Automatic Generation of Real-Time Simulation Code for Vehicle Dynamics using Linear Graph Theory and Symbolic Computing

by

Kevin Warren Morency

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2007
©Kevin Morency 2007

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In recent years, hardware-in-the-loop (HIL) simulation has assumed a prominent role in the vehicle development process. A physical part, which may be a prototype at any stage of development, is tested, while the rest of the vehicle is represented by a mathematical model. Vehicle models used with hardware-in-the-loop must be capable of simulating an event in less time than it takes the event to occur in reality. Fast simulation necessitates a model that is represented by very efficient simulation code. This thesis presents a procedure for automatically generating this simulation code, given a description of the vehicle as input.

For this work, a symbolic formulation procedure based on linear graph theory and the principle of orthogonality is used to generate governing equations for vehicle systems; this procedure forms the basis of the DynaFlexPro software package. In order to generate simulation code for vehicle dynamics studies, the DynaFlexPro component model template was extended to include rules for calculating intermediate variables and rules for calling external functions. These changes enabled the development of a tire component model, known as DynaFlexPro/Tire, that adds critically important (and computationally efficient) blocks to the overall vehicle simulation code. The combination of DynaFlexPro and DynaFlexPro/Tire allows analysts to construct a model for any vehicle topology and gives analysts great freedom to define how tire forces and moments will be calculated. Simulation code describing the vehicle model is automatically generated using symbolic computing techniques. The accuracy of the approach was validated by comparing results for DynaFlexPro vehicle models to results for equivalent models developed in a well-established tool for vehicle dynamics simulation (MSC.ADAMS).

Two different vehicle models were constructed using DynaFlexPro and DynaFlexPro/Tire: a generic 4-wheeled vehicle with independent suspension and an articulated forestry skidder. Both models had an open-loop topology. When appropriate modeling variables were selected, each model was described by a minimal set of ordinary differential equations (ODEs) and the simulation code generated by DynaFlexPro was capable of being used for hardware-in-the-loop applications; the braking and handling behavior of the example models was simulated faster than real-time on a desktop PC with a 3.2 GHz Pentium 4 processor and 1 GB of RAM. For the same vehicle models, a different choice of modeling variables resulted in a mixed set of differential and algebraic equations (DAEs); in that case, HIL-capable simulation code could not be consistently generated. The approach works well for vehicle models described by ODEs, but more research is needed into the treatment of DAEs for real-time simulation of vehicle dynamics.

Acknowledgments

This research would not have been possible without the patience and guidance of Dr. John McPhee, and the extraordinary genius of Dr. Chad Schmitke. These men laid the foundation that my work is built upon.

I would also like to thank Mathieu Léger, Mike Wybenga, Adel Izadbakhsh, Kiumars Jalali, Matt Millard, Tom Uchida, Sukhpreet Sandhu, Reto Moser, and Willem Petersen for the many interesting discussions that contributed greatly to producing an intellectually stimulating work environment.

Financial support was provided by the Natural Sciences and Engineering Research Council of Canada.

Contents

List of Tables

List of Figures

Notation

a	scalar variable
$\dot{a}$	first derivative of a variable with respect to time
$\ddot{a}$	second derivative of a variable with respect to time
$\hat{a}$	unit vector
$\underline{a}$	vector or scalar variable
$\vec{a}$	vector variable
$\mathbf{a}$	column matrix of scalars
$\underline{\mathbf{a}}$	column matrix of vectors (may also contain scalars)
$\mathbf{A}$	matrix of scalars
{ }	ordered list of variables
[ ]	elements of a matrix
[ ]	index into a list, array, or matrix

Chapter 1

Introduction

1.1 Background

Simulation has assumed a central role in modern product design and development. Design changes can be evaluated with a computer model in a fraction of the time it would take to construct, instrument, and test a physical prototype. The material and labor costs associated with computer models are usually much lower than the costs associated with physical testing.

The job of simulating an engineering system consists of three steps:

1. create an idealized model of the system
2. generate equations that describe the idealized model
3. solve the equations numerically

The first step is arguably the most important. It requires application of engineering judgment to determine which characteristics of the system are important and which can be neglected [41]. The important characteristics must then be represented using idealized elements such as rigid bodies, massless springs, resistors, capacitors, etc.

The second step is the primary interest of this work. It involves the formulation of differential and algebraic equations that describe the behavior of the idealized model, and the arrangement of these equations in a form that can be easily and efficiently solved.

The numerical results obtained in step 3 could refer to an equilibrium solution, modal analysis, or time-domain simulation. For mechanical systems, forward dynamics simulation involves numerically integrating the governing equations to obtain the motion of the system at discrete instances in time. Inverse dynamics simulation involves solving for the forces and moments necessary to cause a prescribed motion. In this thesis, the term *simulation* will refer to forward dynamics simulation.

One measure of the effectivenesses of a model concerns how quickly the system behavior can be simulated. In situations where the virtual model interacts with people or

physical parts, results must be provided faster than real-time. That is to say, an event must be simulated in less time than it takes the event to occur in reality. There are a host of interesting applications for real-time simulations, particularly in the automotive industry.

Modern vehicles have many features that require an intelligent control strategy, including traction control, anti-lock brakes, electronic stability programs, and active suspensions. Auto manufacturers must ensure high quality standards by testing the electronic control units (ECUs) before they are used in production cars. Vehicle dynamics controllers typically operate in safety-critical conditions, at the limits of braking and cornering performance, which are difficult to reproduce during driving tests. Physical tests can be dangerous for test drivers and may destroy prototype vehicles. Hardware-in-the-loop (HIL) testing, in which a physical ECU is married to a simulated vehicle, has become indispensable for the development of modern vehicle dynamics controllers [12, 48].

Real-time simulations may also be performed by the controller itself, in order to define the control logic. The dynamics of the vehicle system are non-linear, and traditional proportional-integral-differential (PID) control schemes can only be optimized for vehicle states close to certain reference states. On the other hand, model-based control can be robust over a range of states, including safety-critical situations [1].

Vehicle models are used in driving simulators, which have both hardware and humans in-the-loop. Driving simulators may be used to investigate the behavior of human drivers under simulated conditions and to perform human factors research [22]. They are also useful as a virtual proving ground, where the ride and handling of modeled vehicles can be evaluated. In this case, the driving simulator is equipped with a force-feedback steering wheel and actuators that move the simulator platform in order to allow the driver to experience and evaluate the "feel" of driving the actual vehicle [22, 50].

Commercial software packages exist that are specifically designed for real-time simulation of vehicle dynamics. These packages are based on hard-coded equations that describe a generic vehicle model. Analysts can specify model *parameters* — numeric values for lengths, masses, spring stiffnesses, etc. — but cannot change the equations in which these parameters are used. If an analyst wishes to specify the *topology* — the way components are connected to each other — for his vehicle model, he must use a more general multibody dynamics tool to formulate equations that describe his unique topology. The equations can be formulated numerically or symbolically.

Numeric formulations create matrix structures that describe the dynamics of the system at a given instant in time [6, 17]. Since the numerical matrices are only valid for a given instant, they must be reformulated at every time step during the simulation. MSC.ADAMS® and Altair MotionSolve® are popular, commercially available programs for modeling and simulating multibody systems based on numeric formulations. These programs are not designed for real-time operation and have no provision for interfacing with hardware [42]. They generate non-linear differential and algebraic equations (DAEs)

that require iterative solution methods that run slowly with variable time steps. These factors prohibit the use of numerically formulated models for real-time applications.

Symbolic formulations combine parameters and modeling variables to create a set of equations that describe the system for all time. Although the governing equations are symbolic in nature, numerical methods must still be employed to solve them. Symbolic models lend themselves to real-time applications because they only need to be formulated once, rather than at every time step during the simulation, and also because symbolic equations can be greatly simplified [10, 14]:

- terms multiplied by 0 are automatically removed, and multiplications by 1 are ignored
- trigonometric reductions, substitutions, and simplifications can be performed
- repeated terms can be identified and computed only once, rather than each time they appear in the equations

Because symbolic models separate the formulation phase from the simulation phase, they have the added advantage of being very flexible and portable. The equations do not have to be solved using the same software that was used to formulate them; they can be translated to any simulation language the analyst chooses. In contrast to numeric formulations, symbolic formulations allow the analyst to view the governing equations in a meaningful form, and perhaps gain insight into the system by doing so.

The major disadvantage to symbolic formulations is the tendency of the equations to grow large and unwieldy as the system becomes more complex. For very large systems, the memory required to manage the symbolic equations can become too great for the hardware to handle, and the formulation must be halted. An intelligent formulation procedure can alleviate this difficulty [14].

1.2 Research Contributions

State-of-the-art pneumatic tire models have properties that require special treatment when including them within a larger multibody model, particularly if a symbolic formulation approach is to be used. The main contribution of this research is a methodology for including pneumatic tires as modeling components within a symbolic formulation procedure based on linear graph theory. The linear graph formulation procedure has already been developed by McPhee and Schmitke, and currently forms the basis for the DynaFlexPro software package [26, 46, 47]. Inclusion of a tire component model within DynaFlexPro allows the automatic formulation of simulation code to describe any vehicle topology.

The choice to implement the tire component model as part of DynaFlexPro provides several advantages. The linear graph formulation procedure is equally applicable to mechanical, hydraulic, and electrical domains. Therefore, vehicle models that contain parts

in different energy domains can be easily assembled [55]. A linear graph formulation also provides the analyst with complete freedom to select the modeling variables used to describe the multibody portion of his system. Intelligent selection of modeling variables can result in significant reduction of the number and complexity of the governing equations.

The tire component model must contain intelligent rules for controlling the size and complexity of symbolic expressions used to calculate tire intermediate variables and must allow the calling of external functions from the main simulation code. Once these challenges are overcome, the ultimate goal of this research — enable the automatic generation of vehicle simulation code that is suitable for hardware-in-the-loop applications — can be achieved.

1.3 Thesis Structure

This thesis is divided into 6 chapters. The first describes the motivation for the proposed research and defines its goals. Chapter 2 is a review of literature pertinent to achieving the goals laid out in Chapter 1. Specifically, it looks at different approaches for real-time simulation of vehicle dynamics, symbolic equation formulation, and coordinate selection for multibody systems. Because the research presented in this thesis revolves around multibody vehicle models described by linear graphs, Chapter 2 also provides some background and defines terminology related to multibody dynamics and linear graph theory.

Chapter 3 describes how tire model inputs are calculated based on the kinematic state of the tire, how these inputs are used in two state-of-the-art tire models, and how forces and moments generated by pneumatic tires may be applied to a larger multibody model.

Chapter 4 describes the proposed method for dealing with pneumatic tires in the context of a symbolic formulation based on linear graph theory. A tire component model is presented; instances of the the tire component can be used to construct vehicle models. The mechanisms by which tire components contribute to the automatic generation of simulation code for vehicle models are discussed in Chapter 4. Symbolic computing techniques, used for simplifying the simulation code, are also discussed.

The concepts presented in Chapter 4 were implemented as part of the DynaFlexPro software package, which was used to generate simulation code for two example vehicles, as described in Chapter 5. Results for cornering and braking maneuvers are verified against results from equivalent vehicle models created in MSC.ADAMS. Simulation times for DynaFlexPro models solved in Simulink are compared to real-time in order to assess the suitability of the modeling approach for hardware-in-the-loop applications. The effects of coordinate selection and symbolic expression manipulation on the efficiency of the simulation code generated for the two example vehicles are investigated and discussed.

The thesis concludes with Chapter 8, which provides a summary of the research performed and identifies potential areas for future research.

Chapter 2

Literature Review

2.1 Multibody Dynamics

Multibody systems are comprised of mechanical bodies interconnected by kinematic joints, springs, and dampers, which may also have applied forces, applied moments, or specified motions between bodies. Multibody dynamics software automatically generates the governing equations for a system, given a description of its components and a description of how the components are connected together, known as the system topology.

2.1.1 Dynamic Equations and Constraint Equations

The *degrees of freedom* for a mechanism may be defined as the number of inputs needed to completely specify its motion. The pendulum in Figure 2.1 has one degree of freedom. If the angle θ is known as a function of time, then the motion of the system would be completely specified.

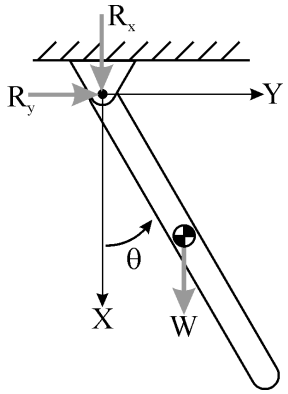


Figure 2.1: Single Pendulum

Following the notation used by Sayers, the variables chosen by the analyst to describe the position and orientation of the system are referred to as *coordinates* and the variables

chosen to represent the velocity and angular velocity of a system are referred to as *generalized speeds* [41, 44]. The more general term *modeling variables* will be used to refer to both coordinates and generalized speeds. There will be a differential equation known as a *kinematic transform* that relates the derivative of each coordinate to the generalized speeds; often, the generalized speeds are direct derivatives of coordinates, but this is not always the case. Because the Newton-Euler equations that describe the dynamics of mechanical systems are written in terms of accelerations and angular accelerations, the derivatives of generalized speeds will appear in the *dynamic equations* [44, 46].

Consider the case where the motion of a single pendulum is modeled using three coordinates: the x position of the center of mass, the y position of the center of mass, and the angle θ. There will also be three generalized speeds. Choosing the generalized speeds to be direct derivatives of the coordinates, they are: v_x, the x component of the velocity of the center of mass expressed in the inertial frame, v_y, the y component of the velocity of the center of mass expressed in the inertial frame, and ω, the angular velocity of the pendulum about an axis out of the page. For this case, the dynamic equations are given by equations (2.1)-(2.3) and the kinematic transforms are given by equations (2.4)-(2.6)

$$
\begin{aligned}
m\dot{v}_x &= W + R_x && (2.1)\\
m\dot{v}_y &= R_y && (2.2)\\
\frac{m}{12}l^2\dot{\omega} &= R_x\frac{l}{2}\sin(\theta) - R_y\frac{l}{2}\cos(\theta) && (2.3)
\end{aligned}
$$

$$
\begin{aligned}
\dot{x} &= v_x && (2.4)\\
\dot{y} &= v_y && (2.5)\\
\dot{\theta} &= \omega && (2.6)
\end{aligned}
$$

where l is the length of the pendulum, which is assumed to be a uniform slender rod, m is its mass, W is its weight, and R_x and R_y are reaction forces at the revolute joint.

If the number of coordinates chosen to model a system is greater than the number of degrees of freedom, then the coordinates are not independent. There will be a set of algebraic equations, referred to as *constraint equations*, that express the dependencies of the chosen coordinates. Holonomic constraints involve position-level coordinates, while non-holonomic constraints involve non-integrable relationships between generalized speeds [17]. The problems described in this thesis only involve holonomic constraints, so the discussion of constraints will be restricted to those of the holonomic variety.

The degrees of freedom f is related to the number of coordinates n and the number of independent constraints m by:

$$f = n - m \qquad (2.7)$$

For the pendulum example, there are two independent constraint equations that relate the three chosen coordinates:

$$x = \frac{l}{2}\cos(\theta) \tag{2.8}$$

$$y = \frac{l}{2}\sin(\theta) \tag{2.9}$$

Equations (2.1 - 2.6) and (2.8 - 2.9) are a set of 8 equations with 8 unknowns (v_x, v_y, ω, x, y, θ, R_x, and R_y). They may be solved to obtain the response of the system.

Many authors assume that generalized speeds are always direct derivatives of coordinates. They do away with the kinematic transforms and write the dynamic equations in terms of the second derivative of the coordinates [6, 14, 17]. Haug proposed a matrix form for the governing equations of multibody systems, which is presented in equations (2.10) and (2.11). Equation (2.10) is a general form of the dynamic equations, in which $\mathbf{M}$ is a generalized mass matrix, $\mathbf{q}$ is a column vector containing the system coordinates, $\mathbf{\Phi}_q^T\boldsymbol{\lambda}$ is a column vector representing the effect of reaction forces on the system dynamics, and $\mathbf{Q}$ is a column vector representing the effect of applied forces and quadratic speed terms (centripetal, Coriolis). Equation (2.11) is a general form of the algebraic constraint equations.

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{\Phi}_q^T\boldsymbol{\lambda} = \mathbf{Q} \tag{2.10}$$

$$\mathbf{\Phi} = \mathbf{0} \tag{2.11}$$

The pendulum equations can be rearranged into Haug's matrix form, as shown in equations (2.12) and (2.13).

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & \frac{m}{12}l^2 \end{bmatrix} \begin{Bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{Bmatrix} + \begin{Bmatrix} -R_x \\ -R_y \\ -R_x\frac{l}{2}\sin(\theta) + R_y\frac{l}{2}\cos(\theta) \end{Bmatrix} = \begin{Bmatrix} W \\ 0 \\ 0 \end{Bmatrix} \tag{2.12}$$

$$\begin{Bmatrix} x - \frac{l}{2}\cos(\theta) \\ y - \frac{l}{2}\sin(\theta) \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{2.13}$$

As noted by Haug, the constraint equations are usually non-linear functions of the coordinates, $\mathbf{q}$. However, once the position variables are known, the velocities and accelerations can be found by solving sets of linear equations [17]. Taking the time derivative of equation (2.11) results in equation (2.14), which is linear in $\dot{\mathbf{q}}$. Differentiating again gives equation (2.15), which is linear in $\ddot{\mathbf{q}}$.

$$\mathbf{\Phi}_q\dot{\mathbf{q}} = -\mathbf{\Phi}_t \tag{2.14}$$

$$\mathbf{\Phi}_q\ddot{\mathbf{q}} = -\left(\mathbf{\Phi}_q\dot{\mathbf{q}}\right)_q\dot{\mathbf{q}} - 2\mathbf{\Phi}_{qt}\dot{\mathbf{q}} - \mathbf{\Phi}_{tt} = \boldsymbol{\gamma} \tag{2.15}$$

The matrix $\mathbf{\Phi}_q$ that appears in the dynamic equations as well as the velocity-level and acceleration-level constraint equations is known as the *Jacobian*. It is a matrix of partial derivatives related to the column position-level constraints, $\mathbf{\Phi}$, and is calculated as per equation (2.16).

$$\mathbf{\Phi}_{q[i,j]} = \frac{\partial \mathbf{\Phi}_{[i]}}{\partial \mathbf{q}_{[j]}} \tag{2.16}$$

The Jacobian matrix for the pendulum example is:

$$\mathbf{\Phi}_q = \begin{bmatrix} 1 & 0 & \frac{l}{2}\sin(\theta) \\ 0 & 1 & -\frac{l}{2}\cos(\theta) \end{bmatrix} \tag{2.17}$$

Multiplying a column vector of Lagrange multipliers, $\boldsymbol{\lambda}$, by the transpose of the Jacobian, $\mathbf{\Phi}_q^T$, will yield a column vector that represents the effect that reaction forces have on the system dynamics. This term has been pointed out in equation (2.10).

2.1.2 Modifying the Form of Equations in Preparation for Simulation

In general, the mathematical description of a multibody system is a mix of differential and algebraic equations (DAEs). The algebraic constraint equations, $\mathbf{\Phi} = \mathbf{0}$, are usually highly non-linear, and therefore require an iterative solution which tends to be computationally expensive. Methods for solving sets of ordinary differential equations (ODEs) are more efficient than methods for solving DAEs. For this reason, many authors have tried to eliminate algebraic equations from the mathematical description of multibody systems. One common method of accomplishing this goal is to use the acceleration form of the constraint equations, equation (2.15), instead of the position form, equation (2.11). The equations to be solved then become:

$$\begin{bmatrix} \mathbf{M} & \mathbf{\Phi}_q^T \\ \mathbf{\Phi}_q & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \\ \boldsymbol{\gamma} \end{bmatrix} \tag{2.18}$$

which contains a set of $n + m$ differential equations involving $\ddot{\mathbf{q}}$. These equations can be integrated to obtain $\dot{\mathbf{q}}$ and $\mathbf{q}$, as well as $\boldsymbol{\lambda}$.

The problem with expressing the system equations in this form is that the constraint equations are only satisfied at the acceleration level. If no corrections are made, a buildup of integration error may cause significant violation of the position and velocity constraints [17]. If the position constraints, $\mathbf{\Phi} = \mathbf{0}$, are not satisfied, the solution may wrongfully indicate that bodies are floating apart when they should in fact be joined.

To keep the error in the position constraints from growing out of control, Baumgarte constraint stabilization may be employed [4]. Adding a few extra terms to equation (2.15) results in a differential equation which is inherently stable, implying that $\dot{\mathbf{\Phi}} \approx \mathbf{\Phi} \approx \mathbf{0}$.

$$\mathbf{\Phi_q}\ddot{\mathbf{q}} = \boldsymbol{\gamma} - \mathbf{2}\alpha\left(\mathbf{\Phi_q}\dot{\mathbf{q}} + \mathbf{\Phi_t}\right) - \beta^{\mathbf{2}}\mathbf{\Phi} = \boldsymbol{\gamma}' \tag{2.19}$$

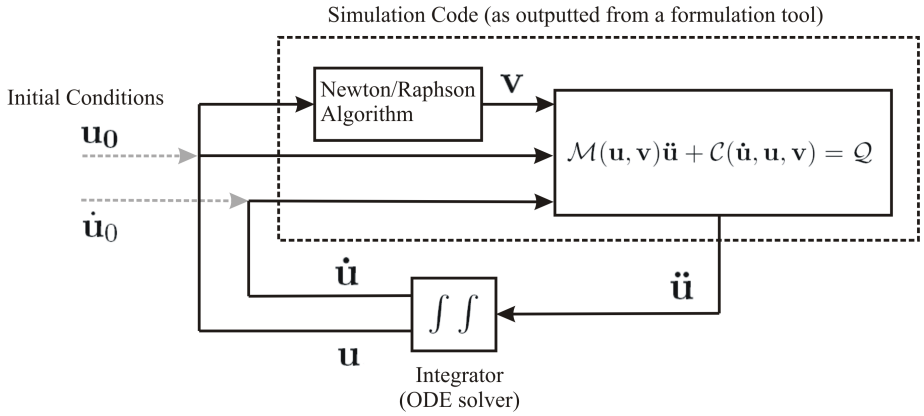


Figure 2.2: Simulation Procedure when Coordinate Partitioning is Used

To implement Baumgarte constraint stabilization, $\boldsymbol{\gamma}$ is replaced by $\boldsymbol{\gamma}'$ in equation (2.18). Unfortunately, there is no general and uniformly valid method of selecting parameters, α and β, that will result in a fast and accurate solution [17].

Some authors have tried to reduce the system equations by eliminating dependent coordinates. It is theoretically possible to express the dynamic equations for a multibody system as a set of f ODEs, where f is the number of degrees of freedom [14, 17]. The first step in this process is to partition the coordinates into a column vector of f independent coordinates $\mathbf{u}$ and a column vector of m dependent coordinates $\mathbf{v}$.

$$\mathbf{q} = \left\{ \begin{array}{c} \mathbf{u} \\ \mathbf{v} \end{array} \right\} \tag{2.20}$$

Fisette et al. show that dependent velocities and accelerations can be explicitly solved for, and eliminated from the dynamic equations [14]. Substituting symbolic expressions for these variables results in a reduced set of dynamic equations, of the form shown in equation (2.21).

$$\mathcal{M}(\mathbf{u},\mathbf{v})\ddot{\mathbf{u}} + \mathcal{C}(\dot{\mathbf{u}},\mathbf{u},\mathbf{v}) = \mathcal{Q} \tag{2.21}$$

Unfortunately, since the constraint equations are non-linear at the position level, the dependent coordinates cannot, in general, be eliminated from the dynamic equations. Figure 2.2 depicts the simulation scheme used by Fisette et al. based on coordinate partitioning and a reduced set of dynamic equations. An iterative Newton/Raphson algorithm for solving algebraic constraint equations is packaged along with the reduced set of dynamic equations inside of simulation code, which was formulated using a symbolic multibody program. Although algebraic equations exist, and are solved iteratively at every time step, from the point of view of the numerical integrator, the simulation code appears to describe a set of ODEs in the independent coordinates $\mathbf{u}$, and any ODE solver can be used for their solution [14].

If it were possible to find an analytical solution to the position-level constraint equations, then the Newton/Raphson block in Figure 2.2 could be replaced by an explicit equation for $\mathbf{v}$ in terms of $\mathbf{u}$, resulting in a more efficient simulation procedure. Kecskemethy and Hiller present a method for automatically recognizing cases where constraint equations can be solved analytically, and for incorporating their solution into the dynamic equations [20]. Elmqvist et al. pre-derived the analytic solution to the constraint equations for a number of assemblies and used these assemblies as building blocks to construct more complex topologies [12, 33].

For general topologies, outside of the exceptions pointed out by Kecskemethy and Elmqvist, analytic solutions to position-level constraint equations cannot be obtained. For general topologies, there are three options for dealing with algebraic constraint equations:

1. Use an ODE solver to integrate a reduced set of dynamic equations, coupled with a Newton/Raphson algorithm to solve the position-level constraint equations numerically at every time step

2. Use an ODE solver to integrate the acceleration form of the constraint equations, which are combined with the dynamic equations, and use Baumgarte constraint stabilization to prevent position-level constraint violation from growing out of control

3. Use a specialized solver capable of handling DAEs directly [9, 31]

2.1.3 Selection of Modeling Variables

It is possible to select different sets of modeling variables to describe the same multibody system. In Section 2.1.1, the equations for a single pendulum were presented in terms of three coordinates — x, y, and θ. An intelligent analyst would recognize that this single degree of freedom system can be represented by a single dynamic equation, and would select a single variable to describe its motion. If the revolute joint angle θ is chosen as a coordinate, then the dynamic equation can be written as shown in equation (2.22) and there will be no algebraic constraint equations. While this is a very simple example, comparing the single equation (2.22) to the five equations (2.12)-(2.13) presented earlier clearly shows that the choice of modeling variables can have a significant impact on both the number of equations used to describe a system and the complexity of those equations. Consequently, there is much literature available on the subject of coordinate selection for multibody mechanical systems [10, 25, 29].

$$\frac{m}{3}l^2\ddot{\theta} = -mg\frac{l}{2}\sin(\theta) \tag{2.22}$$

It is very common for numeric formulations to use absolute coordinates, meaning that 3 rotational coordinates and 3 translational coordinates are used to describe the motion of each body. Six dynamic equations are considered for every body in the system, in addition to numerous constraint equations enforcing joint connections between bodies.

Solving the equations in this form requires special integrators capable of dealing with DAEs, and involves finding not only the motion of the bodies but also the constraint forces [49].

Joint coordinates describe the relative motion between two adjacent bodies. For open-loop topologies, these coordinates provide a minimal set of dynamic equations — one for each degree of freedom — and no constraint equations are needed. The one-to-one mapping of degrees of freedom to modeling variables is lost for systems containing closed kinematic loops. Equations expressed in terms of joint coordinates are fewer in number than equivalent equations using absolute coordinates, but are more highly-coupled and complex [10].

Indirect coordinates measure the relative motion between two bodies that are not necessarily adjacent. McPhee and Redmond show that, for certain problems, indirect coordinates can lead to simpler equations than joint coordinates [25] .

As mentioned in Section 2.1.1, generalized speeds do not have to be direct derivatives of position-level coordinates. Mitiguy and Kane have shown that using different variables for the orientation and angular velocity of rigid bodies can result in a significant reduction in the complexity of the dynamic equations when compared to the more traditional case where angular velocities are simple derivatives of orientation variables [29].

2.2 Real-Time Simulation

2.2.1 Requirements for Real-Time Simulation

Sayers outlined some basic requirements that multibody models must meet if they are to be used for hardware-in-the-loop or human-in-the-loop applications [42]. The obvious requirement is that models must be described by efficient sets of equations so that they can be solved quickly using available numerical methods. Other requirements ensure synchronization between simulated parts and physical parts.

In an HIL environment, the hardware is instrumented with sensors that collect data at a constant sample rate. The sampling interval must be sufficiently small to capture the behavior of the physical parts, which may be rapidly changing and unpredictable. In order to maintain synchronization between the model and the physical parts, data must be exchanged every time the hardware is sampled [42]. Therefore the simulation time step is set equal to the hardware sampling interval.

The simulation must predict the state at the next time step in less time than it takes the step to occur in reality. In Figure 2.3, the gray boxes represent the amount of CPU time needed to advance the solution — i.e. the time when the simulation is active. In an HIL environment, the simulation will predict the state at the next time step, then wait for the hardware to catch up before exchanging data [42]. This is accomplished in Case 1 of Figure 2.3, where a happy face indicates a successful exchange of data between the model and the hardware.

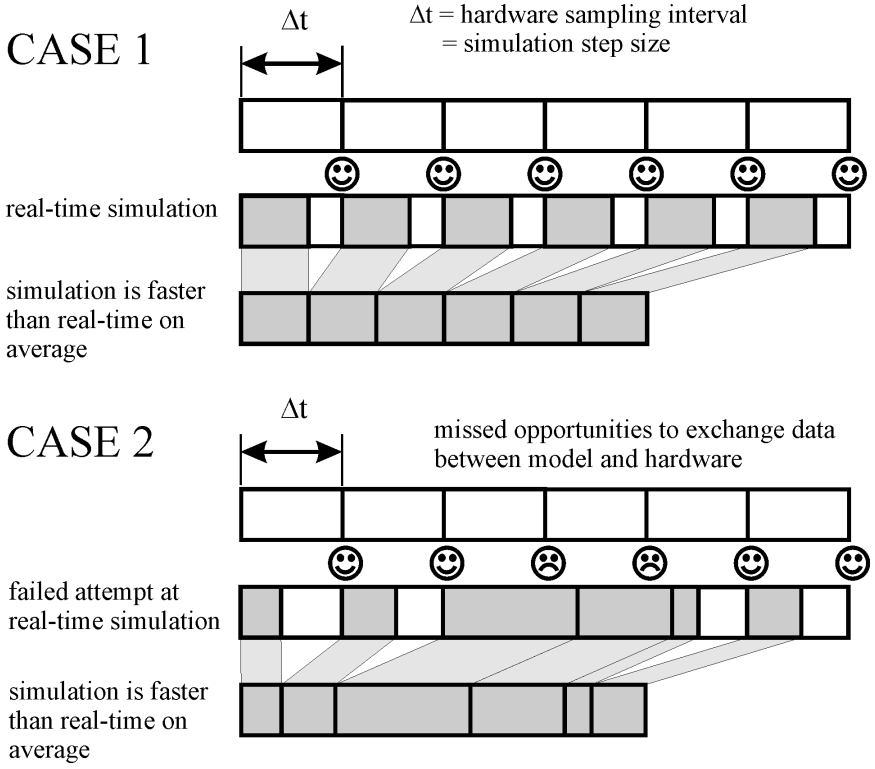


Figure 2.3: Simulation Requirements for Hardware-in-the-Loop Applications

Even if the simulation is faster than real-time on average, there could be time steps where the CPU time needed to advance the solution is greater than the length of the time step. In a HIL environment, this means the simulation missed opportunities to interact with the hardware, which could cause serious problems. Such a situation, depicted in Case 2 of Figure 2.3, must be avoided.

Situations similar to Case 2 are likely to occur when equations have to be solved in an iterative fashion, as with a Newton-Raphson algorithm. Iterative methods are required whenever the mathematical model contains algebraic constraint equations that are to be solved on the position-level, whenever an implicit solver is used, or whenever a variable step size solver is used. The number of iterations needed could be small or large, depending on the state of the system. For this reason, the CPU time needed to advance the solution can vary greatly from time step to time step. Implicit or variables step size solvers can only be used for real-time applications if the number of iterations performed at each time step is limited to a fixed value [12, 13, 41, 48]

Conversely, if the mathematical model is comprised entirely of differential equations, and an explicit, constant step size solver is used, a consistent amount of CPU time will be spent on every time step since the same number of operations will be performed at every time step [42]. This leads to situations similar to that depicted in Case 1 of Figure 2.3.

The requirements of a mathematical model to be used for real-time applications, such as hardware-in-the-loop simulation, may be summarized as follows:

- the model equations must not be too large or complicated — the available computing resources must be able to integrate the equations faster than real-time
- the model must be expressed in a form that requires a consistent amount of computations at every time step — a set of purely differential equations (ODEs) is ideal
- the numerical solver (integrator) must use a consistent amount of computations at every time step — an explicit, constant step size solver is ideal

The most common ODE solver for real-time applications is Euler's method. This is the simplest possible solver, but provides enough accuracy over the small time steps (≈ 1 ms) that are typical for HIL applications [12, 40, 42, 48]. Higher-order methods can be more accurate, but are not typically used when essential variables in the model are sampled from hardware because measurement noise violates the assumed smooth transitions needed for higher-order methods to work well [42, 48].

2.2.2 Specialized Programs for Real-Time Simulation of Vehicle Dynamics

Programs that are designed specifically for real-time simulation of vehicle dynamics contain hard-coded differential equations representing a particular vehicle topology. Some of the most popular commercial programs are CarSim® from Mechanical Simulation Corporation, ADAMS/Car RealTime® from MSC.Software, and ve-DYNA® from Tesis DYNAware. While the user can modify parameters such as center of gravity height and mass/inertia properties, she cannot change the topology of the underlying model [30, 42, 51].

The underlying topology is usually a generic four-wheeled vehicle with independent suspension of the type described by Sayers and depicted in Figure 2.4 [43]. Lumped masses are connected to the vehicle chassis by translational joints in parallel with springs and dampers that represent the suspension compliance. The lumped masses are a simplified representation of the inertia of the suspension components they replace. Each wheel is connected to its corresponding lumped mass with a revolute joint that allows the wheel to spin. The front wheels also have a revolute joint that allows them to steer. This is an open-loop topology, so it is possible to describe its behavior with a set of ODEs (no algebraic constraint equations are required). This topology provides 14 degrees of freedom : 6 associated with the 3-dimensional motion of the vehicle body, 4 associated with suspension deflection, and 4 associated with wheel spin. The steer angles of the front wheels are specified functions of time, so they do not add degrees of freedom [43].

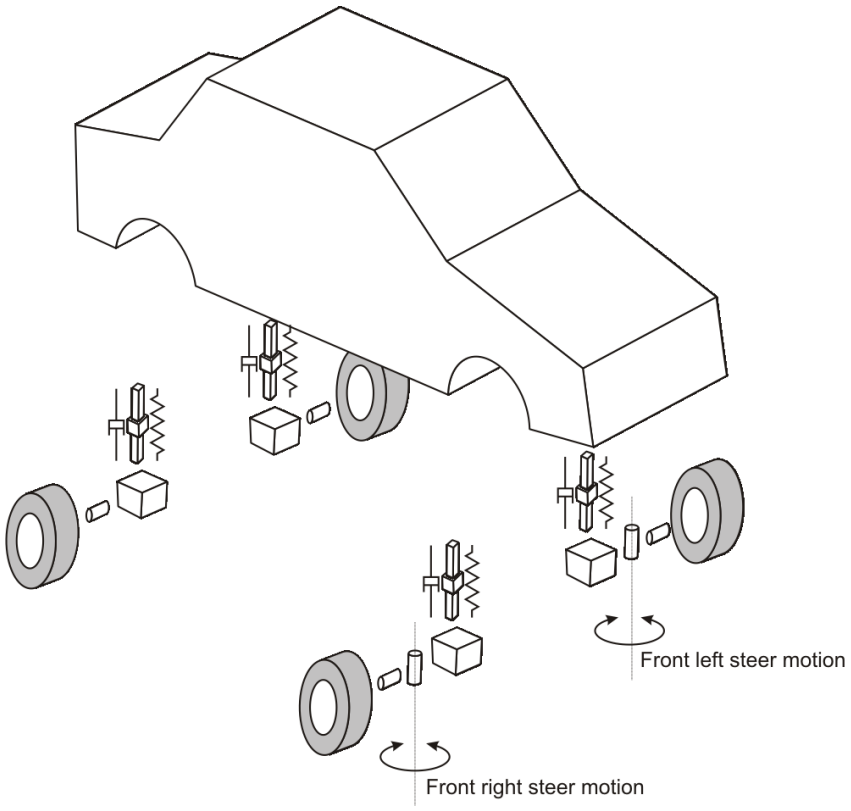


Figure 2.4: Generic Four-Wheeled Vehicle Model with Independent Suspension

Some software providers offer two or three topologies for which simulation code has been pre-derived. For instance, Mechanical Simulation Corporation offers a four-wheeled topology which it calls CarSim®, a two-wheeled topology which it calls BikeSim®, and a topology involving a towed trailer, which it calls TruckSim® [27, 28]. However, it is impossible for the analyst to investigate non-standard vehicles, such as unicycles and three-wheeled vehicles, using these programs.

Because the same generic topology is used to represent all suspensions, the user must specify suspension properties with a series of look-up tables rather than with joints and link geometry [28, 30, 42, 51]. This method essentially assumes that the inertia of the suspension members has a negligible effect on the overall vehicle dynamics, but attempts to incorporate the significant effect that the suspension has on the orientation of the wheel, and therefore the forces and moments generated by the tire. In order to populate the look-up tables, the kinematics and compliance of the suspension must be determined beforehand using a separate modeling/simulation package, or measured on a test rig.

Despite their fixed topology and reliance on look-up tables, these specialized programs have several advantages. They come with complete graphical user interfaces and are loaded with special features including several available tire models, driver models, and 3D roads. CarSim, ADAMS/Car RealTime, and ve-DYNA all have provisions for exporting

their models to Matlab/Simulink [28, 30, 51]. This is a very important feature since much of the software and hardware necessary for hardware-in-the-loop applications is designed to interact with Simulink models. This technology comes from companies like Opal-RT and DSpace. It includes provisions for parallel computing (necessary for real-time simulation of very large models), software that enforces the stop-and-wait behavior noted in Figure 2.3, as well as interface hardware necessary for exchanging information between the model and physical parts [48].

2.3 Symbolic Modeling of Multibody Systems

The pre-derived models discussed in the previous section are extremely efficient and suitable for real-time applications. However, they force the analyst to make his vehicle model conform to a specific topology. If the analyst wants to investigate an issue that is not addressed by that topology, he must develop his own model. Deriving the system equations by hand is time consuming and error-prone. Numeric formulations are not appropriate for real-time applications. A symbolic formulation approach is needed that automatically generates simulation code given a description of the vehicle as input. The following sections describe some important research in the field of symbolic modeling of multibody systems, as well as some existing software tools.

2.3.1 ROBOTRAN

Fisette et al. developed ROBOTRAN, which automatically generates the symbolic equations of motion for any *mechanical* system given a description of the system as input [14]. ROBOTRAN is a standalone program written in the C language; all symbolic operators such as addition and differentiation have been hard-coded by the developers. Mathematical models are exported in the form of Fortran, C, or Matlab routines [14].

ROBOTRAN exclusively uses joint coordinates to model multibody systems. If algebraic constraints exist, then the coordinate partitioning method described in Section 2.1.2 is used to obtain a reduced set of dynamic equations. While the algebraic constraint equations must still be solved using an iterative method within the simulation code, from the point of view of the numerical solver the simulation code appears to represent a set of differential equation in the independent coordinates $\mathbf{u}$ so any ODE solver may be used for their solution [14].

Postiau et al. used ROBOTRAN to formulate a 3D vehicle model with multi-link suspension at all wheels. Simulation code was exported in the C language and solved in the Matlab/Simulink environment. A 10 second double lane change maneuver took 11 seconds to simulate on a Pentium 3 450 MHz computer [14, 37].

2.3.2 AutoSim

In the early 1990s, researchers at the University of Michigan developed AutoSim, which automatically formulates simulation code given a description of a *mechanical* system as input. AutoSim is a standalone program written in the Lisp language. The symbolic formulation is based on Kane's method [21].

The user is not given the option of selecting his own modeling variables. AutoSim primarily uses joint coordinates to describe the motion of the system, but heuristic rules are applied to select different coordinates in certain situations. For instance, if a body is connected to its parent by a joint that allows 3 rotational degrees of freedom, then absolute angular coordinates will be used for that body (orientation measured with respect to the ground frame, as opposed to a frame fixed in the parent body) [41].

An AutoSim user has the option of identifying certain state variables as "small". Products of small variables are dropped if they are of order 2 or higher, and trigonometric functions of small variables are replaced with truncated Taylor series expansions [41, 44]. This mimics the way that a human analyst would formulate equations using a paper and pencil — dropping terms he believes to be insignificant in order to simplify the equations.

AutoSim exports equations in the form of C or Fortran code. Constant terms are computed once, before the simulation starts, instead of being recomputed at every time step [41, 44]. This allows parameters to be left as symbols during the formulation of simulation code (numeric values to be specified later) with no penalty on simulation time.

When the system contains closed kinematic loops, a coordinate partitioning approach is used. The dependent speeds are integrated to give an estimate of the dependent coordinates $\mathbf{v}$ at the next time step. These estimates are unlikely to satisfy the position constraints exactly, so they are corrected using a modified Newton/Raphson algorithm. An iterative method is needed to solve for dependent coordinates that satisfy the position constraints within a certain tolerance. AutoSim specifies the number of iterations in its Newton/Raphson algorithm to be 1 [41]. While this does not ensure that the position constraints are exactly satisfied, it is enough to prevent constraint violations from growing out of control and ensures that a consistent number of operations is performed at every time step during the simulation (one of the requirements for HIL) [41].

