Truck or Bus Dynamic Modeling for a Driving Simulator

Final Report

Z. Bareket
P. Fancher

UMTRI The University of Michigan Transportation Research Institute

DISCLAIMER

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
UMTRI-91-26		

4. Title and Subtitle	5. Report Date
Truck or Bus Dynamic Modeling for a Driving Simulator	June, 1991
	6. Performing Organization Code
	8. Performing Organization Report No.
7. Author(s) **Z. Bareket, P. Fancher**	
9. Performing Organization Name and Address The University of Michigan Transportation Research Institute 2901 Baxter Road, Ann Arbor, Michigan 48109	10. Work Unit No. (TRAIS) 11. Contract or Grant No.
12. Sponsoring Agency Name and Address Great Lakes Center for Truck Transportation Research 210 UMTRI Building 2901 Baxter Road Ann Arbor, Michigan 48109-2150	13. Type of Report and Period Covered **Final Report** 14. Sponsoring Agency Code

15. Supplementary Notes

Supported by a grant from the U.S. Department of Transportation, University Transportation Center Program.

16. Abstract

The development and features of a yaw-plane computer simulation model for a driving simulator are described in this document. The simulated model is capable of predicting responses due to steering, braking, and accelerating. For demonstration purposes, a relatively complex model of a tractor-semitrailer is studied

Virtues and capabilities of an enhanced, full 3-dimensional model are then discussed from the aspect of a driving simulator utilization.

"AUTOSIM," a new program developed at UMTRI for modeling dynamic systems, was used. AUTOSIM allows the analyst to write a description of the dynamic system, and have the computer automatically generate the simulation code. The engineer can then use the program to study the system.

17. Key Words	18. Distribution Statement
Simulation, driving simulator, dynamic model, AUTOSIM, equations of motion, truck, bus	Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
None	None	108	

Preface

The recent emphasis on more comprehensive training of truck drivers has spawned interest in development of driver simulators for training applications. A practical simulator must necessarily be low in cost, and realistically replicate the dynamic behavior of a truck. Several commercial companies are currently developing such systems.

In a driving simulator a computer is used to "calculate" the motions of the vehicle in response to driver inputs. Although comprehensive computer models of trucks and buses are available, simplified versions of the models are needed to reduce the computational requirements for a low-cost simulator. This project represents an effort to develop a simplified, but flexible, truck/bus model for use in a driving simulator in collaboration with commercial developers of the systems.

The model is unique in its simplicity, in its provision for real-time input of driver control actions, and in the flexibility to upgrade the model through auxiliary subroutines as greater computational capability becomes available. As developed here, the program can operate at faster than real time on 32-bit microprocessor machines such as the IBM "386" class desktop computers and the Macintosh II family.

—Thomas D. Gillespie

Director, Great Lakes Center for
Truck Transportation Research

TABLE OF CONTENTS

NOMENCLATURE

Symbol	Definition
α	Slip angle, degrees
a_{x1}	Longitudinal acceleration tractor, g
a_{y1}	Lateral acceleration tractor, g
a_{x2}	Longitudinal acceleration trailer, g
a_{y2}	Lateral acceleration trailer, g
A	Parametric coefficient in the tire model, 1/deg.
B	Parametric coefficient in the tire model, 1/(lb-deg.)
BG_i	Braking gain of axle i, in-lb/psi
BP	Braking system pressure, psi
C_α	Cornering stiffness, lb/deg
F_A	Demanded acceleration tire force, lb
F_B	Demanded braking tire force, lb
F_{zai}	Vertical load of axle i, lb
F_{zais}	Static vertical load of axle i, lb
F_{zi}	Total vertical tire load at point i, lb
F_{zis}	Vertical load on one tire at a position, lb

F_X	Longitudinal tire force, lb
F_Y	Lateral tire force, lb
g	The acceleration of gravity
K_f	Pitch stiffness of the tractor, in-lb/rad
h_1	C. G height of tractor, in
h_2	C. G height of trailer, in
h_c	Fifth wheel height, in
HP	Maximum engine power, horsepower
$K_{\phi i}$	Roll stiffness of axle i, in-lb/rad
L_1	Tractor wheelbase, in
L_{1h}	Distance from tractor front axle to hitch, in
L_2	Trailer wheelbase, in
M_1	Tractor mass, lb-sec^2/in
M_2	Trailer mass, lb-sec^2/in
ϕ	Roll angle, deg.
θ	Pitch angle, deg.
μ	Tire/road frictional coupling
μ_s	Sliding frictional coupling

N_{ti}	Number of tires at position i
PP	Pedal position (brake or accelerator), range {0 to 1}
Q(1), Q(2), Q(3), Q(4)	Translations and rotations (generalized coordinates). (See Figures 1, 2)
T_1	Track of tractor front axle, in
T_2	Track of tractor rear axle, in
T_3	Track of trailer axle, in
TR	Tire radius, in
U(1), U(2), U(3), U(4)	Velocities defined by axes (generalized speeds). (See Figures 1, 2)
W_1	Tractor weight, lb
W_2	Trailer weight, lb
X_{11}	Distance from tractor mass center to front axle, in
X_{2f}	Distance from trailer mass center to hitch, in

1.0 INTRODUCTION

The objective of this work is to generate a yaw-plane computer simulation model for a driving simulator. The model is required to be capable of predicting responses due to steering, braking and accelerating of trucks or buses. To demonstrate these capabilities, a model of a tractor-semitrailer was developed and studied.

Previously developed simulation models for tractor-semitrailer configurations exist at UMTRI. Those models differ from the one described in this report in two major aspects. First, some of them lack the braking-accelerating capability, and second, some are more complicated and detailed in performing the calculations, and therefore take longer to run. One important incentive for a short run-time is due to a prospective application for this model—a real-time driving simulator. In the driving simulator the control inputs such as steering, braking, and accelerating would be entered from the driver through hardware controls whose signals would be digitized. Therefore, a flexible way to input control parameters is important.

A possible layout for such a driving simulator is presented in Figure 1. Such a simulator, if successfully designed and built, would have three major features:

- Real-time response, display, and feedback.
- Compact packaging.
- Accurate and clear responses, so that it can serve as a system for studying the driving process.

Figure 1 presents all the elements and interactions comprising a driving simulator system. The yaw-plane computer simulation model described in this report is represented by the block labelled "Equations of Motion Solver" in Figure 1. Clearly, there are many other important elements of the driving simulator system. Nevertheless, the Equations of Motion Solver is central to the vehicle-oriented perspective of this report.

The subject tractor has three degrees of freedom in the horizontal plane. Specifically, it can move along both horizontal axes and rotate about its vertical axis. A fourth coordinate is the trailer rotation relative to the tractor. The model employs a quasi-static approach in which lateral and longitudinal accelerations due to the planar motion are used to calculate lateral and longitudinal load transfers. In a sense, the model only provides virtual roll and pitch, since it does not have true degrees of freedom for these motions.

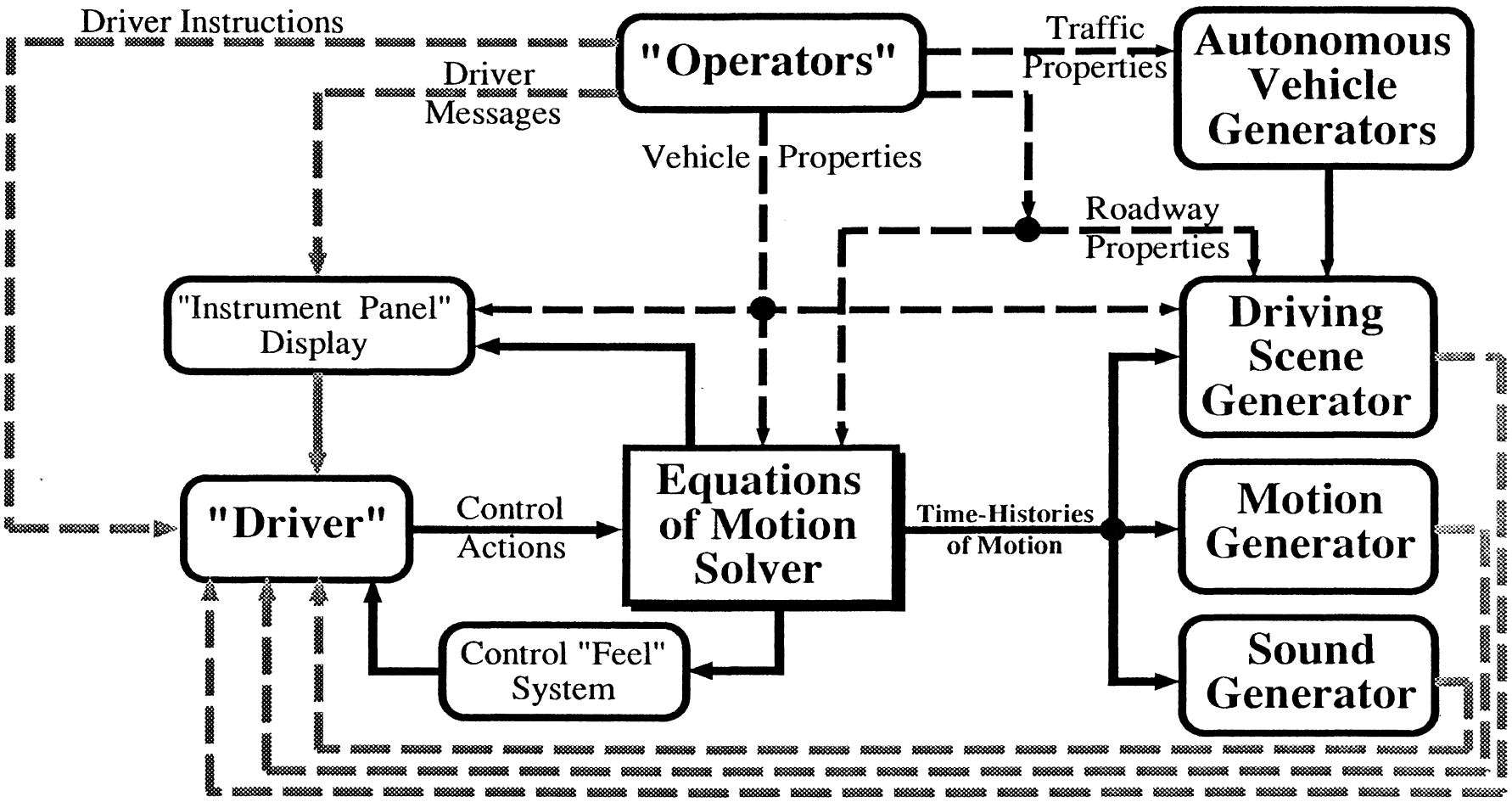


Figure 1: Driving Simulator System

"AUTOSIM", a new program developed at UMTRI, was employed in writing this simulation. AUTOSIM is a computer language designed to generate efficient simulation code in an automatic manner. The vehicle is defined for AUTOSIM in an input file containing a description of the vehicle and the simulation (dimensions, axes systems, parameters, degrees of freedom, constraints, input information to come, required outputs, units, names, etc.). In the same input file AUTOSIM is "introduced" to a set of auxiliary subroutines, with which the simulation code will interact during the simulation. When AUTOSIM is run, it automatically generates the main body of the appropriate FORTRAN simulation code. The auxiliary subroutines are written manually, since they represent additional, special purpose functions. Such functions include the special input means for the control parameters, calculation of roll and pitch effects as pseudo-degrees of freedom, etc.

Section 2 of the report describes details of the model and the simulation algorithm. Results of cases that were studied are discussed in Section 3. Section 4 presents perceptions concerning additional features that might be included in a more detailed model. Conclusions and recommendations, commenting on the results, the model, and the use of AUTOSIM, are listed in Section 5.

Load transfer equations (used in auxiliary subroutines to simulate roll and pitch) and the tire model are listed in Appendix A. The method for calculating the demanded braking and accelerating forces is listed in Appendix B. An echo file, as generated by the simulation program for summarizing the input and simulation data, is listed in Appendix C. The next two Appendices (D and E) list the Fortran code of the simulation. The AUTOSIM generated code is in Appendix D, and the manually written auxiliary subroutines are in Appendix E.

2.0 COMPUTER MODELING

2.1 Model Description

The model that is the subject for this simulation is illustrated in Figures 2 and 3. Three axis systems, of which two are shown (the third is an inertial system), are used to describe the vehicle motion. The first system is attached to the tractor and "moves" with it relative to the inertial system. The axes of the tractor's system are designated as [TRK1] through [TRK3]. Within that system the tractor can translate along [TRK1] and [TRK2] directions, while rotation is allowed only around [TRK3]. The second axis system is attached to the trailer above the fifth wheel coupling, and "moves" with the trailer relative to the tractor. The only degree of freedom of the trailer in this axis system is rotational around [TRL3], relative to the tractor.

A "lumping" approach was employed; that is, each tandem-type suspension is lumped into a single axle, and each axle is considered to be supported by two tires. For the front axle with single tires, for instance, there is no difference between the real and the lumped representation. On the other hand, the tractor's rear suspension which is a tandem with dual tires (a total of 8 tires), is represented as a single axle with two tires. Each of those tires has stiffness and load carrying characteristics that are equivalent to four single tires. Therefore, the particular tractor semi-trailer configuration of this simulation, is supported by six "tires" as shown in Figures 2 and 3.

Conceivably, the model might be considered as three dimensional. Still, in spite of the fact that vertical dimensions are incorporated, it is still a planar model. As described above, the truck has no rotational degrees of freedom around either [TRK1] (roll) or [TRK2] (pitch). The only moment of inertia used by the model in the simulation is for yaw. No roll or pitch inertias are involved, neither are any dynamic equations in those directions (such as those for roll acceleration). Those motions are treated in a quasi-static manner. The lateral and longitudinal accelerations resulting from the planar motion (yaw) are employed "statically" to compute load transfers in the suspensions (which are represented by lumped roll and pitch stiffness values).

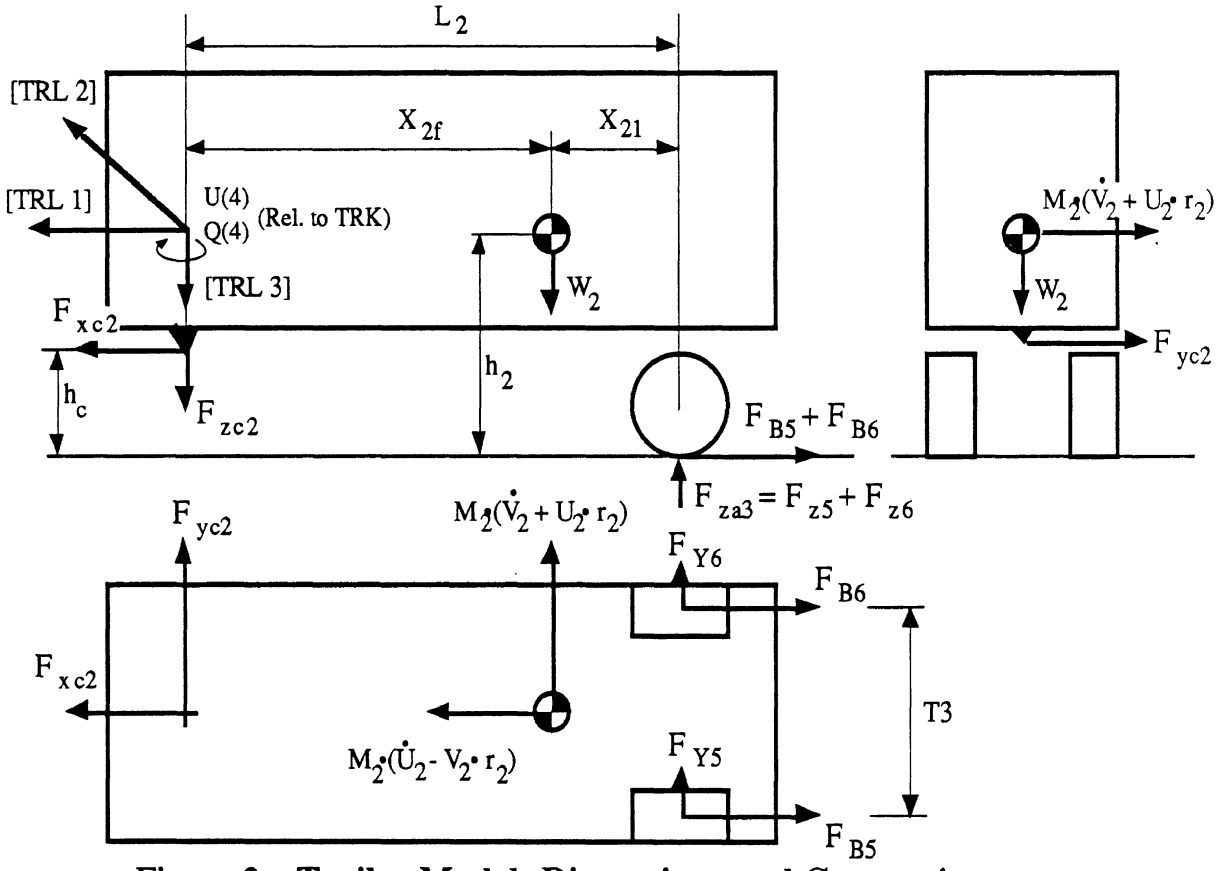


Figure 2: Trailer Model: Dimensions and Conventions

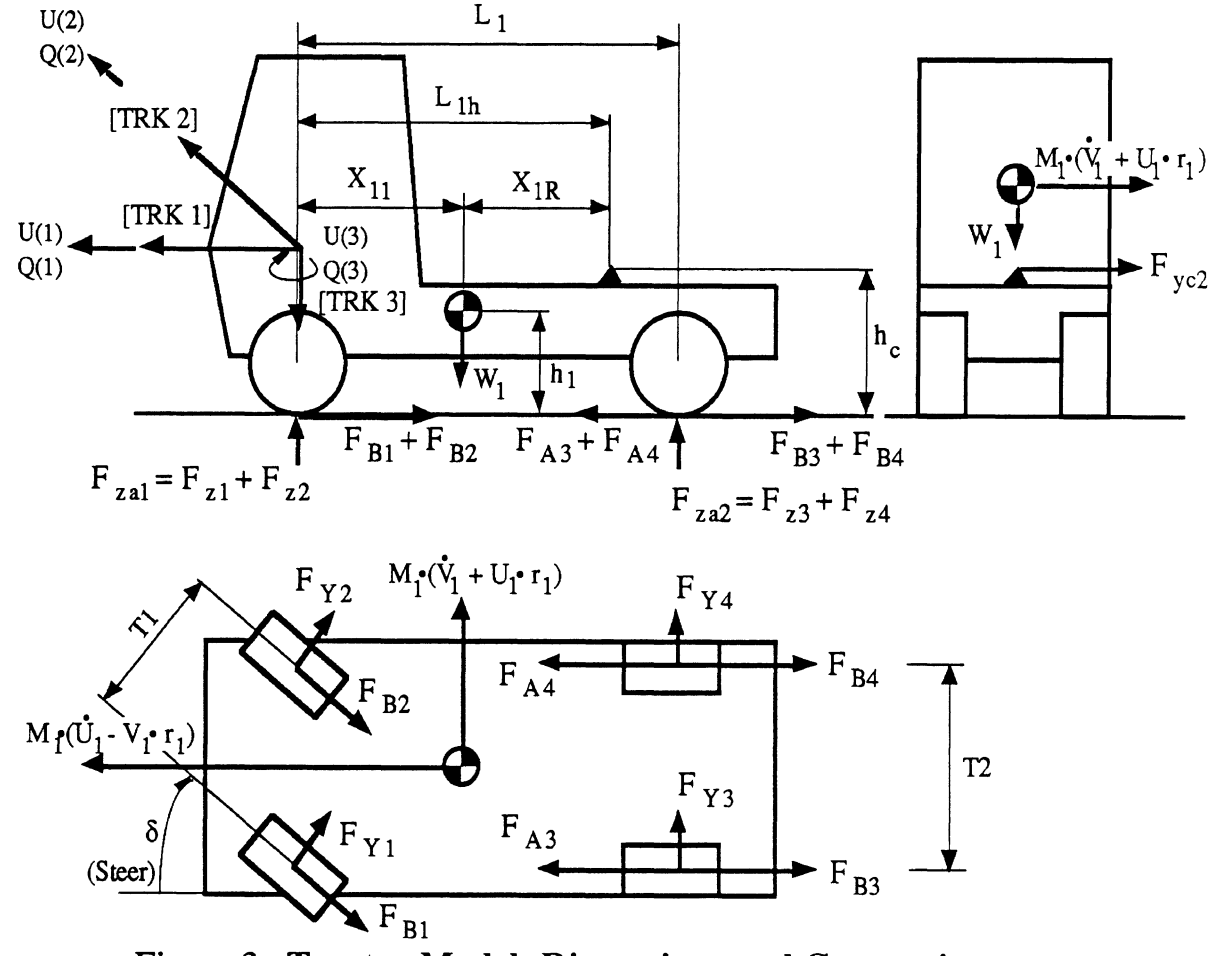


Figure 3: Tractor Model: Dimensions and Conventions

The quasi-static approach might cause computational problems under certain operating conditions unless compensating features are added. In addition, practical considerations that will simulate "real-life" applications need to be taken into account. To prevent computational "blow-outs," and to comply with reality, conditional relationships are used to modify the program as required (and at the same time send an appropriate message to the user of the simulation model). These situations are as follows:

1. Slip angle inputs to the tire shear forces subroutine are set to zero until the vehicle has reached a speed of about 2 mph. Since the slip angle is computed with the velocity in the denominator, very low speeds will generate tremendous slip angles, and subsequently very large lateral forces. That fact is especially crucial during maneuvers that involve steering at low speed, braking to a stop, and starting off. After coming to a stop, it might be that not all the tires are aligned with the forward direction. As the vehicle starts moving again, instantaneous lateral forces and acceleration may be generated that might even "cause" the vehicle to roll-over. In "real life" this phenomenon cannot happen because of the manner in which tire forces are generated. (In a future version of the simulation we would plan to employ a tire model that incorporates relaxation length in the process of the slip angle buildup. This would enable a gradual smooth growth, rather than an instantaneous growth, of lateral forces.)

2. Due to a sensitivity in the numerical method in the subroutine that performs the numerical integration (DIFEQN), when the decrease in forward velocity due to braking is too low, the velocity does not converge to zero. To eliminate the problem, a cutoff level was set at 0.34 mph (6 in./sec.). When braking is applied — once both the forward and lateral velocities get below that level — the program considers the vehicle stationary, and the appropriate variables are set to zero.

3. A jackknife situation starts to develop as the rear suspension of the tractor skids laterally due to braking, so that the semitrailer "folds" towards the tractor, until eventually it hits the cab. For all practical purposes, when the trailer is more than 45 deg. off the tractor centerline, such a situation is not recoverable and jackknife is inevitable. In the simulation, under those conditions (more than 45°, and brakes are applied), the program terminates with a jackknife message being sent to the screen.

4. Extremely tight turns are not allowed. Even without braking, when the angle between the tractor and the trailer reaches 90 deg. the program terminates with an appropriate message being sent to the screen. See Section 4 for further discussion.

5. The rollover process starts when the trailer inner wheels lift off the ground. At this point, the situation is still considered recoverable. Once the inner wheels of the tractor rear suspension have lifted off, a complete rollover will take place. When the above prevails, the program terminates and a rollover message is sent to the screen.

6. Longitudinal acceleration is computed based on braking forces. It is used to compute longitudinal load transfer, and if the vehicle is not perfectly straight it also has a component that contributes to a lateral load transfer. When the vehicle is stationary and the brakes are applied, the model would compute longitudinal deceleration and subsequent load transfers — a situation that obviously cannot be real. To avoid this, no longitudinal decelerations are computed below 35 in./sec. (2 mph).

7. The vehicle cannot be "driven" in reverse. The slip angle is defined as the angle measured from the direction the wheel is pointing to the wheel's direction of motion. In view of the above fact, when the vehicle is moving backwards, the resultant slip angle will be 180°, which will cause instantaneous generation of tremendous lateral forces. Therefore the velocity is automatically set to zero if somehow it gets a negative value.

Some of the above points are essential and represent real situations such as rollover — if it happens, termination is inevitable. On the other hand, some represent drawbacks of the simulation, such as the reverse motion incapability or other cutoffs. Improved and enhanced models can take care of those points. Comments about these issues are presented in Sections 4.0 and 5.0.

2.2 Computational Flow

The final simulation code can be considered to be composed of two main segments. One is the AUTOSIM generated code, and the second is the manually written, special purpose auxiliary subroutines.

