2014

A Virtual Shaker Table for Predicting Loads in Automotive Powertrain Mounts

Xiaowu Yang
University of Windsor

A Virtual Shaker Table for Predicting Loads in Automotive Powertrain Mounts

by
Xiaowu Yang

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of
Mechanical, Automotive, & Materials Engineering
in Partial Fulfilment of the Requirements for
the Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada

2014

A Virtual Shaker Table for Predicting Loads in Automotive Powertrain Mounts

by

Xiaowu Yang

APPROVED BY

Dr. X. Chen
Department of Electrical & Computer Engineering

Dr. J. Johrendt
Department of Mechanical, Automotive, & Materials Engineering

Dr. B. Minaker, Advisor
Department of Mechanical, Automotive, & Materials Engineering

September 11, 2014

Declaration of Co-Authorship

I hereby declare that this thesis incorporates material that is result of joint research, as follows:

This thesis also incorporates the outcome of a joint research undertaken in collaboration with Sida Li under the supervision of Dr. Bruce Minaker. The collaboration is covered in Chapter 6 of the thesis. In all cases, the key ideas, primary contributions, experimental designs, data analysis and interpretation, were performed by the author, and the contribution of co-authors was primarily through the provision of modeling configuration and data exchange.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

Abstract

In the automotive industry, multi-axis shaker tables are often used to study the damage caused by motion-induced inertia loads to components such as engine mounts or fuel tank strips. To assess the component durability characteristics using this approach, prototype parts must be built and a test rig must be installed. This process is both time and budget consuming, so there is an incentive to reduce the number of physical shaking tests. To that end, this thesis introduces a set of software tools that are capable of conducting virtual shaking simulations with quality output results, i.e., a virtual multi-axial shaker table (VMAST). By refining and reproducing vehicle body acceleration signals collected from the proving grounds, the VMAST is able to replay the body motion of a vehicle. The reproduced motion (drive file) can then be used to drive the virtual dynamic shaking. With the additional consideration of vehicle body local flexibility, the flexible motion can be added to the rigid body motion to improve the simulation accuracy. The dynamic shaking simulation can be done natively in MATLAB®, or the drive files derived from MATLAB® can be used by other commercial software, such as Altair® MotionView®. The virtual load data acquisition of the engine bushing mount is implemented during the simulation to predict the fatigue index, which can be referenced in the design procedure. This VMAST provides the automotive engineer with a cost effective tool for analysis, and optimizes the testing process, allowing rapid design iteration.

To my parents, grandparents and Yi
for their love & support.

Acknowledgements

First, I would like to thank my supervisor Dr. Bruce Minaker for his guidance and patience over the last two years. His encouragement and extensive knowledge when the project proved to be the most difficult allowed me to overcome the difficulties and accomplish my goals.

Furthermore, I would like to thank the Chrysler Group LLC, Connect Canada, the Ontario Centres of Excellence, and the University of Windsor–Chrysler Canada Automotive Research and Development Centre for the funding and research facility that allowed me to pursue my research goals.

I would also like to express my thanks to my colleague Sida Li, who I have shared many theoretical thoughts and laughs with over the years.

Last, I need to thank my family and Yi, for their love and support in all aspects of my life.

Contents

†Chapter 6.3.1 is the outcome of joint research.

List of Tables

List of Figures

List of Abbreviations

All abbreviations used in this work are described in this section.

Abbreviation	**Meaning**
ABM	Advanced Bushing Model
ARDC	Automotive Research and Development Center
BDF	Backward Differentiation Formula
CM	Center of Mass
DAE	Differential Algebraic Equation
DOE	Design of Experiments
DOF	A Degree of Freedom
EOM	Equation of Motion
FE	Finite Element
FEA	Finite Element Analysis
FWD	Front Wheel Drive
GA	Genetic Algorithm
GUI	Graphical User Interface
LHS	Left Hand Side
MAST	Multi-Axis Shaker Table
MBD	Multi-Body Dynamics
MBS	Multi-Body System
NVH	Noise Vibration Harshness
ODE	Ordinary Differential Equation
RHS	Right Hand Side
RMS	Root Mean Square
VMAST	Virtual Multi-Axis Shaker Table
VPD	Virtual Prototyping Development

Nomenclature

Mathematical nomenclature throughout this work is listed below, separated by the chapter in which it first appears.

Literature Review

Label	Description
f	Vector of applied forces in global reference frame
h	step size
K	Kinetic energy
m	Running mean removal window size
n	number of points defined for running mean removal
$\vec{n}$	Applied torque vector in global frame
p_0	First data point in running mean removal
q	Generalized coordinates
Q	Generalized forces
t	time
u	Translational velocity
v^p	Velocity of the point of application of the external force F
x	x coordinate
y	y coordinate
Δ	Correction factor
Φ	Constraint equations
λ	Lagrange multiplier
$\vec{\omega}$	Angular velocity vector
ζ	Euler angle rates

Theory

Label	Description
$\vec{a}_A$	Acceleration vector at location A
$\vec{a}_B$	Acceleration vector at location B
$\vec{a}_{B/A}$	Relative acceleration vector from A to B
$\vec{a}_G$	Translational acceleration vector at the center of mass
a_x^A, a_y^A, a_z^A	Translational x, y, z acceleration at location A
$\vec{F}$	Total force vector acting on the center of mass
$\vec{H}_G$	Angular momentum vector
I_{xx}	Moment of inertia around x
I_{yy}	Moment of inertia around y
I_{zz}	Moment of inertia around z
I_{xy}	Product of inertia around x and y
I_{xz}	Product of inertia around x and z
I_{yz}	Product of inertia around y and z
I_G	Inertia tensor
m	Mass of the rigid body, a three by three diagonal matrix
$\vec{M}_G$	Total torque vector acting about the center of mass
$\vec{r}$	Coordinate vector of the global frame
$\vec{r}_{B/A}$	Distance vector from A to B
$\vec{r}_3$	Coordinate vector of the rotating frame around x, y and z
r_x	Distance along x
r_y	Distance along y
r_z	Distance along z
$\boldsymbol{R}$	Transformation matrix from global to rotating frame
$\boldsymbol{R_1}$	Transformation matrix from global to rotating frame (x only)
$\boldsymbol{R_2}$	Transformation matrix from global to rotating frame (x and y)
$\boldsymbol{R_3}$	Transformation matrix from global to rotating frame (x, y and z)
$\vec{v}_A$	Velocity vector at location A
$\vec{v}_B$	Velocity vector at location B
$\vec{v}_{B/A}$	Relative velocity vector from A to B
$\vec{v}_G$	Translational velocity vector at the center of mass
v_x	Translational x velocity in the rotating frame
v_y	Translational y velocity in the rotating frame
v_z	Translational z velocity in the rotating frame

x, y, z	Coordinates of the global frame
x_1, y_1, z_1	Coordinates of the rotating frame around x only
x_2, y_2, z_2	Coordinates of the rotating frame around x and y
x_3, y_3, z_3	Coordinates of the rotating frame around x, y and z
$\vec{\alpha}$	Angular acceleration vector in the body reference frame
$\alpha_x^A, \alpha_y^A, \alpha_z^A$	Rotational x, y, z acceleration at location A
ϕ	Euler angle around x axis
θ	Euler angle around y axis
$\vec{\omega}$	Angular velocity vector of the body reference frame
ω_x	Euler angle rates around x axis
ω_y	Euler angle rates around y axis
ω_z	Euler angle rates around z axis
ψ	Euler angle around z axis

Drive File Development

Label	Description
$\vec{a}_A$	Acceleration vector at accelerometer A
$\vec{a}_B$	Acceleration vector at accelerometer B
$\vec{a}_C$	Acceleration vector at accelerometer C
$\boldsymbol{I}$	3×3 identity matrix
$\tilde{\boldsymbol{r}}_{\boldsymbol{A/G}}$	Skew-symmetric matrix representing distance from CM to A
$\tilde{\boldsymbol{r}}_{\boldsymbol{B/G}}$	Skew-symmetric matrix representing distance from CM to B
$\tilde{\boldsymbol{r}}_{\boldsymbol{C/G}}$	Skew-symmetric matrix representing distance from CM to C
v_{CM}^s	Filtered and scaled velocities at center of mass
v_{CM}	Filtered velocities at center of mass
a_{CM}	Accelerations at center of mass
v_{rx}	Rotational velocity around x
v_{ry}	Rotational velocity around y
v_{rz}	Rotational velocity around z
a_m	Measured accelerations
a_r	Reproduced accelerations
i	Counter

Dynamic Shaking Model

Label	Description
$\vec{a}$	Powertrain translational acceleration vector at CM
C	Bushing damping coefficient
E	Euler angles
$\vec{F}$	Total force vector at powertrain CM
$\vec{F}_B$	Bushing force vector
g	Gravitational constant
K	Bushing stiffness
$\vec{M}_G$	Total moment vector at powertrain CM
$\vec{M}_H$	Half-shaft torque vector
$\vec{r}$	Distance vector from powertrain CM to powertrain-side mount locations
$\vec{v}$	Powertrain translational velocity vector at CM
$\vec{v}_e$	Powertrain-side mount location velocity vector in the local frame
$\vec{v}_f$	Chassis-side mount location velocity vector in the global frame
$\vec{x}$	Powertrain CM location vector in the global frame
$\vec{x}_e$	Powertrain-side mount location vector in the global frame
$\vec{x}_f$	Chassis-side mount location vector in the global frame
$\vec{\alpha}$	Powertrain angular acceleration vector at CM
$\vec{\omega}$	Powertrain angular velocity vector at CM

Chapter 1

Introduction

1.1 Motivation

The multi-axis shaker table (MAST) (Figure 1.1), has been widely used in the automotive industry in order to test vehicles or vehicle components specifically under high frequency excitation for various purposes, such as fatigue analysis, ride quality and noise vibration harshness (NVH). As one of the major tasks of the MAST, the fatigue test is vital for vehicle design engineers in carrying out quality products while maintaining minimum cost. One common example is the powertrain (engine and transmission assembly) mount durability test, in which the powertrain is mounted on the vehicle frame through bushings; the whole assembly is then fastened to the shaker table platform, which is driven by hydraulic cylinders. Acceleration data collected on the vehicle frame during the road test is used as the target output for the shaker to match, and the feedback loop control algorithm is implemented to automatically adjust the actuation force. However, conducting such a test requires both hardware and time: prototype vehicles have to be built, the test rig must be set up, and the test cycle lasts for hours and hours.

To this end, the Chrysler Automotive Research and Development Center (ARDC) and University of Windsor have worked together in a research project to develop a virtual simulation tool that could numerically simulate the shaking process of a vehicle powertrain and predict the powertrain mount loads. The ultimate goal of this project is to replace the physical shaking test with virtual shaking simulation that could precisely predict the vehicle chassis and powertrain motion, as well as mount loads. By virtually changing the powertrain and mount parameters, new simulation results would be derived quickly, and this would largely accelerate the design and test process.

Figure 1.1: Multi-Axis Shaker Table.

1.2 Virtual Multi-Axis Shaker Table

The virtual multi-axis shaker table (VMAST) is a simulation tool developed to numerically simulate the vehicle powertrain behavior under real road input profiles in order to acquire powertrain mount load data. The simulation result can be used to predict the fatigue of the bushing mount. It is a user friendly software package, and it has the advantage of changing the physical properties with ease, which allows rapid design iterations.

Designed to function the same as the physical shaker table, the VMAST is able to cover full six degrees of freedom (DOF) rigid motions of the shaker platform (three translational and three rotational). In addition, it has the capability to recover local deformation on the vehicle frame by introducing extra translational degrees of freedom. The VMAST reads in acceleration data of the vehicle frame collected from the proving ground, and transfers it to displacement for the purpose of driving the dynamic shaking. Mathematically, the total number of degrees of freedom that the virtual shaker can produce is identical to the number of input accelerations. For example, a vehicle frame that has three accelerometers on distinct locations will give nine translational accelerations. The shaker can then solve for nine displacements. The combined motion of the nine displacements contains the chassis rigid motion and the local deformation. A schematic of the VMAST is shown in Figure 1.2; the red arrows represent the rigid motion that the shaker could produce, and the yellow arrows demonstrate the overall degrees of freedom the shaker generates, which consist of rigid and flexible.

The VMAST reproduced time history coordinate data is called the ‘drive file’, as it is used to drive the dynamic shaking. The generation of the drive file will be explained in detail later in the thesis, and is done using MATLAB®.

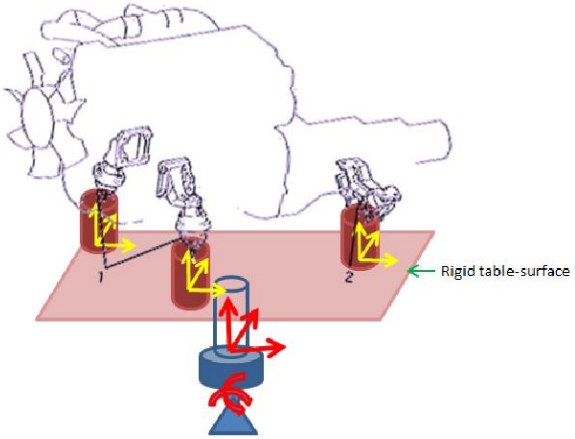


Figure 1.2: Virtual Multi-Axis Shaker Table.

1.3 Virtual Dynamic Shaking

After the drive files are developed, the next step is to conduct the dynamic shaking and collect the result. There are two components to make up the dynamic shaking model: the powertrain and bushing mounts. This model treats the powertrain as a single rigid body with moments and products of inertia, that is connected to the bushing mounts. The model can be built and simulated using MATLAB®. In case a more complex powertrain model is needed, commercial software such as MSC® Adams® or Altair® MotionView® can be used to construct the advanced model where, for example, the powertrain model may be equipped with additional linkages. Both simple and complex models are able to generate simulation results such as the powertrain local acceleration, velocity and displacement, as well as the bushing travel for powertrain motion study, and the bushing force for fatigue analysis.

Depending on the requirements of the simulation, either a simple or sophisticated bushing model can be used. The simplest bushing model uses linear stiffness and damping based on a reasonable estimation, which is computationally fast. The advanced bushing model (ABM) can be applied in the simulation where the most accuracy is needed. It is a virtual bushing model with a parameter identification tool, which identifies the virtual bushing parameters based on real bushing test data. However, it is relatively time consuming.

1.4 Research Objectives

The first objective is to develop a set of MATLAB® codes that generate the desired motion to drive the powertrain. The codes must be able to read the acceleration data collected on the vehicle frame during the road test and reproduce the motion of the vehicle frame. The process should virtually replay the vehicle frame motion due to excitation from the road in the real case.

The second objective is to develop a numerical model in MATLAB® to simulate the shaking procedure

of the powertrain. The model reads in the motion produced in the first step, simulates and generates the time history output. This numerical model is compiled to be a user friendly design tool in cases where the powertrain mechanism is simple. It is easy to implement and is capable of running in batch mode.

The last objective is to use MBD software for complex powertrain mechanism modeling and simulation. In this case, the shaker and powertrain models need to be built with commercial software.

1.5 Structure of the Thesis

The thesis consists of seven chapters. The first chapter introduces and describes the motivations and goals of the research, as well as the research product and its function. The second chapter is a review of the literature, which includes a review of the virtual simulation tools and their corresponding solving algorithms, as well as the previous research in the relevant area. In Chapter 3, the theory used to develop the virtual shaker will be addressed.

In Chapter 4, the drive file development approach is introduced, which includes five detailed steps of the numerical approach to reproduce the vehicle frame motion from accelerations in order to drive the dynamic shaking. The chapter will focus on the application of relevant mathematical algorithms, as well as the data processing method, e.g., the data filtering. In addition, a validation is included, and the assumptions and limitations are also discussed. Lastly, the software application of the drive file development process is described.

In the first half of Chapter 5, the MATLAB® dynamic shaking model is presented. The kinematic and dynamic equations used to solve the model are introduced. The limitation of the MATLAB® model is also discussed. In addition, the software application of the MATLAB® model is described. The second half of this chapter is focused on the model built using MBD commercial software. An Altair® MotionView® shaker and powertrain model is presented. The physical parameters of the vehicle and powertrain model are from the Chrysler PF platform with a 2.4L engine.

In Chapter 6, the simulation and results of both MATLAB® and MotionView® models are discussed. The validation of the simulation results is done through the following: a check on the plausibility of the simulation results in the real world; comparison of the simulation results between the virtual models; comparison of the simulation results against test data.

In Chapter 7, conclusions on the development and application of the VMAST are presented. Potential future work is outlined.

Chapter 2

Literature Review

2.1 Multi-Body Dynamics

The use of multi-body dynamics (MBD) is essential in vehicle design and testing. According to Schwerin[1], multi-body systems (MBS) play an important role in computer aided technical mechanics, and vehicle dynamics is the major area of application. The ability to design and optimize such technical multi-body systems gives a company working in this area the competitive edge on the market. The equations of motion (EOM) are well known to describe the behavior of a multi-body system as a set of mathematical functions in terms of dynamic variables. Much work has been done on studying multi-body systems for automatic generation of the equations of motion, for example, see Minaker and Rieveley[2]. A method was developed for automatic generation of linearised equations of motion that allows for easy inclusion of nonholonomic constraints, and the method presents equations of motion well suited for vehicle stability analysis. They stated that the use of mathematical models in vehicle system design has grown immensely in recent history, leading to the development of models with increasing size and complexity.

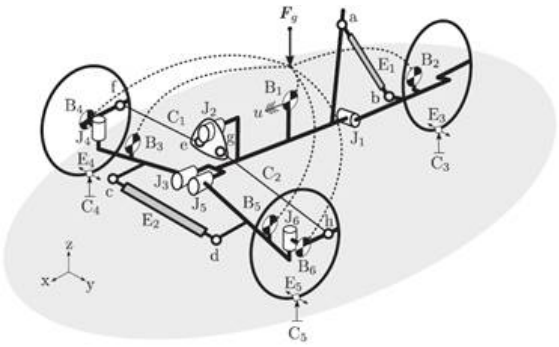


Figure 2.1: Vehicle simulated by Minaker and Rieveley[2] using their proposed method.

2.2 Virtual Simulation

As Grote and Sharp[3] stated, the past decades have seen a rapid advance in computer based prediction and analysis tools. These tools allow for parts and assemblies to be constructed and their behavior predicted in the virtual world of computer simulation. Becker, Salvatore and Zirpoli[4] pointed out that virtual simulation tools now play a very important role in new product development. They have been widely hailed to significantly cut development time and costs. They argued that the contribution of virtual tools to experimentation goes well beyond the incremental improvement of the results obtained with physical experimentation. "Virtual simulation techniques can do much more than just reduce the cost and increase the speed of problem solving," they mentioned.

2.2.1 Modeling Techniques

ADAMS

The Automated Dynamic Analysis of Mechanical Systems (ADAMS) software marketed by MSC® Software has been proven as very beneficial to virtual prototype development (VPD) by reducing product time to market and product development costs.

As proposed by the Simulation Based Engineering Lab (SBEL)[5], the virtual prototyping process of ADAMS® includes four steps. The first step is building a model using bodies, joints, contacts and forces. The next step is testing the design using measures, simulations, animations and plots; the simulation result can be validated by importing test data and superimposing it. Next, one should review the design by adding friction, forcing functions, flexible parts and control systems, and iterate it through parameters and design variables. Lastly, the design is improved using DOEs (Design of Experiments) or optimization. The design process can be automated by using custom menus, macros or custom dialog boxes. This modeling strategy is not only valid in ADAMS®, but also in other multi-body codes as well.

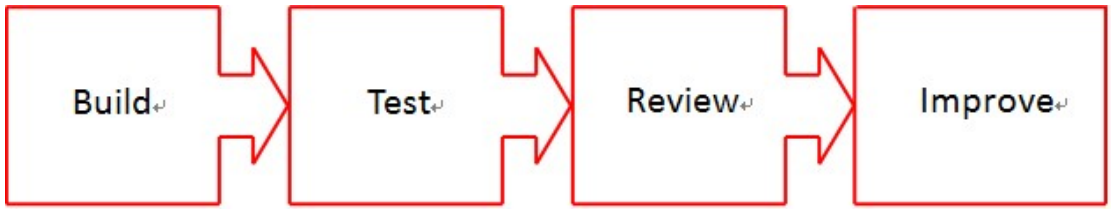


Figure 2.2: Virtual prototyping process by SBEL[5].

MotionView

MotionView® is a solver-neutral multi-body pre-processor developed by Altair® Engineering. It has become a useful and comprehensive multi-body modeling tool (Fothergill[6]). In particular, MotionView®

has the advantage of a particularly user friendly GUI (Graphical User Interface), and better graphical representation when comparing with ADAMS®. Its capability in handling nonlinear elements, such as flexible beams or bushings, as well as modeling complex multi-body system, like a vehicle chassis assembly is competitive with ADAMS®. It has been gaining popularity among engineering companies and academic institutions, especially for those who want to model and simulate multi-body systems without much proficiency or experience. The modeling technique described above is well suited for the MotionView® application.

The MotionView® default solver—MotionSolve®, is a system level, multi-body solver that is based on the principles of mechanics[7]. Using a multi-body system built in MotionView®, MotionSolve® automatically formulates the equations of motion and numerically solves them. MotionSolve® provides four types of simulation: transient, static, quasi-static and linear, among which transient is used to dynamically simulate the system with more than zero degrees of freedom. The names of the solvers used for solving differential algebraic equations (DAEs) in transient simulation are: ABAM, VSTIFF, MSTIFF and DSTIFF. The default solver is DSTIFF. It is claimed in the research that the DSTIFF solver is more robust than the solvers in ADAMS® in dealing with high frequency excitation as an input to multi-body systems.

MATLAB

According to Hanselman and Littlefield[8], MATLAB® was initially developed by a lecturer in the 1970's to help students learn linear algebra. Later, it was marketed and further developed under MathWorks® Inc. MATLAB® is a software package that can be used to perform analysis and solve mathematical and engineering problems. It has excellent programming features and graphics capability that are easy to learn and provide the user with great flexibility. SIMULINK®—a data flow graphical programming tool, is used for modeling, simulating and analyzing dynamic systems. It has a comprehensive library that can be used to simulate linear, non-linear or discrete systems. The programming features and comprehensive mathematical functions enable the modeling of complex MBS in the MATLAB® environment.

Other commercial software such as MapleSim® and CarSim® might be applicable for the research presented in this thesis. However, due to the adequate number of approaches already available, the software is not further studied.

2.2.2 Solving Algorithm

After the MBS is built in the pre-processor, the next step is to simulate the system. Once the simulation is initiated, the MBD software will analyze the information that the user imports, such as the generalized coordinates, constraints, motions and initial conditions. Depending on the simulation type chosen, the next step is to formulate the corresponding equations of motion of the system. In terms of dynamic analysis, the common method used in software such as ADAMS® and MotionView® is presented by

Negrut and Dyer[9]. According to them, the Lagrange formulation of the equations of motion is shown as the following second order differential equations

$$\frac{d}{dt}\left[\left(\frac{\partial K}{\partial \dot{q}}\right)^T\right] - \left(\frac{\partial K}{\partial q}\right)^T + \Phi_q^T \lambda = Q \tag{2.1}$$

where:

$K =$ kinetic energy
$q =$ generalized coordinates
$\Phi =$ constraint equations
$\lambda =$ Lagrange multiplier
$Q =$ generalized forces

The second order equations can be further reduced to first order considering the choice of the generalized coordinates

$$q = \left\{ \begin{array}{c} p \\ \varepsilon \end{array} \right\} \tag{2.2}$$

where p is the positon and ε is the orientation (Euler angles). The first order equations are shown as:

$$\frac{d}{dt}\left[\begin{array}{c} \left(\frac{\partial K}{\partial u}\right)^T \\ \left(\frac{\partial K}{\partial \zeta}\right)^T \end{array} \right] - \left[\begin{array}{c} \left(\frac{\partial K}{\partial p}\right)^T \\ \left(\frac{\partial K}{\partial \varepsilon}\right)^T \end{array} \right] + \left[\begin{array}{c} \Phi_p^T \lambda \\ \Phi_\varepsilon^T \lambda \end{array} \right] = \left[\begin{array}{c} (\prod p)^T f \\ (\prod R)^T \bar{n} \end{array} \right] \tag{2.3}$$

The projection operators in the generalized forces term are computed as:

$$\prod p = \frac{\partial v^p}{\partial u} \tag{2.4}$$

$$\prod R = \frac{\partial \vec{\omega}}{\partial \zeta} \tag{2.5}$$

where:

$u =$ translational velocity
$\zeta =$ Euler angle rates
$f =$ vector of applied forces in global reference frame
$n =$ applied torque in global frame
$v^{\mathrm{p}} =$ velocity of the point of application P of the external force F
$\vec{\omega} =$ angular velocity

Since

$$\frac{d}{dt}\left(\frac{\partial K}{\partial u}\right)^T = M\dot{u} \tag{2.6}$$

$$\left(\frac{\partial K}{\partial p}\right)^T = 0 \tag{2.7}$$

with the angular momentum defined as:

$$\Gamma = \frac{\partial K}{\partial \zeta} = B^T \vec{J} B\dot{\zeta} \tag{2.8}$$

the EOM of Eq. (2.3) are reformulated as:

$$M\dot{u} + \Phi_p^T \lambda = \left(\prod p\right)^T f \tag{2.9}$$

$$\Gamma - \frac{\partial K}{\partial \varepsilon} + \Phi_\varepsilon^T \lambda = \left(\prod R\right)^T \bar{n} \tag{2.10}$$

The first order differential equations above are called in what follows *kinetic differential equations*, and they indicate how external forces determine the time variation of the translational and angular momentum.