Sayers used AutoSim to formulate a model for a generic 4-wheeled vehicle as described in his 1996 paper [43]. This model underwent some fine-tuning by hand and was incorporated into the CarSim® commercial software package. CarSim was used for hardware-in-the-loop testing of a brake system electronic control unit (ECU), during which the vehicle model ran faster than real-time on a laptop equipped with a Pentium 2 233MHz processor [42].

2.3.3 Modelica / Dymola

Modelica is an object-oriented language for describing *multi-domain* engineering systems (i.e. systems that may contain combinations of mechanical, hydraulic, and electrical parts). The ability to model the interactions between different energy domains is noteworthy since modern vehicles are fitted with many electrical actuators, and are becoming increasingly mechatronic.

The Modelica language is intended to serve as a standard format so that models arising in different domains can be exchanged between tools and users [11]. Each component is represented as an object, which may contain differential, algebraic, and discrete equations as well as internal functions. In order for systems to be simulated, a Modelica translator must be used to assemble the component equations into a meaningful form and numerically integrate them. The most popular Modelica translator is Dymola® from DynaSim.

Dymola users are not given the option of selecting their own modeling variables. Dymola uses graph-theoretic algorithms to automatically select modeling variables, but the literature is not clear on the extent to which graph theory is used to assemble the system equations [12, 13]. If algebraic constraint equations are present, specialized techniques are used for manipulating the DAEs into a form that can be solved efficiently [7, 12, 13].

Elmqvist et al. used Dymola to construct models of various vehicle topologies [12]. One of these topologies was a generic four-wheeled vehicle with independent suspension, similar to the one described by Sayers [43], which used look-up tables to modify wheel orientation as a function of suspension deflection. This model was exported from Dymola in the form of C code, which was compiled on a real-time target machine using software from Matlab and Opal-RT. The target machine had a 3 GHz Pentium 4 processor. A double lane-change maneuver was simulated using Euler's method and a constant step size of 1 ms. The model easily ran in real-time, as only 0.1 ms of CPU time was needed to advance the solution at each time step[12]. As part of the same study, a model of a vehicle featuring MacPherson strut front suspension and trailing arm rear suspension was created. An analytic solution to the constraint equations associated with the MacPherson strut suspension is possible, and was easily incorporated into the model by using Dymola *assemblies* [12]. This eliminated the need for algebraic constraint equations and allowed the model to be described by a set of ODEs. The simulation code was exported in the same manner as before, and a double lane-change maneuver was simulated. The model ran faster than real-time, as 0.3 ms of CPU time was needed to advance the solution for each 1 ms time step [12].

2.3.4 DynaFlexPro

McPhee and Schmitke have developed DynaFlexPro, which automatically formulates symbolic equations for *multi-domain* engineering systems. The formulation procedure is based on linear graph theory and the principle of orthogonality. A brief overview of linear graph

theory is given in the next section. For more detail, the reader is referred to [26, 46, 47].

DynaFlexPro is implemented in the general-purpose computer algebra package, Maple®. Many of Maple's symbolic math capabilities are used by DynaFlexPro, including the **combine()** and **simplify()** commands for performing trigonometric reductions, and the **CodeGeneration** package for optimizing procedures to eliminate unnecessary calculations and translating simulation code to different languages. A user of DynaFlexPro can inspect the system equations in the Maple environment and perform additional operations on them using Maple commands if she sees fit. This can be quite useful for certain applications. For example, taking symbolic derivatives of the governing equations can be beneficial for the computation of design sensitivities [36]. DynaFlexPro can generate optimized simulation code in the Maple, Matlab, and C languages.

The linear graph formulation used by DynaFlexPro offers the analyst a great deal of choice and flexibility in the area of coordinate selection. McPhee and Redmond show that analysts can choose absolute, joint, or indirect coordinates by selecting an appropriate tree for the linear graph formulation [25]. It is also possible to select a hybrid of these coordinate types, which McPhee referred to as branch coordinates [24]. Users may choose generalized speeds that are not direct derivatives of position coordinates by using different unit vectors to span the position-level across space and velocity-level across space for edges in the linear graph [46].

2.4 Linear Graph Theory

2.4.1 Basic Terminology

Graph theory represents physical systems as a collection of nodes and edges. In electrical systems, the nodes are points where the components may be soldered to one another and edges represent the components themselves. In mechanical systems, where orientation is important, nodes represent reference frames, and edges represent transformations between reference frames [46].

An edge in a linear graph has two types of variables associated with it. *Across* variables represent quantities measured by a device in parallel with the edge, while *through* variables represent quantities measured by a device in series with the edge [2, 47]. In this thesis, $\underline{x}_k$ is an across variable associated with edge k, and $\underline{y}_k$ is a through variable associated with edge k.

The energy domain in which an edge exists dictates the physical meaning of its through and across variables, as shown in Table 2.1. In the electrical domain, through variables are currents i and across variables are voltages e. In the mechanical translational domain, through variables are forces $\vec{F}$ and across variables are positions $\vec{r}$, velocities $\vec{v}$, and accelerations $\dot{\vec{v}}$. In the mechanical rotational domain, through variables are moments $\vec{M}$ and across variables are angles $\vec{\theta}$, angular velocities $\vec{\omega}$, and angular accelerations $\dot{\vec{\omega}}$ [46].

The representation of angles as vectors follows the notation used by Schmitke [45].

	Derivative Level	Electrical	Mechanical (Translation)	Mechanical (Rotation)
$\underline{x}_k$	0	e_k	$\vec{r}_k$	$\vec{\theta}_k$
	1	$\dot{e}_k$	$\vec{v}_k$	$\vec{\omega}_k$
	2		$\dot{\vec{v}}_k$	$\dot{\vec{\omega}}_k$
$\underline{y}_k$	0	i_k	$\vec{F}_k$	$\vec{M}_k$
	1	$\dot{i}_k$	$\vec{F}_k$	$\vec{M}_k$
	2		$\vec{F}_k$	$\vec{M}_k$

Table 2.1: Through and Across Variables in Different Domains

3D rotations are not commutative, and therefore do not satisfy the strict mathematical definition of a vector. However, vector notation can be used successfully in a linear graph formulation if the order of rotations is given proper consideration. For instance, the orientation across an edge with three rotational degrees of freedom may be written

$$\vec{\theta}_{edge} = \theta_{1\_edge}\hat{u}_1 + \theta_{2\_edge}\hat{u}_2 + \theta_{3\_edge}\hat{u}_3 \tag{2.23}$$

where each unit vector is a function of the previous rotations. If 313 Euler angles are used to keep track of an edge's rotation, then $\hat{u}_1$ describes the z direction of the edge's start (tail) frame, $\hat{u}_2$ describes an x axis that has been rotated an angle θ_{1_edge} about $\hat{u}_1$, and $\hat{u}_3$ describes a z axis that has been rotated an angle θ_{2_edge} about $\hat{u}_2$.

McPhee and Schmitke present a component template that can be used to represent subsystems of a larger model [46]. A linear graph *component* may contain more than one edge and may span several domains. For example, the revolute joint component model shown in Figure 2.5 contains an edge in the mechanical translational domain and two edges in the mechanical rotational domain.

The physical nature of a component is associated with *terminal equations* that relate the through and across variables of its edges [2, 46]. A terminal equation is defined for each derivative level of each edge. In mechanical systems, this means that separate terminal equations are used to describe the position, velocity, and acceleration behavior associated with a particular edge. Often, higher derivative level terminal equations are simple derivatives of lower derivative level equations, but this is not always the case [46].

As an example, terminal equations for revolute joint edges are presented in Table 2.2. The terminal equations for the translational edge ensure that the origin of the F1 frame, attached to body 1, is coincident with the origin of the F2 frame, attached to body 2. The terminal equations for the first edge in the rotational domain state that this edge develops no torque. In fact, the only reason edge R1 is part of the revolute joint component is because its angular velocity, $\vec{\omega}_{R1}$ is needed in the terminal equations for the second rotational edge. The position-level terminal equation for edge R2 states that the rotation transformation between frame F1 and F2 is the result of a single rotation about

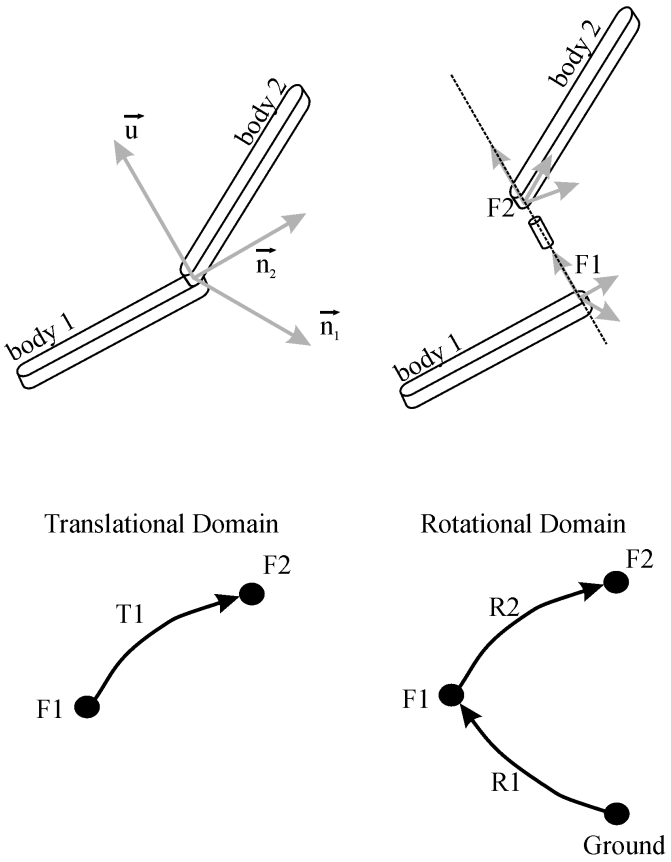


Figure 2.5: A Linear Graph Representation of a Revolute Joint Component

the the revolute axis $\hat{u}$. This is also evident from the velocity-level terminal equation for this edge. The acceleration-level equation for edge R2 must contain the cross product of $\vec{\omega}_{R1}$ and $\vec{\omega}_{R2}$ in order to account for gyroscopic effects.

Edge	0th Derivative Level (position)	1st Derivative Level (velocity)	2nd Derivative Level (acceleration)
T1	$\vec{r}_{T1} = \vec{0}$	$\vec{v}_{T1} = \vec{0}$	$\dot{\vec{v}}_{T1} = \vec{0}$
R1	$\vec{M}_{R1} = \vec{0}$	$\vec{M}_{R1} = \vec{0}$	$\vec{M}_{T1} = \vec{0}$
R2	$\vec{\theta}_{R2} = \beta\hat{u}$	$\vec{\omega}_{R2} = \dot{\beta}\hat{u}$	$\dot{\vec{\omega}}_{R2} = \ddot{\beta}\hat{u} + \vec{\omega}_{R1} \times \vec{\omega}_{R2}$

Table 2.2: Terminal Equations for Revolute Joint Component

2.4.2 Tree Selection

One of the unique aspects of linear graph theory is that it separates individual component models, defined by *terminal equations*, from the *topological equations* that describe how components are connected. Regardless of the nature/complexity of a system's compo-

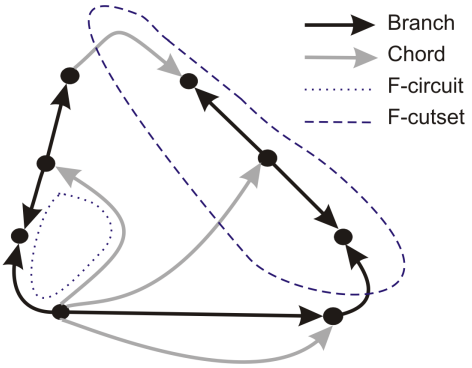


Figure 2.6: A Linear Graph with Valid Tree Selection

nents, the topological equations are always linear and may be formulated in a systematic fashion [2, 46].

An effective method for generating the topological equations begins by selecting a system *tree*. A tree is a connected subgraph that includes all of the nodes in a graph without any loops. The edges in the tree are called *branches* while the cotree edges (those not in the tree) are referred to as *chords* [2, 46]. Figure 2.6 shows a linear graph with a valid tree selection. Tree edges are shown in black and cotree edges are shown in gray. Note that it is possible to select different, equally valid, trees for the same graph.

Along with the selection of a tree comes the designation of the system's primary variables (variables that appear in the governing equations) and secondary variables (variables that do not appear in the governing equations). Primary variables consist of branch across variables and cotree through variables ($\underline{\mathbf{x}}_B$, $\underline{\mathbf{y}}_C$). It follows that branch through variables and cotree across variables are secondary ($\underline{\mathbf{y}}_B$, $\underline{\mathbf{x}}_C$) [2]. Analysts choose the modeling variables used to describe a system simply by selecting a tree for the linear graph representation of that system [24].

2.4.3 Topological Equations

A *cutset* is set of edges that, when removed, divide the graph into two separate parts. A fundamental cutset (f-cutset) consists of a single branch and a unique set of chords [2, 47]. There is one unique f-cutset for each branch in the tree. Referring to Figure 2.6, if the edges that cross the dashed border were removed, then the nodes and edges contained within the dashed outline would be isolated from the rest of the graph. Since the edges crossing the dashed border consist of a single branch and two chords, they constitute an f-cutset.

The algebraic sum of the through variables associated with cutset edges is zero. For mechanical systems, the cutset equations correspond to dynamic equilibrium conditions, expressed in d'Alembert form. For electrical systems, cutset equations are a way of expressing Kirchoff's current law [2, 47]. A mathematical representation of the fundamental

cutset equations is:

$$\begin{bmatrix} \mathbf{1} & \mathbf{A_f} \end{bmatrix} \begin{bmatrix} \underline{\mathbf{y}}_B \\ \underline{\mathbf{y}}_C \end{bmatrix} = \mathbf{0} \tag{2.24}$$

Here, $\mathbf{1}$ is an identity matrix, while $\mathbf{A_f}$ is a matrix consisting of -1's, 1's and 0's indicating whether a chord is part of and negative, part of and positive, or not part of a given fundamental cutset. For a more detailed look at formulating $\mathbf{A_f}$, see [24].

The cutset equations may be rearranged in order to express the (secondary) branch through variables in terms of the (primary) chord through variables. Equation (2.25) accomplishes the task, and is known as the *chord transformations* [24, 47].

$$\underline{\mathbf{y}}_B = -\mathbf{A_f}\underline{\mathbf{y}}_C \tag{2.25}$$

A *circuit* is a set of edges that form a closed loop. A fundamental circuit (f-circuit) contains one chord and a unique set of branches. There is one f-circuit for each chord in the cotree [2, 47]. Referring to Figure 2.6, the edges following the dotted outline form a closed loop. Since these edges consist of a single chord and and two branches, they constitute an f-circuit.

The algebraic sum of the across variables associated with circuit edges is zero. For mechanical systems, circuit equations enforce vector loop closure. In electrical systems, circuit equations correspond to Kirchoff's voltage law [2]. A mathematical representation of the fundamental circuit equations is:

$$\begin{bmatrix} \mathbf{B_f} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \underline{\mathbf{x}}_B \\ \underline{\mathbf{x}}_C \end{bmatrix} = \mathbf{0} \tag{2.26}$$

In this case, $\mathbf{B_f}$ is a matrix consisting of -1's, 1's and 0's indicating whether a branch is part of and negative, part of and positive, or not part of a given fundamental circuit. For more details on formulating $\mathbf{B_f}$, see [24].

The circuit equations may be rearranged to express the (secondary) chord across variables in terms of the (primary) branch across variables. Equation (2.27) is known as the *branch transformations* [24, 47].

$$\underline{\mathbf{x}}_C = -\mathbf{B_f}\underline{\mathbf{x}}_B \tag{2.27}$$

It is interesting to note that, as a consequence of the definition of f-cutsets and f-circuits, the matrices $\mathbf{A_f}$ and $\mathbf{B_f}$ are orthogonal [2, 47]:

$$\mathbf{A_f} = -\mathbf{B_f}^T \tag{2.28}$$

2.4.4 Through and Across Space

An *across space* for an edge consists of an *ordered* set of unit vectors that span the space in which the edge allows its across variable to vary. Similarly, a *through space* is a set of *ordered* unit vectors that span the space in which the edge allows its through variable to vary. The order of unit vectors is important for finite 3D rotations, which are non-commutative [46]. Note that across and through spaces are generalizations of the purely mechanical concepts of motion and reaction spaces used by McPhee [24].

Through and across spaces may be dependent on derivative level; it is possible to use a different set of unit vectors to define the position-level across space of an edge than is used to define the velocity-level across space. The only restriction is that the spanned space reflect the physical nature of the component the edge is used to model [46].

The revolute joint example reinforces the concepts of through and across space. Referring to Table 2.3, the across space for the translational edge of the revolute joint component is null, meaning that the translational across variables ($\vec{r}_{T1}$, $\vec{v}_{T1}$) are not free to vary. In fact, they are zero, as stated in Table 2.2. The through space for the translational edge could be any three unit vectors that span a 3-dimensional space. This indicates that the translational through variable, $\vec{F}_{T1}$, is free to vary in all directions. The first edge in the rotational domain has a 3-dimensional across space, indicating that its across variables ($\vec{\theta}_{R1}$, $\vec{\omega}_{R1}$) are unrestricted in the values they may assume. The through space for this edge is null, since the through variable, $\vec{M}_{R1}$, is specified to be zero. The most interesting edge for a revolute joint is the second edge in the rotational domain. The across space for edge R2 is defined by the single unit vector, $\hat{u}$ since this is the only axis about which the across variables ($\vec{\theta}_{R2}$, $\vec{\omega}_{R2}$) are free to change. The through space may consist of any two unit vectors that span the two dimensional space which is orthogonal to $\hat{u}$. Moments about these axes may assume any values necessary to prohibit rotation about $\hat{n}_1$ and $\hat{n}_1$.

Edge	Derivative Level	Across Space	Through Space
T1	0th (position)	*null*	$\{\hat{i}_G, \hat{j}_G, \hat{k}_G\}$
	1st (velocity)	*null*	$\{\hat{i}_G, \hat{j}_G, \hat{k}_G\}$
R1	0th (orientation)	$\{\hat{i}_G, \hat{j}_G, \hat{k}_G\}$	*null*
	1st (angular velocity)	$\{\hat{i}_G, \hat{j}_G, \hat{k}_G\}$	*null*
R2	0th (orientation)	$\{\hat{u}\}$	$\{\hat{n}_1, \hat{n}_2\}$
	1st (angular velocity)	$\{\hat{u}\}$	$\{\hat{n}_1, \hat{n}_2\}$

Table 2.3: Through and Across Spaces for Revolute Joint Component

2.4.5 Formulation of System Equations

Combining topological equations (associated with the connectivity of nodes and edges) with terminal equations that describe the physical behavior of individual components yields a set of governing equations for the system. While many formulation procedures

can be used to assemble system equations based on a linear graph, we will only concern ourselves with an approach based on the principle of orthogonality, which is covered in detail by Schmitke and McPhee [46, 47] and is used by the DynaFlexPro software package. This formulation procedure is consistent across all domains, and may be used to generate equations for 1D electrical and hydraulic systems as well as 3D mechanical systems.

A mathematical representation of the principle of orthogonality is:

$$\sum_{i=1}^{edges} \underline{\mathbf{x}}_{[i]} \cdot \underline{\mathbf{y}}_{[i]} = \sum_{i=1}^{edges} \underline{\mathbf{y}}_{[i]} \cdot \underline{\mathbf{x}}_{[i]} = 0 \tag{2.29}$$

Equation (2.29) holds as long as the algebraic sum of across variables around any closed loop is zero, and the algebraic sum of through variables for any cutset is zero [2]. Often, the dot product of the through and across variables has units of power or energy. In this case, equation (2.29) simply states that the total energy in the system is conserved.

Andrews [2] and Schmitke [47] have shown that the principle of orthogonality also holds for virtual increments of the through and across variables that are consistent with the through and across spaces of their corresponding edge.

$$\sum_{i=1}^{edges} \underline{\delta\mathbf{x}}_{[i]} \cdot \underline{\mathbf{y}}_{[i]} = \sum_{i=1}^{edges} \underline{\delta\mathbf{y}}_{[i]} \cdot \underline{\mathbf{x}}_{[i]} = 0 \tag{2.30}$$

Substituting the branch and chord transformations into Equation (2.30) yields:

$$\sum_{i=1}^{edges} \left(\begin{bmatrix} \mathbf{I} \\ -\mathbf{B_f} \end{bmatrix} \underline{\delta\mathbf{x}}_B \right)_{[i]} \cdot \underline{\mathbf{y}}_{[i]} = 0 \tag{2.31}$$

$$\sum_{i=1}^{edges} \left(\begin{bmatrix} -\mathbf{A_f} \\ \mathbf{I} \end{bmatrix} \underline{\delta\mathbf{y}}_C \right)_{[i]} \cdot \underline{\mathbf{x}}_{[i]} = 0 \tag{2.32}$$

Rearranging these, and making use of equation(2.28), we obtain:

$$\sum_{i=1}^{edges} \left(\begin{bmatrix} \mathbf{I} & \mathbf{A_f} \end{bmatrix} \underline{\mathbf{y}}_{[i]} \right) \cdot \underline{\delta\mathbf{x}}_{B[i]} = 0 \tag{2.33}$$

$$\sum_{i=1}^{edges} \left(\begin{bmatrix} \mathbf{B_f} & \mathbf{I} \end{bmatrix} \underline{\mathbf{x}}_{[i]} \right) \cdot \underline{\delta\mathbf{y}}_{C[i]} = 0 \tag{2.34}$$

The first set of equations (2.33) represents the projection of fundamental cutset equations onto the across space for edges selected into the tree [47]. For mechanical systems, these become the dynamic equations [47]. It follows that motions consistent with the across space of edges selected into the tree will become modeling variables. By selecting different unit vectors to describe the position-level and velocity-level across space of an edge selected into the tree, an analyst can force the linear graph formulation to use generalized speeds which are not direct derivatives of position-level coordinates. With this

type of flexibility, an analyst can use velocities and angular velocities expressed in a body fixed reference frame as generalized speeds, as recommended by Mitiguy and Kane [29], while the position and orientation coordinates are expressed using any other unit vectors.

The second set of equations (2.34) represent the projection of fundamental circuit equations onto the through space for edges in the cotree [47]. For mechanical systems, these projected equations provide the mathematical constraints (if any) on the generalized coordinates $\mathbf{q}$ [24]. It follows that the reaction forces and moments used to enforce constraints will be consistent with the through space for edges selected into the cotree.

Schmitke and McPhee organized the entire set of system equations into a form that is valid for multi-domain systems [46], as shown in equations (2.35)-(2.37).

$$\mathbf{M}\dot{\mathbf{p}} + \mathbf{C}^T\mathbf{f} = \mathbf{b}(\mathbf{p}, \mathbf{q}, t) \tag{2.35}$$

$$\mathbf{\Phi}(\mathbf{q}, t) = \mathbf{0} \tag{2.36}$$

$$\dot{\mathbf{q}} = \mathbf{h}(\mathbf{p}, \mathbf{q}, t) \tag{2.37}$$

Equation (2.35) contains differential equations involving the generalized speeds $\mathbf{p}$ and is analogous to equation (2.10) presented by Haug [17]. The matrix $\mathbf{M}$ contains the coefficients of $\dot{\mathbf{p}}$ in the system's dynamic equations. The portion of $\mathbf{M}$ that corresponds to mechanical variables may be thought of as a mass matrix, while the portion that corresponds to electrical variables may be thought of as an inductance/capacitance matrix. The column matrix $\mathbf{f}$ contains variables related to the system's constraint equations — either Lagrange multipliers or mechanical constraint forces — and the matrix $\mathbf{C}^T$ gives the coefficients of $\mathbf{f}$ in the dynamic equations [46]. Equation (2.36) contains algebraic constraint equations involving the 0th derivative level modeling variables $\mathbf{q}$. Equation (2.37) describes kinematic transforms relating the derivatives of the coordinates $\dot{\mathbf{q}}$ to generalized speeds $\mathbf{p}$, coordinates $\mathbf{q}$, and time.

In order to numerically solve the equations they generated, Schmitke and McPhee replaced the algebraic constraint equations with their second derivative, giving a set of ordinary differential equations (ODEs) which were expressed in explicit first-order form:

$$\begin{bmatrix} \dot{\mathbf{p}} \\ \mathbf{f} \end{bmatrix} = \begin{bmatrix} \mathbf{M} & \mathbf{C^T} \\ \mathbf{\Phi}_q\mathbf{h_p} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b}(\mathbf{p}, \mathbf{q}, \mathbf{t}) \\ \mathbf{e}(\mathbf{p}, \mathbf{q}, \mathbf{t}) \end{bmatrix} \tag{2.38}$$

$$\dot{\mathbf{q}} = \mathbf{h}(\mathbf{p}, \mathbf{q}, t) \tag{2.39}$$

where $\mathbf{\Phi}_q$ represents the Jacobian of constraints with respect to coordinates $\mathbf{q}$, and $\mathbf{h_p}$ represents the Jacobian of the kinematic transforms with respect to the generalized speeds $\mathbf{p}$ [46].

It should be noted that Schmitke and McPhee never actually performed the symbolic matrix inverse that is implied by equation 2.38. They used a symbolic LU-factorization approach to obtain explicit equations for $\dot{\mathbf{p}}$ and $\mathbf{f}$ [46]. Simulation code could then be

assembled containing differential equations of the form $\dot{\mathbf{x}} = f(\mathbf{x})$, where the column matrix of state variables $\mathbf{x}$ contains both the generalized speeds and the position-level coordinates:

$$\mathbf{x} = \left\{ \begin{array}{c} \mathbf{p} \\ \mathbf{q} \end{array} \right\} \tag{2.40}$$

Rudolf made an attempt to incorporate tire forces and moments into a linear graph formulation of a vehicle dynamics model. His attempt failed due to equations that grew too large for the symbolic math software to handle [39]. This issue has recently been addressed, and an improved method for including tire forces and moments in a linear graph formulation is presented in Chapter 4 of this thesis.

Chapter 3

Pneumatic Tires in a Multibody Systems Context

3.1 Overview

An automobile is controlled by forces developed in just four small patches where the tires contact the road. The only other external forces acting on the vehicle are due to gravity and aerodynamics, aerodynamics having a secondary influence [43].

Pneumatic tires are complicated things. The carcass is constructed from a metal belt and polymer fibers woven at precise angles to the belt. The carcass is coated with a blend of rubber compounds that has intricate tread patterns carved into it. Due to the number of different materials involved, the non-linear and anisotropic nature of those materials, and the influence of uncontrolled factors like tread wear and inflation pressure, an accurate description of tire behavior is beyond the scope of classical mechanics and the theory of elasticity. It is possible to model the interaction of the different materials making up a tire using a finite element approach, but such an approach would be computationally expensive and prohibitively slow for full vehicle simulations.

Instead of pursuing complex theoretical solutions, most analysts will rely on physical tests to gather information about how a certain class or brand of tire behaves. A representative tire (or tires) is tested on a machine that measures force and moment components for various wheel orientations, forward speeds, spin rates, and tire/road normal forces. The measured data is set up in tabular form. This data may be interpolated during computer simulation in order to determine the forces and moments to apply to a vehicle model. Since the number of measured data points is likely to be in the millions, interpolating test data directly would be extremely inefficient and lead to slow simulations.

Alternatively, mathematical functions may be used to fit curves to the measured data. These are referred to as *tire models*. Tire models attempt to extract important characteristics from physical tests and present them as a set of equations that have low computational cost compared with direct interpolation of test data. Tire models are sometimes described

as being semi-empirical because their parameter values are obtained from test data, but the structure of the equations often stems from a theoretical description of tire behavior [34, 35].

Before going into any further detail on tire models, I will first present some basic terminology. The tire axis system recommended by the International Organization for Standardization (ISO) will be discussed and the quantities that are required as input to most tire models will be explained. The chapter concludes with a discussion of how forces and moments calculated by a tire model can be applied to a multibody vehicle model.

3.2 ISO Tire Axis System

The forces and moments calculated by tire models are resolved in the ISO tire axis system, which is shown in Figure 3.1 [15]. The longitudinal force F_x and overturning moment M_x are applied in the ISO X direction. The lateral force F_y and rolling resistance moment M_y are applied in the ISO Y direction. The normal force F_z and the aligning moment M_z are applied in the ISO Z direction. The normal force F_z is often referred to as a vertical force [6, 15, 32]. The International Organization for Standardization does not specifically address inclined roads, but in this thesis the ISO axis system will be applied to cases where the road may be inclined. The ISO Z direction is taken to be normal to the road plane (not necessarily vertical), and the term "normal force" is preferred over the term "vertical force" to describe F_z.

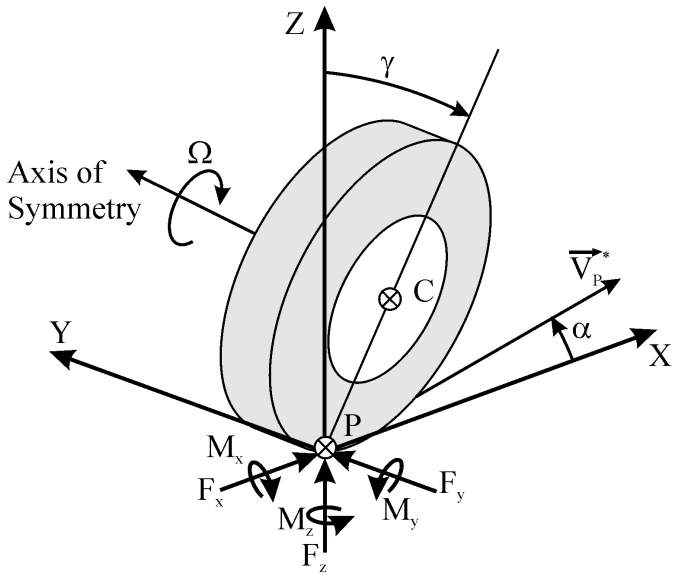


Figure 3.1: ISO Tire Axis System

Expressions for the ISO unit vectors can be obtained using simple vector operations. The ISO Z unit vector is normal to the road plane. The direction of the axis of tire

symmetry is used to find the ISO X unit vector: $\hat{u}_x$ points forward in the direction of wheel heading, and is perpendicular to both $\hat{u}_z$ and the axis of symmetry. The ISO Y unit vector defines the lateral direction for the tire and is chosen to make the coordinate system right-handed.

$$\hat{u}_z = \hat{n}_{road} \tag{3.1}$$

$$\hat{u}_x = \frac{\hat{u}_{SymAxis} \times \hat{u}_z}{|\hat{u}_{SymAxis} \times \hat{u}_z|} \tag{3.2}$$

$$\hat{u}_y = \hat{u}_z \times \hat{u}_x \tag{3.3}$$

In addition to giving direction to the forces and moments output from tire models, the ISO unit vectors are used in the expressions for tire model inputs.

3.3 Common Inputs to Tire Models

Most tire models require a subset of seven variables as input: normal force, inclination angle, forward speed, spin rate, effective rolling radius, longitudinal slip, and slip angle. These quantities depend on the position, velocity, orientation, and angular velocity of the tire with respect to the road.

Several authors have suggested using a linear spring-damper analogy to estimate the normal force between the tire and the road [6, 37, 43]. In equation (3.4), δ is the penetration of the undeformed tire into the road plane, as shown in Figure 3.2. The max function is used to ensure that the normal force between the tire and the road is always compressive, and is zero when the tire leaves the ground.

$$F_z = \max\left(k_z\delta + c_z\dot{\delta}, 0\right) \tag{3.4}$$

Equation (3.4) may be used whenever the road can be approximated, at least locally, by a plane. If the road contains small obstacles, close to the size of the tire contact patch, then a more in-depth analysis is required to calculate the normal force [6, 34].

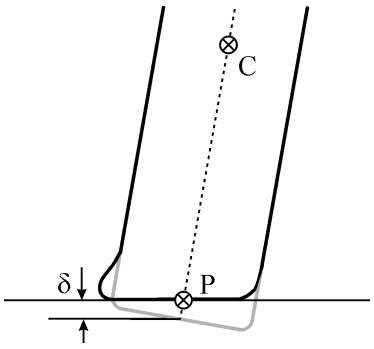


Figure 3.2: Penetration of Undeformed Tire into a Planar Road

The inclination angle γ is a measure of how much the wheel plane is tilted away from the XZ plane shown in Figure 3.1. The inclination angle may be found from geometric relations involving the axis of tire symmetry and the ISO unit vectors:

$$\gamma = \arcsin\left((\hat{u}_y \times \hat{u}_{SymAxis}) \cdot \hat{u}_x\right) \tag{3.5}$$

The tire's forward speed is easily determined by taking the component of the tire center velocity in the ISO X direction:

$$V_{Cx} = \vec{v}_C \cdot \hat{u}_x \tag{3.6}$$

The angular velocity of the tire can be resolved into 3 components. The components can be orthogonal, as in equation (3.7), or non-orthogonal, as in equation (3.8).

$$\vec{\omega}_C = \omega_x \hat{u}_x + \omega_{z(orthogonal)} \hat{u}_z + \omega_y \hat{u}_y \tag{3.7}$$

$$\vec{\omega}_C = \omega_x \hat{u}_x + \omega_{z(non-orthogonal)} \hat{u}_z + \Omega \hat{u}_{SymAxis} \tag{3.8}$$

where:

$$\omega_{z(non-orthogonal)} = \omega_{z(orthogonal)} - \Omega \sin(\gamma) \tag{3.9}$$

The component about the tire's axis of symmetry is of special significance. In order to obtain an expression for Ω, take the dot-product of equation (3.8) with $\hat{u}_{SymAxis}$:

$$\Omega = \vec{\omega}_C \cdot \hat{u}_{SymAxis} - \omega_{z(non-orthogonal)} \sin(\gamma) \tag{3.10}$$

The vector operations (dot product, cross-product, etc) performed by multibody dynamics software are coded with the assumption that the column matrix representation of a vector has three orthogonal components (i.e. vectors are resolved in orthogonal reference frames) [6, 17]. Substituting equation (3.9) into equation (3.10) results in an expression for Ω that is easier to use because it depends on quantities that are more readily available in multibody dynamics software:

$$\Omega = \frac{\vec{\omega}_C \cdot \hat{u}_{SymAxis} - \omega_{z(orthogonal)} \sin(\gamma)}{\cos^2(\gamma)} \tag{3.11}$$

Ω is often referred to in the literature as "spin rate" [5, 6, 43] or "speed of rotation" [34, 35]. In a strict mathematical sense, Ω is the component of the tire's angular velocity about its symmetry axis. It is entirely possible to "spin" or "rotate" a tire about an axis other than its axis of symmetry. However, for the remainder of this thesis, I will adopt the common practice of referring to Ω as spin rate.

A free rolling wheel is defined as a loaded wheel subject to neither braking nor driving torques [15]. Although no external torques are applied, the tire will still be subject to its own rolling resistance and will eventually slow down. ISO defines Ω_0 as the rotational speed of a free rolling wheel at zero inclination angle and zero slip angle (slip angle will be

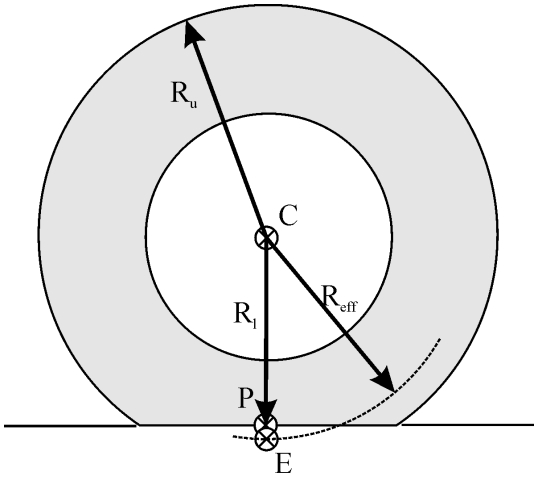


Figure 3.3: Important Tire Radii

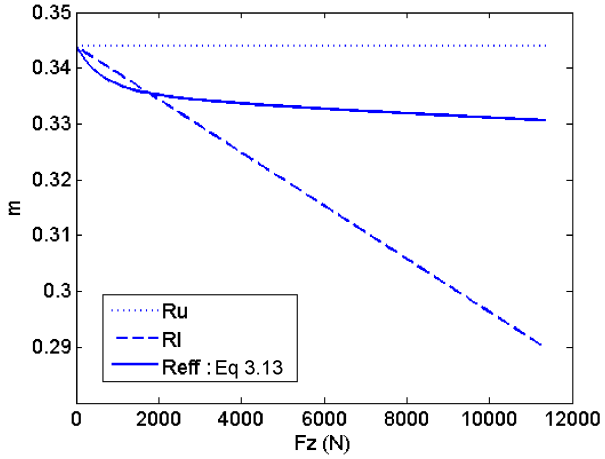


Figure 3.4: Three Methods for Estimating Effective Rolling Radius

defined later in this section). Ω_0 can be determined experimentally by allowing a wheel to roll freely and measuring its rate of rotation [15].

The effective rolling radius R_{eff} is closely related to Ω_0. It is the radius of an imaginary rigid disk that rolls without slipping and has the same free rolling properties as the actual tire. That is to say, $R_{eff} = V_{Cx}/\Omega_0$, where V_{Cx} and Ω_0 are measured experimentally. Because tires are not rigid and there is always some degree of slip and distortion taking place within the contact patch, the effective rolling radius is not equal to the actual radius of the tire. The effective rolling radius usually lies somewhere between the unloaded radius and loaded radius, as shown in Figure 3.3.

The effective rolling radius is dependent on normal force, and may also depend on inclination angle and tire slip [34]. The developers of MSC.ADAMS have proposed a non-linear formula for estimating the effective rolling radius as a function of normal load [32].

Equation 3.13 is referred to in this thesis as the ADAMS/Pacejka method for estimating effective rolling radius. The method was never actually published by Pacejka, but in the ADAMS 2005 software, it is used exclusively in conjunction with Pacejka's magic formula tire models [32].

$$\rho = R_u - R_l \tag{3.12}$$

$$R_{eff} = R_u - \rho_{Fz0}\left(D\arctan\left(B\frac{\rho}{\rho_{Fz0}}\right) + F\frac{\rho}{\rho_{Fz0}}\right) \tag{3.13}$$

In equation (3.13), ρ is the tire deflection, ρ_{Fzo} is the tire deflection at some nominal vertical force, and B, D, and F are parameters that describe the tire [32]. Values of $\rho_{Fzo} = 0.0231$, $B = 8.4$, $D = 0.27$, $F = 0.07$, and a tire vertical stiffness of 210000 N/m were used to create the plot shown in Figure 3.4. Note that the loaded radius decreases linearly with increasing load, as predicted by the linear spring-damper analogy, while the effective rolling radius levels off at high loads.

The ISO definition of longitudinal slip is presented in equation (3.14). It can be seen that the longitudinal slip S is a measure of how fast the tire is spinning compared to the free rolling condition. During braking, the wheel spins slower than the free rolling condition and longitudinal slip is negative. During acceleration, the wheel spins faster than the free rolling condition and longitudinal slip is positive [15].

$$S = \frac{\Omega - \Omega_0}{\Omega_0} \tag{3.14}$$

Substituting $\Omega_0 = V_{Cx}/R_{eff}$ into equation (3.14) results in another popular expression for longitudinal slip:

$$S = \frac{\Omega R_{eff} - V_{Cx}}{V_{Cx}} \tag{3.15}$$

It should be noted that since Ω_0 is only defined for a wheel rolling with zero inclination angle, equations (3.14) and (3.15) are only valid when the inclination angle is zero.

