

# Real-Time Multi-Body Vehicle Dynamics Using A Modular Modeling Methodology

Richard Romano

Realtime Technologies, Inc.

Copyright © 2003 Society of Automotive Engineers, Inc.

## ABSTRACT

Simulations of ground vehicles are extensively used by military and commercial vehicle developers to aid in the design process. In the past, ground vehicle simulations have focused on non-real-time models. However with the advancement of computers and modeling methodologies, real-time multi-body models have become one of the standard tools used by vehicle developers. Multi-body models are composed of joint, body, and force elements which map well into a modular modeling approach. Based on recursive techniques a set of reusable components were developed for use in a graphical simulation and modeling environment. The components were then connected to form a real-time multi-body model of a Ford Taurus. Finally, the Taurus model was integrated with simulator cueing subsystems to build a complete driving simulator. The performance of the Taurus model was compared with test data. It was found that the vehicle model was both accurate and ran much faster than real-time. Due to the model formulation, the current set of modular components are limited to modeling open treed systems with either a fixed or mobile base body.

## INTRODUCTION

Over the past twenty years offline vehicle dynamics simulations have been used extensively in the automotive industry. As computer performance increased vehicle dynamics applications were introduced into hardware in the loop simulation and operator in the loop simulation laboratories. To meet the needs of vehicle designers the complexity and fidelity of the vehicle dynamics used in operator in the loop simulation has been increasing and typically multi-body dynamics are used in this environment. Multi-body dynamics is based on classical mechanics. The multi-body method utilizes a finite set of elements including rigid bodies, joints, springs, dampers, and actuators. In the first steps of standardization of a datamodel of multi-body systems for use with computer codes, the following assumptions have been agreed upon [1]:

1. A multi-body system consists of rigid bodies and ideal joints. A body may degenerate to a particle or to a body without inertia.
2. The topology of the multi-body system is arbitrary. Chains, trees and closed loops are admitted.
3. Bodies, joints and actuators are summarized in libraries of standard elements.

Kecskemethy [2] developed a formalization of representing multi-body elements using a modular dataflow representation. Figure 1 shows a Rigid Link and Joint. In the figure,  $q$  is the position vector of the frame. The derivatives of  $q$  contain the velocities and accelerations of the frame.  $Q$  represents the generalized forces in each coordinate frame. The generalized forces are made up of both inertial forces and external forces and are generated by mass, spring, damper, or active elements as given in Figure 2.

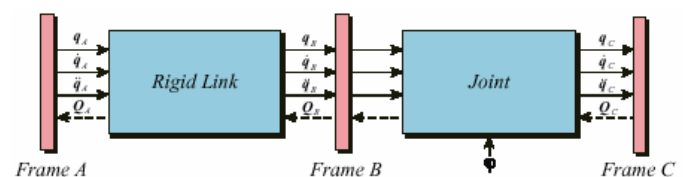


Figure 1: Rigid Link and Joint (from Krebs [3])

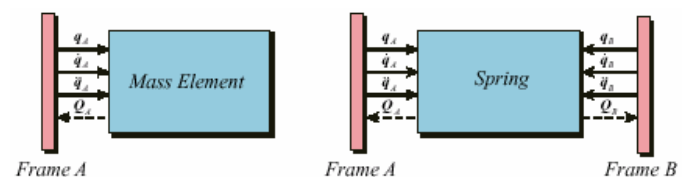


Figure 2: Mass Element and Spring (from Krebs [3])

Each element provides a function  $f$  mapping the position  $q$  from the current frame to the next frame. The mapping functions are typically straight forward.

Kecskemethy [2] and Craig [4] give good descriptions of mapping functions for revolute and prismatic joints. In the case of a joint, the function  $f$  takes into account the joint position, velocity and acceleration. As given by Krebs [3], the position  $q$  and its derivatives in Frame B can be calculated from Frame A as:

$$\begin{aligned} q_B &= f(q_A) \\ \dot{q}_B &= J_f \cdot \dot{q}_A \\ \ddot{q}_B &= J_f \cdot \ddot{q}_A + \dot{J}_f \cdot \dot{q}_A \end{aligned}$$

and the generalized forces of Frame A can be calculated from Frame B as:

$$Q_A = J_f^T \cdot Q_B$$

where:

$$J_f = \frac{\partial f}{\partial q_A}$$

is the Jacobian matrix.