According to the description of mechanical rigid bodies, (that is their degrees of freedom and constraints as introduced to AUTOSIM in an input file), AUTOSIM generates a comprehensive simulation code that incorporates all the dynamic effects involved. This code includes, in addition to the main program, subroutines that read input parameters, initialize variables, perform numerical integration, and store the outputs in an ERD file, so that they can be graphed later. During any of those steps, if required, interaction with auxiliary subroutines can take place. Figure 4 portrays the form of the program:

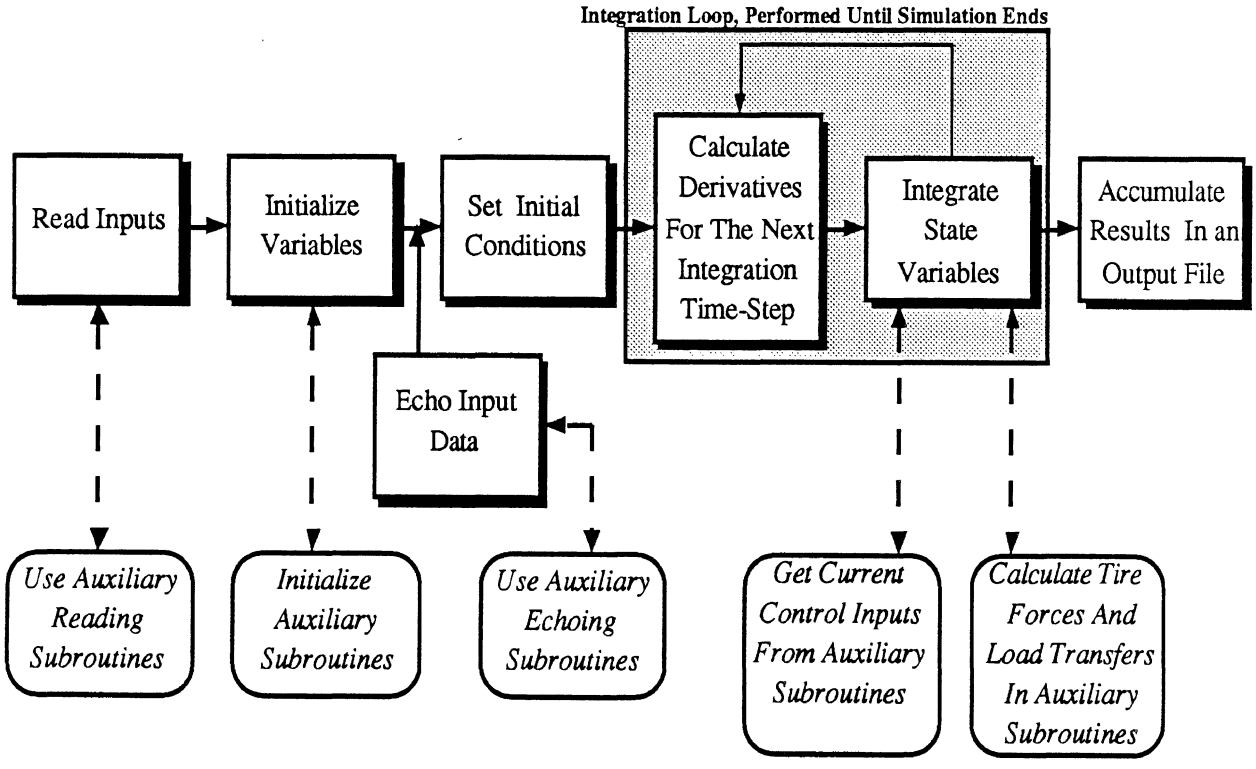


Figure 4: AUTOSIM generated code, diagram

One function of the auxiliary subroutines is to serve as an interface to read and echo the input data in a formatted way. Such an echoed data set is listed in Appendix B. Another task is to read and interpolate during each time step the control inputs for the simulated model. These two duties are performed by auxiliary subroutines so that the simulation will have a flexible, easy to modify input interface. One of the advantages of such an interface is embodied in the fact that when it is incorporated into a driving simulator, it will be able to pick up control inputs from digitized hardware control devices, instead of from input files.

For the analyst, the more crucial task of the auxiliary subroutines is to make the simulation program versatile. The tire forces and load transfers shown in Figure 4, are calculated in auxiliary subroutines, and therefore serve as modular components of the simulator. Almost any model representing the tire can be written as a separate subroutine, and can be integrated into the program. The simulation presented in this report incorporates the tire model and load transfer scheme presented in Appendix A.

3.0 SELECTED STUDY MATRIX

To validate qualitatively the simulation program, and to perceive whether it produces output results that are reasonable, a matrix of simulation runs was conducted. The various runs incorporated different combinations of steering, braking and accelerating.

From the various simulation runs that were performed, the results of the following selected study matrix are presented herein:

- 30 mph, 2 deg. step steer
- 40 mph, 2 deg. step steer
- 38.2 mph, 2 deg. step steer
- Low friction ($\mu = 0.35$), 35 mph, 3 deg. step steer
- Tractor rear tires with increased stiffness (B/2), 38.2 mph, 2 deg. step steer
- Severe steer and brake maneuver
- A maneuver resulting in a jackknife

In the first case a moderate maneuver was studied, and its results are shown in Figures 5 through 12. The vehicle was simulated to be moving in a straight line at a speed of 30 mph, and then at 1.0 sec., a step steer of 2 deg in the front wheels was introduced.

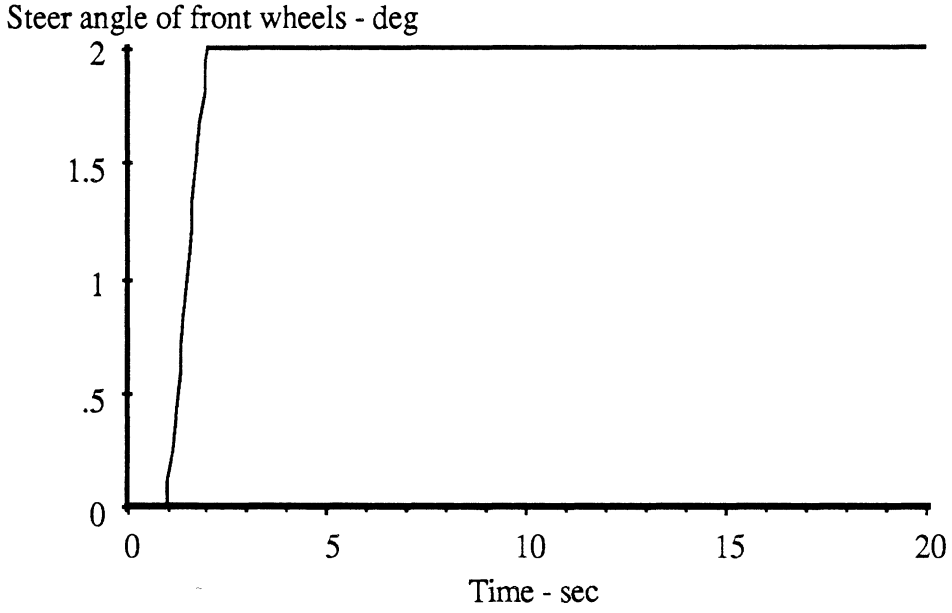


Figure 5: Step steer input profile

As a result of the negative longitudinal component (along [TRK1]) of the lateral force generated by the steered front wheels, the vehicle slowed down (Figure 6). Because of that

slowing, the variables, plotted in Figures 7 through 12, show a tendency to reduce in magnitude instead of maintaining a steady state level.

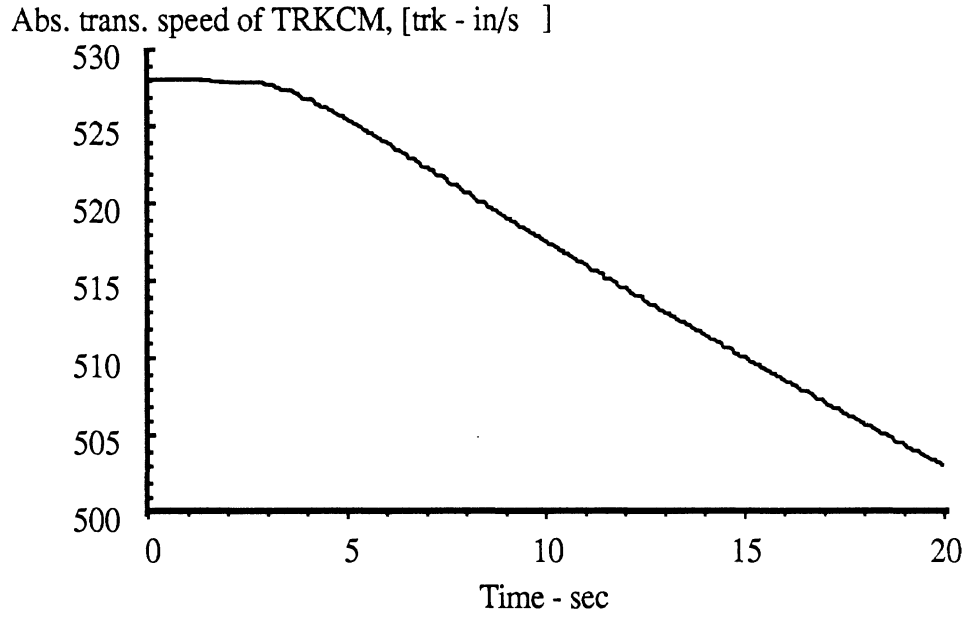


Figure 6: Head-on velocity of vehicle

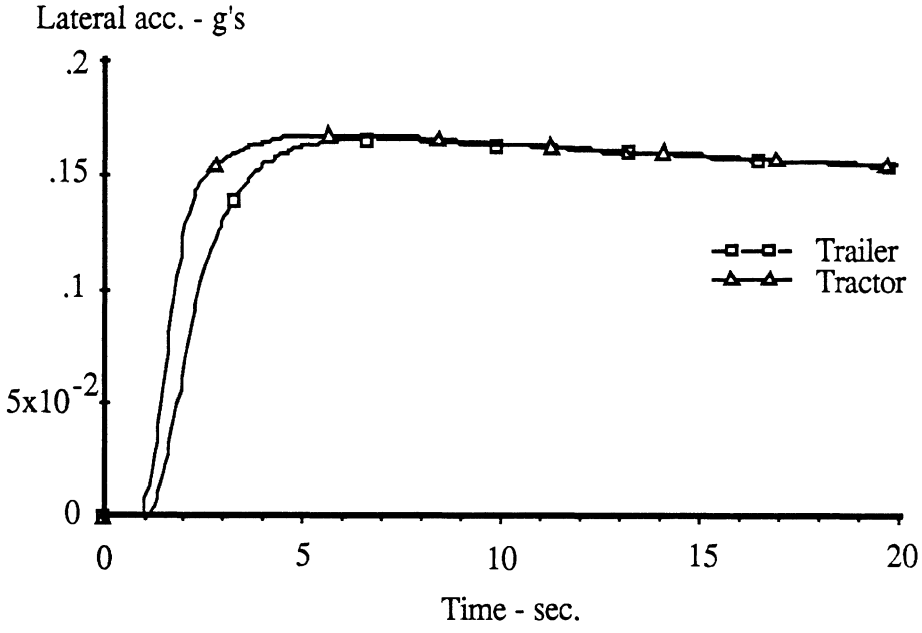


Figure 7: Lateral accelerations of tractor and trailer

As the truck negotiates the turn, Figure 7 clearly shows the lag in lateral acceleration between the trailer and the tractor. Due to the above mentioned decrease in speed, the lateral accelerations decrease, too, and do not maintain a steady state value.

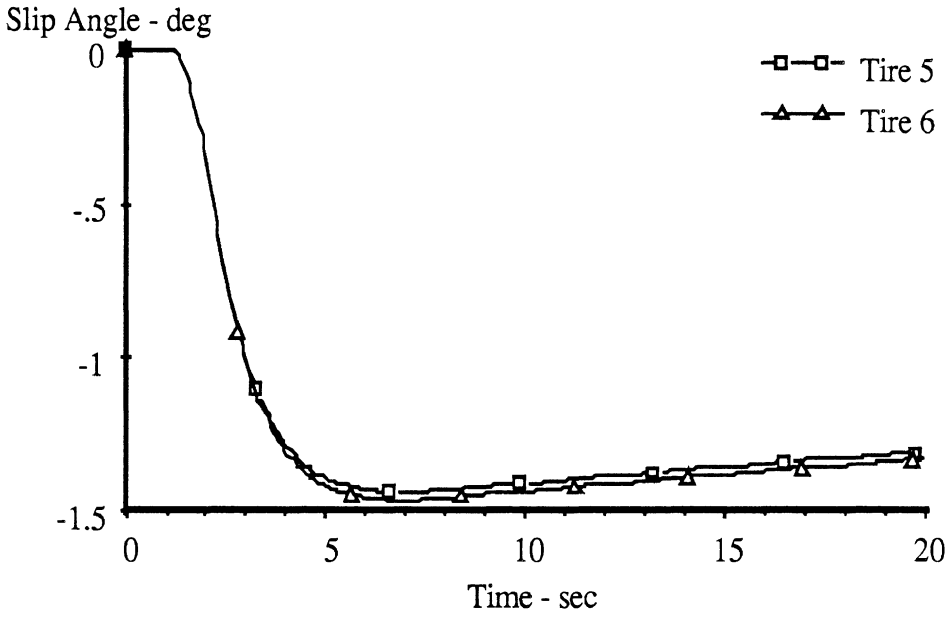


Figure 8: Slip angles at the trailer wheels

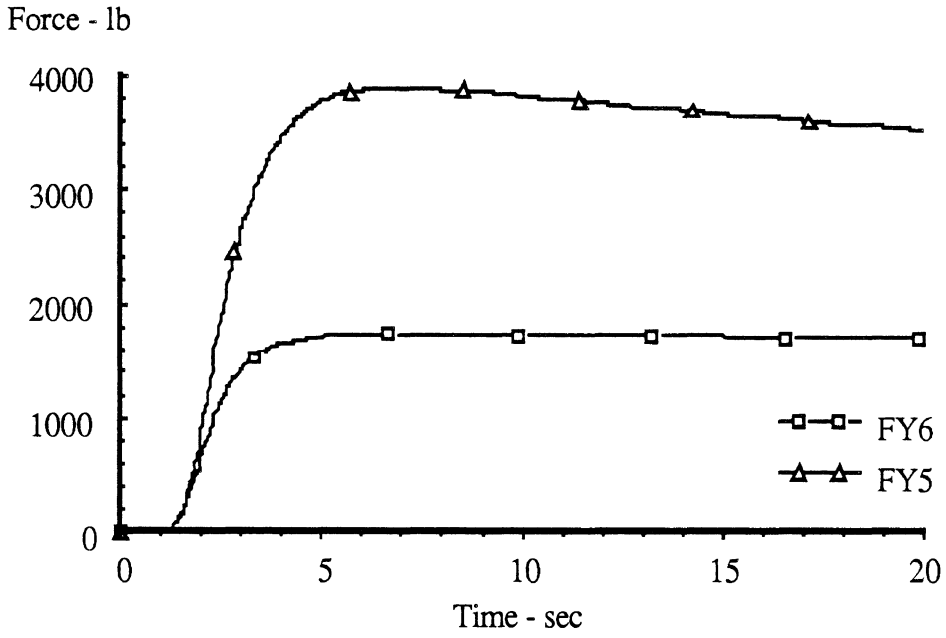


Figure 9: Lateral forces at the trailer wheels

The slip angles are almost identical for both tires of the trailer (Figure 8). They are not truely identical, though, due to the trailer's yaw. The load sensitivity of the tire model (see Appendix A), causes very different lateral forces to be developed at the right and left hand side tires (Figure 9) in spite of the nearly common slip angle.

The following Figures 10 and 11 show the vertical load transfer between the wheels on the right and left hand sides on the trailer and the tractor's rear axles. (As mentioned before, these tire loads do not maintain steady state values due to the slowing down of the vehicle.)

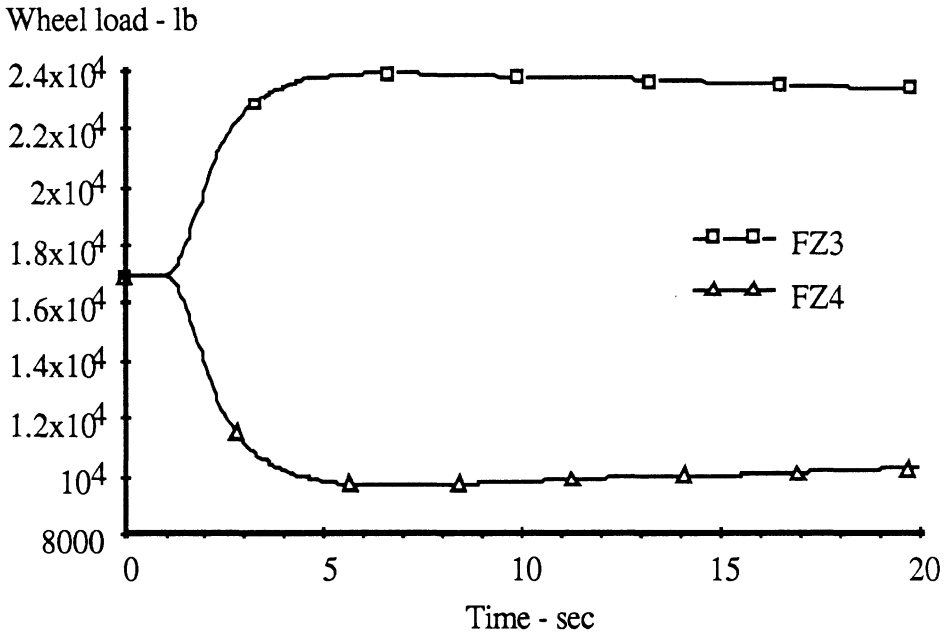


Figure 10: Lateral transfer of vertical wheel loads, rear axle, tractor

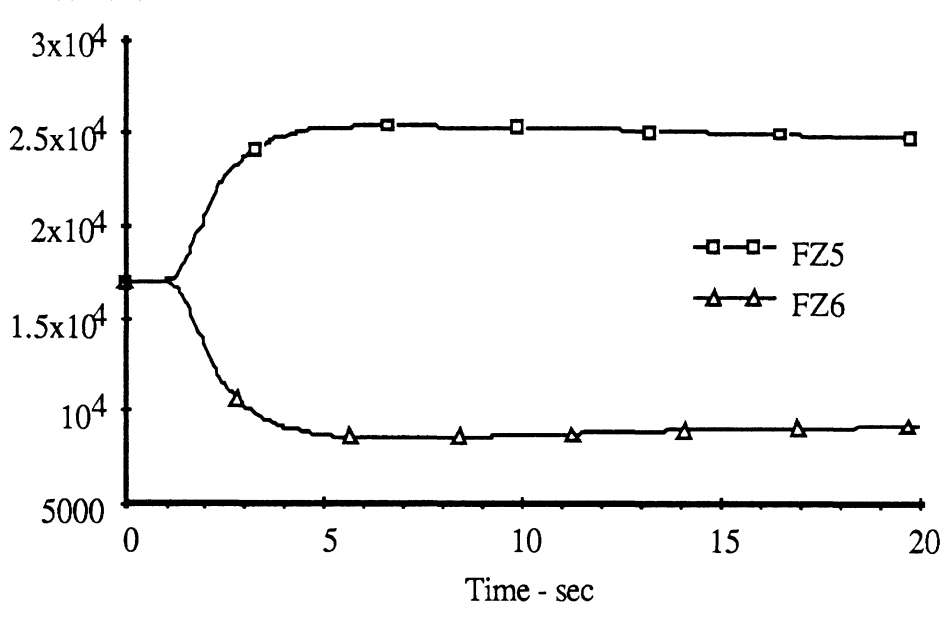


Figure 11: Lateral transfer of vertical wheel loads, trailer axle

Figure 12 indicates that the articulation angle between the tractor and trailer is initially established at a high transient rate, and then the rotational rate of the trailer with respect to the tractor is small.

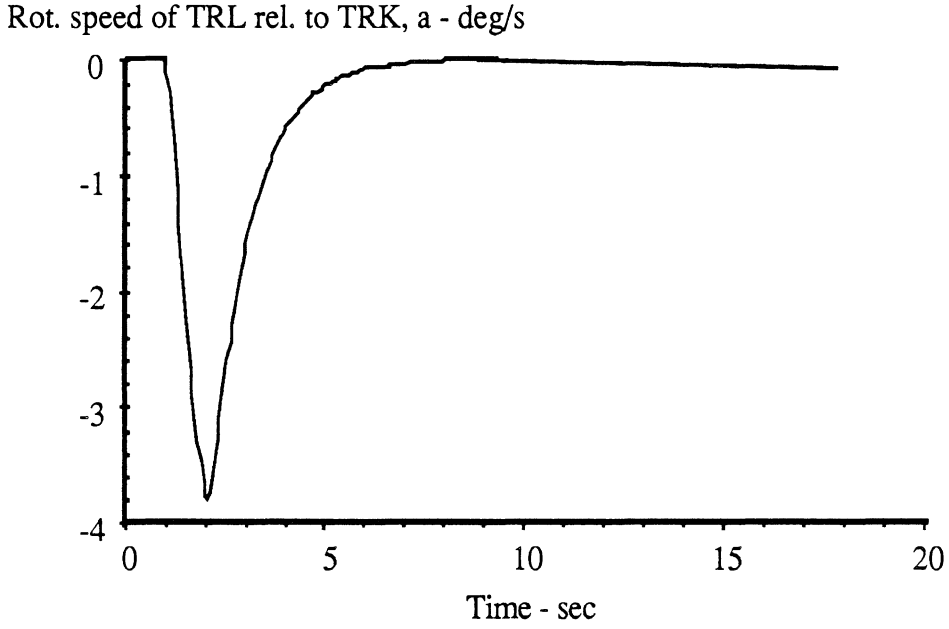


Figure 12: Articulation rate between trailer and tractor

The next run was identical to the previous one except that the speed was increased to 40 mph. The vehicle rolled over as a result of the high lateral accelerations that developed.

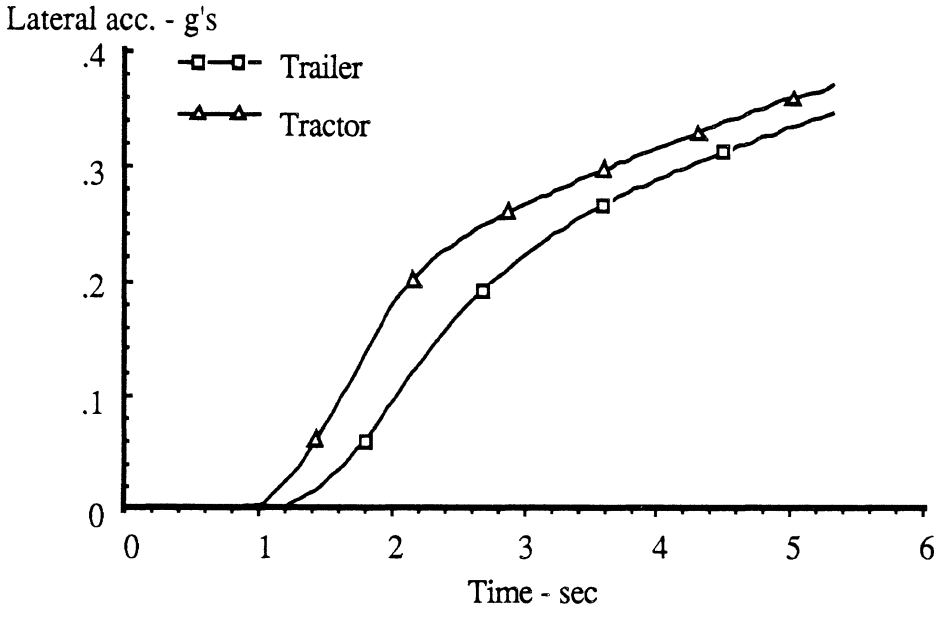


Figure 13: Lateral accelerations in tractor and trailer, 40 mph

Now consider the step steer maneuver at 40 mph. The results in Figure 13 indicate that this vehicle reached a lateral acceleration of 0.36g before it rolled over. Unlike the situation at 30 mph where the articulation rate drops to zero during the maneuver (see Figure 12), at 40 mph the rate of change in articulation angle slows down, and then suddenly increases as the vehicle starts to roll over (see Figure 14).