Consider a point E, located along the vector $\vec{r}_{P/C}$ a distance R_{eff} away from the tire center C. In practice, point E is very close to the center of tire contact P, but does not exactly coincide with it unless $R_{eff} = Rl$. The numerator of equation (3.15) gives the negative X component of the velocity of point E. Therefore, equation (3.15) can be rewritten as:

$$S = \frac{-V_{Ex}}{V_{Cx}} \tag{3.16}$$

which describes the longitudinal slip of a tire with zero inclination angle. To be applicable to inclined wheels, the expression for S must be written in terms of the velocity at point E only. For this, it is necessary to consider an imaginary body that translates, yaws, and tilts with the actual tire, but does not spin with it. Postiau et al. referred to such a body

as a geometric wheel, and their terminology will be adopted here [37]. A star * indicates the velocity of a point on the geometric wheel. For a tire rolling with zero inclination angle, $V_{Cx} = V^*_{Ex}$. Substituting this into equation (3.16) gives the following expression for longitudinal slip:

$$S = \frac{-V_{Ex}}{V^*_{Ex}} \tag{3.17}$$

where:

$$V_{Ex} = \left(\vec{v}_C + \omega_C \times \vec{r}_{E/C}\right) \cdot \hat{u}_x \tag{3.18}$$

$$V^*_{Ex} = V_{Ex} + \Omega R_{eff} \tag{3.19}$$

Equation (3.17) is recommended by Pacejka [34] as well as Blundell and Harty [6]. It is equally valid for inclined and non-inclined tires. One small modification remains to be made, which will allow the correct sign of longitudinal force to be calculated by a tire model. The sign of longitudinal slip S and longitudinal force F_x should always oppose the sign of V_{Ex}. To ensure this, V^*_{Ex} is replaced with its absolute value in equation (3.17):

$$S = \frac{-V_{Ex}}{\left|V^*_{Ex}\right|} \tag{3.20}$$

ISO defines slip angle α as the angle from the X axis to the tangent of the trajectory of the center of tire contact [15], displayed graphically in Figure 3.1. The center of tire contact, P, is not a fixed point on the tire; to follow its trajectory, wheel spin motions must be ignored, and the expression for slip angle written in terms of the velocity of point P on a geometric wheel.

$$\alpha = \arctan\left(\frac{V^*_{Py}}{\left|V^*_{Px}\right|}\right) = \arctan\left(\frac{V_{Py}}{\left|V^*_{Px}\right|}\right) \tag{3.21}$$

Note that $V_{Py} = V^*_{Py}$ because wheel spin does not affect the Y component of velocity for point P. The absolute value of V^*_{Px} is used in the denominator of the slip angle expression in order to ensure that slip angle always has the correct sign, whether the tire is moving forward or in reverse.

Tire forces do not develop instantly, but build as the tire rolls. Two methods are commonly used to model dynamic lag in tire forces:

1. Use a tire model with the dynamics built-in
2. Use a steady-state tire model that takes delayed longitudinal slip and slip angle as input

The second approach is preferable since it allows the analyst to select an appropriate tire model for his application independent of the method used to introduce lag [35, 43].

The accepted approach for introducing lag in tire slip variables is to use first-order differential equations involving a parameter called relaxation length, which is similar to a time constant, except that it has units of length [5, 34, 43]. The equations proposed by Bernard and Clover [5] and modified by Pacejka [34] are shown in equations (3.22) - (3.25). Here, B_{long} is the longitudinal relaxation length and B_{lat} is the lateral relaxation length.

$$\frac{dq_1}{dt} = \frac{-V_{Ex}}{B_{long}} - \frac{q_1 \left|V_{Ex}^*\right|}{B_{long}} \tag{3.22}$$

$$\frac{dq_2}{dt} = \frac{V_{Py}}{B_{lat}} - \frac{q_2 \left|V_{Px}^*\right|}{B_{lat}} \tag{3.23}$$

$$S = q_1 \tag{3.24}$$

$$\alpha = \arctan(q_2) \tag{3.25}$$

For steady-state conditions, the delayed slip equations (3.22) - (3.25) reduce to the kinematic slip equations (3.20) and (3.21). The kinematic slip equations cannot be used at zero velocity conditions ($V_x = 0$) because V_x appears in the denominator. The delayed slip equations can be used at zero velocity conditions. This is an important property for driving simulators, where the driver expects to be able to bring the vehicle to a stop, and potentially start it again, without the simulator reporting a divide-by-zero error [5].

Bernard and Clover [5] used constant relaxation lengths in their work, but constant relaxation lengths may not provide enough fidelity for studies in which tire transients are very important. Pacejka presents physical test results that indicate relaxation lengths can vary significantly with normal force F_z, longitudinal slip S, and lateral slip α [34]. The developers of MSC.ADAMS proposed a method for calculating the dependency of relaxation lengths on normal force and inclination angle [32]. ADAMS refers to this as the "stretched string" method of modeling tire transient behavior because its derivation considers the behavior of a string that follows the tire circumference and is kept under a certain pretension by a uniform radial force distribution that is comparable with inflation pressure in real tires [34]. The ADAMS 2005 stretched string equations for relaxation length are presented in equations (3.26 - 3.28), where F_{z0} is the nominal normal force between the tire and the road, R_0 is the unloaded radius, and $\lambda_{\sigma\kappa}$, $\lambda_{\sigma\alpha}$, $\lambda_{F_{z0}}$, P_{Tx1}, P_{Tx2}, P_{Tx3}, P_{Ty1}, P_{Ty2}, and P_{Ky3} are parameters that describe the transient behavior of the tire, as determined from physical test.

$$dF_z = \frac{F_z - F_{z0}}{F_{z0}} \tag{3.26}$$

$$B_{long} = \lambda_{\sigma\kappa} R_0 \frac{F_z}{F_{z0}} \left(P_{Tx1} + P_{Tx2} dF_z\right) e^{(P_{Tx3} dF_z)} \tag{3.27}$$

$$B_{lat} = \lambda_{\sigma\alpha} P_{Ty1} F_{z0} \sin\left(2 \arctan\left(\frac{F_z}{P_{Ty2} F_{z0} \lambda_{F_{z0}}}\right)\right) \left(1 - P_{Ky3} \left|\gamma_y\right|\right) R_0 \lambda_{F_{z0}} \tag{3.28}$$

It is a relatively simple matter to replace the constant relaxation lengths in equations (3.22) and (3.23) with values that are functions of normal force and the tire's kinematic state, whether these values are determined using the ADAMS stretched string equations or another method.

While the delayed slip equations may be used at low speed and zero speed conditions, Pacejka points out that their accuracy is compromised at low speeds [34]. At small values of V_{Ex}^* and V_{Px}^*, equations (3.22) and (3.23) act as integrators of V_{Ex} and V_{Py}. This could give rise to very large values of q_1 and q_2, and consequently large values of S and α. To prevent S and α from taking on unreasonably large values, Pacejka proposed setting $\frac{dq1}{dt}$ and $\frac{dq2}{dt}$ equal to zero when the forward speed was under a certain threshold and the current magnitudes of S and α were over a certain threshold and growing. His limiting algorithm is summarized in Figure 3.5.

if:	$\lvert V_{Ex}^* \rvert < V_{low}$ and $\lvert S \rvert > S_{high}$ and $\left(\frac{-V_{Ex}}{B_{long}} - \frac{S\lvert V_{Ex}^* \rvert}{B_{long}} \right)(S) > 0$
then:	$\frac{dq_1}{dt} = 0$
else:	equation 3.22 apples
if:	$\lvert V_{Px}^* \rvert < V_{low}$ and $\lvert \alpha \rvert > \alpha_{high}$ and $\left(\frac{V_{Py}}{B_{lat}} - \frac{\tan(\alpha)\lvert V_{Px}^* \rvert}{B_{lat}} \right)(\alpha) > 0$
then:	$\frac{dq_2}{dt} = 0$
else:	equation 3.23 apples

Figure 3.5: Pacejka's Algorithm for Limiting Derivatives of Tire Slip States

3.4 Tire Models

A detailed discussion of tire carcass construction and the mechanisms by which stresses develop within the tire contact patch is beyond the scope of this thesis, but is covered in several books on the topic [6, 16, 56]. A few qualitative statements concerning tire force and moment generation are presented below. These statements hold true for the majority of pneumatic tires and are intended to give the reader an idea of the type of behavior that tire models attempt to capture.

- at low slip values, longitudinal force F_x is roughly proportional to longitudinal slip and lateral force F_y is roughly proportional to slip angle
- at high slip, lateral and longitudinal forces become saturated, their maximum values being determined by the coefficient of friction between the tire and road
- F_x and F_y increase with increasing normal force, but the dependency is non-linear

- a negative inclination angle tends to increase lateral force, F_y, but any non-zero inclination angle decreases the area of the tire contact patch, and therefore has a diminishing effect on other tire forces and moments
- in combined slip situations (both lateral and longitudinal slip are present), the value of F_x will be lower than it would have been with pure longitudinal slip, and the value of F_y will be lower than it would have been with pure lateral slip
- when a tire is steered, an aligning moment is produced which tends to bring the tire back to the straight ahead position. M_z depends non-linearly on slip angle, with longitudinal slip, normal force, and inclination angle having secondary effects

Many authors have proposed tire models for vehicle acceleration, braking, and handling studies. A review of some of these models is given by Pacejka and Sharp [35], and also by Blundell and Harty [6]. Tire models vary greatly in their complexity and the number of parameters that must be determined from physical tests. Selecting an appropriate tire model is a compromise between availability of test data, accuracy of the curve fit, and the computational cost associated with using the tire model as part of a larger vehicle model. The optimal tire model is often not the most sophisticated one, but the one whose degree of refinement is just enough to predict vehicle behavior to the desired accuracy.

3.4.1 The Fiala Tire Model

The Fiala tire model requires only 6 parameters to describe a tire, as summarized in Table 3.1. These parameters are directly related to the physical properties of the tire.

D_2	width of the tire (units – length)
C_S	longitudinal stiffness; the slope of the F_x vs. S curve at $S = 0$ (units – force)
C_α	lateral stiffness; the slope of the F_y vs. α curve at $\alpha = 0$ (units – force/radian)
C_r	coefficient of rolling resistance (units – length)
μ_0	peak coefficient of friction between the tire and road
μ_1	sliding coefficient of friction between the tire and road

Table 3.1: Fiala Tire Model Parameters

Despite the advantage of a simple parameter set, the Fiala model has several limitations:

- The model cannot accurately represent combined slip situations
- The effect of inclination angle on forces and moments is ignored
- The variation of lateral stiffness and longitudinal stiffness with normal load is not considered

- The force and moment curves cannot be offset from the origin (zero slip always implies zero F_x, F_y. and M_z)
- Overturning moment is not considered

The equations for the Fiala tire model are presented below so that the reader can understand how the tire model takes normal force (F_z) and kinematic inputs (S, α, Ω) and returns the forces and moments at the tire contact patch.

The first step to implementing the Fiala tire model is to calculate intermediate terms that are used repeatedly throughout the force and moment expressions. In what follows, $S_{L\alpha}$ is a the comprehensive slip ratio — a combination of longitudinal and lateral slip parameters, μ is the current value of coefficient of friction and is somewhere between the maximum (μ_0) and minimum (μ_1) values, and H is an intermediate parameter associated with slip angle.

$$S_{L\alpha} = \sqrt{S^2 + \tan^2(\alpha)} \tag{3.29}$$

$$\mu = \mu_0 - S_{L\alpha}(\mu_0 - \mu_1) \tag{3.30}$$

$$H = 1 - \frac{C_\alpha \left|\tan(\alpha)\right|}{3\mu \left|F_z\right|} \tag{3.31}$$

The force and moment values returned by the Fiala tire model are:

$$F_x = \begin{cases} C_S S & \text{if } |S| < \left|\frac{\mu F_z}{2C_S}\right| \\ -\text{sign}(S)\left(\mu F_z - \left(\frac{(\mu F_z)^2}{4|S|C_S}\right)\right) & \text{otherwise} \end{cases} \tag{3.32}$$

$$F_y = \begin{cases} -\mu \left|F_z\right| \left(1 - H^3\right) \text{sign}(\alpha) & \text{if } |\alpha| < \arctan\left(\left|\frac{3\mu F_z}{2_\alpha}\right|\right) \\ -\mu \left|F_z\right| \text{sign}(\alpha) & \text{otherwise} \end{cases} \tag{3.33}$$

$$M_x = 0 \tag{3.34}$$

$$M_y = \begin{cases} C_r F_z & \text{if } \Omega < 0 \\ -C_r F_z & \text{otherwise} \end{cases} \tag{3.35}$$

$$M_z = \begin{cases} \mu F_z D_2 \left(1 - H\right) H^3 \text{sign}(\alpha) & \text{if } |\alpha| < \arctan\left(\left|\frac{3\mu F_z}{2C_\alpha}\right|\right) \\ 0 & \text{otherwise} \end{cases} \tag{3.36}$$

Graphs showing typical outputs of the Fiala tire model are presented in Figures 3.7 - 3.9. Results for the relatively simple Fiala tire model are plotted with results from a more complete Pacejka 2002 tire model.

Note that the Fiala model has several "if" constructs that make it impossible to evaluate tire forces and moments unless numeric values are supplied for the tire model inputs. For instance, there are two possible expressions for longitudinal force F_x, as shown

in equation 3.32. The expression to be used depends on the numerical value of S. "If" constructs are not unique to the Fiala tire model; they appear in many other popular tire models. In a multibody dynamics context, expressions for tire forces and moments cannot generally be evaluated during equation *formulation*. They must be evaluated during *simulation* when (at a given time step) the numeric values for tire model inputs are known. This is an important fact to keep in mind when assembling simulation code for systems containing pneumatic tires — a topic that will be covered further in Chapter 4.

3.4.2 The Magic Formula Tire Model

Magic formula tire models are the most accurate and widely accepted tire models for use in flat road handling studies [6]. They make frequent use of compound trigonometric formulas that result in an uncanny fit to measured force and moment curves. The magic formula tire model is undergoing continual development and several different versions have been proposed over the years. The first version, introduced by Bakker et al. in 1986, only calculated F_x as a function of S, and F_y and M_z as a functions of α. It did not consider rolling resistance (M_y), overturning moment (M_x), combined slip, or the effects of inclination angle [3]. A more recent version, described by Pacejka in his 2002 book, calculates all of the forces and moments generated at the tire contact patch and addresses their (non-linear) dependence on all of the tire model inputs discussed earlier in this chapter [34].

The magic formula used to calculate longitudinal force, lateral force and aligning moment is:

$$x = X + S_h \tag{3.37}$$

$$y(x) = Dsin\left(C\arctan\left(Bx - E\left(Bx - \arctan(Bx)\right)\right)\right) \tag{3.38}$$

$$Y(X) = y(x) + S_v \tag{3.39}$$

where Y is the quantity to be determined (one of F_x, F_y, or M_z) and X is the independent variable of interest (S or α). Horizontal and vertical shifts are represented by S_h and S_v respectively, and are used to account for cases where the curve does not pass through the origin. It is well known that inclination angles cause a vertical shift of the lateral force curve, so for $Y = F_y$, S_v would be a function of γ.

The other coefficients in equation (3.38) are best described with reference to Figure 3.6. D controls the peak value, C is a shape factor that controls stretching in the the x direction, and B is referred to as a stiffness factor since it affects the slope at the origin. The actual slope is given by BCD. E is a curvature factor that affects the position, x_m, at which the peak value occurs [6].

The main requirement for modeling tires using the magic formula is to determine appropriate values for the coefficients S_h, S_v, B, C, D, and E for each force and moment curve given the normal force and the tire's kinematic inputs. This process can be

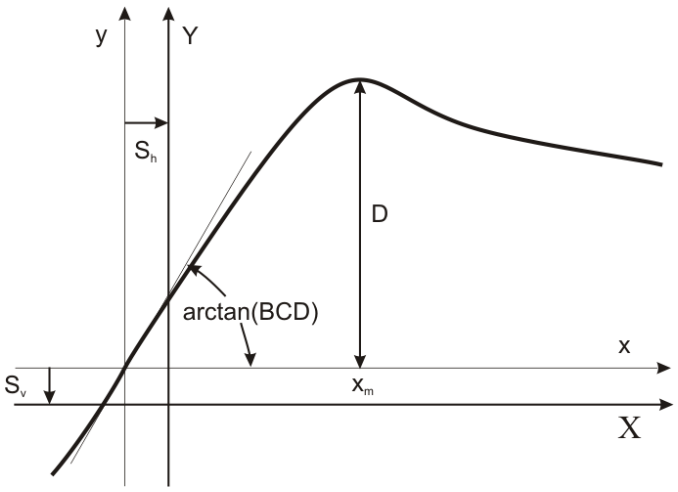


Figure 3.6: Coefficients used in the Magic Formula

extremely involved for complex tire models. The magic formula tire model described in Pacejka's 2002 book needs 117 parameters to describe a single tire [34]. The equations used by Pacejka are too lengthy to present here, but some graphs are presented that show the properties of an example tire as described by Pacejka's model.

The Fiala tire model parameters used to create Figures 3.7 - 3.9 can be found in Table A.7, and the Pacejka 2002 tire model parameters used to create these figures can be found in Table A.4. The nominal condition is defined as the state in which the tire operates at its nominal load ($F_z = 5900N$), at inclination angle equal to zero ($\gamma = 0$), and with either pure longitudinal slip ($\alpha = 0$) or pure lateral slip ($S = 0$). Any changes from the nominal condition are indicated on the graphs. It can be seen that the Fiala model closely matches the Pacejka 2002 model for nominal conditions. For cases involving inclination angles, combined slip, or off-nominal normal loads, the Fiala model produces different results than the Pacejka 2002 model. The Pacejka 2002 model can be assumed more accurate than the Fiala model because it has a greater reliance on physical test data (117 parameters determined by test compared with 6 for the Fiala model) and also because it has achieved more widespread acceptance in the literature.

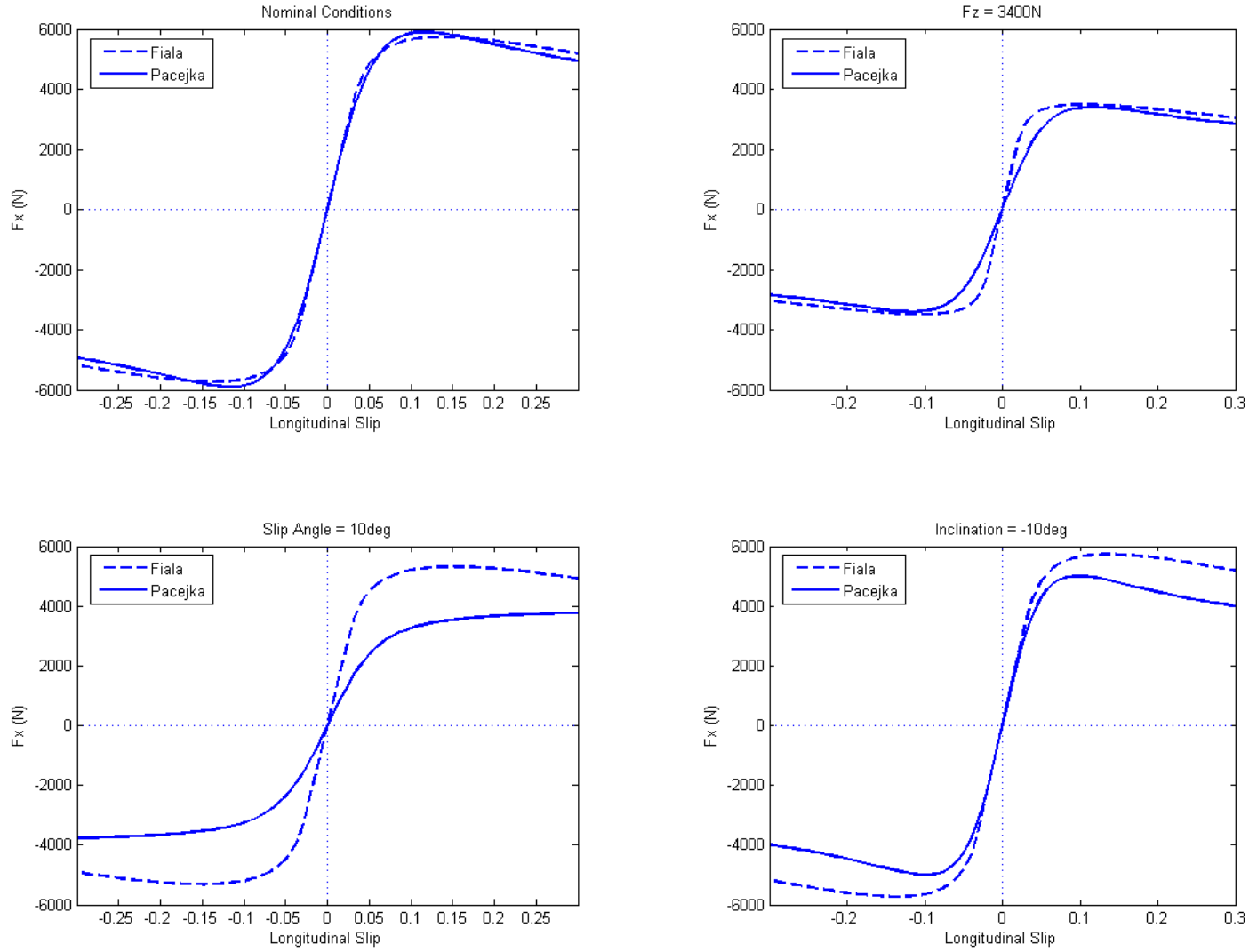


Figure 3.7: Longitudinal Force Curves for Fiala and Pacejka 2002 Tire Models

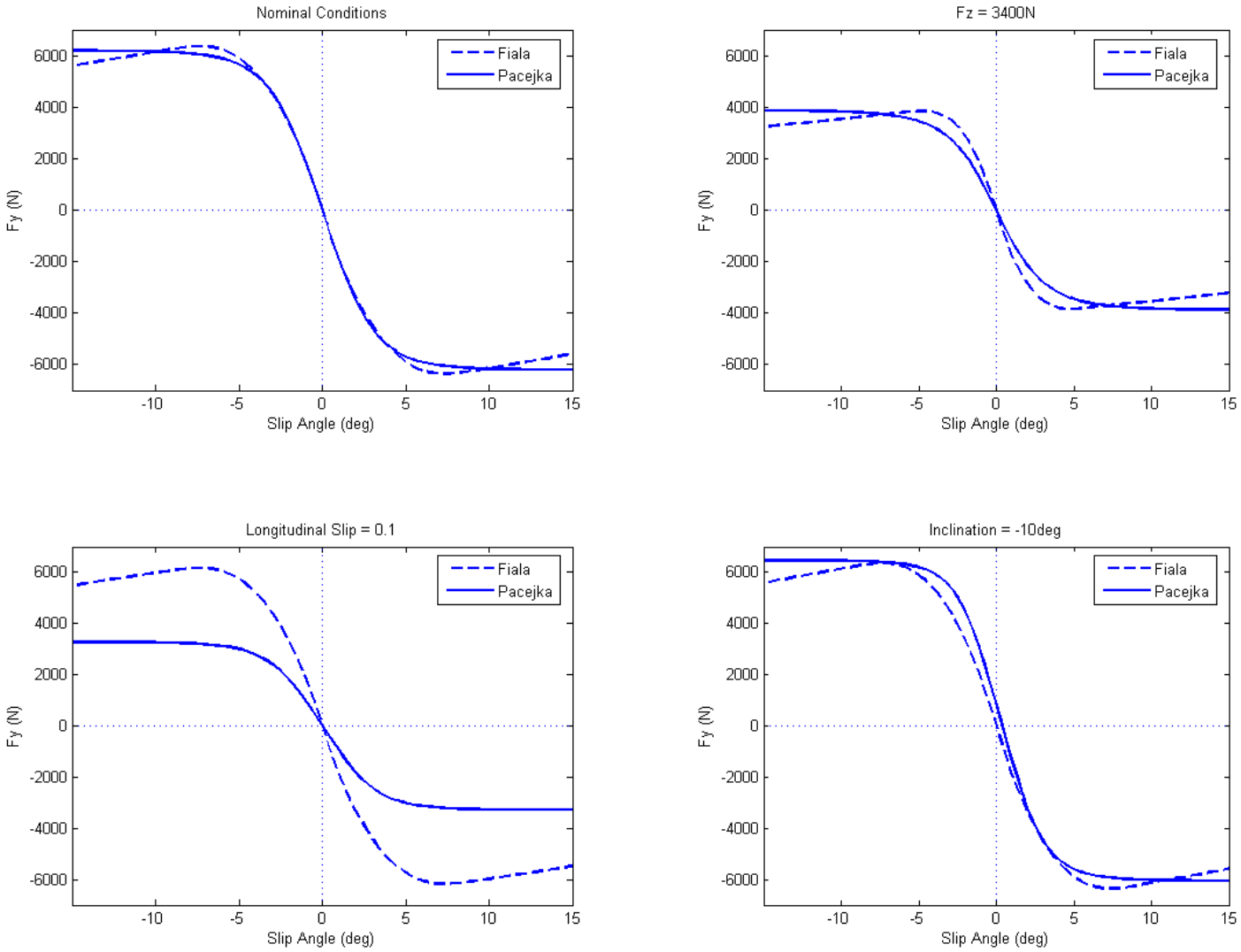


Figure 3.8: Lateral Force Curves for Fiala and Pacejka 2002 Tire Models

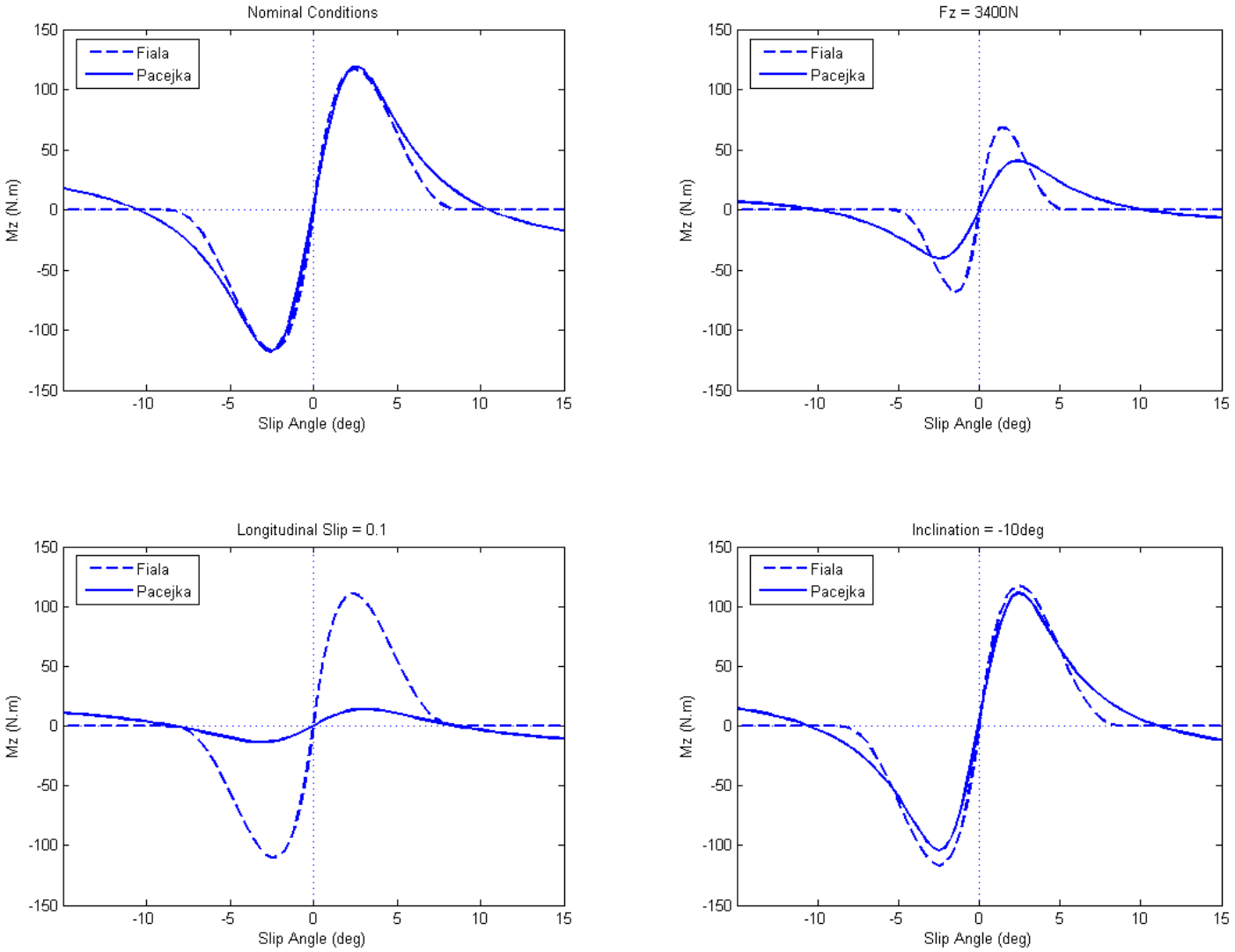


Figure 3.9: Aligning Moment Curves for Fiala and Pacejka 2002 Tire Models

3.5 Applying Tire Forces and Moments to a Multibody Model

Tire models calculate the forces and moments that act at the contact patch between the tire and the road (point P in Figure 3.10). However, since P is not fixed to the road or to the tire, it is inconvenient to apply fores at this point in a multibody model.

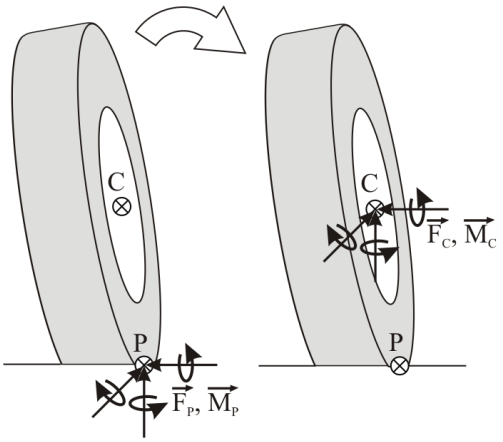


Figure 3.10: Tire Forces Applied to the Wheel Center

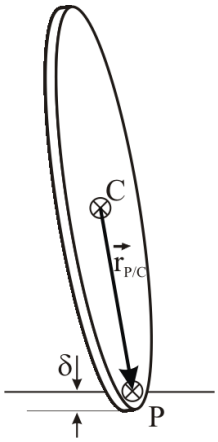


Figure 3.11: Planar Disk Approximation Used to Locate Point P

It is a common practice to apply forces and moments to the tire center (point C in Figure 3.10) [6, 37, 43, 54]. Moving the point of force application requires the introduction of additional moments that are dependent on the vector from the tire center to the tire contact patch, $\vec{r}_{P/C}$:

$$\vec{F}_C = \vec{F}_P \tag{3.40}$$

$$\vec{M}_C = \vec{M}_P + \vec{r}_{P/C} \times \vec{F}_P \tag{3.41}$$

The shape of the tire must be known in order to determine $\vec{r}_{P/C}$. Blundell and Harty assume a toroidal tire shape in order to locate the center of the contact patch with respect to the wheel center [6], while Postiau and Sayers both use a planar disk assumption [37, 43]. In practice, both methods give very similar results for $\vec{r}_{P/C}$. The planar disk approximation, depicted in Figure 3.11, was used for the present work because of its ease of implementation.

For the same reasons that tire forces and moments are applied to point C, which is a fixed point on the tire, they are reacted at at some point Q, which is a fixed point on the road. In order for the imagined system shown in Figure 3.12, with reaction forces applied to the road at point Q, to be equivalent to the actual system, with reaction forces applied to the road at point P, extra moment terms must be introduced as described in equations (3.42) and (3.43).

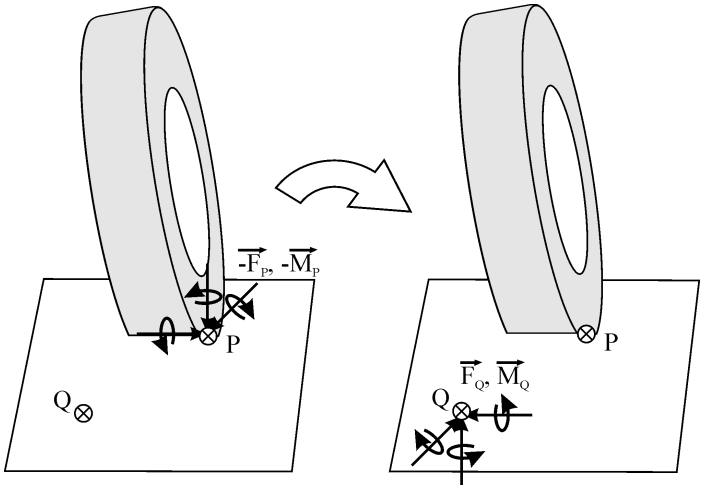


Figure 3.12: Tire Forces Reacted at Road Reference Frame Q

$$\vec{F}_Q = -\vec{F}_P \tag{3.42}$$

$$\vec{M}_Q = -\vec{M}_P + \vec{r}_{P/Q} \times \left(-\vec{F}_P\right) \tag{3.43}$$

Chapter 4

The Pneumatic Tire as a DynaFlexPro Component

4.1 Overview

The three steps of the modeling and simulation process were described at the beginning of this thesis. It is assumed that the first step, deciding how to represent an actual system as a connection of idealized components (rigid bodies, joints, etc.) has been completed.

The second step involves generating a set of equations that describe system behavior. In the context of this thesis, equation formulation is accomplished using linear graph theory and the principle of orthogonality. These concepts were introduced in Chapter 2. This chapter describes how they apply to vehicle systems containing tires.

Before moving on to the third step — solving equations numerically — the system equations must be arranged as part of simulation code that calculates derivatives of state variables given the current values of state variables as input. For systems containing tires, this involves more than simply writing differential equations in first order form. Provisions have to be made for calculating tire-specific intermediate variables and for calling external functions. These issues are dealt with using a new DynaFlexPro component model template; in addition to the topological and edge-level information used during the linear graph formulation process, the tire component model can contain rules for assembling blocks of simulation code that are not directly related to graph theory, but are essential for simulating vehicle systems. Symbolic computing techniques are used to control the size of tire intermediate expressions and to reduce the computational cost of the simulation code that is generated.

4.2 Topological and Edge-Level Information

A pneumatic tire can be represented by a linear graph with two parts: one in the mechanical translational domain, and one in the mechanical rotational domain, as shown in

Figure 4.1. The node labeled "Ground" represents an inertial reference frame, the node labeled "C" represents a reference frame fixed to the tire and located at the tire center, and the node labeled "Q" represents a reference frame attached to the road and oriented such that the z axis is normal to the road plane. One of the main assumptions made for this work is that the road is planar. The dotted edges are not part of the tire component model. They represent the method used to define the kinematics of the road reference frame with respect to the inertial frame; in practice, they may be a single edge or a collection of edges. When using a tire component as part of a larger model, the kinematics of the road reference frame must be defined in some way.

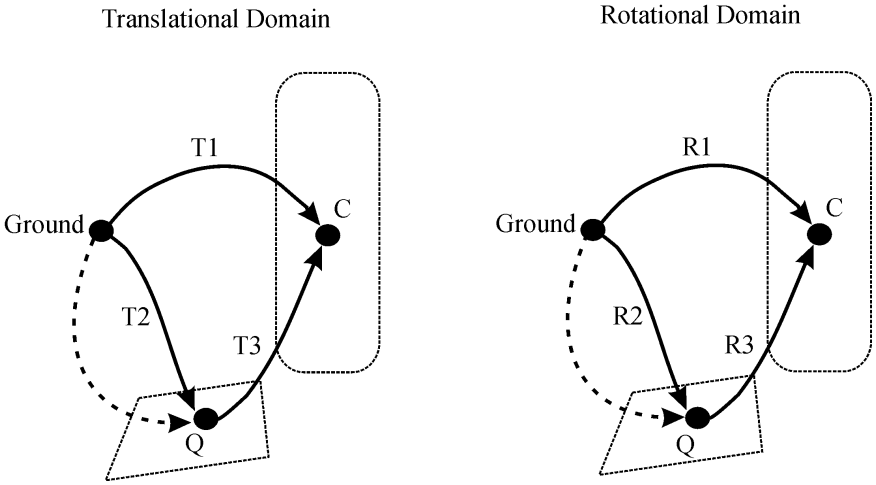


Figure 4.1: Linear Graph Representation of Tire Component Model

Edge T1 represents the mass of the tire and its weight in a local gravitational field. Edge T2 measures the location of the road reference frame with respect to the Ground frame. Edge T3 represents forces applied to the tire center as a result of tire/road interactions.

Edge R1 represents the rotational inertia of the the tire. Edge R2 measures the orientation of the road plane with respect to the Ground frame. Edge R3 represents moments applied to the tire center as a result of tire/road interactions.

Recall that every edge in a linear graph component has terminal equations that relate its through and across variables. Further insight into the physics of the tire component model can be gained by inspecting its terminal equations, which are presented in Table 4.1.

The terminal equations for edges T1 and R1 express Newton-Euler equations for a rigid body in D'Alembert from. Here m is the mass of the tire, $\mathbf{J}$ is its inertia tensor, and $\vec{g}$ is the local acceleration due to gravity. The rigid body assumption is valid as long as the deformation of the tire does not significantly affect its center of mass location or its rotational inertia.

Following common practice, tire forces and moments, which naturally occur at the

Edge	Terminal Equation (same at all derivative levels)
T1	$\vec{F}_{T1} = -m\dot{\vec{v}}_{T1} + m\vec{g}$
T2	$\vec{F}_{T2} = \vec{0}$
T3	$\vec{F}_{T3} = \vec{F}_P$
R1	$\vec{M}_{R1} = -\mathbf{J}\dot{\vec{\omega}}_{R1} - \vec{\omega}_{R1} \times (\mathbf{J}\vec{\omega}_{R1})$
R2	$\vec{M}_{R2} = \vec{r}_{Q/C} \times \left(\vec{F}_P\right)$
R3	$\vec{M}_{R3} = \vec{M}_P + \vec{r}_{P/C} \times \vec{F}_P$

Table 4.1: Terminal Equations for Tire Component Model

tire contact patch, are applied to the tire center. The terminal equations for edges T3 and R3 come directly from equations (3.40) and (3.41).

As discussed in Section 3.5, tire forces and moments will be reacted at point Q on the road plane. In order to correctly represent the physics of tire/road interaction, the moment applied to node Q must be different than the moment applied to node C. A linear graph edge can only have one through variable associated with it per derivative level. Therefore, edge R3 cannot adequately describe both the moment applied to the tire center and the moment applied to the road reference frame; in this work, edge R3 represents moments applied to the tire center. In order to ensure that the correct total moment is applied to the road reference frame, an additional moment ($\vec{r}_{Q/C} \times \vec{F}_P$) is applied via edge R2. Consider the algebraic sum of moments applied *by the tire component* to node Q (not including moments that the dotted edge might apply):

$$\begin{aligned} \sum \vec{M}_{\text{applied by tire component to Q}} &= \vec{M}_{R2} - \vec{M}_{R3} \\ &= \vec{r}_{Q/C} \times \vec{F}_P - \left(\vec{M}_P + \vec{r}_{P/C} \times \vec{F}_P\right) \\ &= -\vec{M}_P + \left(\vec{r}_{P/C} - \vec{r}_{Q/C}\right) \times \left(-\vec{F}_P\right) \\ &= -\vec{M}_P + \vec{r}_{P/Q} \times \left(-\vec{F}_P\right) \end{aligned} \tag{4.1}$$

Equation (4.1) is equivalent to equation (3.43). Therefore, the forces and moments listed in Table 4.1 are equivalent to the actual loading case where forces and moments act between the tire and road at the contact point P. Instead of applying forces and moments at point P, they are applied at the tire center C and reacted at the road reference frame Q.

Every edge in the tire component has a full three-dimensional across space and a *null* through space; all of the across variables (positions, velocities, orientations, angular velocities) are free to assume any value, but through variables (forces and moments) are not free to vary. If the across variables are known, then the through variables are given by the equations laid out in Table 4.1. Although specific unit vectors are listed in Table

Edge	Derivative Level	Across Space	Through Space
T1	0th (position)	$\{\hat{i}_G, \hat{j}_G, \hat{k}_G\}$	*null*
	1st (velocity)	$\{\hat{i}_C, \hat{j}_C, \hat{k}_C\}$	*null*
T2	0th (position)	$\{\hat{i}_G, \hat{j}_G, \hat{k}_G\}$	*null*
	1st (velocity)	$\{\hat{i}_C, \hat{j}_C, \hat{k}_C\}$	*null*
T3	0th (position)	$\{\hat{i}_G, \hat{j}_G, \hat{k}_G\}$	*null*
	1st (velocity)	$\{\hat{i}_C, \hat{j}_C, \hat{k}_C\}$	*null*
R1	0th (orientation)	$\{\hat{u}_{EA1}, \hat{u}_{EA2}, \hat{u}_{EA3}\}$	*null*
	1st (angular velocity)	$\{\hat{i}_C, \hat{j}_C, \hat{k}_C\}$	*null*
R2	0th (orientation)	$\{\hat{u}_{EA1}, \hat{u}_{EA2}, \hat{u}_{EA3}\}$	*null*
	1st (angular velocity)	$\{\hat{i}_C, \hat{j}_C, \hat{k}_C\}$	*null*
R3	0th (orientation)	$\{\hat{u}_{EA1}, \hat{u}_{EA2}, \hat{u}_{EA3}\}$	*null*
	1st (angular velocity)	$\{\hat{i}_C, \hat{j}_C, \hat{k}_C\}$	*null*

Table 4.2: Through and Across Spaces for Tire Component Model

4.2, the across space of a tire component edge could, in principle, be described by any three unit vectors that span a three-dimensional space.

