbb27 = Kentb27.*sind(27*Theta);
Kentbb28 = Kentb28.*sind(28*Theta);
Kentbb29 = Kentb29.*sind(29*Theta);
Kentbb30 = Kentb30.*sind(30*Theta);
Kentbb31 = Kentb31.*sind(31*Theta);
Kentbb32 = Kentb32.*sind(32*Theta);
Kentbb33 = Kentb33.*sind(33*Theta);
Kentbb34 = Kentb34.*sind(34*Theta);
Kentbb35 = Kentb35.*sind(35*Theta);
Kentbb36 = Kentb36.*sind(36*Theta);
```

```
%% RESULTANT AMPLITUDES

Kentamp1 = sqrt(Kenta1^2 + Kentb1^2);
Kentamp2 = sqrt(Kenta2^2 + Kentb2^2);
Kentamp3 = sqrt(Kenta3^2 + Kentb3^2);
Kentamp4 = sqrt(Kenta4^2 + Kentb4^2);
Kentamp5 = sqrt(Kenta5^2 + Kentb5^2);
Kentamp6 = sqrt(Kenta6^2 + Kentb6^2);
Kentamp7 = sqrt(Kenta7^2 + Kentb7^2);
Kentamp8 = sqrt(Kenta8^2 + Kentb8^2);
Kentamp9 = sqrt(Kenta9^2 + Kentb9^2);
Kentamp10 = sqrt(Kenta10^2 + Kentb10^2);
Kentamp11 = sqrt(Kenta11^2 + Kentb11^2);
Kentamp12 = sqrt(Kenta12^2 + Kentb12^2);
Kentamp13 = sqrt(Kenta13^2 + Kentb13^2);
Kentamp14 = sqrt(Kenta14^2 + Kentb14^2);
Kentamp15 = sqrt(Kenta15^2 + Kentb15^2);
Kentamp16 = sqrt(Kenta16^2 + Kentb16^2);
Kentamp17 = sqrt(Kenta17^2 + Kentb17^2);
Kentamp18 = sqrt(Kenta18^2 + Kentb18^2);
Kentamp19 = sqrt(Kenta19^2 + Kentb19^2);
Kentamp20 = sqrt(Kenta20^2 + Kentb20^2);
Kentamp21 = sqrt(Kenta21^2 + Kentb21^2);
Kentamp22 = sqrt(Kenta22^2 + Kentb22^2);
Kentamp23 = sqrt(Kenta23^2 + Kentb23^2);
Kentamp24 = sqrt(Kenta24^2 + Kentb24^2);
Kentamp25 = sqrt(Kenta25^2 + Kentb25^2);
Kentamp26 = sqrt(Kenta26^2 + Kentb26^2);
Kentamp27 = sqrt(Kenta27^2 + Kentb27^2);
Kentamp28 = sqrt(Kenta28^2 + Kentb28^2);
Kentamp29 = sqrt(Kenta29^2 + Kentb29^2);
Kentamp30 = sqrt(Kenta30^2 + Kentb30^2);
Kentamp31 = sqrt(Kenta31^2 + Kentb31^2);
Kentamp32 = sqrt(Kenta32^2 + Kentb32^2);
Kentamp33 = sqrt(Kenta33^2 + Kentb33^2);
Kentamp34 = sqrt(Kenta34^2 + Kentb34^2);
Kentamp35 = sqrt(Kenta35^2 + Kentb35^2);
Kentamp36 = sqrt(Kenta36^2 + Kentb36^2);

Kentamp = [Kentamp1 Kentamp2 Kentamp3 Kentamp4 Kentamp5
Kentamp6 Kentamp7 Kentamp8 Kentamp9 Kentamp10 Kentamp11
Kentamp12 Kentamp13 Kentamp14 Kentamp15 Kentamp16
Kentamp17 Kentamp18 Kentamp19 Kentamp20 Kentamp21
Kentamp22 Kentamp23 Kentamp24 Kentamp25 Kentamp26
Kentamp27 Kentamp28 Kentamp29 Kentamp30 Kentamp31
Kentamp32 Kentamp33 Kentamp34 Kentamp35 Kentamp36];
```

```
%% RESULTANT PHASES

Kentphase1 = atand(Kenta1 / Kentb1);
Kentphase2 = atand(Kenta2 / Kentb2);
Kentphase3 = atand(Kenta3 / Kentb3);
Kentphase4 = atand(Kenta4 / Kentb4);
Kentphase5 = atand(Kenta5 / Kentb5);
Kentphase6 = atand(Kenta6 / Kentb6);
Kentphase7 = atand(Kenta7 / Kentb7);
Kentphase8 = atand(Kenta8 / Kentb8);
Kentphase9 = atand(Kenta9 / Kentb9);
Kentphase10 = atand(Kenta10 / Kentb10);
Kentphase11 = atand(Kenta11 / Kentb11);
Kentphase12 = atand(Kenta12 / Kentb12);
Kentphase13 = atand(Kenta13 / Kentb13);
Kentphase14 = atand(Kenta14 / Kentb14);
Kentphase15 = atand(Kenta15 / Kentb15);
Kentphase16 = atand(Kenta16 / Kentb16);
Kentphase17 = atand(Kenta17 / Kentb17);
Kentphase18 = atand(Kenta18 / Kentb18);
Kentphase19 = atand(Kenta19 / Kentb19);
Kentphase20 = atand(Kenta20 / Kentb20);
Kentphase21 = atand(Kenta21 / Kentb21);
Kentphase22 = atand(Kenta22 / Kentb22);
Kentphase23 = atand(Kenta23 / Kentb23);
Kentphase24 = atand(Kenta24 / Kentb24);
Kentphase25 = atand(Kenta25 / Kentb25);
Kentphase26 = atand(Kenta26 / Kentb26);
Kentphase27 = atand(Kenta27 / Kentb27);
Kentphase28 = atand(Kenta28 / Kentb28);
Kentphase29 = atand(Kenta29 / Kentb29);
Kentphase30 = atand(Kenta30 / Kentb30);
Kentphase31 = atand(Kenta31 / Kentb31);
Kentphase32 = atand(Kenta32 / Kentb32);
Kentphase33 = atand(Kenta33 / Kentb33);
Kentphase34 = atand(Kenta34 / Kentb34);
Kentphase35 = atand(Kenta35 / Kentb35);
Kentphase36 = atand(Kenta36 + Kentb36);

Kentphase = [Kentphase1 Kentphase2 Kentphase3 Kentphase4
Kentphase5 Kentphase6 Kentphase7 Kentphase8 Kentphase9
Kentphase10 Kentphase11 Kentphase12 Kentphase13
Kentphase14 Kentphase15 Kentphase16 Kentphase17
Kentphase18 Kentphase19 Kentphase20 Kentphase21
Kentphase22 Kentphase23 Kentphase24 Kentphase25
Kentphase26 Kentphase27 Kentphase28 Kentphase29
```

```
Kentphase30 Kentphase31 Kentphase32 Kentphase33
Kentphase34 Kentphase35 Kentphase36];
```