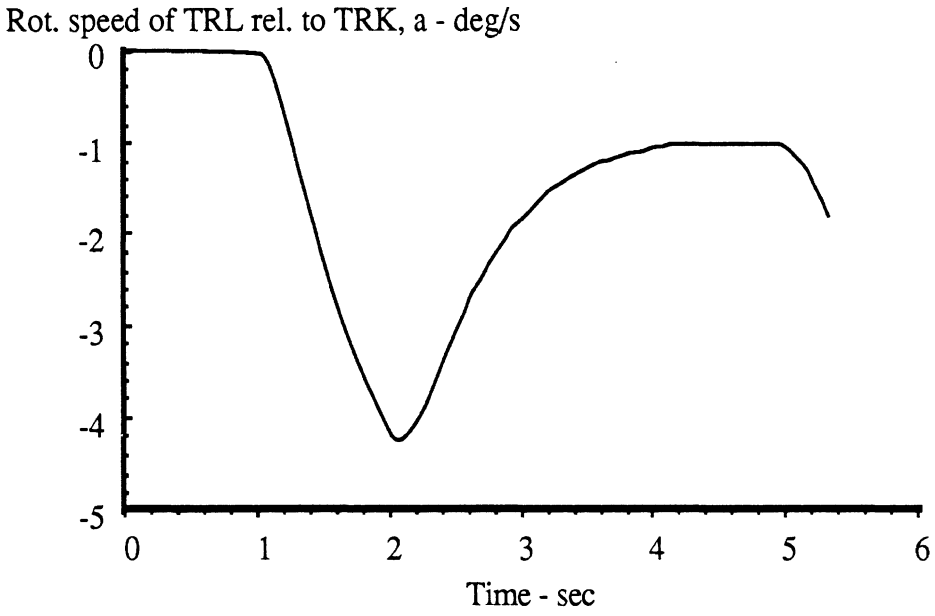


Figure 14: Articulation rate between trailer and tractor, 40 mph

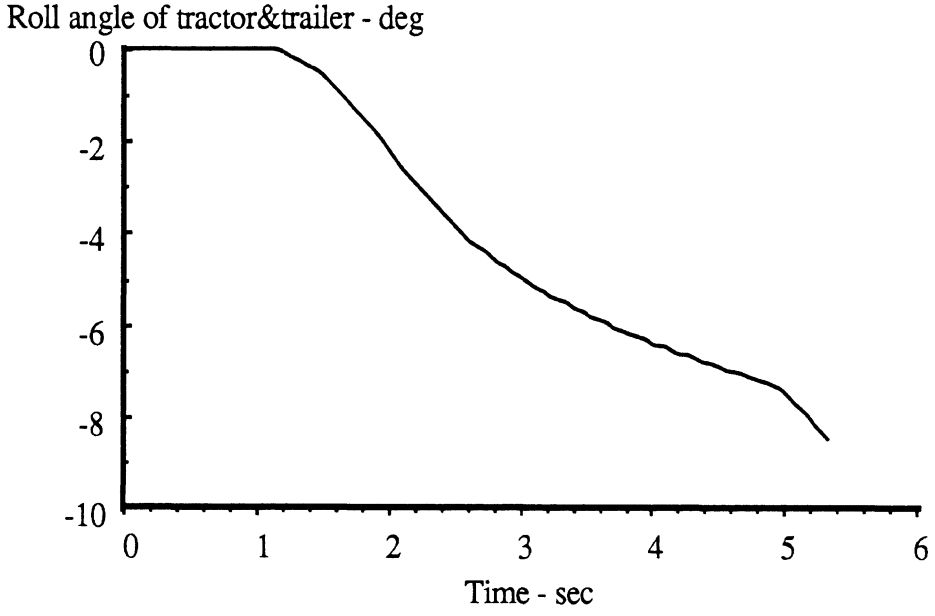


Figure 15: Roll angle, 40 mph

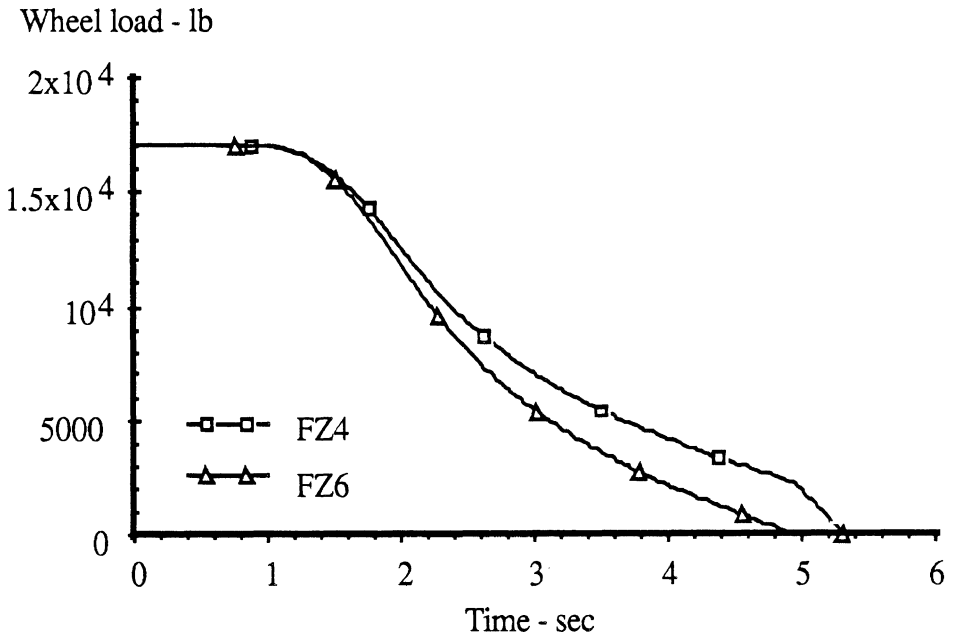


Figure 16: Vertical loads on the inner wheels (FZ6 and FZ4) of the trailer and tractor rear axles, respectively

The rollover process can be seen in Figures 15 and 16. The roll angle increases as the vehicle goes into the turn, (at a higher rate at first due to the higher speed). At the same time the vertical load on the inner wheels decreases (transferred to the outer wheels). At about 4.8 sec., the trailer's inner wheel lifted off the ground. Due to the limited restoring moment now available, load is taken off wheel #4 more rapidly, and then about 0.5 seconds later it lifts off, too, resulting in rollover.

At this point in the qualitative evaluation of the program, a series of simulation runs were used to find the speed that will bring the vehicle close to its rollover threshold—without rolling over. The maximum speed found for this vehicle to go through the above described maneuver without rolling over was 38.2 mph. Figures 17 through 22 show the simulated vehicle response.

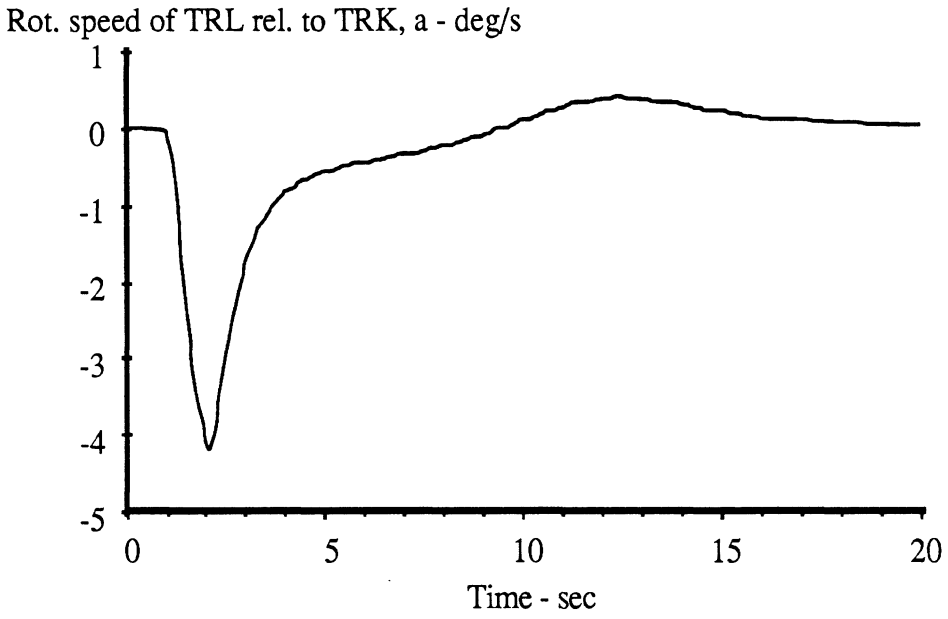


Figure 17: Articulation rate between trailer and tractor

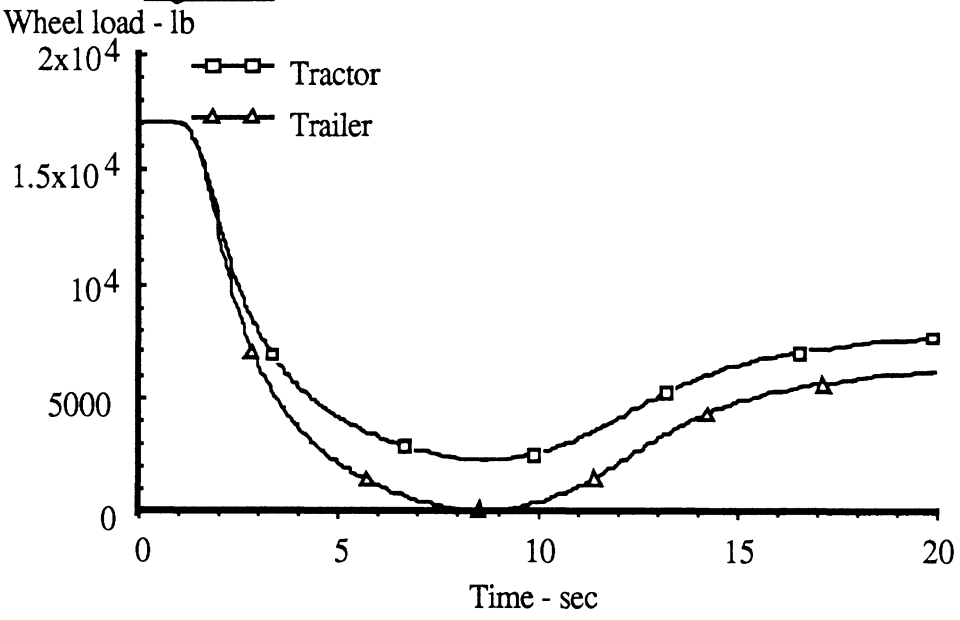


Figure 18: Vertical load on inner wheels

As seen in Figure 18, the trailer inner wheels had lifted off at about 7.5 seconds, and stayed lifted for about 1.0 second. This was the most critical time during this maneuver. Due to the slowing down and the marginal effect that losing some of the restoring moment had on the directional stability of the simulated vehicle, the tractor wheels did not lift, and the trailer inner wheel regained contact with the ground.

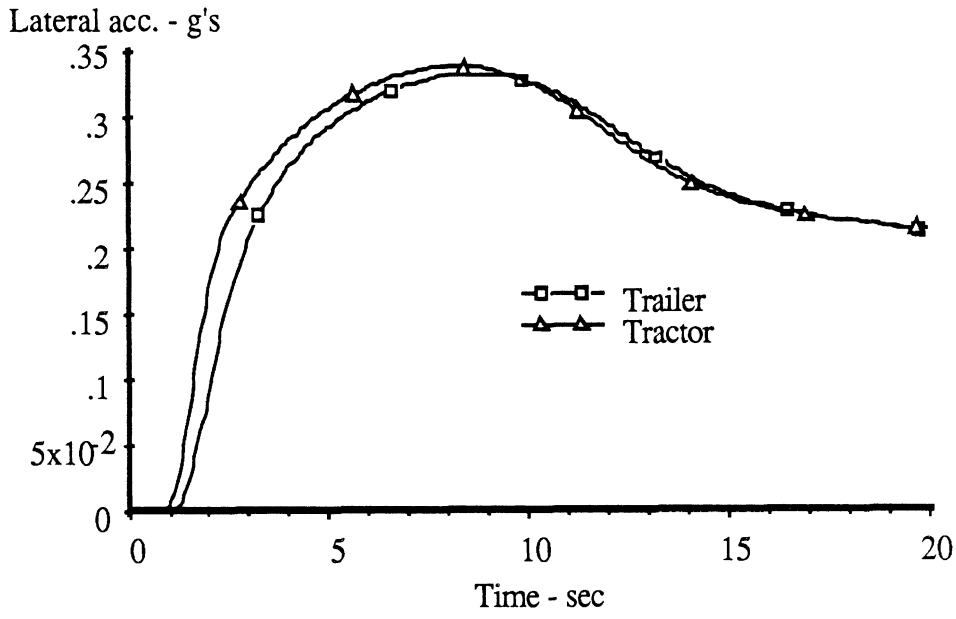


Figure 19: Lateral accelerations of tractor and trailer

Furthermore, due to the decreased ground contact of the trailer's inside wheels, the trailer develops a higher yaw rate than the tractor, and that results (as shown in Figure 17) in a small, opposite-polarity, positive-articulation rate. (See also the discussion following Figure 22.)

Since this run was performed at the stability boundaries of the subject tractor semitrailer, just before it rolls over, Figures 19 and 20 can be regarded as representing the vehicle stability characteristics. The peak value for the lateral acceleration of the trailer is about 0.33 g, which can be regarded as its rollover threshold. If the vehicle is subjected to a higher lateral acceleration it will capsize. The resultant maximum roll angle that can be imposed on the vehicle without losing control is somewhat over 7 deg.

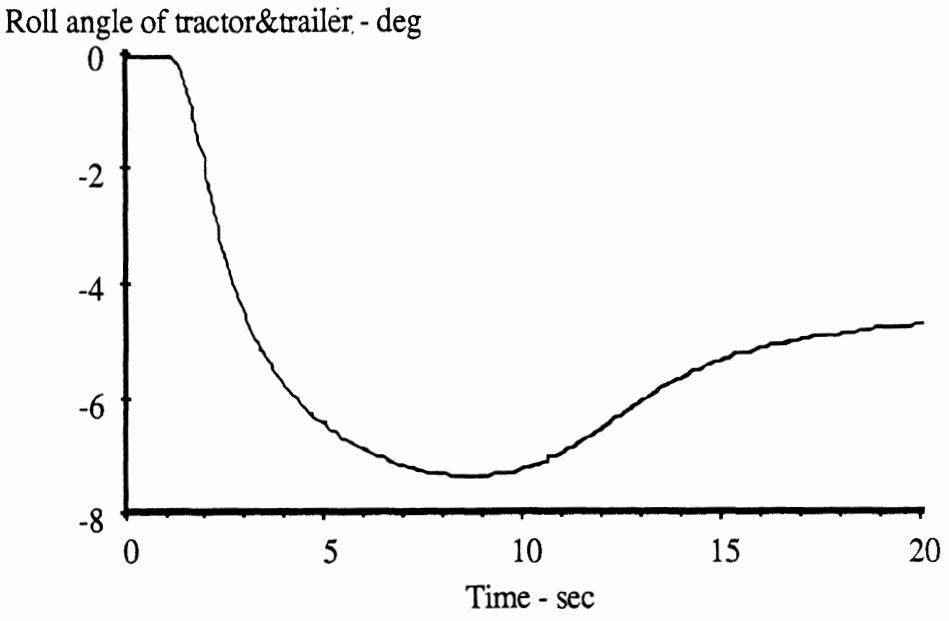


Figure 20: Roll angle, 38.2 mph

Since slip angle is defined in the simulation as the angle between the wheel direction and its vector of motion, Figure 21 does not show any evidence of the fact that wheel #6 lifted off. Figure 22 on the other hand, shows zero lateral force for the time period that the wheel is in the air. At that time, the trailer axle develops a high slip angle (-4 deg.) to support the lateral acceleration. The slip angles of the tractor front and rear axles are -2.5 and -3 deg. respectively (their plots are not shown). As a result, the yaw rate of the trailer is higher than that of the tractor, and the articulation rate (Figure 17) reverses its polarity to a positive value.

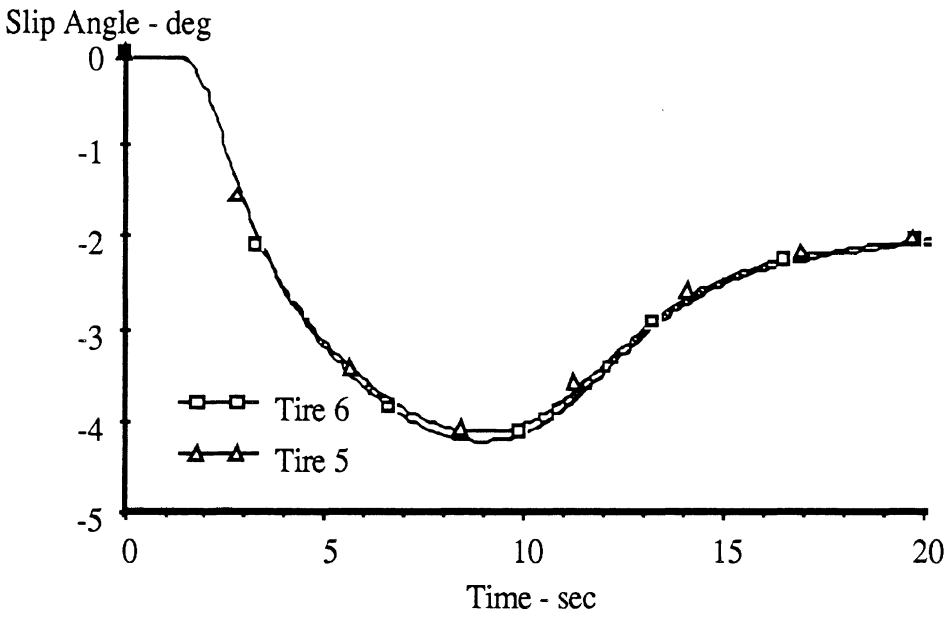


Figure 21: Slip angles, trailer tires

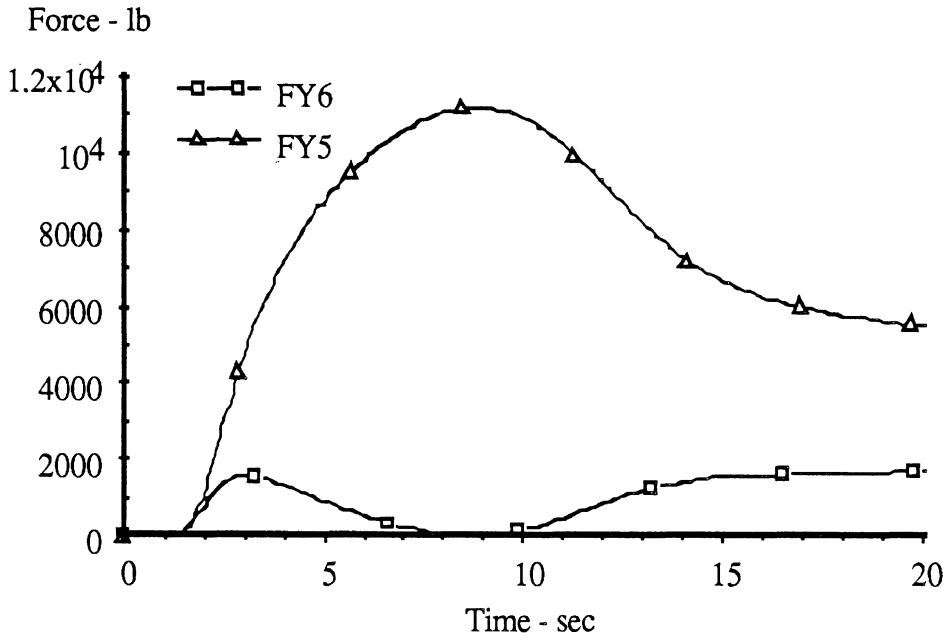


Figure 22: Lateral forces, trailer tires

For comparison purposes, the lateral accelerations obtained in the three simulation runs were plotted together (see Figure 23).

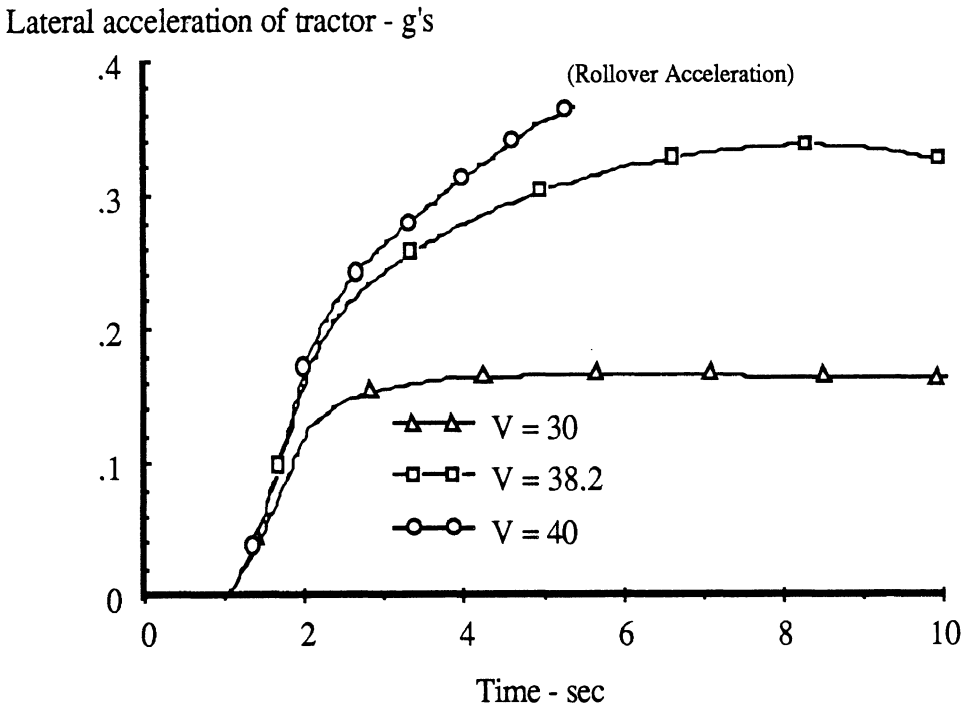


Figure 23: Lateral acceleration of three runs

As expected, the graph that relates to the 40 mph run indicates a significantly higher acceleration. This graph stops at about 5.2 seconds because that was the time it rolled over.

The next run that was studied incorporated a 3 deg. step steer instead of 2 deg., but the more significant variation was that the simulated maneuver was performed on a slippery surface—frictional coefficient of 0.35 instead of 0.8. The result of such a maneuver under these conditions was a lateral skid of the tractor leading to a jackknife, even though no braking was applied. The simulated vehicle lost all directional stability; but even though it was close to losing roll stability, it did not roll over.

Due to the reduced adhesion, the forces that could be generated by the tire in the contact patch were much smaller than those generated by the tires in the earlier runs on the higher friction surface. The consequences and the ensuing skid are demonstrated in Figures 24 and 25. The rate of change of articulation angle (Figure 24) behaves at first in a regular manner—increasing rate of change, and then a decrease. But, unlike Figure 12, where the rate stabilizes itself at zero, here it is increasing rapidly (at about time = 5.5 sec.), and gets out of control. By examining Figure 25, which shows the value of the articulation angle, the increased rate of change is also recognizable. That plot stops at an angle of 90 deg., which is a predetermined value in the program (see note #4 at the end of Section 2.1).

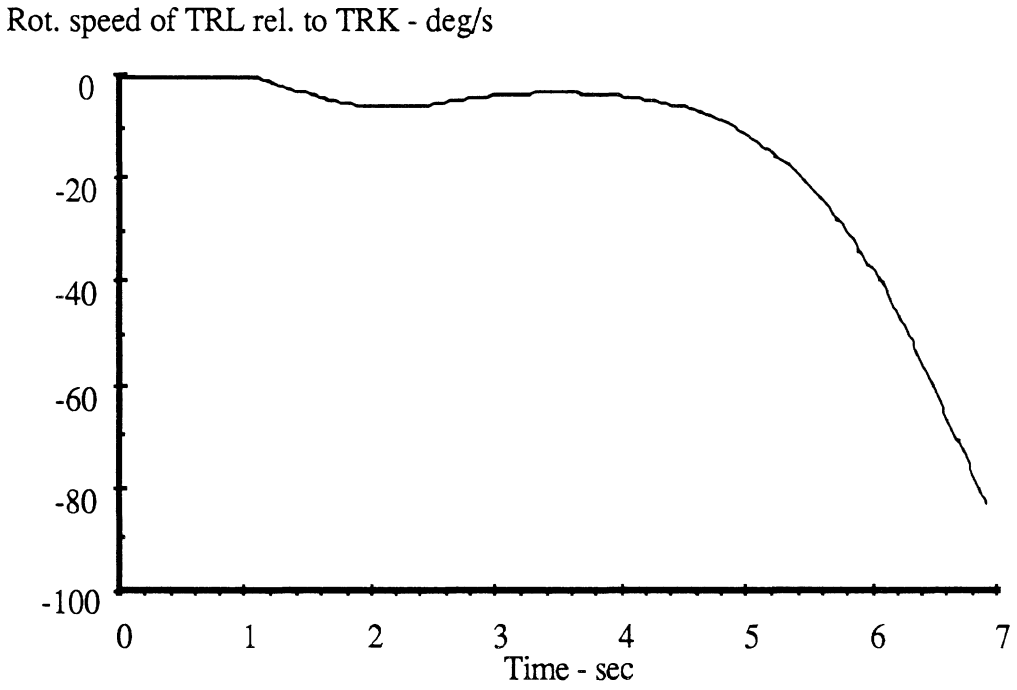


Figure 24: Articulation rate between trailer and tractor

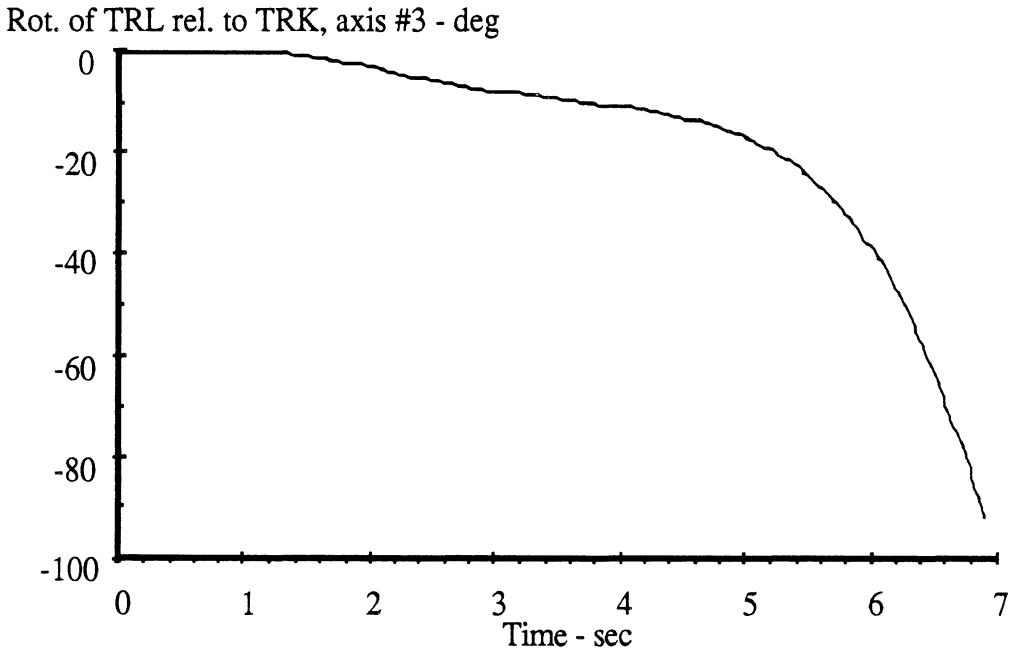


Figure 25: Articulation angle between trailer and tractor

A situation that is close to marginal roll stability is demonstrated in Figures 26 through 28. It should be noted that the simulated conditions of this maneuver brought the vehicle close to the limits of its roll performance envelope, but it was not at the actual boundaries. Clearly, with all other parameters held fixed and just increasing the frictional coefficient of the surface, we get closer to the roll stability boundary of the vehicle. Another run that was performed separately employed a surface with $\mu = 0.36$. Under these conditions the

trailer rear suspension lifted off and it almost rolled over. During the run discussed herein, the load on the inner wheel of the trailer rear axle did not diminish. Still, the wheel was quite close to being lifted off as seen in Figure 26 (FZ6). (The actual value was 180 lb., which, due to the scale, looks like zero.)