In order to generate efficient simulation code, it is generally a good idea to reduce the number of equations that are used to describe the system by making the number of dynamic equations as close as possible to the number of degrees of freedom. Dynamic equations come from the projection of fundamental cutset equations onto the across space for edges selected into the tree [46, 47]. Therefore, in order to minimize the number of dynamic equations, the analyst should select tree edges that have a limited (or null) across space. Because the tire component edges all have a full across space, they are poor candidates for selection into the system tree. It is *usually* advisable to place all of the tire edges in the cotree.

Using the theory presented in this section and Section 2.4, it is possible to formulate the system equations for a vehicle containing pneumatic tires. Unfortunately, these system equations only represent a portion of the simulation code needed for vehicle dynamics studies. The terminal equations of the tire component are written in terms of $\vec{r}_{P/C}$, $\vec{r}_{Q/C}$, $\vec{F}_P$, and $\vec{M}_P$. The latter two vectors can be resolved in the ISO tire axis system:

$$\vec{F}_P = F_x\hat{u}_x + F_y\hat{u}_y + F_z\hat{u}_z \tag{4.2}$$

$$\vec{M}_P = M_x\hat{u}_x + M_y\hat{u}_y + M_z\hat{u}_z \tag{4.3}$$

where F_x, F_y, M_x, M_y, and M_z are the scalar quantities output by a tire model. These will appear as symbols (dummy variables) in the system equations, as will the scalar components of $\vec{r}_{P/C}$, $\vec{r}_{Q/C}$, and the ISO unit vectors $\hat{u}_x$, $\hat{u}_y$, $\hat{u}_z$. Before the system equations can be solved, we need a method of assigning values to these dummy variables and some kind of provision for calling tire model functions.

4.3 Extended Linear Graph Component Template

Schmitke and McPhee proposed an extended linear graph component template that is capable of containing component-level information in addition to the more traditional topological and edge-level information [46]. Figure 4.2 describes how information is organized within the extended linear graph component template. Items marked with an asterisk (*) were not addressed by Schmitke and McPhee but play a major role in the current work.

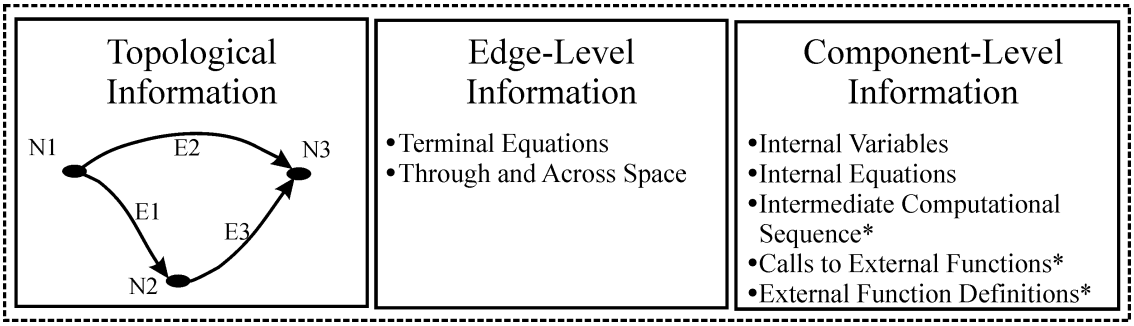


Figure 4.2: Extended Linear Graph Component Template

The connectivity of nodes and edges is considered *topological* information and is necessary for generating system equations, as explained in Section 2.4.

Terminal equations relate the through and across variables of edges within a particular component. There is a terminal equation, a through space, and an across space associated with every derivative level of every edge. These pieces of information are closely tied to individual edges and are referred to as *edge-level* information.

Component-level information may broadly be defined as any information that cannot be associated with an individual edge. Schmitke and McPhee used component-level information to support their subsystem modeling goals. They created linear graph components that represented subsystems of large mechatronic devices; a component might represent one leg of a 6-legged robot. They called these *equivalent subsystem components* or ESCs. An ESC would typically involve only a few nodes to represent the reference frames where the subsystem attached to its surroundings, and a bare minimum of edges that served to connect the boundary nodes. The dynamics of subsystems were described by *internal* variables and *internal* equations which were pre-derived and stored as component-level information [45, 46]. This approach offered Schmitke and McPhee the following advantages:

1. Large models could be broken into smaller blocks for ease of understanding.

2. Subsystem models could be retrieved from a library of ESCs. The fact that equations governing subsystem behavior had already been formulated and stored within the ESC reduced the time needed to formulate system-level equations [46].

When component-level algebraic equations existed, they were simply appended to the list of system-level algebraic equations obtained through the normal formulation procedure (projecting f-circuit equations onto the through space of cotree edges) [45, 46]. Similarly, when component-level differential equations existed, they were simply appended to the list of system-level dynamic equations obtained through the normal formulation procedure (projecting f-cutset equations onto the across space of tree edges) [45, 46]. The list of state variables $\mathbf{x}$ was augmented to contain not only the generalized speeds $\mathbf{p}$ and position coordinates $\mathbf{q}$, but also any additional state variables introduced by the component-level differential equations.

In this work, the equations used to introduce lag in tire slip parameters — equations (3.22) and (3.23) — are implemented as component-level differential equations. These equations cannot be associated with a particular edge in the tire component model, and therefore cannot be listed as terminal equations. The state variables q_1 and q_2 do not qualify as either through or across variables; they must be included as internal component-level variables.

4.4 Structure of Simulation Code

Schmitke and McPhee concerned themselves with the formulation of system equations using linear graph theory. The system equations could have their roots in any combination of topological, edge-level, and component-level information and were expressed in the form of first order differential equations to facilitate their numerical solution [46].

In the current work, the generation of simulation code involves more than writing system equations in first order form. While still a necessary step, it is not the only step in the process.

Figure 4.3 shows the proposed new structure for simulation code, involving three distinct blocks :

1. Calculation of intermediate variables
2. Calls to external functions
3. System equations

The techniques for automatically generating this kind of simulation code are the core ideas presented in this thesis. Block numbers will frequently be used to refer to certain parts of the simulation code shown in Figure 4.3.

The rules for generating blocks 1 and 2 of the simulation code are stored as component-level information. In the DynaFlexPro software implementation, based on the component template shown in Figure 4.2, every component has the potential to define intermediate variables and the ability to call external functions. While this thesis focuses on a tire component model with vehicle applications, the concepts of intermediate variables and

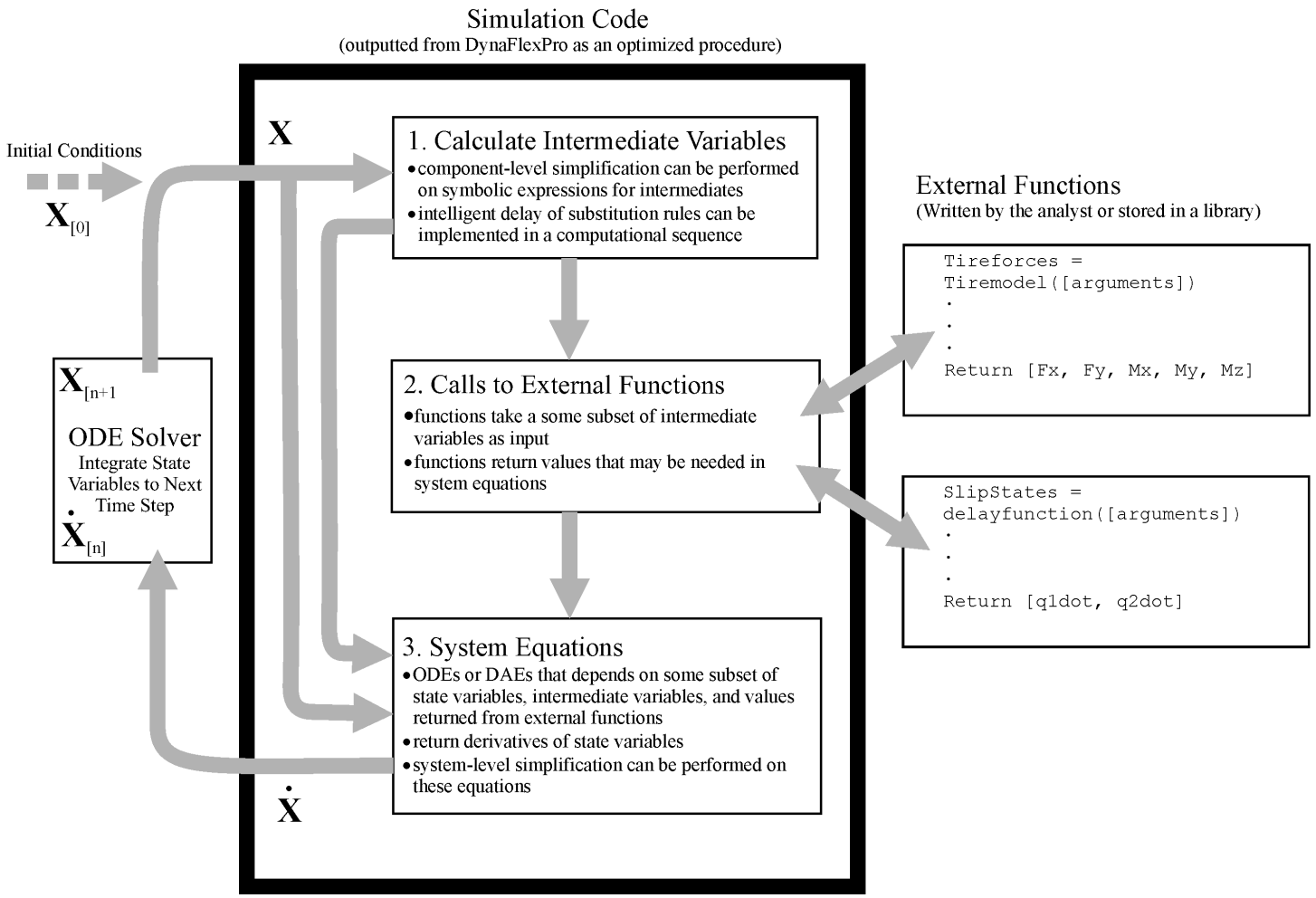


Figure 4.3: Structure of Simulation Code Produced with DynaFlexPro/Tire

external function calls can be extended to any component that requires functions to be evaluated at simulation time, including components that describe contact between two parts.

External functions are written in the same language as the main simulation code. They are called from the main simulation code, but are not part of it. External functions are used to calculate variables that appear in the system equations, are not state variables, and which cannot be fully evaluated during formulation. Tire model functions fit this description perfectly. Tire model functions return F_x, F_y, M_x, M_y, and M_z and usually contain "if" constructs that can only be evaluated during simulation when (at a given time step) the numeric values for tire model inputs are known. Depending on the particular vehicle being modeled and the goals of the simulation, an external function may be used to return the derivatives of extra state variables used to model lag in the tire slip variables. At first glance, there is no obvious reason to evaluate equations (3.22) - (3.23) with an external function. However, for certain applications it may be necessary to make the relaxation lengths B_{long} and B_{lat} dependent on tire operating state or to limit the values of $\frac{dq_1}{dt}$ and $\frac{dq_2}{dt}$ according to the rules laid out in Figure 3.5. In those cases, an external function would be required in order to deal with the "if" constructs needed for the calculation of $\frac{dq_1}{dt}$ and $\frac{dq_2}{dt}$.

The use of external functions also provides the analyst with the freedom to modify the way that tire forces and moments are calculated, or the way that lag in tire slip parameters is modeled, without having to reformulate the system equations or rebuild the main simulation code. Custom external functions can be written and used with an existing model.

Intermediate variables may or may not appear directly in the system equations. For a vehicle model, the components of the vector $\vec{r}_{P/C}$ of each tire will appear in the system equations as dummy variables. Values are assigned to these dummy variables in block 1 of the simulation code. We may classify them as intermediate variables that appear directly in the system equations. The slip angles of each tire are intermediate variables that do not appear in the system equations. Slip angles are calculated in block 1 of the simulation code so that they can be passed as arguments to tire model functions in block 2 of the simulation code. That is the mechanism by which slip angles, and many other intermediate variables, contribute to advancing the solution.

Sections 2.4 and 4.2 have covered the method by which block 3 equations are generated symbolically. In the coming sections, the rules for generating blocks 1 and 2 of the simulation code are discussed. These rules are stored as component-level information within the tire component model and draw on elements of linear graph theory and symbolic computing.

4.5 Generating Block 1 of Simulation Code

4.5.1 Symbolic Expressions for the Kinematics of the Tire Center Frame in Terms of Branch Across Variables

Block 1 of the simulation code must calculate tire intermediate variables in terms of state variables, which, for mechanical systems, are across variables belonging to edges selected into the tree. As presented in Chapter 3, the intermediate variables used as tire model arguments depend on the position, orientation, velocity, and angular velocity of the tire center frame C with respect to the road reference frame Q — i.e. the across variables associated with edges T3 and R3 of the tire component. Therefore, the first step to obtaining symbolic expressions for tire intermediates is to find symbolic expressions for the across variables of tire edges T3 and R3 in terms of branch across variables.

Figure 4.4 shows a linear graph illustrating typical use of a tire component (all tire component edges are placed in the cotree). The edges labeled "RoadKin" are used to define the kinematics of the road reference frame Q with respect to the inertial Ground frame. Edges labeled "E" might represent a suspension system for supporting the tire, in which case nodes N1 and N2 would correspond to points (reference frames) on the suspension parts. Because edges Tire_T3 and Tire_R3 are in the cotree, their across variables will not appear in the system equations and will not be state variables for the simulation code.

Linear graph branch transformations provide expressions for all secondary (chord) across variables in terms of primary (branch) across variables. During formulation, this information is used to eliminate all secondary across variables from the system equations. For the system shown in Figure 4.4, equation (2.27) will give expressions for the across variables of the edges Tire_T3, Tire_R3, and all of the other chords, in terms of the analyst's chosen modeling variables.

In order to use equation (2.27), there must be a chord joining the nodes of interest. Looking ahead to computer implementation, we would like a method for generating a symbolic expression, in terms of primary variables, for an across variable measured between any two nodes in the graph, regardless of whether they are joined by a chord. Branch transformations are based on circuit equations. Let's take a step backward and look at the circuit containing edges Tire_T3, RoadKin_T1, E_T1, E_T2, and E_T3. This circuit happens to be an f-circuit since it contains a single chord. At the 0th derivative level, we have:

$$-\vec{r}_{Tire\_T3} - \vec{r}_{RoadKin\_T1} + \vec{r}_{E\_T1} + \vec{r}_{E\_T2} + \vec{r}_{E\_T3} = 0 \tag{4.4}$$

which can be solved for the displacement of node C with respect to node Q:

$$\vec{r}_{C/Q} = \vec{r}_{Tire\_T3} = -\vec{r}_{RoadKin\_T1} + \vec{r}_{E\_T1} + \vec{r}_{E\_T2} + \vec{r}_{E\_T3} \tag{4.5}$$

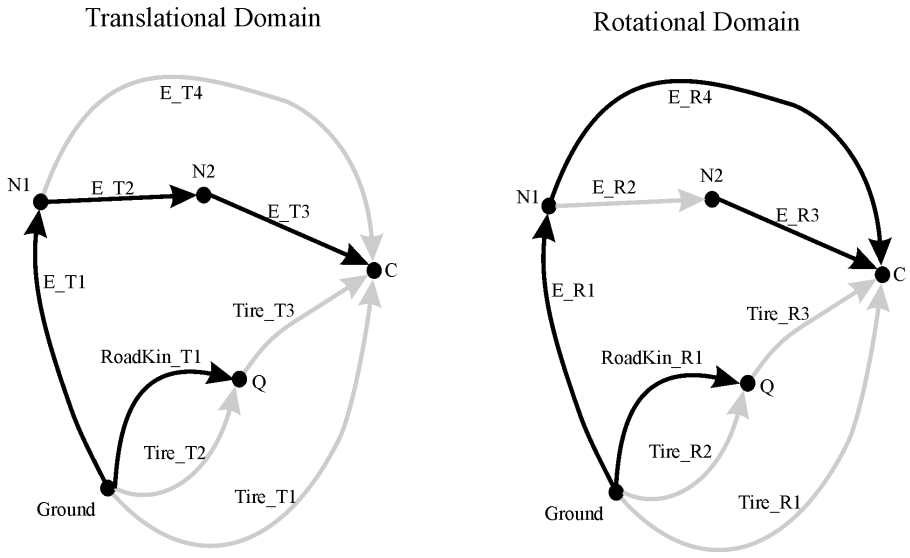


Figure 4.4: A Linear Graph with Tire Component

The displacement of node C with respect to node Q is the algebraic sum of displacements for branches along the path from node Q to node C. In fact, as long as the variable of interest is a vector or scalar, one can obtain an expression for an across variable measured between any two nodes by algebraically adding the across variables for branches joining those two nodes. This is a general way of expressing linear graph branch transformations that works whether or not there is a chord joining the nodes of interest. The method can be applied to find the velocity (translational domain, 1st derivative level) and angular velocity (rotational domain, 1st derivative level) between nodes C and Q in Figure 4.4:

$$\vec{v}_{C/Q} = \vec{v}_{Tire\_T3} = -\vec{v}_{RoadKin\_T1} + \vec{v}_{E\_T1} + \vec{v}_{E\_T2} + \vec{v}_{E\_T3} \tag{4.6}$$

$$\vec{\omega}_{C/Q} = \vec{\omega}_{Tire\_R3} = -\vec{\omega}_{RoadKin\_T1} + \vec{\omega}_{E\_R1} + \vec{\omega}_{E\_R4} \tag{4.7}$$

Since finite 3D rotations are neither vectors nor scalars, the method for determining the orientation (rotational domain, 0th derivative level across variable) between two nodes is slightly different. In this case, multiplication of rotation matrices is performed:

$$\mathbf{R}_{C/Q} = \mathbf{R}_{Tire\_R3} = [\mathbf{R}_{RoadKin\_T1}]^T \mathbf{R}_{E\_R1} \mathbf{R}_{E\_R4} \tag{4.8}$$

In order for equations like (4.5) - (4.7) to be used in practice, all of the vectors must be expressed in a common reference frame before they are added. This is accomplished by multiplying a column matrix representation of each vector by an appropriate rotation matrix. Since the rotation matrix between any two reference frames can be found using

equations like (4.8), this is a relatively straightforward task, easily programmed into software.

A user of DynaFlexPro can obtain symbolic expressions, in terms of the chosen modeling variables, for the across variables between any two nodes in a linear graph. If the quantity is a vector, it can be expressed in terms of components in any reference frame, as long as the reference frame corresponds to a node in the graph. Within the tire component model, these concepts are used to extract expressions for $\vec{r}_{Tire\_T3}$, $\vec{v}_{Tire\_T3}$, $\vec{\omega}_{Tire\_R3}$, and to resolve these vectors in the road reference frame Q.

4.5.2 Tire Intermediate Variables

The tire intermediate variables used for this work are listed in Table 4.3. It is impossible to predict a comprehensive list of intermediate variables that includes the inputs needed for every conceivable tire model, but this list is fairly extensive and should work for the majority of cases.

When an intermediate variable is represented by a column matrix of scalars, it contains the components of the corresponding vector resolved in the road reference frame Q. Note that some intermediate variables have options for which expression to use. When an analyst decides to use a tire component in his model, he must choose between one of three symbolic expressions for R_{eff} and also must choose whether to calculate S and α based on their kinematic definitions, or based on time-delayed state variables q_1 and q_2.

The expressions for intermediate variables toward the top of Table 4.3 depend on $\mathbf{r_{Tire\_T3}}$, $\mathbf{v_{Tire\_T3}}$, and $\boldsymbol{\omega}_{\mathbf{Tire\_R3}}$, which have been expressed in terms of state variables using the general form of the linear graph branch transformations discussed in Section 4.5.1, and also on $\mathbf{u_{SymAxis}}$, which is a column matrix representation of the unit vector that defines the axis of tire symmetry. The expressions for variables toward the end of the list depend on the variables toward the top of the list. It is evident that certain variables must be evaluated before others, as indicated by the pseudo-Gantt chart in Table 4.3. Longer bars do not necessarily indicate that an expression takes longer to evaluate. They indicate an increased amount of freedom in the order of evaluation. For instance, the expression for $\mathbf{r_{Q/C}}$ can be evaluated any time since it does not depend on other intermediate variables and no other intermediate variables depend on it. In contrast, α must be evaluated toward the end of the list because it requires several other intermediate expressions to be evaluated before it.

4.5.3 Evaluating Tire Intermediate Variables with Complete Substitution

In this section, the cost of intermediate expressions related to the front left tire of a generic 4-wheeled vehicle model will be presented in order to highlight some important issues involved with evaluating tire intermediate variables. For more information on the

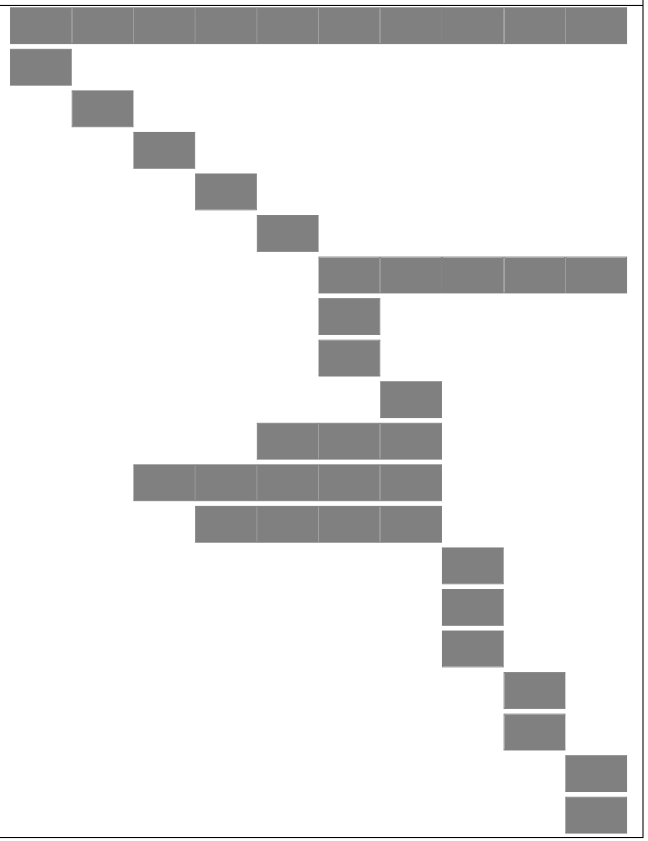
Intermediate	Symbolic Math Expression	Order of Evaluation
$\mathbf{r_{Q/C}}$	$-\mathbf{r_{Tire\_T3}}$	
$\mathbf{u_z}$	$[0,\ 0,\ 1]^T$	
$\mathbf{u_x}$	$(\mathbf{u_{SymAxis}} \times \mathbf{u_z})/(\lvert\mathbf{u_{SymAxis}} \times \mathbf{u_z}\rvert)$	
$\mathbf{u_y}$	$\mathbf{u_z} \times \mathbf{u_x}$	
γ	$\arcsin((\mathbf{u_y} \times \mathbf{u_{SymAxis}}) \cdot \mathbf{u_x})$	
R_l	$(\mathbf{r_{Tire\_T3}} \cdot \mathbf{u_z}/(\cos(\gamma))$	
F_z	$\max(k_z(R_u - R_l)\cos(\gamma) - c_z(\mathbf{v_{Tire\_T3}}) \cdot \mathbf{u_z},\quad 0)$	
$\mathbf{r_{P/C}}$	$[\ R_l \sin(\gamma)\mathbf{u_{y[1]}},\ R_l \sin(\gamma)\mathbf{u_{y[2]}},\ -R_l \cos(\gamma)\ ]^T$	
R_{eff}	$R_l \quad or \quad R_u \quad or \quad$ equation (3.13)	
$\mathbf{r_{E/C}}$	$\mathbf{r_{E/C}} = R_{eff}/R_l \cdot \mathbf{r_{P/C}}$	
Ω	$(\boldsymbol{\omega}_{\mathbf{Tire\_R3}} \cdot \mathbf{u_{SymAxis}} - \boldsymbol{\omega}_{\mathbf{Tire\_R3}} \cdot \mathbf{u_z})\sin(\gamma))/\cos^2(\gamma)$	
V_{Cx}	$\mathbf{v_{Tire\_T3}} \cdot \mathbf{u_x}$	
V_{Cy}	$\mathbf{v_{Tire\_T3}} \cdot \mathbf{u_y}$	
V_{Px}	$V_{Cx} + (\boldsymbol{\omega}_{\mathbf{Tire\_R3}} \times \mathbf{r_{P/C}}) \cdot \mathbf{u_x}$	
V_{Py}	$V_{Cy} + (\boldsymbol{\omega}_{\mathbf{Tire\_R3}} \times \mathbf{r_{P/C}}) \cdot \mathbf{u_y}$	
V_{Ex}	$V_{Cx} + (\boldsymbol{\omega}_{\mathbf{Tire\_R3}} \times \mathbf{r_{E/C}}) \cdot \mathbf{u_x}$	
V^*_{Px}	$V_{Px} + \Omega R_l$	
V^*_{Ex}	$V_{Ex} + \Omega R_{eff}$	
S	$-V_{Ex}/\lvert V^*_{Ex}\rvert \quad or \quad q_1$	
α	$\arctan(V_{Py}/\lvert V^*_{Px}\rvert) \quad or \quad \arctan(q_2)$	

Table 4.3: Tire Component Intermediate Variables and Possible Orders of Evaluation

vehicle model used for this case study, the reader is referred to Section 5.3.

We will seek a symbolic expression for each tire intermediate in terms of state variables, beginning with $\mathbf{r_{Q/C}}$ and proceeding down the ordered list toward slip angle. At each step along the way, the expressions for previous tire intermediates are substituted into the appropriate formula in Table 4.3 to get an expression for the next tire intermediate. The expressions themselves are too large to present here; instead, Table 4.4 displays a count of the additions, multiplications, divisions, and function calls (sin, cos, sqrt, etc.) contained within each expression. These counts were obtained using Maple's **codegen:-cost** command and give a reasonable measure of the size and complexity of each symbolic expression.

	Cost of Expression			
Tire Intermediate	$+$	$\times$	$\div$	f
$\mathbf{r_{Q/C}}$	13	27	0	64
$\mathbf{u_z}$	0	0	0	0
$\mathbf{u_x}$	14	44	0	112
$\mathbf{u_y}$	15	44	0	112
γ	17	54	0	132
R_l	20	59	1	145
F_z	31	74	1	168
$\mathbf{r_{P/C}}$	128	390	3	945
R_{eff}	41	121	2	291
$\mathbf{r_{E/C}}$	197	576	6	1383
Ω	67	166	1	430
V_{Cx}	37	78	0	186
V_{Cy}	37	78	0	186
V_{Px}	243	672	4	1644
V_{Py}	243	672	4	1644
V_{Ex}	335	920	8	2228
V^*_{Px}	331	898	6	2219
V^*_{Ex}	446	1208	11	2949
S	too large to evaluate			
α	too large to evaluate			

Table 4.4: Cost of Tire Intermediate Expressions with Complete Substitution

The symbolic expressions for intermediate variables near the top of the list have a relatively small computational cost. However, the process of substituting every symbolic expression into the next results in a rapid increase in the size and complexity of the tire intermediate expressions. Toward the end of the list, the expressions get so large that the symbolic math software (Maple, in this case) has trouble dealing with them. For example, when Maple was asked to divide the large expression for V_{Ex} by the large expression for

V_{Ex}^* in order to obtain an expression for longitudinal slip S, the PC being used (1 GB of RAM) ran out of available memory and Maple could not return a result. A more intelligent way of expressing tire intermediates in terms of state variables is needed — one which does not involve enormous expressions that exceed the limits of what modern computers and state-of-the-art symbolic math software can handle.

4.5.4 Evaluating Tire Intermediate Variables using a Computational Sequence

A computational sequence can be used to control the size of tire intermediate expressions. Instead of continuously substituting large symbolic expressions in an attempt to obtain an expression for each tire intermediate in terms of state variables, tire intermediates can be written in terms of dummy names referring to other tire intermediates that are higher in the list.

$$
\begin{aligned}
&\vdots \\
&R_l\text{LHS} = R_l\text{RHS} \quad \text{(a large symbolic expression)} \\
&\vdots \\
&\mathbf{r}_{\mathbf{P/C}_{[1]}}\text{LHS} = R_l\text{LHS} \cdot \sin(\gamma\text{LHS}) \cdot \mathbf{u}_{\mathbf{y}_{[1]}}\text{LHS} \\
&\mathbf{r}_{\mathbf{P/C}_{[2]}}\text{LHS} = R_l\text{LHS} \cdot \sin(\gamma\text{LHS}) \cdot \mathbf{u}_{\mathbf{y}_{[2]}}\text{LHS} \\
&\mathbf{r}_{\mathbf{P/C}_{[3]}}\text{LHS} = -R_l\text{LHS} \cdot \cos(\gamma\text{LHS}) \\
&R_{eff}\text{LHS} = 0.35145 + 0.00465789 \arctan(-146.33 + 412.2 R_l\text{LHS}) + 0.01 R_l\text{LHS} \\
&\mathbf{r}_{\mathbf{E/C}_{[1]}}\text{LHS} = R_{eff}\text{LHS}/R_l\text{LHS} \cdot \mathbf{r}_{\mathbf{P/C}_{[1]}}\text{LHS} \\
&\mathbf{r}_{\mathbf{E/C}_{[2]}}\text{LHS} = R_{eff}\text{LHS}/R_l\text{LHS} \cdot \mathbf{r}_{\mathbf{P/C}_{[2]}}\text{LHS} \\
&\mathbf{r}_{\mathbf{E/C}_{[3]}}\text{LHS} = R_{eff}\text{LHS}/R_l\text{LHS} \cdot \mathbf{r}_{\mathbf{P/C}_{[3]}}\text{LHS} \\
&\vdots
\end{aligned}
$$

Figure 4.5: Portion of a Computational Sequence

Consider the portion of a computational sequence shown in Figure 4.5. $R_l\text{LHS}$ will be calculated near the beginning of the computational sequence using the symbolic expression $R_l\text{RHS}$ that defines the loaded radius for a particular tire. $R_l\text{RHS}$ is not substituted into the expressions for tire intermediates that depend on R_l. Instead, the dummy name $R_l\text{LHS}$ is used. At simulation time, once the numeric value of $R_l\text{LHS}$ is calculated, that number can be used to calculate $\mathbf{r}_{\mathbf{P/C}}$, R_{eff}, $\mathbf{r}_{\mathbf{E/C}}$, etc. based on relatively simple expressions. In Figure 4.5, the ADAMS/Pacejka method of calculating R_{eff} has been used according to equation (3.13), with numeric parameters specified for ρ, B, D, and F.

In Figure 4.5, one multiplication and one division are used to calculate each component of $\mathbf{r}_{\mathbf{E/C}}$ in terms of dummy variables that have been assigned earlier in the computational

Tire Intermediate	Expressions Not Substituted (left as dummy variables)	Cost of Expression			
		$+$	$\times$	$\div$	f
$\mathbf{r_{Q/C}}$		13	27	0	64
$\mathbf{u_z}$		0	0	0	0
$\mathbf{u_x}$		14	44	0	112
$\mathbf{u_y}$		15	44	0	112
γ	$\mathbf{u_x}$, $\mathbf{u_y}$	15	39	0	93
R_l	γ	3	5	1	13
F_z	R_l	11	15	0	23
$\mathbf{r_{P/C}}$	$\mathbf{u_y}$, γ, R_l	1	5	0	3
R_{eff}	R_l	1	3	0	1
$\mathbf{r_{E/C}}$	R_l, R_{eff}, $\mathbf{r_{P/C}}$	0	3	3	0
Ω	γ	32	58	1	168
V_{Cx}	$\mathbf{u_x}$	33	46	0	97
V_{Cy}	$\mathbf{u_y}$	33	46	0	97
V_{Px}	V_{Cx}, $\mathbf{r_{P/C}}$, $\mathbf{u_x}$	41	71	0	197
V_{Py}	V_{Cy}, $\mathbf{r_{P/C}}$, $\mathbf{u_y}$	41	71	0	197
V_{Ex}	V_{Cx}, $\mathbf{r_{E/C}}$, $\mathbf{u_x}$	41	71	0	197
V^*_{Px}	V_{Px}, Ω, R_l	1	1	0	0
V^*_{Ex}	V_{Ex}, Ω, R_{eff}	1	1	0	0
S	V_{Ex} , V^*_{Ex}	0	0	1	1
α	V_{Py}, V^*_{Px}	0	0	1	2

Table 4.5: Cost of Tire Intermediate Expressions with Delay of Substitution

sequence, for a grand total of 3 multiplications and 3 divisions used to calculate $\mathbf{r_{E/C}}$. It would be significantly more costly to evaluate $\mathbf{r_{E/C}}$ for the case where all symbolic expressions were substituted — as listed in Table 4.4, it would take 197 additions 576 multiplications, 6 divisions and 1383 functions to do the same job that 3 multiplications and 3 divisions do as part of a computational sequence.

The rules for generating a computational sequence that defines tire intermediates in terms of state variables are stored as component-level information. The rules used for the present work are summarized in Table 4.5, which shows the variables that have been left as dummy names in expressions for tire intermediates. If a tire intermediate depends on $\mathbf{r_{Tire\_T3}}$, $\mathbf{v_{Tire\_T3}}$, $\boldsymbol{\omega}_{\mathbf{Tire\_R3}}$ or $\mathbf{u_{SymAxis}}$, then symbolic expressions for these variables in terms of state variables are substituted. If a tire intermediate depends on another tire intermediate that is higher in the list, that variable is left as a dummy name. The only exception is the column matrix representation of the ISO Z unit vector, which is always aligned with the z axis of the road reference frame. Substituting $\mathbf{u_z} = [0,\ 0,\ 1]^T$ actually results in simpler expressions for the tire intermediates that depends on $\mathbf{u_z}$, so this expression is always substituted.

These rules were used to construct a computational sequence capable of calculating tire intermediates for the generic 4-wheeled vehicle example. The cost of the expressions used in this computational sequence are displayed in Table 4.5. Use of a computational sequence drastically reduces the size and complexity of tire intermediate expressions compared to the case where complete symbolic substitution is used. Table 4.4 and Table 4.5 compare the cost of evaluating the same variables for the same vehicle, using two different methods.

The rules for generating a computational sequence used in this work and presented in 4.5 do not claim to be ideal. Other schemes for delaying the substitution of certain symbolic expressions are possible. For example, $\mathbf{r}_{\mathbf{Tire\_T3}}$, $\mathbf{v}_{\mathbf{Tire\_T3}}$, $\boldsymbol{\omega}_{\mathbf{Tire\_R3}}$ and $\mathbf{u}_{\mathbf{SymAxis}}$ could have been added to the list of tire intermediate variables, and dummy names for these variables could have been used in the expressions for tire intermediates that depend on them. The ideal rules for generating an intermediate computational sequence are dependent on the vehicle model being considered. The rules expressed in Table 4.5 are used in this thesis because they are logical, they prevent symbolic expressions for tire intermediates from growing out of control, and they produce an effective and compact block 1 for the simulation code of the example problems considered. This should not deter future investigation of alternative rules for generating computational sequences for tire intermediate variables.

4.6 Generating Block 2 of Simulation Code

Block 2 of the simulation code contains calls to external functions that return values needed to calculate derivatives of state variables in block 3. Block 2 is the shortest block of the simulation code, but is vital to any vehicle model because it is the block that returns tire forces and moments.

Four pieces of information are necessary to construct block 2 of the simulation code:

1. For each component in the system, the names of external functions to be called
2. The names of intermediate variables required as arguments to each external function, in the order in which they are expected
3. A list of numeric parameters that are required by each external function, in the order in which they are expected
4. The names of the variables whose values are returned by each external function, in the order in which they are returned

Figure 4.6 depicts how this information is used to generate the necessary code for calling an external function. First the numeric parameters are assigned to elements of an array. Then the function gets called by name, passing a list of intermediate variables and the recently defined array of numeric parameters as arguments. The external function returns values in array form, which are then assigned to unique variable names. Variables

ExFunParams[1] = user-specified number
ExFunParams[2] = user-specified number
⋮
ExFunParams[n] = user-specified number
[ExFunValues] = ExFunName([IntVarNames], ExFunParams)
ReturnedName1 = ExFunValues[1]
ReturnedName2 = ExFunValues[2]
⋮
ReturnedName$n$ = ExFunValues[n]

Figure 4.6: Rules for Generating Block 2 of Simulation Code

with these names appear in the system equations contained in block 3 of the simulation code.

It is instructive to consider an example involving the type of external functions used by tire components. Imagine you have a model of a bicycle. You give the instance of the tire component used to represent the front tire the unique name "Front". The label "Front" is automatically appended to the names of variables associated with this tire in order to differentiate them from similar variables associated with the rear tire. You give the instance of the tire component used to represent the rear tire the unique name "Rear". You want to calculate forces and moments generated by the front tire using a Fiala tire model function, and forces and moments generated by the rear tire using a Pacejka 2002 tire model function. Because of your vast experience with imaginary bicycles, you know that lag is not important for the front tire, but may be important for the rear tire. You wish to use an external function called "StretchedString" to evaluate the derivatives of the rear tire's state variables. Block 2 of the simulation code for this vehicle would look very similar to Figure 4.7.

The Fiala function requires 4 tire intermediates as input — F_z, S, α, and Ω — as well as a list of 6 parameters describing the force and moment generating capabilities of the front tire, which are assigned to the array TireModelParams_Front. The values returned from the Fiala function are assigned to the variables F_{x_Front}, F_{y_Front}, M_{x_Front}, M_{y_Front}, and M_{z_Front}, which appear in the system equations located in block 3 of the simulation code.

The Pacejka 2002 tire model function requires 7 tire intermediates as input — F_z, S, α, Ω, γ, R_{eff}, and V_{Cx} — as well as a list of 117 parameters describing the force and moment generating capabilities of the rear tire. The call to the Pacjeka02 function necessary to calculate rear tire forces and moments is handled in the same manner as the call to the Fiala function used for the front tire.

Because you have no interest in the transient behavior of the front tire, its slip angle

Block 1

$\vdots$

$S_{Front}\text{LHS} = -V_{ExFront}\text{LHS} / \left|V^*_{ExFront}\text{LHS}\right|$

$\alpha_{Front}\text{LHS} = \arctan(V_{PyFront}\text{LHS} / \left|V^*_{PxFront}\text{LHS}\right|)$

$S_{Rear}\text{LHS} = \mathbf{x}_{[k]}$

$\alpha_{Rear}\text{LHS} = \arctan(\mathbf{x}_{[k+1]})$

Block 2

TireModelParams_Front[1] = 0.16

$\vdots$

TireModelParams_Front[6] = 0.2

ForceValues_Front = Fiala($F_{z\_Front}\text{LHS}, S_{Front}\text{LHS}, \alpha_{Front}\text{LHS}, \Omega_{Front}\text{LHS}$, TireModelParams_Front)

F_{x_Front} = ForceValues_Front[1]

$\vdots$

M_{z_Front} = ForceValues_Front[5]

TireModelParams_Rear[1] = 4850

$\vdots$

TireModelParams_Rear[117] = −0.24116

ForceValues_Rear = Pacejka02($F_{z\_Rear}\text{LHS}, S_{Rear}\text{LHS}, \alpha_{Rear}\text{LHS}, \Omega_{Rear}\text{LHS}, \gamma_{Rear}\text{LHS}, R_{eff\_Rear}\text{LHS}, V_{Cx_Rear}\text{LHS}$, TireModelParams_Rear)

F_{x_Rear} = ForceValues_Rear[1]

$\vdots$

M_{z_Rear} = ForceValues_Rear[5]

SlipStateParams_Rear[1] = 5900

$\vdots$

SlipStateParams_Rear[10] = −0.90729

SlipDerivatives_Rear = StretchedString($F_{z\_Rear}\text{LHS}, \gamma_{Rear}\text{LHS}, V_{PyRear}\text{LHS}, V_{ExRear}\text{LHS}, V^*_{PxRear}\text{LHS}, V^*_{ExRear}\text{LHS}$, SlipStateParams_Rear)

$q1dot_{Rear}$ = SlipDerivatives_Rear[1]

$q2dot_{Rear}$ = SlipDerivatives_Rear[2]

Block 3

$\vdots$

$\dot{\mathbf{x}}_{[k]} = q1dot_{Rear}$

$\dot{\mathbf{x}}_{[k+1]} = q2dot_{Rear}$

Figure 4.7: Block 2 of Simulation Code for Imaginary Bicycle

and longitudinal slip are calculated based on their kinematic definitions. The statements $S_{Front}\text{LHS} = -V_{ExFront}\text{LHS}/\,|V^*_{ExFront}\text{LHS}|$ and $\alpha_{Front}\text{LHS} = \arctan(V_{PyFront}\text{LHS}/\,|V^*_{PxFront}\text{LHS}|)$ will appear in the computational sequence of block 1.