At last, the time variation of the generalized coordinates is related to the translational and angular momentum by means of the *kinematic differential equations*. By assembling the kinetic and kinematic differential equations, a set of equations for each rigid body in the MBS are derived as follows:

$$M\dot{u} + \Phi_p^T \lambda - \left(\prod p\right)^T f = 0 \tag{2.11}$$

$$\Gamma - B^T \vec{J} B\zeta = 0 \tag{2.12}$$

$$\Gamma - \frac{\partial K}{\partial \varepsilon} + \Phi_\varepsilon^T \lambda - \left(\prod R\right)^T \bar{n} = 0 \tag{2.13}$$

$$\dot{p} - u = 0 \tag{2.14}$$

$$\dot{\varepsilon} - \zeta = 0 \tag{2.15}$$

It is notable that the solution to the above differential equations must also satisfy the kinematic constraint equations of the system. The equations are:

$$\Phi\left(q,t\right) = 0 \tag{2.16}$$

$$\Phi_q\left(q,t\right)\dot{q} = -\Phi_t\left(q,t\right) \tag{2.17}$$

$$\Phi_q\left(q,t\right)\ddot{q} = -\left(\Phi_q\dot{q}\right)_q\dot{q} - 2\Phi_{qt}\dot{q} - \Phi_{tt}\left(q,t\right) \tag{2.18}$$

This set of differential and constraint equations from Eq. (2.11) to Eq. (2.18) are called the Differential Algebraic Equations (DAEs). In many cases, the DAEs of the MBS cannot be solved symbolically. Therefore, a numerical solution is used find a close approximation. As Negrut and Dyer stated[9], the most common DAE solver that ADAMS® uses is the GSTIFF-I3. Here the GSTIFF stands for the solution of stiff differential equations, and I3 means the index of the DAEs is three, which means the number of times the constraints in the system must be differentiated to get the system into ODEs is three. It is always index three for a mechanical system. In mathematics, a stiff equation is a differential equation for which certain numerical methods for solving the equation are numerically unstable, unless the step size is taken to be extremely small. Lambert[10] described stiffness as: "If a numerical method with a finite region of absolute stability, applied to a system with any initial conditions, is forced to use in a certain interval of integration a step length which is excessively small in relation to the smoothness of the exact solution in that interval, then the system is said to be stiff in that interval." According to Gear and Skeel[11], the earliest paper on stiff differential equations was given by Curtiss and Hirschfelder[12], in which they developed the method using backward differentiation formula (BDF) in solving differential equations that failed to come up with stable numerical solutions with other methods. The next significant development was the definition of A-stability by Dahlquist[13]. The theory was extended by Daniel and Moore[14], mentioning that the order of an A-stable method cannot exceed twice its number of derivatives involved implicitly in each step. Later approaches including Widlund[15] and Gear[16][17] focused on breaking through the order limitation implied by A-stability. Both of them explored non-A-stable methods, which were realized to be most effective for general stiff problems.

When solving the DAEs, GSTIFF-I3 will first apply an order one implicit integration formula, in order to convert the DAEs into a set of algebraic equations. The one step, A-stable BDF replaces the derivative $\dot{y}_1$ at time t_1 with

$$\dot{y}_1 = \frac{1}{h} y_1 - \frac{1}{h} y_0 \tag{2.19}$$

Then the derivative in the above equation can be 'discretized' by replacing $\dot{y}_1$ with a nonlinear function $g(y,t)$. The equation becomes:

$$\frac{1}{h} y_1 - \frac{1}{h} y_0 - g(t_1, y_1) = 0 \tag{2.20}$$

To retrieve y_1, this 'discretization equation' is solved using a Newton-Raphson type iterative algorithm as follows:

$$x^{(1)} = x^{(0)} - \frac{f(x^{(0)})}{f'(x^{(0)})} \tag{2.21}$$

The algorithm continues by setting $x^{(0)}$ with $x^{(1)}$.

By applying the discretization approach to Eq. 2.11 to 2.15 and the position constraint equation form Eq. (2.16) to (2.18) (for index 3 scenario), all the first order time derivatives in the DAEs are discretized.

At this stage, the unknowns in the nonlinear equations are defined as:

$$y^T = \begin{bmatrix} u & \Gamma & \zeta & p & \varepsilon & \lambda & f & \bar{n} \end{bmatrix}^T \tag{2.22}$$

the nonlinear system is rewritten as:

$$\Psi(y) = 0 \tag{2.23}$$

The Newton-Raphson method is then used to solve the system. Once the initial value of $y^{(0)}$ is predicted, the iteration runs as:

$$\Psi_y(y_0)\Delta^{(j)} = -\Psi(y^{(j)}) \tag{2.24}$$

$$y^{(j+1)} = y^{(j)} + \Delta^{(j)} \tag{2.25}$$

until the correction $\Delta^{(j)}$ are small enough (the value of Δ is set by the user).

Competitive with the ADAMS® solver, MotionSolve® dynamic simulation accounts for all inertia effects, all applied forces and internal constraints. In dynamic simulations, the MotionSolve® solver analyzes the MBS and generates a set of DAEs. It solves the equations of motion in their most general form, including nonlinear effects[7]. The default DAE solver that MotionSolve® uses is DSTIFF, and the default setting for the DAE index is 3, which means the position constraint is included. According to the MotionSolve® reference guide[18], it solves a system of differential/algebraic equations of the form $G(t, y, y') = 0$, using a combination of BDF methods. This method is based on the available integrator DASPK, which is developed by Brown, Hindmarsh and Petzold[19].

2.2.3 Data Processing

Virtual simulation uses the data collected externally to reproduce the real case. However, as mentioned by Pyle[20], the data-gathering methods are often loosely controlled, resulting in out-of-range values, impossible data combinations, missing values, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is important.

The acceleration data that VMAST reads in is a high frequency, time-history signal collected from tests in the proving grounds. The data sometimes has undesired high frequency content that could lead to a nonsensical outcome if further processed. Another source of data error faced in the shaker developing process is the integration constant, which is an ambiguity inherent in the construction of antiderivatives. It is an arbitrary constant that affects the accuracy of the integration result.

In order to obtain accurate and meaningful simulation results, the input data and intermediate results must be properly processed. Two methods are studied and found effective in eliminating the numerical error buried in the data.

Butterworth Filter

The Butterworth filter is a type of signal processing filter designed to have as flat a response as possible in the passband. It is also referred to as a maximum flat magnitude filter. The filter was first described by Butterworth[21] stating: "An ideal electrical filter should not only completely reject the unwanted frequencies but should also have uniform sensitivity for the wanted frequencies."

The frequency response of the Butterworth filter is maximally flat in the passband and rolls off towards zero in the stopband. The advantage of the Butterworth filter is that it is very good at simulating the passband of an ideal filter. In other words, it gives the minimum distortion in the passband. However, as it goes to zero gradually, some parts of the stopband are still kept. As the design characteristic of the Butterworth filter, the filter order, which is defined as the maximum delay (in samples) used in creating each output sample, can be increased to enable a better filtering performance.

As a result, the low pass Butterworth filter is a good choice for eliminating the high frequency content buried in the acceleration data while preserving the desired frequencies in order to gain maximum accuracy in the simulation.

Running Mean Removal

The running mean is a calculation to analyze data points by creating a series of averages of different subsets of the full data set. Commercial software, such as nCode GlyphWorks®[22] and MATLAB® provide the function of auto generation of the running mean on each data point. It works based on the user selected number of points n. Starting from the first data point p_0, this function calculates the average of a 'window size' m, where $m = 2n + 1$, and the length of the window being calculated ranges from $p_0 - n$ to $p_0 + n$. Then the calculation proceeds to the next data point until it reaches the last one. When calculating the first or the last group of n data points, the size of the window exceeds the size of the data points available. Thus, there are several options of padding the edge of the data. They are: 'edge', which pads the data with the first and last values; 'zero', which pads the data with zeros; 'mean', which pads the data with the mean of the subset; 'window', which pads the data with the first half and last half window.

Once the mean is obtained, the next step is to subtract it from each data point. This function is to remove the trend from the signal, which appears as the integration constant on the intermediate result during the 'drive file' generation procedure.

2.3 Previous Work

As Dressler, Speckert and Bitsch mentioned[23], in recent years so-called 'virtual test rigs' have become more and more important in the development of cars and trucks. Originally, the idea was to substitute expensive durability tests with computer simulation. Meanwhile, the focus has changed towards a more

cooperative usage of numerical and laboratory rig simulation. In the early development stages, when no physical prototypes are available yet, numerical simulation is used to analyze and optimize the design.

2.3.1 Quarter-Car Test Rig

Many papers have been published on the development of virtual test rigs for vehicle component tests. One of them was by Sandu, Andersen and Southward[24], in which they developed a multi-body dynamics model of a quarter-car test rig equipped with a McPherson strut suspension. Both linear and nonlinear models were developed in order to predict the dynamic response of a quarter car suspension. The linear model consists of a sprung mass, an unsprung mass, and wheelpan actuator plate bodies. The sprung mass is supported by the primary-ride spring and included mass from the sprung mass plate and strut tower. The assumptions made for the linear model is that the motion of the masses could be approximated as linear translational. Also, the springs used in the system are assumed to have linear behavior. The equations of motion of the model are formulated by using the Lagrange multiplier to assemble the constraint forces with the inertia forces.

The major difference between the linear and nonlinear models is that the nonlinear model accounts for the kinematic joint constraints. Instead of positioning the sprung and unsprung mass in the uniform vertical direction, the unsprung mass is connected to the sprung mass through control arms, which breaks the geometrical linearity. The structure is assumed to be rigid. The same algorithm was used to formulate the nonlinear system equations.

The HHT method integrator was used to solve the DAEs of both the linear and nonlinear models, and the simulation results were compared against experimental data. The performance metric chosen to judge how well the mathematical model predicts reality is the ratio of the root-mean-square (RMS) of the error between simulation and experimental results to the experimental RMS. It is shown in the paper that both linear and nonlinear models predict the dynamic response of the quarter car model, such as the accelerations of both the sprung and unsprung masses, and the nonlinear model has better performance. The simulation result can be further implemented in studying the dynamic behavior of other nonlinear components, like dampers and bushings.

2.3.2 Full Vehicle Test Rig

The virtual test rig brought by Mántaras and Luque[25] focused on investigating the three-dimensional position of the vehicle body (i.e., roll, pitch and height of the center of gravity) and the main parameters (e.g., caster, camber, steer angle, kingpin inclination, toe in/out, roll axis, bump steer, Ackerman angle), that influence the handling of the vehicle. They proposed a three-dimensional kinematic model of the front and rear suspension and the steering system, in order to predict the vehicle positions as well as the parameters.

First of all, the coordinate frames are determined. One reference frame is attached to each wheel, plus a reference frame for the vehicle and a global reference frame (non-moving or inertial). The orientation of the vehicle is defined using Cardan angles. The next step is to determine the kinematic equations of each component. The kinematic equations of each wheel are determined using the three-dimensional constraint equations for the point of origin of the reference frame. The movement of the rear axle, in spite of being a three-dimensional movement, is considered as a combination of two plain movements. The vehicle kinematic model is built based on the assumption that the road is a plane, so the calculation of a plane tangent to the wheels can be used. At this stage, the full vehicle model is developed and ready for a virtual test.

The virtual test rig was set up using MATLAB® and Simulink®. The geometric parameters of the vehicle components are pre-defined by the user. The inputs to the test rig are, as a function of time, the roll angle and pitch angle of the vehicle body, the vehicle center of gravity height and the steering angle. The outputs of the simulation are the vehicle camber, caster, etc., which can be fed back to the system to optimize the vehicle geometry.

The author validated the virtual test rig simulation result against a measured result, where a small difference can be anticipated in the comparison, this being essentially a function of errors of measurement associated with the instruments used. Also, a MBS vehicle model was built in ADAMS® to validate the virtual test rig.

Another full vehicle virtual test rig was introduced by Dressler, Speckert and Bitsch[23]. In their paper, the numerical simulation models of complex servo-hydraulic test systems and their test specimen were demonstrated. The full vehicle test rig is composed of four hexapod-based suspension test rigs. The hexapod technology was illustrated by driving a platform that is connected to the wheel hub using six actuators; thus, the platform supports all six DOF of the hub. At a closer look, the hexapod consists of a base and a top platform, which are connected via six identical actuators. The joints between the actuators and the platforms have two rotational DOF. One actuator contains the piston and the cylinder, which in turn are connected using a cylindrical joint. This construction contains six DOFs. The force and torque balance equations of the construction are listed and imported into MATLAB® to calculate the actuator forces for certain suspension tests. The full vehicle model has one hexapod test rig connected to each wheel hub. Besides, each individual test rig is upgraded to have an additional six DOF, representing the tire model.

The research team also carried out physical test track simulations to compare with virtual simulation in results on forces, accelerations and displacements, as well as damage-related histograms, e.g., rainflow counting, range pair counting and damage values.

As a conclusion, current virtual test rigs are capable of producing quality simulation results that can be used in accelerating the design process and to contribute to a more cost and time efficient test procedure. However, the existing literature is mainly focused on the testing of vehicle handling and ride

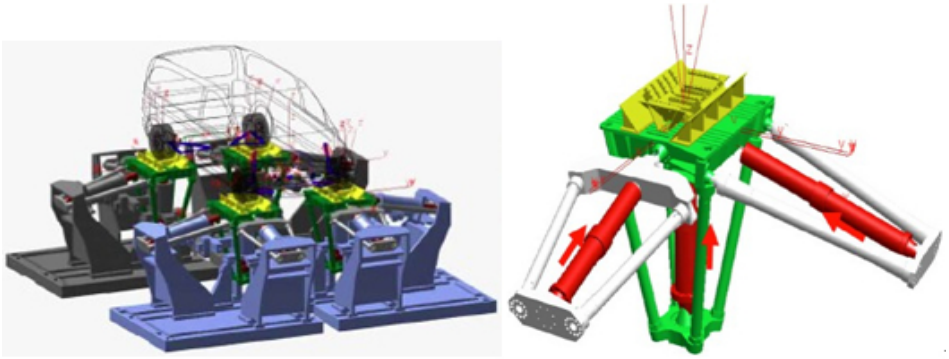


Figure 2.3: Full vehicle test rig and basic test rig mechanics[23].

behavior, and virtual simulation studies of the vehicle powertrain's (engine and transmission assembly) dynamic behavior under external excitation have been sparse. The research presented in this paper aimed to build a virtual test rig to simulate the vehicle powertain dynamic response under road input and collect corresponding inertia load data for the purpose of fatigue analysis.

Chapter 3

Theory

A series of kinematic and dynamic theories used in the drive file generation procedures and virtual shaker model development are introduced in this chapter, including the rigid body kinematics, coordinate transformation and equations of motion.

3.1 Rigid Body Kinematics

In physics, a rigid body is an idealization of a solid body in which deformation is neglected. In other words, the distance between any two given points of a rigid body remains constant in time regardless of external forces exerted on it. In this way, the kinematic relationships between any two points are studied.

There are two important relationships. First, the relationship of translational velocity between two points is formulated as:

$$\vec{v}_B = \vec{v}_A + \vec{v}_{B/A} \tag{3.1}$$

where $\vec{v}_{B/A}$ is the is the relative velocity, and it can be represented as:

$$\vec{v}_{B/A} = \vec{\omega} \times \vec{r}_{B/A} + \vec{\dot{r}}_{B/A} \tag{3.2}$$

where:

$\vec{\omega}$ = angular velocity of the body reference frame
$\vec{r}_{B/A}$ = distance vector from A to B
$\vec{\dot{r}}_{B/A}$ = rate of change of $\vec{r}_{B/A}$ in the rotating frame

It is worth pointing out that the angular velocity is the same for any point fixed to the body. If point A and point B are both fixed to the same body, then $\vec{r}_{B/A} = 0$.

The equation of acceleration between two points is:

$$\vec{a}_B = \vec{a}_A + \vec{a}_{B/A} \tag{3.3}$$

where $\vec{a}_{B/A}$ is the is the relative acceleration, and it can be represented as:

$$\vec{a}_{B/A} = \vec{\alpha} \times \vec{r}_{B/A} + \vec{\omega} \times \left(\vec{\omega} \times \vec{r}_{B/A}\right) + 2\vec{\omega} \times \vec{\dot{r}}_{B/A} + \vec{\ddot{r}}_{B/A} \tag{3.4}$$

where:

$\vec{\alpha}$ = angular acceleration of the body reference frame
$\vec{\omega}$ = angular velocity of the body reference frame
$\vec{r}_{B/A}$ = distance vector from A to B
$\vec{\dot{r}}_{B/A}$ = rate of change of $\vec{r}_{B/A}$ in the rotating frame
$\vec{\ddot{r}}_{B/A}$ = rate of change of $\vec{\dot{r}}_{B/A}$ in the rotating frame

The four terms in the RHS following the sequence are called the tangential acceleration, the centripetal acceleration, the Coriolis acceleration and the radial acceleration. For a rigid body whose angular velocity is very small, the centripetal acceleration term can be neglected as the square product of the angular velocity approximately equals to zero. The Coriolis acceleration is generated by the Coriolis effect, which is a deflection of moving objects when they are viewed in a rotating reference frame. So if observed from the rigid body local frame, the point on the rigid body has no Coriolis effect and the Coriolis acceleration term can be neglected. Also, the last term is zero for a rigid body. Thus, Eq. (3.3) can be rewritten as:

$$\vec{a}_B = \vec{a}_A + \vec{\alpha} \times \vec{r}_{B/A} \tag{3.5}$$

In the three dimensional coordinate system, the acceleration of point A forms the vector as:

$$\vec{a}_A^T = \begin{bmatrix} a_x^A & a_y^A & a_z^A \end{bmatrix}^T \tag{3.6}$$

Also, the angular acceleration is written as:

$$\vec{\alpha}_A^T = \begin{bmatrix} \alpha_x^A & \alpha_y^A & \alpha_z^A \end{bmatrix}^T \tag{3.7}$$

Similarly, the acceleration at point B can also be written as a vector form. If the LHS of Eq. (3.5) is known, then there are six unknowns in the RHS, the system is undetermined. If the acceleration of a

third point on the rigid body is known, then there are six known value against six unknowns. The number of knowns is equal to the number of unknowns, and the system seems determined. However, there is redundancy in the equations. Thus, the accelerations of at least one additional point are required to solve the system, but there will be nine knowns against six unknowns, and the system is overdetermined. The least squares method is then used to approximate the solutions.

3.2 Coordinate Transformation

The rotation of a rigid body is often described using an angular parameter such as angular displacement, velocity and acceleration, with respect to the body reference frame. Yet, the pure translational motion is described with respect to a coordinate frame that is fixed. So when there is both translational and rotational motion, a method is applied to transfer the coordinates from a rotating frame into a fixed frame. In order to describe the method, the concept of Euler angles is first introduced. The Euler angles

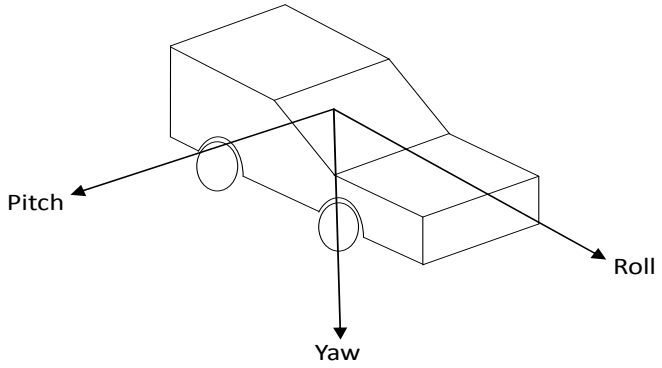


Figure 3.1: Yaw, pitch and roll angles.

are three angles introduced by Leonhard Euler to describe the orientation of a rigid body. They are typically denoted as ψ, θ and ϕ. The three angles represent a sequence of three elemental rotations, i.e., rotations about the axes of a coordinate system. In the automotive industry, the yaw-pitch-roll angles are often used to describe vehicle orientation (Figure 3.1). When calculating the orientation, the Euler angles work in the selected sequence. In a three dimensional coordinate system, the yaw-pitch-roll (also called 3-2-1) sequence transforms the coordinate in the following way: first, the frame rotates around x axis with an angle ϕ. The geometric relationship between the new coordinates and the old ones is:

$$\begin{Bmatrix} x_1 \\ y_1 \\ z_1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} \tag{3.8}$$

where x_1, y_1 and z_1 are the new coordinates. The next step is to rotate around y axis with an angle θ. The coordinates derived in the last step are used as the datum in this rotation, and the formulation is:

$$\begin{Bmatrix} x_2 \\ y_2 \\ z_2 \end{Bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{Bmatrix} x_1 \\ y_1 \\ z_1 \end{Bmatrix} \tag{3.9}$$

In the last step, the reference frame rotates around z axis with an angle ψ, whose equation is as following:

$$\begin{Bmatrix} x_3 \\ y_3 \\ z_3 \end{Bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x_2 \\ y_2 \\ z_2 \end{Bmatrix} \tag{3.10}$$

The equations above can be written in the form:

$$\vec{r}_3 = R_3 R_2 R_1 \vec{r} \tag{3.11}$$

where R_1,R_2 and R_3 are the transformation matrix in Eq. (3.8), (3.9) and (3.10) respectively. The product of the three matrices formulates the final transformation matrix. It describes the rotations regarding a rotating frame. If the rotations are about space fixed axes, the order of the rotations is reversed and the transformation matrix becomes the following; see Baruh[26].

$$R = R_1 R_2 R_3 = \begin{bmatrix} \mathrm{c}\psi\mathrm{c}\theta & \mathrm{s}\psi\mathrm{c}\theta & -\mathrm{s}\theta \\ -\mathrm{s}\psi\mathrm{c}\theta + \mathrm{c}\psi\mathrm{sin}\theta\mathrm{s}\phi & \mathrm{c}\psi\mathrm{c}\phi + \mathrm{s}\psi\mathrm{s}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{s}\phi \\ \mathrm{s}\psi\mathrm{s}\phi + \mathrm{c}\psi\mathrm{s}\theta\mathrm{c}\phi & -\mathrm{c}\psi\mathrm{s}\phi + \mathrm{s}\psi\mathrm{s}\theta\mathrm{c}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix} \tag{3.12}$$

where $s = sin$ and $c = cos$.

Similarly, the Euler angle rates (first derivative of Euler angles) have the following relationship with the angular velocity:

$$\omega_x = \dot{\phi} - \dot{\psi}\sin\theta \tag{3.13}$$

$$\omega_y = \dot{\theta}\cos\phi + \dot{\psi}\cos\theta\sin\phi \tag{3.14}$$

$$\omega_z = -\dot{\theta}\sin\phi + \dot{\psi}\cos\theta\cos\phi \tag{3.15}$$

where:

$$\vec{\omega} = \text{angular velocity expressed in the body fixed frame}$$

The equations can be rearranged to solve for Euler angle rates:

$$\begin{Bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{Bmatrix} = \begin{bmatrix} 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \\ 0 & \cos\phi & -\sin\phi \\ 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \end{bmatrix} \begin{Bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} \tag{3.16}$$

If the angular velocities of the rigid body are known, the above equations become ODEs and the Euler angle rates and Euler angles can be solved using an ODE solver.