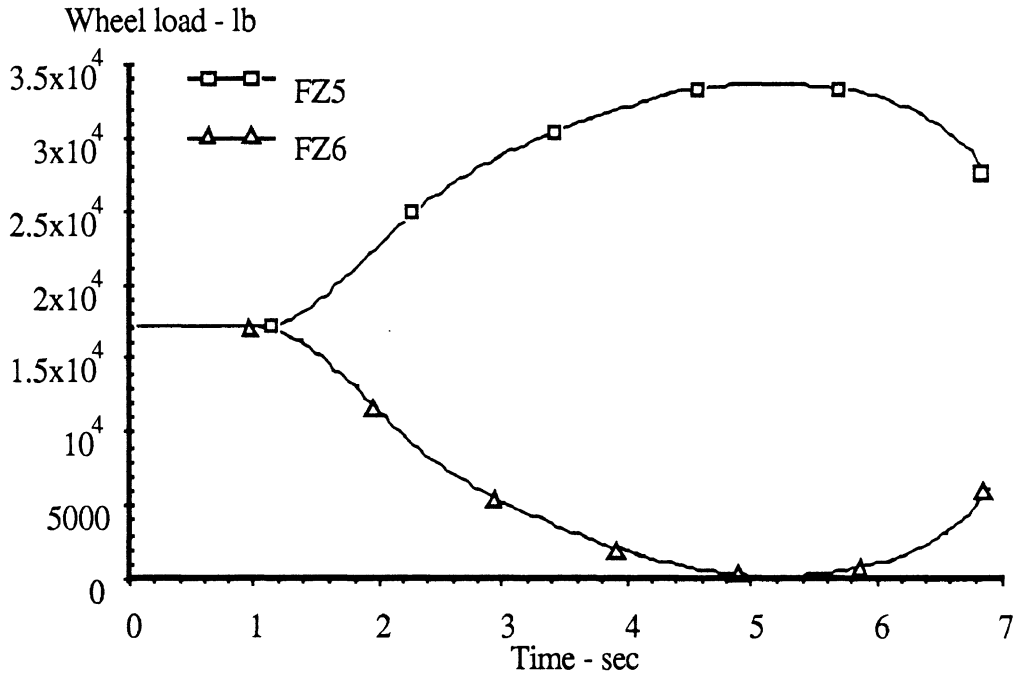


Figure 26: Lateral transfer of vertical wheel loads, trailer axle

Additional evidence, indicating the marginal conditions under which the maneuver was performed, is presented in Figure 27. The trailer reached a level of lateral acceleration of approximately 0.32 g, while it was already established that the subject vehicle has a rollover threshold of at least 0.33 g (see Figure 19). The resultant roll angle during this maneuver, shown in Figure 28, was about 7.2 deg. which is close to the limit value implied by Figure 20.

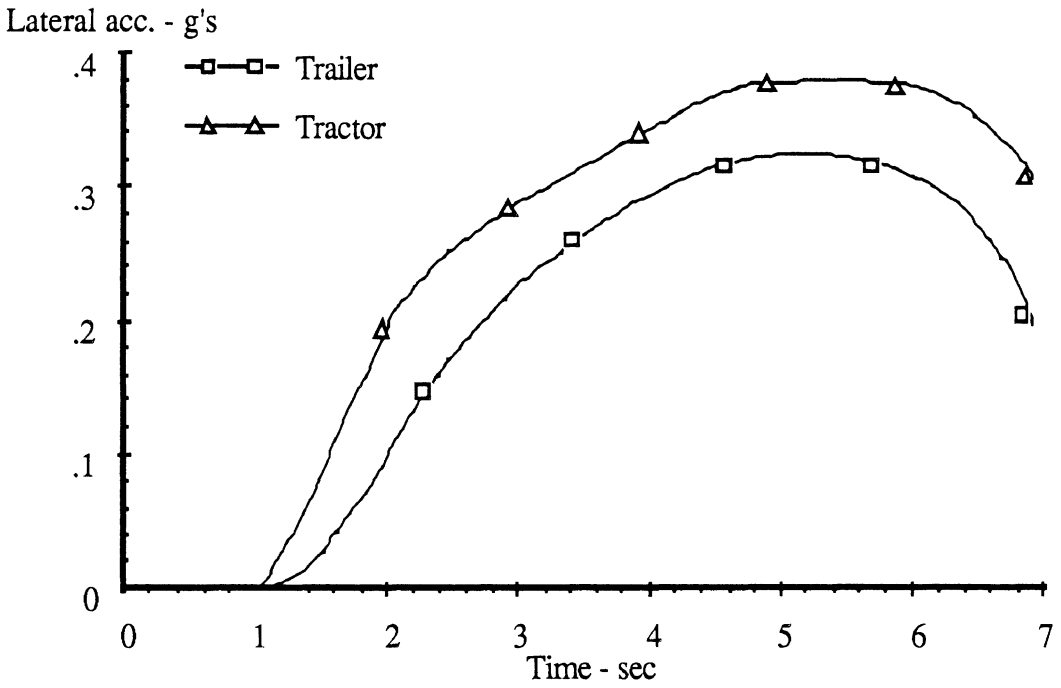


Figure 27: Lateral accelerations in tractor and trailer

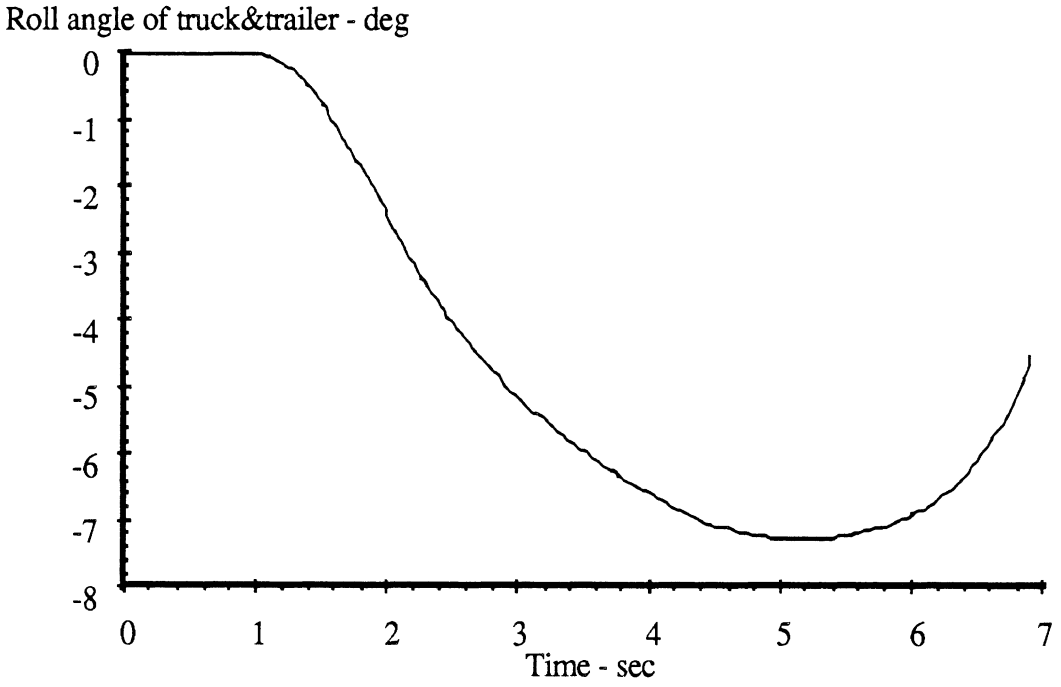


Figure 28: Roll angle

Figures 29 through 31 provide a close look at the conditions prevailing at the contact patch of the inner, rear wheel of the trailer (tire 6). In the previous discussion (e.g., Figure

24), the time of about 5.5 sec. was said to be the point where the adhesion at the tractor rear axle collapses, directional stability is lost, and the tractor starts spinning. Figure 29 clarifies that even better. After the tractor steered itself into the turn, forces through the hitch cause the buildup of a slip angle in tire 6. Due to the increasingly growing demands for lateral traction force, the slip angle increases steadily. At 5.2 sec., the rear wheels of the tractor start skidding, reducing the lateral force exerted on the trailer through the hitch. Subsequently, the growth slope of slip angle at tire 6 decreases rapidly. Its contribution to the forces generated in this situation is only due to sliding friction. After a maximum value of articulation angle of -7 deg. (at 5.6 sec.) was attained, the slip angle quickly starts to drop down.

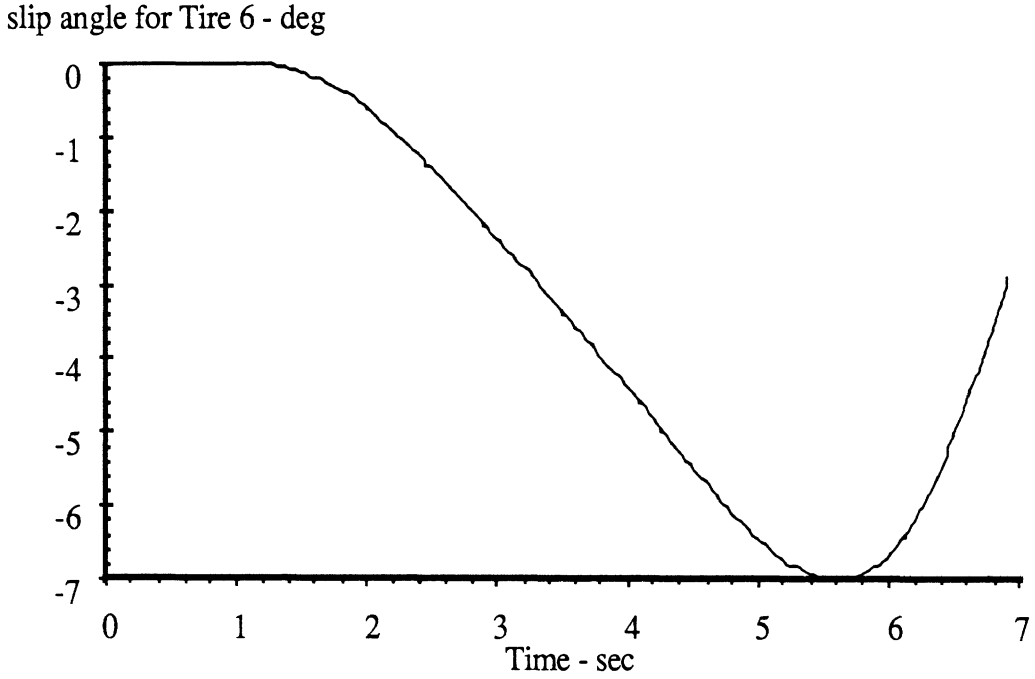


Figure 29: Slip angle at the trailer inner wheel

Since the lateral force development is a result of a non-linear dependency between slip angle and vertical wheel load, the plot of lateral force in Figure 30 should be examined as an outcome of Figures 26 and 29 combined. At first, the lateral force increases with slip angle, while the vertical force starts dropping. At about $t = 2.8$ sec., even though the slip angle is still increasing, the lateral force attains a maximum due to the decreased value of vertical load. Most of the total lateral traction force required of the trailer rear suspension for maintaining its directional stability is now provided by the outer wheel whose vertical load increases steadily (FZ5 in Figure 26). The slip angle keeps growing, but the resultant lateral force drops down. At 5.2 sec., FZ6 and the contribution of the inner wheel to

traction become minimal, and that is also the time when lateral acceleration and roll angle reach their peak (Figures 27, 28). Due to the skidding of the tractor rear wheels, lateral acceleration drops, and weight that was transferred before to the outer wheel of the trailer is now being regained by the inner wheel. As a result of the increased vertical load, lateral force can quickly build up.

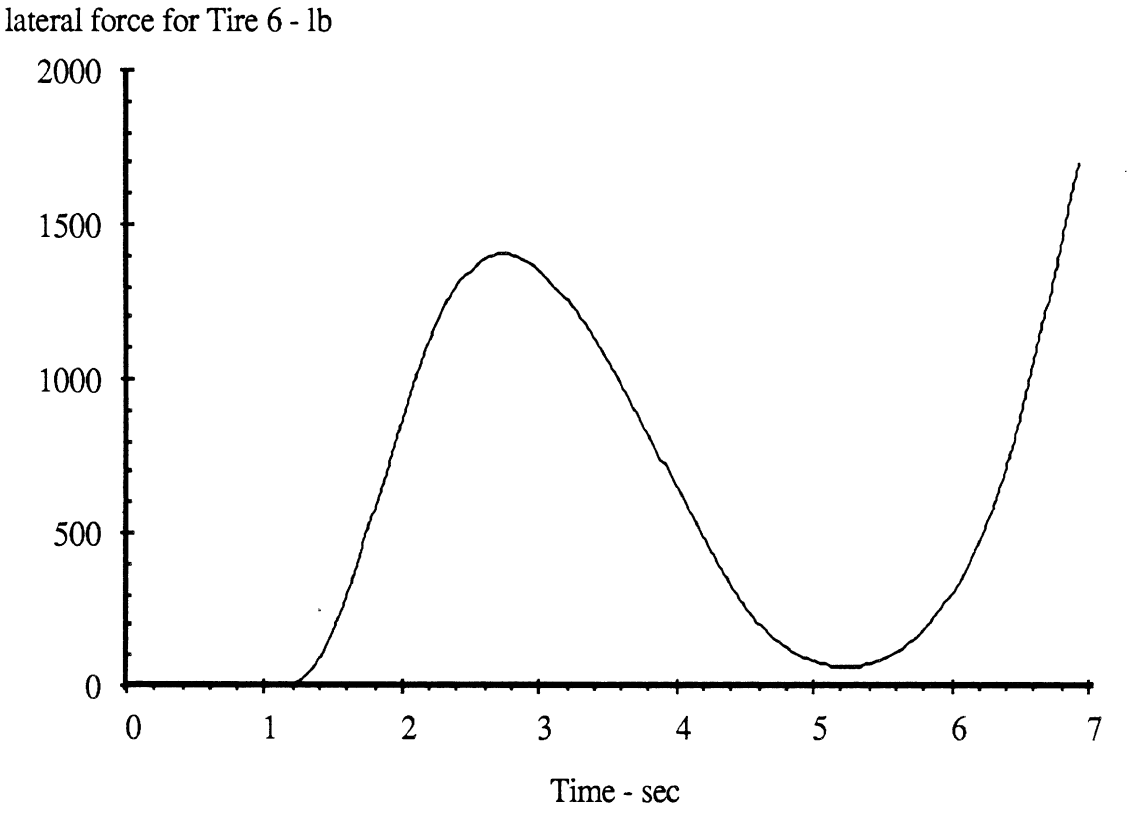


Figure 30: Lateral force at the trailer inner wheel

The same course of events is demonstrated through a different relationship in Figure 31. The lateral force attains a maximum at a slip angle of -1.8 deg., and then drops down. At the end of the slip angle scale (-7 deg.), which corresponds to the time when the tractor skids, slip angle decreases in magnitude while the lateral force builds back up due to the regained vertical load.

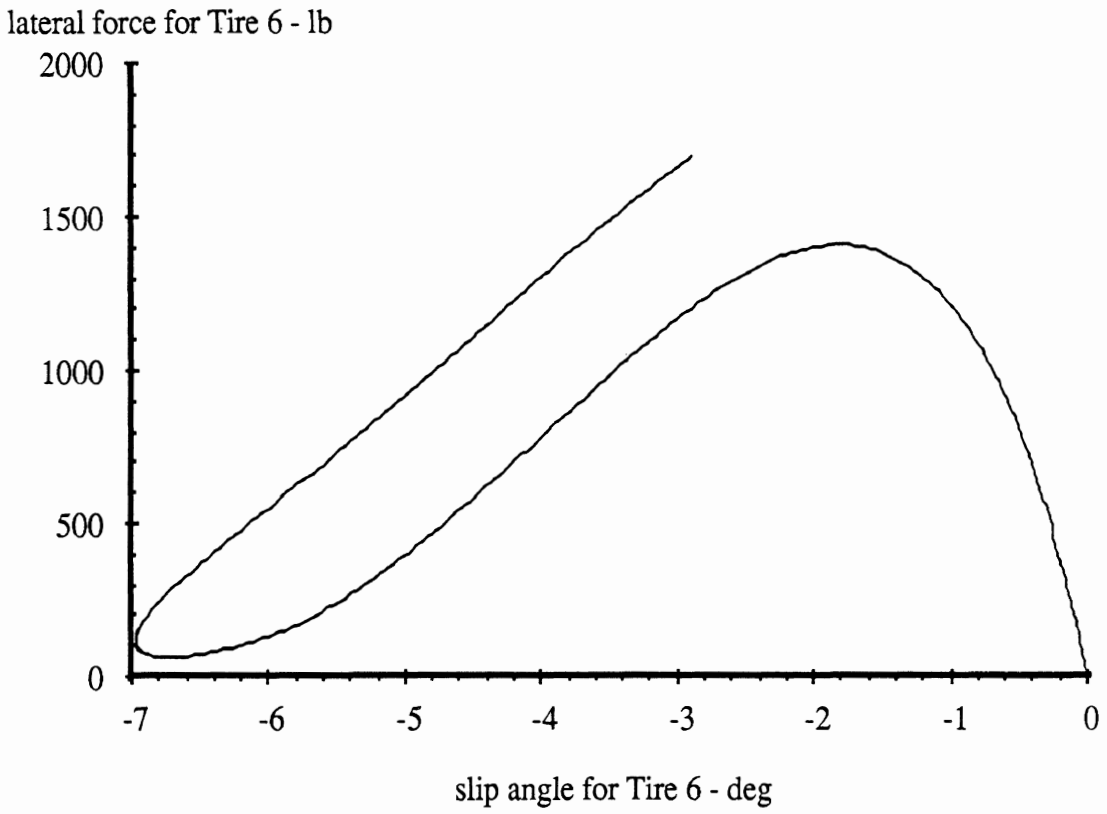


Figure 31: Lateral force — slip angle dependency at the trailer inner wheel

According to equation (1) in Appendix A, the normalized cornering stiffness of the tire in the model employed is given by:

$$\frac{C_\alpha}{F_Z} = A - B \cdot F_Z$$

The coefficient B determines the influence of the wheel vertical load on the tire stiffness. The next run attempted to study the effect of the empirical parameter B in the tire model on the behavior of the vehicle. The data set used was identical to the one employed for the third simulation in the study matrix list (38.2 mph, 2 deg. step steer), with one modification. The rear tires of the tractor were stiffened by setting the coefficient B to half of its original value. According to the above equation, by doing so the tire now is less sensitive to load variations, and the degradation in its stiffness due to increased load is less than it was before.

Figures 32 through 35 show the results of this run compared with the results without the stiff tires at the rear suspension of the tractor (a run that was discussed under

Figures 17 through 22). As a direct outcome of the stiffer tires at the rear, the slip angle they develop is now much smaller (Figure 32), the tractor gains directional stability, and consequently its yaw rate drops drastically (Figure 33).

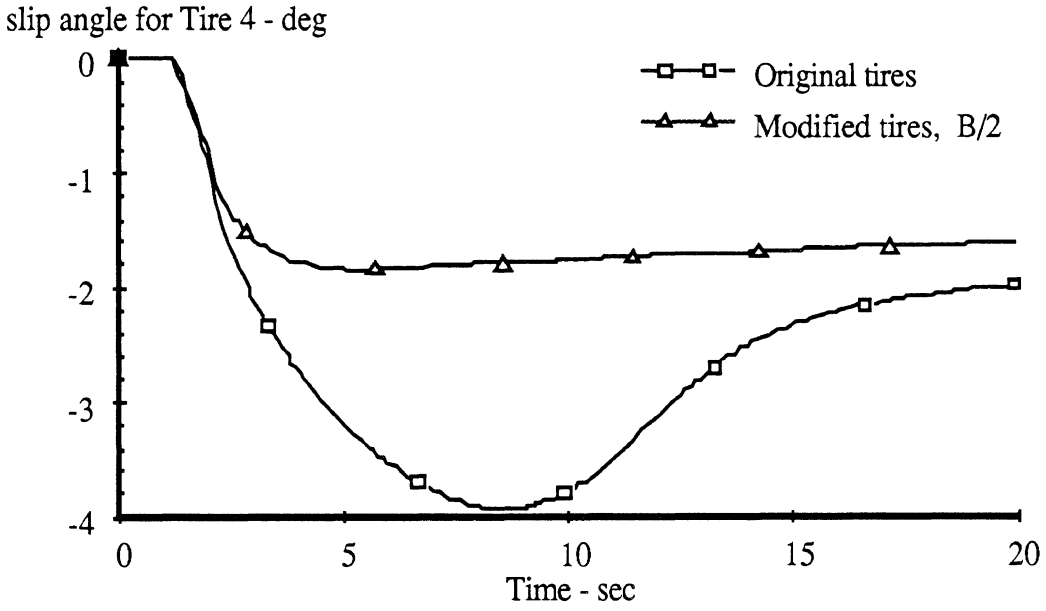


Figure 32: Slip angle comparison at the tractor inner rear wheel

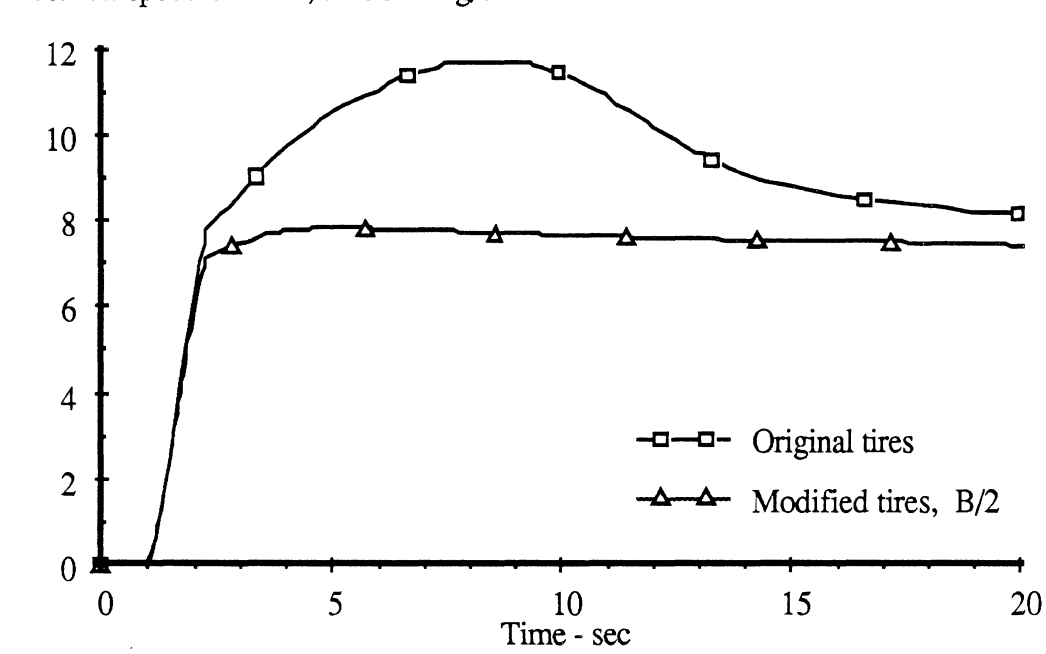


Figure 33: Yaw rate comparison of the tractor

An immediate effect that the reduced yaw rate has on the vehicle is a proportional drop in the developed lateral acceleration. Without the stiff tires, the tractor-semitrailer was on

the verge of rollover at 0.33 g. Figure 34 shows that under the modified conditions, the vehicle develops only 0.23 g, and demonstrates a much more stable behavior.