Unlike the front tire, the rear tire has extra state variables associated with its transient behavior. If q_{1Rear} and q_{2Rear} correspond to the the state variables $\mathbf{x}_{[k]}$ and $\mathbf{x}_{[k+1]}$, the statements $S_{Rear}\text{LHS} = \mathbf{x}_{[k]}$ and $\alpha_{Rear}\text{LHS} = \arctan(\mathbf{x}_{[k+1]})$ will appear in the computational sequence of block 1. The stretched string external function will be used to calculate the derivatives of q_{1Rear} and q_{2Rear} in block 2 of the simulation code. This function requires 6 tire intermediate variables as input — F_z, γ, V_{Py}, V_{Ex}, V^*_{Px}, and V^*_{Ex} — as well as 10 parameters describing the tire's dynamic behavior. The statements $\dot{\mathbf{x}}_{[k]} = q1dot_{Rear}$ and $\dot{\mathbf{x}}_{[k+1]} = q2dot_{Rear}$ will appear in block 3 of the simulation code.

External functions may be developed independently of the simulation code generated by DynaFlexPro. An analyst always has the option to write his own custom tire model function. The only requirement is that she provide DynaFlexPro with information regarding the name of the function to be called, the inputs required by the function, and the values returned by the function so that block 2 of the simulation code can be properly constructed.

Defining an external function can be time-consuming for an analyst. In addition to offering analysts the flexibility to write their own external functions, the current approach offers the convenience of using pre-defined external functions that are stored in a component-level library. An analyst working in an organization that frequently uses Pacejka 2002 tire models for their simulations would appreciate having a Pacejka 2002 tire model function stored as component-level information within the DynaFlexPro tire component model. Whenever a new vehicle model is created, she would simply specify "Pacejka 2002" as the tire model type for each instance of the tire component. She would not need to explicitly specify which intermediate variables the tire model function requires as input, how many numeric parameters are needed, or how many force and moment values are returned as output. Those pieces of information would be stored as component-level information linked to the "Pacejka 2002" tire model option. The function definition itself could also be stored as component-level information. This is a particularly attractive option for a package like DynaFlexPro, which is written in the Maple language. Code that exists in the Maple language can easily be translated to C, Fortran, or Matlab using Maple's **CodeGeneration** package [23]. Tire model functions written in the Maple language, and stored as component-level information within the DynaFlexPro tire component model, can be exported to the simulation language and used along with the main simulation code to numerically solve for the vehicle's time domain response.

The commercial version of the DynaFlexPro tire component model, known as DynaFlexPro/Tire, contains a library with three pre-defined tire model functions — the Fiala and Pacejka 2002 tire model functions discussed in Section 3.4, and the Calspan tire model function used for the IAVSD benchmarking study [21] — as well as a stretched

string function for modeling transient behavior [19].

4.7 Expression Manipulation Routines

As mentioned before, the tire component model proposed in this chapter has been implemented in the DynaFlexPro software, which is built on top of the symbolic math program, Maple, and makes extensive use of Maple's symbolic math tools. Maple offers two commands — **simplify()** and **combine()** — for manipulating symbolic expressions into different forms [23]. These commands apply rules (pre-defined by Maple) for simplifying trigonometric functions, radicals, logarithmic functions, exponential functions, powers, and various special functions [23]. Of these, trigonometric functions are most common to equations describing multibody systems.

The **simplify()** command prefers to arrange expressions such that trigonometric functions are multiplied together or raised to a power. By contrast, the **combine()** command attempts to combine sums and products of trig functions into a single term [23]. Both commands have the ability to reduce the computational cost associated with evaluating an expression, but they also have the ability to increase the computational cost. The utility of **simplify()** and **combine()** depends on the expression to be evaluated. For the examples in Figure 4.8, **combine()** works best on expression a, and **simplify()** works best on expression b.

In this thesis, the term *system-level simplification* refers to expression manipulation done on the system equations, which are the basis of block 3 of the simulation code. DynaFlexPro allows users the option of using **simplify()** or **combine()** to manipulate the form of the dynamic equations that define the derivatives of generalized speeds **p**. The user is also given the choice of using **simplify()** or **combine()** on the algebraic constraint equations that express the dependencies of the position coordinates **q**. Of course, choosing not to apply system-level simplification is also valid.

$a = \sin(x)\cos(y) + \sin(y)\cos(x)$
$\mathbf{simplify}(a) = \sin(x)\cos(y) + \sin(y)\cos(x)$
$\mathbf{combine}(a) = \sin(x+y)$
$b = \cos^5(x) + \sin^4(x) + 2\cos^2(x) - 2\sin^2(x) - \cos(2x) - 2\sin^2(x) - \cos(2x)$
$\mathbf{simplify}(b) = \cos^4(x)\,(cos(x)+1)$
$\mathbf{combine}(b) = \frac{1}{16}\cos(5x) + \frac{5}{16}\cos(3x) + \frac{5}{8}\cos(x) + \frac{3}{8} + \frac{1}{8}\cos(4x) + \frac{1}{2}\cos(2x)$

Figure 4.8: Maple's Expression Manipulation Commands Applied to Trigonometric Examples

The term *component-level simplification* will be used to refer to expression manipulation done on the symbolic expressions for tire intermediates found in the computational sequence that makes up block 1 of the simulation code. Here again, an analyst has the option of using the **simplify()** command, the **combine()** command, or to use no expression manipulation at all.

4.8 Optimization of Simulation Code

When simulation code is assembled, it is in the form of a procedure that calculates derivatives of state variables given their current values as input. At the heart of this procedure is a computational sequence that may be divided into 3 blocks, as shown in Figure 4.3. Maple's **CodeGeneration** package has the ability to optimize a procedure to minimize the computational cost associated with its evaluation. This is done by identifying repeated terms and storing them as temporary variables so that they can be evaluated once and used several times, as opposed to being re-evaluated every time they are needed. During the optimization process, useless variables — variables that were assigned somewhere in the original procedure but do not contribute to the calculation of outputs — are dropped, since calculating them would be a waste of CPU time.

To better illustrate the concept of code optimization, consider Figure 4.9. A procedure is defined that takes the variables x, y, z, and β as input and returns the value of $Var3\text{LHS}$ as output. The original procedure uses a very inefficient computational sequence to calculate $Var3\text{LHS}$. There are several $\cos(x)$ and $\sin(y)$ terms repeated, resulting in a large number of function evaluations. The variable $Var2\text{LHS}$ is assigned in the original computational sequence but serves no useful purpose; it is not needed for calculating $Var3\text{LHS}$, which is the only value returned by this procedure. Calculation of $Var2\text{LHS}$ is a waste of time.

In the optimized procedure, the statements necessary to calculate $Var2\text{LHS}$ have been dropped. Temporary variables have been introduced for terms that appear more than once in the expressions for $Var1\text{LHS}$ and $Var3\text{LHS}$. For instance, $\cos(x)$ and $\sin(y)$ are only calculated once, greatly reducing the number of function evaluations.

The principles of code optimization are extremely important for the present work. They reduce the computational cost associated with vehicle simulation code that has been constructed using the techniques presented earlier in this chapter, helping to make it suitable for real-time applications. The ability to drop useless terms is important for vehicle models containing instances of the DynaFlexPro tire component. The same (extensive) list of tire intermediate variables is assigned during block 1 of the simulation code every time a tire component is used. If the analyst has chosen an extremely simple tire model — for instance, one that depends only on inclination angle γ — then many of the tire intermediates assigned in block 1 will never be used. These useless variables will be automatically removed by the code optimization routine.

Original Procedure

5 additions, 6 multiplications, 3 divisions, 13 functions

proc (x, y, z, β)

$$Var1_{\text{LHS}} = \frac{\cos(x) + \cos(x)\sin(y) + \cos^2(x) + \sin(y)}{\cos(x)\sin(y)}$$

$$Var2_{\text{LHS}} = \frac{\ln(4\tan(\beta)}{1 + \sin^2(y)}$$

$$Var3_{\text{LHS}} = \frac{Var1\text{LHS}\cos(x)}{Var1\text{LHS} + \cos(x)\sin(y)}$$

return $Var3_{\text{LHS}}$

end proc

Optimized Procedure

4 additions, 3 multiplications, 2 divisions, 2 functions

proc (x, y, z, β)

$t1 = \cos(x)$

$t2 = \sin(y)$

$t3 = t1 \cdot t2$

$t4 = t1^2$

$$Var1_{\text{LHS}} = \frac{t1 + t3 + t4 + t2}{t3}$$

$$Var3_{\text{LHS}} = \frac{Var1_{\text{LHS}} \cdot t1}{Var1_{\text{LHS}} + t3}$$

return $Var3_{\text{LHS}}$

end proc

Figure 4.9: Code Optimization Example

Chapter 5

Vehicle Model Examples

5.1 Overview

The tire component model presented in Chapter 4 was programmed in the Maple symbolic math language and became known as DynaFlexPro/Tire, an add-on package for the DynaFlexPro multibody modeling and equation formulation tool. DynaFlexPro and DynaFlexPro/Tire were used to generate simulation code describing the behavior of two example vehicles — a 4-wheeled vehicle with independent suspension and an articulated forestry skidder. A braking maneuver and a steering maneuver were investigated for each of these vehicles.

These examples are used to validate the theory behind DynaFlexPro/Tire by testing its accuracy versus a well-established vehicle dynamics modeling tool (MSC.ADAMS). In addition, the example models are used to investigate options available with linear graph theory and symbolic computing for influencing the efficiency of simulation code. Simulation times for DynaFlexPro models being solved in the Simulink environment are compared to real-time in order to asses the suitability of these models for hardware-in-the-loop simulation.

5.2 Notes on Simulation Times in Simulink

For the examples in this chapter, simulation code is generated in the Maple environment using DynaFlexPro and output from Maple in the C language. All external functions necessary to run a particular simulation (tire model functions, functions for returning derivatives of slip states) were also exported from Maple as C code. Using a procedure recommended by the Mathworks, the simulation code and external functions were used to construct a wrapper S-Function suitable for use with Matlab/Simulink and Real-Time Workshop [52].

The ordinary Simulink environment has some computational overhead associated with managing the interaction of model blocks with each other and with the integrator [42].

Simulink offers an "accelerator" mode which uses Real-Time Workshop to optimize the Simulink model for speed by compiling the code in a way that eliminates unnecessary calls to the Simulink application program interface (API) [53]. Real-Time Workshop also allows code to be compiled in "external" mode, meaning that the model is compiled on a target computer which communicates to the host computer running Matlab/Simulink. "External" mode is particularly useful for large models for which real-time simulation is only possible using a parallel computing approach. In this case, portions of the overall model can be solved simultaneously on different target machines [53]. There is a small amount of overhead involved with running simulations in the Simulink environment on a Windows computer. Even in the accelerator mode, Simulink requires a fraction of a second to initialize a model before the simulation begins [53]. In addition, the Windows operating system may have processes running in the background which could affect simulation time. In contrast, real-time operating systems, such as QNX, require minimal computer resources to run (i.e. the operating system has very little "overhead") [38].

All Simulink times reported in this thesis were obtained from models run in "accelerator" mode on a PC with Windows XP operating system, 3.2GHz Pentium 4 processor, and 1 GB of RAM. Had the simulations been run in "external" mode on a real-time target machine, simulation times might be marginally faster.

The main goal of this work is to use linear graph theory and symbolic computing to automatically generate vehicle simulation code that can be solved faster than real-time and is suitable for HIL. To ensure that the simulation times reported would be relevant for hardware-in-the-loop applications, Euler's method was used to integrate the ODEs describing the vehicle, and a constant step size of 1 millisecond (ms) was specified. This solver and step size are very typical for HIL applications, as discussed in the literature [12, 40, 42, 48]. Unless explicitly stated, all simulation times for DynaFlexPro models solved in Simulink are associated with Euler's method for solving ODEs and and a constant step size of 1 ms.

For the example problems considered, the vehicle model S-Function used for the steering maneuver is identical to that used for the braking maneuver. Since an explicit constant step size solver is used, we would not expect the simulation times to depend on the maneuver being performed. However, the braking maneuver consistently takes slightly longer to simulate. The reason has to do with the need for a torque calculator block in Simulink, as shown in Figure 5.1. This block is needed because the torque signal applied to the vehicle models is defined in a piecewise manner. The torque calculator block takes the current simulation time as input and outputs the necessary torque signal, adding some computational cost to the total simulation. The sine steer maneuver, by contrast, does not require a torque calculator block and uses a relatively simple sine wave input to define the steer motion and its derivatives, as shown in Figure 5.2.

For timed runs, the only outputs of the Simulink vehicle models are state variables. The tire force and moment responses reported in this chapter were obtained from un-

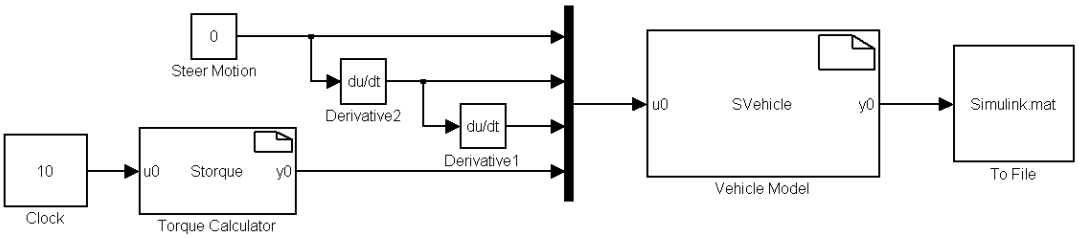


Figure 5.1: Simulink Implementation of Brake Torque

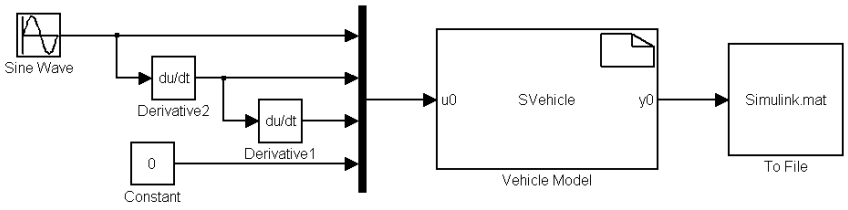


Figure 5.2: Simulink Implementation of Steering Motion

timed runs in Simulink using the identical vehicle model and solver settings as the timed runs, but with additional code associated with defining the additional force and moment outputs.

5.3 Generic 4-Wheeled Vehicle with Independent Suspension

5.3.1 Description

The first example is a 14 degree of freedom 4-wheeled vehicle with independent suspension at every corner. The topology of this vehicle was discussed in Section 2.2.2. It is redisplayed in Figure 5.3, which shows identical left and right steering motions and identical braking torques applied to all 4 wheels. This is the case we will be investigating for our example maneuvers.

Michael Sayers recommended this topology to engineers who want to simulate the essential handling and braking behavior of an automobile without the monumental effort of modeling all of the small details (bushing rates, linkage geometry, etc.) [43]. Since then, it has become the basis for several software packages designed to do real-time vehicle dynamics simulation, including CarSim and ADAMS/Car RealTime [28, 30, 42].

Sayers noted that it is possible to tilt the axis of the prismatic joints to account for the fact that suspension deflection may not be in a purely vertical direction and described a simple method for including extra roll stiffness in the model (which might be provided via anti-roll bars on the actual vehicle). Sayers used look-up tables to provide spring and

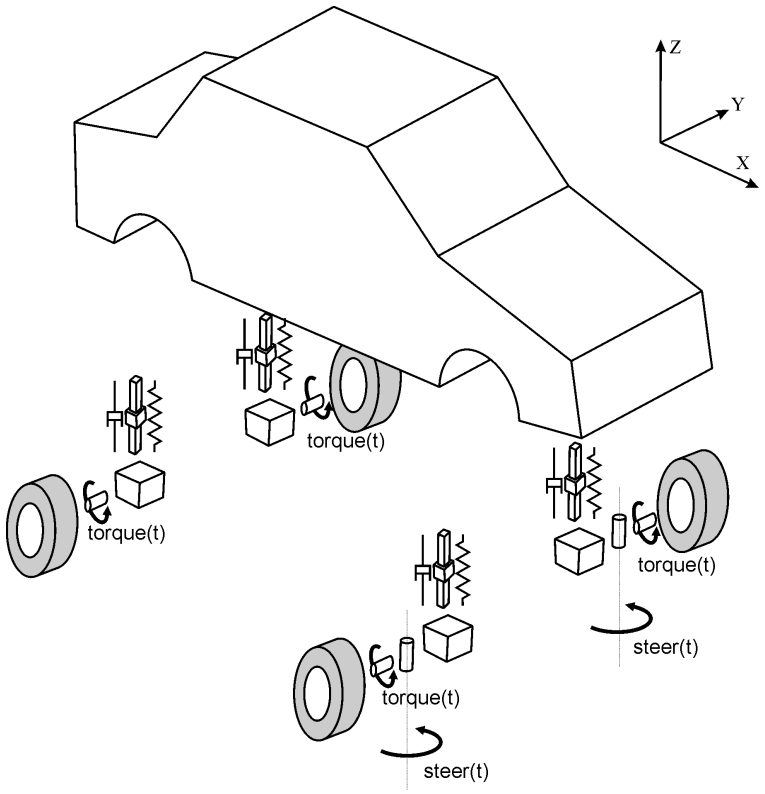


Figure 5.3: Generic Four-Wheeled Vehicle Model with Independent Suspension

damping forces as functions of suspension deflection and to modify the orientation of the wheels as a function of suspension deflection [42].

For this example model, the axes of all prismatic joints are parallel with the z axis of vehicle body's CoM frame, the suspension does not include anti-roll bars, the spring and damping rates are linear, and the wheel orientations are not modified as functions of suspension deflection. The linear graph modeling method described in Chapters 2 and 4 does not prevent the inclusion of any of the complexities added by Sayers. They were left out of this example purely in the interest of simplicity and clarity of presentation.

Tires were included in the DynaFlexPro model using 4 identical instances of the tire component described in Chapter 4. The ADAMS/Pacejka method of evaluating effective rolling radius was used, according to equation (3.13), with parameters found in Table A.5. Tire transient behavior was modeled using extra state variables, as defined by equations (3.22) -(3.25); an external function was used to calculate longitudinal and lateral relaxation lengths as functions of normal force and inclination angle according to the ADAMS stretched string equations (3.27) and (3.28), with parameters located in Table A.6. A Pacejka 2002 tire model function was used to calculate the tire forces and moments. Tire model parameters are listed in Table A.4.

All other model parameters, including masses, rotational inertias, spring and damping

rates, as well as the geometry enforced by rigid arm components, can be found in Tables A.1, A.2, and A.3.

The first step to generating simulation code is to represent the model as a linear graph. The ModelBuilder graphical user interface for DynaFlexPro can be used to automatically assemble a linear graph representation of a system model. This thesis does not discuss the ModelBuilder user interface; linear graphs are presented directly. For a discussion of the graphical user interface, refer to the DynaFlexPro user's manual [19].

A linear graph representation of the generic 4-wheeled vehicle model is presented in Figures 5.4 and 5.5. The graph looks identical at every corner, with the exception that the front corners have extra edges for applying steering motion. For clarity of presentation, only the left front corner is shown in detail. Note that all edges that enforce any type of rotational constraint (this includes rigid arms, prismatic joints, revolute joints, and motion drivers) will have two edges in the rotational domain. The second edge is the one which defines the allowed/enforced rotational motions, while the first edge only exists to ensure proper calculation of the second edge's angular acceleration (refer to the revolute joint example in Section 2.4 for more information).

The edge naming convention used for linear graphs in this chapter is as follows:

- The prefix "RA" identifies an edge as part of a rigid arm component. Rigid arms are used to enforce a constant translation and rotation transformation between two frames.
- The prefix "RB" identifies an edge as part of a rigid body component, which represents a body's inertia and weight in a local gravitational field.
- The prefix "Mot" identifies an edge as part of a motion driver component. Motion drivers are used to enforce specified time-varying translation and rotation between two frames.
- The prefix "Rev" identifies an edge as part of a revolute joint component.
- The prefix "Pris" identifies an edge as part of a prismatic joint component.
- The middle part of the edge name contains a number identifying which corner of the vehicle it belongs to. The front left corner is labeled 1, the front right corner is labeled 2, the rear left corner is labeled 3, and the rear right corner is labeled 4.
- The suffix of the edge name contains either the letter T or R to indicate which domain it belongs to (translational or rotational) as well as a number identifying the edge's order within a certain component.

For the generic 4-wheeled vehicle example, a single road reference frame Q is connected to the Ground frame by a rigid arm component that enforces zero translation and rotation. Therefore, all instances of the tire component are associated with a single road plane (they

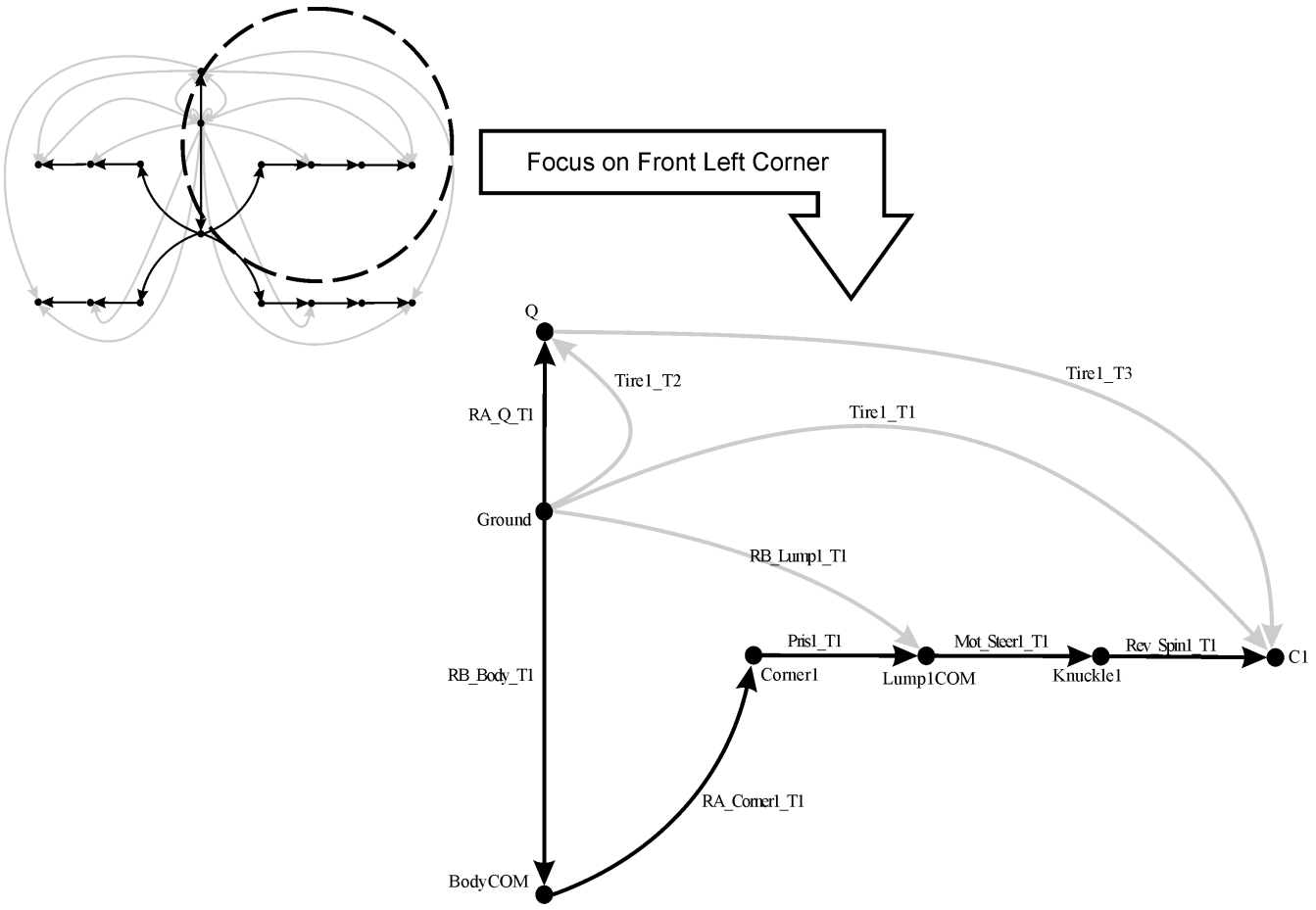


Figure 5.4: Translational Domain Graph of Generic 4-Wheeled Vehicle (all Tire Component Edges in Cotree)

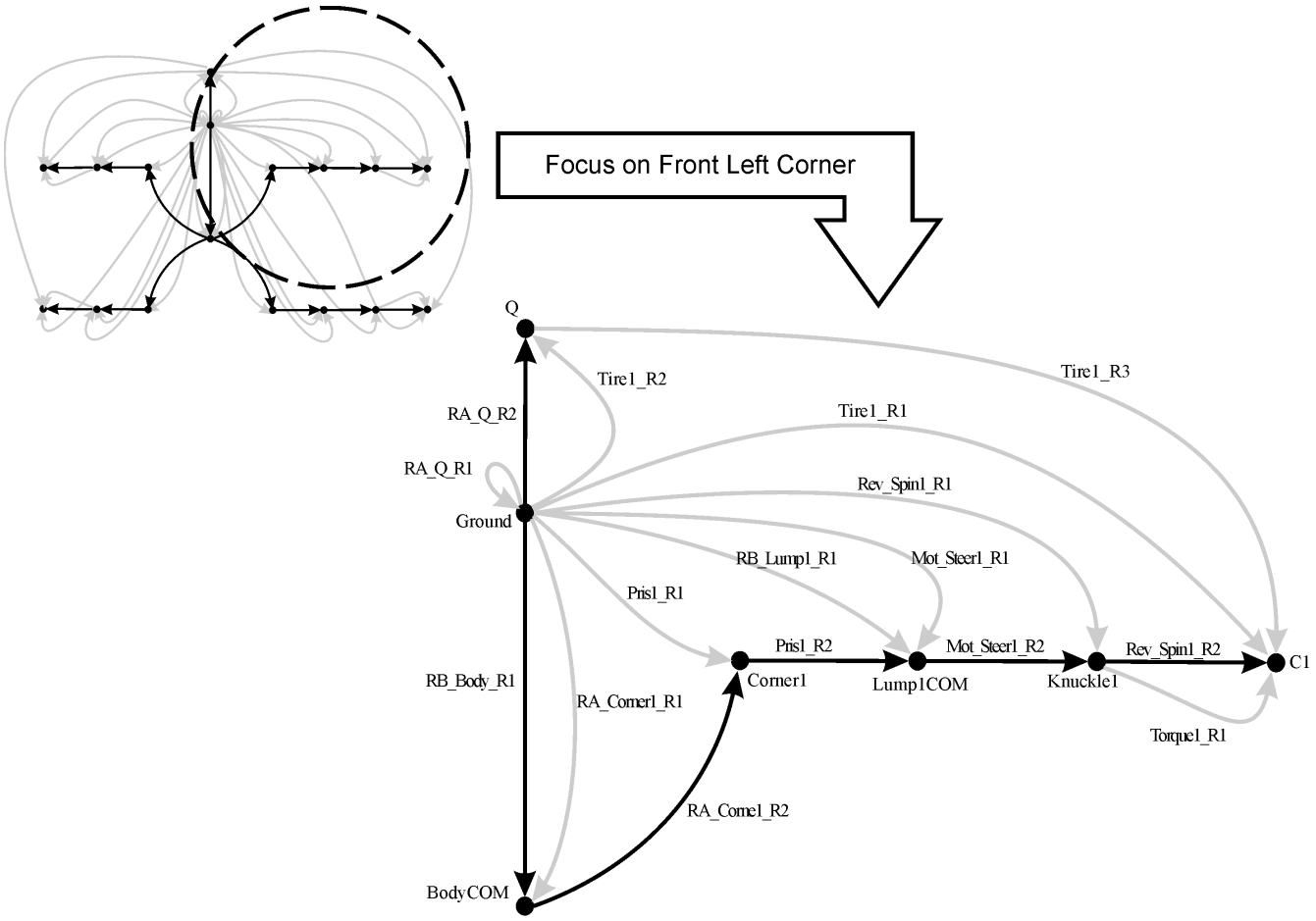


Figure 5.5: Rotational Domain Graph of Generic 4-Wheeled Vehicle (all Tire Component Edges in Cotree)

could have been associated with different road planes), and the road is always stationary and aligned with the Ground frame.

The principle of orthogonality provides a dynamic equation for each across space motion allowed by edges selected into the tree. Vehicle body edges, revolute joint edges, and prismatic joint edges are selected into both the translational and rotational tree for this example, resulting in dynamic equations for the 3D translational and rotational motions of the vehicle body, the extension of each prismatic joint, and the spin of each revolute joint. This gives a total of 14 dynamic equations in terms of generalized speeds and another 14 kinematic transformation equations expressing the derivatives of position variables in terms of generalized speeds. For this example, 8 additional differential equations (2 for each instance of the tire component) are added in order to model tire transient behavior. The end result is a set of $14 + 14 + 8 = 36$ ordinary differential equations (ODEs) and no algebraic equations that describes the vehicle behavior. A list of the 36 state variables, as well as the initial conditions used for this example can be found in Table A.8.

Note that the generalized speeds associated with the vehicle body are components of the its velocity and angular velocity expressed in the body's center of mass (CoM) frame. The coordinates used to track the position of the vehicle body are expressed in the Ground frame, and the coordinates used to track the orientation of the vehicle body with respect to the Ground frame are 321 Euler angles. These generalized speeds and coordinates were selected by choosing the unit vectors used to span the full 3-dimensional across space for edges RB_Body_T1 and RB_Body_R1. The unit vectors used on the 0th derivative level (position) are different than the unit vectors used on the 1st derivative level (velocity).

System equations were formulated in DynaFlexPro, specifying **simplify()** as the type of expression manipulation to be performed on both the dynamic equations and the algebraic constraint equations. Component-level simplification of the tire intermediate expressions was not performed.

5.3.2 Validation using ADAMS

An ADAMS model was constructed which uses all of the same vehicle parameters and tire model details as the DynaFlexPro model, with one exception: in the ADAMS model, a small mass and rotational inertia is assigned to knuckle parts that exist between the front wheel steering joints and the front wheel spin joints. The symbolic formulation used by DynaFlexPro allows joints and motion divers to be connected directly in series, but ADAMS requires there to be a body with some inertia in between these elements.

Sine Steer Maneuver

The first maneuver involves applying an identical sine steer motion to each of the front tires. The motion has an amplitude of 1 degree and a period of 10 seconds, as shown in Figure 5.6. The braking torque on all 4 wheels was set to zero so that the vehicle would

coast through the maneuver.

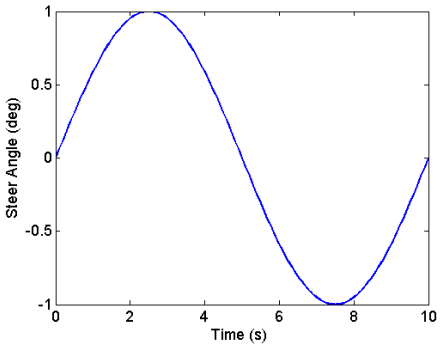


Figure 5.6: Steering Motion Applied to Generic 4-Wheeled Vehicle

Simulation of the DynaFlexPro model was performed in Simulink using Euler's method and a constant step size of 1 ms, and the ADAMS model was simulated using a GSTIFF solver with SI2 formulation and an error tolerance of 1E-3. A convergence study was performed and it was found that neither decreasing the step size in Simulink nor tightening the error tolerance in ADAMS would significantly change the results.

Results for the sine steer maneuver are plotted in Figure 5.7. The sine steer motion causes the vehicle to complete a rather wide lane change, as evident when the vehicle path is viewed in the x-y plane. The yaw rate plotted here is the z component of the vehicle body's angular velocity expressed in the body's CoM frame and the roll angle is the third in a series of 321 Euler angles used to track the orientation of the body CoM frame with respect to the Ground frame. Both the yaw rate and roll angle response have a sinusoidal shape that is similar to the steering input applied.

Steering motions cause tire slip angles to develop and give rise to significant lateral forces and aligning moments. Figure 5.7 includes a plot of F_y and M_z for the front left tire, which shows that the method used by DynaFlexPro/Tire to calculate lateral forces and aligning moments produces extremely similar results to the method used by ADAMS. The Pacejka 2002 tire model used for this example includes a strong dependence of lateral force and aligning moment on normal force. During cornering, the vehicle rolls toward the outside of a turn, increasing the normal force on the outside tires and decreasing the normal force on the inside tires. During the first 5 seconds of simulation, the vehicle is turning to the left; the load on the front left tire decreases as does its capacity to generate lateral forces and aligning moments. During the last five seconds of simulation, the vehicle is turning to the right; the load on the front left tire increases, allowing it to generate significantly greater lateral forces and aligning moments. This is the reason why the peak in the F_y and M_z curves at 7.5-8 seconds is greater than the peak in the F_y and M_z curves at 2.5-3 seconds.

Excellent agreement is observed between the ADAMS model and the DynaFlexPro model for the sine steer maneuver, indicating that, with the options specified for the

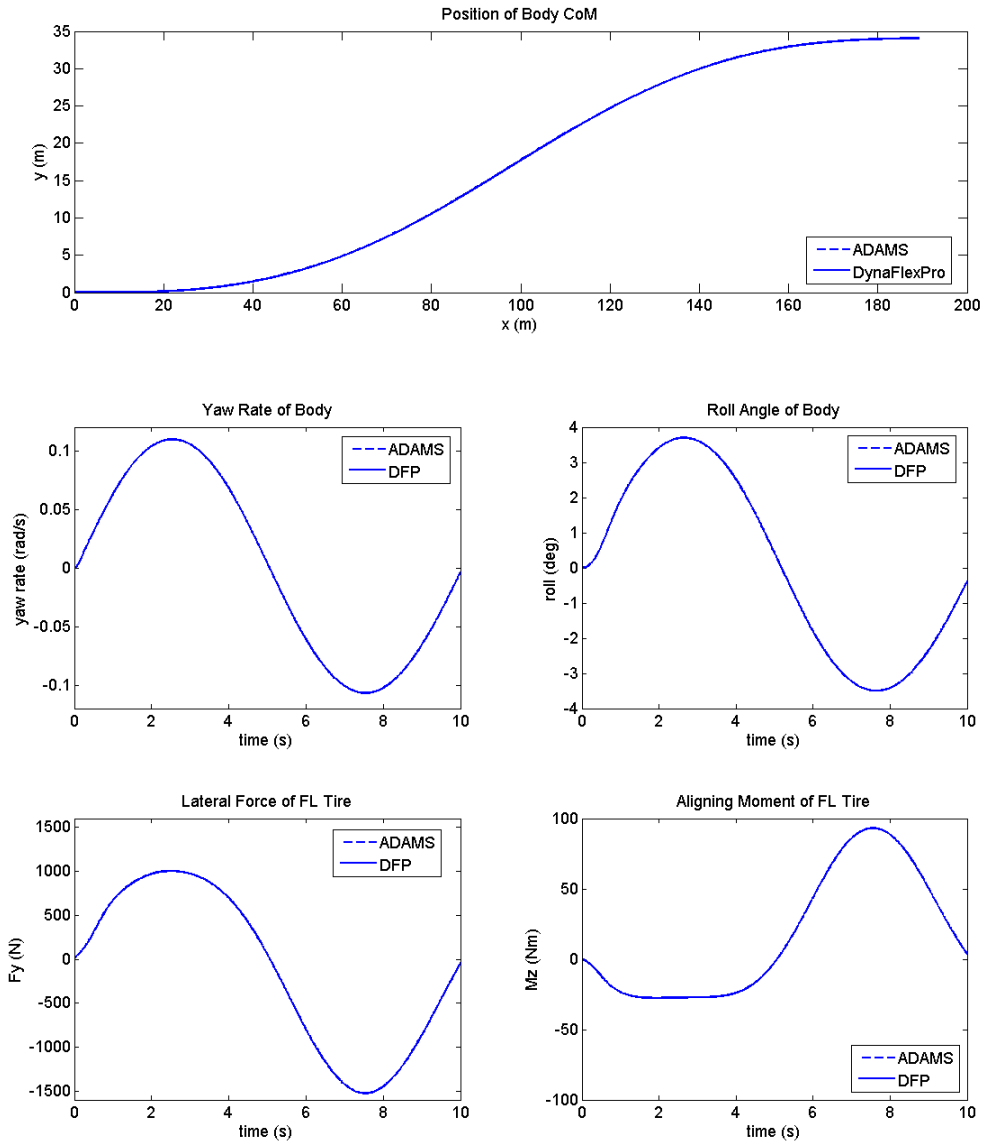


Figure 5.7: Response of Generic 4-Wheeled Vehicle for Sine Steer Maneuver

generic 4-wheeled vehicle model, the implementation of the DynaFlexPro tire component is accurate for maneuvers dominated by steering motions, lateral tire forces, and aligning moments.

Braking Maneuver

The second maneuver involves applying an identical braking torque to all four wheels. The brake torque starts at 0 Nm, ramps to 1000 Nm between time = 2 and time = 2.5 seconds, is held at 1000 Nm until time = 4.5 seconds, and ramps down to 0 Nm by time = 5.0 seconds, as shown in Figure 5.8. The steering motion for both front wheels was set to zero to indicate a driver that is attempting to hold the vehicle in a straight line.

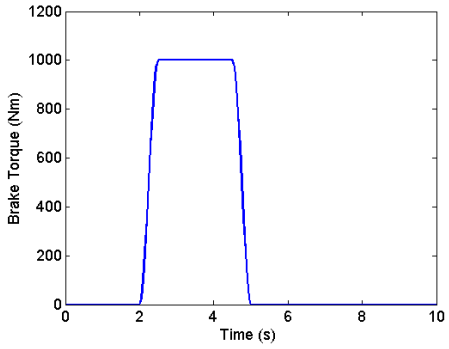


Figure 5.8: Brake Torque Applied to Generic 4-Wheeled Vehicle

Once again, simulation was performed in Simulink using Euler's method and a constant step size of 1 ms, and in ADAMS using the GSTIFF solver with SI2 formulation and an error tolerance of 1E-3.

Results for the braking maneuver are plotted in Figure 5.9. When the braking torque is applied, longitudinal tire forces develop that cause the vehicle to slow from approximately 20 m/s to approximately 7 m/s. During the braking event, the vehicle body pitches forward several degrees. The pitch angle reported is the second in a series of 321 Euler angles used to track the orientation of the body CoM frame with respect to the Ground frame.

The presence of a rolling resistance moment ensures that, when no braking torque is applied, the steady state value of longitudinal tire forces will be negative and the vehicle will continue to slow down. Figure 5.9 highlights the strong dependency of rolling resistance on normal force predicted by the Pacejka 2002 tire model. When the vehicle pitches forward, the normal force on the front tires increase, as does the magnitude of the rolling resistance moment they produce.

Excellent agreement is observed between the ADAMS model and the DynaFlexPro model for the braking maneuver, indicating that, with the options specified for the generic 4-wheeled vehicle model, the implementation of the DynaFlexPro tire component is ac-

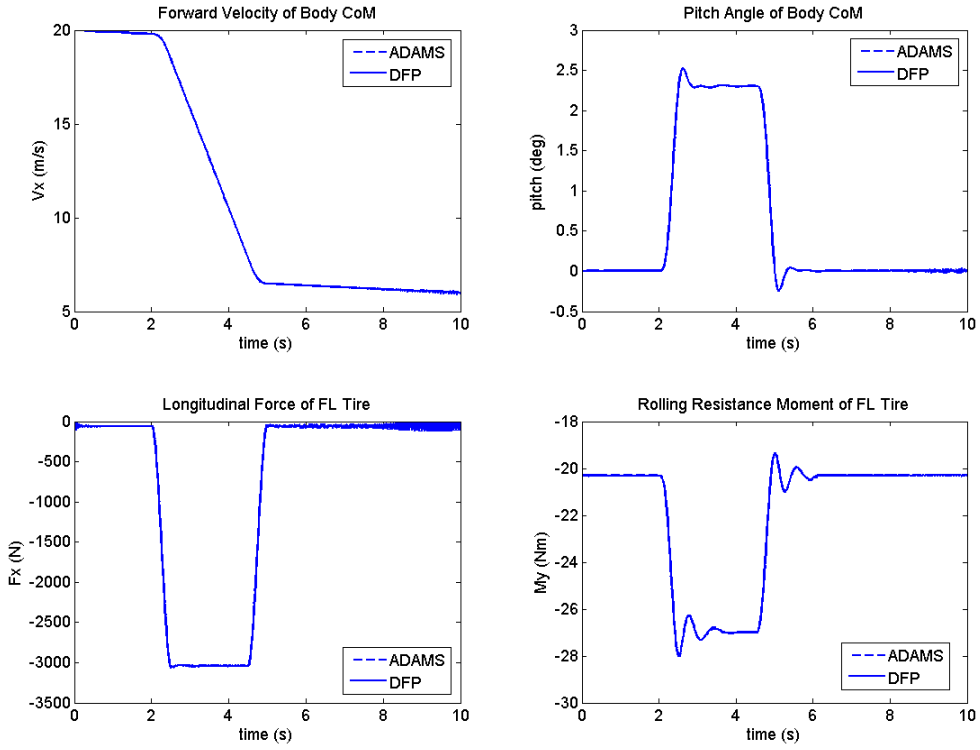


Figure 5.9: Response of Generic 4-Wheeled Vehicle for Braking Maneuver

curate for maneuvers dominated by longitudinal tire forces and rolling resistance.