The approach used in this paper is to develop a set of software components that represent links, joints, bodies, and actuators as defined in Figure 1 and 2. By calling the software components in a particular order a multi-body structure can be developed as a set of link, joint, mass, and force elements. By calculating inside each joint software component a joint torque  $\tau$  as the dot product between the generalized force  $Q$  acting on the joint and the joint axis, the set of joint torques required to generate a particular mechanism acceleration, position and velocity can be determined. The resulting software structure solves the inverse dynamics problem: what joint torques are required to yield a particular set of positions, velocities, and accelerations of a mechanism. Introducing a set of generalized coordinates:  $\Theta$ , that represent joint angles and position, in general the set of joint torques  $\tau$  required will take the following form:

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta)$$

where  $M$  is defined as the mass matrix,  $V$  represents centrifugal and coriolis effects, and  $G$  represents gravity effects. With a set of software that computationally solves the inverse dynamics problem, the difficulty is solving the forward dynamics problem. That is given a set of joint torques what is the resulting motion of the actuator. Walker and Orin [5] developed several methods of solving for the joint accelerations  $\ddot{\Theta}$  given a set of joint torques. Once the joint accelerations are known the joint velocities and positions can be calculated using numerical integration techniques. The method used in this paper is similar to Method 3 in [5].

For a particular time step the method can be described as follows:

1. Using the joint positions and velocities from the previous step calculate a torque bias vector  $b$  that is the torque at each joint required for zero joint acceleration:  $b = V(\Theta, \dot{\Theta}) + G(\Theta)$ .
2. With the joint velocities and gravity set to zero (i.e. with the torque bias vector set to zero), calculate the joint torque vector for a unit joint acceleration for each joint separately (i.e. all other joint accelerations set to zero). The joint torques calculated are essentially a column of the mass matrix. Once all the columns of the mass matrix have been computed, and given the actual torques acting at each joint, the acceleration of the joints can be calculated as:

$$\ddot{\Theta} = M^{-1}(\Theta)(\tau - b)$$

The joint accelerations can then be integrated to yield the joint positions and velocities for the next step.

## IMPLEMENTATION

The links, joints, bodies, and actuators were developed and assembled as modular components in SimCreator a graphical simulation tool from Realtime Technologies, Inc. The link or offset component and a revolute mechanism are shown in Figure 3.

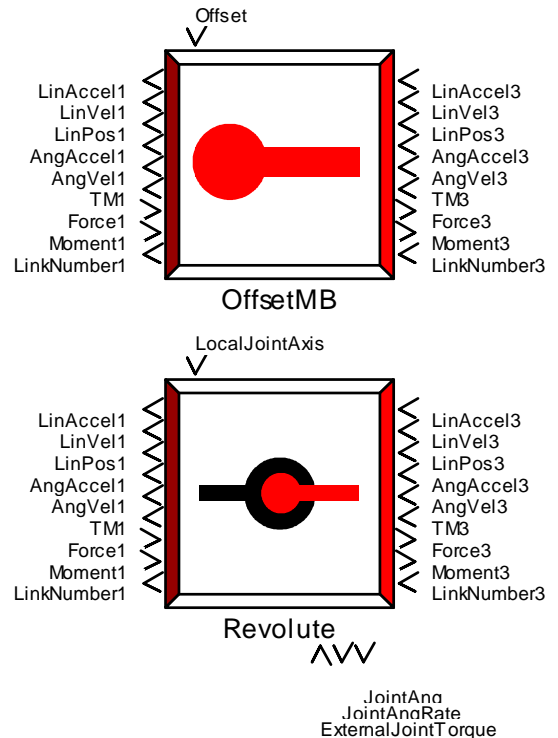


Figure 3: SimCreator Based Link and Joint

In the SimCreator components in Figure 3, the data passed downstream are the linear acceleration, velocity and position of the frame (LinPos1, LinVel1, LinAccel1), and the angular acceleration, velocity and transformation matrix of the frame (AngAccel1, AngVel1, TM1). The generalized forces are passed back upstream in Force1 and Moment1. Finally a set of link numbers are passed downstream. These itemized the set of joints that are upstream of the current component. In this way joint forces and accelerations can be associated with the correct set of joints. The velocity and position vectors (LinVel1, LinPos1, and AngVel1) are each of length three, containing the component values for three dimensions. The acceleration vectors, forces and moments are each a 3xN matrix. Each index across the acceleration matrix represents the acceleration in the frame due to a unit acceleration of a particular joint. Each index across the forces and moments matrices represents the generalized forces due to a unit acceleration of a particular joint. These matrices are used to calculate the columns of the mass matrix. Due to the model formulation, the current set of modular components are limited to modeling open treed systems with either a fixed or mobile base body.