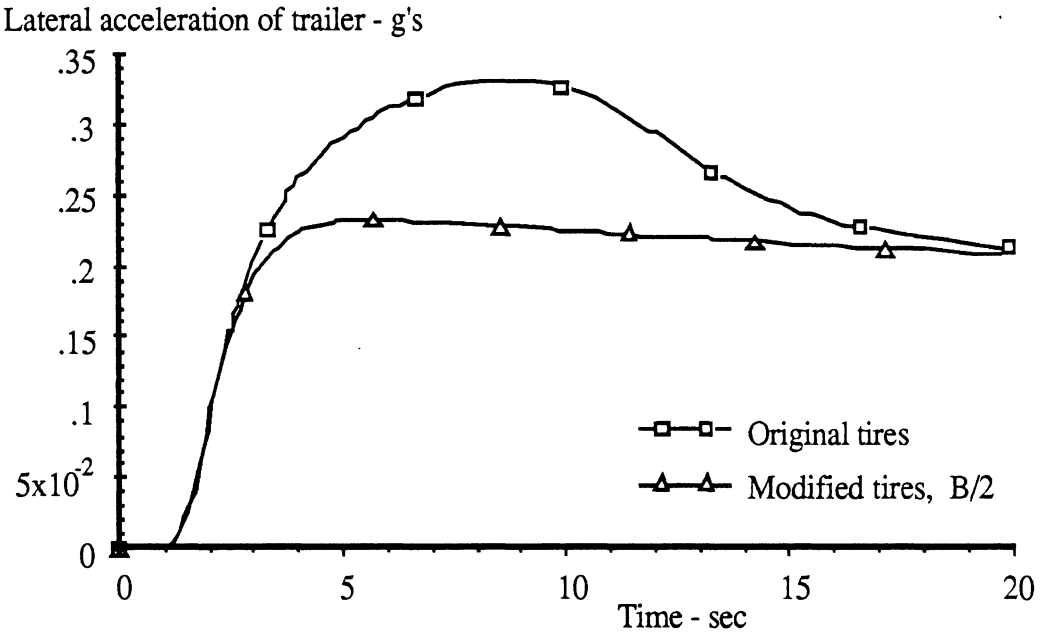


Figure 34: Lateral acceleration comparison

The improved directional stability can also be observed in Figure 35. In the earlier run, the rate of change in articulation angle stabilized after a long time (about 18 sec. following the steering input) in an oscillatory manner. The last run in contrast, attained steady state without oscillations after 4 sec.

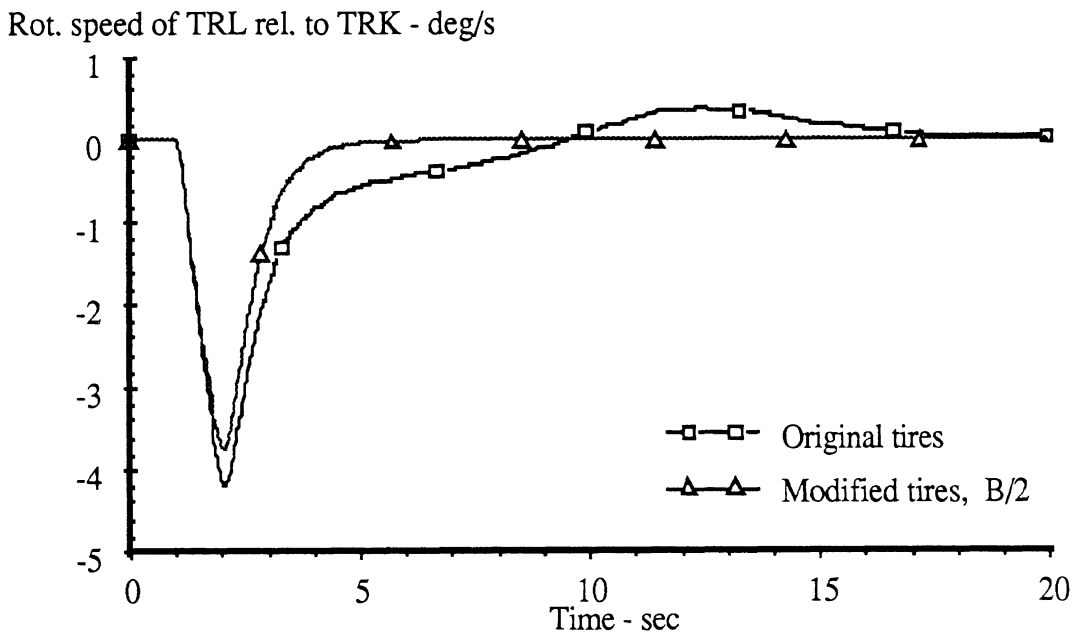


Figure 35: Articulation angle change rate comparison

The next simulation run that was studied involved a severe steer and brake maneuver. It incorporated a combination of two control inputs, none of which can cause by itself the vehicle to rollover, but the outcome of a particular sequence of the two is an abrupt, bona fide rollover. During this simulation, the tractor applies a 2 deg. step steer input while traveling at 38 mph, a maneuver that was found to be controllable and within the stability boundaries of the vehicle. But then, while in the curve, the brakes are fully applied for one second and then released. Figure 36 shows the lateral acceleration buildup as the tractor semitrailer negotiates the turn until $t = 7$ sec. At this point the brakes are applied, the wheels lock up, and due to the skidding and sudden drop of adhesion, the lateral acceleration dips momentarily to zero. As a result of sliding friction there are still lateral forces, and the lateral acceleration starts to grow gradually. After one second, at $t = 8$, the brakes are released. The instantaneous reduction of braking force causes the generation of a large cornering force, and subsequently a pulse-like lateral acceleration that leads to rollover.

It should be noted at this point that the instantaneous occurrence of events in this process, which is an inevitable result of the quasi-static approach fostered in this model, is to some extent a distortion of the expected real-life behavior. Comments on this aspect of the model are presented in Section 5.

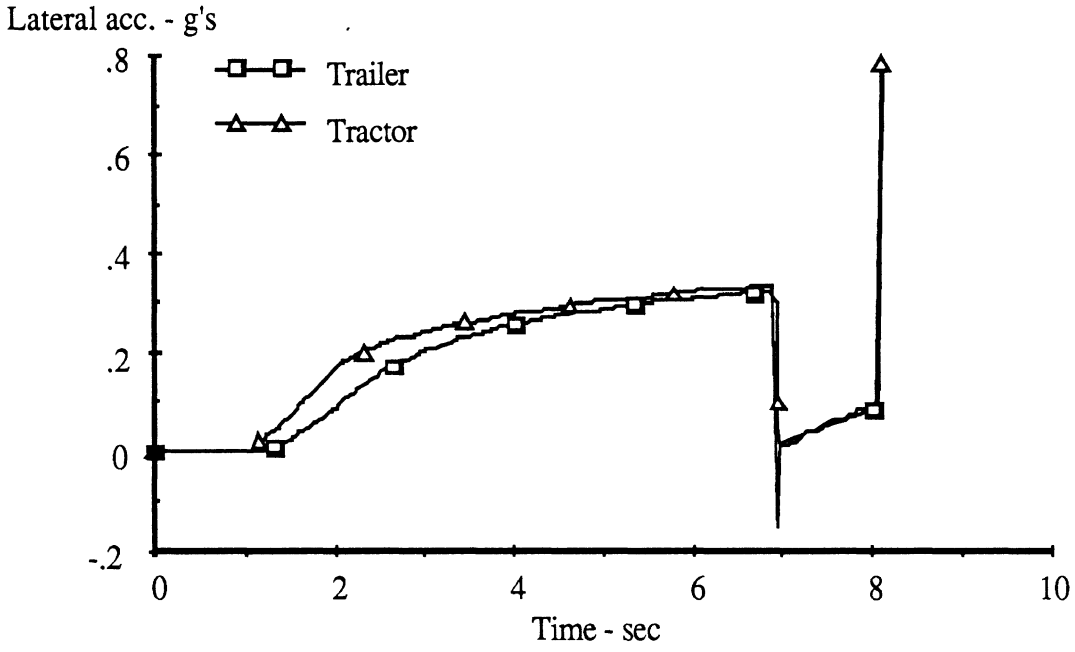


Figure 36: Lateral accelerations in tractor and trailer

Another perspective of the process described above is provided by Figure 37, which presents the roll angle of the vehicle.

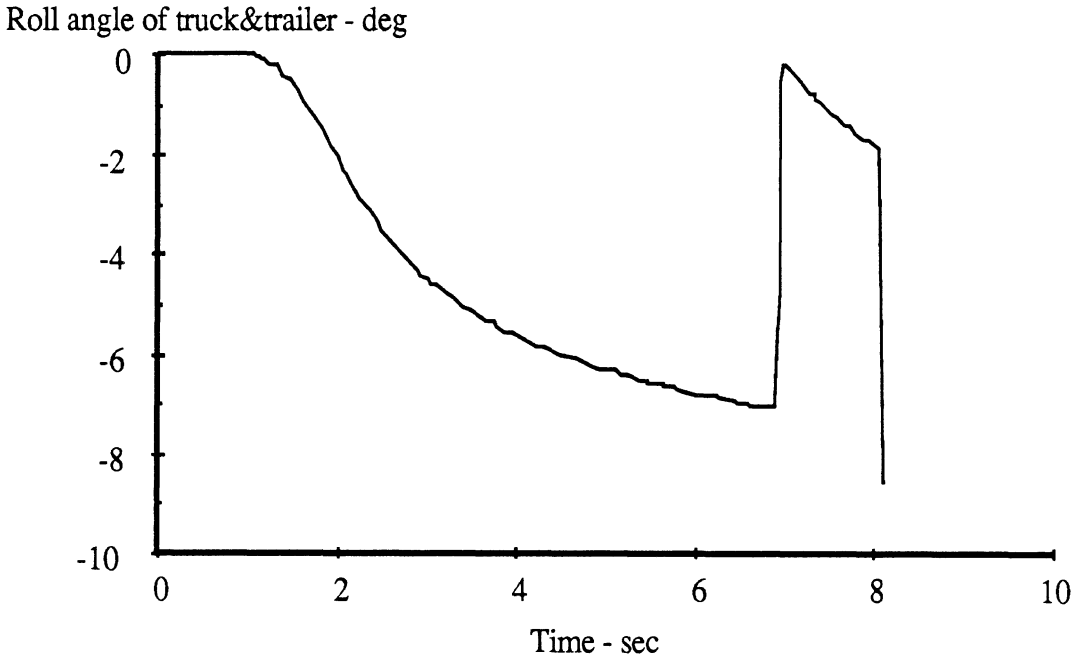


Figure 37: Roll angle

The effect of variations of vertical loads on the individual wheels at the rear axle of the tractor is reflected in Figure 38. Under normal negotiation of a turn, load is transferred from the inner wheel (FZ6) to the outer one (FZ5). As the brakes are applied to lock the

wheels causing the lateral acceleration and roll angle to drop to zero (Figures 36, 37), the vehicle "straightens up" so that the wheels almost restore their static vertical loads. Some load transfer then takes place as a result of lateral force from sliding friction, and when the brakes are released (thereby restoring cornering forces), load is then transferred to a point where it diminishes to zero for the inner wheels, and they are lifted off the ground.

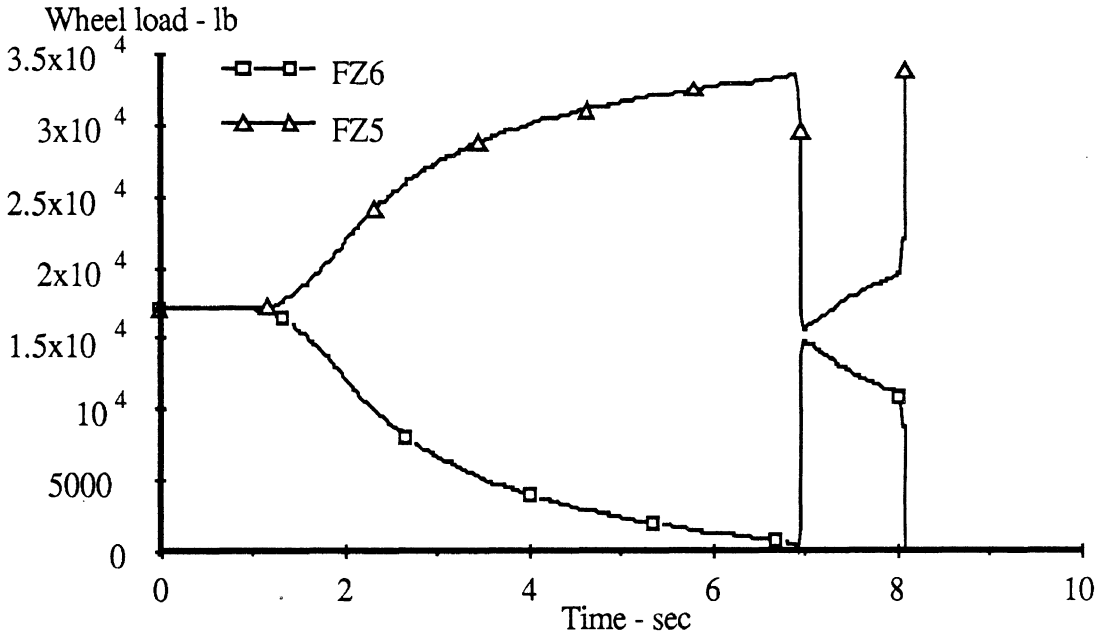


Figure 38: Vertical load on inner and outer wheels at tractor rear axle

The evolution of adhesion forces in the contact patch of the outer rear wheel of the tractor is presented in Figure 39. As the vehicle negotiates the turn, and due to the increased vertical load on the wheel, the lateral force (FY5) keeps growing until 7 sec. when the brakes are applied. At this point, since all the available traction is "taken" by the locked wheels, the lateral force almost diminishes while the longitudinal braking force (FX5) is introduced in an instantaneous manner. After one second when the brakes are released, the longitudinal force drops to zero and the lateral force is recovered, causing the vehicle to roll over.

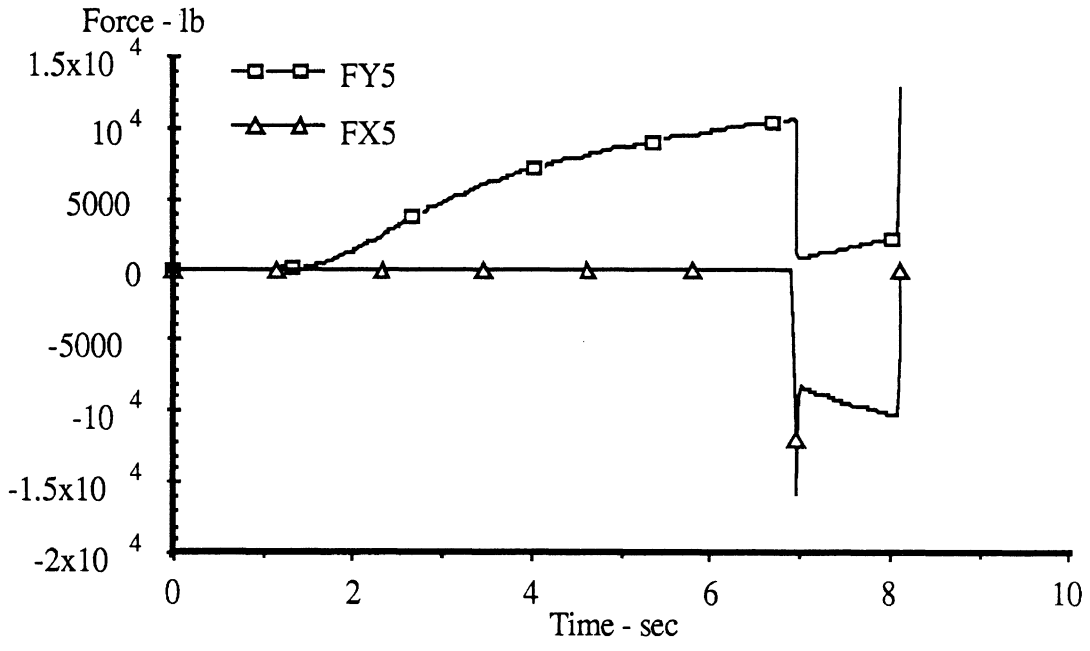


Figure 39: Lateral and longitudinal forces at outer rear tractor wheel

Next, the following run simulated driving conditions under which in "real-life", a jackknife situation would occur. The tractor semitrailer was driven on pavement inclined towards being slippery, with a friction coefficient of 0.65. A minimal steering input was then introduced, just enough to create a small articulation angle between the tractor and the trailer, so that the combination will be able to buckle under braking. Fig 40 presents the tractor heading angle and tractor/semitrailer articulation angle. It is recognizable that at each moment the yaw and articulation angles are almost the same size. This shows that when the brakes were applied the trailer hardly turned, the articulation angle is mostly due to the spinning-skidding tractor, with only a minor change in the forward direction of the semitrailer.

The longitudinal and lateral forces at the front outer wheel of the tractor are shown in Figure 41. As the tractor enters the turn, at t = 1 sec., FY starts growing. When the

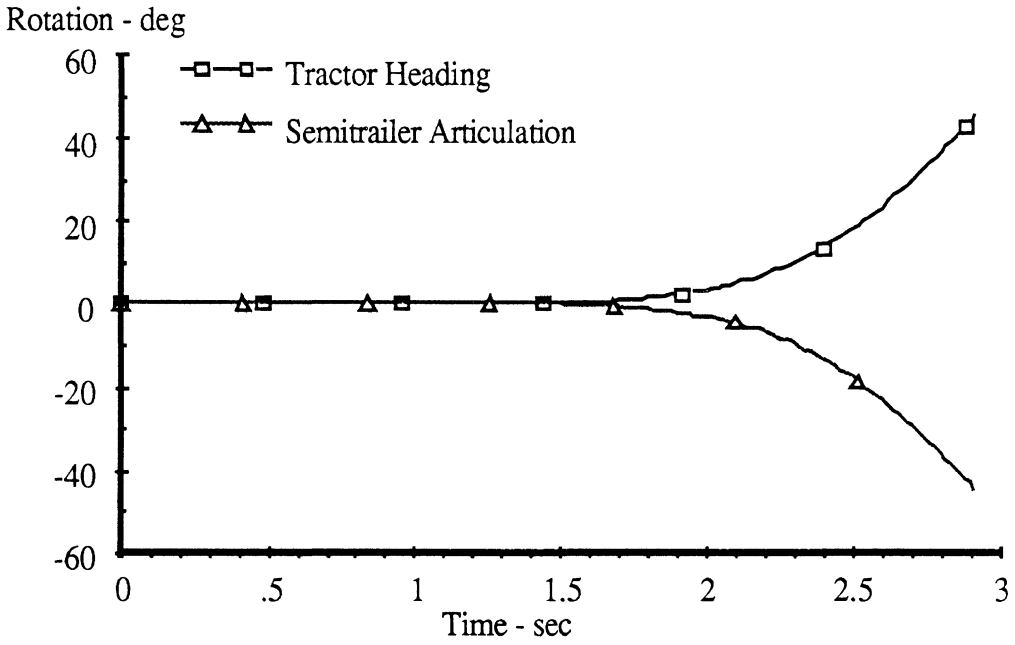


Figure 40: Yaw and articulation angles for tractor and semitrailer

brakes are applied, FX grows to -5000 lb., which is the braking force determined by the braking system characteristics. It stays constant until about t = 2.35 sec., since the vertical load is sufficient to support the traction forces without skidding. At this time, FY had grown to a point that the vertical load is no longer sufficient to provide enough adhesion for both FX and FY, and the wheel skids.

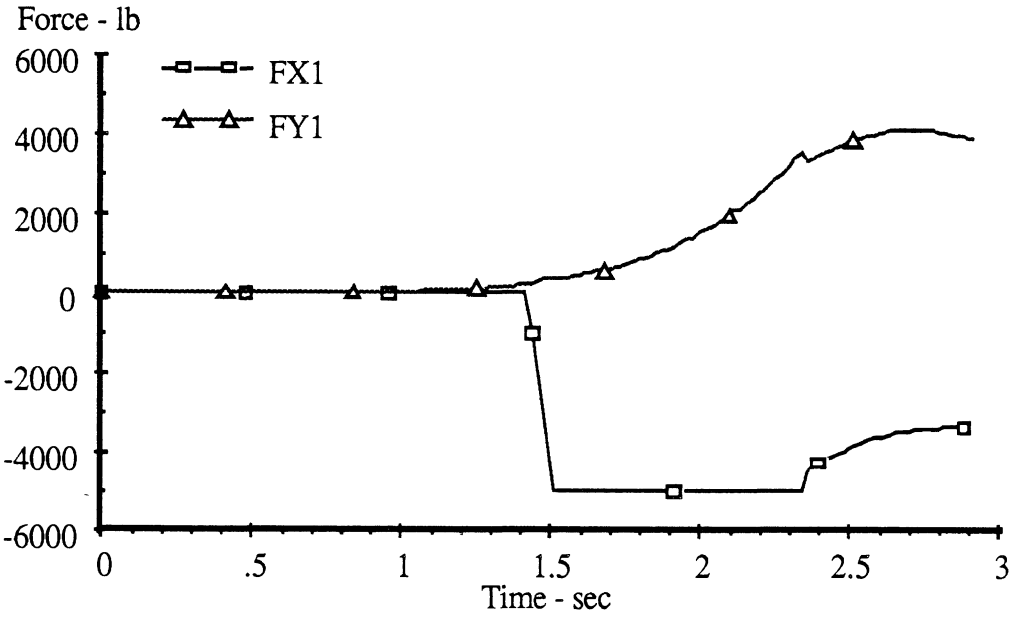


Figure 41: Tractive forces at outer front tractor wheel

Figure 42 presents the lateral accelerations of the tractor and the semitrailer. At first, there is a steady growth of lateral acceleration due to the entrance into the turn, but when the brakes are applied, some wheels skid, and the accelerations drop. The observation that was made before that the trailer had hardly entered the turn, is emphasized here again. The line that represents the trailer lateral acceleration deviates only slightly from zero. Near the end of the jackknife at about t = 2.5 sec., there is a small drop in the tractor's acceleration due to a front wheel skid that causes reduced adhesion, and subsequently a reduction in the lateral force generated.

Figure 42: Lateral acceleration of tractor and semitrailer

The maneuvering situations described in detail in this section illustrate some of the driving situations that would be interesting to study and to use in driver training using a truck driving simulator.

4.0 FUTURE MODEL

The simulation model studied in this report was developed in order to examine and demonstrate two main aspects of its usage in a driving simulator: first, the process of generating an efficient simulation code by AUTOSIM; second, the feasibility of using such a code employing a set of control inputs similar by nature to those expected to be used in a real driving simulator. The output of this simulation as presented in the previous section seems to be valid and adequately detailed (within the model limitations). Nevertheless, the development of a more advanced and more comprehensive simulation model for incorporation into a full-scale driving simulator is a desired next step.

The additional virtues and details of such an enhanced model would be obtained by modifying the vehicle model represented by the block entitled "**Equation of Motion Solver**" in Figure 1. As indicated in Section 2.2 and reflected in Figure 4, the simulation code is composed of two main elements: (1) the code generated by AUTOSIM according to the description of the dynamic system, and (2) the Auxiliary Subroutines that serve as means to introduce (a) control inputs and (b) special purpose physical qualities representing real life operating conditions. The following lists of supplemental features and details that are desired to be added to the simulation, are therefore presented with relation to the appropriate element of the code:

Additions to the AUTOSIM code

- Dynamic computations instead of the quasi-static approach for roll and pitch.
- Detachment of the different units one from the other, in order to enable each of them to roll independently within the restrictions imposed by their hitches.
- Incorporating a flexible means of introducing different hitching mechanisms into the model.
- Separation of the sprung masses from the unsprung masses in each of the units simulated.
- Modified definition and computation of tire slip angles in order to enable driving in reverse.

Auxiliary Subroutines

- Introduction of suspension non-linearities.
- Tire model that will incorporate a true low speed development of slip angles.

- Roll-steer effects.
- Braking and accelerating delays.
- Ability to introduce, during the simulation, variations of environmental conditions such as road friction, wind gusts, or superelevation in curves.
- Steering system compliance.
- Slosh and slide of load.

5.0 CONCLUSIONS AND RECOMMENDATIONS

Comparing the results of the model developed in this work with results from other existing, well-proven models, shows that this model is reasonably accurate. Yet, by employing the newly developed AUTOSIM, the resultant simulation code is significantly more efficient, and therefore faster. This virtue of higher computation speed, enables performing a real-time simulation for relatively complex models.

The selected study matrix in Section 3, where the simulation results are presented, demonstrates limitations of the quasi-static approach. The lack of roll dynamics reduces the level of expected directional stability due to the simulations "ability" to undergo instantaneous changes of roll. Implementing such a model into a driving simulator would require adding the roll dynamic.