3.3 Equations of Motion

The equations of motion are equations that describe the behavior of a MBS in terms of its motion as a function of time. The equations for translation

$$\sum \vec{F} = m\vec{a}_G \tag{3.17}$$

where the subscript G stands for the center of mass, work equally well in two dimensional and three dimensional coordinate systems. However, the rotational equations are not so simple. First, the moments of inertia are defined as:

$$I_{xx} = \int \left(r_y^2 + r_z^2\right) dm \tag{3.18}$$

$$I_{yy} = \int \left(r_x^2 + r_z^2\right) dm \tag{3.19}$$

$$I_{zz} = \int \left(r_x^2 + r_y^2\right) dm \tag{3.20}$$

where $r_i (i = x, y, z)$ is the length along the coordinate axes, and m is the mass of the object. Also, the cross-products of inertia are defined as:

$$I_{xy} = \int r_x r_y dm \tag{3.21}$$

$$I_{xz} = \int r_x r_z dm \tag{3.22}$$

$$I_{yz} = \int r_y r_z dm \tag{3.23}$$

Now the inertia can be written into a matrix from as:

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \tag{3.24}$$

where I is called the inertia tensor. The angular momentum is expressed as:

$$\vec{H}_G = I_G \vec{\omega} \tag{3.25}$$

The combined translational and rotational dynamics of a rigid body can be expressed by the Newton-Euler equations:

$$\sum \vec{F} = m\dot{\vec{v}}_G + \vec{\omega} \times m\vec{v}_G \tag{3.26}$$

$$\sum \vec{M}_G = I_G \vec{\alpha} + \vec{\omega} \times I_G \vec{\omega} \tag{3.27}$$

where:

$\vec{F}$ = total force acting on the center of mass
m = mass of the rigid body, a three by three diagonal matrix
$\vec{v}_G$ = translational velocity of the center of mass
$\dot{\vec{v}}_G$ = translational acceleration of the center of mass
$\vec{M}_G$ = total torque acting about the center of mass
I_G = inertia tensor
$\vec{\alpha}$ = angular acceleration
$\vec{\omega}$ = angular velocity

The second term on the RHS of Eq. (3.26) is considered if a rotating frame is used for $\vec{v}$. The total torque is the derivative of the angular momentum, and the second term in the RHS is called the *gyroscopic moments*.

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = R \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} \tag{3.28}$$

Note that $\sum \vec{F}$ and $\sum \vec{M}$ are functions of x, y, z, ψ, θ, ϕ and their derivatives, thus an ODE solver is also required. In addition, the velocity in the body fixed frame has the relationship with the displacement in the global frame shown in Eq. (3.28), where R is the transformation matrix which transforms the coordinates in a body fixed frame to global frame.

Chapter 4

Drive File Development

4.1 Displacement Driven Simulation

The VMAST resembles the real shaker by matching the test accelerations. However, instead of adjusting the acceleration magnitude with a feedback loop, it uses time-history displacement signals which are recovered from the test accelerations, and their corresponding accelerations closely match those from the test, which promises the accuracy of the simulated inertia load.

It seems more time efficient to drive the dynamic simulation with accelerations directly, as the displacement conversion is omitted. In fact, the experimental results showed that the test acceleration is not suitable for driving the simulation. First of all, the test accelerations can not be used unprocessed, as the numerical errors buried in the data are destined to cause faulty results. However, the numerical processing on the acceleration data is not easily done because there is no criterion as a reference. Unlike displacements, which represent some physical relationships, such as the chassis rigidity and orientations that can be easily observed and controlled, the accelerations don't obviously reveal such relationships. Thus, it is very challenging to control the numerical processing on the acceleration data, and this is why the accelerations have to be converted.

The number of driving displacement signals depends on the number of accelerometers that were used to collect the test accelerations. The accelerometer records data in its own coordinate frame on the three axial directions (x, y and z). If the vehicle chassis motion was recorded by three accelerometers, the driving displacement should correspondingly have nine signals.

4.2 Drive File Development

This section describes the details of the conversion from acceleration to displacement. The powertrain parameter is adopted from the Chrysler PF platform, 2.4 FWD, and the road profile is CPG010 (Chrysler

indoor road test data, collected on Chelsea Proving Grounds). This transverse powertrain has three mounts (each labeled as A, B and C), two of which are in the front, supporting the engine. The left and right side are defined by the driver's view while sitting in the driver's seat. The last one holds the transmission. The accelerometers are installed on the vehicle chassis right beside the mount locations, and labeled as A, B and C; see Figure 4.1. The road profile contains nine pieces of acceleration data. Each three describe the axial translational accelerations of one accelerometer in its local coordinate frame. The input parameters were measured in a global fixed frame, see Table 4.1 and 4.2 for details. It is worth pointing out that although the initial location of the accelerometers were measured in a different coordinate frame than accelerations, there is no conflict as the locations are only used to find out the relative distance.

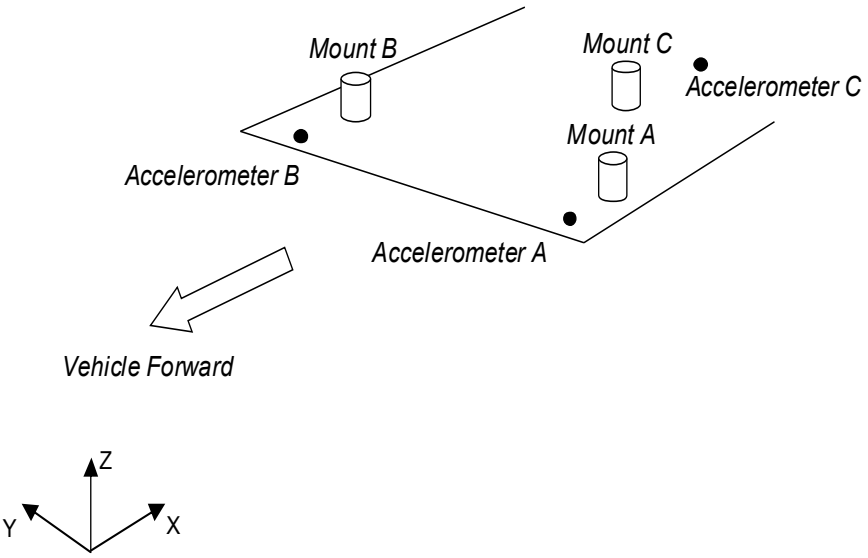


Figure 4.1: Schematic of chassis-side mount and accelerometer locations.

Table 4.1: Vehicle chassis parameters.

Item Name	Location $[x,y,z]$ [mm]	Item Name	Location $[x,y,z]$ [mm]
Left Front Mount A	[-182,-453,370]	Left Front Accelerometer A	[-590, -455, 280]
Right Front Mount B	[-200, 492, 391]	Right Front Accelerometer B	[-590, 490, 280]
Rear Mount C	[184, -131, -86]	Rear Accelerometer C	[515, -225, -80]

Table 4.2: Road test parameters.

Road Test Name	CPG010
Unit	g
Number of Channels	9
Time Length	235.998 s
Sample Rates	512 Hz
Channel Number	Polarity
Channel 10	-1
Channel 11	1
Channel 12	1
Channel 13	-1
Channel 14	1
Channel 15	1
Channel 16	-1
Channel 17	1
Channel 18	1

The drive file development process can be outlined using a flowchart, see Figure 4.2. A five-step process describes the generation of the final displacement signals (drive file). This algorithm is programmed using MATLAB®.

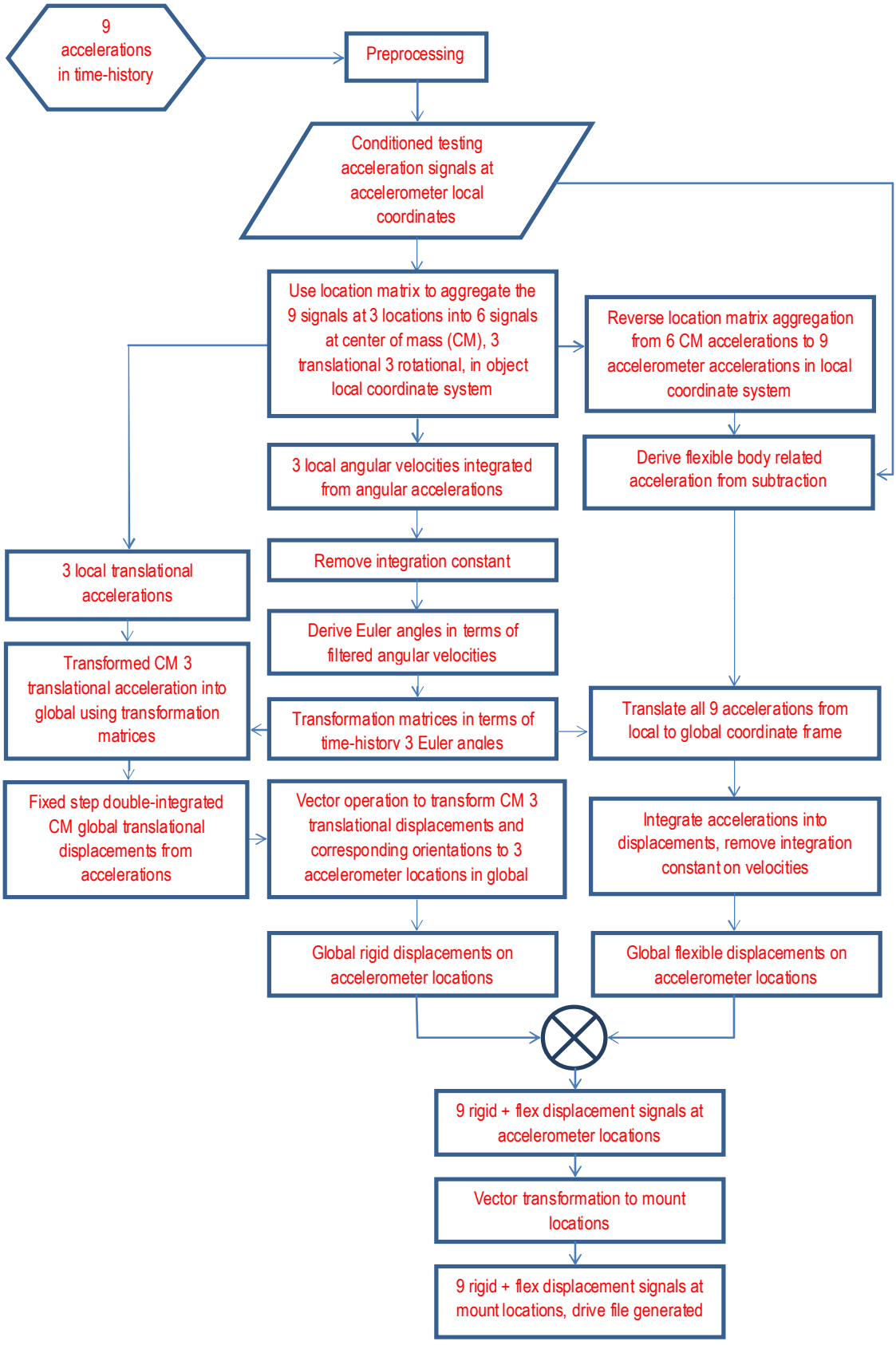


Figure 4.2: Drive file generation procedure.

Step 1: Preprocessing of Input Signals

First, each acceleration data is checked and corrected for any error during collection, e.g., the first data point must be zeroed out. Next, the necessary unit conversions are completed; the gravity unit g is transformed to mm/s^2. A low-pass Butterworth filter with the cut-off frequency of 50Hz is applied to eliminate the noise in the raw data. The cutoff frequency is chosen based on experimental results to effectively remove the noise while preserving the desired content. The pre-processing script also assigns channel numbers and polarities to the input accelerations, to make sure each acceleration matches with the correct axis and orientation of corresponding accelerometer. A total of nine input acceleration signals, three for each accelerometer, will be passed to the next stage.

Step 2: Aggregation at Center of Rotation

As the vehicle chassis is initially assumed to be a rigid body, its motion can be expressed with six co-ordinates, i.e., three translational and three rotational movements about a reference point fixed in the chassis. Theoretically, the reference point can be any point in the chassis coordinate frame, so in this case, the point is chosen as the centre of mass (CM) of the powertrain. One question raised is how to solve this over-determined system, which has six output accelerations at the reference with nine inputs from the accelerometers. This is done by ignoring the centripetal acceleration terms, and using a least-squares approach to find the best match of the accelerometer data to a sum of linear accelerations of the reference point plus the tangential terms due to angular acceleration, as shown in Eq. (4.1).

$$\begin{Bmatrix} \vec{a}_A \\ \vec{a}_B \\ \vec{a}_C \end{Bmatrix} = \begin{bmatrix} \mathbf{I} & -\tilde{r}_{A/G} \\ \mathbf{I} & -\tilde{r}_{B/G} \\ \mathbf{I} & -\tilde{r}_{C/G} \end{bmatrix} \begin{Bmatrix} \vec{a}_G \\ \vec{\alpha} \end{Bmatrix} \tag{4.1}$$

where:

$\vec{a}_A =$ accelerations of accelerometer A
$\vec{a}_B =$ accelerations of accelerometer B
$\vec{a}_C =$ accelerations of accelerometer C
$\mathrm{I} = 3 \times 3$ identity matrix
$\vec{a}_G =$ acceleration of CM
$\vec{\alpha} =$ angular acceleration of the vehicle chassis

Because the centripetal terms depend on the square of the angular velocity, for small motions, they will be much less significant than the tangential terms. Note the tilde represents the skew-symmetric matrix,

which is 3×3 and represents the cross product of $\vec{\alpha} \times \vec{r}$. Take the matrix of point A as an example, that:

$$-\tilde{r}_{A/G} = \begin{bmatrix} 0 & -r_z^{A/G} & r_y^{A/G} \\ r_z^{A/G} & 0 & -r_x^{A/G} \\ -r_y^{A/G} & r_x^{A/G} & 0 \end{bmatrix} \tag{4.2}$$

where $\vec{r}_{A/G}$ is a vector containing the coordinates $(x,\ y,\ z)$ of the relative distance between rotational center and point A. Thus $\vec{a}_A$ is expressed as following:

$$a_x^A = a_x^G - r_z^{A/G}\alpha_y + r_y^{A/G}\alpha_z \tag{4.3}$$

$$a_y^A = a_y^G + r_z^{A/G}\alpha_x - r_x^{A/G}\alpha_z \tag{4.4}$$

$$a_z^A = a_z^G - r_y^{A/G}\alpha_x + r_x^{A/G}\alpha_y \tag{4.5}$$

In MATLAB®, the matrix multiplication $Ax = b$ can be written as $x = A\backslash b$, and x in the latter one is computed using least squares method. In this way, $\vec{a}_G$ and $\vec{\alpha}$ in Eq. (4.1) are calculated.

Once the six motions at the reference are determined, the process is reversed, in order to re-generate the accelerations on each of the accelerometers. A 3×3 correlation plot visually describes the fitness of the match between the test-collected and re-generated accelerations, as shown in Figure 4.3. The rows from top to bottom are accelerometer locations A, B and C and the column from left to right are x, y and z directions. This validation ensures that the rigid body assumption covers most of the chassis motion, as all local flexibilities are lost during the transformation. It will also check for possible errors in pre-processing. Typically, test-collected and re-generated data match well, because motion of the vehicle components is dominated by the chassis response to the ground condition.

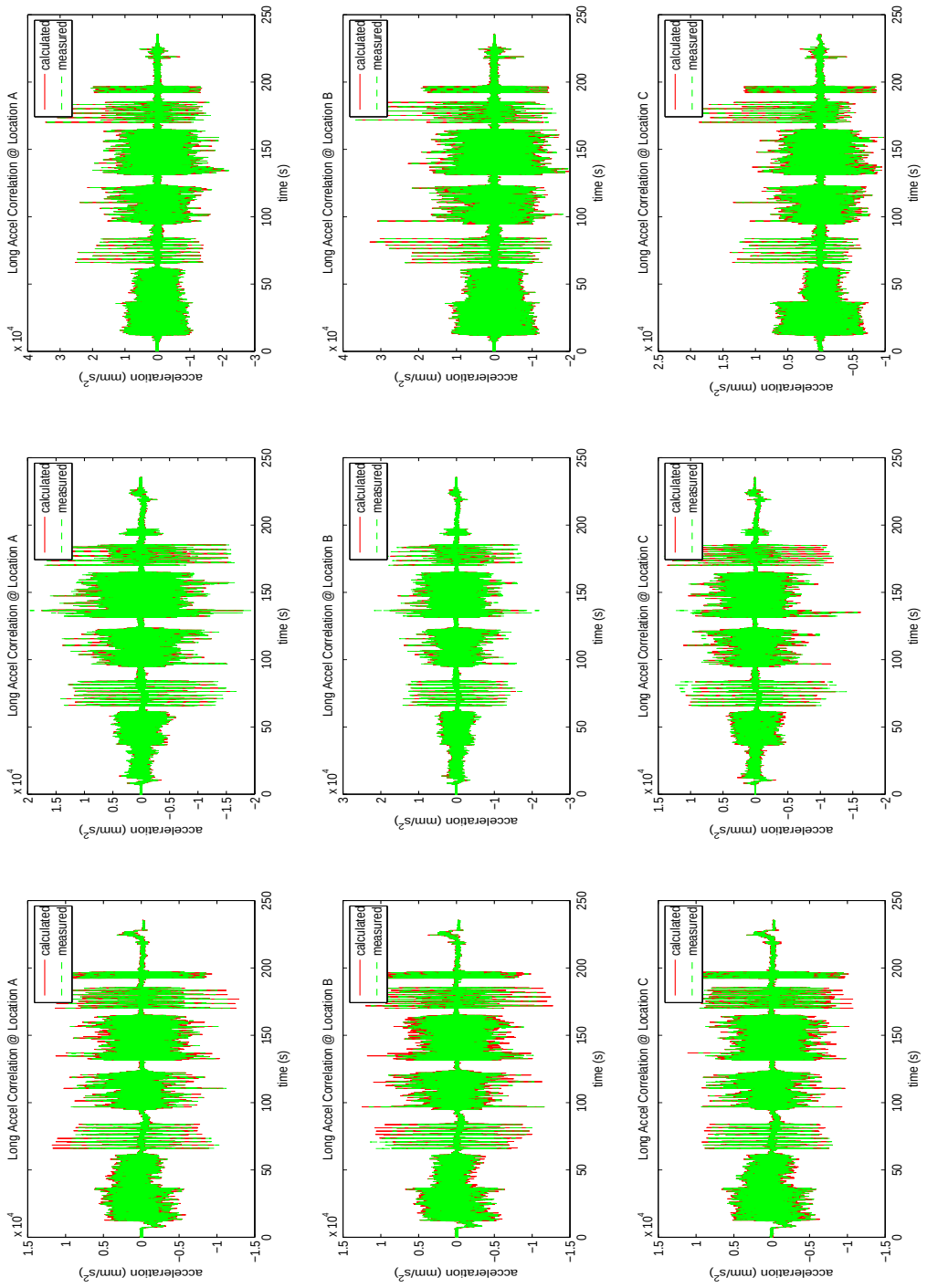


Figure 4.3: Comparison between rigid and measured accelerations.

Step 3: Development of Transformation Matrices

An accelerometer only reads measurements in its local coordinate system. When attached to the chassis, the coordinate system used by the accelerometer coincides with that of the vehicle chassis. Due to the relative rotation between the fixed ground and the moving vehicle chassis, all collected accelerations need to be translated into the ground coordinate system, in order to describe the global motion of the chassis. This is where the transformation matrix, moving vectors from one coordinate frame to the other, is needed. It is worth pointing out that only one frame is needed to present chassis-side accelerations and chassis flexible displacements (introduced in the following content), as the accelerometers remain mostly aligned and the flexible displacements are assumed to be mostly in translation.

The transformation matrix uses Euler angles to determine the orientation information, and 3-2-1 Euler angles (also known as yaw-pitch-roll angles) are used, because they are common expressions of vehicle rotation. To determine the Euler angles, local angular accelerations at the reference point are integrated over time to determine the local angular velocities. A running mean removal filter is necessary to eliminate low-frequency content that will inflate the result. The filter window size is selected depending on the magnitude of the resultant Euler angles. For this road test event, the pitch and roll angles are tested to be typically within 3 degrees (0.04 radians). The yaw angle is less significant in this vibration dominated event, as the yaw angle commonly represents the vehicle cornering, where the frequency is very low and the inertia effect is small. Thus, the filter running window size is 513 data points for this event. While removing the integration constant, the filter might slightly spoil the original signal. To fix this problem, a scaling factor s is applied to the filtered velocities. That:

$$v_{CM}^{s} = s v_{CM} \tag{4.6}$$

$$s = \frac{RMS(a_{CM})}{RMS(v'_{CM})} \tag{4.7}$$

where:

v_{CM}^{s} = filtered and scaled velocities at center of mass
v_{CM} = filtered velocities at center of mass
a_{CM} = accelerations at center of mass
RMS = root mean square

The six scale factors for six signals are listed in Table 4.3. It is shown that the numbers are all close to one, which means the difference in the RMS is small, thus the filter does not affect the useful data much.

Table 4.3: Rigid motion velocity scale factors for event CPG010.

Velocity	Scale Factor
V_x	1.0150
V_y	1.0166
V_z	1.0088
V_{rx}	1.0058
V_{ry}	1.0208
V_{rz}	1.0291

The processed angular velocities are then used to solve the kinematic differential equations, relating angular velocity, Euler angles, and Euler angle rates, as shown in Eq. (4.8).

$$\begin{Bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{Bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} \tag{4.8}$$

The differential equations are solved by an ordinary differential equation (ODE) solver in MATLAB®, and the solver implements the improved Euler (Heun's) method of order 2. Unlike other solver, such as *ode45*, this non-adaptive method uses fixed step size which allows the employment of each data point in the calculation, thus making it suitable for processing the test data. The returned Euler angles form a time-history solution, see Figure 4.4 where the data are the roll (blue), pitch (green) and yaw (red) angle respectively. The Euler angles can be integrated into a series of transformation matrices from the local to the global coordinate system. The well-known 3-2-1 transformation matrix is used; for more information see Table 7.1 in Baruh[26].

If a vector is to be translated back from the global to the local coordinate system, the existing transformation matrices need to be inverted, which fortunately is the same as the transpose for R. Thus, a set of transposed transformation matrices is also stored for further use.

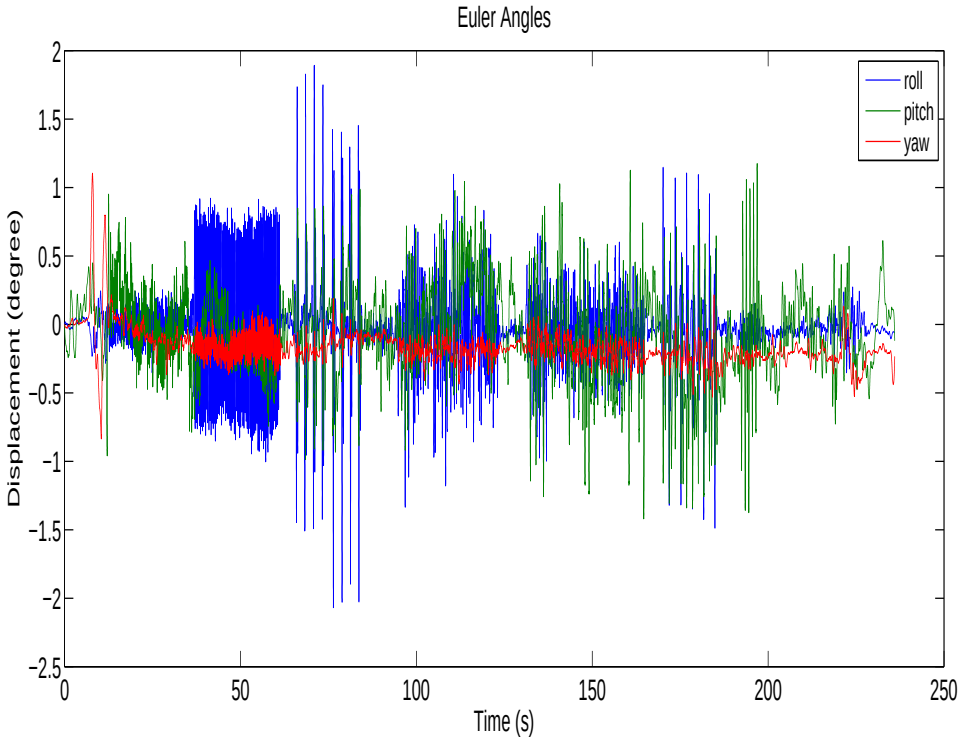


Figure 4.4: Euler angles of vehicle chassis derived for event CPG010.

Step 4: Determination of Accelerometer Locations

With the translation between local and global coordinate systems established, the local translational accelerations at the CM can be converted to global translational accelerations. By integrating these accelerations twice, the displacements of the CM in the fixed global reference frame are obtained. Since the initial coordinates of accelerometer locations are measured from the vehicle, the time history of these locations can be determined by adding the transformed global relative displacements in each direction to the global CM location. Results often show that the computed magnitudes of reference point motion are much larger than in reality, due to the integration of low-frequency noise in the data. For inertia load acquisition purpose, this is not a major concern, as the final target is to regenerate the closest accelerations. The detailed operation is explained in the following.

The time-history global locations of the CM are in a series of vectors, starting from the ground origin to the CM. As a vector operation, the distance from the ground origin to each of the accelerometer points must equal the sum of the global CM location and the global CM-to-accelerometer location vectors. The latter can be translated from the local CM-to-accelerometer vectors, which are fixed in the local frame and equal to the coordinate differences between the CM and each of the accelerometers. In this way, the time-history locations of each accelerometer are determined.