**VALIDATION**

A double pendulum was modeled using the developed components. This is shown in Figure 4.

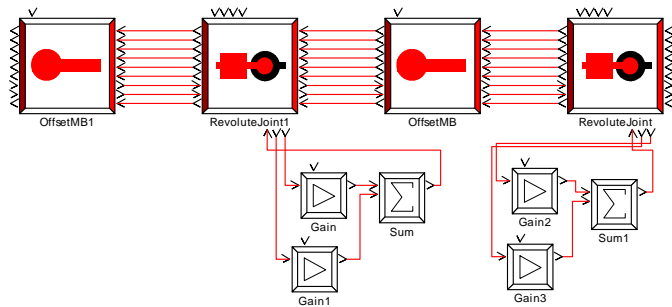


Figure 4: Double Pendulum

The double pendulum was configured to have the same geometry as Hwang [6]. Hwang performed a dynamic analysis of a double pendulum using DADS and two other multi-body dynamics packages. The result of their DADS model is given in Figure 5. The results of the SimCreator model are given in Figure 6. Comparison of the results of several plots generated in SimCreator showed excellent agreement with Hwang.

**VEHICLE MODEL**

A vehicle dynamics model was developed similar to Sayers [7]. The vehicle was modeled with a base body with six degrees of freedom and four prismatic joints representing each of the corners of the vehicle. The prismatic joints were tilted to take into account anti-dive, anti-squat geometry and roll center height. The vehicle

model is shown in Figure 8. The powertrain, shown in Figure 7, and other subsystems were modeled in a similar fashion to previous vehicle models developed using SimCreator [8].

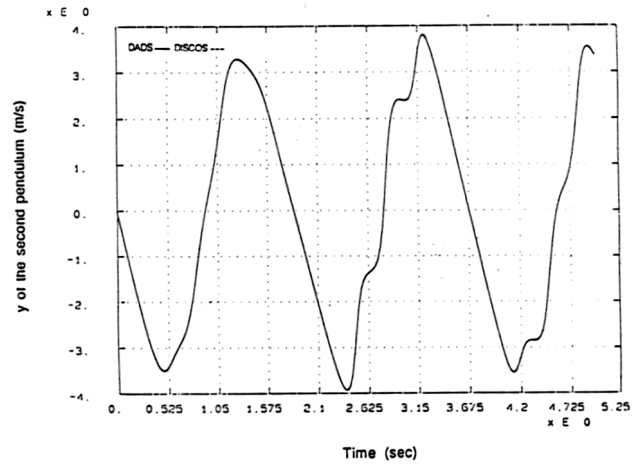


Figure 5: DADS Position of Second Pendulum (from Hwang [6])