Other drawbacks one might observe in this simplified approach of modeling the truck or the bus, include: lack of detailed description of components (e.g., suspensions); missing handling aspects of the vehicle (e.g., roll-steer); and limited performance envelope (e.g., reverse motion).

On the other hand, this modeling approach entails some important advantages: high computational efficiency and threfore speed; and "modular" construction of the simulation code that enables adding "tailored" modeling of components (e.g., tires, suspensions). Perhaps the most powerful benefit of AUTOSIM is that it provides the analyst with an easy tool for generating or modifying simulation codes. This flexibility actually eliminates the drawbacks mentioned above. All the additional virtues of a more enhanced model are relatively easy to append. The features and details listed in section 4.0, and links for the provision of control interaction (e.g., environmental changes) can be added to generate a comprehensive, dynamically complete model for a driving simulator.

It is only natural that such a sophisticated, refined model with added features will be developed as a desirable next step in this process of modeling a truck or a bus for implementation in a driving simulator.

The use of AUTOSIM in generating the simulation frees the analyst from most of the "peripheral" programming "headaches," so that he can concentrate in describing the dynamic behavior and aspects of the system. It does pose some limitations, though, in

deriving the code: variations in the vehicle configuration for instance, cannot be automatically integrated into the program initially. If one wishes to generate a code that will incorporate the embodied option of one, two, or three axles in a suspension, for example, it will be necessary to derive either a separate code for each case, or manually modify the code of one of the cases to include configurations with additional axles.

AUTOSIM manipulates the equations to be used in a simulation before writing the code, and simplifies them so that the resulting code is more efficient than if it were to be written in a "conventional" programming manner.

REFERENCES

1. Michael W. Sayers, *"Symbolic Computer Methods to Automatically Formulate Vehicle Simulation Codes."* A Doctoral Dissertation (Partial Fulfillment of Requirements), The Univ. of Michigan, Trans. Res. Inst., Feb. 1990.

2. Mike Sayers, *"AUTOSIM Users Manual, Version 1.0 B4."* The Univ. of Michigan, Trans. Res. Inst., Mar. 1990.

3. Philip M. Leucht, *"Directional Dynamics of the Tractor-Semitrailer Vehicle."* A Doctoral Dissertation (Partial Fulfillment of Requirements), The Univ. of Michigan, Trans. Res. Inst., Oct. 1970.

4. C. Mallikarjunarao, P. Fancher, *"Analysis of the Directional Response Characteristics of Double Tankers."* SAE paper 781064, Dec. 1978.

5. Paul S. Fancher, Arvind Mathew, *"A Vehicle.Dynamics Handbook for Single-Unit and Articulated Heavy Trucks."* DOT HS 807 185 Final Report, The Univ. of Michigan, Trans. Res. Inst., May 1987.

APPENDIX A

TIRE MODEL AND LOAD TRANSFER EQUATIONS

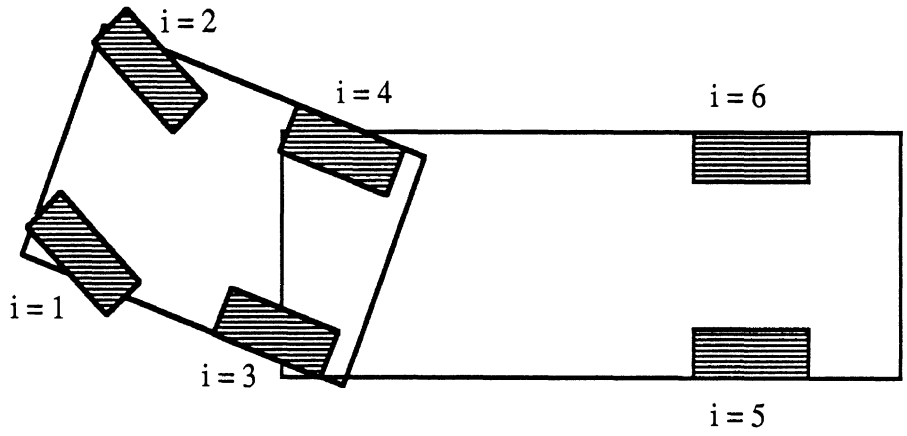
This appendix lists the computational methods for evaluating the tire shear forces and load transfer used in the auxiliary subroutines of the simulation.

- TIRE MODEL

The representation of tire cornering stiffness employed in the simulation is a simplified one, based on two empirical coefficients — A and B, through which the vertical tire load is related to its cornering stiffness using the following general relationship:

$$\frac{C_\alpha}{F_Z} = A - B \cdot F_Z \tag{1}$$

As described in the tractor semi-trailer model of this particular simulation, the tires are lumped into 6 equivalent tires, whose positions are shown in the following Figure:



The equations listed in this tire model, refer to each position with an index letter "i".

Three steps are involved in calculating the tire shear forces:

Step one

The first step is to check and see whether the required braking forces (if they exist) are within the frictional capabilities of the tire-road interface, or if the friction limits are exceeded and the tire locked-up. At this point, since it is a preliminary check, lateral force requirements are not taken into account yet. This check is done for each lumped tire-ground contact point separately:

If $F_B \geq \mu \cdot F_Z \cos(\alpha)$ then the friction limits are exceeded, the tire is locked-up, and the available longitudinal and lateral forces are determined by the sliding values as follows:

(2) $$F_{Xi} = -\mu_S \cdot F_{Zi} \cdot \cos(\alpha_i)$$

(3) $$F_{Yi} = -\mu_S \cdot F_{Zi} \cdot \sin(\alpha_i)$$

Step two

In the case where braking force requirements do not cause a lock-up, step two is evaluated. Based on the slip angle of the tire in position "i" (which was computed in the main body of the program) and the empirical coefficients, eq. (1) is used to determine the lateral force produced by tire i in the absence of any longitudinal force:

(4) $$\frac{C_{\alpha i}}{F_{zis}} = A_i - B_i F_{zis}$$

Define $\bar{\alpha}_i$ to be:

(5) $$\bar{\alpha}_i = \frac{C_{\alpha i}}{F_{zis}} \cdot \frac{\alpha_i}{\mu}$$

According to the value of $\overline{\alpha}_i$, the lateral force generated by the lumped tire in position "i" is (represented as an intermediate value for later use):

For $|\overline{\alpha}_i| \geq 3$:

(6) $$F_{Y(int)} = -\mu F_{zi}\left[\text{sign}(\overline{\alpha}_i)\right]$$

For $|\overline{\alpha}_i| < 3$:

(7) $$F_{Y(int)} = \mu F_{zi}\left(-\overline{\alpha}_i + \frac{1}{3}\overline{\alpha}_i \cdot |\overline{\alpha}_i| - \frac{1}{27}\overline{\alpha}_i^{\,3}\right)$$

It should be noted that equations (6) and (7) make use of F_{zi} which is the total load on all N_{ti} tires at position "i", where (4) and (5) use F_{zis} which is the vertical load on one tire at position "i".

Step three

The third step is to see if the friction can support both the longitudinal and lateral force components. If condition (8) prevails, then the friction is sufficient, and the lateral and longitudinal forces are as determined according to (9), (10). Note that the vehicle simulated here can generate braking forces (F_{Bi}) in all positions 1 through 6, where acceleration forces (F_{Ai}) can only be generated in positions 3 and 4 (tractor's rear axle):

(8) $$\left[F_{Y(int)}\right]^2 + \left[F_{Ai} - F_{Bi}\right]^2 \leq \left[\mu \cdot F_{zi}\right]^2$$

(9) $$F_{Yi} = F_{Y(int)}$$

(10) $$F_{Xi} = F_{Ai} - F_{Bi}$$

If the friction cannot support the shear forces, and condition (8) does not hold, then the lateral and longitudinal forces are computed as follows:

$$F_{Yi} = \left[\frac{\mu \cdot F_{zi}}{\sqrt{F_{Y(int)}^2 - (F_{Ai} - F_{Bi})^2}} \right] \cdot F_{Y(int)} \tag{11}$$

$$F_{Xi} = \left[\frac{\mu \cdot F_{zi}}{\sqrt{F_{Y(int)}^2 - (F_{Ai} - F_{Bi})^2}} \right] \cdot (F_{Ai} - F_{Bi}) \tag{12}$$

- LOAD TRANSFER

Since the model described in this work is a planar one, the load transfer calculations are based on a quasi-static approach. That is, for each time step, the accelerations and tire shear forces produced are used to statically determine longitudinal and lateral load transfer.

Longitudinal Effects

The longitudinal load transfer is calculated in a quasi-static manner according to the following equations:

(1) $$F_{za3} = \left(\frac{1}{L_2}\right)\cdot\left[M_2 \cdot a_{x2}\cdot(h_2 - h_5) + W_2 \cdot X_{2f} + (F_{x5} + F_{x6})\cdot h_5\right]$$

(2) $$F_{za2} = \left(\frac{1}{L_1}\right)\cdot\left[M_1 \cdot a_{x1}\cdot h_1 + W_1 \cdot X_{11} + F_{xc1}\cdot h_5 + F_{zc2}\cdot L_{1h}\right]$$

Where :

$$F_{zc2} = W_2 - F_{za3}$$

$$F_{xc2} = -(F_{x5} + F_{x6}) + M_2 \cdot a_{x2}$$

$$F_{yc2} = -(F_{y5} + F_{y6}) + M_2 \cdot a_{y2}$$

$$F_{xc1} = F_{xc2}\cdot\cos[Q(4)] + F_{yc2}\cdot\sin[Q(4)]$$

(3) $$F_{za1} = W_1 + F_{zc2} - F_{za2}$$

After evaluating the momentary axle loads, the pitch angle can be computed. The polarity of the pitch angle is defined as negative when pitching forward. In the process of calculating the pitch angle, two assumptions were made:

- Only the tractor pitches, and when doing so it pivots around its rear axle.
- Due to the front axle bumpers, the truck cannot pitch more than 2.5 deg.

According to the first assumption, the pitch angle is evaluated as follows:

$$\theta = -\frac{\left(F_{za1} - F_{za1s}\right)}{K_f \cdot L_1} \tag{4}$$

The second assumption is employed by simply setting the magnitude of θ to be 2.5 degrees if it comes out to be a larger negative value than that.

Lateral Effects

The lateral load transfer is calculated in a similar static manner as the longitudinal. The suspension and the tire properties with respect to rolling are combined into a roll restoring moment represented by a roll stiffness parameter for each axle ($K_{\phi i}$). The rollover threshold in the model is defined as the point where the inner wheels of the rear axle of the tractor lift off.

The following simplified equation treats the vehicle (both the tractor and the trailer) as one rolling body, such that:

$$\phi = -\frac{\left(M_1 \cdot a_{y1} \cdot h_1 + M_2 \cdot a_{y2} \cdot h_2\right)}{\left(K_{\phi 1} + K_{\phi 2} + K_{\phi 3}\right) - \left(W_1 \cdot h_1 + W_2 \cdot h_2\right)} \tag{5}$$

The above equation applies only as long as no wheels have lifted off of the ground. An assumption is made that the trailer wheels will be the first to lift off, followed by the rear

axle of the tractor. That sequential assumption can be put into an algebraic inequality as follows:

$$(6) \qquad \frac{K_{\phi 1}}{F_{za1} \cdot T_1} < \frac{K_{\phi 2}}{F_{za2} \cdot T_2} < \frac{K_{\phi 3}}{F_{za3} \cdot T_3}$$

The equations and relations for the vertical "lumped wheels" loads resulting from the above assumption and equation (5) are as follows (expressions set 7):

$$(7) \qquad F_{z1} = \frac{F_{za1}}{2} - \frac{K_{\phi 1} \cdot \phi}{T_1}$$

$$F_{z2} = \frac{F_{za1}}{2} + \frac{K_{\phi 1} \cdot \phi}{T_1}$$

$$F_{z3} = \frac{F_{za2}}{2} - \frac{K_{\phi 2} \cdot \phi}{T_2}$$

$$F_{z4} = \frac{F_{za2}}{2} + \frac{K_{\phi 2} \cdot \phi}{T_2}$$

$$F_{z5} = \frac{F_{za3}}{2} - \frac{K_{\phi 3} \cdot \phi}{T_3}$$

$$F_{z6} = \frac{F_{za3}}{2} + \frac{K_{\phi 3} \cdot \phi}{T_3}$$

Once the trailer wheels lifted off the ground, the roll angle had reached a level such that:

$$(8) \qquad \left| K_{\phi 3} \cdot \phi \right| > \frac{F_{za3}}{2} \cdot T_3$$

At this point the previous equations (7) do not hold anymore. The restoring moment from the trailer suspension has saturated and reached its maximum value. It now has a fixed value of $(F_{za3} \cdot T_3)/2$

The new equation for the roll angle, replacing equation (5), is now:

$$(9) \qquad \phi = -\frac{\left(M_1 \cdot a_{y1} \cdot h_1 + M_2 \cdot a_{y2} \cdot h_2 + \frac{F_{za3} \cdot T_3}{2} \cdot \left(\text{sign}(a_{y2})\right)\right)}{\left(K_{\phi 1} + K_{\phi 2}\right) - \left(W_1 \cdot h_1 + W_2 \cdot h_2\right)}$$

The equations and relations for the vertical "lumped wheels" loads at positions 1 through 4 remain as in expressions set (7). The loads on tires 5 and 6 are now:

$$(10) \qquad F_{z5} = F_{za3} \qquad \text{(for negative roll angle)}$$

$$0 \qquad \text{(for positive roll angle)}$$

$$F_{z6} = 0 \qquad \text{(for negative roll angle)}$$

$$F_{za3} \qquad \text{(for positive roll angle)}$$

Finally, once a condition similar to the one expressed in equation (8) applies to the rear axle of the tractor, then:

$$(11) \qquad \left|K_{\phi 2} \cdot \phi\right| > \frac{F_{za2}}{2} \cdot T_2$$

This means that this axle has lifted off of the ground, and the rollover threshold has been exceeded.

APPENDIX B

BRAKING AND ACCELERATION EQUATIONS

This appendix lists the computational methods for evaluating the demanded braking and acceleration forces as interpreted from a pedal position input, based on braking system and engine performance. The following equations are incorporated in the auxiliary subroutines of the simulation.

- BRAKING

The braking model employed in the simulation incorporates the following parameters representing braking system components:

- Brake pedal position (BP): Ranges from 0.0 (no braking applied) to 1.0 (full braking force demanded) according to a table in the input file. It governs the braking pressure being "sent to the actuators."
- Brake system pressure (PP): The steady state pressure in the tractor's air tanks. Set to be constant at 100 psi (but can be easily changed).
- Axle brake gain (BG): A parameter whose value is set in the input file (according to the simulated tractor's characteristics) representing the ratio between the imposed braking moment on the axle and the braking pressure (as dictated by the pedal).
- Tire radius (TR): Set in the input file according to the simulated data.

The following equation combines the above parameters to calculate the demanded braking force of a lumped tire on axle i (this braking force is identical for the two lumped tires on the axle).

$$F_B = 0.5 \cdot \frac{BP \cdot PP \cdot BG_i}{TR}$$

- ACCELERATION

The required acceleration force calculation in the simulation incorporates the following parameters:

- Acceleration pedal position (AP): Ranges from 0.0 (no acceleration demand) to 1.0 (full accelerating force demanded) according to a table in the input file.
- Maximum available engine power (HP).
- Vehicle forward speed (U(1)).

The total driving force provided by the engine, according to the pedal position, is expressed by the following equation. The individual tire's force is the result of the total force, divided by the number of lumped tires on driving axles. In this particular simulation there is one lumped driving axle, hence two lumped driving wheels, so that the tire's force is half the result of the equation:

$$\overline{F_A} = \frac{HP \cdot 550 \cdot 12 \cdot AP}{U(1)}$$

In order to prevent infinite force when accelerating from zero, a boundary of about 5 mph was set, below which the speed (U(1)) is held constant at 5 mph in the equation above.

APPENDIX C

SIMULATION DATA AND CONSTANT PARAMETERS

This appendix lists the file that contains the echoed input data. The program reads only data that is accompanied by an appropriate, predetermined keyword and it echoes the data to an echo file using the same keywords. All other characters that may by found in the input file are ignored. In light of this fact, the echo file listed below can be used, as it is, as an input file.

```
PARSFILE
* Echo file created by:
* Simulation of planar tractor/semi truck.
* Version created by AUTOSIM 1.0 B5 on April 16, 1990.
* (c) The Regents of The University of Michigan, 1989, 1990. All rights reserved.

TITLE  Severe Maneuver, V=38

* Input File: in
* Run was made 17:10 on Apr 16, 1990

* PARAMETER VALUES

BRAKE
       2000.00     Gain for axle 1 (axle brake torque[in-lb]/input pressure[psi])
       6000.00     Gain for axle 2 (axle brake torque[in-lb]/input pressure[psi])
       6000.00     Gain for axle 3 (axle brake torque[in-lb]/input pressure[psi])
       20.0000     Rolling radius of the tires [in]
        6              Number of points in time vs. pedal position table (number)
  .000000    ,   .000000          point for table-lookup: [sec], position
  6.90000    ,   .000000          point for table-lookup: [sec], position
  7.10000    ,   1.00000          point for table-lookup: [sec], position
  8.00000    ,   1.00000          point for table-lookup: [sec], position
  8.10000    ,   .000000          point for table-lookup: [sec], position
  9.00000    ,   .000000          point for table-lookup: [sec], position
```

```
CDRAG     .000000        coefficient in aerodynamic drag force (lb-sec2/in2)
CRR       .000000        coefficient in argument to FRR in rolling resistance drag
                                                                    force (-)
FRICTION
      .80                Tire/Road friction value
      .90                Ratio between sliding friction and "normal" friction
H1        36.0000        c.g. height of tractor (in)
H2        78.0000        c.g. height of semi-trailer (in)
HC        50.0000        height of fifth-wheel connection (in)
IPRINT    1.00000        number of time steps between output printing (counts)
ITRK33    233000.        moment of inertia of TRK (in-lb-s2)
ITRL33  0.470000E+07     moment of inertia of TRL (in-lb-s2)
L1        150.000        tractor wheelbase (in)
L1H       138.000        distance from tractor front axle to hitch (in)
L2        488.000        trailer wheelbase (in)
MTRK      16000.0        mass of TRK (lbm)
MTRL      62000.0        mass of TRL (lbm)
ROL&PTCH
    2000.0           Longitudinal (Pitch) Stiffness of Tractor
 3                   Total number of lumped axles
 1146000.0           Lateral (Roll) Stiffness of Axle #1
 8022000.0           Lateral (Roll) Stiffness of Axle #2
10314000.0           Lateral (Roll) Stiffness of Axle #3
STEER
        4               Number of points in time vs. front-axle steer (number)
  .000000    ,   .000000          point for table-lookup (sec), (deg)
  1.00000    ,   .000000          point for table-lookup (sec), (deg)
  2.00000    ,   2.00000          point for table-lookup (sec), (deg)
  5.00000    ,   2.00000          point for table-lookup (sec), (deg)
STEP    0.200000E-01     simulation time step (sec)
STOPT     10.0000        simulation stop time (sec)
T1        80.0000        track for axle #1 (in)
T2        72.0000        track for axle #2 (in)
T3        78.0000        track for axle #3 (in)
```

```
THROTTLE
      300              Maximum road Horsepower of the engine
        4              Number of points in time vs. pedal position (number)
  .000000    ,   .000000          point for table-lookup [sec], position
  .000000    ,   .000000          point for table-lookup [sec], position
  .000000    ,   .000000          point for table-lookup [sec], position
  .000000    ,   .000000          point for table-lookup [sec], position
TIRE1                 Tire model of form: Calph = A*Fz - B*Fz**2
  .169000             Coefficient A in tire model (1/deg)
 0.867000E-05         Coefficient B in tire model (1/(lb-deg))
   1.                 Number of tires at location (count)
  5123.12             nominal static load (lb)
  638.251             nominal cornering stiffness (lb/deg)
TIRE2                 Tire model of form: Calph = A*Fz - B*Fz**2
  .169000             Coefficient A in tire model (1/deg)
 0.867000E-05         Coefficient B in tire model (1/(lb-deg))
   1.                 Number of tires at location (count)
  5123.12             nominal static load (lb)
  638.251             nominal cornering stiffness (lb/deg)
TIRE3                 Tire model of form: Calph = A*Fz - B*Fz**2
  .169000             Coefficient A in tire model (1/deg)
 0.867000E-05         Coefficient B in tire model (1/(lb-deg))
   4.                 Number of tires at location (count)
  4228.95             nominal static load (lb)
  559.639             nominal cornering stiffness (lb/deg)
TIRE4                 Tire model of form: Calph = A*Fz - B*Fz**2
  .169000             Coefficient A in tire model (1/deg)
 0.867000E-05         Coefficient B in tire model (1/(lb-deg))
   4.                 Number of tires at location (count)
  4228.95             nominal static load (lb)
  559.639             nominal cornering stiffness (lb/deg)
```

```
TIRE5                   Tire model of form: Calph = A*Fz - B*Fz**2
  .169000               Coefficient A in tire model (1/deg)
 0.867000E-05           Coefficient B in tire model (1/(lb-deg))
   4.                   Number of tires at location (count)
  4240.27               nominal static load (lb)
  560.720               nominal cornering stiffness (lb/deg)


TIRE6                   Tire model of form: Calph = A*Fz - B*Fz**2
  .169000               Coefficient A in tire model (1/deg)
 0.867000E-05           Coefficient B in tire model (1/(lb-deg))
   4.                   Number of tires at location (count)
  4240.27               nominal static load (lb)
  560.720               nominal cornering stiffness (lb/deg)
X11      75.0000        distance from tractor mass center to front axle (in)
X2F      267.000        distance from trailer mass center to hitch (in)


* INITIAL CONDITIONS

Q(1)      .000000       Translation of TRK0 relative to O, [n1]. (in)
Q(2)      .000000       Translation of TRK0 relative to O, [n2]. (in)
Q(3)      .000000       Rot. of TRK relative to N about axis #3. (deg)
Q(4)      .000000       Rot. of TRL relative to TRK about axis #3. (deg)
U(1)      668.800       Abs. trans. speed of TRK*, axis 1. (in/s)
U(2)      .000000       Abs. trans. speed of TRK*, axis 2. (in/s)
U(3)      .000000       Abs. rot. speed of TRK, axis 3. (deg/s)
U(4)      .000000       Rot. speed of TRL relative to TRK, axis 3. (deg/s)

END
```