5.3.3 Simulation Time in Simulink

For the generic 4-wheeled vehicle example, a 10 second sine steer maneuver was simulated in 0.45 seconds (average of 5 runs) and a 10 second braking maneuver was simulated in 0.48 seconds (average of 5 runs). This example model can be simulated more than 20 times faster than real-time using an explicit constant step size solver, which makes it very suitable for hardware-in-the-loop applications.

As a point of comparison, we may consider that it takes 4.33 seconds to simulate the sine steer maneuver and 7.97 seconds to simulate the braking maneuver in ADAMS using the GSTIFF solver with SI2 formulation and an error tolerance of 1E-3. This is significantly slower than the DynaFlexPro models being solved in Simulink with Euler's method and a step size of 1 ms.

It is normal for different maneuvers to take different amounts of time to simulate in ADAMS. GSTIFF is an implicit DAE solver. All of the solvers available in ADAMS are either implicit, operate with variable step sizes, or both [31]. The number of iterations required to meet the specified error tolerance and choose the next step size could vary

considerably with the state of the vehicle, and hence with the maneuver being performed. There is no provision for limiting the number of iterations performed at each time step, making the ADAMS solvers inappropriate for hardware-in-the-loop simulations, as discussed in Section 2.2.1.

5.3.4 Component-Level Simplification for Tire Intermediates

As discussed in Section 4.7, expression manipulation rules can be applied to the symbolic expressions obtained for tire intermediates. Different simplification schemes were applied to the tire intermediates of the generic 4-wheeled vehicle model. These simplification schemes affect the time required to generate optimized simulation code and the time required to simulate the test maneuvers in Simulink. These effects are summarized in Table 5.1.

The generation of simulation code involves several steps, including constructing a computational sequence to calculate intermediate variables in terms of state variables. Table 5.1 shows that the time necessary to generate simulation code is heavily influenced by the type of expression manipulation routine applied to the symbolic expressions for tire intermediates. The times listed there are each an average of 5 runs. When no simplification is used on the tire intermediate expressions, the time to generate simulation code is relatively low. Repeated use of Maple's **simplify()** command nearly doubles the time necessary to generate simulation code, and repeated use of Maple's **combine()** command triples it. This is to be expected since simplifying (or attempting to simplify) symbolic expressions is a computationally intensive task.

The cost of simulation code refers to the number of additions, multiplications, divisions, and function calls contained within the optimized simulation code generated by DynaFlexPro. It may seem surprising that use of the **simplify()** and **combine()** commands on tire intermediate expressions actually increases the cost of the simulation code. However, it should be noted that the computational sequence that makes up block 1 of the simulation code has a structure with several repeated terms (e.g. the components of $\mathbf{u_{SymAxis}}$ in Table 4.3) and naturally lends itself to code optimization. The **simplify()** and **combine()** commands may reduce the complexity of individual expressions, but they destroy the structure of the original computational sequence, making it less suitable for code optimization. For the generic 4-wheeled vehicle model, it is better not to use Maple's simplification functions on the tire intermediate expressions. This conclusion is supported by simulation times for the model obtained from Simulink. Here again, an average of 5 runs is used. The simulation times are longer when **simplify()** or **combine()** have been used on the tire intermediate expressions.

Intermediate Simplification Type	Time to Generate Sim. Code (s)	Cost of Sim. Code				Simulation Time (s)	
		+	×	÷	f	Sine Steer	Braking
none	13.5	1293	1773	24	61	0.45	0.48
simplify	23.8	2365	3889	28	118	0.51	0.53
combine	40.2	2867	4539	24	192	0.59	0.63

Table 5.1: Component-Level Simplification Applied to Tire Intermediates for Generic 4-Wheeled Vehicle

5.3.5 Selecting a Tire Component Edge to the System Tree

Thus far, all of the tire component edges have been placed in the system cotree. We now consider a different tree selection for the generic 4-wheeled vehicle model, in which edges T1 and R1 of every tire component are selected into the system tree.

As defined in Chapter 2, a tree is a connected subgraph that includes all of the nodes in a graph without any loops. In order to maintain a valid translational tree, the prismatic joint edges are placed in the translational cotree, as shown in Figure 5.10, with the consequence that motions along the prismatic joint axes will no longer be considered as state variables. In order to maintain a valid rotational tree, the revolute joint edges are placed in the rotational cotree, as shown in Figure 5.11, with the consequence that revolute joint rotations will no longer be considered as state variables.

With tire component edges T1 and R1 in the tree, the position, orientation, velocity, and angular velocity of each tire center frame with respect to the Ground frame will be state variables. This shortens the chain of branches whose across variables must be added to obtain symbolic expressions for $\vec{r}_{C/Q}$, $\vec{v}_{C/Q}$, $\vec{\omega}_{C/Q}$, and $\mathbf{R}_{C/Q}$ in terms of state variables, as described in Section 4.5.1. Therefore, selecting tire edges into the tree should decrease the complexity of the symbolic expressions for tire intermediates and reduce the computational cost associated with block 1 of the simulation code.

For each tire component edge selected into the system tree, 3 new dynamic equations will be formulated because all tire component edges have a full across space. Selecting 8 tire component edges to the tree (T1 and R1 for 4 tires) introduces 24 dynamic equations. Adding the 6 dynamic equations related to rotational and translational motion of the vehicle body, we arrive at a total of 30 dynamic equations used to describe the 14 degree of freedom generic 4-wheeled vehicle model. There will be $30-14 = 16$ algebraic constraint equations necessary to enforce the dependencies of the chosen coordinates. The system is described by a set of 30 dynamic equations and 16 algebraic equations. We should expect the system equations (block 3 of the simulation code) to be more costly to evaluate than the case where all tire edges were in the cotree and the dynamic equations were a minimal set of 14 ODEs.

Several methods of solving DAEs have been presented in the literature [4, 9, 14, 17].

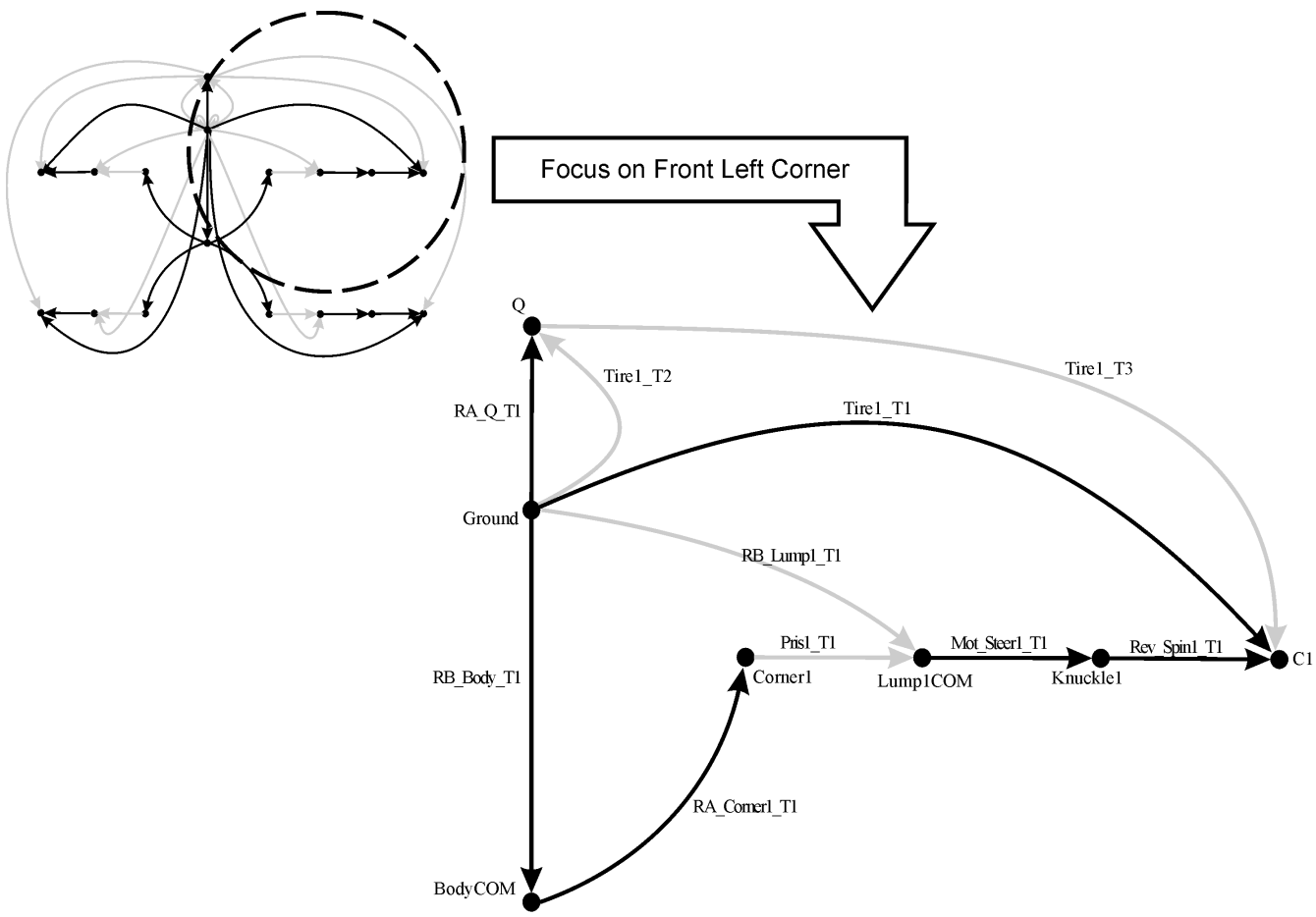


Figure 5.10: Translational Domain Graph of Generic 4-Wheeled Vehicle (Edge T1 of Tire Component in the Tree)

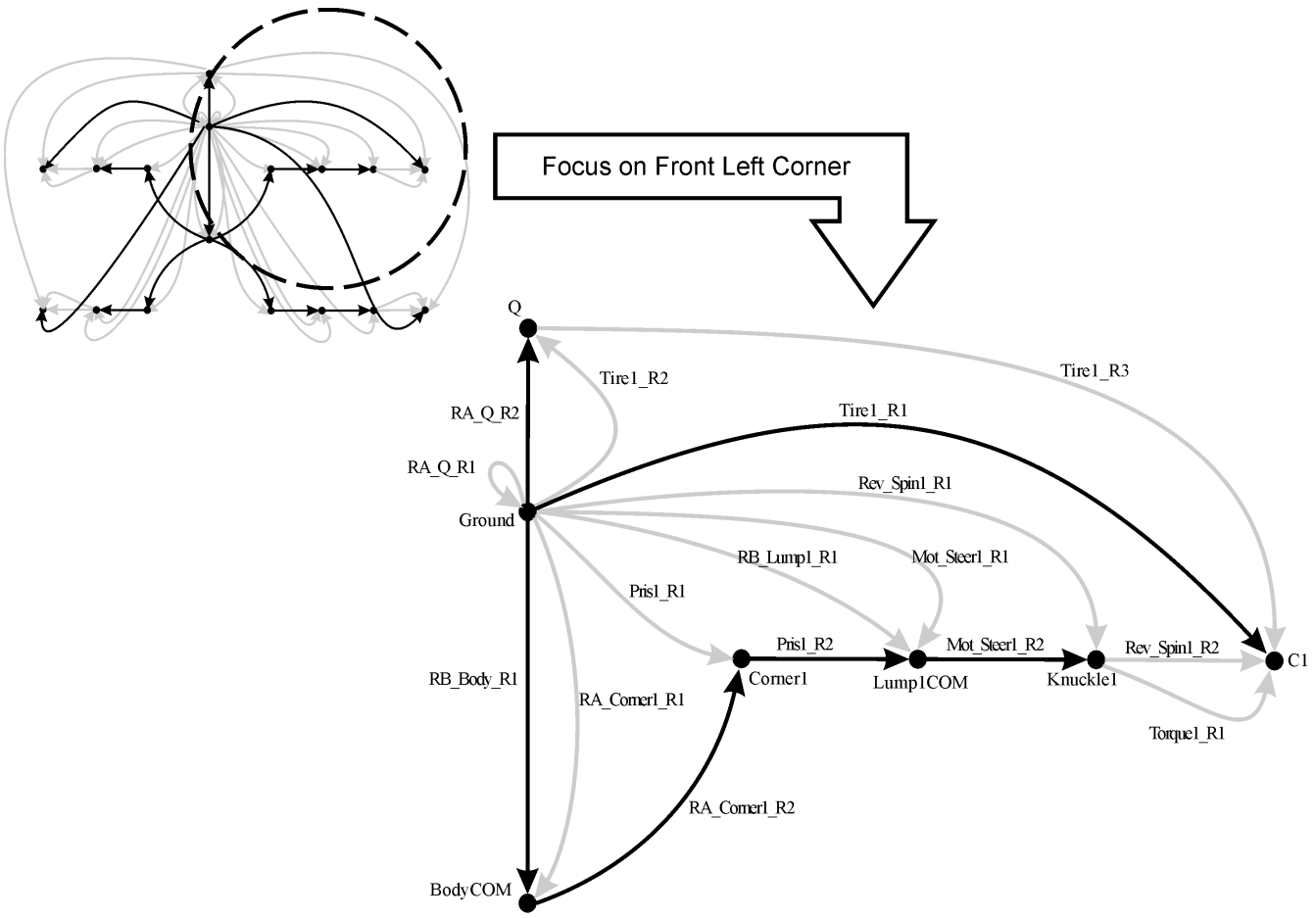


Figure 5.11: Rotational Domain Graph of Generic 4-Wheeled Vehicle (Edge R1 of Tire Component in the Tree)

Placement of	Cost of Sim Code				Simulation Time (s)	
Tire Edges	+	×	÷	f	Sine Steer	Braking
cotree	1293	1773	24	61	0.45	0.48
tree	5644	9611	37	95	0.65 *	0.65 *

* Results have not converged with Euler's method and a step size of 1 ms

Table 5.2: Effect of Tree Selection on Simulation Code and Simulation Times for Generic 4-Wheeled Vehicle Model

In this thesis, DAEs are converted to ODEs by differentiating the algebraic constraint equations twice and Baumgarte constraint stabilization is used to keep position-level constraint violation from growing out of control (the method was described in Section 2.1.2). For the generic 4-wheeled vehicle example, parameters $\alpha = 20$ and $\beta = 20$ were chosen for the Baumgarte constraint stabilization.

Along with a different tree selection comes a different list of state variables. With edges T1 and R1 selected into the tree for each tire, the system equations are based on 30 generalized speeds $\mathbf{p}$, 30 coordinates $\mathbf{q}$, and 8 additional state variables used to model lag in tire slip parameters, for a grand total of 68 state variables. The initial conditions for these state variables are listed in Table A.9. They were chosen to be equivalent to the initial conditions used for validation versus ADAMS in Section 5.3.2.

For the case where tire edges are selected into the system tree, the system equations (block 3 of the simulation code) should be more costly to evaluate and the intermediate computational sequence (block 1 of the simulation code) should be less costly to evaluate, compared to the case where tire edges are selected to the cotree. Table 5.2 shows that the increased cost of the system equations far outweighs the decreased cost of the tire component intermediate expressions, resulting in an overall increase in the cost of the simulation code.

Costlier simulation code results in longer simulation times for the case where tire edges are selected into the tree. The times reported in Table 5.2 are the average of 5 Simulink runs using Euler's method and a fixed step size of 1 ms. These times are somewhat misleading, however. The need for Baumgarte constraint stabilization to control the error in the position level constraint equations results in significant stiffening of the differential equations used to describe the system. While a step size of 1 ms is sufficient to ensure a converged solution when simulation code is generated with tires in the cotree, it is not sufficient to ensure a converged solution when the simulation code is generated with tire edges in the tree. In that case, a step size of 0.02 ms was needed for convergence, as shown in Figure 5.12, which focuses on the yaw rate response for the sine steer maneuver.

Reducing the step size to 0.02 ms resulted in a simulation time of 32.67 seconds for the 10 second sine steer maneuver. When tires edges are selected into the system tree for the generic 4-wheeled vehicle, Baumgarte constraint stabilization is used to handle the resulting DAEs, and Euler's method is used to solve the equations, the simulation code

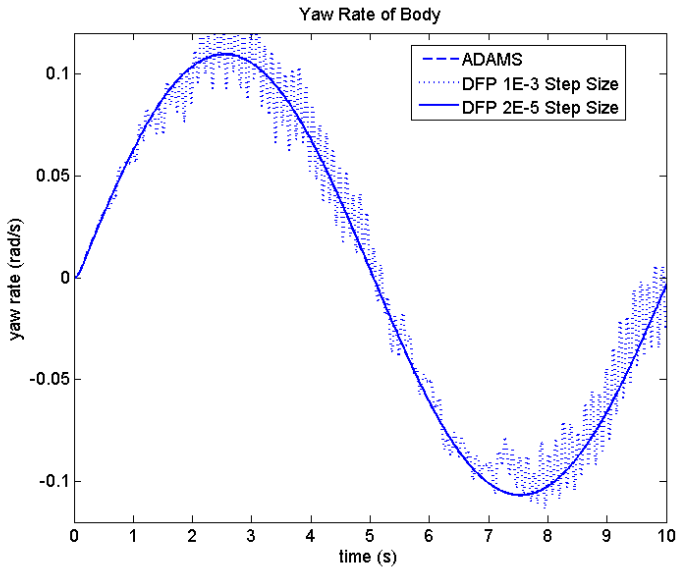


Figure 5.12: Yaw Rate Response of Generic 4-Wheeled Vehicle for Sine Steer Maneuver – Simulation Code Generated with Tire Edges T1 and R1 in Tree

is not suitable for real-time applications because of the extremely small step sizes that must be used to ensure an accurate solution.

Other methods of solving DAEs might work better for the generic 4-wheeled vehicle model with tire tire edges are selected into the tree. The position level algebraic constraint equations could be solved within the simulation code using a Newton/Raphson algorithm, as was done by Fisette et al. [14]. As long as the number of iterations spent on a single time step is not allowed to grow too large, these methods have the potential to be used for HIL applications. In this case, block 3 of the simulation code must be modified to contain a Newton/Raphson loop and some kind of intelligence to limit the number of iterations performed within that loop.

The use of an implicit solver for systems containing algebraic constraints, whether the constraints are solved on the acceleration level with Baumgarte constraint stabilization or on the position level with a modified Newton/Raphson algorithm, should also be investigated. The number of iterations that the implicit solver uses to predict a solution at the next time step must be limited to a fixed number if the simulation is to be used for HIL applications where synchronization with hardware is important.

Implementation and testing of modified Newton/Raphson algorithms and modified implicit solvers is left for future research.

5.3.6 The Effect of Tire Model Complexity

An analyst might question how much computational effort is needed to evaluate the main simulation code — the bold block in Figure 4.3 — compared to the computational effort needed to evaluate external functions. It is an important question to ask when building vehicle models. Some tire model functions are long and complex, and their use may affect the suitability of a vehicle model for real-time applications.

Because tire model functions frequently use "if" constructs, the cost of evaluating a tire model function can depend on the values of the arguments being passed to it. That being said, a reasonable cost measure can be obtained by counting the number of additions, multiplications, divisions, and functions that are needed to evaluate a tire model function when certain nominal arguments are passed to it ($S \approx 0$, $\alpha \approx 0$, $\gamma \approx 0$, $\Omega > 0$, $V_{Cx} > 0$). This type of cost evaluation was performed on the Fiala and Pacejka 2002 tire model functions, with results presented in Table 5.3. The Pacejka 2002 tire model function is much more costly to evaluate than the Fiala tire model function.

The DynaFlexPro simulation code (and corresponding Simulink S-Function) used to describe the generic 4-wheeled vehicle were rebuilt using a Fiala tire model whose parameters were chosen to be equivalent to the Pacejka 2002 tire model used for validation versus ADAMS. Parameters for the Fiala tire model can be found in Table A.7. Only the tire model was changed. All of the other tire component options remain the same. The ADAMS/Pacejka method of evaluating effective rolling radius is still used, as is the the stretched string method of calculating derivatives of tire slip state variables. No component-level simplification was used for tire intermediate expressions. All tire component edges are in the cotree, resulting in a model described by a set of ODEs with no algebraic constraint equations.

The Pacejka tire model requires a list of seven intermediate variables as input (F_z, S, α, γ, Ω, R_{eff}, and V_{Cx}), whereas the Fiala tire model needs only four (F_z, S, α, Ω). Because the extra three intermediate variables are not needed for the Fiala tire model function, these variables will be candidates for removal during the code optimization process. However, as described in Section 4.5.3, the values of γ, R_{eff}, and V_{Cx} calculated early in block 1 are used later in block 1 to calculate S, α, Ω. Thus, even though the Fiala tire model function takes fewer arguments than the Pacejka tire model function, all of the same intermediate variables need to be assigned at some point in the computational sequence that makes up block 1 of the simulation code. Therefore, we should not expect

	Cost of Tire Model Function			
Tire Model	$+$	$\times$	$\div$	f
Pacejka 2002	84	205	23	46
Fiala	16	33	5	11

Table 5.3: Computational Cost of Tire Model Functions

	Cost of Sim. Code					Simulation Time (s)	
Tire Model	+	×	÷	f	assign.	Sine Steer	Braking
Pacejka 2002	1293	1773	24	61	1470	0.45	0.48
Fiala	1233	1684	24	61	976	0.30	0.36

Table 5.4: Effect of Tire Model on Simulation Code and Simulation Times for Generic 4-Wheeled Vehicle

a large difference in the cost of simulation code generated using a Pacejka tire model, versus simulation code generated for the same vehicle using a Fiala tire model. Table 5.4 confirms that the number of additions, multiplications, divisions, and function calls present in the optimized simulation code does not change appreciably when the tire model function is changed from Pacejka 2002 to Fiala.

Even though block 1 of the simulation code will remain relatively constant regardless of whether a Fiala or Pacejka tire model is used, block 2 of the simulation code will be very different. When a Pacejka 2002 tire model function is called, 117 parameters must be assigned for each tire in the system, but when a Fiala tire model function is called, only 6 parameters need to be assigned for each tire. Thus, the total number of assignment statements in the simulation code will be much higher when the Pacejka 2002 tire model is used, as shown in Table 5.4

The time to simulate the sine steer and braking maneuvers in Simulink show that using the simpler Fiala tire model on the generic 4-wheeled vehicle results in approximately 30% improvment in simulation time compared to when the more complex Pacejka 2002 tire model is used. Using the Pacejka model results in a slower simulation because of the large number of tire parameters that must be assigned in the simulation code and passed to the tire model function, and also because of the higher computational cost of the tire model function itself (keep in mind that the tire model function gets called 4 times every time step — once for each tire).

The choice of tire model can have a significant effect on simulation times for a complete vehicle model. However, computational cost is not the only factor that must be taken into account when selecting an appropriate tire model. The tire model must be capable of predicting tire behavior, and thus vehicle behavior, to the desired level of accuracy. In order to better evaluate the accuracy/efficiency trade-off, simulation results for the generic 4-wheeled vehicle with Fiala tire model were plotted with simulation results for the same vehicle with a Pacejka 2002 tire model. Results are shown in Figures 5.13 and 5.14.

Results for the braking maneuver are insensitive to the change in tire model. Evidently, with the tire parameters used for this example, the Pacejka tire model predicts longitudinal forces that are very similar to the Fiala tire model. This also has much to do with the type of maneuver being performed. The braking maneuver does not include any lateral tire slip; it is dominated by longitudinal tire slip. As discussed in Chapter 3, the Fiala model

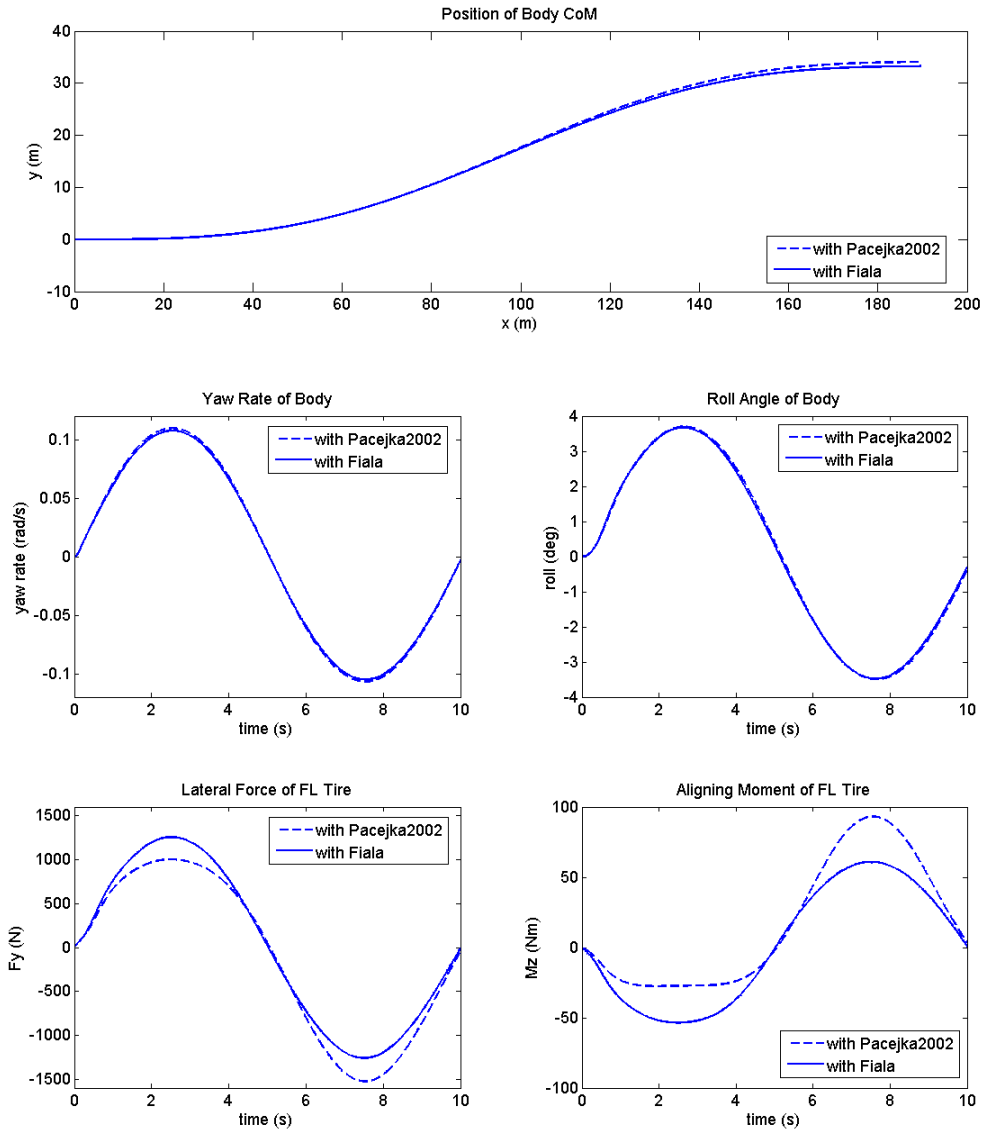


Figure 5.13: Sine Steer Results for DynaFlexPro Model with Different Tire Models

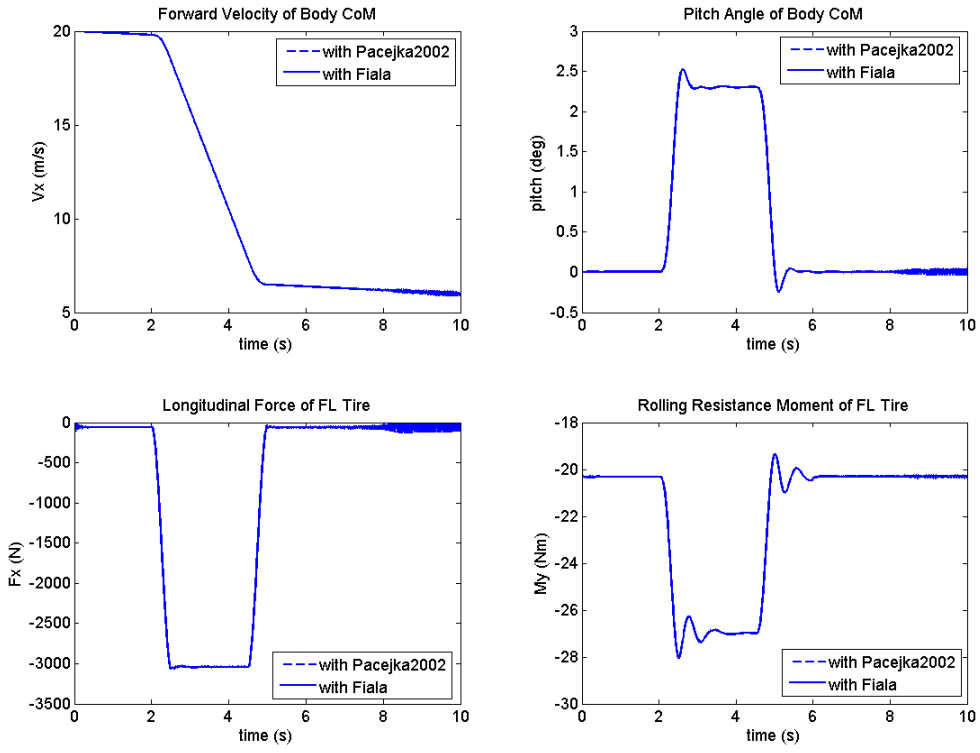


Figure 5.14: Braking Results for DynaFlexPro Model with Different Tire Models

does not have the ability to accurately model combined slip conditions (where significant levels of both lateral and longitudinal slip are present), while the Pacejka 2002 tire model does have the ability to deal with combined slip. This test maneuver does not shine a spotlight on that difference.

Results for the sine steer maneuver are slightly more sensitive to the change in tire model. As shown in Figure 5.13, the two tire models predict different lateral forces and aligning moments for the front left tire. For the Pacejka tire model, the calculation F_y and M_z used is heavily influenced by normal force and inclination angle, while these factors do not have much of an influence on the values of F_y and M_z calculated by the Fiala tire model. However, during a cornering event, the normal force on the outside tire increases by roughly the same amount that the normal force on the inside tire decreases, causing opposite effects on the lateral forces and aligning moments generated by the inside and outside tires. This is the reason why, although the values of F_y and M_z predicted for an individual tire might be different for the Pacejka and Fiala tire models, the overall vehicle response, as indicated by the x-y path, yaw rate, and roll angle, is quite similar.

When making the choice between a simple Fiala tire model or a more complicated Pacejka tire model, the analyst should ask questions like "What is the maneuver being simulated?", "What responses are important to me?", "Are the simulation time savings associated with a simpler tire model worth some loss of fidelity?", and "Do I have enough tire data to populate a more complicated tire model, or am I restricted to a simpler one?". These are questions that must be addressed by an educated analyst, in a certain context, when the specific goals of the simulation are well-defined. The purpose of this work is not to provide modeling advice to engineers, but rather to provide them with a process for handling pneumatic tires in a linear graph framework, implemented as a software tool that automatically generates efficient simulation code *after* the engineer has decided how to model the vehicle. From that point of view, I can only provide some indications about the simulation time penalties associated with certain tire models, and point out the strengths and weaknesses of each. It is left up to the analyst to make the tough modeling decisions.

5.4 Articulated Forestry Skidder

5.4.1 Description

The second example problem involves building a DynaFlexPro model of a Timberjack grapple skidder that was previously modeled in ADAMS as part of a stability study [18]. A top view of the vehicle topology is displayed in Figure 5.15. The vehicle body is split into two halves connected by a revolute joint that allows articulating motion. The vehicle steers by rotating the two halves of the body with respect to each other. There is no suspension; the only compliance is provided by large pneumatic tires [18]. The front and rear axles feature open differentials, which allow all tires to rotate at different speeds. For this model, the tire spin degrees of freedom are provided by 4 revolute joints that connect the tires to the rest of the vehicle.

The articulated forestry skidder is an excellent example of a non-standard vehicle topology that would be impossible to model using commercial packages like CarSim and ADAMS/Car RealTime, which have pre-defined simulation code based on a generic vehicle topology.

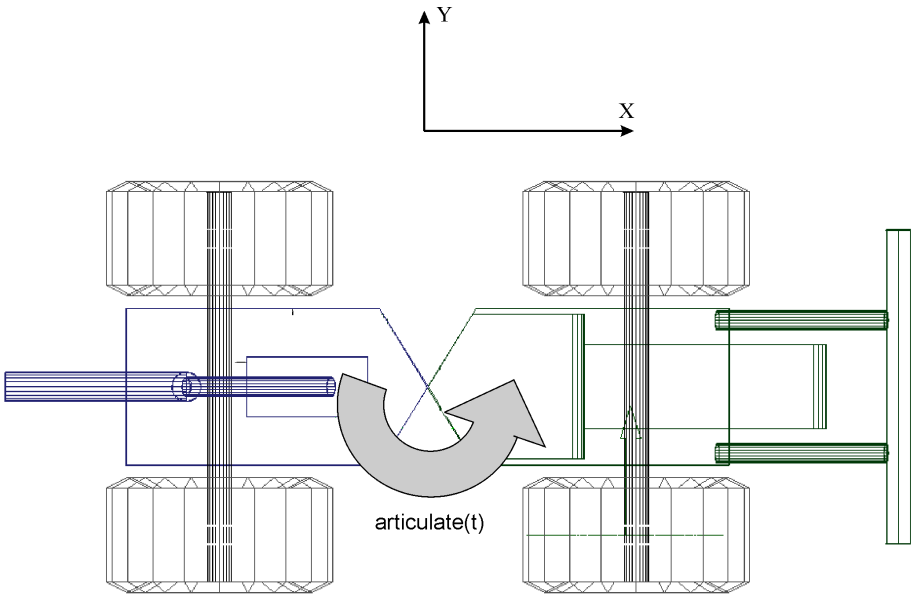


Figure 5.15: Articulated Forestry Skidder

The tire component options used in DynaFlexPro are meant to mimic the tire model used in ADAMS. The effective rolling radius is assumed equal to the loaded radius. Tire lag is not included in this model; the slip parameters are calculated based on the purely kinematic relationships given by equations (3.20) and (3.21). Tire forces and moments are evaluated using the Fiala tire model function. Different Fiala parameters are specified for the front and rear tires, as shown in Table A.13.

All other model parameters, including masses, rotational inertias, tire stiffness, and geometry enforced by rigid arm components can be found in Tables A.10, A.11, and A.12.

The linear graph representation of the forestry skidder is shown in Figures 5.16 and Figures 5.17. The graph for the front half of the vehicle looks identical to the graph for the rear half of the vehicle, with the exception that the front body is selected into the translational and rotational trees while the rear body is not. For clarity of presentation, only the front half of the vehicle is shown in detail. A single road reference frame Q is connected to the Ground frame by a rigid arm component that enforces zero translation and rotation, so the tires are associated with a road that is always stationary and aligned with the Ground frame.

The skidder model has 10 degrees of freedom: 6 for the 3D translational and rotational motion of the front body, and 4 for the spin of the tires. The articulating joint does not add a degree of freedom since its motion is a known function of time that will be specified during the simulated maneuvers. In the linear graph, the articulating joint is represented by a motion driver component. Torque driver edges in the rotational domain allow for the application of a braking torque to each of the 4 wheels. Selecting front body edges and revolute joint edges into the translational and rotational trees ensures that DynaFlexPro will use their across variables to construct the system equations. For this tree selection, the 10 DOF vehicle will be modeled using 10 position co-ordinates and 10 generalized speeds. The system equations will be a set of 20 first order differential equations with no algebraic constraints.

Note that the generalized speeds associated with the front body are components of its velocity and angular velocity expressed in the front body's CoM frame. The coordinates used to track the position of the front body are expressed in the Ground frame, and the coordinates used to track the orientation of the front body with respect to the Ground frame are 321 Euler angles. These generalized speeds and coordinates were selected by choosing the unit vectors used to span the full 3-dimensional across space for edges RB_FBody_T1 and RB_FBody_R1. Different unit vectors were chosen for the 0th (position) and 1st (velocity) derivative levels.

System equations were formulated in DynaFlexPro, specifying **simplify()** as the type of expression manipulation to be performed on both the dynamic equations and the algebraic constraint equations. Component-level simplification of the tire intermediate expressions was not performed.

5.4.2 Validation using ADAMS

Sine Steer Maneuver

A wide lane change was simulated by specifying a sinusoidal articulating motion. The motion had an amplitude of 5 degrees and a period of 10 seconds, as shown in Figure 5.18. The braking torque on all 4 wheels was set equal to zero so that the vehicle would

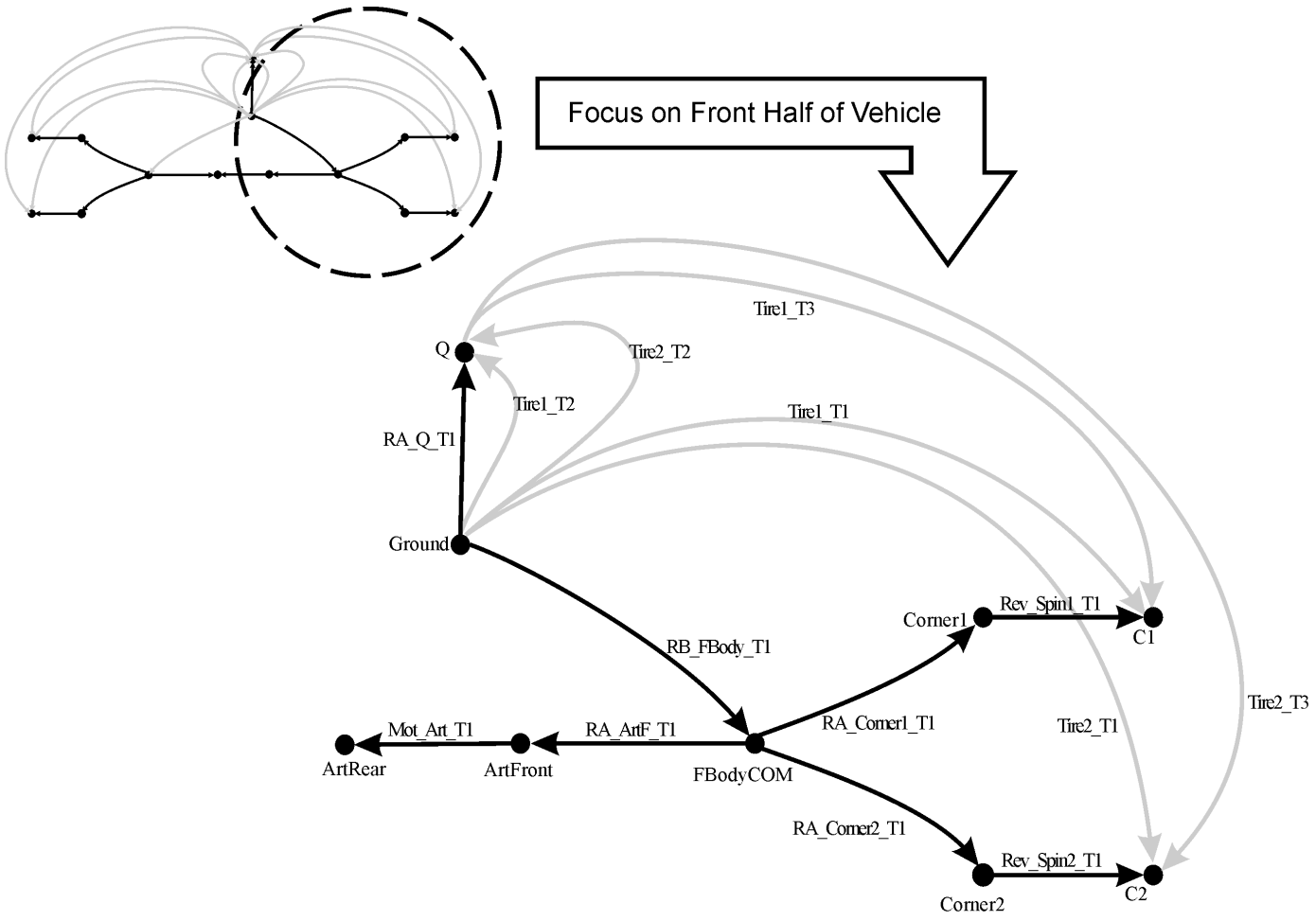


Figure 5.16: Translational Domain Graph of Articulated Forestry Skidder (all Tire Component Edges in Cotree)

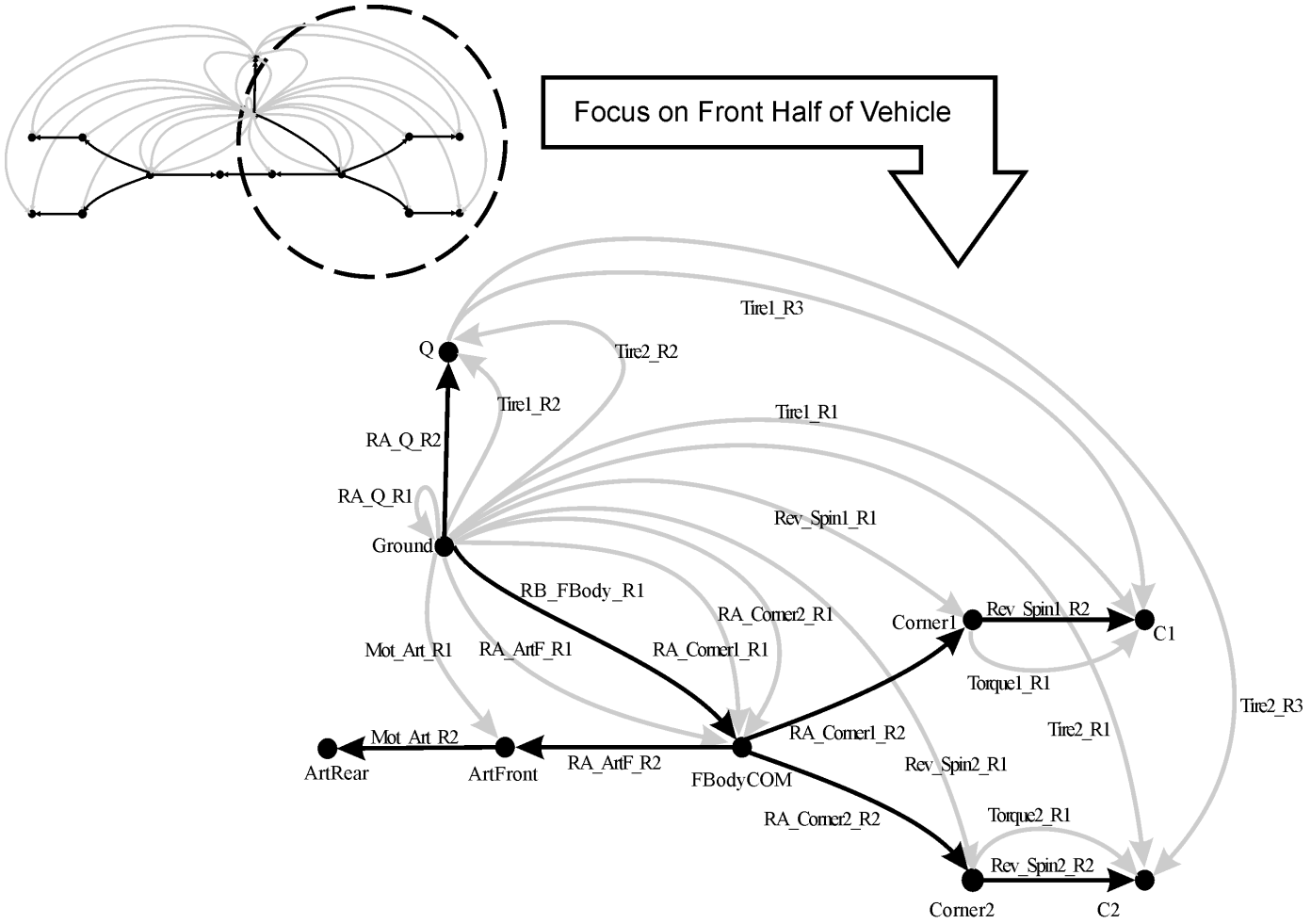


Figure 5.17: Rotational Domain Graph of Articulated Forestry Skidder (all Tire Component Edges in Cotree)

coast through the maneuver. The initial forward speed of the front body, expressed in the front body CoM frame, was set to 10 m/s. Initial conditions for all other state variables are listed in Table A.14.