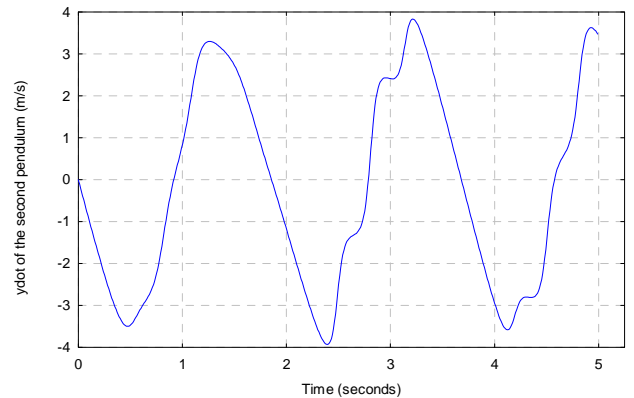


Figure 6: SimCreator Position of Second Pendulum

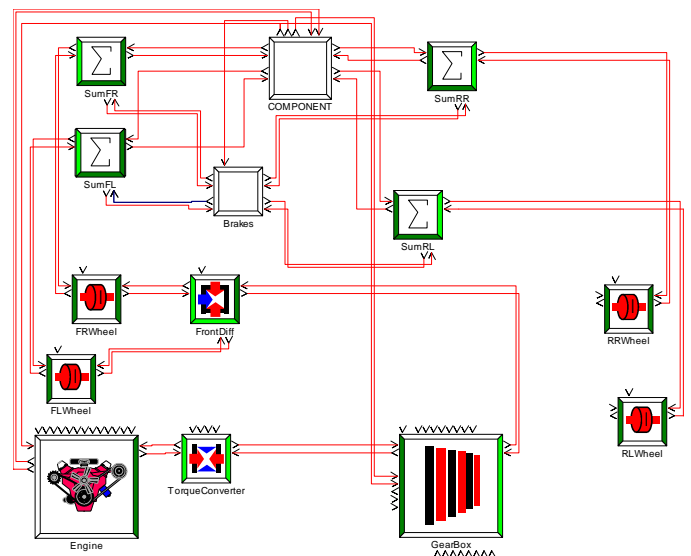


Figure 4: Front Wheel Drive Powertrain

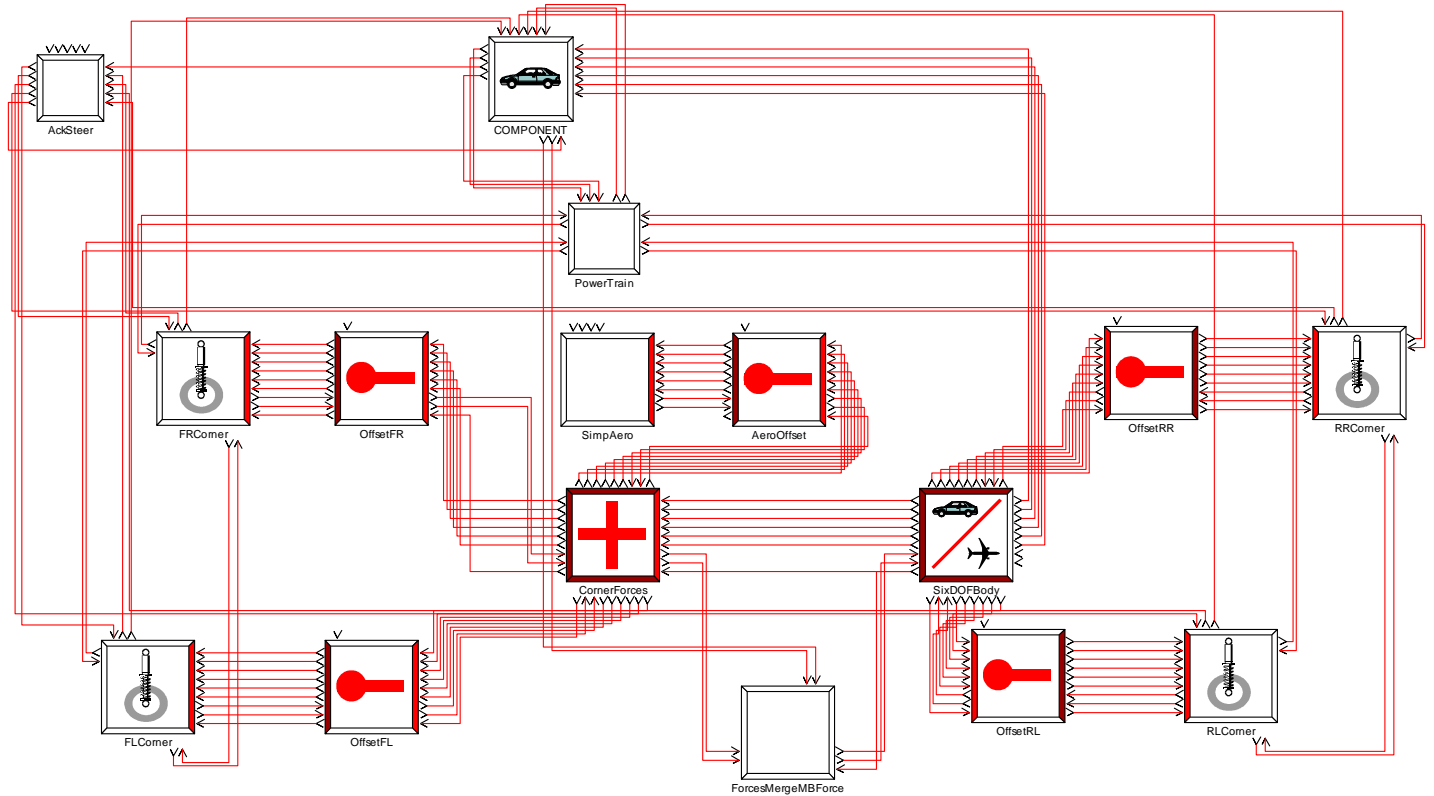


Figure 8: Complete Vehicle Dynamics

SimCreator components shown in Figure 8 are as follows: PowerTrain – calculates all powertrain and brake system effects and outputs the four wheel speeds of the vehicle, SimpAero – calculates the aerodynamic forces on the vehicle, SixDOFBody – is the base body of the multi-body dynamics model, Offset – calculates the position, velocity and acceleration at an offset and translates the generalized forces at the offset back to the center of gravity, CornerForces – sums up the generalized forces of the multi-body tree structure, Corner – calculates the independent suspension and the tire model at a vehicle corner, and outputs the forces acting at the corner and the torque acting on the wheel. The Corner module contains inside it the prismatic joint as well as a tire model.

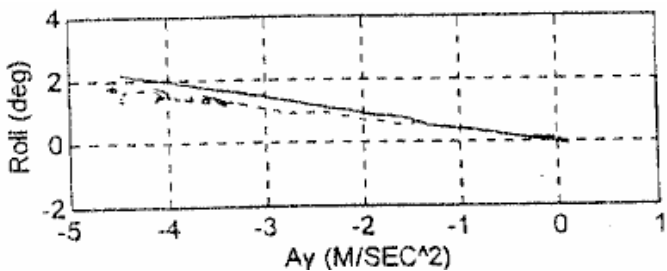


Figure 9: Lateral Response of Ford Taurus (from Salaani [10])

The vehicle dynamics model was configured to represent a Ford Taurus using parameters collected by Salaani et al [9][10][11]. The Ford Taurus vehicle parameters collected by Salaani included mass