Step 5: Flexibility Recovery and Drive File Generation

At the beginning of the process, local chassis flexibilities, included in the readings of the accelerometers, were discarded during the rigid-body assumption. To improve the quality of the output data, it is expected that this flexibility data can be recovered. Continuing with the idea of adding length vectors, local flexible displacements are recovered by integrating differences between test collected accelerations and regenerated rigid accelerations, and then translated and added to calculated global accelerometer locations. However, the overall rigidity of the chassis should not be broken. For the specific testing vehicle, the overall flexibility is restricted by a small fixed value of maximum relative displacement between any two of the accelerometers. This value is based on experience and is vehicle dependent. This criterion introduces another running mean removal filter that adjusts the local original and regenerated acceleration differences. The window size of the filter depends on whether the relative displacements satisfy the target. For this vehicle, the flexible displacement is restricted to be within 3mm. So the window size is selected to be 129 data points. Figure 4.5 demonstrates the flexible displacements on accelerometer *A*, while Figure 4.6 displays the relative displacements between each accelerometer. From the top subplot in Figure 4.6, the variation of the relative displacement is very small (within 3mm) so visually it is almost straight, but the magnified relative distance between accelerometer *B* and *C* (bottom subplot) shows the fluctuation and it is exactly within the predefined value.

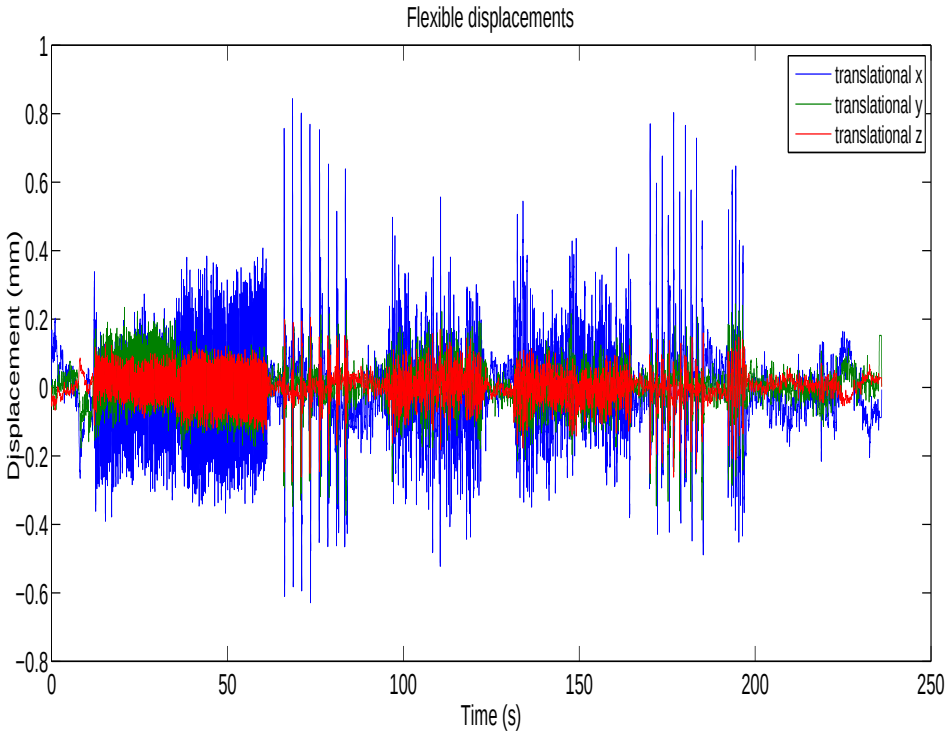


Figure 4.5: Flexible displacements on accelerometer *A* location.

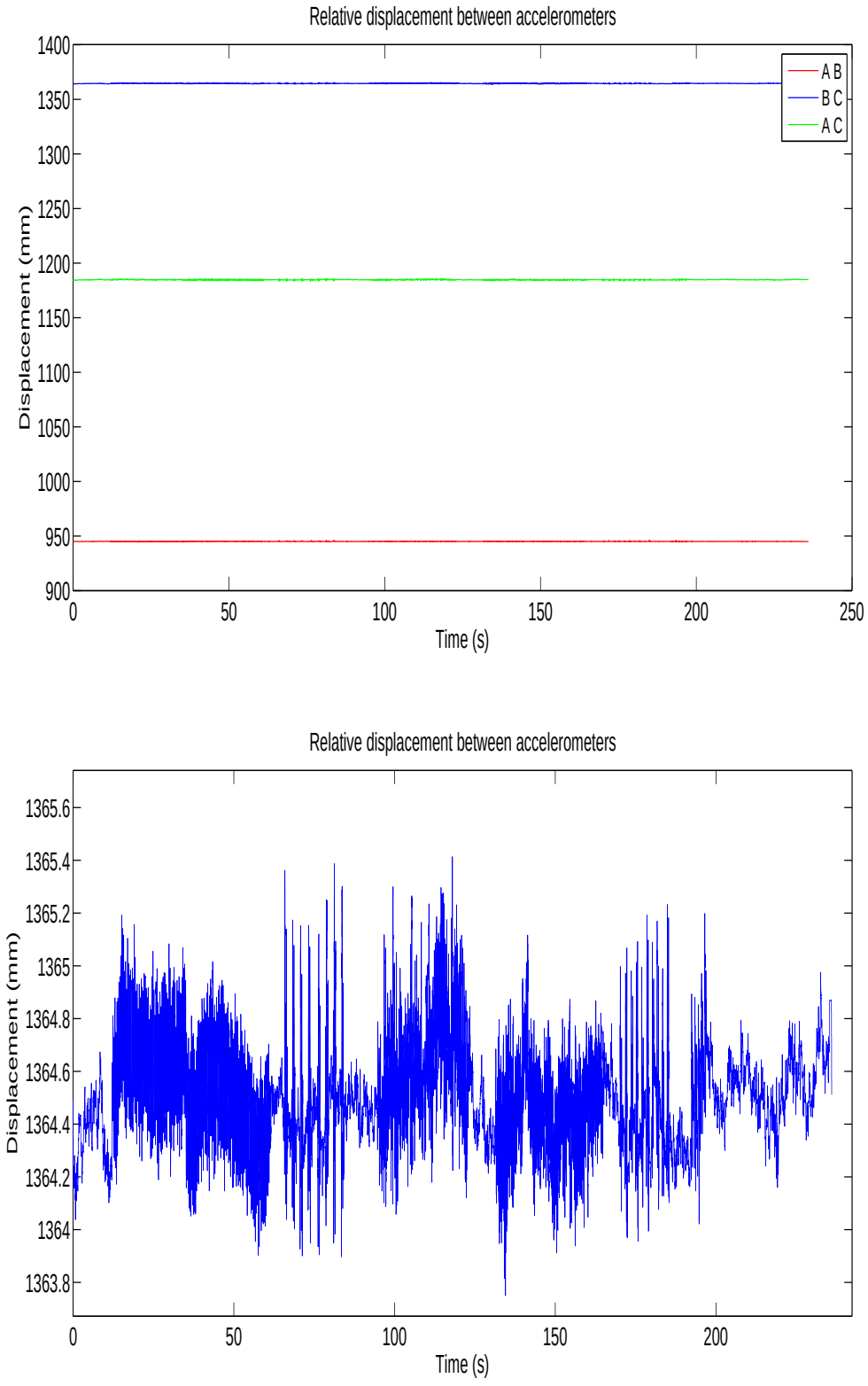


Figure 4.6: Relative displacements between accelerometers (above) and partial magnification (below).

After the nine flexibility-considered displacements of three accelerometers have been generated, the displacements are again differentiated and translated into local accelerations. The nine newly generated accelerations are again compared with the original local accelerations, in terms of a 3×3 correlation plot, see Figure 4.7. This plot shows the overall improvement in terms of acceleration comparisons. The error between the reproduced and measured accelerations is calculated numerically as

$$error = \sqrt{\sum \left(\frac{RMS(a_m^i)}{RMS(a_r^i)} - 1 \right)^2} \tag{4.9}$$

where:

$a_m =$ measured accelerations
$a_r =$ reproduced accelerations
$i = 1,2,\ldots,9$

It is obvious that the smaller this accumulative error is, the better correlation the result has. The cumulative error for event CPG010 is 0.0759, so the average error for each reproduced acceleration data is 0.0253, which means the average RMS of the reproduced acceleration is 97.47% of that of the test acceleration. Therefore, this error is considered very small, and the reproduced accelerations are accepted.

As the accelerometers were placed close to the true locations of the engine mounts on the testing vehicle, it is reasonable to assume that the length vector from each individual accelerometer to its corresponding mount is rigid. With this assumption, translational displacements of each accelerometer can eventually be transformed to the correct location of each engine mount. The nine output displacements of the engine mounts, stored in what is called a 'drive file', will be used as the inputs for the dynamic simulation.

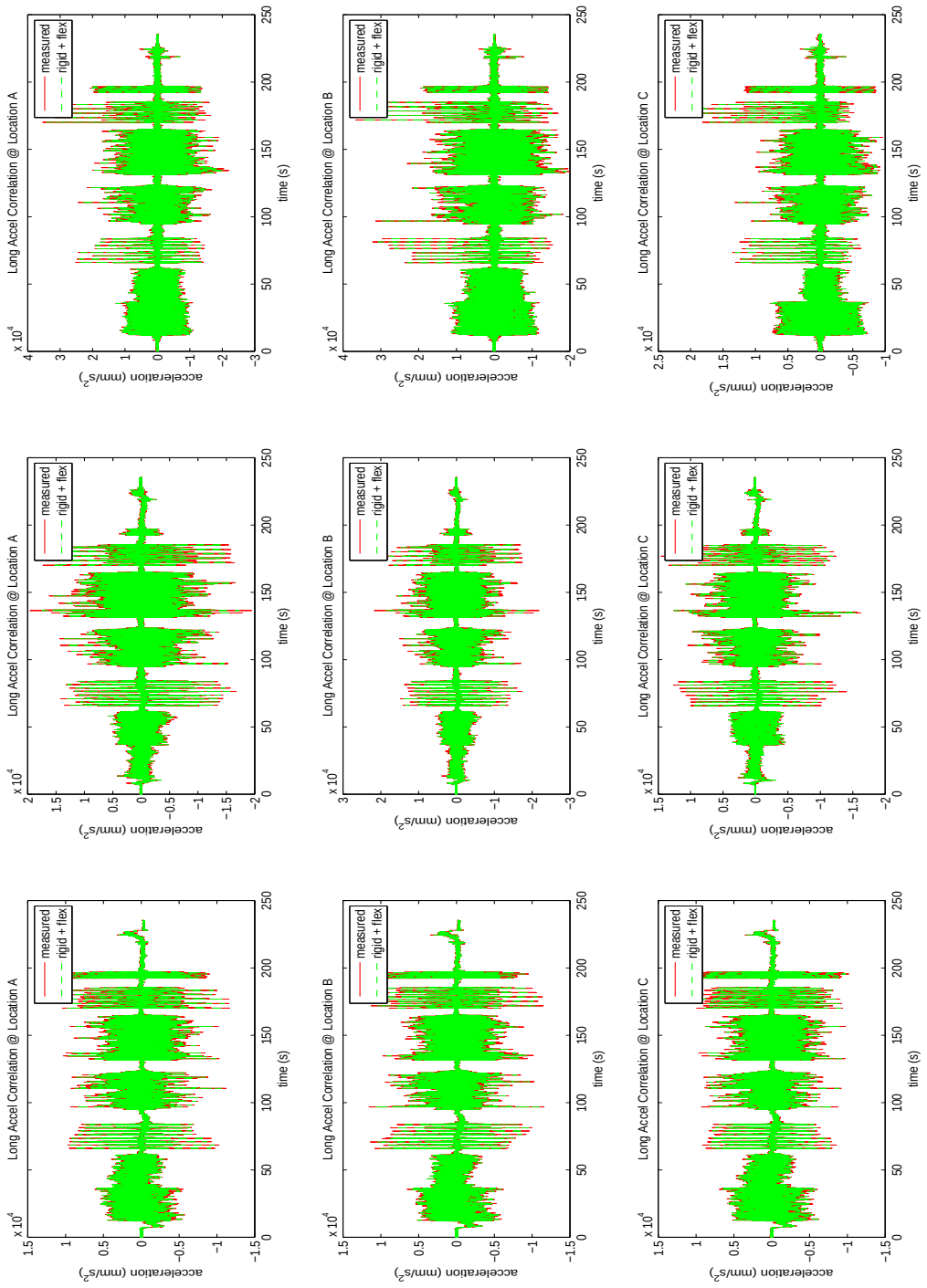


Figure 4.7: Comparison between flexibility-considered and measured accelerations.

4.3 Validation

4.3.1 Validation Against Artificial Data

To validate acceleration-displacement conversion algorithm, it is most straightforward to artificially create a set of input data, and compare the generated result with the known output. To this end, a set of six displacement signals are created. They are three translational motions and three rotational motions; all of them are occurring at one point in order to simulate the combined translational and rotational motion of a rigid body. All the motions have gradually changing frequencies and amplitudes, which are specified in Table 4.4.

Table 4.4: Artificial motions.

Motion	Frequency	Amplitude
X	10-30 Hz	150-50 mm
Y	10-30 Hz	150-50 mm
Z	10-30 Hz	150-50 mm
Roll	1-15 Hz	0.6-6 deg
Pitch	1-15 Hz	0.6-6 deg
Yaw	1-15 Hz	0.6-6 deg

A model representing the rigid body is built and simulated in MotionView®, the CM and virtual accelerometer locations are listed in Table 4.5. The resulting time-history accelerations and displacements at the accelerometer locations are collected. The accelerations are then used as the input for the drive file generation.

Table 4.5: Body CM and virtual accelerometer locations.

	Location $[x,y,z]$ [mm]
Center of Mass	[-223,25,145]
Accelerometer *A*	[-590,-455,280]
Accelerometer *B*	[-590,490,280]
Accelerometer *C*	[515,-225,-80]

Although the created motion is purely rigid, the generated drive file is still expected to catch up with the simulation result by recovering zero flexibility. The comparison between the final reproduced accelerations and artificial accelerations is shown in Figure 4.8. Overall, the reproduced accelerations (green) catch the original accelerations (red) very well. However, the magnitudes of the reproduced accelerations shrink slightly, due to the application of the filters, such as running mean removal, that a portion of the desired frequencies is eliminated. But this is inevitable in dealing with the numerical

signal. The current approach is to manually adjust the filter parameter until the most reasonable orientations along with the minimum error are obtained. The accumulative error for the accelerations is 0.1841, so the average error is 0.061, and the average RMS of the predicted acceleration is 93.86% of that of the artificial one. Although the correlation is worse than that of CPG010, it is still very good.

The comparison between the drive file and the original motion at accelerometer locations are displayed in Figure 4.9. Again, both the overall and partial magnified plots show very good correlation. The reproduced displacements almost match line to line with the artificial displacements. The accumulative error for displacements is 0.0891.

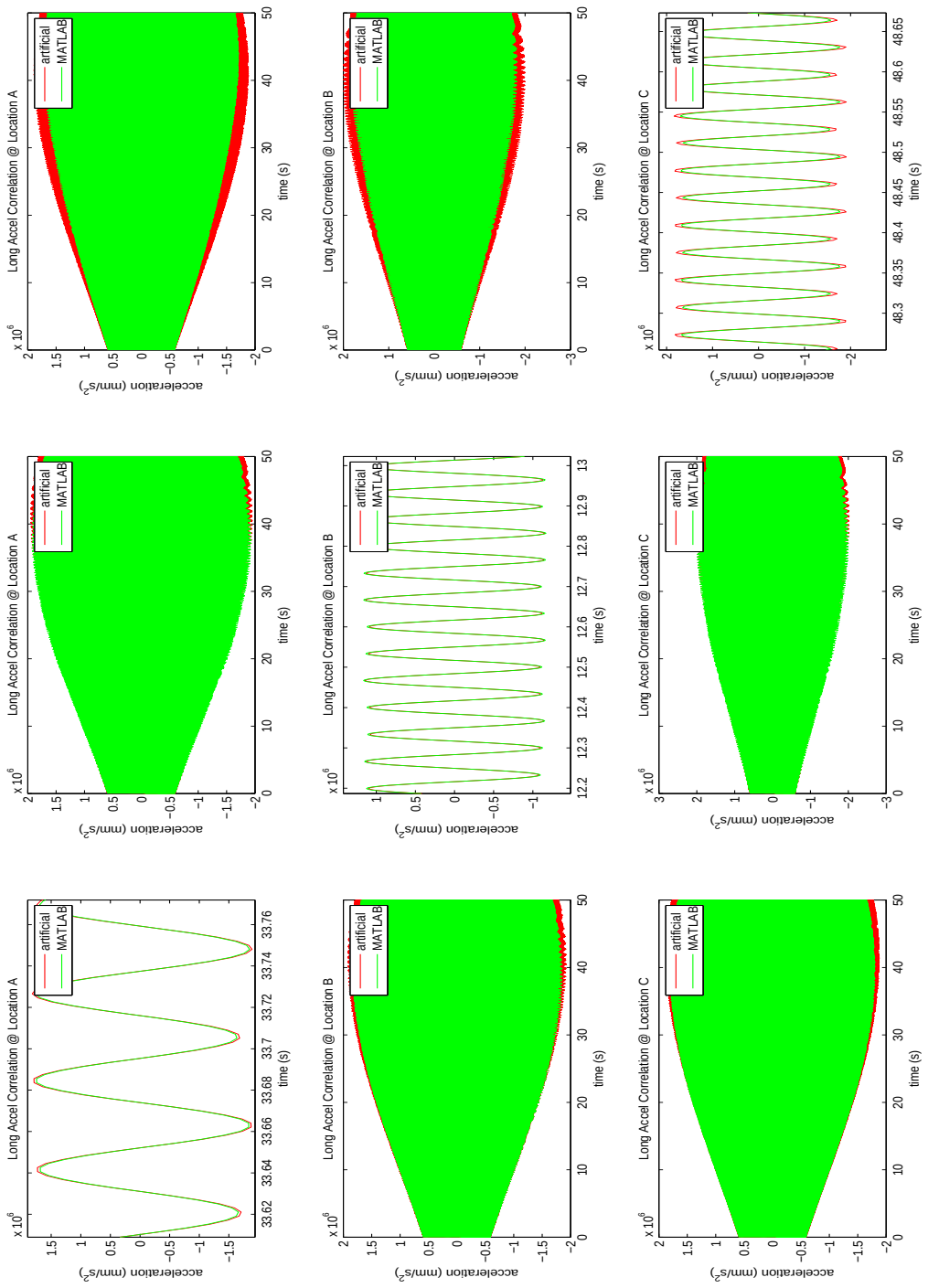


Figure 4.8: Comparison between simulated and artificial accelerations.

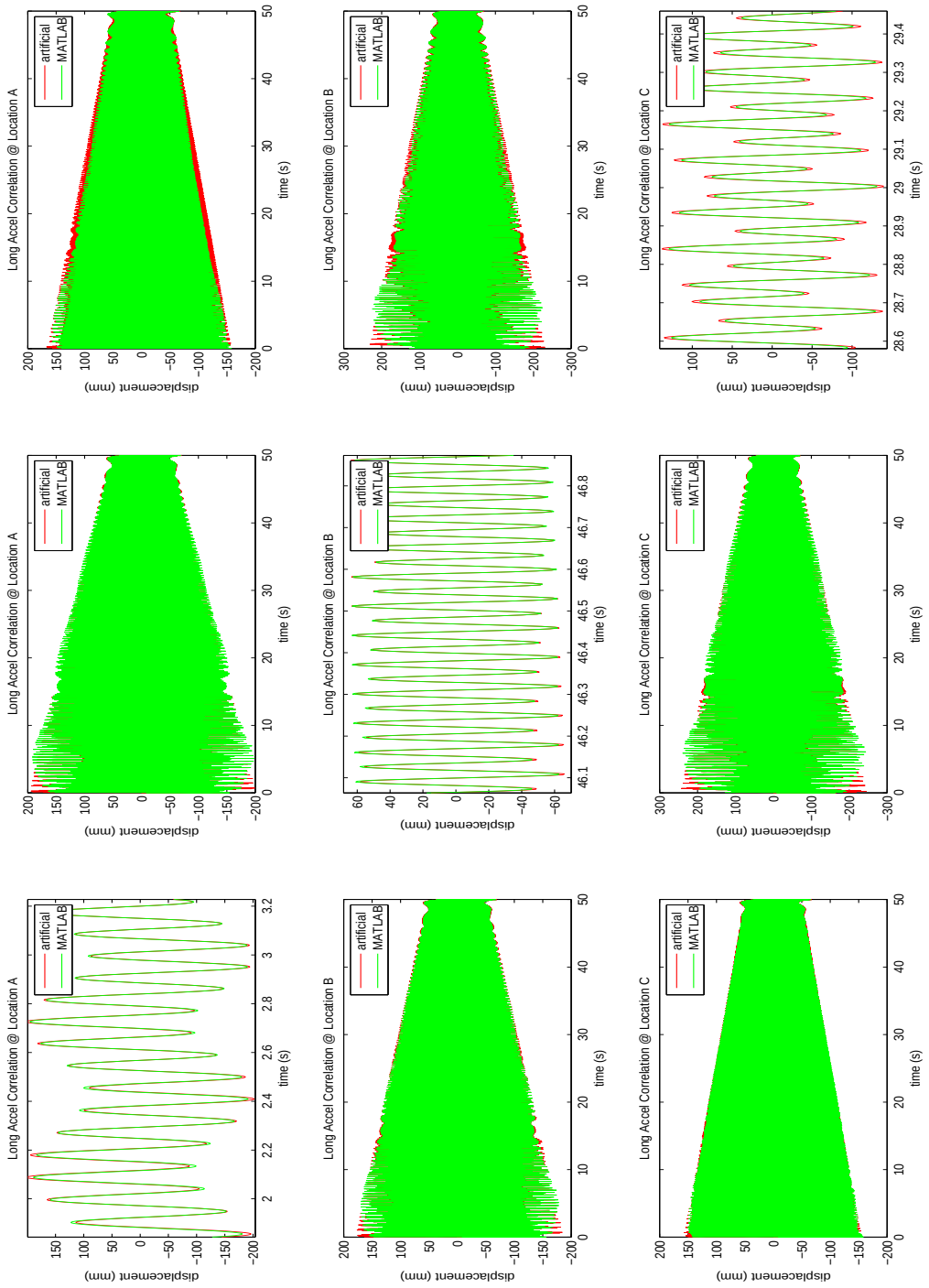


Figure 4.9: Comparison between flexibility-considered and measured displacements.

4.3.2 Validation Against Test Data

In the previous section, it is proven that a qualified drive file is generated for event CPG010. To further validate the drive file generation process, a total of fifteen events are tested. The error of the corresponding accelerations for each event is listed in Table 4.6.

Table 4.6: Drive file corresponding accelerations error for 15 events.

Road Test Event	Error
CPG010	0.0759
CPG015	0.0824
CPG021	0.0655
CPG032	0.0749
CPG03A	0.0888
CPG042	0.0884
CPG04H	0.0834
CPG05B	0.0513
CPG08M	0.0433
CPG08U	0.1477
CPG04P	0.0566
CPG04Q	0.0827
CPG500	0.1425
CPG574	0.0771
CPG579	0.0554

From the table, it can be seen that the largest cumulative error is 0.1477 amongst the fifteen events, which means the average error between the RMS of each reproduced and test acceleration signal is less than 5%. So the drive file is qualified for all fifteen events.

4.4 Discussion

The drive file development process is an integration of physical and numerical methods that successfully reproduces known accelerations in the form of displacements, in order to carry out dynamic simulation. During the process, the error buried in the raw data that could result in simulation failure is eliminated. This method is the core function of the VMAST.

In this section, some methods used in the drive file development process are discussed in detail. Besides, the assumptions and limitations are explained, due to their importance in comprehending the purpose of the VMAST. Also, possible improvements are discussed.

4.4.1 Filtering

There are two kinds of filters used: the Butterworth filter and running mean removal. Visually, the numerical error usually has a jagged look; the shape is mostly regular and the frequency is high, which is distinct from normal vibrations. The numerical error might result from the data collection process, e.g., insufficient precision of the apparatus. It also appears in the simulated result, and the causes can be the data precision, or the accumulated integration error. In this case, the raw acceleration data, as well as the simulated accelerations, are filtered with a low pass Butterworth filter to eliminate the noise.

The running mean removal works by subtracting the mean of a certain data length (window) from the central point of the window, to remove the trend in the data. Comparing with linear trend removal, the running mean removal has the advantage of removing 'local' trends that appear in partial lengths of the data. However, unlike the high-pass filter, the running mean removal does not capture frequencies with wavelengths that are integer multiples of the window length. The fluctuations in the data due to the vehicle motion are relatively preserved. However, the desired contents are more or less distorted with the application of the running mean removal. This could result in a problem in reproducing the vehicle chassis orientations, which is discussed in the following section.