APPENDIX D

"AUTOSIM" GENERATED MAIN PROGRAM AND SUBROUTINES

This appendix lists the file that contains the main program and subroutines which were generated by "AUTOSIM." Calls for the auxiliary subroutines are made from within this file.

```
C  Simulation of planar tractor/semi truck.
C  Version created by AUTOSIM 1.0 B5 on April 16, 1990.
C  (c) The Regents of The University of Michigan, 1989, 1990. All
C  rights reserved.
C
      IMPLICIT NONE
      CHARACTER*80  INFILE, TITLE
      REAL          PARS, STEP, STOPT, T, Y, YP
      INTEGER       I, IECHO, IFILE, ILOOP1, ILOOP2, IPRNT2, ISEC1,
     &              ISEC2, NCOORD, NLOOP, NPARS, NSPEED, NTOT
C
      PARAMETER    (NCOORD = 4, NSPEED = 4, IFILE = 1, NTOT = 8)
      DIMENSION    Y(NTOT), YP(NTOT)
      PARAMETER    (NPARS = 20)
      DIMENSION    PARS(NPARS)
      COMMON       /INPARS/ PARS, TITLE, INFILE
      SAVE         /INPARS/
C
      EQUIVALENCE (PARS(14), STEP), (PARS(15), STOPT)
C
      WRITE(*, '(5A)')
     & ' Simulation of planar tractor/semi truck.'
      WRITE(*, '(5A)')
     & ' Version created by AUTOSIM 1.0 B5 on April 16, 1990.'
      WRITE(*, '(5A)')
     & ' (c) The Regents of The University of Michigan, 1989, 1990.'
     &,' All rights reserved.'
      WRITE(*, '(5A)')
     & ' '
C
C  Read input data
C
      CALL INPUT(Y, Y(NCOORD + 1))
      IPRNT2 = PARS(6)
C
C  Compute constants in common block /PRCMP/ before starting.
C
      CALL PRECMP
C
C   Option to echo data and initial conditions
C
      CALL ECHO(IFILE, Y, Y(NCOORD + 1), 'INITIAL')
C
```

```
C   Set up output file with simulated time histories
C
      CALL OPNOUT(IFILE)
      CALL TIME(ISEC1)
C
C Start by evaluating derivatives and printing variables at t=0
C
      T = 0.
      CALL DIFEQN(T, Y, YP, Y(NCOORD + 1), YP(NCOORD + 1))
      CALL UPDATE(T, Y, YP, Y(NCOORD + 1), YP(NCOORD + 1))
      CALL OUTPUT(IFILE, T, Y, YP, Y(NCOORD + 1), YP(NCOORD + 1))
C
C Integration loop. Continue until printout time reaches final time.
C
      NLOOP = STOPT / STEP / IPRNT2 + 1
      DO 60 ILOOP1 = 1, NLOOP
        DO 50 ILOOP2 = 1, IPRNT2
          CALL INTEQS(T, Y, YP, NTOT, STEP)
          T = T + STEP
          CALL FCT(T, Y, YP)
          CALL UPDATE(T, Y, YP, Y(NCOORD + 1), YP(NCOORD + 1))
          IF (T .GE. STOPT) GO TO 55
   50   CONTINUE
   55   CALL OUTPUT (IFILE, T, Y, YP, Y(NCOORD + 1), YP(NCOORD + 1))
        IF (T .GE. STOPT) GO TO 70
   60 CONTINUE
   70 CONTINUE
C
      CALL TIME (ISEC2)
C
C End of integration loop.  Print final status of run
C
      WRITE(*,*) ' Termination at time =', T, ' sec.'
      WRITE(*,*) ' Computation efficiency: ', (ISEC2 - ISEC1) / T,
     &           ' sec/sim. sec'
      WRITE(*,*) ' '

      CLOSE(IFILE)
C
C   Option to echo data and final conditions
C
      CALL ECHO(IFILE, Y, Y(NCOORD + 1), 'FINAL')
      PAUSE ' Done'
      END

C=====================================================================
      BLOCK DATA
C=====================================================================
      CHARACTER*80  INFILE, TITLE
      REAL          PARS
      INTEGER       NPARS
C
      PARAMETER    (NPARS = 20)
      DIMENSION    PARS(NPARS)
      COMMON       /INPARS/ PARS, TITLE, INFILE
      SAVE         /INPARS/
C
```

```
C This is an include file to initialize parameters in the
C tractor-semitrailer simulation that are "hidden" to
C AUTOSIM.

      include tire.inc
      include tables.inc
      include brakes.inc
      integer totsiz
      parameter (totsiz = 9*tabsiz)

     DATA ATIRE, BTIRE, NTIRE  /6*0.169, 6*0.867E-05, 1, 1, 4, 4, 4, 4/
      DATA FRICT, FRATIO  /.8, .9/
      DATA NTAB /3*2/
      DATA TABLES /TOTSIZ*0./
      DATA BGAIN1, BGAIN2, BGAIN3, RTIRE / 2000., 6000., 6000., 20./
C
C Resume with AUTOSIM-generated stuff...
C
      DATA          PARS /0.0, 0.0, 36.0, 78.0, 50.0, 3.0, 233000.0,
     &              4700000.0, 150.0, 138.0, 488.0, 16000.0, 62000.0,
     &              0.02, 5.0, 80.0, 72.0, 78.0, 75.0, 267.0/
      DATA          INFILE /' '/
      DATA          TITLE  /'Default parameter values'/
      END


C=======================================================================
      SUBROUTINE DIFEQN(T, Q, QP, U, UP)
C=======================================================================
C  This subroutine defines the equations of motion for the Planar
C  tractor/semi truck, which includes 4 degrees of freedom.
C
C  --> T  real  time
C  --> Q  real  array of 4 generalized coordinates
C  <-- QP real  array of derivitives of Q
C  --> U  real  array of 4 generalized speeds
C  <-- UP real  array of derivatives of U
C
C  Each derivative evaluation requires 142 multiply/divides, 119
C  add/subtracts, and 28 function/subroutine calls.
C
C  (c) The Regents of The University of Michigan, 1989, 1990. All
C  rights reserved.
C
      IMPLICIT NONE
      CHARACTER*80  INFILE, TITLE
      REAL          C, CDRAG, CHKA, CRR, DEGREES, FA, FB1, FB2, FB3,
     &              FORCEM, FRR, FX1, FX2, FX3, FX4, FX5, FX6, FY1,
     &              FY2, FY3, FY4, FY5, FY6, FZ1, FZ2, FZ3, FZ4, FZ5,
     &              FZ6, GEES, H1, H2, HC, IPRINT, ITRK33, ITRL33, L1,
     &              L1H, L2, MTRK, MTRL, PARS, PC, PHEE, Q, QP, S,
     &              STEER, STEP, STOPT, T, T1, T2, T3, THETA, U, UP,
     &              X11, X2F, Z
      INTEGER       NCOORD, NPARS, NSPEED
C
```

```
      PARAMETER     (NCOORD = 4, NSPEED = 4)
      DIMENSION     Q(NCOORD), QP(NCOORD), U(NSPEED), UP(NSPEED)
      DIMENSION     C(4), FORCEM(2), S(4), Z(86)
      COMMON        /DYVARS/ C, FORCEM, S, Z, STEER, FA, FB1, FB2, FB3,
     &              FZ1, FZ2, FZ3, FZ4, FZ5, FZ6, PHEE, THETA, FY1, FX1,
     &              FY2, FX2, FY3, FX3, FY4, FX4, FY5, FX5, FY6, FX6
      SAVE          /DYVARS/
C
      PARAMETER     (NPARS = 20)
      DIMENSION     PARS(NPARS)
      COMMON        /INPARS/ PARS, TITLE, INFILE
      SAVE          /INPARS/
C
      EQUIVALENCE   (PARS(1), CDRAG), (PARS(2), CRR), (PARS(3), H1),
     &              (PARS(4), H2), (PARS(5), HC), (PARS(6), IPRINT),
     &              (PARS(7), ITRK33), (PARS(8), ITRL33), (PARS(9), L1),
     &              (PARS(10), L1H), (PARS(11), L2), (PARS(12), MTRK),
     &              (PARS(13), MTRL), (PARS(14), STEP), (PARS(15),
     &              STOPT), (PARS(16), T1), (PARS(17), T2), (PARS(18),
     &              T3), (PARS(19), X11), (PARS(20), X2F)
      DIMENSION     PC(34)
      COMMON        /PRCMP/ PC
      SAVE          /PRCMP/
      PARAMETER     (GEES = 386.2, DEGREES = 57.29577951308232)
      S(3) = SIN(Q(3))
      S(4) = SIN(Q(4))
      C(3) = COS(Q(3))
      C(4) = COS(Q(4))
C
C  Kinematical equations
C
      Z(1) = (X11*U(3) + U(2))
      QP(1) = (U(1)*C(3) -Z(1)*S(3))
      QP(2) = (Z(1)*C(3) + U(1)*S(3))
      QP(3) = U(3)
      QP(4) = U(4)
C
C  External subroutines and extra variables
C
      CALL GETFBS(FB1, FB2, FB3)
      CALL GETFA(FA)
      CALL GETSTR(STEER)
      Z(2) = COS(STEER)
      Z(3) = Z(1)*Z(2)
      Z(4) = PC(1)*U(3)
      Z(5) = (U(1) + Z(4))
      Z(6) = SIN(STEER)
      Z(7) = Z(1)*Z(6)
      Z(8) = Z(1)**2
      Z(9) = CHKA(ATAN2((Z(3) -Z(5)*Z(6)), (Z(7) + Z(5)*Z(2))),
     &        SQRT(Z(8) + Z(5)**2))
C
C  Compute FY1 and FX1 for Tire 1.
C
      CALL TIRE(1, Z(9), FZ1, FY1, FB1, 0, FX1, U(1))
      Z(10) = -(U(1) -Z(4))
      Z(11) = CHKA(ATAN2((Z(3) + Z(10)*Z(6)), (Z(7) -Z(10)*Z(2))),
     &         SQRT(Z(8) + Z(10)**2))
```

```
C  Compute FY2 and FX2 for Tire 2.
C
      CALL TIRE(2, Z(11), FZ2, FY2, FB1, 0, FX2, U(1))
      Z(12) = -(PC(4)*U(3) -U(2))
      Z(13) = PC(2)*U(3)
      Z(14) = (U(1) + Z(13))
      Z(15) = L1*Z(12)
      Z(16) = Z(1)*Z(12)
      Z(17) = CHKA(ATAN2(Z(12), Z(14)), SQRT(U(1)*Z(14) + U(3)
     &        *(PC(2)*Z(14) -Z(15)) + Z(16)))
C
C  Compute FY3 and FX3 for Tire 3.
C
      CALL TIRE(3, Z(17), FZ3, FY3, FB2, FA, FX3, U(1))
      Z(18) = -(U(1) -Z(13))
      Z(19) = CHKA(ATAN2(Z(12), -Z(18)), SQRT(-(-Z(16) + U(3)*(Z(15)
     &         -PC(2)*Z(18)) + U(1)*Z(18))))
C
C  Compute FY4 and FX4 for Tire 4.
C
      CALL TIRE(4, Z(19), FZ4, FY4, FB2, FA, FX4, U(1))
      Z(20) = (U(3) + U(4))
      Z(21) = PC(5)*U(3)
      Z(22) = (U(2) + Z(21))
      Z(23) = Z(22)*C(4)
      Z(24) = U(1)*S(4)
      Z(25) = (L2*Z(20) -Z(23) + Z(24))
      Z(26) = PC(3)*Z(20)
      Z(27) = U(1)*C(4)
      Z(28) = Z(22)*S(4)
      Z(29) = (Z(26) + Z(27) + Z(28))
      Z(30) = L2*Z(25)
      Z(31) = PC(3)*C(4)
      Z(32) = L2*S(4)
      Z(33) = L2*C(4)
      Z(34) = PC(3)*S(4)
      Z(35) = CHKA(-ATAN2(Z(25), Z(29)), SQRT(Z(20)*(PC(3)*Z(29) +
     &         Z(30)) + U(1)*(U(1) + Z(20)*(Z(31) + Z(32))) + Z(22)
     &        *(U(2) + Z(21) -Z(20)*(Z(33) -Z(34)))))
C
C  Compute FY5 and FX5 for Tire 5.
C
      CALL TIRE(5, Z(35), FZ5, FY5, FB3, 0, FX5, U(1))
      Z(36) = CHKA(-ATAN2(Z(25), (-Z(26) + Z(27) + Z(28))),
     &         SQRT(-Z(20)*(PC(3)*(-Z(26) + Z(27) + Z(28)) -Z(30)) +
     &         U(1)*(U(1) -Z(20)*(Z(31) -Z(32))) + Z(22)*(U(2) + Z(21)
     &         -Z(20)*(Z(33) + Z(34)))))
C
C  Compute FY6 and FX6 for Tire 6.
C
      CALL TIRE(6, Z(36), FZ6, FY6, FB3, 0, FX6, U(1))
C
C  define expression for aerodynamic drag force
      Z(37) = U(1)**2
      FORCEM(1) = CDRAG*Z(37)
C
C  define expression for rolling resistance drag force
      FORCEM(2) = FRR(PC(7), U(1))
```

```
C
C  Dynamical equations
C
      Z(38) = PC(5)*S(4)
      Z(39) = (X2F -PC(5)*C(4))
      Z(40) = U(3)*U(1)
      Z(41) = U(3)*U(2)
      Z(42) = (PC(5)*U(3)**2 + Z(41))
      Z(43) = (Z(40)*C(4) + Z(42)*S(4))
      Z(44) = X2F*Z(20)**2
      Z(45) = (Z(44) -Z(42)*C(4) + Z(40)*S(4))
      Z(46) = (FY1 + FY2)
      Z(47) = (FX1 + FX2)
      Z(48) = PC(1)*Z(2)
      Z(49) = X11*Z(6)
      Z(50) = X11*Z(2)
      Z(51) = PC(1)*Z(6)
      Z(52) = (FY5 + FY6)
      Z(53) = (FX5 + FX6)
      Z(54) = -(PC(3)*(FX5 -FX6) -PC(8)*Z(52))
      Z(55) = MTRL*S(4)
      Z(56) = MTRL*C(4)
      Z(57) = MTRL*Z(39)
      Z(58) = MTRL*Z(38)
      Z(59) = (-FX3 -FX4 -MTRK*Z(41) -Z(43)*Z(55) + Z(45)*Z(56) +
     &         FORCEM(1) + FORCEM(2) -Z(47)*Z(2) + Z(46)*Z(6)
     &         -Z(53)*C(4) + Z(52)*S(4))
      Z(60) = (FY3 + FY4 -MTRK*Z(40) -Z(45)*Z(55) -Z(43)*Z(56) +
     &         Z(46)*Z(2) + Z(47)*Z(6) + Z(52)*C(4) + Z(53)*S(4))
      Z(61) = -(-PC(2)*(FX3 -FX4) + PC(4)*(FY3 + FY4) + FX2*(Z(48)
     &         -Z(49)) -FX1*(Z(48) + Z(49)) -FY1*(Z(50) -Z(51)) -FY2
     &        *(Z(50) + Z(51)) + Z(39)*Z(52) -Z(38)*Z(53) + Z(54)
     &         -Z(43)*Z(57) + Z(45)*Z(58))
      Z(62) = (Z(58)*C(4) + Z(57)*S(4))
      Z(63) = (Z(57)*C(4) -Z(58)*S(4))
      Z(64) = PC(9)*S(4)
      Z(65) = PC(9)*C(4)
      Z(66) = (ITRL33 + PC(9)*Z(39))
      Z(67) = PC(12)*Z(62)
      Z(68) = PC(12)*Z(64)
      Z(69) = PC(12)*Z(63)
      Z(70) = PC(12)*Z(65)
      Z(71) = (PC(10) + Z(39)*Z(57) + Z(38)*Z(58) -Z(62)*Z(67)
     &         -Z(63)*Z(69))
      Z(72) = (Z(66) -Z(62)*Z(68) -Z(63)*Z(70))/Z(71)
      Z(73) = (Z(66) -Z(64)*Z(67) -Z(65)*Z(69))
      Z(74) = Z(59)*Z(67)
      Z(75) = Z(60)*Z(69)
      Z(76) = -(PC(9)*Z(43) -X2F*Z(52) -Z(54) + Z(59)*Z(68) +
     &         Z(60)*Z(70) + Z(72)*(-Z(61) -Z(74) -Z(75)))/(PC(11)
     &         -Z(64)*Z(68) -Z(65)*Z(70) -Z(72)*Z(73))
      Z(77) = (-Z(61) -Z(74) -Z(75) -Z(73)*Z(76))/Z(71)
      UP(4) = -Z(76)
      UP(3) = -Z(77)
      UP(2) = PC(12)*(Z(60) -Z(65)*Z(76) -Z(63)*Z(77))
      UP(1) = -PC(12)*(Z(59) -Z(64)*Z(76) -Z(62)*Z(77))
      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE ECHO(IFILE, Q, U, WHEN)
C=======================================================================
C  This subroutine prompts the user for the name of an optional echo
C  file for the planar tractor/semi truck.  If a file is selected, all
C  of the parameter values and initial conditions are written to
C  confirm that the intended values were used in the simulation.
C
C  (c) The Regents of The University of Michigan, 1989, 1990. All
C  rights reserved.
C
      IMPLICIT NONE
      CHARACTER*(*) WHEN
      CHARACTER*24  TIMEDT
      CHARACTER*80  INFILE, OPNFIL, TITLE
      REAL          CDRAG, CRR, DEGREES, GEES, H1, H2, HC, IPRINT,
     &              ITRK33, ITRL33, L1, L1H, L2, MTRK, MTRL, PARS, Q,
     &              STEP, STOPT, T, T1, T2, T3, U, X11, X2F
      INTEGER       IFILE, NCOORD, NPARS, NSPEED
C
      PARAMETER     (NPARS = 20)
      DIMENSION     PARS(NPARS)
      COMMON        /INPARS/ PARS, TITLE, INFILE
      SAVE          /INPARS/
C
      EQUIVALENCE   (PARS(1), CDRAG), (PARS(2), CRR), (PARS(3), H1),
     &              (PARS(4), H2), (PARS(5), HC), (PARS(6), IPRINT),
     &              (PARS(7), ITRK33), (PARS(8), ITRL33), (PARS(9), L1),
     &              (PARS(10), L1H), (PARS(11), L2), (PARS(12), MTRK),
     &              (PARS(13), MTRL), (PARS(14), STEP), (PARS(15),
     &              STOPT), (PARS(16), T1), (PARS(17), T2), (PARS(18),
     &              T3), (PARS(19), X11), (PARS(20), X2F)
      PARAMETER     (NCOORD = 4, NSPEED = 4)
      DIMENSION     Q(NCOORD), U(NSPEED)
      PARAMETER     (GEES = 386.2, DEGREES = 57.29577951308232)
C
      IF (WHEN .EQ. 'INITIAL') THEN
        IF (OPNFIL('Name of (optional) file to echo initial conditions',
     &             'OPTOUT', IFILE) .EQ. ' ')  RETURN
      ELSE
        IF (OPNFIL('Name of (optional) file to echo final conditions',
     &             'OPTOUT', IFILE) .EQ. ' ')  RETURN
      END IF
C
      CALL TIMDAT(TIMEDT)
C
      WRITE(IFILE, '(A)') 'PARSFILE'
      WRITE(IFILE, '(5A)')
     & '* Echo file created by:'
      WRITE(IFILE, '(5A)')
     & '* Simulation of planar tractor/semi truck.'
      WRITE(IFILE, '(5A)')
     & '* Version created by AUTOSIM 1.0 B5 on April 16, 1990.'
      WRITE(IFILE, '(5A)')
     & '* (c) The Regents of The University of Michigan, 1989, 1990.'
     &,' All rights reserved.'
      WRITE(IFILE, '(5A)')
     & ' '
```

```
      WRITE(IFILE, '(A,T8,A)') 'TITLE', TITLE
      WRITE(IFILE, '(/A,A)') '* Input File: ', INFILE
      WRITE(IFILE, '(A, A)') '* Run was made ', TIMEDT
      WRITE(IFILE, '(/A/)') '* PARAMETER VALUES'
      CALL BFECHO(IFILE)
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'CDRAG', CDRAG,
     & 'coefficient in aerodynamic drag force (lb-sec2/in2)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'CRR', CRR,
     & 'coefficient in argument to FRR in rolling resistance drag'
     &,' force (-)'
      CALL FRECHO(IFILE)
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'H1', H1,
     & 'c.g. height of tractor (in)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'H2', H2,
     & 'c.g. height of semi-trailer (in)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'HC', HC,
     & 'height of fifth-wheel connection (in)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'IPRINT', IPRINT,
     & 'number of time steps between output printing (counts)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'ITRK33', ITRK33,
     & 'moment of inertia of TRK (in-lb-s2)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'ITRL33', ITRL33,
     & 'moment of inertia of TRL (in-lb-s2)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'L1', L1,
     & 'tractor wheelbase (in)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'L1H', L1H,
     & 'distance from tractor front axle to hitch (in)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'L2', L2,
     & 'trailer wheelbase (in)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'MTRK', GEES*MTRK,
     & 'mass of TRK (lbm)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'MTRL', GEES*MTRL,
     & 'mass of TRL (lbm)'
      CALL KFECHO(IFILE)
      CALL STECHO(IFILE)
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'STEP', STEP,
     & 'simulation time step (sec)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'STOPT', STOPT,
     & 'simulation stop time (sec)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'T1', T1,
     & 'track for axle #1 (in)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'T2', T2,
     & 'track for axle #2 (in)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'T3', T3,
     & 'track for axle #3 (in)'
      CALL THECHO(IFILE)
      CALL TECHO(1, IFILE)
      CALL TECHO(2, IFILE)
      CALL TECHO(3, IFILE)
      CALL TECHO(4, IFILE)
      CALL TECHO(5, IFILE)
      CALL TECHO(6, IFILE)
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'X11', X11,
     & 'distance from tractor mass center to front axle (in)'
      WRITE(IFILE, '(A,T8,G13.6,T24,5A)') 'X2F', X2F,
     & 'distance from trailer mass center to hitch (in)'
      WRITE(IFILE, '(/A,A,A/)') '* ', WHEN, ' CONDITIONS'
```

```
      WRITE(IFILE, '(A, T8, G13.6, T24, 5A)') 'Q(1)', Q(1),
     & 'Translation of TRK0 relative to O, [n1]. (in)'
      WRITE(IFILE, '(A, T8, G13.6, T24, 5A)') 'Q(2)', Q(2),
     & 'Translation of TRK0 relative to O, [n2]. (in)'
      WRITE(IFILE, '(A, T8, G13.6, T24, 5A)') 'Q(3)', DEGREES*Q(3),
     & 'Rot. of TRK relative to N about axis #3. (deg)'
      WRITE(IFILE, '(A, T8, G13.6, T24, 5A)') 'Q(4)', DEGREES*Q(4),
     & 'Rot. of TRL relative to TRK about axis #3. (deg)'
      WRITE(IFILE, '(A, T8, G13.6, T24, 5A)') 'U(1)', U(1),
     & 'Abs. trans. speed of TRK*, axis 1. (in/s)'
      WRITE(IFILE, '(A, T8, G13.6, T24, 5A)') 'U(2)', U(2),
     & 'Abs. trans. speed of TRK*, axis 2. (in/s)'
      WRITE(IFILE, '(A, T8, G13.6, T24, 5A)') 'U(3)', DEGREES*U(3),
     & 'Abs. rot. speed of TRK, axis 3. (deg/s)'
      WRITE(IFILE, '(A, T8, G13.6, T24, 5A)') 'U(4)', DEGREES*U(4),
     & 'Rot. speed of TRL relative to TRK, axis 3. (deg/s)'
      WRITE(IFILE, '(/A)') 'END'
      CLOSE(IFILE)
      RETURN
      END


C=======================================================================
      SUBROUTINE FCT(T, Y, YP)
C=======================================================================
C  This is an interface between DIFEQN and integrators that treat the
C  state variables as a single array.
C
C  By Mike Sayers, March 1990.
C
C  --> T    real     time
C  --> Y    real     array of state variables.
C  <-- YP   real     array of derivitives.
C
      IMPLICIT NONE
      INTEGER      NCOORD
      REAL         T, Y, YP
      DIMENSION    Y(*), YP(*)
      PARAMETER    (NCOORD = 4)
C
      CALL DIFEQN(T, Y, YP, Y(NCOORD + 1), YP(NCOORD + 1))
      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE INPUT(Q, U)
C=======================================================================
C
C  This subroutine prompts the user for the name of an optional
C  parameter file for the Planar tractor/semi truck.  If a file is
C  selected, parameter values are read to override the default values.
C
C  (c) The Regents of The University of Michigan, 1989, 1990. All
C  rights reserved.
C
      IMPLICIT NONE
      LOGICAL       ISIT
      CHARACTER*80  BUFFER, ECHFIL, INFILE, OPNFIL, QUEUE, TITLE
      CHARACTER*8   CHAR8, NAMES, QC, UC
      REAL          CDRAG, CRR, DEGREES, GEES, H1, H2, HC, IPRINT,
     &              ITRK33, ITRL33, L1, L1H, L2, MTRK, MTRL, PARS,
     &              PSCALE, Q, QINIT, QSCALE, STEP, STOPT, T, T1, T2,
     &              T3, U, UINIT, USCALE, X11, X2F
      INTEGER       IFILE, ILOOP, IQUEUE, LENSTR, LSTRNG, MAXQ, NCOORD,
     &              NPARS, NQUEUE, NSPEED
C
      PARAMETER    (NPARS = 20)
      DIMENSION    PARS(NPARS)
      COMMON       /INPARS/ PARS, TITLE, INFILE
      SAVE         /INPARS/
C
      EQUIVALENCE  (PARS(1), CDRAG), (PARS(2), CRR), (PARS(3), H1),
     &             (PARS(4), H2), (PARS(5), HC), (PARS(6), IPRINT),
     &             (PARS(7), ITRK33), (PARS(8), ITRL33), (PARS(9), L1),
     &             (PARS(10), L1H), (PARS(11), L2), (PARS(12), MTRK),
     &             (PARS(13), MTRL), (PARS(14), STEP), (PARS(15),
     &             STOPT), (PARS(16), T1), (PARS(17), T2), (PARS(18),
     &             T3), (PARS(19), X11), (PARS(20), X2F)
C
      PARAMETER    (GEES = 386.2, DEGREES = 57.29577951308232)
      PARAMETER    (NCOORD = 4, NSPEED = 4, MAXQ = 20, IFILE = 1)
      DIMENSION    NAMES(NPARS), Q(NCOORD), QC(NCOORD), QINIT(NCOORD),
     &             QUEUE(MAXQ), QSCALE(NCOORD), U(NSPEED), UC(NSPEED),
     &             UINIT(NSPEED), USCALE(NSPEED)
      DATA         QC /'Q(1)', 'Q(2)', 'Q(3)', 'Q(4)'/
      DATA         UC /'U(1)', 'U(2)', 'U(3)', 'U(4)'/
      DATA         QSCALE /1, 1, DEGREES, DEGREES/
      DATA         QINIT /0.0, 0.0, 0.0, 0.0/
      DATA         USCALE /1, 1, DEGREES, DEGREES/
      DATA         UINIT /0.0, 0.0, 0.0, 0.0/
      DATA         NAMES /'CDRAG', 'CRR', 'H1', 'H2', 'HC', 'IPRINT',
     &             'ITRK33', 'ITRL33', 'L1', 'L1H', 'L2', 'MTRK',
     &             'MTRL', 'STEP', 'STOPT', 'T1', 'T2', 'T3', 'X11',
     &             'X2F'/
      NQUEUE = 0
      IQUEUE = 0
C
```

```
C  Open file with parameter values and initial conditions
C
    5 INFILE = OPNFIL ('Name of (optional) file with parameter values',
     &         'OPTIN', IFILE)
    6 IF (INFILE .NE. ' ') THEN
        READ(IFILE, '(A)') CHAR8
C
        IF (CHAR8 .EQ. 'END') GO TO 100
        IF (CHAR8 .NE. 'PARSFILE') THEN
          CLOSE(IFILE)
          WRITE (*, '(A)') ' Error--File did not begin with "PARSFILE"'
          IF (IQUEUE .EQ. 0) THEN
            GO TO 5