properties, offsets, suspension and tire characteristics, and powertrain information. In addition to parameters, Salaani collected test data from an actual vehicle. The test data was used to validate the Ford Taurus model. Figures 9 and 10 show the roll response of the model to a lateral handling maneuver using a smooth low rate steering input at 32 m/s. The dotted line in Figure 9 represents test data. The results match up well with the roll angles predicted by the SimCreator model in Figure 10. A variety of maneuvers and responses were tested and compared with Salaani [10]. Results were found to be reasonably accurate given that all of the vehicle parameters and all the test inputs (steering wheel, accelerator and brake force) were not available.

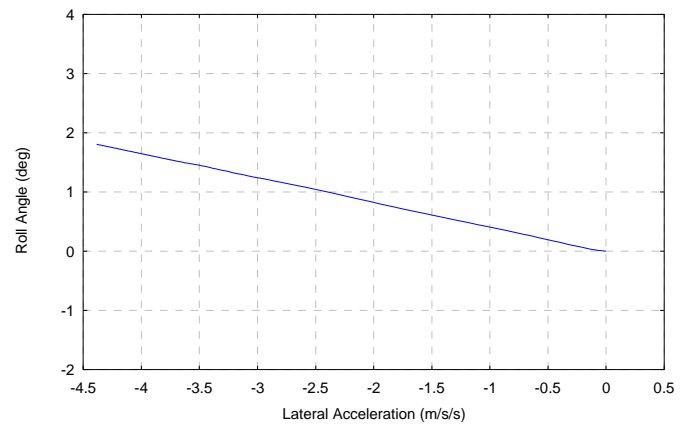


Figure 10: Lateral Response of SimCreator Model

## DRIVING SIMULATOR

The multi-body vehicle dynamics were inserted into a driving simulator to test its use in an operator in the loop

environment. A typical driving simulator has the following cueing systems: audio, out the window graphics, driving control interface, and motion. These can be encapsulated as modules and integrated with the vehicle dynamics to form the driving simulator shown in Figure 11.

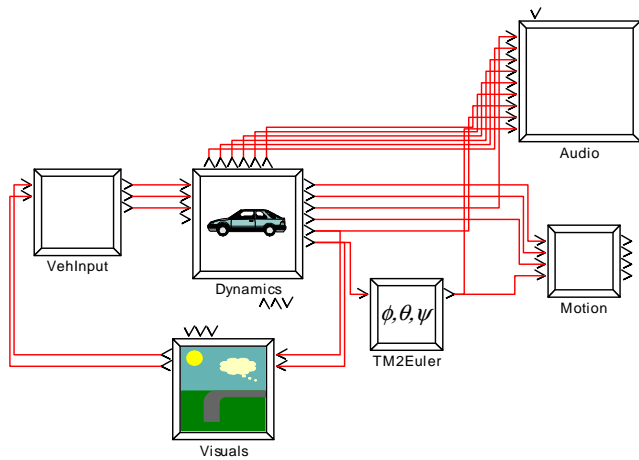


Figure 11: Driving Simulator

## PERFORMANCE

The vehicle dynamics shown in Figure 8 contains 61 states including the powertrain. They were integrated using a Runge Kutta second order method. Running the vehicle dynamics on a single 600 MHz Pentium III processor was found to take 0.2125 ms per update. A typical update rate for the model is 2.5 ms leaving plenty of time for other calculations.