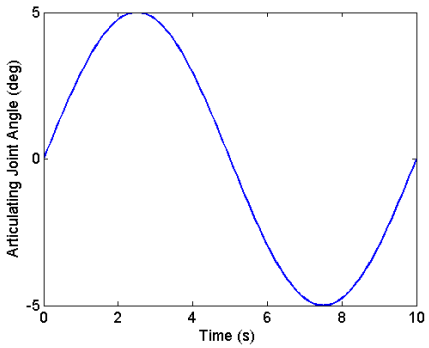


Figure 5.18: Articulating Motion Applied to Forestry Skidder Model

As with previous examples, simulation code was exported from DynaFlexPro in the C language and solved in Simulink's "accelerator" mode using the Euler solver and a constant step size of 1 ms. The ADAMS model was solved using the GSTIFF solver with an SI2 formulation and an error tolerance of 1E-3. A convergence study confirmed that neither decreasing the step size in Simulink nor tightening the error tolerance in ADAMS would significantly change the results.

Results for the sine steer maneuver are plotted in Figure 5.19. The vehicle completes a lane change, as evident when the path of the front body CoM is viewed in the x-y plane. The yaw rate and roll angle plots have a sinusoidal shape similar to the articulating motion that was applied. The articulating motion gives rise to lateral tire slip, and therefore significant lateral forces and aligning moments. Figure 5.19 includes a plot of F_y and M_z for the front left tire, which shows that the method used by DynaFlexPro/Tire to calculate lateral forces and aligning moments produces extremely similar results to the method used by ADAMS.

Excellent agreement is observed between the ADAMS and DynaFlexPro models, indicating that, with the tire options specified for the forestry skidder model, the DynaFlexPro tire component behaves accurately for maneuvers dominated by steering motions, lateral tire forces, and aligning moments.

Braking Maneuver

A braking maneuver was simulated by applying an identical torque to all four wheels. The brake torque starts at 0 Nm, ramps to 20000 Nm between time = 2.0 seconds and time = 2.5 seconds, is held at 20000 Nm until time = 3.5 seconds, and ramps down to 0 Nm by time = 4.0 seconds, as shown in Figure 5.20. The articulating motion was held constant at zero degrees in an attempt to force the vehicle to move forward in a straight

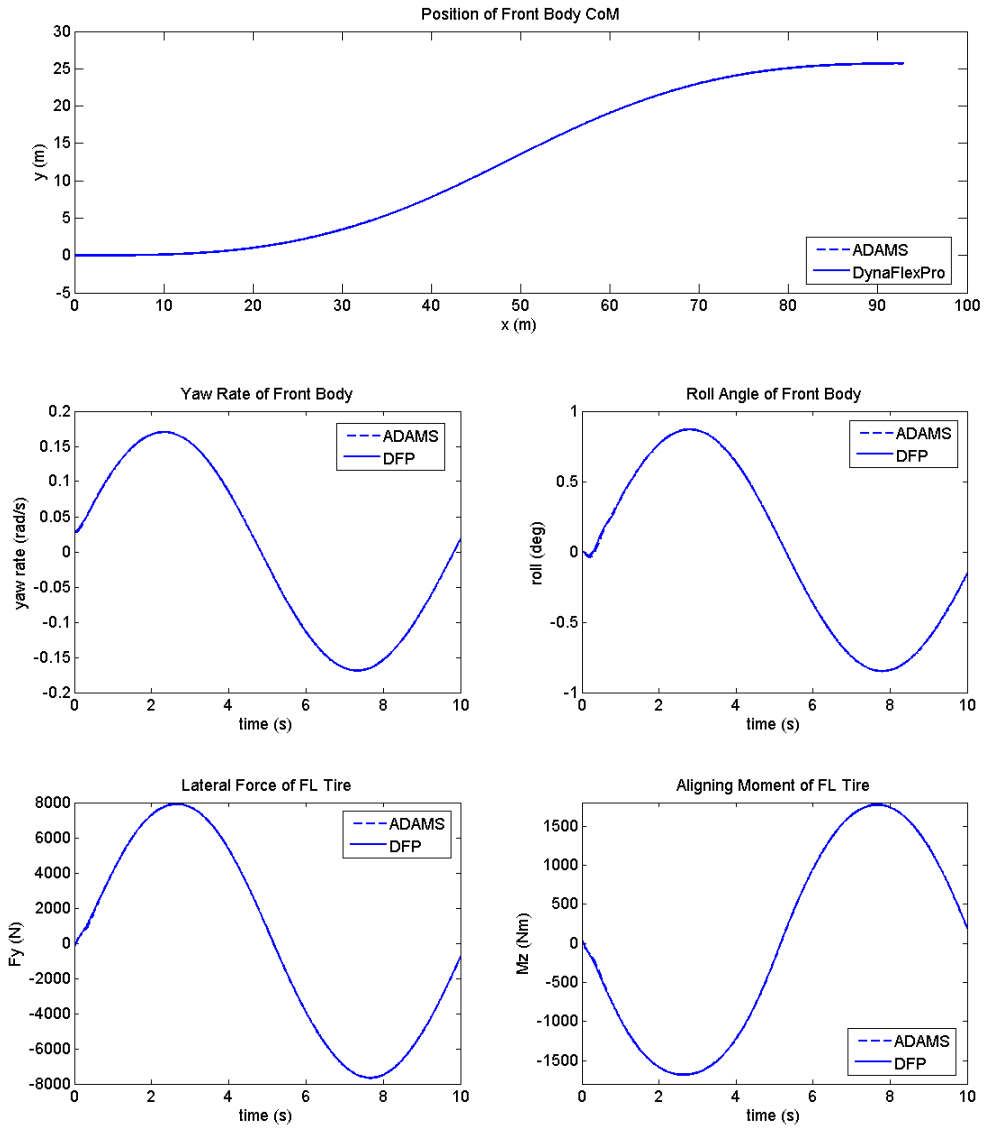


Figure 5.19: Response of Articulated Forestry Skidder for Sine Steer Maneuver

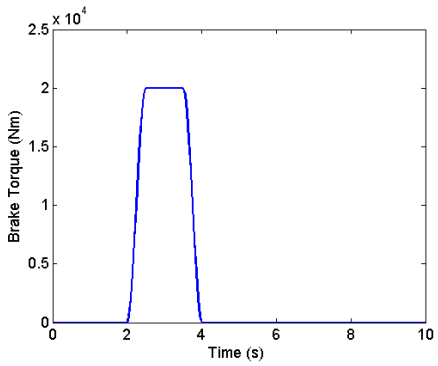


Figure 5.20: Brake Torque Applied to Articulated Forestry Skidder Model

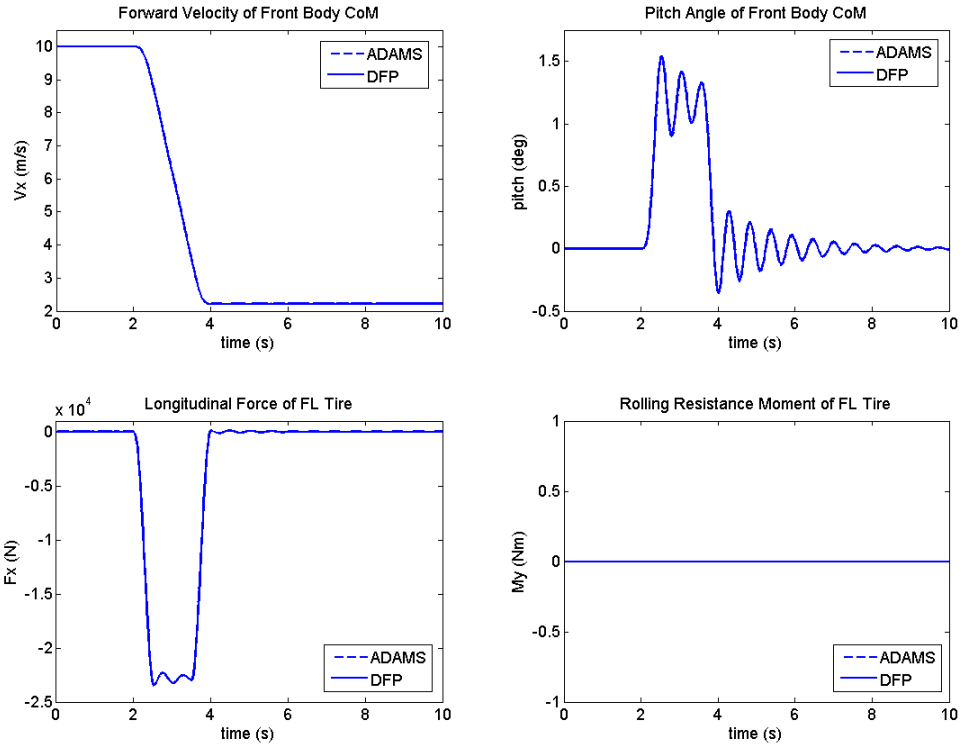


Figure 5.21: Response of Articulated Forestry Skidder for Braking Maneuver

line. Initial conditions for state variables are listed in Table A.15.

Simulation of the DynaFlexPro model was performed in Simulink using Euler's method and a constant step size of 1 ms and the ADAMS model was simulated using the GSTIFF solver with SI2 formulation and an error tolerance of 1E-3.

Results for the braking maneuver are plotted in Figure 5.21. When the braking torque is applied, longitudinal tire forces develop that cause the vehicle to slow from its initial speed of 10 m/s to a final speed of approximately 2 m/s. During braking, the front body pitches forward approximately 1.5 degrees as the load on the front tires increases and the load on the rear tires decreases. After braking, the front body eventually settles to its initial pitch angle of 0 degrees. There is considerable pitch oscillation because the vehicle model does not have a suspension system capable of damping out these motions.

The Fiala model rolling resistance coefficient was specified to be zero for the tires on the articulated forestry skidder model. Therefore, the rolling resistance moment predicted by the Fiala tire model is zero, as shown in Figure 5.21. As a consequence, the longitudinal force settles to zero when the braking moment is released, and the vehicle does not continue to slow down.

Once again, there is excellent agreement between the ADAMS model and the DynaFlexPro model, indicating that for the tire options used with the forestry skidder example, the DynaFlexPro tire component model is accurate for maneuvers dominated by longitudinal tire forces.

5.4.3 Simulation Time in Simulink

For the articulated forestry skidder model, the 10 second steering maneuver was simulated in 0.31 seconds and the 10 second braking maneuver was simulated in 0.32 seconds. Both times are an average of 5 runs in Simulink's "accelerator" mode. The forestry skidder model can be simulated over 30 times faster than real-time on a desktop PC, using a constant step size explicit solver. It more than meets the requirements for hardware-in-the-loop applications.

As a point of comparison, it took 1.31 seconds to simulate the sine steer maneuver and 1.27 seconds to simulate the braking maneuver in ADAMS using the GSTIFF with SI2 formulation. This is significantly slower than the DynaFlexPro-generated model being solved in Simulink using Euler's method. However, comparison of ADAMS solution times to Simulink solution times should be done with reservation because the nature of the solvers is so different.

5.4.4 Component-Level Simplification for Tire Intermediates

As with the previous example, different simplification schemes were investigated for tire intermediate variable symbolic expressions. The effects are summarized in Table 5.5. The same trends that were identified for the generic 4-wheeled vehicle with independent sus-

Intermediate Simplification Type	Time to Generate Sim. Code (s)	Cost of Sim. Code				Simulation Time (s)	
		+	×	÷	f	Sine Steer	Braking
none	11.4	1457	2282	16	49	0.31	0.32
simplify	19.6	2445	4484	20	118	0.36	0.37
combine	37.2	3303	5803	16	198	0.44	0.46

Table 5.5: Component-Level Simplification Applied to Tire Intermediates for Articulated Forestry Skidder

pension can be identified for the articulated forestry skidder. Using Maple's **simplify()** command or **combine()** command on the symbolic expressions for tire intermediates increases the time necessary to generate simulation code, only serves to make the simulation code more computationally expensive, and results in slower simulation times. For both of the example problems considered, it is best not to use Maple's expression manipulation commands on tire intermediate expressions.

5.4.5 Selecting a Tire Component Edge to the System Tree

The normal use of the linear graph tire component involves placing all tire component edges in the cotree. As we did with the previous example, we will investigate the effects of placing edges T1 and R1 for each tire in the system tree of the articulated forestry skidder model. In order to maintain a valid tree selection, all revolute joint edges were moved to the cotree, as shown in Figure 5.22 and 5.23. As a result, the revolute joint rotations will no longer be considered as state variables.

With tire component edges T1 and R1 in the tree, the position, orientation, velocity, and angular velocity of each tire center frame with respect to the Ground frame will be state variables and 24 dynamic equations related to the tires will be formulated. Adding the 6 dynamic equations related to rotational and translational motion of the front half of the body, we arrive at a total of 30 dynamic equations used to describe the 10 degree of freedom forestry skidder model. There will be $30 - 10 = 20$ algebraic constraint equations necessary to enforce the dependencies of the chosen coordinates. The system is described by a set of 30 dynamic equations and 20 algebraic constraint equations. We should expect the system equations (block 3 of the simulation code) to be more costly to evaluate than the case where all tire edges were in the cotree and the dynamic equations were a minimal set of 10 ODEs.

As with the generic 4-wheeled vehicle example, algebraic constraint equations were differentiated twice, combined with the dynamic equations, and Baumgarte constraint stabilization used to keep position-level constraint violation from growing out of control. Parameters of $\alpha = 20$ and $\beta = 20$ were chosen for the Baumgarte constraint stabilization.

With edges T1 and R1 selected into the tree for each tire, the system equations are

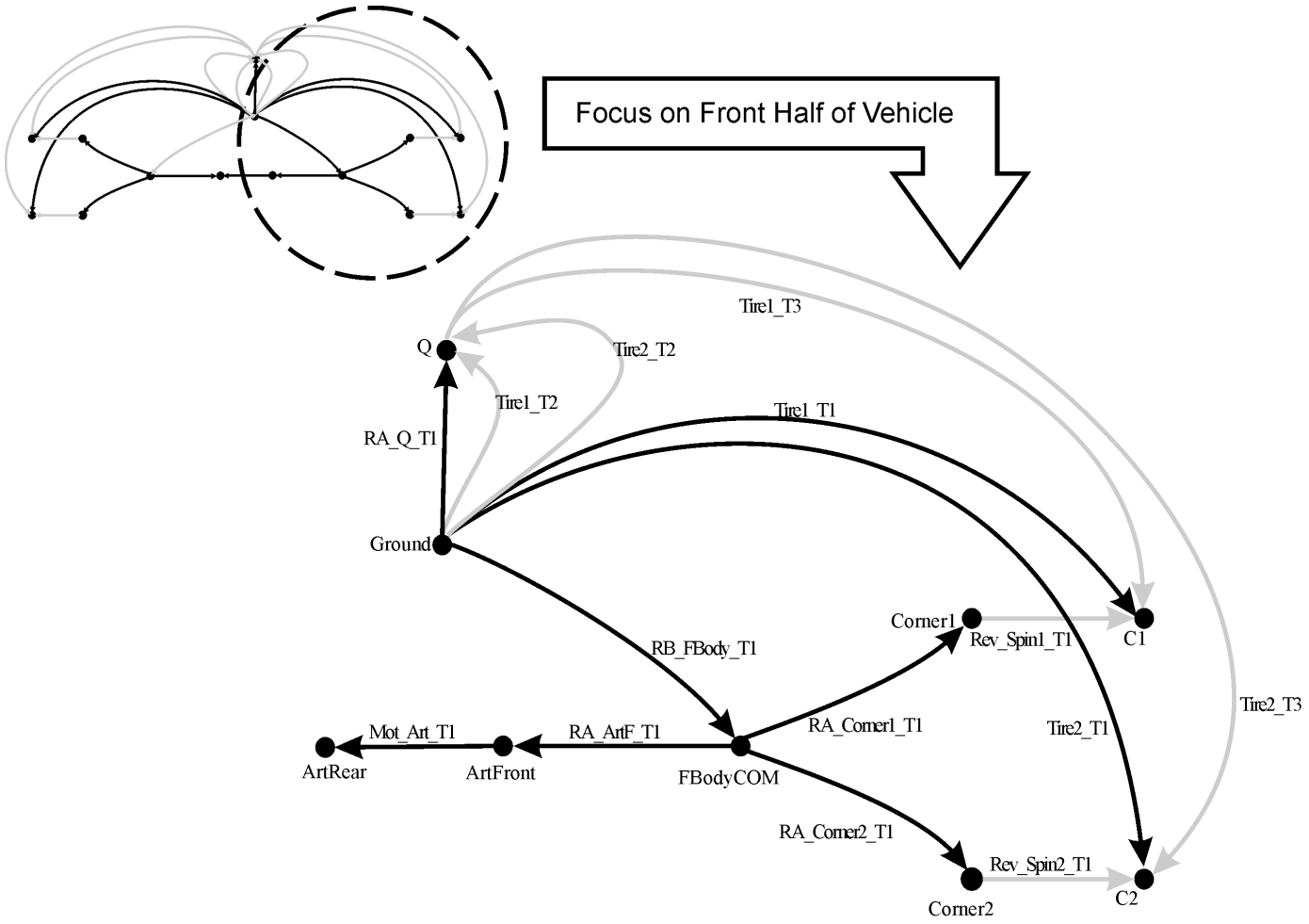


Figure 5.22: Translational Domain Graph of Articulated Forestry Skidder (Edge T1 of Tire Component in the Tree)

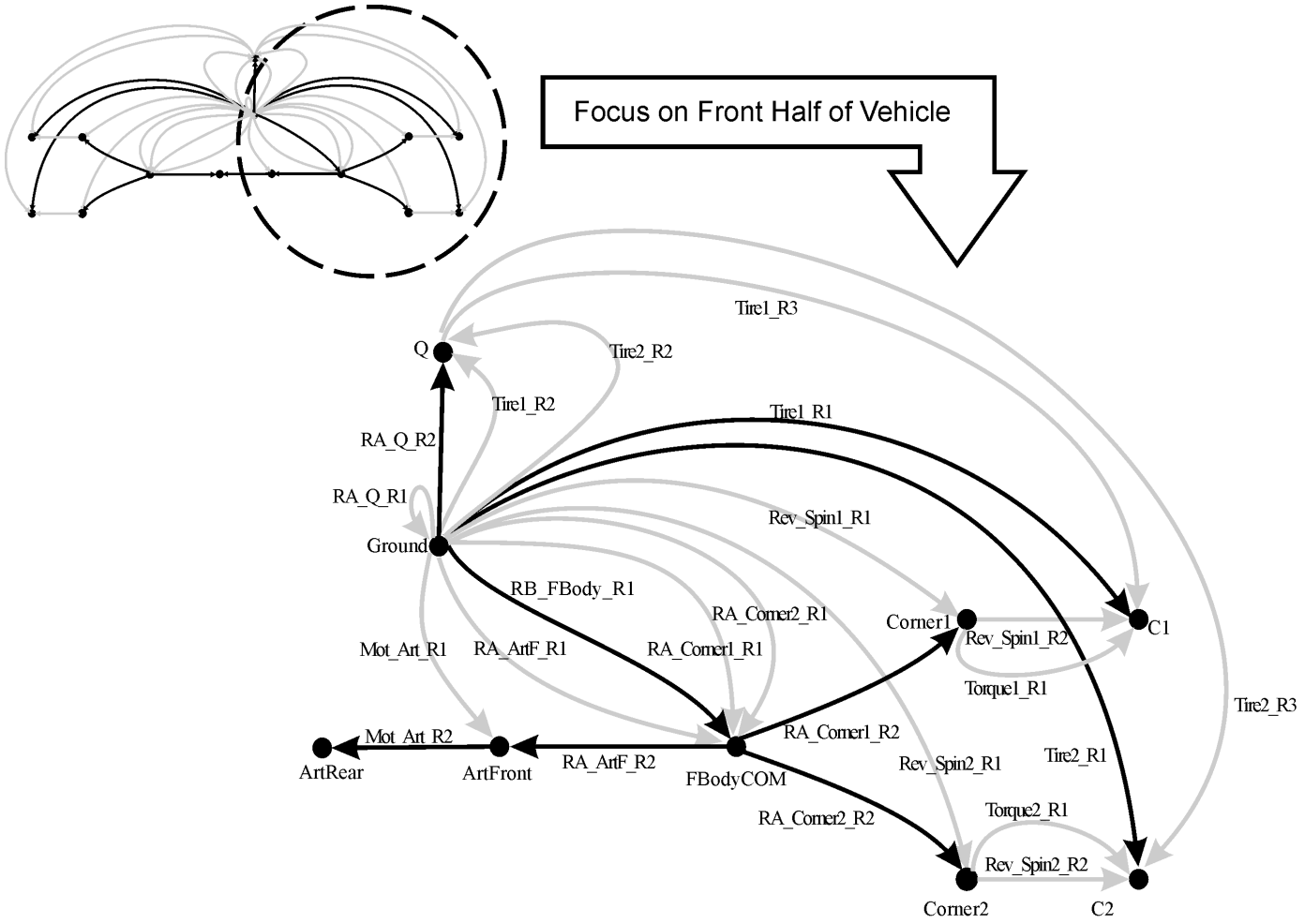


Figure 5.23: Rotational Domain Graph of Articulated Forestry Skidder (Edge R1 of Tire Component in the Tree)

Placement of Tire Edges	Cost of Sim. Code				Simulation Time (s)	
	+	×	÷	f	Sine Steer	Braking
cotree	1457	2282	16	49	0.31	0.32
tree	6765	8705	36	95	0.51	0.54

Table 5.6: Effect of Tree Selection on Simulation Code and Simulation Times for Articulated Forestry Skidder Model

based on 30 generalized speeds $\mathbf{p}$ and 30 coordinates $\mathbf{q}$ for a total of 60 state variables. The initial conditions for these state variables were chosen to be equivalent to the initial conditions used for validation versus ADAMS in Section 5.4.2. The initial conditions used for the sine steer and braking maneuvers are listed in Table A.16 and Table A.17, respectively.

Having a tire edge in the tree shortens the chain of branches whose across variables must be added to obtain symbolic expressions for $\vec{r}_{C/Q}$, $\vec{v}_{C/Q}$, $\vec{\omega}_{C/Q}$, and $\mathbf{R}_{C/Q}$ in terms of state variables, and should reduce the computational cost associated with block 1 of the simulation code. However, it also increases the number of dynamic equations associated with block 3 of the simulation code. Table 5.6 shows that the increased cost of the system equations far outweighs the decreased cost of the tire component intermediate variables for the articulated forestry skidder. Overall, the simulation code is more costly when tire edges are placed in the tree, which results in longer simulation times.

Recall that when tire edges were selected into the tree of the generic 4-wheeled vehicle model and Baumgarte constraint stabilization was employed to handle the algebraic constraints, the step size for the Euler solver had to be significantly reduced to obtain a converged solution. However, when the same process was employed with the forestry skidder model, a converged solution was obtained with Euler's method and a constant step size of 1 ms. Results for the braking and steering maneuver look identical to those presented in Figures 5.19 and 5.21. Although the solution is slower when tire edges have been placed in the tree, the simulation code for the forestry skidder is still suitable for real-time applications and hardware-in-the-loop.

The suitability of Baumgarte constraint stabilization coupled with an explicit fixed step solver for simulating vehicle models with algebraic constraints is highly dependent on the details of the vehicle model and tire model. For some models with algebraic constraints (like the forestry skidder with tire edges in the tree), the process can result in faster than real-time simulations with a converged solution. For other models with algebraic constraints (like the generic 4-wheeled vehicle with tire edges in the tree) the required step sizes are prohibitively small for real-time simulation. Future research should be done to determine a method for handling algebraic constraint equations that consistently enables real-time simulation for a wide variety of vehicle models.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

A tire component model was developed that allows the inclusion of pneumatic tires as part of a larger multibody, multi-domain model, whose governing equations can be automatically formulated using a procedure based on linear graph theory. New types of component-level information were introduced, including rules for generating an intermediate computational sequence and rules for calling external functions from within the main simulation code. These rules help to automate the process of generating efficient simulation code for vehicle models containing tires, and draw on elements of linear graph theory and symbolic computing.

The approach offers analysts a unique set of advantages:

- The DynaFlexPro tire component is designed to work within a formulation procedure based on linear graph theory and the principle of orthogonality, a method that applies equally to mechanical, hydraulic, and electrical domains. Vogt et al. have recently used the DynaFlexPro and DynaFlexPro/Tire to build a mechatronic vehicle model [55].

- In contrast to specialized programs for real-time simulation of vehicle dynamics that use hard-coded equations representing a generic model topology, the approach presented in this thesis allows the automatic generation of efficient simulation code describing any model topology.

- The linear graph formulation procedure offers the analyst complete freedom to select the modeling variables used to describe the multibody portion of his vehicle model, simply by selecting a system tree and specifying the unit vectors used to span the across space of tree edges.

- DynaFlexPro and DynaFlexPro/Tire are built on top of the general purpose symbolic math tool, Maple, which allows engineers to view system equations in a mean-

ingful form and perform additional operations on them if they see fit. All of Maple's functionality is available to them.

- Several tire component options are available which allow analysts to choose the expression used to estimate effective rolling radius, whether or not to introduce differential equations for tire transient behavior, and which tire model function to use. All of these decisions can be made independently.

- The method of constructing vehicle simulation code allows analysts the flexibility to define their own external functions for use with instances of the tire component. This enables research to be conducted into new tire models or new ways of evaluating derivatives of tire slip state variables; engineers are not restrained to a finite list of available functions.

DynaFlexPro and DynaFlexPro/Tire were used to generate simulation code for a generic 4-wheeled vehicle with independent suspension and an articulated forestry skidder; both vehicle models have an open-loop topology, which allowed formulation of their governing equations as a set of ODEs. Validation of the DynaFlexPro generated vehicle models against equivalent models built in MSC.ADAMS demonstrated the accuracy of the tire component model and its software implementation.

For the example models considered, with a tree selection (choice of modeling variables) that results in a minimal set of ODEs with no algebraic constraints, the simulation code generated is suitable for real-time applications and hardware-in-the-loop. The simulation code was exported in the C language and solved in Simulink, where vehicle behavior was simulated 20-30 times faster than real-time using a constant step size explicit solver on a PC with a 3.2 GHz Pentium 4 processor and 1GB of RAM.

With a tree selection that included tire component edges, and which resulted in the formulation of system equations in terms of both differential and algebraic equations, the picture was less clear. The algebraic constraint equations were dealt with by differentiating them twice and using Baumgarte constraint stabilization to control the error in the position-level constraints. This technique worked well for the forestry skidder model, which displayed a converged solution when solved using Euler's method and a step size of 1 ms, and could be solved faster than real-time in Simulink. The technique did not work as well for the generic 4-wheeled vehicle model. With tire edges selected into the tree and Baumgarte constraint stabilization used to handle the algebraic constraint equations, a step size of 0.02 ms was required to ensure a converged solution, and the generic 4-wheeled vehicle model could not be solved faster than real-time on a PC with a 3.2 GHz Pentium 4 processor and 1GB of RAM.

6.2 Recommendations for Future Research

One of the main accomplishments of this work was the implementation of a method for controlling the size of tire intermediate expressions. Within the computational sequence used to evaluate tire intermediates, the substitution of certain expressions was delayed and dummy names used in their place. The rules that define which symbolic expressions get substituted, and which are left as dummy names, work well for the example problems considered in this thesis but may not be optimal for all conceivable vehicle models. Future research should be performed using different rules for constructing a tire intermediate computational sequence, applied to many different vehicle models, in order to identify how and under what conditions the computational effort associated with calculating tire intermediate variables can be reduced.

The results presented in this thesis are inconclusive about the suitability of vehicle models described by DAEs for real-time applications. In the current version of DynaFlexPro, algebraic constraint equations can only be solved on the acceleration level. Work should be done to allow the inclusion of a modified Newton/Raphson algorithm within the simulation code that is generated by DynaFlexPro. This would allow algebraic equations to be solved on the position level, and has the potential to be used for hardware-in-the-loop applications, as long as the number of iterations used by the Newton/Raphson algorithm at each time step is limited to a fixed value.

A constant step size explicit solver (Euler's method) was used to simulate the behavior of example vehicle models. Implicit solvers perform better than explicit solvers for stiff ODEs, like the ones obtained when acceleration-level constraint equations are combined with the dynamic equations. If implicit solvers are to be used for hardware-in-the-loop applications, the number of iterations performed by the solver at each time step must be limited to a fixed value. Modified implicit solvers should be developed and used in conjunction with DynaFlexPro-generated vehicle models in the future.

The importance of investigating modified Newton/Raphson methods for solving algebraic constraint equations and modified implicit solvers for solving stiff sets of differential equations increases when closed-loop vehicle topologies are considered, for which algebraic constraint equations are unavoidable. The use of specialized DAE solvers might also be investigated, as long as the number of iterations performed by the solver can be limited for HIL applications.

Another area of research involves the choice of modeling variables used to describe vehicle systems containing tires. Selecting a tire edge into the system tree shortens the chain of branches whose across variables must be added in order to obtain expressions for the kinematics of the tire in terms of the chosen modeling variables. Placing a tire edge in the tree was not advantageous for the open-loop vehicle topologies examined in this thesis because this type of tree selection forced the formulation of system equations as DAEs, when a set of ODEs would have been more efficient. For certain closed loop topologies, where a set of DAEs is unavoidable, the advantages of placing tire component

edges in the tree may outweigh the disadvantages.

Thus far, the linear graph tire component has been applied to models involving stationary roads, but the method it is equally applicable to models involving moving road planes (for example, a four-post shaker). Models involving moving roads should be built and tested to ensure robustness of the method.

A main limitation of the current tire component model is its restriction to planar roads. The possibility of extending the theory presented in this thesis to 3D roads should be investigated, including the possibility of incorporating tire models that are specifically designed for predicting vehicle response over rough terrain, such as the Flexible Ring Tire Model (FTire) [8].

Appendix A

Model Parameters and Initial Conditions

A.1 Generic 4-Wheeled Vehicle with Independent Suspension

For definition of the x, y, and z directions, refer to Figure 5.3. All body-fixed reference frames are initially aligned with the inertial (Ground) reference frame.

Body	Mass (kg)	Rotational Inertia ($kg \cdot m^2$)					
		Ixx	Iyy	Izz	Ixy	Ixz	Iyz
Vehicle Body	2077	330	1925	1925	0	110	0
Lumped Mass *	10	1.0	0.5	1.0	0	0	0
Tires *	28	0.78	1.56	0.78	0	0	0
Front Knuckles **	0	0	0	0	0	0	0

* same at every corner

** only present in ADAMS model

Table A.1: Inertia Properties for Generic 4-Wheeled Vehicle

Spring/Damper	Stiffness($\frac{N}{m}$)	Free Length(m)	Damping($\frac{N \cdot s}{m}$)
Front Corner *	48289	0.674	3075
Rear Corner *	30518	0.72	2331
Tires **	$3.04E5$	0.355	500

* same left to right

** same at every corner

Table A.2: Spring/Damper Properties for Generic 4-Wheeled Vehicle

Rigid Arm	Translation(m)			Rotation(rad)		
	x	y	z	$\theta 1$	$\theta 2$	$\theta 3$
BodyCOM to FLcorner	1.353	0.76	0.0	0.0	0.0	0.0
BodyCOM to FRcorner	1.353	−0.76	0.0	0.0	0.0	0.0
BodyCOM to RLcorner	−1.487	0.795	0.0	0.0	0.0	0.0
BodyCOM to RRcorner	−1.487	−0.795	0.0	0.0	0.0	0.0
Ground to Road Frame Q	0.0	0.0	0.0	0.0	0.0	0.0

Table A.3: Enforced Geometry for Generic 4-Wheeled Vehicle

Parameter	Value	Description
$Fz0$	4850	Nominal normal force
$R0$	0.344	Unloaded radius
$V0$	16.6	Reference Velocity
$LFz0$	1	Scale factor of nominal load
LCx	1	Scale factor of Fx shape factor
$L\mu x$	1	Scale factor of Fx peak friction coefficient
LEx	1	Scale factor of Fx curvature factor
LKx	1	Scale factor of Fx slip stiffness
LHx	1	Scale factor of Fx horizontal shift
LVx	1	Scale factor of Fx vertical shift
$L\gamma x$	1	Scale factor of camber for Fx
LCy	1	Scale factor of Mz shape factor
$L\mu y$	1	Scale factor of Mz peak friction coefficient
LEy	1	Scale factor of Mz curvature factor
LKy	1	Scale factor of Mz cornering stiffness
LHy	1	Scale factor of Mz horizontal shift
LVy	1	Scale factor of Mz vertical shift
$L\gamma y$	1	Scale factor of camber for Mz
$Ltrail$	1	Scale factor of peak pneumatic trail
$Lres$	1	Scale factor for offset of residual residual torque
$L\gamma z$	1	Scale factor of camber for Mz
$LX\alpha$	1	Scale factor of slip angle influence on Fx
$LY\kappa$	1	Scale factor of longitudinal slip influence on Mz
$LVy\kappa$	1	Scale factor of longitudinal slip influence on Mz
LS	1	Scale factor of moment arm of Fx about vertical axis
LMx	1	Scale factor of overturning couple
$LVMx$	1	Scale factor of Mx vertical shift

Table A.4: Pacejka 2002 Tire Model Parameters for Generic 4-Wheeled Vehicle

continued on next page

continued from previous page

Parameter	Value	Description
LMy	1	Scale factor of rolling resistance torque
$PCx1$	1.6411	Shape factor for longitudinal force
$PDx1$	1.1739	Longitudinal friction, μx, at $Fz0$ and zero inclination
$PDx2$	-0.16395	Variation of friction, μx, with load
$PDx3$	5.0	Variation of friction, μx, with inclination
$PEx1$	0.46403	Longitudinal curvature EFx at $Fz0$
$PEx2$	0.25022	Variation of curvature EFx with load
$PEx3$	0.067842	Variation of curvature EFx with load squared
$PEx4$	$-3.7604E-5$	Brake/drive asymmetry factor for EFz
$PKx1$	22.303	Longitudinal slip stiffness $\frac{Kfx}{Fz}$ at $Fz0$
$PKx2$	0.48896	Variation of slip stiffness $\frac{Kfx}{Fz}$ with load
$PKx3$	0.21253	Exponent in slip stiffness $\frac{Kfx}{Fz}$ with load
$PHx1$	0.0012297	Horizontal shift of longitudinal slip at $Fz0$
$PHx2$	0.0004318	Variation of horizontal shift with load
$PVx1$	$-8.8098E-6$	Vertical shift at $Fz0$
$PVx2$	$1.862E-5$	Variation of vertical shift with load
$RBx1$	13.276	Slope factor for combined slip Fx reduction
$RBx2$	-13.778	Variation of slope Fx reduction with longitudinal slip
$RCx1$	1.2568	Shape factor for combined slip Fx reduction
$REx1$	0.65225	Curvature factor for combined slip Fx reduction
$REx2$	-0.24948	Variation of curvature factor with load
$RHx1$	0.0050722	Shift factor for combined slip Fx reduction
$QSx1$	0.0	Vertical force induced overturning moment
$QSx2$	0.0	Camber induced overturning couple
$QSx3$	0.0	Aligning moment induced overturning couple
$PCy1$	1.3507	Shape factor for pure lateral force
$PDy1$	1.0489	Lateral peak friction, μy
$PDy2$	-0.18033	Variation of μy with load
$PDy3$	-2.8821	Variation of μy with inclination squared
$PEy1$	-0.0074722	Lateral force curvature factor at $Fz0$
$PEy2$	-0.0063208	Variation of curvature with load
$PEy3$	-9.9935	Dependency of curvature on the sign of slip angle
$PEy4$	-760.14	Variation of curvature with camber
$PKy1$	-21.92	Maximum value of cornering stiffness $\frac{KMz}{Fznom}$
$PKy2$	2.0012	Load for max cornering stiffness
$PKy3$	-0.024778	Variation of KMz with camber

Table A.4: Pacejka 2002 Tire Model Parameters for Generic 4-Wheeled Vehicle

continued on next page

continued from previous page

Parameter	Value	Description
$PHy1$	0.0026747	Horizontal shift of lateral force at $Fz0$
$PHy2$	$8.9094E-5$	Variation of horizontal shift with load
$PHy3$	0.031415	Variation of horizontal shift with camber
$PVy1$	0.037318	Vertical shift $\frac{Svy}{Fz}$ at $Fz0$
$PVy2$	-0.010049	Variation of vertical shift with load
$PVy3$	-0.32931	Variation of vertical shift with camber
$PVy4$	-0.69553	Variation of vertical shift with camber and load
$RBy1$	7.1433	Slope factor for combined slip Mz reduction
$RBy2$	9.1916	Variation of slope factor with slip angle
$RBy3$	-0.027856	shift term for alpha in slope factor
$RCy1$	1.0719	Shape factor for combined slip Mz reduction
$REy1$	-0.27572	Curvature factor for combined slip Mz reduction
$REy2$	0.32802	Variation of curvature factor with load
$RHy1$	$5.7448E-6$	Shift factor for combined slip Mz reduction
$RHy2$	$-3.1368E-5$	Variation of shift factor with load
$RVy1$	-0.027825	longitudinal slip induced side forces at $Fz0$
$RVy2$	0.053604	Variation of vertical shift with load
$RVy3$	-0.27568	Variation of vertical shift with camber
$RVy4$	12.12	Variation of vertical shift with slip angle
$RVy5$	1.9	Variation of vertical shift with longitudinal slip
$RVy6$	-10.704	Variation of vertical shift with $\arctan(S)$
$QSy1$	0.01	rolling resistance torque coefficient
$QSy2$	0.0	rolling resistance influenced by Fx
$QSy3$	0.0	rolling resistance influenced by V_{Cx}
$QSy4$	0.0	rolling resistance influenced by V_{Cx}^4
$QBz1$	10.904	Pneumatic trail slope factor, Bt, at $Fz0$
$QBz2$	-1.8412	Variation of Bt with load
$QBz3$	-0.52041	Variation of Bt with load squared
$QBz4$	0.039211	Variation of Bt with camber
$QBz5$	0.41511	Variation of Bt with absolute camber
$QBz9$	8.9846	Slope factor for residual torque Br
$QBz10$	0.0	Slope factor for residual torque Br
$QCz1$	1.2136	Shape factor for pneumatic trail
$QDz1$	0.093509	Controls Peak pneumatic trail, Dt
$QDz2$	-0.0092183	Variation of peak Dt with load
$QDz3$	-0.057061	Variation of peak Dt with camber