4.4.2 Reproduced Chassis Orientations

Unlike the translation of the vehicle chassis, the reproduced rotational motions have to be constrained in a reasonable range in order to be realistically logical. However, rather than speaking of 'calculating' the vehicle chassis orientations, in fact, it is more suitable to refer to 'estimating'. Despite the existence of the algebraic kinematic equations, they are unable to be used directly, as the data is not necessarily to be 'theoretical' due to the numerical error brought by measuring error, integration error and chassis flexibility. Therefore, the numerical methods are used to 'trim' the data until it is physically reasonable. This is where the running mean removal and scaling are used, and this also causes a problem; the data can never be trimmed to precisely match the real values. This results in a limited application of the numerical methods. Due to the characteristic of the running mean removal filter, in the case where the low frequency rotational motion is large, or the low frequency rotational motion contributes the majority of the body motion, the affect on the error brought by numerical prediction will proportionally increase; thus, the accuracy of the data is reduced.

This is why the VMAST cannot be used to simulate large rotations, such as test CPG090, in which the vehicle is doing 'figure 8' cornering. All the road test events are 'vibration based', like driving through potholes, i.e., the magnitude of the rotational motion is very small and the frequency is high. However, the VMAST is developed to collect the inertia induced load initiated by vibration. For a passenger vehicle, the large rotational motions (larger than 10 degrees) are usually in low frequencies, and rarely happen in daily driving, so they are excluded in the VMAST simulation.

Improvements

As mentioned above, both the algebraic equations and the numerical methods fail to give the precise orientation. There is no breakthrough in the mathematical algorithms if the accuracy of the simulation is meant to be improved, so it has to be the improvements in data collection. As far as the current technology is concerned, the vehicle chassis could be equipped with some gyroscopic devices that measure the angular velocity directly. In this way, the orientations can be more precisely computed.

4.4.3 Flexibility Recovery

The motion caused by the twist of the chassis or frame, as well as the powertrain mount bracket's local deformation would ideally be captured by using an effective stiffness matrix. This would allow the deformation response to new loads to be determined during simulation. However, this would involve some sort of parameter identification technique, and it is expected that it would be both time consuming and challenging to obtain accurate results.

Therefore, a more time efficient and straightforward method is used. In this case, the vehicle chassis flexibility is assumed to be the translational deflections of the mount brackets in body fixed x, y and z directions. The flexibility is integrated from the accelerations, which is the difference between the test accelerations and calculated rigid accelerations. The running mean removal is applied on the intermediate integration result in order to prevent drifting in the final result. Because there is no measured data to compare to, the range of the deflection can only be estimated based on experience.

However, as the flexibility is very small (usually within five percent of the chassis local motion), the inaccuracy of the estimation would not affect the result. On the contrary, the quality of the drive file is improved with the flexibility recovered for the fifteen road tests.

As the deflections are very small, strain gauges might be modified to measure the displacements by attaching them to the mount brackets and aligned with the translational directions to further improve accuracy.

4.4.4 Drive File

In the dynamic simulation, the function of the drive file is to provide the accelerations of the vehicle chassis experienced during the test. As long as the drive file could fulfill this task, while maintaining the relative rigidity of the vehicle chassis, the resulting global displacement does not necessarily need to be restrained in a physically reasonable range; i.e., the absolute position is not a concern. A typical time-history plot of the drive file is shown in Figure 4.10. The overall time-history contains a low frequency drift. This is because the integration constant is too large, thus visually overwhelming the high frequency content.

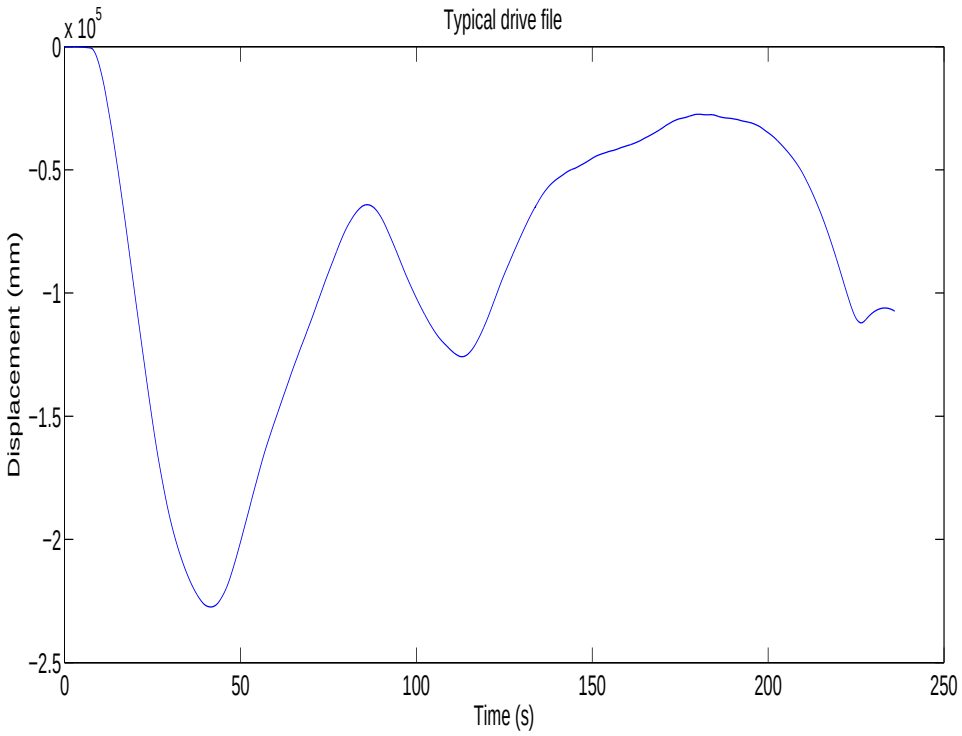


Figure 4.10: Typical drive file.

4.5 Software Application

The drive file generation process is coded in MATLAB®. In order to provide convenient access and operation to the source codes, the codes are complied to be an executable program, and its operation and functions are discussed.

To start the drive file generation, the test accelerations are required to be placed under the same folder as the program files. The name of the input files has to follow the format of 'road test event name' plus 'channel name' plus 'file format', e.g., 'lap_cpg010_10.dac', in order to be read by the program. Next, the user must input initial values of the variables, as well as the numbers of simulation parameters, which are listed in a text file; see Table 4.7.

If the program is used to run batch events, all the event names need to be input and stored in a string variable. Correspondingly, the channel numbers and polarities are multiple. The raw input data is processed to minimize error, especially the measurement error. This is where the initial processing is applied, and it provides the user with several options with the following commands: 'none' applies no initial processing; 'offset' offsets the data by the value of the first entry, thus the first entry is zeroed out; 'detrend' removes the linear trend in the data, unlike running mean removal, it calculates the trend based on the full length of the data; 'offset detrend' offsets the data followed by detrend. In the vibration based events, the measurement error, such as nonzero first entry and drifting, can be removed with initial

Table 4.7: Drive file generation inputs.

Item Name	Unit	Note
Frame-side mount locations	mm	Input coordinate [x, y, z]
Frame-side accelerometer locations	mm	Input coordinate [x, y, z]
Rigid body rotation center	mm	Input coordinate [x, y, z]
Event name(s)		Road test event name(s)
Event channels		Channel numbers
Polarity		-1 if test and simulation directions are opposite
Initial processing method		Numerical treatment to the input data
Butterworth filter frequency	Hz	Input data filter
Flexibility recovery		Switch on or off flexibility recovery

processing. On the other hand, the high frequency noise is filtered with a Butterworth low pass filter with a cut-off frequency of 50Hz. This program also provides the user with the option to turn off the flexibility recovery function, when needed.

As the program starts running, the first plot generated is the comparison between the measured and reproduced rigid body accelerations with respect to the vehicle chassis frame. Due to the relative rigidity of the vehicle chassis, if the mismatch is too large, the initial processing should be redone. If the initial processing methods provided do not work effectively, the input data can be manually processed in commercial software, such as nCode Glyph®.

Next, the first interactive dialog box opens, which is for the running mean removal of the integrated CM rotational accelerations; see Figure 4.11. The window size, dimension and edge treatment are defined by the user. The dimension of an array is the number of indices needed to select an element, so for the time-history data with $1 \times n$ array, the dimension is two.

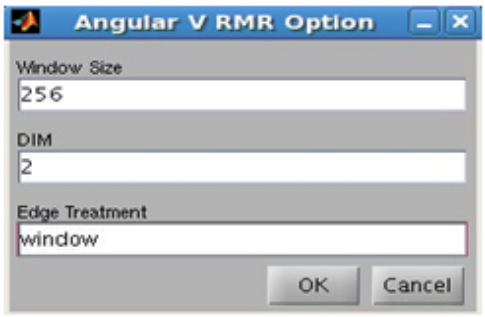


Figure 4.11: Angular velocity running mean removal dialog box.

It is important to optimize the running mean removed rotational velocities as the filter might remove

useful content from the original data. The optimization is done by applying scale factors to the filtered velocities, and to bring the RMS of the corresponding accelerations back to the same of the non-filtered ones. There are two RMS recovery approaches provided. The band-pass filter RMS recovery option recovers RMS only based on the band-pass filtered result. It provides the user with the opportunity to judge the most applicable frequency range so that the rest of the data is not affected. The RMS recovery applies the scale factor directly to the data without any filtering. The user can choose either of the two approaches by entering '1' in the corresponding box, while entering '0' for another. If the band-pass method is used, enter the frequency range and filter order, see Figure 4.12.

Figure 4.12: RMS recovery dialog box.

The reproduced Euler angles of the vehicle chassis are then plotted. If the angles are not satisfactory, the process can be redone starting from the running mean removal by adjusting the parameters. Once satisfactory Euler angles are obtained, the program will continue with the flexibility recovery or generate the rigid body drive file, depending on the user's choice. If the flexibility recovery is chosen, another running mean removal dialog box will pop up, where the purpose is to prevent integration constants in the flexible displacements. Again, there is the option to redo the filtering.

Once the flexible displacements are accepted, the plot showing the comparison of the flexibility included and test accelerations, as well as the error are displayed. The relative displacements between the mounts are also plotted to check the relative rigidity of the vehicle chassis.

In addition, the program provides the option to generate global rigid-body-based axial displacements on any location in the vehicle chassis by applying the appropriate vector operation. As mentioned in the previous content, there is no stiffness matrix for the vehicle chassis, and neither for the test data, except for the three specified locations. In this case, the flexible displacements of another location is estimated by the user according to the existing flexibility on the specific locations. It is carried out using the equation

$$flex = flex_A \cdot ratio_A + flex_B \cdot ratio_B + flex_C \cdot ratio_C \tag{4.10}$$

where the flexible displacements on the three locations are numerically combined. The ratio varies between 0 to 1, and is estimated based on the location and user's experience.

The outputs of the drive file development process, shown in Table 4.8, are saved under the corresponding folder. At this stage, the drive file has been generated, and can be used as the input to the dynamic shaking simulation.

Table 4.8: Drive file generation outputs.

Item Name	Format	Note
GLBL_MNT_LOCATION_	DAC	Frame-side mount displacements in the global frame (drive file) Name followed by channel numbers
ADD_LOC_DISP_	DAC	Global displacement of user defined location Name followed by axial directions (x, y and z)
GLBL_MNT_LOCATION	MAT	Drive file in MATLAB® time-series format Used for MATLAB® dynamic shaking
ADD_LOC_DISP	MAT	Global displacement of user defined location in MATLAB® time-series format

The time cost for drive file generation depends on the length of data and computation hardware. Normally, for a event with time length around 200 seconds (such as CPG010), the simulation time is about 20 minutes.

Chapter 5

Dynamic Shaking Model

In this chapter, the theory used for the MATLAB® dynamic shaking model will be introduced, followed by the description of its limitations. The next half of this chapter is focused on the MotionView® model.

5.1 MATLAB Dynamic Shaking Model

In the real case, the vehicle powertrain is sitting on the vehicle chassis through several flexible mounts (bushings) that hold the engine in place and absorb shocks from both the chassis and the powertrain. This model adopts the Chrysler PF 2.4L FWD powertrain (PF 2.4 for short), and there are three bushings: two of them are in the front, supporting the engine; one is in the rear holding the transmission. Thus, the virtual model consists of three parts: frame-side input motion, the bushing model, and the powertrain body, which is modeled as a rigid mass at the CM location. A schematic is shown in Figure 5.1. It is worth pointing out that the powertrain and bushing mechanisms are relatively simplified in this model, and this is discussed in the following content.

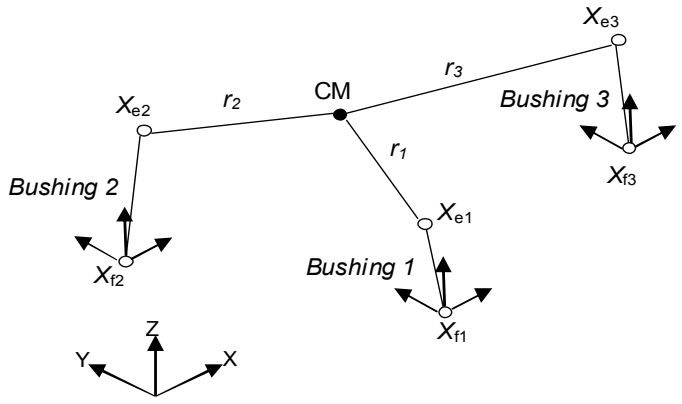


Figure 5.1: Schematic of the VMAST and powertrain dynamic shaking model.

The drive file is the input motion to this model and it acts on the frame-side locations of the bushings, which is in the fixed frame as described in the previous chapter. To determine the locations of the powertrain-side bushings, $\vec{x}_e$, in the fixed global coordinate frame, the vector operation is done by adding the relative displacement between powertrain CM and bushing locations $\vec{r}$ to the powertrain CM displacement $\vec{x}$. The variable $\vec{x}$ is given in the fixed coordinate frame, yet $\vec{r}$ is given in the powertrain local frame. It is worth pointing out that as the powertrain is modeled as a rigid body, $\vec{r}$ is constant all the time. In order to unify the coordinate frame, $\vec{r}$ is transformed to the fixed frame by multiplying by the transformation matrix R, where the transformation matrix is a function of the Euler angles E of the powertrain. Correspondingly, the transpose of R will transform from global to local coordinate frame. The whole process is shown in Eq. (5.1).

$$\vec{x}_e = R(E)\,\vec{r} + \vec{x} \tag{5.1}$$

In this model, the bushing is treated as a spring-damper with linear stiffness and linear damping, and the stiffness and damping coefficients are based on test results. Note that this model can easily cope with nonlinear stiffness and damping by converting the time-history data into data arrays. Furthermore, the model can be modified to incorporate more complex bushing models, e.g., Ok, Yoo and Sohn[27]. The load on the bushing depends on its deflection as well as the rate of deflection. The force equation for each bushing is given in Eq. (5.2).

$$\vec{F}_B = KR'(\vec{x}_f - \vec{x}_e) + C\left(R'\vec{v}_f - \vec{v}_e\right) \tag{5.2}$$

The stiffness K is constant for a linear bushing, or a set of vectors representing nonlinear stiffness curves. The stiffness is defined for each translational direction of the bushing reference frame, which in this model, is attached to the powertrain and moves along with it. It is the same for the damping coefficient C. The frame-side bushing locations $\vec{x}_f$ and the corresponding velocity $\vec{v}_f$ are given in the global frame; each is a three by one vector. The velocity $\vec{v}_f$ is derived from displacement by using second order three point differentiation. The powertrain-side bushing location velocity in the powertrain local frame is $\vec{v}_e$, and can be computed from the relative motion kinematics, as shown in Eq. (5.3).

$$\vec{v}_e = \vec{v} + \vec{\omega} \times \vec{r} \tag{5.3}$$

where:

$\vec{v}$ = powertrain translational velocity at CM

$\vec{\omega}$ = powertrain angular velocity of CM

The powertrain is treated as a single rigid body of mass m and with inertia I_G. The linear velocity

and acceleration of the powertrain CM are $\vec{v}$ and $\vec{a}$, and the angular velocity and acceleration are $\vec{\omega}$ and $\vec{\alpha}$ respectively. The Newton-Euler equations of motion, given in Eq. (5.4, 5.5) are used to solve for the powertrain accelerations.

$$\vec{F} = m\vec{a} + \vec{\omega} \times m\vec{v} \tag{5.4}$$

$$\vec{M}_G = I_G\vec{\alpha} + \vec{\omega} \times I_G\vec{\omega} \tag{5.5}$$

The total force $\vec{F}$ and moment $\vec{M}_G$ acting on CM can also be expressed as

$$\vec{F} = \sum \vec{F}_B - R'mg \tag{5.6}$$

$$\vec{M}_G = \sum \left(\vec{r} \times \vec{F}_B\right) + \vec{M}_H \tag{5.7}$$

The powertrain gravity is originally acting in the global z direction. As the powertrain forces are acting in the local frame, the gravity is transformed from the global to the powertrain local frame to calculate the sum in Eq. (5.6). The front half-shaft torques $\vec{M}_H$ acting on the engine CM are also considered. The left and right torques are equal in magnitude and direction. The resulting powertrain accelerations from the solution of the equations of motion are integrated to determine the linear and angular velocities. The linear velocity is transformed to global coordinates and integrated to find the global location, as shown in Eq. (5.8), while the angular velocity is used to solve the kinematic differential equations to find the Euler angles of the powertrain, as shown in Eq. (3.16) and from these, its transformation matrix.

$$\dot{\vec{x}} = R(E)\vec{v} \tag{5.8}$$

The equations from Eq. (5.1) to Eq. (5.8) together with Eq. (3.16) can be rearranged in the form of first order differential equations as:

$$\dot{\vec{y}} = f(y) \tag{5.9}$$

The set of equations are solved using a fixed step second order ordinary differential equation solver (ODE2). The vector $\vec{y}$ is expressed as

$$\vec{y} = \begin{bmatrix} \vec{E} \\ \vec{x} \\ \vec{\omega} \\ \vec{v} \end{bmatrix} \tag{5.10}$$

Each entry is a 3×1 vector, so $\vec{y}$ is a 12×1 vector. The initial conditions of the variables are all zeros

except for $\vec{x}$, which are the frame-side bushing locations in the fixed frame. It is assumed that for the first time step, the corresponding frame-side and powertrain-side locations of each bushing overlap each other.

5.1.1 Limitation

First of all, the powertrain is modeled as a rigid mass, and the bushings are acting on the powertrain body directly. However, for some of the real powertrains, the mechanisms are complex. For example, the PF 2.4 powertain has an extra revolute joint that connects the rear bushing.

In terms of bushing properties, it is assumed that the bushing coordinate frame aligns with the powertrain coordinate frame all the time. This assumption is valid for the bushing whose stiffness and damping are relatively large while deformation is small. If the bushing deformation is too large, the accuracy of the simulation result will be affected. In addition, the bushing is assumed to have no rotational deformation, so the force is contributed by the translational deflections only. This assumption is not only valid for this PF 2.4 powertrain, but for most of the vehicle powertrain layouts as well. Due to the geometry of the powertrain mounts and their locations, the rotational motion of the mounts is very small, thus the corresponding torques are small and can be neglected.

Although the MATLAB® dynamic shaking model has its limitations, it is still important to simulate and validate it. Because it is the theoretical foundation of dynamic shaking, and the simulation results are helpful for understanding the dynamic behavior of the powertrain and bushings. Moreover, this model can be a very useful tool in simulating the powertrain with a simple mechanism. It has good management of batch simulations, and good connection with the drive file development too, as both functions are coded in MATLAB®. In addition, the MATLAB® simulation has the advantage of applying many simulation methods, such as the genetic algorithm and parallel computing, as they are built in tools in MATLAB®.

5.1.2 Software Application

In the same way as the drive file generation process, the MATLAB® dynamic shaking model is compiled. The parameters need to be manually imported by the user, which are listed in Table 5.1.

Once the parameters are set up, the user could start the simulation and a progress bar will pop up indicating the remaining time for the simulation. Depending on the length of the drive file, as well as the complexity of the model, the simulation time could vary from 15 to 30 minutes. For instance, the most time efficient simulation is to use linear bushing parameters, while the ABM may require longer calculation time. The outputs of the MATLAB® dynamic shaking model are listed in Table 5.2.

Table 5.1: MATLAB® dynamic shaking inputs.

Item Name	Unit	Note
Powertrain-side mount locations	mm	Input coordinate $[x,y,z]$
Powertrain center of mass	mm	Input coordinate $[x,y,z]$
Powertrain mass	kg	
Bushing stiffness	N/mm	Three axial directions
Bushing damping	N.mm/s	Three axial directions
Half-shaft torque		Names of test left and right half shaft torques for PF 2.4 FWD
Drive file		Name of the drive file

Table 5.2: MATLAB® dynamic shaking outputs.

Item Name	Unit	Note
Powertrain Euler angles	deg	Roll, pitch and yaw
Powertrain-side accelerations	mm/s^2	Measured on three accelerometer locations
Powertrain CM translational accelerations	mm/s^2	x, y, z
Powertrain CM rotational accelerations	deg/s^2	rx, ry, rz
Bushing deflections	mm	Translational deflections of left front, right front and rear bushings
Bushing loads	N	Translational loads of left front, right front and rear bushings

5.2 MotionView Shaking Model

Due to the limitations of the MATLAB® dynamic shaking model, an advanced model is built using MotionView®. The MotionView® model has the similar modeling idea, while the powertrain and bushings are more complex. The distinct advantage of using MotionView® is that it handles complex mechanisms easily. Besides, the bushing is a default part in MotionView®, thus the rotational terms as well as coordinate frame can be easily defined by the user. The user friendly GUI makes the modification to the model more convenient in this software. In most of the cases, the accuracy of the simulation results is improved by adopting a more sophisticated and realistic model. The completed powertrain and shaker model is shown in Figure 5.2.

The model consists of seven elements, and they are: point, body, joint, bushing, motion, force and sensor. The details of each element are described in the following.

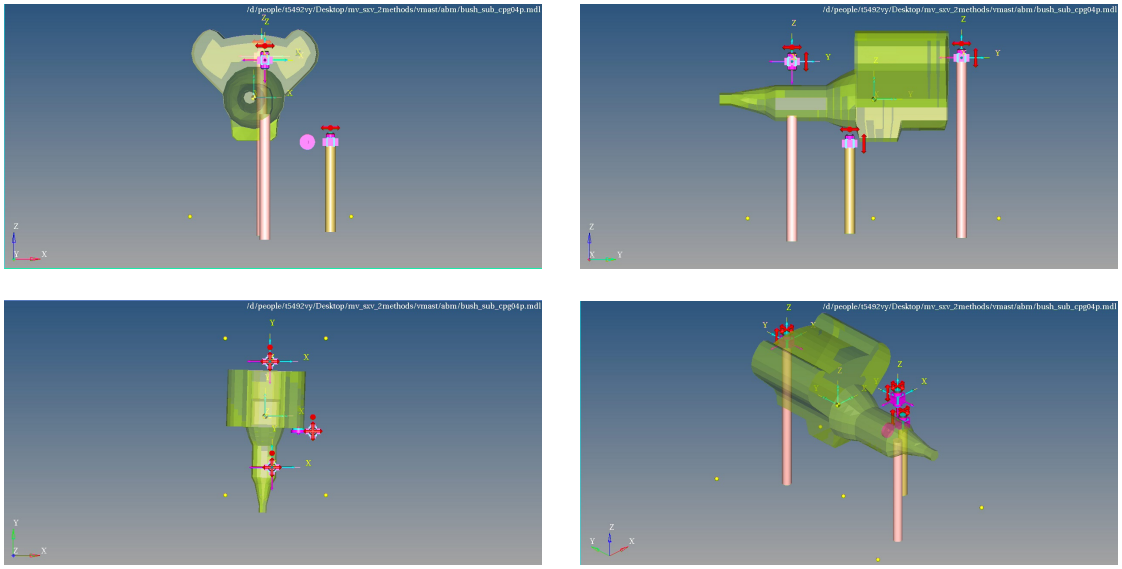


Figure 5.2: Side views and isometric view of the MotionView® model.

Point

In this model, except for the bushing locations and powertrain CM that are mentioned in the previous pages, the powertrain accelerometer locations are included. The purpose is to collect powertrain-side simulated accelerations for validation. Note that for both MATLAB® and MotionView® model, the powertrain-side accelerometers are at the same locations of the bushings. The locations are listed in Table 5.3.

Table 5.3: Powertrain-side accelerometer locations.

Accelerometer	Location [mm]
Accelerometer A	[-182,-453,370]
Accelerometer B	[-200,492,391]
Accelerometer C	[184,-131,-86]

Body

The powertrain rigid body is defined by its CM location. It is not necessary to specify the geometry and material of the body; thus, the mass and inertia are manually imported by the user. For this PF 2.4 FWD, the mass is 244kg and the inertia values are specified in Table 5.4. Three rigid bodies are defined for each chassis-side bushing location, thus a total of nine bodies for the three locations. These so called 'sliders' are to carry out the chassis-side motion in order to drive the powertrain. Theoretically, these sliders have zero mass and inertia.