## ADVANTAGES OF THE GRAPHICAL ENVIRONMENT

There are several advantages to using a graphical simulation tool. The simulation tool provides a structured framework to build C Code components in. The simulation tool executes the multi-body components in the model such that all data required for the current component has been calculated in previous components. Therefore the order of calculation of position, velocity, and force analysis steps, which would normally be laid out explicitly in recursive dynamics formulations, are automatically determined by the framework. Graphical simulation tools provide a unified integration algorithm with the ability to globally set integration methods and integration time steps. It is typically easy to add states to the integrator from any component. Any component output can be easily collected as data during the simulation and plotted. Inputs to the model and initial conditions of the states can be accessed from a single user interface. The graphical environment allows users greater insight into the model being developed. It also allows easy connections between components and provides for a hierarchical view of the model. In addition components and models are fully encapsulated and easy to share

between users. This encourages component and model reuse.

## CONCLUSION

It was found that a multi-body vehicle dynamics model and a driving simulator could be designed using modular components. Links, joints, masses, and forces mapped well into independent components. The multi-body components were found to accurately model both a double pendulum and a Ford Taurus. The performance of the model was found to be extremely efficient taking only 0.2125 ms per time step to model the Ford Taurus on a 600 MHz Pentium III. This makes the model usable for real-time simulation.

## CONTACT

Richard Romano has been working in driving simulation for twelve years focusing on motion cueing, vehicle dynamics, and human factors research. Dr. Romano was the manager of simulator research and development at the Iowa Driving Simulator and supervised the brake system simulation group at ITT Automotive. He is now president of Realtime Technologies, Inc. He may be contacted at raromano@ix.netcom.com.

## REFERENCES

1. Schiehlen, W., "Multibody System Dynamics: Roots and Perspectives," *Multibody System Dynamics* 1: 149-188, 1997.
2. Kinematics and Dynamics of Multi-Body Systems, Eds. J. Angeles and A. Kecskeméthy, CISM Courses and Lectures No. 360, Springer-Verlag, Wien, New York, 1995.
3. Krebs, M., "Vehicle Modeling for High-Dynamic Driving Simulator Applications," *Proceedings of the 1st Human-Centered Transportation Simulation Conference*, Iowa City, IA, 2001.
4. Craig, J.J., *Introduction to Robotics Mechanics and Control*, Addison Wesley, New York, 1989.
5. Walker, M. W. and D. E. Orin., "Efficient Dynamic Computer Simulation of Robotic Mechanisms," *ASME Journal of Dynamic Systems, Measurements and Control*, Vol. 104, 1982, pp. 205-211.
6. Hwang, H.Y., Kim, S.S., Haug, E.J., and H.J. Lai, "Theoretical Cross Verifications and Comparative Studies of Multibody Simulation Codes DADS, DISCOS, and Contops," *Technical Report R-66*, Center for Simulation and Design Optimization, University of Iowa, 1988.
7. Sayers, M. W. and D. Han. A Generic Multibody Vehicle Model for Simulating Handling and Braking. *Journal of Vehicle System Dynamics*, Supplement 25, 1996, pp. 599-613.
8. Romano, R., "Realtime Driving Simulation Using A Modular Modeling Methodology", *SAE Technical Paper Series No. 2000-01-1297*, March 2000.
9. Salaani, M.K., "Parameter Measurement and Development of a NADSdyna Validation Data Set for a 1994 Ford Taurus", *SAE Technical Paper Series No. 970564*, February, 1997.
10. Salaani, M.K., Heydinger, G.J., and D.A. Guenther, "Validation Results from Using NADSdyna Vehicle Dynamics Simulation", *SAE Technical Paper Series No. 970565*, February, 1997.

11. Salaani, M.K., and G.J. Heydinger, "Powertrain and Brake Modeling of the 1994 Ford Taurus for the National Advanced Driving Simulator", SAE Technical Paper Series No. 981190, March, 1998.