Table A.4: Pacejka 2002 Tire Model Parameters for Generic 4-Wheeled Vehicle

continued on next page

continued from previous page

Parameter	Value	Description
$QDz4$	0.73954	Variation of peak Dt with camber squared
$QDz6$	-0.0067783	Controls peak residual torque, Dr
$QDz7$	0.0052254	Variation of peak Dr with load
$QDz8$	-0.18175	Variation of peak Dr with camber
$QDz9$	0.029952	Variation of peak Dr with camber and load
$QEz1$	-1.5697	Pneumatic trail curvature at $Fz0$
$QEz2$	0.33394	Variation of trail curvature with load
$QEz3$	0.0	Variation of trail curvature with load squared
$QEz4$	0.26711	Variation of trail curvature with sign of α
$QEz5$	-3.594	Variation of trail curvature with γ and sign of α
$QHz1$	0.0047326	Pneumatic trail horizontal shift at $Fz0$
$QHz2$	0.0026687	Variation of trail horizontal shift with load
$QHz3$	0.11998	Variation of trail horizontal shift with γ
$QHz4$	0.059083	Variation of trail horizontal shift with γ and load
$SSz1$	0.033372	Nominal value of $\frac{S}{R0}$: effect of Fx on Mz
$SSz2$	0.0043624	Variation of S with Mz
$SSz3$	0.56742	Variation of S with camber
$SSz4$	-0.24116	Variation of S with load and camber

Table A.4: Pacejka 2002 Tire Model Parameters for Generic 4-Wheeled Vehicle

Parameter	Value	Description
$Fz0$	5900	Nominal normal force
$BReff$	8	Low load stiffness parameter
$DReff$	0.24	Peak Value of R_{eff}
$FReff$	0.01	High load stiffness parameter

Table A.5: Parameters for Calculating Effective Rolling Radius

Parameter	Value	Description
$Fz0$	4850	Nominal Normal Force
$R0$	0.344	Unloaded Radius
$LS\kappa$	1.0	Scale Factor for B_{long}
$LS\alpha$	1.0	Scale Factor for B_{lat}
$PTx1$	2.3657	B_{long} at Nominal Load
$PTx2$	1.4112	Variation of B_{long} with Load
$PTx3$	0.56626	Variation of B_{long} with Exponent of Load
$PTy1$	2.1439	Peak Value of B_{lat}
$PTy2$	1.9829	Value of $Fz/Fz0$ at which B_{lat} is Extreme
$PKy3$	-0.90729	Variation of B_{lat} with Inclination

Table A.6: Parameters for Stretched String Method of Calculating Relaxation Lengths

Parameter	Value	Description
D_2	0.16	Width of Tire (m)
C_S	115000	Longitudinal Stiffness (N)
C_α	117000	Lateral Stiffness (N/rad)
C_r	0.01	Coefficient of Rolling Resistance (m)
μ_0	1.22	Peak Coefficient of Friction
μ_1	0.2	Sliding Coefficient of Friction

Table A.7: Fiala Tire Model Parameters for Generic 4-Wheeled Vehicle — Roughly Equivalent to Pacejka Parameters Presented in Table A.4

State Variable	Initial Condition	Description
$vx\,(t)$	20.0	X Component of Vehicle Body Velocity [1]
$vy\,(t)$	0.0	Y Component of Vehicle Body Velocity [1]
$vz\,(t)$	0.0	Z Component of Vehicle Body Velocity [1]
$\omega 1\,(t)$	0.0	X Component of Vehicle Body Angular Velocity [1]
$\omega 2\,(t)$	0.0	Y Component of Vehicle Body Angular Velocity [1]
$\omega 3\,(t)$	0.0	Z Component of Vehicle Body Angular Velocity [1]
$\frac{d}{dt}s1\,(t)$	0.0	Rate of FL Suspension Extension [2]
$\frac{d}{dt}s2\,(t)$	0.0	Rate of FR Suspension Extension [2]
$\frac{d}{dt}s3\,(t)$	0.0	Rate of RL Suspension Extension [2]
$\frac{d}{dt}s4\,(t)$	0.0	Rate of RR Suspension Deflection [2]
$\frac{d}{dt}phi1\,(t)$	57.45475438	Spin Rate of FL Tire [3]
$\frac{d}{dt}phi2\,(t)$	57.45475438	Spin Rate of FR Tire [3]
$\frac{d}{dt}phi3\,(t)$	57.43825388	Spin Rate of RL Tire [3]
$\frac{d}{dt}phi4\,(t)$	57.43825388	Spin Rate of RR Tire [3]
$x\,(t)$	0.0	X Position of Vehicle Body COM [4]
$y\,(t)$	0.0	Y Position of Vehicle Body COM [4]
$z\,(t)$	0.8995	Z Position of Vehicle Body COM [4]
$yaw\,(t)$	0.0	First 321 Euler Angle [5]
$pitch\,(t)$	0.0	Second 321 Euler Angle [5]
$roll\,(t)$	0.0	Third 321 Euler Angle [5]
$s1\,(t)$	0.5632	Position of FL Suspension Lumped Mass [2]
$s2\,(t)$	0.5632	Position of FR Suspension Lumped Mass [2]
$s3\,(t)$	0.5617	Position of RL Suspension Lumped Mass [2]
$s4\,(t)$	0.5617	Position of RR Suspension Lumped Mass [2]
$phi1\,(t)$	0.0	Rotation Angle of FL tire [3]
$phi2\,(t)$	0.0	Rotation Angle of FR tire [3]
$phi3\,(t)$	0.0	Rotation Angle of RL tire [3]
$phi4\,(t)$	0.0	Rotation Angle of RR tire [3]
$q1Tire1\,(t)$	0.0	Delayed Longitudinal Slip for FL Tire
$q2Tire1\,(t)$	0.0	Tangent of Delayed Slip Angle for FL Tire
$q1Tire2\,(t)$	0.0	Delayed Longitudinal Slip for FR Tire

Table A.8: Initial Conditions for Generic 4-Wheeled Vehicle with Tire Edges in Cotree

continued on next page

[1] expressed in BodyCoM frame
[2] along prismatic joint axis
[3] along revolute joint axis
[4] expressed in Ground frame
[5] used to track orientation of BodyCoM frame with respect to Ground frame

continued from previous page

State Variable	Initial Condition	Description
$q2Tire2\,(t)$	0.0	Tangent of Delayed Slip Angle for FR Tire
$q1Tire3\,(t)$	0.0	Delayed Longitudinal Slip for RL Tire
$q2Tire3\,(t)$	0.0	Tangent of Delayed Slip Angle for RL Tire
$q1Tire4\,(t)$	0.0	Delayed Longitudinal Slip for RR Tire
$q2Tire4\,(t)$	0.0	Tangent of Delayed Slip Angle for RR Tire

Table A.8: Initial Conditions for Generic 4-Wheeled Vehicle with Tire Edges in Cotree

State Variable	Initial Condition	Description
$vx\,(t)$	20.0	X Component of Vehicle Body Velocity [1]
$vy\,(t)$	0.0	Y Component of Vehicle Body Velocity [1]
$vz\,(t)$	0.0	Z Component of Vehicle Body Velocity [1]
$\omega 1\,(t)$	0.0	X Component of Vehicle Body Angular Velocity [1]
$\omega 2\,(t)$	0.0	Y Component of Vehicle Body Angular Velocity [1]
$\omega 3\,(t)$	0.0	Z Component of Vehicle Body Angular Velocity [1]
$\frac{d}{dt}xTire1\,(t)$	20.0	FL Tire X Velocity [2]
$\frac{d}{dt}yTire1\,(t)$	0.0	FL Tire Y Velocity [2]
$\frac{d}{dt}zTire1\,(t)$	0.0	FL Tire Z Velocity [2]
$\omega xTire1\,(t)$	0.0	X Component of FL Tire Angular Velocity [3]
$\omega yTire1\,(t)$	57.455	Y Component of FL Tire Angular Velocity [3]
$\omega zTire1\,(t)$	0.0	Z Component of FL Tire Angular Velocity [3]
$\frac{d}{dt}xTire2\,(t)$	20.0	FR Tire X Velocity [2]
$\frac{d}{dt}yTire2\,(t)$	0.0	FR Tire Y Velocity [2]
$\frac{d}{dt}zTire2\,(t)$	0.0	FR Tire Z Velocity [2]
$\omega xTire2\,(t)$	0.0	X Component of FR Tire Angular Velocity [3]
$\omega yTire2\,(t)$	57.455	Y Component of FR Tire Angular Velocity [3]
$\omega zTire2\,(t)$	0.0	Z Component of FR Tire Angular Velocity [3]
$\frac{d}{dt}xTire3\,(t)$	20.0	RL Tire X Velocity [2]
$\frac{d}{dt}yTire3\,(t)$	0.0	RL Tire Y Velocity [2]
$\frac{d}{dt}zTire3\,(t)$	0.0	RL Tire Z Velocity [2]

Table A.9: Initial Conditions for Generic 4-Wheeled Vehicle with Tire Edges in Tree

continued on next page

[1] expressed in BodyCoM frame
[2] expressed in Ground frame
[3] expressed in TireCoM frame

continued from previous page

State Variable	Initial Condition	Description
$\omega xTire3\,(t)$	0.0	X Component of RL Tire Angular Velocity [3]
$\omega yTire3\,(t)$	57.438	Y Component of RL Tire Angular Velocity [3]
$\omega zTire3\,(t)$	0.0	Z Component of RL Tire Angular Velocity [3]
$\frac{d}{dt}xTire4\,(t)$	20.0	RR Tire X Velocity [2]
$\frac{d}{dt}yTire4\,(t)$	0.0	RR Tire Y Velocity [2]
$\frac{d}{dt}zTire4\,(t)$	0.0	RR Tire Z Velocity [2]
$\omega xTire4\,(t)$	0.0	X Component of RR Tire Angular Velocity [3]
$\omega yTire4\,(t)$	57.438	Y Component of RR Tire Angular Velocity [3]
$\omega zTire4\,(t)$	0.0	Z Component of RR Tire Angular Velocity [3]
$x\,(t)$	0.0	X Position of Vehicle Body COM [2]
$y\,(t)$	0.0	Y Position of Vehicle Body COM [2]
$z\,(t)$	0.8995	Z Position of Vehicle Body COM [2]
$yaw\,(t)$	0.0	First 321 Euler Angle [4]
$pitch\,(t)$	0.0	Second 321 Euler Angle [4]
$roll\,(t)$	0.0	Third 321 Euler Angle [4]
$xTire1\,(t)$	1.353	X Position of FL Tire COM [2]
$yTire1\,(t)$	0.76	Y Position of FL Tire COM [2]
$zTire1\,(t)$	0.3363	Z Position of FL Tire COM [2]
$yawTire1\,(t)$	0.0	First 312 Euler Angle [5]
$tiltTire1\,(t)$	0.0	Second 312 Euler Angle [5]
$spinTire1\,(t)$	0.0	Third 312 Euler Angle [5]
$xTire2\,(t)$	1.353	X Position of FR Tire COM [2]
$yTire2\,(t)$	-0.76	Y Position of FR Tire COM [2]
$zTire2\,(t)$	0.3363	Z Position of FR Tire COM [2]
$yawTire2\,(t)$	0.0	First 312 Euler Angle [5]
$tiltTire2\,(t)$	0.0	Second 312 Euler Angle [5]
$spinTire2\,(t)$	0.0	Third 312 Euler Angle [5]
$xTire3\,(t)$	-1.487	X Position of RL Tire COM [2]
$yTire3\,(t)$	0.795	Y Position of RL Tire COM [2]
$zTire3\,(t)$	0.3378	Z Position of RL Tire COM [2]
$yawTire3\,(t)$	0.0	First 312 Euler Angle [5]

Table A.9: Initial Conditions for Generic 4-Wheeled Vehicle with Tire Edges in Tree

continued on next page

[2] expressed in Ground frame
[3] expressed in TireCoM frame
[4] used to track orientation of BodyCoM frame with respect to Ground frame
[5] used to track orientation of tire center frame with respect to Ground frame

continued from previous page

State Variable	Initial Condition	Description
$tiltTire3\,(t)$	0.0	Second 312 Euler Angle [5]
$spinTire3\,(t)$	0.0	Third 312 Euler Angle [5]
$xTire4\,(t)$	-1.487	X Position of RR Tire COM [2]
$yTire4\,(t)$	-0.795	Y Position of RR Tire COM [2]
$zTire4\,(t)$	0.3378	Z Position of RR Tire COM [2]
$yawTire4\,(t)$	0.0	First 312 Euler Angle [5]
$tiltTire4\,(t)$	0.0	Second 312 Euler Angle [5]
$spinTire4\,(t)$	0.0	Third 312 Euler Angle [5]
$q1Tire1\,(t)$	0.0	Delayed Longitudinal Slip for FL Tire
$q2Tire1\,(t)$	0.0	Tangent of Delayed Slip Angle for FL Tire
$q1Tire2\,(t)$	0.0	Delayed Longitudinal Slip for FR Tire
$q2Tire2\,(t)$	0.0	Tangent of Delayed Slip Angle for FR Tire
$q1Tire3\,(t)$	0.0	Delayed Longitudinal Slip for RL Tire
$q2Tire3\,(t)$	0.0	Tangent of Delayed Slip Angle for RL Tire
$q1Tire4\,(t)$	0.0	Delayed Longitudinal Slip for RR Tire
$q2Tire4\,(t)$	0.0	Tangent of Delayed Slip Angle for RR Tire

Table A.9: Initial Conditions for Generic 4-Wheeled Vehicle with Tire Edges in Tree

[2]expressed in Ground frame

[5]used to track orientation of tire center frame with respect to Ground frame

A.2 Articulated Forestry Skidder

For definition of the x, y, and z directions, refer to Figure 5.15. All body-fixed reference frames are initially aligned with the inertial (Ground) reference frame.

Body	Mass (kg)	Rotational Inertia about COM($kg \cdot m^2$)					
		Ixx	Iyy	Izz	Ixy	Ixz	Iyz
Front Body	7280	7280	7280	7280	0	0	0
Rear Body	7280	7280	7280	7280	0	0	0
Tires *	557	112.5	225	112.5	0	0	0

* same at every corner

Table A.10: Inertia Properties for Forestry Skidder

Spring/Damper	Stiffness ($\frac{N}{m}$)	Free Length (m)	Damping ($\frac{N \cdot s}{m}$)
Tires *	$5.0E5$	0.940	$5.0E3$

* same at every corner

Table A.11: Spring/Damper Properties for Forestry Skidder

Rigid Arm	Translation(m)			Rotation(rad)		
	x	y	z	$\theta 1$	$\theta 2$	$\theta 3$
FrontBodyCOM to Articulating Joint	-1.227	0.0	-0.3861	0.0	0.0	0.0
FrontBodyCOM to FLcorner	0.5	1.2275	-0.5461	0.0	0.0	0.0
FrontBodyCOM to FRcorner	0.5	-1.2275	-0.5461	0.0	0.0	0.0
RearBodyCOM to Articulating Joint	1.227	0.0	-0.3861	0.0	0.0	0.0
RearBodyCOM to FRcorner	-0.5	1.2275	-0.5461	0.0	0.0	0.0
RearBodyCOM to FRcorner	-0.5	-1.2275	-0.5461	0.0	0.0	0.0
Ground to Road Frame Q	0.0	0.0	0.0	0.0	0.0	0.0

Table A.12: Enforced Geometry for Forestry Skidder

Parameter	Front Tires	Rear Tires	Description
D_2	0.775	0.775	Width of Tire (m)
C_S	153890	238220	Longitudinal Stiffness (N)
C_α	153890	238220	Lateral Stiffness (N/rad)
C_r	0.00	0.00	Coefficient of Rolling Resistance (m)
μ_0	1.0	1.0	Peak Coefficient of Friction
μ_1	1.0	1.0	Sliding Coefficient of Friction

Table A.13: Fiala Tire Model Parameters for Forestry Skidder

State Variable	Initial Condition	Description
$vx\,(t)$	10.0	X Component of Front Body Velocity [1]
$vy\,(t)$	0.0	Y Component of Front Body Velocity [1]
$vz\,(t)$	0.0	Z Component of Front Body Velocity [1]
$\omega 1\,(t)$	0.0	X Component of Front Body Angular Velocity [1]
$\omega 2\,(t)$	0.0	Y Component of Front Body Angular Velocity [1]
$\omega 3\,(t)$	0.0235	Z Component of Front Body Angular Velocity [1]
$\frac{d}{dt}phi1\,(t)$	11.659428	Spin Rate of FL Tire [2]
$\frac{d}{dt}phi2\,(t)$	11.659428	Spin Rate of FR Tire [2]
$\frac{d}{dt}phi3\,(t)$	11.659428	Spin Rate of RL Tire [2]
$\frac{d}{dt}phi4\,(t)$	11.659428	Spin Rate of RR Tire [2]
$x\,(t)$	0.0	X Position of Front Body COM [3]
$y\,(t)$	0.0	Y Position of Front Body COM [3]
$z\,(t)$	1.403775	Z Position of Front Body COM [3]
$yaw\,(t)$	0.0	First 321 Euler Angle [4]
$pitch\,(t)$	0.0	Second 321 Euler Angle [4]
$roll\,(t)$	0.0	Third 321 Euler Angle [4]
$phi1\,(t)$	0.0	Rotation Angle of FL tire [2]
$phi2\,(t)$	0.0	Rotation Angle of FR tire [2]
$phi3\,(t)$	0.0	Rotation Angle of RL tire [2]
$phi4\,(t)$	0.0	Rotation Angle of RR tire [2]

Table A.14: Initial Conditions for Forestry Skidder — Sine Steer Event, Tire Edges in Cotree

[1] expressed in front body CoM frame
[2] along revolute joint axis
[3] expressed in Ground frame
[4] used to track orientation of front body CoM frame with respect to Ground frame

State Variable	Initial Condition	Description
$vx\,(t)$	10.0	X Component of Front Body Velocity [1]
$vy\,(t)$	0.0	Y Component of Front Body Velocity [1]
$vz\,(t)$	0.0	Z Component of Front Body Velocity [1]
$\omega 1\,(t)$	0.0	X Component of Front Body Angular Velocity [1]
$\omega 2\,(t)$	0.0	Y Component of Front Body Angular Velocity [1]
$\omega 3\,(t)$	0.0	Z Component of Front Body Angular Velocity [1]
$\frac{d}{dt}phi1\,(t)$	11.659428	Spin Rate of FL Tire [2]
$\frac{d}{dt}phi2\,(t)$	11.659428	Spin Rate of FR Tire [2]
$\frac{d}{dt}phi3\,(t)$	11.659428	Spin Rate of RL Tire [2]
$\frac{d}{dt}phi4\,(t)$	11.659428	Spin Rate of RR Tire [2]
$x\,(t)$	0.0	X Position of Front Body COM [3]
$y\,(t)$	0.0	Y Position of Front Body COM [3]
$z\,(t)$	1.403775	Z Position of Front Body COM [3]
$yaw\,(t)$	0.0	First 321 Euler Angle [4]
$pitch\,(t)$	0.0	Second 321 Euler Angle [4]
$roll\,(t)$	0.0	Third 321 Euler Angle [4]
$phi1\,(t)$	0.0	Rotation Angle of FL tire [2]
$phi2\,(t)$	0.0	Rotation Angle of FR tire [2]
$phi3\,(t)$	0.0	Rotation Angle of RL tire [2]
$phi4\,(t)$	0.0	Rotation Angle of RR tire [2]

Table A.15: Initial Conditions for Forestry Skidder — Braking Event, Tire Edges in Cotree

[1] expressed in front body CoM frame
[2] along revolute joint axis
[3] expressed in Ground frame
[4] used to track orientation of front body CoM frame with respect to Ground frame

State Variable	Initial Condition	Description
$vx\,(t)$	10.0	X Component of Front Body Velocity [1]
$vy\,(t)$	0.0	Y Component of Front Body Velocity [1]
$vz\,(t)$	0.0	Z Component of Front Body Velocity [1]
$\omega 1\,(t)$	0.0	X Component of Front Body Angular Velocity [1]
$\omega 2\,(t)$	0.0	Y Component of Front Body Angular Velocity [1]
$\omega 3\,(t)$	0.0235	Z Component of Front Body Angular Velocity [1]
$\frac{d}{dt}xTire1\,(t)$	9.9711	FL Tire X Velocity [2]
$\frac{d}{dt}yTire1\,(t)$	0.0118	FL Tire Y Velocity [2]
$\frac{d}{dt}zTire1\,(t)$	0.0	FL Tire Z Velocity [2]
$\omega xTire1\,(t)$	0.0	X Component of FL Tire Angular Velocity [3]
$\omega yTire1\,(t)$	11.6594	Y Component of FL Tire Angular Velocity [3]
$\omega zTire1\,(t)$	0.0	Z Component of FL Tire Angular Velocity [3]
$\frac{d}{dt}xTire2\,(t)$	10.289	FR Tire X Velocity [2]
$\frac{d}{dt}yTire2\,(t)$	0.0118	FR Tire Y Velocity [2]
$\frac{d}{dt}zTire2\,(t)$	0.0	FR Tire Z Velocity [2]
$\omega xTire2\,(t)$	0.0	X Component of FR Tire Angular Velocity [3]
$\omega yTire2\,(t)$	11.6594	Y Component of FR Tire Angular Velocity [3]
$\omega zTire2\,(t)$	0.0	Z Component of FR Tire Angular Velocity [3]
$\frac{d}{dt}xTire3\,(t)$	10.384	RL Tire X Velocity [2]
$\frac{d}{dt}yTire3\,(t)$	0.0	RL Tire Y Velocity [2]
$\frac{d}{dt}zTire3\,(t)$	0.0	RL Tire Z Velocity [2]
$\omega xTire3\,(t)$	0.0	X Component of RL Tire Angular Velocity [3]
$\omega yTire3\,(t)$	11.6594	Y Component of RL Tire Angular Velocity [3]
$\omega zTire3\,(t)$	0.0	Z Component of RL Tire Angular Velocity [3]
$\frac{d}{dt}xTire4\,(t)$	9.9616	RR Tire X Velocity [2]
$\frac{d}{dt}yTire4\,(t)$	0.0252	RR Tire Y Velocity [2]
$\frac{d}{dt}zTire4\,(t)$	0.0	RR Tire Z Velocity [2]
$\omega xTire4\,(t)$	0.0	X Component of RR Tire Angular Velocity [3]
$\omega yTire4\,(t)$	11.6594	Y Component of RR Tire Angular Velocity [3]
$\omega zTire4\,(t)$	0.0	Z Component of RR Tire Angular Velocity [3]
$x\,(t)$	0.0	X Position of Front Body COM [2]
$y\,(t)$	0.0	Y Position of Front Body COM [2]
$z\,(t)$	1.4038	Z Position of Front Body COM [2]

Table A.16: Initial Conditions for Forestry Skidder — Sine Steer Event, Tire Edge in Tree

continued on next page

[1] expressed in front body CoM frame
[2] expressed in Ground frame
[3] expressed in tire CoM frame

continued from previous page

State Variable	Initial Condition	Description
$yaw\,(t)$	0.0	First 321 Euler Angle [4]
$pitch\,(t)$	0.0	Second 321 Euler Angle [4]
$roll\,(t)$	0.0	Third 321 Euler Angle [4]
$xTire1\,(t)$	0.5	X Position of FR Tire COM [2]
$yTire1\,(t)$	1.2275	Y Position of FR Tire COM [2]
$zTire1\,(t)$	0.8577	Z Position of FR Tire COM [2]
$yawTire1\,(t)$	0.0	First 312 Euler Angle [5]
$tiltTire1\,(t)$	0.0	Second 312 Euler Angle [5]
$spinTire1\,(t)$	0.0	Third 312 Euler Angle [5]
$xTire2\,(t)$	0.5	X Position of FR Tire COM [2]
$yTire2\,(t)$	-1.2275	Y Position of FR Tire COM [2]
$zTire2\,(t)$	0.8577	Z Position of FR Tire COM [2]
$yawTire2\,(t)$	0.0	First 312 Euler Angle [5]
$tiltTire2\,(t)$	0.0	Second 312 Euler Angle [5]
$spinTire2\,(t)$	0.0	Third 312 Euler Angle [5]
$xTire3\,(t)$	-2.954	X Position of RL Tire COM [2]
$yTire3\,(t)$	1.2275	Y Position of RL Tire COM [2]
$zTire3\,(t)$	0.8577	Z Position of RL Tire COM [2]
$yawTire3\,(t)$	0.0	First 312 Euler Angle [5]
$tiltTire3\,(t)$	0.0	Second 312 Euler Angle [5]
$spinTire3\,(t)$	0.0	Third 312 Euler Angle [5]
$xTire4\,(t)$	-2.954	X Position of RR Tire COM [2]
$yTire4\,(t)$	-1.2275	Y Position of RR Tire COM [2]
$zTire4\,(t)$	0.8577	Z Position of RR Tire COM [2]
$yawTire4\,(t)$	0.0	First 312 Euler Angle [5]
$tiltTire4\,(t)$	0.0	Second 312 Euler Angle [5]
$spinTire4\,(t)$	0.0	Third 312 Euler Angle [5]

Table A.16: Initial Conditions for Forestry Skidder — Sine Steer Event, Tire Edge in Tree

[4] used to track orientation of front body CoM frame with respect to Ground frame
[2] expressed in Ground frame
[5] used to track orientation of tire CoM frame with respect to Ground frame

State Variable	Initial Condition	Description
$vx\,(t)$	10.0	X Component of Front Body Velocity [1]
$vy\,(t)$	0.0	Y Component of Front Body Velocity [1]
$vz\,(t)$	0.0	Z Component of Front Body Velocity [1]
$\omega 1\,(t)$	0.0	X Component of Front Body Angular Velocity [1]
$\omega 2\,(t)$	0.0	Y Component of Front Body Angular Velocity [1]
$\omega 3\,(t)$	0.0	Z Component of Front Body Angular Velocity [1]
$\frac{d}{dt}xTire1\,(t)$	10.0	FL Tire X Velocity [2]
$\frac{d}{dt}yTire1\,(t)$	0.0	FL Tire Y Velocity [2]
$\frac{d}{dt}zTire1\,(t)$	0.0	FL Tire Z Velocity [2]
$\omega xTire1\,(t)$	0.0	X Component of FL Tire Angular Velocity [3]
$\omega yTire1\,(t)$	11.6594	Y Component of FL Tire Angular Velocity [3]
$\omega zTire1\,(t)$	0.0	Z Component of FL Tire Angular Velocity [3]
$\frac{d}{dt}xTire2\,(t)$	10.0	FR Tire X Velocity [2]
$\frac{d}{dt}yTire2\,(t)$	0.0	FR Tire Y Velocity [2]
$\frac{d}{dt}zTire2\,(t)$	0.0	FR Tire Z Velocity [2]
$\omega xTire2\,(t)$	0.0	X Component of FR Tire Angular Velocity [3]
$\omega yTire2\,(t)$	11.6594	Y Component of FR Tire Angular Velocity [3]
$\omega zTire2\,(t)$	0.0	Z Component of FR Tire Angular Velocity [3]
$\frac{d}{dt}xTire3\,(t)$	10.0	RL Tire X Velocity [2]
$\frac{d}{dt}yTire3\,(t)$	0.0	RL Tire Y Velocity [2]
$\frac{d}{dt}zTire3\,(t)$	0.0	RL Tire Z Velocity [2]
$\omega xTire3\,(t)$	0.0	X Component of RL Tire Angular Velocity [3]
$\omega yTire3\,(t)$	11.6594	Y Component of RL Tire Angular Velocity [3]
$\omega zTire3\,(t)$	0.0	Z Component of RL Tire Angular Velocity [3]
$\frac{d}{dt}xTire4\,(t)$	10.0	RR Tire X Velocity [2]
$\frac{d}{dt}yTire4\,(t)$	0.0	RR Tire Y Velocity [2]
$\frac{d}{dt}zTire4\,(t)$	0.0	RR Tire Z Velocity [2]
$\omega xTire4\,(t)$	0.0	X Component of RR Tire Angular Velocity [3]
$\omega yTire4\,(t)$	11.6594	Y Component of RR Tire Angular Velocity [3]
$\omega zTire4\,(t)$	0.0	Z Component of RR Tire Angular Velocity [3]
$x\,(t)$	0.0	X Position of Front Body COM [2]
$y\,(t)$	0.0	Y Position of Front Body COM [2]
$z\,(t)$	1.4038	Z Position of Front Body COM [2]

Table A.17: Initial Conditions for Forestry Skidder — Braking Event, Tire Edge in Tree

continued on next page

[1] expressed in front body CoM frame
[2] expressed in Ground frame
[3] expressed in tire CoM frame

continued from previous page

State Variable	Initial Condition	Description
$yaw\,(t)$	0.0	First 321 Euler Angle [4]
$pitch\,(t)$	0.0	Second 321 Euler Angle [4]
$roll\,(t)$	0.0	Third 321 Euler Angle [4]
$xTire1\,(t)$	0.5	X Position of FR Tire COM [2]
$yTire1\,(t)$	1.2275	Y Position of FR Tire COM [2]
$zTire1\,(t)$	0.8577	Z Position of FR Tire COM [2]
$yawTire1\,(t)$	0.0	First 312 Euler Angle [5]
$tiltTire1\,(t)$	0.0	Second 312 Euler Angle [5]
$spinTire1\,(t)$	0.0	Third 312 Euler Angle [5]
$xTire2\,(t)$	0.5	X Position of FR Tire COM [2]
$yTire2\,(t)$	-1.2275	Y Position of FR Tire COM [2]
$zTire2\,(t)$	0.8577	Z Position of FR Tire COM [2]
$yawTire2\,(t)$	0.0	First 312 Euler Angle [5]
$tiltTire2\,(t)$	0.0	Second 312 Euler Angle [5]
$spinTire2\,(t)$	0.0	Third 312 Euler Angle [5]
$xTire3\,(t)$	-2.954	X Position of RL Tire COM [2]
$yTire3\,(t)$	1.2275	Y Position of RL Tire COM [2]
$zTire3\,(t)$	0.8577	Z Position of RL Tire COM [2]
$yawTire3\,(t)$	0.0	First 312 Euler Angle [5]
$tiltTire3\,(t)$	0.0	Second 312 Euler Angle [5]
$spinTire3\,(t)$	0.0	Third 312 Euler Angle [5]
$xTire4\,(t)$	-2.954	X Position of RR Tire COM [2]
$yTire4\,(t)$	-1.2275	Y Position of RR Tire COM [2]
$zTire4\,(t)$	0.8577	Z Position of RR Tire COM [2]
$yawTire4\,(t)$	0.0	First 312 Euler Angle [5]
$tiltTire4\,(t)$	0.0	Second 312 Euler Angle [5]
$spinTire4\,(t)$	0.0	Third 312 Euler Angle [5]

Table A.17: Initial Conditions for Forestry Skidder — Braking Event, Tire Edge in Tree

[4] used to track orientation of front body CoM frame with respect to Ground frame
[2] expressed in Ground frame
[5] used to track orientation of tire CoM frame with respect to Ground frame

Bibliography

[1] C.H An, C.G. Atkeson, and J.M. Hollerbach. *Model Based Control of a Robot Manipulator*. The MIT Press, Cambridge, Massachusetts, 1988.

[2] G. Andrews and H. Kesavan. The vector-network model: A new approach to vector dynamics. *Mechanism and Machine Theory*, 10:57–75, 1975.

[3] E. Bakker, L. Nyborg, and H.B. Pacejka. Tyre modelling for use in vehicle dynamics studies. *SAE Technical Paper Series*, (870421), 1986.

[4] J. Baumgarte. Stabilization of constrains and integrals of motion. *Computer Methods in Applied Mechanics*, 1:1 – 16, 1972.

[5] J.E. Bernard and C.L. Clover. Tire modelling for low-speed and high-speed calculations. *SAE Technical Paper Series*, (950311), 1995.

[6] M. Blundell and D. Harty. *The Multibody Systems Approach to Vehicle Dynamics*. Society of Automotive Engineers, Inc., Warrendale, PA, 2004.

[7] E. Carpanzano and C. Maffezzoni. Symbolic manipulation techniques for model simplification in object-oriented modelling of large scale continuous systems. *Mathematics and Computers in Simulation*, 48(2):133 – 50, 1998.

[8] Cosin Consulting, www.ftire.com. *FTire Flexible Ring Tire Model Documentation and User's Guide*, 2007.

[9] J. Cuadrado, J. Cardenal, and E. Bayo. Modeling and solution methods for efficient real-time simulation of multibody dynamics. *Multibody System Dynamics*, 1(3):259 – 280, 1997.

[10] J.G. de Jalón and E. Bayo. *Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge*. Springer-Verlag, New York, 1994.

[11] H. Elmqvist, F.E. Cellier, and M. Otter. Object-oriented modeling of hybrid systems. *Proceedings of the 1993 European Simulation Symposium*, 1993.

[12] H. Elmqvist, S.E. Mattsson, H. Olsson, J. Andreasson, M. Otter, C. Schweiger, and D. Brück. Realtime simulation of detailed vehicle and powertrain dynamics. *SAE Technical Paper Series*, (2004-01-0768), 2004.

[13] H. Elmqvist and H. Olsson. New methods for hardware-in-the-loop simulation of stiff models. In *Proceedings of the 2nd International Modelica Conference*, pages 59 – 64, 2002.

[14] P. Fisette, T. Postiau, L. Sass, and J.C. Samin. Fully symbolic generation of complex multibody models. *Mechanics of Structures and Machines*, 30(1):31 – 82, 2002.

[15] International Organization for Standardization. Iso 8855, road vehicles – vehicle dynamics and road holding ability – vocabulary, 1991.

[16] T.D. Gillepsie. *Fundamentals of Vehicle Dynamics*. Society of Automotive Engineers, Inc., Warrendale, PA, 1992.

[17] E.J. Haug. *Computer Aided Kinematics and Dynamics of Mechanical Systems*. Allyn and Bacon, Needham Heights, MA, 1989.

[18] Y. He, A. Khajepour, J. McPhee, and X. Wang. Dynamic modelling and stability analysis of articulated frame steer vehicles. *International Journal of Heavy Vehicle Systems*, 12(1):28 – 59, 2005.

[19] MotionPro Inc. *DynaFlexPro User's Manual*. Maplesoft, http://www.maplesoft.com/dynaflexpro, 2006.

[20] A. Kecskemethy, T. Krupp, and M. Hiller. Symbolic processing of multiloop mechanism dynamics using closed-form kinematics solutions. *Multibody System Dynamics*, 1(1):23 – 45, 1997.

[21] W. Kortüm and R.S. Sharp, editors. *Multibody Computer Codes in Vehicle System Dynamics*. Lisse, Swetz, and Zeitlinger, 1993.

[22] J-Y. Lee, J-S. Lee, and W-S. Lee. A symbolic computation method for automatic generation of a full vehicle model simulation code for a driving simulator. In *Proceedings of the Asian Conference on Multibody Dynamics*, 2004.

[23] Maplesoft. *Maple 10 User's Manual*, 2006.

[24] J. McPhee. Automatic generation of motion equations for planar mechanical systems using the new set of "branch coordinates". *Mechanism and Machine Theory*, 33:805–823, 1998.

[25] J. McPhee and S. Redmond. Modelling multibody systems with indirect coordinates. *Computer Methods in Applied Mechanics and Engineering*, in press, 2005.

[26] J. McPhee, C. Schmitke, and S. Redmond. Dynamic modelling of mechatronic multibody systems with symbolic computing and linear graph theory. *Mathematical and Computer Modelling of Dynamical Systems*, 10(1):17 – 23, 2004.

[27] Mechanical Simulation Corporation. *TruckSim Version 6: Math Models*, 2006.

[28] Mechanical Simulation Corporation. *Version 6 CarSim Math Model Features*, 2006.

[29] P. Mitiguy and T. Kane. Motion variables leading to efficient equations of motion. *International Journal of Robotics Research*, 15(5):522 – 532, 1996.

[30] MSC.Software Corporation. *ADAMS/Car RealTime Manual*, 2005.

[31] MSC.Software Corporation. *ADAMS/Solver Manual*, 2005.

[32] MSC.Software Corporation. *ADAMS/Tire Manual*, 2005.

[33] M. Otter, H. Elmqvist, and S.E. Mattsson. The new modelica multibody library. In *Proceedings of the 3rd International Modelica Conference*, pages 311 – 330, 2003.

[34] H.B. Pacejka. *Tire and Vehicle Dynamics*. Society of Automotive Engineers, Inc., Warrendale, PA, 2002.

[35] H.B. Pacejka and R.S. Sharp. Shear force development by pneumatic tyres in steady state conditions. a review of modelling aspects. *Vehicle System Dynamics*, 20(3-4):121 – 176, 1991.

[36] V.J. Pesch. Optimization of planar mechanism kinematics with symbolic computation. Master's thesis, University of Illinois at Urbana-Champaign, 1994.

[37] T. Postiau, L. Sass, P. Fisette, and J-C. Samin. High-performance multibody models of road vehicles: Fully symbolic implementation and parallel computation. *20th International Congress of Theoretical and Applied Mechanics*, pages 57 – 83, 2000.

[38] QNX Software Systems, www.qnx.com. *QNX Neutrino RTOS - At a Glance*, 2006.

[39] C. Rudolf. Road vehicle modelling using symbolic multibody dynamics. Diploma thesis, University of Karlsruhe, 2003.

[40] W. Rulka and E. Pankiewicz. Mbs approach to generate equations of motion for hil-simulations in vehicle dynamics. *Multibody Systems Dynamics*, 14(3-4):367 – 386, 2005.

[41] M.W. Sayers. *Symbolic Computer Methods to Automatically Formulate Vehicle Simulation Codes*. Phd thesis, University of Michigan, 1990.

[42] M.W. Sayers. Vehicle models for rts applications. *Vehicle System Dynamics*, 32(4):421 – 438, 1999.

[43] M.W. Sayers and D. Han. Generic multibody vehicle model for simulating handling and braking. *Vehicle System Dynamics*, 25(Suppl):599 – 613, 1996.

[44] M.W. Sayers and A. Stribersky. Computer algebra in nonlinear analyses of the straightline stability of combination vehicles. *Computers and Structures*, 44(1-2):179 – 86, 1992.

[45] C. Schmitke. *Modelling Multibody Multi-Domain Systems Using Subsystems and Linear Graph Theory.* Phd thesis, University of Waterloo, 2004.

[46] C. Schmitke and J. McPhee. Forming equivalent subsystem components to facilitate the modelling of mechatronic multibody systems. *Multibody System Dynamics*, 14:81–109, 2005.

[47] C. Schmitke and J. McPhee. Using linear graph theory and the principle of orthogonality to model multibody, multi-domain systems. *Advanced Engineering Informatics*, 2007.

[48] H. Schuette and P. Waeltermann. Hardware-in-the-loop testing of vehicle dynamics controllers - a technical survey. *SAE Technical Paper Series*, (2005-01-1660), 2005.

[49] R.S. Sharp. Application of multi-body computer codes to road vehicle dynamics modelling problems. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 208(1):55 – 61, 1994.

[50] T Shiiba and Y. Suda. Real-time multibody analysis environment for driving simulator. In *ASME 2005 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, 2005.

[51] Tesis DYNAware, http://www.tesis.de. *ve-Dyna User's Guide*, 2006.

[52] the Mathworks, http://www.mathworks.com. *Real-Time Workshop Targer Language Compiler*, 2006.

[53] the Mathworks, http://www.mathworks.com. *Real-Time Workshop User's Guide*, 2006.

[54] J.J.M. van Oosten, H.-J. Unrau, A. Riedel, and A. Bakker. Standardization in tire modeling and tire testing — tydex workgroup, time project. *Tire Science and Technology*, 27(3):188 – 202, 1999.

[55] H. Vogt, C. Schmitke, K. Jalali, and J. McPhee. Real-time dynamics simulation of an electric vehicle with in-wheel motors. In *Proceedings of the Multibody Dynamics Conference*, 2007.

[56] J.Y. Wong. *Theory of Ground Vehicles.* John Wiley and Sons, Inc., New York, NY, second edition, 1993.