Table 5.4: Powertrain inertia.

Inertia	Value [kg.mm^2]
I_{xx}	19.062E6
I_{yy}	9.324E6
I_{zz}	16.234E6
I_{xy}	-1.510E6
I_{xz}	0.272E6
I_{yz}	2.417E6

Joint

A joint is used to define the relative motion between two bodies, or to constrain the degrees of freedom of a body. In this model, the sliders are constrained by translational joints allowing only translational motion. At each bushing location, the translational joints connect the sliders in the following way: first, the joint allowing only global x direction motion connects slider 1 to the ground; then, another joint allowing only global y motion connects slider 2 and slider 1; finally, slider 3 is connected to slider 2 and moves relative to slider 2 in the global z direction only. Additionally, a revolute joint is added to represent a rotational degree of freedom around the global y axis between the powertrain body and rear bushing.

Bushing

The bushings are defined at their specified locations. In terms of connectivity, each bushing is connected to their corresponding chassis-side slider 3. For the powertrain-side, the front two bushings are connected to the powertrain directly, while the rear bushing is connected to the powertrain through the revolute joint described above. The nonlinear time-history translational stiffnesses are tested and imported into the model while the tested translational damping coefficients are linear. Besides, the linear rotational stiffnesses are applied. Each bushing has a 'marker' attached to it that establishes the local coordinate frame.

Motion

The drive file developed previously is used as the motion of the sliders. Nine displacement signals are applied to the corresponding sliders, e.g., the left front bushing x direction displacement is attached to the corresponding slider 1. In this way, the motion on slider 3 is the combination of three translational motions in the global frame.

Force

The gravity force is considered for the powertrain body. The front left and right half-shaft torques are applied at the CM of the powertrain.

Sensor

Sensors are defined by the user to measure the specified time-history simulation result. In this model, the measured parameters are listed below:

- Powertrain CM displacement
- Powertrain-side accelerometer location accelerations
- Bushing deflections
- Bushing loads

Chapter 6

Dynamic Simulation Result Validation and Discussion

In this chapter, both the MATLAB® and MotionView® models are simulated. In addition, the limitations as well as assumptions are also discussed. In order to carry out a systematic validation process, first of all, a dynamic shaking model with relatively simple powertrain and bushing properties are simulated in both MATLAB® and MotionView®. Both simulation results are compared with each other and checked for physical plausibility to initially validate the shaking algorithm. To further validate the method, a complex shaking model that resembles the real powertrain is simulated using MotionView®, and this time, the simulation results are compared with the test data.

6.1 Simulating the Models

The inputs to the MATLAB® model are the drive file developed in the previous section, and two measured half shaft torques. As mentioned before, this model does not fully represent the powertrain mechanism; thus, the simulation results are not compared with the test data. In order to validate its algorithm, a MotionView® model (basic model) equipped with the same parameters and inputs is also simulated to work as a reference. In this case, the three bushings are assumed to have the same linear stiffness and damping, which are based on a reasonable estimation, given in Table 6.1.

The MotionView® model (advanced model) with the complex powertrain mechanism is also simulated and the results are compared with the test data. In this case, the nonlinear translational and linear rotational stiffnesses are used, the values are given in Table 6.2. The damping coefficients are derived based on bushing test data, given in Table 6.3.

Table 6.1: Linear bushing parameters.

Direction	Stiffness [Nmm]	Damping [Nmm/s]
x	2000	2
y	2000	2
z	2000	1.5

Table 6.2: Bushing rotational stiffnesses, MotionView® advanced model.

Bushing	rx [Nmm/deg]	ry [Nmm/deg]	rz [Nmm/deg]
Left Front	2.0E6	5.0E6	1.0E6
Right Front	2.0E6	5.0E6	1.0E6
Rear	0	0	0

Table 6.3: Bushing damping coefficients, MotionView® advanced model.

Bushing	x [Nmm/s]	y [Nmm/s]	z [Nmm/s]
Left Front	1.5	1.5	2.5
Right Front	1.5	1.5	1.5
Rear	2.5	1	1

Table 6.4: MotionView® DSTIFF Solver parameters.

Parameter	Value
integr_tol	1.0E-5
vel_tol_factor	1000
h_max	1/512
h_min	1.0E-6
h0_max	1.0E-8
max_order	5
dae_constr_tol	1.0E-6
dae_corrector_maxit	4
dae_corrector_minit	1

The MATLAB® simulation is solved using a fixed step second order ODE solver, and the time step is the same as the sampling step, which is (1/512) s. Meanwhile, DSTIFF is implemented in MotionView® to solve for the transient simulation. As introduced in the previous chapter, it is an index 3 DAE solver

that uses a varying time step. The solver parameters are chosen to maintain the minimum error in each step while resulting in a relatively fast computing time (less than 30 minutes). Theoretically, the computing time might be longer for MotionView® comparing with MATLAB® when solving the same system, as the solver might adopt smaller step size depending on the error. The values of the parameters are given in Table 6.4.

According to the MotionSolve® Reference Guide [18], 'integr_tol' represents the maximum absolute error per step the integrator is allowed in computing the displacement, velocity and differential equation states. Since the displacement and velocity have different units, they are subject to different error tolerances, thus, the velocity error tolerance factor 'vel_tol_factor' is defined to be 1000 times the integrator tolerance. The maximum step size 'h_max' is taken to be the data sampling step size in order to obtain maximum accuracy, while the minimum step size 'h_min' is 1.0E-6. The maximum initial step size 'h0_max' is '1.0E-8'. It is much smaller than the maximum step size in order to prevent instability at the beginning of the simulation. The integrator takes the maximum order ('max_order') of five.

The DAEs of this model are of index three, thus, the position constraints are considered. The tolerance on the algebraic constraint equations ('dae_constr_tol') that the corrector must satisfy at convergence is '1.0E-6'. The maximum and minimum number of iterations ('dae_corrector_maxit' and 'dae_corrector_minit') that the corrector is allowed to take are four and one repectively. The former constrains the maximum number of iterations to achieve convergence, while the latter defines the minimum number of iterations before the corrector divergence is checked.

6.2 Validation

In order to validate the dynamic shaking algorithm, as well as to test the consistency of the models' performance, all fifteen road test events mentioned in the drive file development process are simulated on the models. As an example, the simulation results of GPG010 are discussed in detail in the following sections.

6.2.1 MATLAB® Simulation Result Validation

The MATLAB® time-history simulation results include the powertrain orientation, translational accelerations of the powertrain-side accelerometer, both translational and rotational accelerations of the powertrain CM, bushing deflections, and loads. As a reference, the outputs are also generated from the MotionView® basic model.

Instead of looking at the powertrain mount load directly, a number of outputs are also checked. The validation starts by looking at the orientations of the powertrain. In reality, the powertrain's rotation in response to road input should be relatively small. If the orientations are too large, the model fails to represent the real situation. In this road test event, the orientations of the powertrain are expected

to fluctuate violently as the test ground is a rough terrain, but the range is almost always less than 10 degrees roll and pitch. To illustrate the comparison, the roll angles of the two models are shown in Figure 6.1. It can be seen that the roll angles predicted by both models are almost identical. The time-history captured is when the powertrain is experiencing a consistent vibration. The magnified plot shows the magnitude of the roll is around 1 degree. Overall, the roll motion does not exceed a magnitude of 2.5 degrees. The time-history error can be quantified by subtracting both signals, and the result shows that, for the full time length, the error is within 0.5% of the MotionView® simulation result. The figures demonstrating the comparisons of the other two directions omitted, as the results are both good.

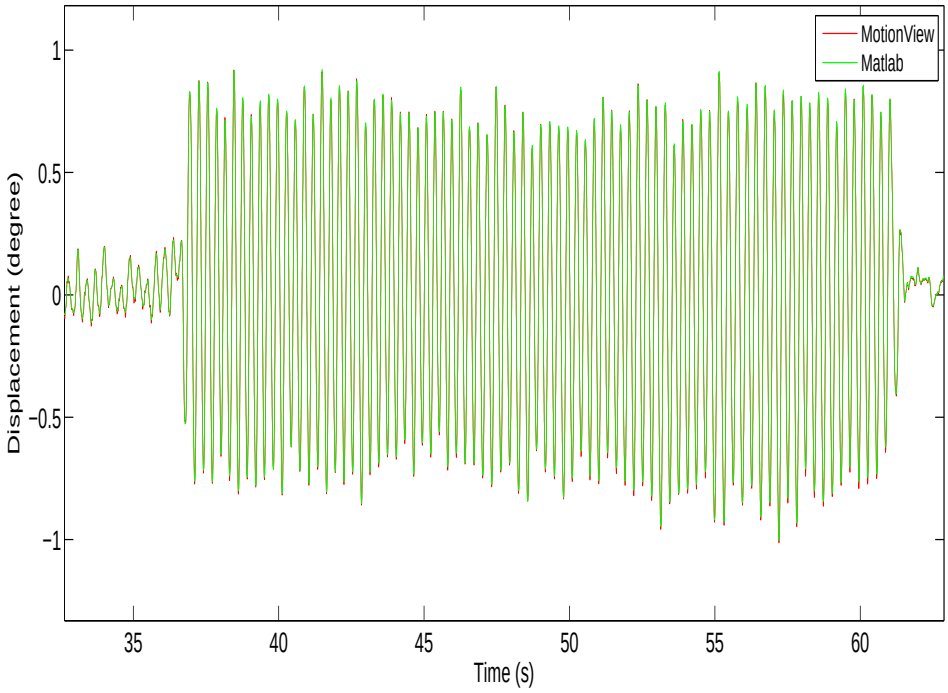


Figure 6.1: Roll angles of the powertrain, MATLAB® and MotionView®.

Also, the orientation of the powertrain can be compared with those of the chassis to check their plausibility. Although the orientations are expected to be different, the time-history plots of each angle should be similar and the difference should be small, e.g., see Figure 6.2. The pitch angle is selected to demonstrate the difference, as for a transverse powertrain, the engine's own vibration adds to the pitch motion. This phenomenon is seen on the plot; while the powertrain pitch is similar to that of the chassis, it has additional small vibrations superimposed. This is the powertrain vibration excited by the half-shaft input.

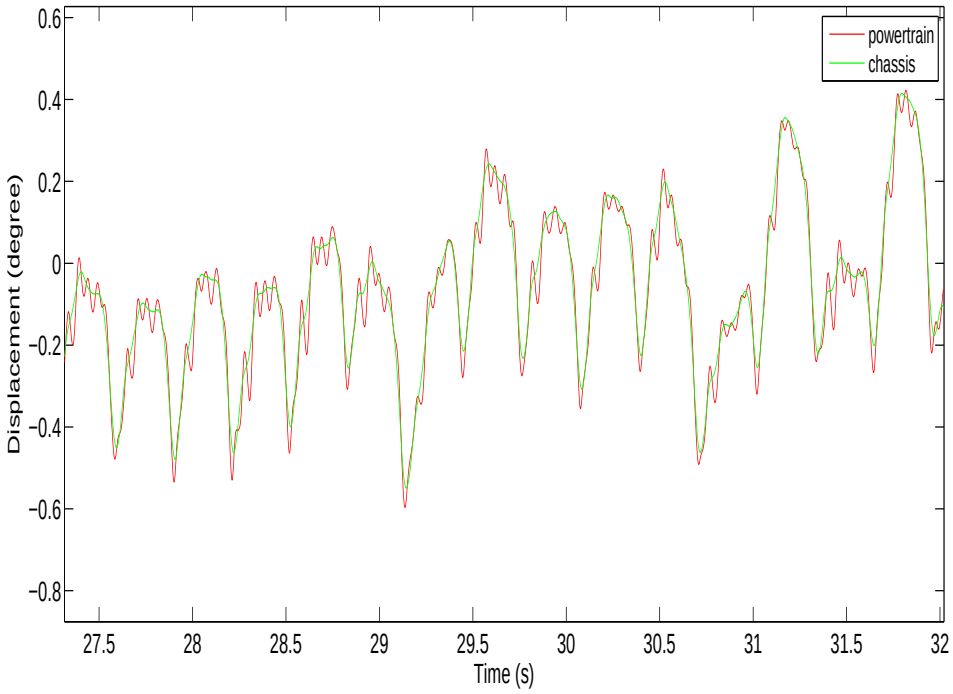


Figure 6.2: Pitch angles of the powertrain and chassis from MATLAB®.

In terms of accelerations, all six powertrain CM accelerations match well between the two models, so here the CM translational x acceleration, as well as the magnified rotational x acceleration are shown in Figure 6.3 and 6.4.

Also, the accelerations of the powertrain-side accelerometers are compared. This together with the CM accelerations are used to check the effectiveness of the Newton-Euler equations and vector operation used in the MATLAB® model. Figure 6.5 illustrates the comparison of the three translational accelerations on accelerometer A (left front) location. The time-history plots are magnified for a clearer view. Overall, the correlation is almost line to line, and the time-history error of each plot is less than 1.5% of the MotionView® simulation result.

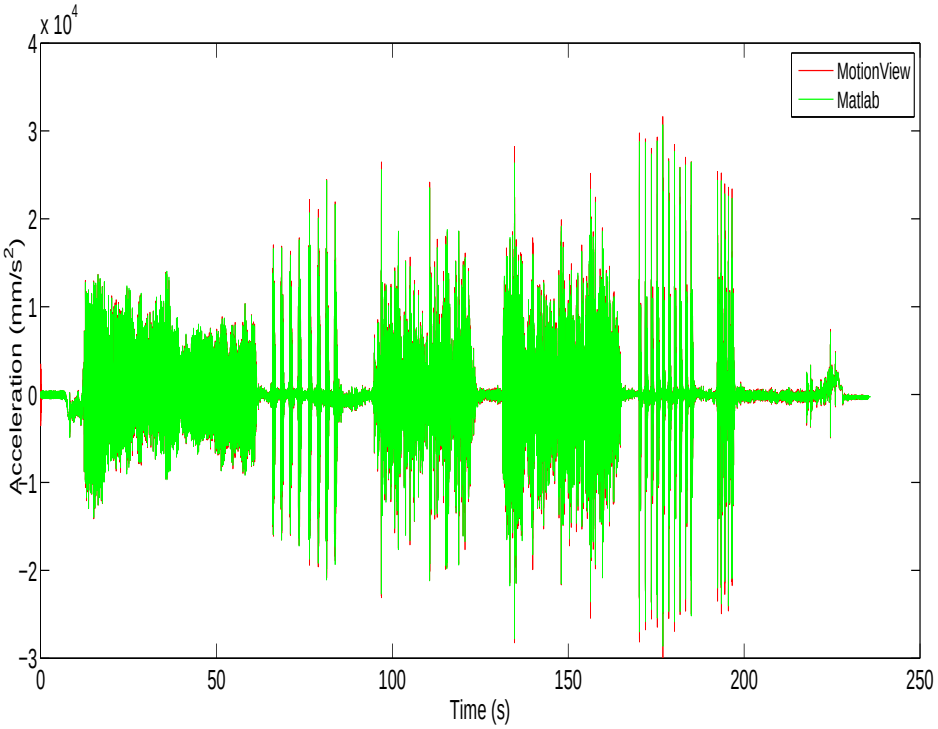


Figure 6.3: Powertrain CM translational x acceleration, MATLAB® and MotionView®.

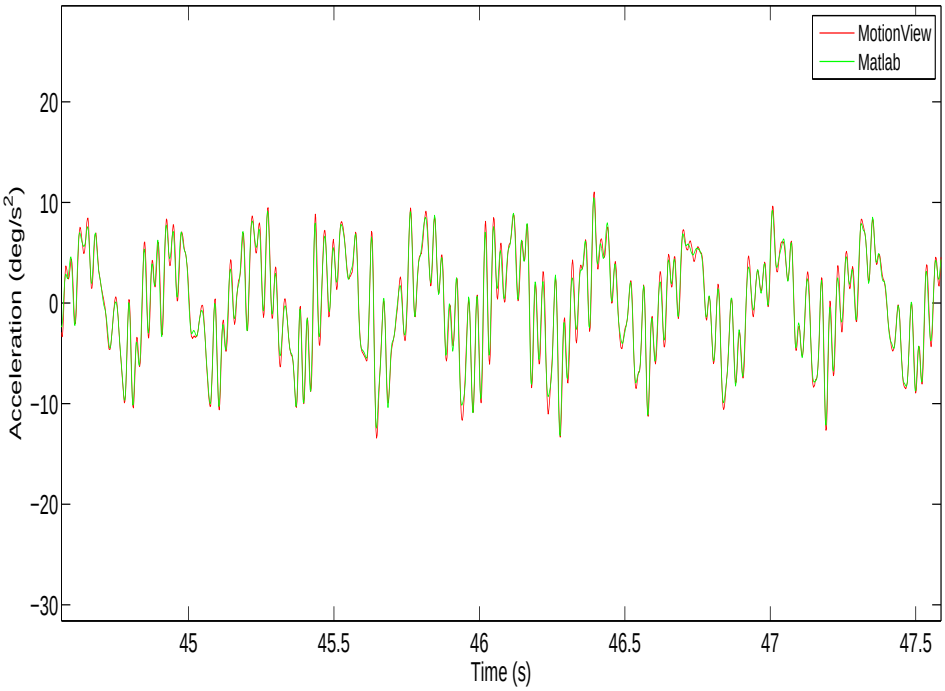


Figure 6.4: Powertrain CM rotational x acceleration, MATLAB® and MotionView®.

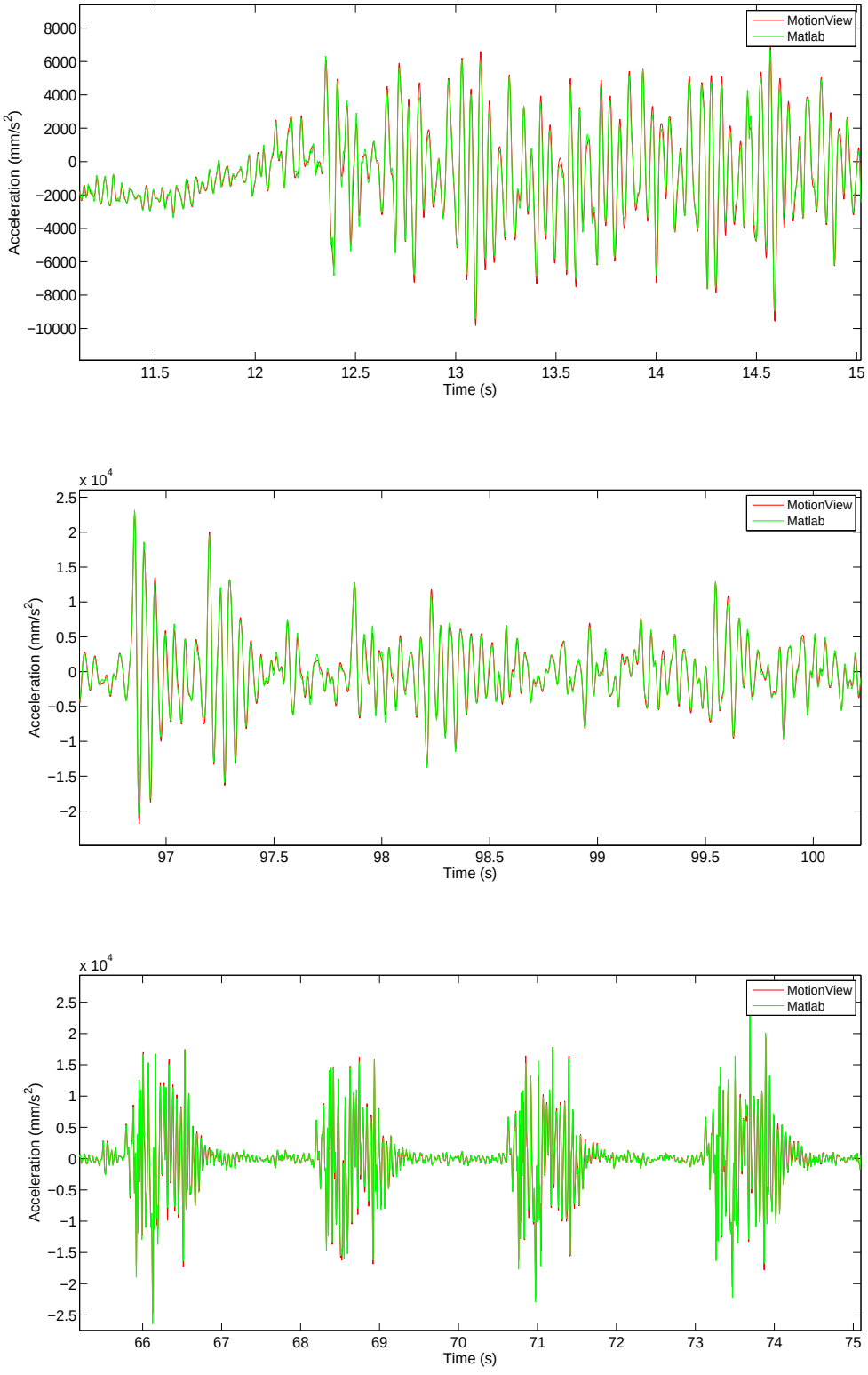


Figure 6.5: Translational x, y and z accelerations on accelerometer A, MATLAB® and MotionView®.

In order to validate the equations for bushing deflection and force, the z axial deflections of the left front and right front bushings are compared and shown in Figure 6.6 and 6.7. Again, both the full and partial time-history plots demonstrate good correlations between the two models. For this PF 2.4, the bushing z direction has more allowance than the other two. However, it is still very small, typically less than 10 mm for the engine mounts. The overall deflections of the bushings are within 2 mm, which reflects reality.

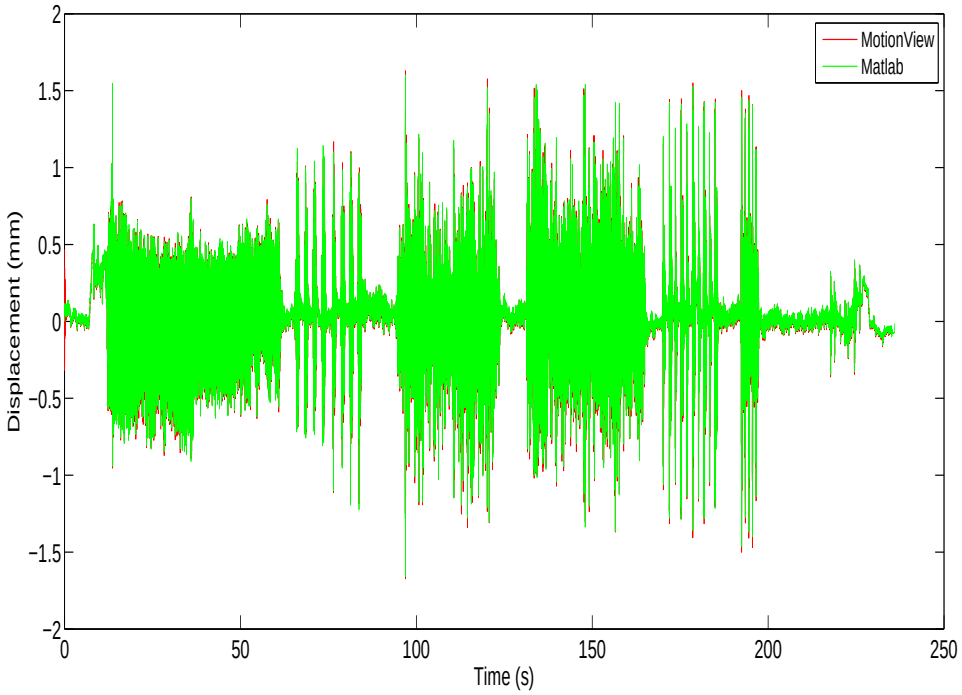


Figure 6.6: z axis deflection of left front bushing, MATLAB® and MotionView®.

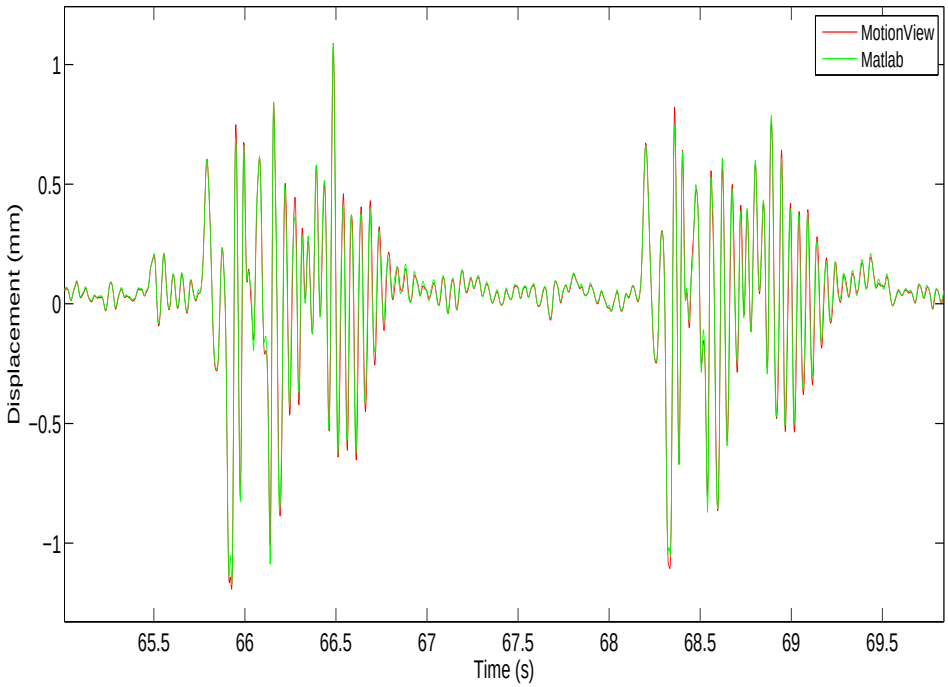


Figure 6.7: z axis deflection of right front bushing, MATLAB® and MotionView®.

Lastly, the z axial loads on the left front and right front bushings are derived and compared in Figure 6.8, 6.9. With the matching deflections shown above, the forces are expected to correlate well, which is the case.

Numerically, the maximum and minimum values, as well as the root mean square numbers are calculated compared for corresponding simulation results of the two models. The comparison shows that the errors are usually within three percent of the data.

Since the simulation results from both models agree with each other, and are physically sound, it can be concluded that the physical theories and equations are used correctly. However, it doesn't mean that the dynamic shaking model could predict the real case as there is no test data to compare with; thus, it is called a 'preliminary validation'. To fulfill the validation process, the simulation results are compared with test data, which is discussed in the following section.

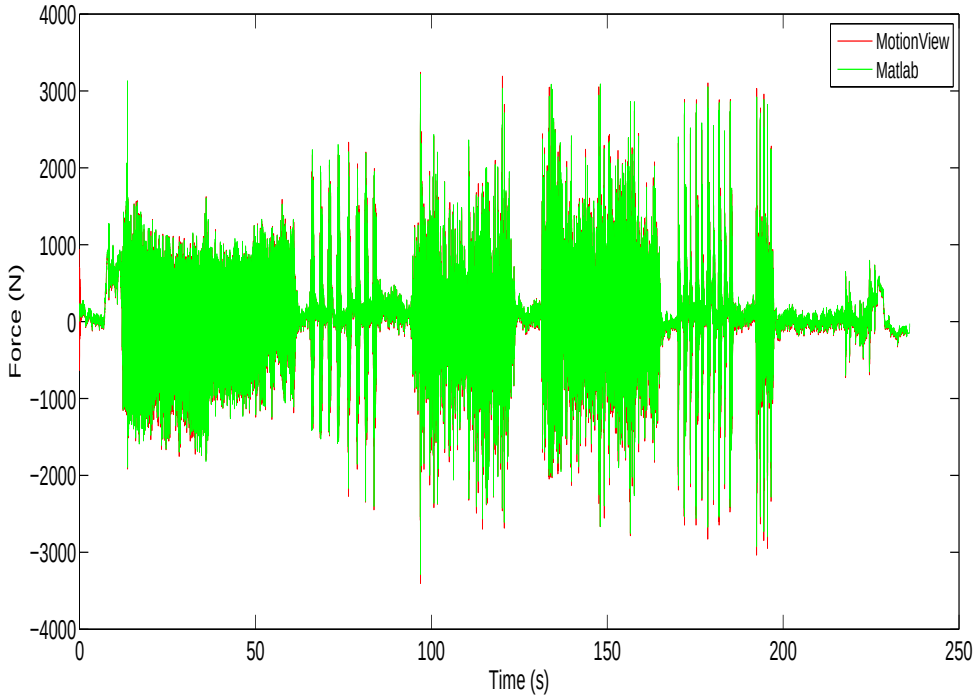


Figure 6.8: z axis load of left front bushing, MATLAB® and MotionView®.

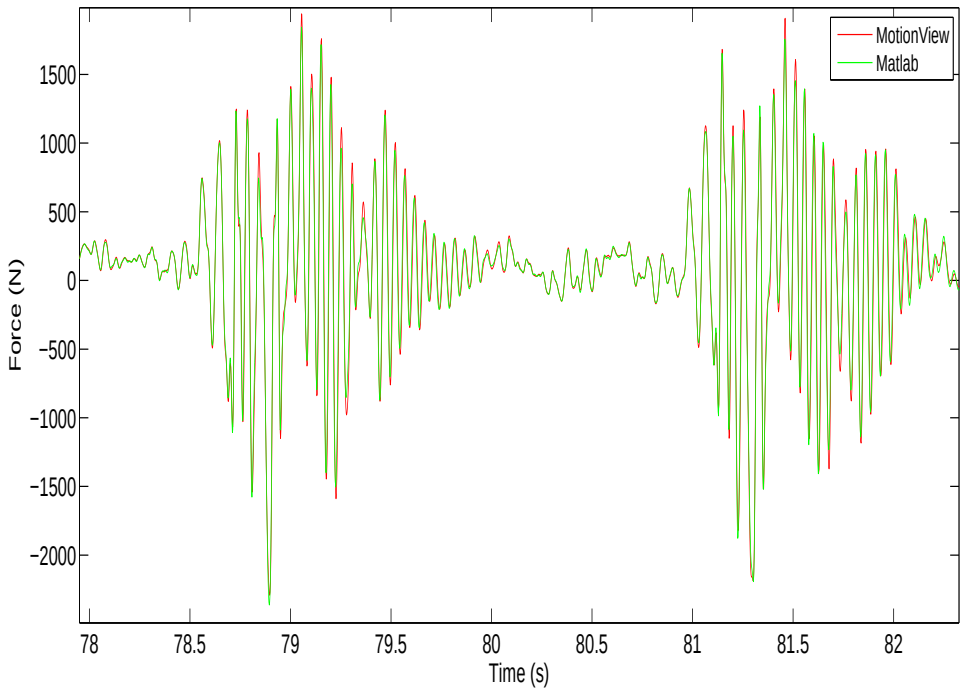
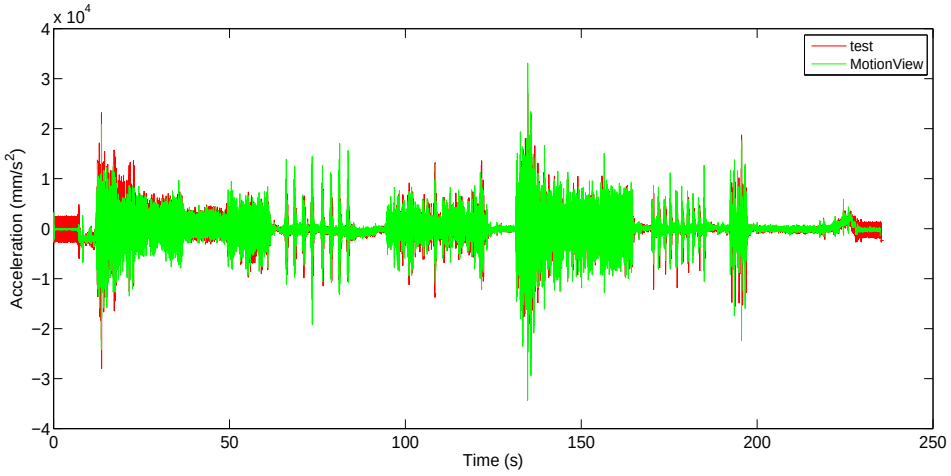


Figure 6.9: z axis load of right front bushing, MATLAB® and MotionView®.

6.2.2 MotionView® Simulation Result Validation

The same outputs covered in the last section are also derived from the MotionView® advanced model simulation. The simulation results such as the powertrain CM orientations and bushing deflections are again checked, and the values are all physically reasonable. This time, the focus is on the comparison of the powertrain-side accelerometer accelerations, as well as the bushing loads, as the data is also collected from road test. Again, as the company's standard road test, the simulation results of CPG010 are demonstrated.

As the powertrain is assumed to be rigid, the acceleration correlation of one location is sufficient to present the overall quality of the simulated powertrain-side accelerations. Therefore, the comparison is carried out on the front left accelerometer location, shown in Figure 6.10. The first plot demonstrates the full time-history of the x axial acceleration, and the simulation result (green) matches both the low frequency content and high frequency reversals of the test signal (red). Magnitude-wise, the test acceleration is especially larger in the first and last ten seconds of the simulation. The second and third plot are the magnified y and z translational accelerations respectively. Again, the vibrations especially each peak, is captured by the simulation. This proves that not only does the dynamic shaking model respond to the drive motion in a realistic way, but also the drive file reproduces the details of the original data. However, the mismatch is still shown.



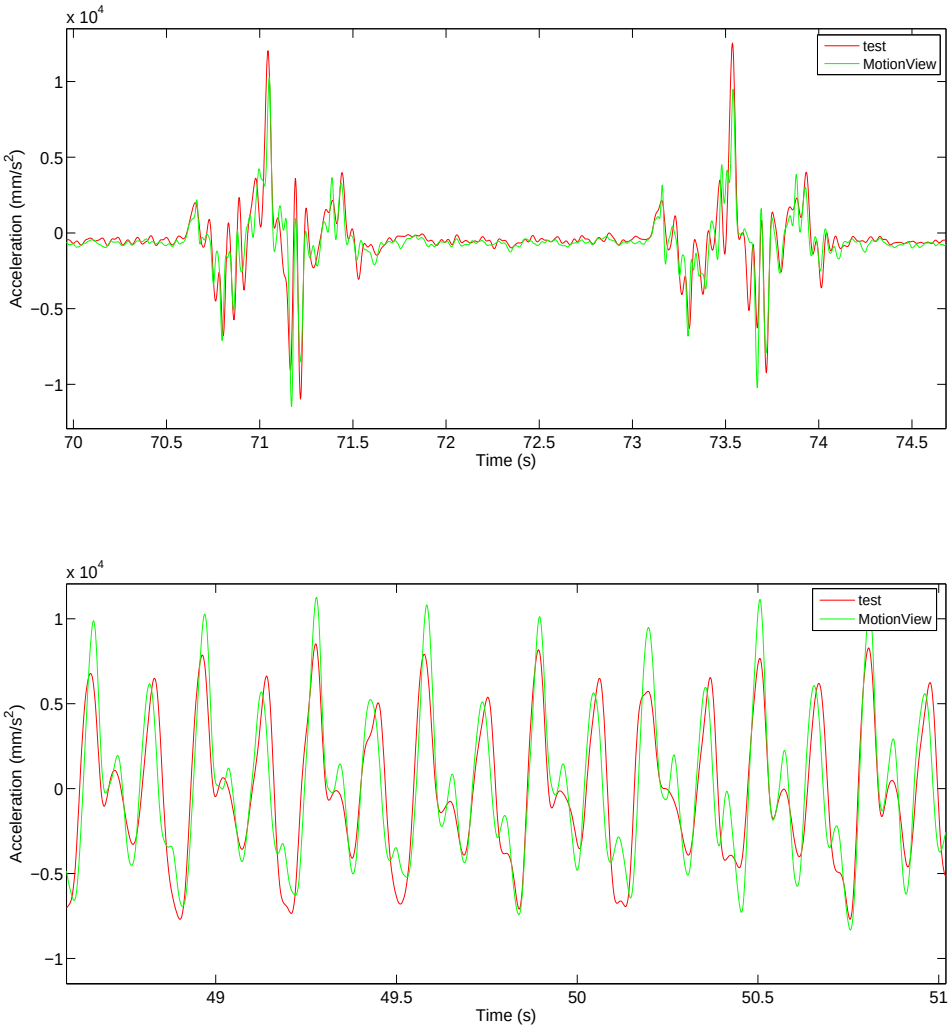


Figure 6.10: Powertrain-side accelerations, MotionView® and test.

The comparison of the simulation and test bushing forces are displayed in Figure 6.11 to 6.13. The translational z forces of the front left and right bushing, as well as the translational x force of the rear bushing are chosen, as they are the primary parameters considered in the design process. As the inertia load is closely related to acceleration, the simulated bushing force is not expected to have line to line correlation with the test data, and the plots reveal this fact.

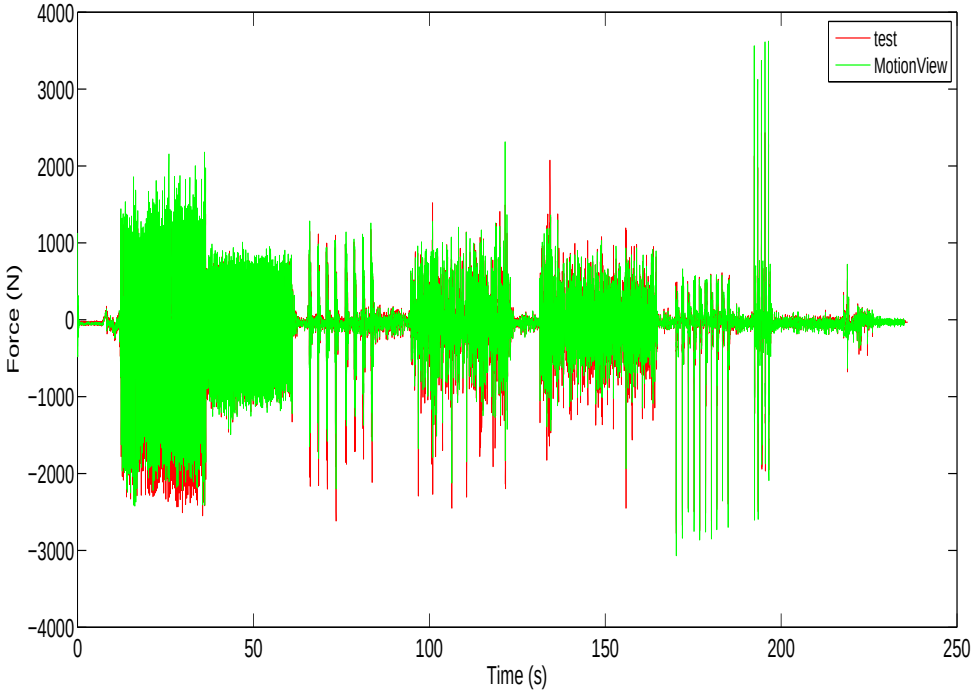


Figure 6.11: Left front bushing z axial force, MotionView® and test.

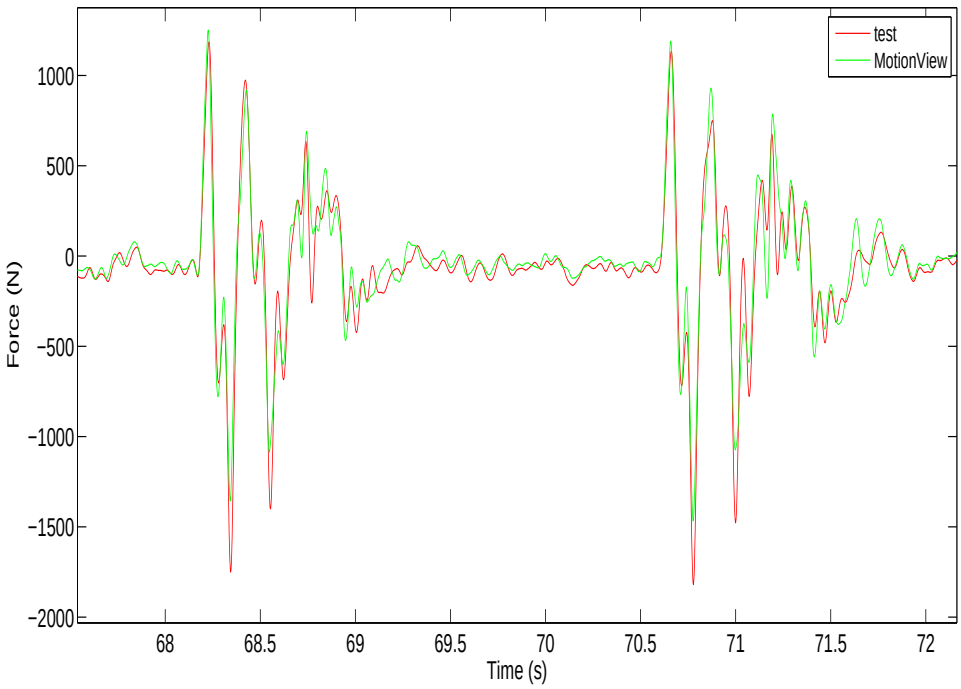


Figure 6.12: Right front bushing z axial force, MotionView® and test.

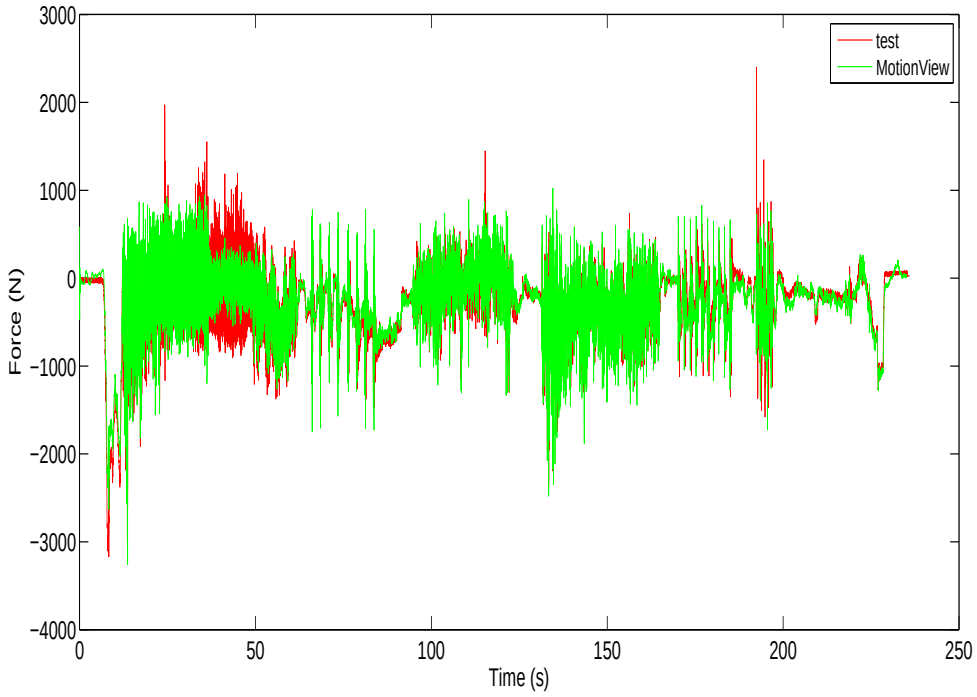


Figure 6.13: Rear bushing x axial force, MotionView® and test.

Again, the simulation result could follow the fluctuation of the test signal, but the magnitude is sometimes off. By just looking at the plots, there is no way to determine how good the correlation is with test data. So the simulation and test data are quantified by applying the rainflow-counting algorithm; see Matsuishi and Endo[28]. This method is able to calculate the fatigue data of a spectrum of varying stress. For one time-history of bushing force, the fatigue data is represented as a dimensionless number. Overall, for the fifteen road test events, the fatigue numbers of bushing forces from both simulation and test data are plotted using bar charts, where six of them are shown in Figure 6.14, and the remaining events are displayed in Appendix A. The x axis label 'LZ' represents the z axial force of left bushing; 'RZ' represents the z axial force of right bushing and 'RRX' stands for the x axial force of the rear bushing. The fatigue numbers are also listed in Table 6.5.

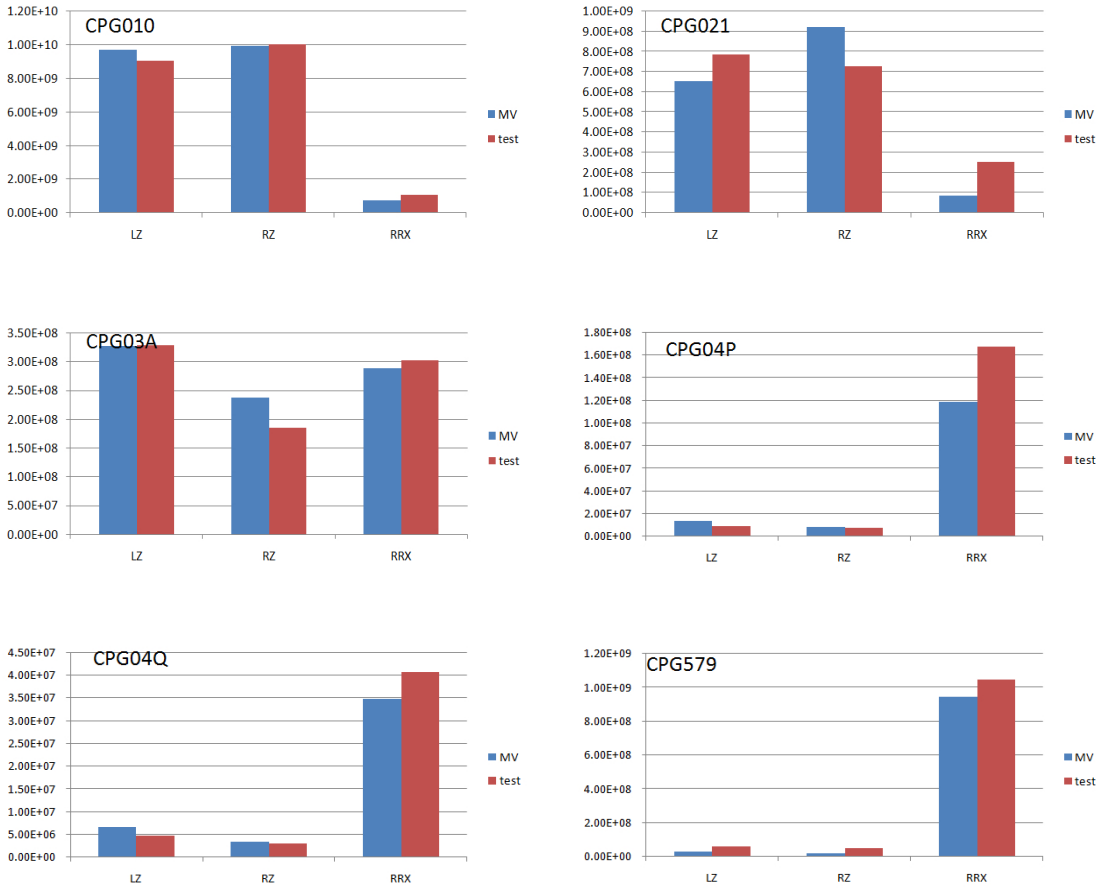


Figure 6.14: Bushing forces fatigue data comparison for six events.

Table 6.5: Fatigue numbers of the six road test events, MotionView® and test.

	Left z	Right z	Rear x
CPG010 MV	9.72E9	9.95E9	7.58E8
CPG010 test	9.04E9	1.00E10	1.09E9
CPG021 MV	6.54E8	9.24E8	8.41E7
CPG021 test	7.87E8	7.29E8	2.54E8
CPG03A MV	3.26E8	2.38E8	2.88E8
CPG03A test	3.28E8	1.84E8	3.02E8
CPG04P MV	1.39E7	8.13E6	1.18E8
CPG04P test	9.08E6	7.77E6	1.67E8
CPG04Q MV	6.41E6	3.28E6	3.48E7
CPG04Q test	4.56E6	2.84E6	4.06E7
CPG579 MV	3.04E7	2.03E7	9.44E8
CPG579 test	6.27E7	4.91E7	1.04E9

It can be observed that the errors between simulated and test damages are within 50% of test data for the majority of the events simulated. It is worth noting that this damage calculating method is very sensitive to the mismatching of data in the peaks and valleys. One large mismatching peak in the simulation data could result in significant growth of the error, e.g., it was witnessed in the simulation that three times difference of one peak resulted in ten times difference of the fatigue number. In addition, the consistency of the model's dynamic response can be seen by looking at the tests with similar conditions. For instance, the damage from CPG04P and CPG04Q have the same trend, that the simulated damages are slightly higher than those of the test for the front two bushings, but lower for the rear one.

As a conclusion, the overall correlation of the time-history simulation results against test data is good. Despite the existence of mismatches in the peaks and valleys, the simulation results capture each fluctuation, at both high and low frequencies. The rainflow-counting algorithm is used as an auxiliary method to judge the quality of the reproduced bushing forces. Although numerically the errors are large for some road test events, considering the characteristic of this method, the results are acceptable. At this stage, it can be concluded that the dynamic shaking algorithm is fully validated. However, the causes of the mismatches are further studied and discussed in the following section.

6.3 Discussion

6.3.1 Bushing Modeling[†]

Regardless of the fidelity of the reproduced frame-side accelerations, the resultant accelerations and forces of the dynamic simulations is highly dependent on the properties of the bushing used to couple the chassis and the powertrain. As a result, matching the simulation results to recorded data will pose a challenge. Bushing models with only nonlinear stiffness are not sufficiently accurate in predicting the real case. Inaccuracies in the simulation results are not only due to the nonlinearity of the force vs. deflection relationship, but also the level of damping in the model. Relatively small modification of the damping values can result in dramatic difference in the simulation results. The PF 2.4 powertrain uses hydro mounts (see Appendix B for details) which have nonlinear damping. Additionally, there is usually a time lag between the bushing input force and output displacement in the dynamic simulation, which is called the hysteresis effect. So the simulation results will also be affected if this phenomenon is not captured. Besides the stiffness and damping, the geometry of the bushing should also be specified in order to further improve the simulation accuracy.

One possible solution is to use the advanced bushing model (ABM), see Li[29]. The ABM is a virtual bushing model which adopts nonlinear stiffness and damping properties. It also captures the hysteresis effect. The ABM bushing parameters are fitted from bushing test data, and the ABM is an improvement in reproducing the dynamics of the real bushing. However, for this PF 2.4 powertrain, only the front two

[†]Chapter 6.3.1 is the outcome of joint research.

bushings' z direction are fitted. To test the performance of the dynamic model with the ABM, the front bushings translational z properties in the MotionView® advanced model are switched to use the ABM parameters, and the events CPG010, CPG03A and CPG04P are simulated.

First of all, the comparison between CPG03A and test left front bushing translational z forces is shown in Figure 6.15. By just looking at the partial time-history plot, it is hard to judge whether there is improvement or not, because the mismatch can still be seen at each peak and valley. The same situation exists for the other two events; thus, the fatigue data is referred, see Table 6.6.

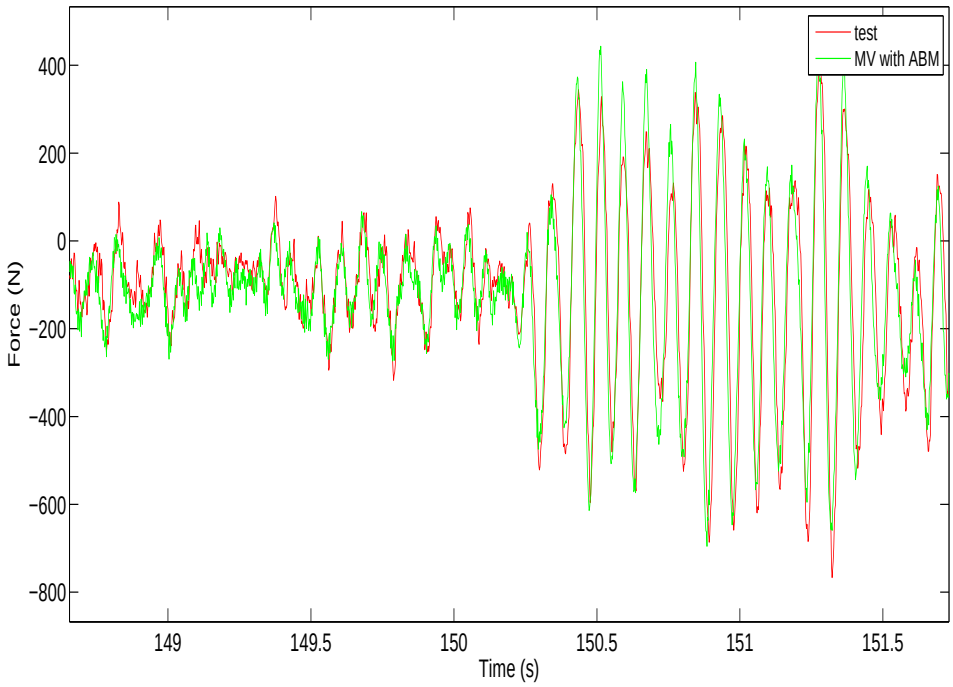


Figure 6.15: Left front z axial bushing force, ABM and test.

Because the rest of the dynamic model remains the same, the fatigue with the ABM can be compared with those of the original model for possible improvements. Out of the three events, CPG03A is seen to have improvement on both sides with the application of ABM. The left z in CPG04P is improved, but the right z is worse. In terms of CPG010, both sides have worse correlation with the test data. So the simulated bushing force is not guaranteed to have improvement with the ABM working in only one direction. However, if all the bushings are adopted to ABM at all directions, there is a huge potential that the simulated bushing load is much improved. The reason is simple: in order to reproduce reality, the model has to be realistic.

Table 6.6: Fatigue numbers of the front bushing vertical forces with ABM.

	Left z	Right z
CPG03A no ABM	3.26E8	2.38E8
CPG03A ABM	3.49E8	2.05E8
CPG03A test	3.42E8	1.94E8
CPG04P no ABM	1.39E7	8.13E6
CPG04P ABM	1.24E7	8.53E6
CPG04P test	8.71E6	7.27E6
CPG010 no ABM	9.72E9	9.95E9
CPG010 ABM	4.92E9	4.22E9
CPG010 test	9.29E9	1.04E10

6.3.2 Powertrain Modeling

If one aims for extreme accuracy, the engine and transmission should be modeled specifically. For instance, the motion of the pistons should be included as it has dynamic effect on the bushings. However, such a model would acquire many specifications of the powetrain that are not available in this project, and the simulation time would be exponentially increased. But the powertrain internal dynamics is only partially neglected, as the half-shaft torques are recovered.

6.3.3 Data Collection and Solving

The DAC format (see Glyph Reference Guide[30] for details), has been used as a routine to record accelerations during data collection. The DAC format stores data with single precision. However, numerical errors are detected in the double differentiated DAC format driving displacements, which are the accelerations. In this case, the simulated bushing loads with ABM are affected, as the bushing mass is considered. Although the bushing mass can be turned off in order to avoid this problem, the simulation accuracy will in turn be influenced. To solve this, a file format with higher precision and still compatible with MotionView® needs to be found to replace the current DAC format.

Numerical error can also generated by the solver. Since the DAEs are stiff, the solver time step may need to be small in order to prevent instability and reduce error. However, the simulation time is in turn increased. So the current solver parameter are chosen to balance between simulation time and accuracy.

6.3.4 Design Application

Although the dynamic shaking model does not predict the bushing loads perfectly, it is still of great value. For some road test events such as CPG010 and CPG03A, the bushing time-history loads as well as the fatigue indices have been well predicted. Such simulation results are already good references in the

design and test procedure.

Furthermore, as the theoretical foundation has been built up and validated, once the more realistic powertrain and bushing models are applied, the simulation results can be further improved until approaching an ideal correlation with the test data for all events. Such a model can then be used in the design and test procedure as a simulation tool that allows easy parameter tuning and fast design iteration. For example, if the influence of adjusting the powertrain mount stiffness or damping needs to be studied, the modification of the mount parameters can be easily done with the virtual model, and a time efficient simulation allows a result to be found quickly. Or if there are new powertrain mounts to be tested, they can be fitted and imported to the shaking model, and the simulated time-history loads and fatigues can be referenced.

Chapter 7

Conclusions and Recommendations

7.1 Conclusions

The purpose of this project is to develop a set of virtual simulation tools, which could numerically reproduce the vehicle chassis motion, and use it to drive the vehicle powertrain dynamic shaking while predicting the powertrain mount loads.

To accomplish this goal, a simulation tool is first developed with the capability of converting the road test accelerations of a vehicle chassis into displacements, in order to numerically simulate the vehicle powertrain dynamic behavior under road test conditions. The produced displacements are in the global reference frame, and are called a 'drive file'. The dynamic shaking is driven by a drive file, and the loads in the powertrain mounts are predicted; their fatigue indices are calculated for design reference as well.

The purpose of developing the drive file is to effectively remove the numerical errors in the test accelerations, while maintaining physical plausibility. In this case, the rotational accelerations of the chassis CM are derived based on a rigid body assumption. Meanwhile, the integrated angular accelerations are filtered and used to find the vehicle chassis orientation (Euler angles), in order to determine the relationship between chassis local and global frame. In this way, the global location of the chassis-side mounts can be determined. The flexibility of the vehicle chassis is assumed to be integrated from the acceleration differences between those from the calculation and test. The flexible displacements are finally added to the global mount locations to form the flexibility-considered drive file. In terms of validation, except for time-history comparison, the reproduced and test accelerations are compared in root mean square. The result shows that the average error between RMS of each reproduced and test acceleration signal is less five percent for the fifteen road tests, which proves the good quality of the drive file.

The MATLAB® dynamic shaking model has relatively simple powertrain and bushing model. In order to preliminarily validate its algorithm, an identical model is built in MotionView®, and the simulation results are compared between both models. The comparison shows good correlation for all the outputs.

Besides, some outputs such as the powertrain orientations and bushing deflections are demonstrating physical plausibility. To fully validate the dynamic shaking algorithm, a complex model resembling the real powertrain is simulated with MotionView®. The bushing model adopts tested nonlinear stiffnesses and estimated linear damping coefficients. When comparing with test data, the simulated time-history results demonstrate good dynamic response of the model to the drive file by capturing the fluctuations. In other words, the dynamic model accurately responds to the excitations. However, considerable mismatches are seen for the peaks and valleys. The bushing loads are quantified using the rainflow-counting algorithm, and the resultant fatigue indices are compared between simulation and test. The comparison shows that the relative errors are within fifty percent for the majority of the events simulated. Considering the sensitivity of the rainflow-counting algorithm to large peaks and valleys in the data, the fatigue indices are acceptable.

Both MATLAB® and MotionView® dynamic shaking models are of great value. The MATLAB® shaking simulation tool is very useful in simulating the powertrain with simple mechanism. It has a good management of batch simulations, as well as good connection with the dive file development. In addition, the MATLAB® simulation has the advantage of applying many simulation methods, such as the genetic algorithm and parallel computing.

Even though the bushing loads are not precisely predicted for every road test event, the MotionView® dynamic shaking model is of great value. Since the theoretical foundation has been built up and validated, once the powertrain and bushing models are further refined, the simulation results are expected to improve consequently, and such a model can then be used to improve the design and test procedure. By virtually changing the powertrain and mount parameters, the new simulation results would be quickly derived, and this would largely accelerate the design and test process.

7.2 Recommendations

First of all, for the drive file development process, except for the possible improvements discussed in Section 4.4, it is recommended to optimize the drive file using a genetic algorithm (GA); see Micheal[31] for details. The target is to minimize the error of the reproduced accelerations. So instead of manually tunning the filter parameters, the GA will automatically find the combination which results in the minimum error. Nevertheless, the numerical range of the filter parameters need to be artificially constrained in order to prevent physically unreasonable results. However, the current limitations for applying such a method is the insufficient computing power. The characteristic of the GA could lead to days of computing time with the available hardware. So once the equipments permit, the GA will be applied.

Second, in terms of dynamic shaking, the predicted bushing loads are not seen to have consistent improvements for different road tests with the implementation of ABM. The cause is that the current ABM is only developed for the bushing vertical direction. So the future work is to develop the bushing models

with physical bushing parameters, and apply them to the dynamic shaking model. Some sophisticated phenomena, such as coupling effects of bushing translational stiffnesses, will be considered in order to further improve simulation accuracy. Furthermore, the bushing geometry needs to be specified in the software such as CATIA®. The geometry of the bushing has significant effect on the bushing motion, e.g., a typical hydro mount has a extended linkage that allows angular motion. For such bushing mechanisms, the dynamic response is not only depending on the stiffness and damping, but the geometry as well. On the other hand, the geometry could restrain the bushing motion in the extreme condition, such as contact.

Lastly, the powertrain model can be modified to improve simulation accuracy. If the powertrain's flexibility is considerable, a finite element (FE) model could be implemented in the MotionView® model, as this software is capable of conducting finite element analysis (FEA). Furthermore, the powertrain's working process could be simulated by adding the components, such as pistons and crankshaft. As these components are always of large masses, and are moving with high frequencies, their dynamic effects are significant. Therefore, in the future research, a powertrain model that could reproduce its working condition is going to be developed.

References

[1] R. V. Schwerin, *Multi-body System Simulation. Numerical Methods, Algorithms and Software*. Berlin: Springer, 1999.

[2] B. Minaker and R. Rieveley, "Automatic generation of the non-holonomic equations of motion for vehicle stability analysis," *Vehicle System Dynamics*, vol. 48, pp. 1043–1063, September 2010.

[3] P. Grote and M. Sharp, "Defining the vehicle development process," tech. rep., MTS System Cooperation, 2001.

[4] M. Becker, P. Salvatore, and F. Zirpoli, "The impact of virtual simulation tools on problem-solving and new product development organization," *ELSEVIER*, vol. 34, pp. 1305–1321, November 2005.

[5] SBEL, "Introduction to ADAMS/View," tech. rep., University of Wisconsin–Madison.

[6] D. Fothergill, "The use of MBD modeling techniques in the design and development of a suspension system," tech. rep., ARRK Technical Service.

[7] *MotionSolve 12.0 User's Guide*.

[8] D. Hanselman and B. Littlefield, *Mastering MATLAB® 7*. Prentice Hall, 2004.

[9] D. Negrut and A. Dyer, "ADAMS/Solver primer," tech. rep., MSC Software, August 2004.

[10] J. Lambert, *Numerical Methods for Ordinary Differential Systems*. New York: Wiley, 1992.

[11] C. Gear and R. Skeel, "The development of ODE methods: a symbiosis between hardware and numerical analysis," in *Proceeding HSNC '87 Proceedings of the ACM conference on History of scientific and numeric computation*, (New York, NY, USA), pp. 105–115, Association for Computing Machinery (ACM), 1987.

[12] C. Curtiss and J. Hirschfelder, "Integration of stiff equations," in *Proceeding of the National Academic of Science of the United States of America*, vol. 38, pp. 235–243, March 1952.

[13] G. Dahlquist, "A special stability problem for linear multistep methods," *BIT Numerical Mathematics*, vol. 3, pp. 27–43, 1963.

[14] J. Daniel and R. Moore, *Computation and Theory in Ordinary Differential Equations*. New York: W.H.Freeman & Co Ltd, December 1970.

[15] O. Widlund, "A note on conditionally stable linear multistep methods," *BIT Numerical Mathematics*, vol. 7, pp. 65–70, 1967.

[16] C. Gear, "The automatic generation of stiff differential equations," Tech. Rep. 221, University of Illinois Department of Computer Science, Urbana, IL, 1967.

[17] C. Gear, "The automatic integration of stiff ordinary differential equations," *Communications of the ACM*, vol. 14, pp. 176–179, March 1971.

[18] *RADIOSS, MotionSolve, and OptiStruct*.

[19] P. Brown, A. Hindmarsh, and L. Petzold, "Using Krylov methods in the solution of large-scale differential-algebraic systems," *SIAM Journal on Scientific Computing*, vol. 15, pp. 1467–1488, November 1994.

[20] D. Pyle, *Data Preparation for Data Mining*. San Fransisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[21] S. Butterworth, "On the theory if filter amplifiers," *Experimental wireless & the wireless engineer*, vol. 7, pp. 536–541, October 1930.

[22] *Glyphworks Tutorial 9.0*.

[23] K. Dressler, M. Speckert, and G. Bitsch, "Virtual durability test rigs for automotive engineering," *Vehicle System Dynamics*, vol. 47, pp. 387–401, April 2009.

[24] C. Sandu, E. Andersen, and S. Southward, "Multibody dynamics modeling and system identification of a quarter-car test rig with McPherson strut suspension," *Vehicle System Dynamics*, vol. 49, pp. 153–179, January–February 2011.

[25] D. Mántaras and P. Luque, "Virtual test rig to improve the design and optimization process of the vehicle steering and suspension systems," *Vehicle System Dynamics*, vol. 50, pp. 1563–1584, October 2012.

[26] H. Baruh, *Analytical Dynamics*. McGraw-Hill Science/Engineering/Math, October 1998.

[27] J. Ok, J. Sohn, and W. Yoo, "Development of nonlinear coupled mode bushing model based on the Bouc-Wen hysteretic mode," in *Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE*, vol. 5, (Las Vegas, NV, USA), pp. 1987–1994, The American Society of Mechanical Engineers (ASME), 2007.

[28] M. Matsuishi and T. Endo, "Fatigue of metals subjected to varying stress," *Japan Society of Mechanical Engineers*, 1968.

[29] S. Li, "Development of a bushing model for vehicle durability simulation," Master's thesis, University of Windsor, 2014.

[30] *Glyph Reference Guide*.

[31] M. Vose, *The Simple Genetic Algorithm: Foundations and Theory*. The MIT Press, August 1999.

Appendix A

Permission to Include Joint Research Results

Dear Xiaowu,

Permission is hereby granted for you include the Advanced Bushing Model (ABM), which is from the joint research undertaken in collaboration with Xiaowu Yang under the supervision of professor Dr. Bruce Minaker, in your in your Masters Thesis for the University of Windsor. Permission is for this one-time use only, and does not cover any third party copyrighted work which may appear in the material requested.

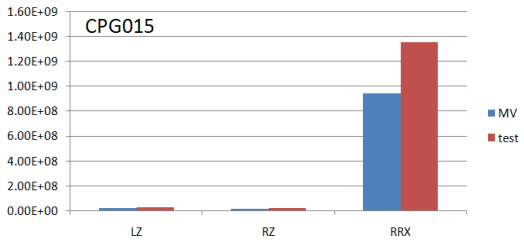
Regards,

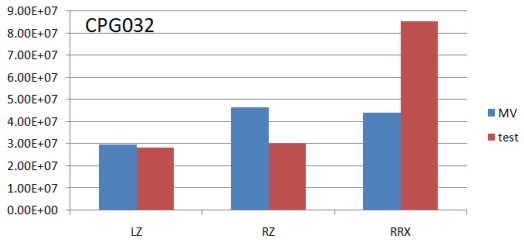
Sida Li
MASc Candidate
University of Windsor
Email: li11111o@uwindsor.ca

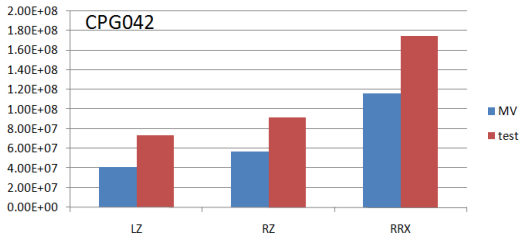
Appendix B

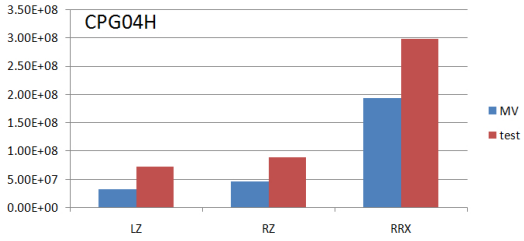
Fatigue Data Comparison

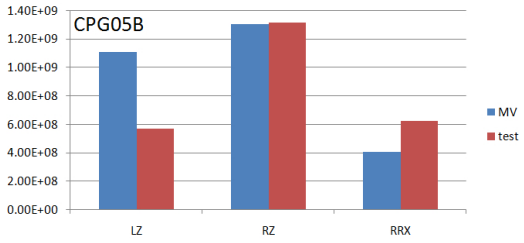
The fatigue data comparison for the rest nine events are shown in the charts below.











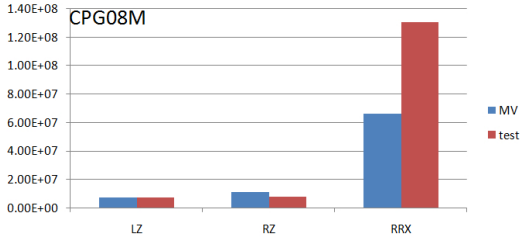


Figure B.1: Bushing forces fatigue data comparison for nine events.

Table B.1: Fatigue numbers of the nine road test events, MotionView® and test.

	Left z	Right z	Rear x
CPG015 MV	1.92E7	1.29E7	9.43E8
CPG015 test	2.38E7	1.61E7	1.35E9
CPG032 MV	2.97E7	4.64E7	4.42E7
CPG032 test	2.84E7	3.00E7	8.53E7
CPG042 MV	4.08E7	5.69E7	1.16E8
CPG042 test	7.32E7	9.11E7	1.74E8
CPG04H MV	3.14E7	4.48E7	1.93E8
CPG04H test	7.15E7	8.86E7	2.98E8
CPG05B MV	1.11E9	1.30E9	4.02E8
CPG05B test	5.64E8	1.31E9	6.22E8
CPG08M MV	7.12E6	1.10E7	6.57E7
CPG08M test	7.13E6	7.66E6	1.30E8
CPG08U MV	2.06E10	2.56E10	9.29E8
CPG08U test	3.18E10	3.38E10	3.96E9
CPG500 MV	1.72E6	9.16E5	3.38E9
CPG500 test	1.02E5	5.19E4	7.04E9
CPG574 MV	2.01E7	1.37E7	2.21E9
CPG574 test	2.57E7	2.08E7	3.37E9

Appendix C

Hydro Mount

Hydro mounts provide optimum ride comfort. They combine the acoustic isolation function of a conventional rubber mount with balanced damping performance.

The front two mounts of the PF 2.4 powertrain that ABM simulated are hydro mounts. According to Li[29], these two mounts have similar structures, both having steel frame with a hydraulic chamber at the center of the frame wrapped by synthetic rubber. Geometries of both mounts vary because of different fixtures and locations, see Figure B.1 for a typical hydro mount.

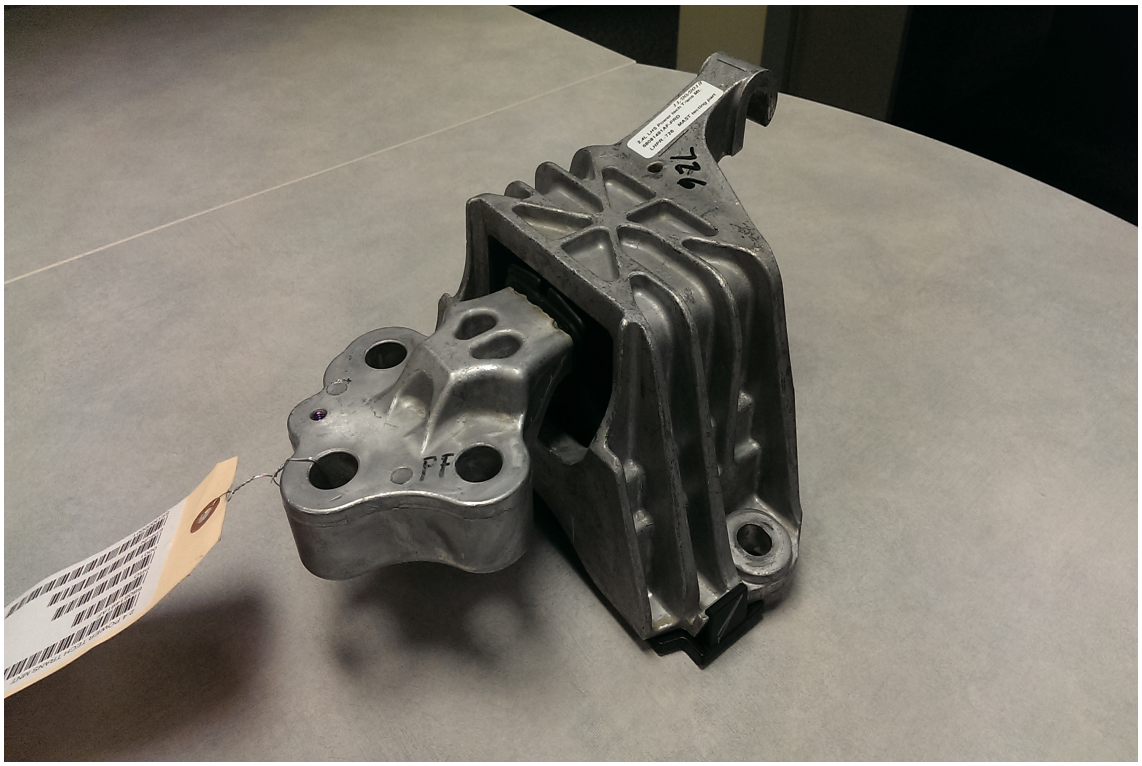


Figure C.1: Typical powertrain Hydro mount.

VITA AUCTORIS

Xiaowu Yang was born in 1989 in Zhenjiang, Jiangsu, China. He graduated from Jiangsu Provincial Zhenjiang No.1 High School in 2008. From there he went on to the Jiangsu University where he obtained a B.Eng. in Automotive Engineering in 2012. He is currently a candidate for the Master's degree in Mechanical Engineering at the University of Windsor and hopes to graduate in Summer 2014.