          ELSE
            GO TO 100
          END IF
        END IF
C
C
C  Read line from file.  CHAR8 is the keyword, checked for:
C   o  TITLE keyword,
C   o  parameter keyword (from NAMES array),
C   o  initial value of generalized coordinate (keyword from QC array),
C   o  initial value of generalized speed (keyword from UC array),
C   o  (possibly) keyword for other input subroutine, or
C   o  END keyword.
C  All other lines are ignored.  Also, all lines are ignored after END
C  is found, and any line  with a '*' in column 1 is ignored.
C
C
   10   READ(IFILE, '(A8,A80)', END=100, ERR=100) CHAR8, BUFFER
        IF (CHAR8 .EQ. 'TITLE') THEN
          TITLE = BUFFER
          GO TO 10
        ELSE IF (CHAR8 .EQ. 'END') THEN
          GO TO 100
        ELSE IF (CHAR8(1:1) .EQ. '*') THEN
          GO TO 10
        ELSE IF (CHAR8 .EQ. 'PARSFILE') THEN
          INQUIRE (FILE=BUFFER, EXIST=ISIT)
          IF (ISIT) THEN
            NQUEUE = NQUEUE + 1
            QUEUE (NQUEUE) = BUFFER
          ELSE
            LSTRNG = LENSTR (BUFFER)
            WRITE (*,'(A,A,A)') 'Error--PARSFILE "', BUFFER(:LSTRNG),
     &                          '"not found (skipped).'
          END IF
          GO TO 10
        ELSE IF (CHAR8 .EQ. 'TIRE1') THEN
          CALL TREAD(1, IFILE)
          GO TO 10
        ELSE IF (CHAR8 .EQ. 'TIRE2') THEN
          CALL TREAD(2, IFILE)
          GO TO 10
        ELSE IF (CHAR8 .EQ. 'TIRE3') THEN
          CALL TREAD(3, IFILE)
          GO TO 10
```

```
        ELSE IF (CHAR8 .EQ. 'TIRE4') THEN
          CALL TREAD(4, IFILE)
          GO TO 10
        ELSE IF (CHAR8 .EQ. 'TIRE5') THEN
          CALL TREAD(5, IFILE)
          GO TO 10
        ELSE IF (CHAR8 .EQ. 'TIRE6') THEN
          CALL TREAD(6, IFILE)
          GO TO 10
        ELSE IF (CHAR8 .EQ. 'STEER') THEN
          CALL STREAD(IFILE)
          GO TO 10
        ELSE IF (CHAR8 .EQ. 'THROTTLE') THEN
          CALL THREAD(IFILE)
          GO TO 10
        ELSE IF (CHAR8 .EQ. 'BRAKE') THEN
          CALL BFREAD(IFILE)
          GO TO 10
        ELSE IF (CHAR8 .EQ. 'FRICTION') THEN
          CALL FRREAD(IFILE)
          GO TO 10
        ELSE IF (CHAR8 .EQ. 'ROL&PTCH') THEN
          CALL KFREAD(IFILE)
          GO TO 10
        END IF
C
C  Check for names of parameters
C
        DO 20 ILOOP = 1, NPARS
          IF (CHAR8 .EQ. NAMES(ILOOP)) THEN
            READ(BUFFER, '(G13.0)') PARS(ILOOP)
            GO TO 10
          END IF
   20   CONTINUE
C
C  Check for names of generalized coordinates (initial conditions)
C
        DO 30 ILOOP = 1, NCOORD
          IF (CHAR8 .EQ. QC(ILOOP)) THEN
            READ(BUFFER, '(G13.0)') QINIT(ILOOP)
            GO TO 10
          END IF
   30   CONTINUE
C
C  Check for names of generalized speeds (initial conditions)
C
        DO 40 ILOOP = 1, NSPEED
          IF (CHAR8 .EQ. UC(ILOOP)) THEN
            READ(BUFFER, '(G13.0)') UINIT(ILOOP)
            GO TO 10
          END IF
   40   CONTINUE
C
        GO TO 10
      END IF
C
```

```
C  Close this file and process other PARS files that were referenced.
C
  100 CLOSE (IFILE)
      IF (IQUEUE .LT. NQUEUE) THEN
        IQUEUE = IQUEUE + 1
        INFILE = QUEUE (IQUEUE)
        OPEN (IFILE, STATUS='OLD', FILE=INFILE)
        WRITE (*, '(A,A)') ' Reading from PARSFILE ', INFILE
        GO TO 6
      END IF
C
C  Set initial conditions
C
      DO 110 ILOOP = 1, NCOORD
  110 Q(ILOOP) = QINIT(ILOOP) / QSCALE(ILOOP)
C
      DO 120 ILOOP = 1, NSPEED
  120 U(ILOOP) = UINIT(ILOOP) / USCALE(ILOOP)
C
C  Convert units as needed.
C
      MTRK = MTRK/GEES
      MTRL = MTRL/GEES
C
C  Let external subroutines initialize.
C
      CALL TINIT(1, 0.5*GEES*(MTRK -(X11*MTRK + L1H*(1
     &      -X2F/L2)*MTRL)/L1 + (1 -X2F/L2)*MTRL))
      CALL TINIT(2, 0.5*GEES*(MTRK -(X11*MTRK + L1H*(1
     &      -X2F/L2)*MTRL)/L1 + (1 -X2F/L2)*MTRL))
      CALL TINIT(3, 0.5*GEES*(X11*MTRK + L1H*(1 -X2F/L2)*MTRL)/L1)
      CALL TINIT(4, 0.5*GEES*(X11*MTRK + L1H*(1 -X2F/L2)*MTRL)/L1)
      CALL TINIT(5, 0.5*X2F*GEES*MTRL/L2)
      CALL TINIT(6, 0.5*X2F*GEES*MTRL/L2)
C
C  Initialize control inputs to 0
C
      CALL STRSYS(0.0)
      CALL BRAKE(0.0, 0.5*GEES*(MTRK -(X11*MTRK + L1H*(1
     &      -X2F/L2)*MTRL)/L1 + (1 -X2F/L2)*MTRL), 0.5*GEES*(MTRK
     &      -(X11*MTRK + L1H*(1 -X2F/L2)*MTRL)/L1 + (1 -X2F/L2)*MTRL),
     &      0.5*GEES*(X11*MTRK + L1H*(1 -X2F/L2)*MTRL)/L1, 0.5*GEES
     &     *(X11*MTRK + L1H*(1 -X2F/L2)*MTRL)/L1, 0.5*X2F*GEES*MTRL/L2,
     &      0.5*X2F*GEES*MTRL/L2, T)
      CALL ACCEL(0.0, U(1))
      CALL INROP(0.0, 0.0, U(1))
      CALL INVEL(U(1), TITLE, STOPT, 1.0)
C
      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE INTEQS(T, Y, YP, N, STEP)
C=======================================================================
C  This is a RK2 numerical integration subroutine.
C  FCT is a subroutine that computes the derivatives YP.
C  By Mike Sayers, March 1990.
C  --> T    real     time
C  <-> Y    real     array of state variables.  As input, these have
C                    values for t=T.  As output, they are for t=T+STEP
C  <-> YP   real     array of derivitives. These must be given values
C                    for t=T, and will be modified.  The returned
C                    values are NOT at T + STEP.
C  --> N    integer  number of state variables
C  --> STEP real     time step
      IMPLICIT NONE
      INTEGER    I, N
      REAL       T, Y, YP, DT, YM, STEP, STEP2
      DIMENSION  Y(*), YP(*), YM(100)
      STEP2 = STEP * .5
      DO 10 I = 1, N
        YM(I) = Y(I) + STEP2 * YP(I)
   10 CONTINUE
      CALL FCT(T + STEP2, YM, YP)
      DO 20 I = 1, N
        Y(I) = Y(I) + STEP * YP(I)
   20 CONTINUE
      RETURN
      END
C=======================================================================
      FUNCTION LENSTR (STRING)
C=======================================================================
C  count characters in left-justified string.   M. Sayers, 8-9-87
      CHARACTER*(*) STRING
      N = LEN (STRING)
      DO 10 L = N, 1, -1
        IF (STRING(L:L) .NE. ' ') THEN
          LENSTR = L
          RETURN
        END IF
   10 CONTINUE
      LENSTR = 1
      RETURN
      END
C=======================================================================
      FUNCTION NORMA(A)
C=======================================================================
C  normalize angle
      REAL A, NORMA, PI
      PARAMETER (PI=3.141592653589793)
      IF (A .GE. 4.*PI) THEN
        NORMA = A - 6.*PI
      ELSE IF (A .LE. -4.*PI) THEN
        NORMA = A + 6.*PI
      ELSE
        NORMA = A
      END IF
      RETURN
      END
```

```
C=======================================================================
      FUNCTION OPNFIL (PROMPT, STAT, IFILE)
C=======================================================================
C  This function tries to get a file name from the user and open the
C  file.
C
C  --> PROMPT string   Message to prompt user
C  --> STAT   string   Status of file ("NEW"    = mandatory output,
C                                     "OLD"    = mandatory input,
C                                     "OPTIN"  = optional input,
C                                     "OPTOUT" = optional output)
C  --> IFILE  integer  Fortran I/O unit for file
C  <-- OPNFIL string   name of file opened or " " if no file selected
C M. Sayers January 30, 1989
C
      LOGICAL       ISIT
      CHARACTER*(*) PROMPT, STAT, OPNFIL
      CHARACTER*3   STAT2
      INTEGER       IFILE, L, LENSTR
C  Set Fortran STATUS type
C
      IF (STAT .EQ. 'NEW' .OR. STAT .EQ. 'OPTOUT') THEN
        STAT2 = 'NEW'
      ELSE
        STAT2 = 'OLD'
      END IF
C  Ask user for file name, and check for no response (blank line)
C
  100 WRITE(*, '(A, A, A\)') ' ', PROMPT, ': '
      READ(*, '(A)') OPNFIL
      IF (OPNFIL .EQ. ' ') THEN
        IF (STAT .EQ. 'OPTIN' .OR. STAT .EQ. 'OPTOUT') THEN
          RETURN
        ELSE IF (STAT .EQ. 'NEW') THEN
          WRITE (*,'(A)') ' Output file is required!'
          GO TO 100
        ELSE
          WRITE (*,'(A)') ' Input file is required!'
          GO TO 100
        END IF
      END IF
C  Deal with existance of file (or lack thereof)
C
      INQUIRE (FILE=OPNFIL, EXIST=ISIT)
      IF ((.NOT. ISIT) .AND. (STAT2 .EQ. 'OLD')) THEN
        L = LENSTR(OPNFIL)
        WRITE (*, '(A, A, A)') ' File "', OPNFIL(:L),
     &                 '" does not exist.  Try again.'
        GO TO 100
      ELSE IF (ISIT .AND. STAT2 .EQ. 'NEW') THEN
        OPEN (IFILE, FILE=OPNFIL)
        CLOSE (IFILE, STATUS='DELETE')
      END IF
C  Open file and write blank line on screen
      OPEN(IFILE, STATUS=STAT2, FILE=OPNFIL)
      WRITE (*,'(A)') ' '
      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE OPNOUT(IFILE)
C=======================================================================
C  This subroutine prompts the user for the name of a file set that
C  will be created to store time histories of the 45 output variables
C  computed by the Planar tractor/semi truck simulation program.
C  A text file is created and opened, and labeling information is
C  written to facilitate post-processing of the data.  Then, the text
C  file is closed and a corresponding binary file is created and opened
C  to store the numerical values of the output variables.
C
C  (c) The Regents of The University of Michigan, 1989, 1990. All
C  rights reserved.
C
      IMPLICIT NONE
      CHARACTER*80  FNOUT, INFILE, OPNFIL, TITLE
      LOGICAL       ISIT
      REAL          CDRAG, CRR, DEGREES, GEES, H1, H2, HC, IPRINT,
     &              ITRK33, ITRL33, L1, L1H, L2, MTRK, MTRL, PARS,
     &              STEP, STOPT, T, T1, T2, T3, X11, X2F
      INTEGER       IFILE, ILOOP, IPRNT2, LENSTR, LSTRNG, MAXBUF,
     &              NBYTES, NCHAN, NCOORD, NPARS, NRECS, NSAMP, NSCAN,
     &              NSPEED, NUMKEY, NVARS
      CHARACTER*32  GENNAM, LONGNM, RIGBOD
      CHARACTER*24  TIMEDT
      CHARACTER*8   CHAR8, SHORTN, UNITSN
C
      PARAMETER    (NPARS = 20)
      DIMENSION    PARS(NPARS)
      COMMON       /INPARS/ PARS, TITLE, INFILE
      SAVE         /INPARS/
      EQUIVALENCE (PARS(14), STEP), (PARS(15), STOPT)
C
      PARAMETER    (NVARS = 45, NUMKEY = 1)
      DIMENSION    LONGNM(NVARS), GENNAM(NVARS), RIGBOD(NVARS),
     &             SHORTN(NVARS), UNITSN(NVARS)
C
C  Prompt user to provide name of output file.  File is opened and
C  attached to Fortran unit IFILE.
C
      FNOUT = OPNFIL('Name of (required) file for time history outputs',
     &    'NEW', IFILE)
      NCHAN = 0
      IPRNT2 = PARS(6)
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'slip angle for Tire 1'
      SHORTN(NCHAN) = 'alpha 1'
      GENNAM(NCHAN) = 'Slip Angle'
      UNITSN(NCHAN) = 'deg'
      RIGBOD(NCHAN) = 'Tire 1'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'slip angle for Tire 2'
      SHORTN(NCHAN) = 'alpha 2'
      GENNAM(NCHAN) = 'Slip Angle'
      UNITSN(NCHAN) = 'deg'
      RIGBOD(NCHAN) = 'Tire 2'
```

```
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'slip angle for Tire 3'
      SHORTN(NCHAN) = 'alpha 3'
      GENNAM(NCHAN) = 'Slip Angle'
      UNITSN(NCHAN) = 'deg'
      RIGBOD(NCHAN) = 'Tire 3'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'slip angle for Tire 4'
      SHORTN(NCHAN) = 'alpha 4'
      GENNAM(NCHAN) = 'Slip Angle'
      UNITSN(NCHAN) = 'deg'
      RIGBOD(NCHAN) = 'Tire 4'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'slip angle for Tire 5'
      SHORTN(NCHAN) = 'alpha 5'
      GENNAM(NCHAN) = 'Slip Angle'
      UNITSN(NCHAN) = 'deg'
      RIGBOD(NCHAN) = 'Tire 5'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'slip angle for Tire 6'
      SHORTN(NCHAN) = 'alpha 6'
      GENNAM(NCHAN) = 'Slip Angle'
      UNITSN(NCHAN) = 'deg'
      RIGBOD(NCHAN) = 'Tire 6'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Trans. of TRK0 rel. to O, [n1]'
      SHORTN(NCHAN) = 'Q(1)'
      GENNAM(NCHAN) = 'Translation'
      UNITSN(NCHAN) = 'in'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Trans. of TRK0 rel. to O, [n2]'
      SHORTN(NCHAN) = 'Q(2)'
      GENNAM(NCHAN) = 'Translation'
      UNITSN(NCHAN) = 'in'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Rot. of TRK rel. to N, axis #3'
      SHORTN(NCHAN) = 'Q(3)'
      GENNAM(NCHAN) = 'Rotation'
      UNITSN(NCHAN) = 'deg'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Rot. of TRL rel. to TRK, axis #3'
      SHORTN(NCHAN) = 'Q(4)'
      GENNAM(NCHAN) = 'Rotation'
      UNITSN(NCHAN) = 'deg'
      RIGBOD(NCHAN) = 'Semitrailer'
C
```

```
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Abs. trans. speed of TRKCM, [trk1]'
      SHORTN(NCHAN) = 'U(1)'
      GENNAM(NCHAN) = 'Speed'
      UNITSN(NCHAN) = 'in/s'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Abs. trans. speed of TRKCM, [trk2]'
      SHORTN(NCHAN) = 'U(2)'
      GENNAM(NCHAN) = 'Speed'
      UNITSN(NCHAN) = 'in/s'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Abs. rot. speed of TRK, axis 3'
      SHORTN(NCHAN) = 'U(3)'
      GENNAM(NCHAN) = 'Angular speed'
      UNITSN(NCHAN) = 'deg/s'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Rot. speed of TRL rel. to TRK'
      SHORTN(NCHAN) = 'U(4)'
      GENNAM(NCHAN) = 'Angular speed'
      UNITSN(NCHAN) = 'deg/s'
      RIGBOD(NCHAN) = 'Semitrailer'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Abs. trans. accel. of TRKCM, [trk1]'
      SHORTN(NCHAN) = 'UP(1)'
      GENNAM(NCHAN) = 'Acceleration'
      UNITSN(NCHAN) = 'g''s'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Abs. trans. accel. of TRKCM, [trk2]'
      SHORTN(NCHAN) = 'UP(2)'
      GENNAM(NCHAN) = 'Acceleration'
      UNITSN(NCHAN) = 'g''s'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Abs. rot. accel. of TRK, axis 3'
      SHORTN(NCHAN) = 'UP(3)'
      GENNAM(NCHAN) = 'Rotational acceleration'
      UNITSN(NCHAN) = 'deg/s2'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Rot. accel. of TRL rel. to TRK'
      SHORTN(NCHAN) = 'UP(4)'
      GENNAM(NCHAN) = 'Rotational acceleration'
      UNITSN(NCHAN) = 'deg/s2'
      RIGBOD(NCHAN) = 'Semitrailer'
C
```

```
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'lateral force for Tire 1'
      SHORTN(NCHAN) = 'FY1'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'long. force for Tire 1'
      SHORTN(NCHAN) = 'FX1'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'lateral force for Tire 2'
      SHORTN(NCHAN) = 'FY2'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'long. force for Tire 2'
      SHORTN(NCHAN) = 'FX2'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'lateral force for Tire 3'
      SHORTN(NCHAN) = 'FY3'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'long. force for Tire 3'
      SHORTN(NCHAN) = 'FX3'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'lateral force for Tire 4'
      SHORTN(NCHAN) = 'FY4'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'long. force for Tire 4'
      SHORTN(NCHAN) = 'FX4'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Tractor'
C
```

```
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'lateral force for Tire 5'
      SHORTN(NCHAN) = 'FY5'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Semitrailer'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'long. force for Tire 5'
      SHORTN(NCHAN) = 'FX5'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Semitrailer'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'lateral force for Tire 6'
      SHORTN(NCHAN) = 'FY6'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Semitrailer'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'long. force for Tire 6'
      SHORTN(NCHAN) = 'FX6'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Semitrailer'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'aerodynamic drag force'
      SHORTN(NCHAN) = 'FAERO'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'rolling resistance drag force'
      SHORTN(NCHAN) = 'FRR'
      GENNAM(NCHAN) = 'Force'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'Tractor'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Steer angle of front wheels'
      SHORTN(NCHAN) = 'Steer'
      GENNAM(NCHAN) = 'Steer angle'
      UNITSN(NCHAN) = 'deg'
      RIGBOD(NCHAN) = 'Input'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Roll angle of truck&trailer'
      SHORTN(NCHAN) = 'Roll'
      GENNAM(NCHAN) = 'Roll angle'
      UNITSN(NCHAN) = 'deg'
      RIGBOD(NCHAN) = 'trk'
C
```

```
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Pitch angle of the truck'
      SHORTN(NCHAN) = 'Ptch-trk'
      GENNAM(NCHAN) = 'Truck pitch'
      UNITSN(NCHAN) = 'deg'
      RIGBOD(NCHAN) = 'trk'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Lateral acceleration of trailer'
      SHORTN(NCHAN) = 'Aytr'
      GENNAM(NCHAN) = 'Lateral acc.'
      UNITSN(NCHAN) = 'g''s'
      RIGBOD(NCHAN) = 'trl'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Lateral acceleration of truck'
      SHORTN(NCHAN) = 'Ayt'
      GENNAM(NCHAN) = 'Lateral acc.'
      UNITSN(NCHAN) = 'g''s'
      RIGBOD(NCHAN) = 'trk'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Longitudinal acceleration of trailer'
      SHORTN(NCHAN) = 'Axtr'
      GENNAM(NCHAN) = 'Long. acc.'
      UNITSN(NCHAN) = 'g''s'
      RIGBOD(NCHAN) = 'trl'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Longitudinal acceleration of truck'
      SHORTN(NCHAN) = 'Axt'
      GENNAM(NCHAN) = 'Long. acc.'
      UNITSN(NCHAN) = 'g''s'
      RIGBOD(NCHAN) = 'trk'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Truck front/left wheel load'
      SHORTN(NCHAN) = 'FZ1'
      GENNAM(NCHAN) = 'Wheel load'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'trk'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Truck front/right wheel load'
      SHORTN(NCHAN) = 'FZ2'
      GENNAM(NCHAN) = 'Wheel load'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'trk'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Truck rear/left wheel load'
      SHORTN(NCHAN) = 'FZ3'
      GENNAM(NCHAN) = 'Wheel load'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'trk'
C
```

```
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Truck rear/right wheel load'
      SHORTN(NCHAN) = 'FZ4'
      GENNAM(NCHAN) = 'Wheel load'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'trk'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Trailer left wheel load'
      SHORTN(NCHAN) = 'FZ5'
      GENNAM(NCHAN) = 'Wheel load'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'trl'
C
      NCHAN = NCHAN + 1
      LONGNM(NCHAN) = 'Trailer right wheel load'
      SHORTN(NCHAN) = 'FZ6'
      GENNAM(NCHAN) = 'Wheel load'
      UNITSN(NCHAN) = 'lb'
      RIGBOD(NCHAN) = 'trl'
C
C   Write Header Info for ERD file
C
C  Set parameters needed to write header for ERD format file
C  NUMKEY = 1 for 32-bit floating-point binary
C  NSAMP  = number of samples
C  NRECS  = number of "records" in output file
C  NBYTES = number of bytes/record
C
      NSAMP = STOPT / STEP / IPRNT2 + 1
      NBYTES = 4 * NCHAN
      NRECS = NSAMP
C
C  Write standard ERD file heading.
C
      WRITE(IFILE, '(A)') 'ERDFILEV2.00'
      WRITE(IFILE, 100) NCHAN, NSAMP, NRECS, NBYTES, NUMKEY, STEP*IPRNT2
      WRITE(IFILE, '(A,A)') 'TITLE    ', TITLE
      WRITE(IFILE, 110) 'SHORTNAM', (SHORTN(ILOOP), ILOOP=1, NCHAN)
      WRITE(IFILE, 120) 'LONGNAME', (LONGNM(ILOOP), ILOOP=1, NCHAN)
      WRITE(IFILE, 110) 'UNITSNAM', (UNITSN(ILOOP), ILOOP=1, NCHAN)
      WRITE(IFILE, 120) 'GENNAME ', (GENNAM(ILOOP), ILOOP=1, NCHAN)
      WRITE(IFILE, 120) 'RIGIBODY', (RIGBOD(ILOOP), ILOOP=1, NCHAN)
      WRITE(IFILE, '(A)') 'XLABEL  Time'
      WRITE(IFILE, '(A)') 'XUNITS  sec'
C
      IF (INFILE .EQ. ' ') THEN
        WRITE(IFILE, '(A)') 'HISTORY No input file (used defaults)'
      ELSE
        WRITE(IFILE, '(A, A)') 'HISTORY Input parameter file was ',
     &        INFILE
      END IF
      CALL TIMDAT(TIMEDT)
      WRITE(IFILE, '(A,A)')
     & 'HISTORY Data generated with Planar tractor/semi truck at '
     &, TIMEDT
      WRITE(IFILE, '(A)') 'END'
C
```

```
C  Close (text) header and create binary file.
C
      CLOSE(IFILE)
      LSTRNG = LENSTR(FNOUT)
      FNOUT = FNOUT (:LSTRNG) // '.BIN'
      INQUIRE(FILE=FNOUT, EXIST=ISIT)
      IF (ISIT) THEN
        OPEN (IFILE, FILE=FNOUT)
        CLOSE (IFILE, STATUS='DELETE')
      END IF
C
      OPEN(IFILE, FILE=FNOUT, STATUS='NEW', ACCESS='SEQUENTIAL',
     &      FORM='UNFORMATTED')
C
  100 FORMAT (5(I6,','),G13.6)
  110 FORMAT (A8, 45A8)
  120 FORMAT (A8, 31A32, 1(/'&1000   ', 31A32))
      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE OUTPUT(IFILE, T, Q, QP, U, UP)
C=======================================================================
C
C  --> IFILE integer Fortran i/o unit for output
C  --> T     real    time
C  --> Q     real    array of 4 generalized coordinates
C  --> QP    real    array of derivitives of Q
C  --> U     real    array of 4 generalized speeds
C  --> UP    real    array of derivatives of U
C
C  This subroutine writes the values of the 45 output variables
C  computed by the Planar tractor/semi truck simulation program into an
C  output file, using the values at time T.
C
C  (c) The Regents of The University of Michigan, 1989, 1990. All
C  rights reserved.
C
      IMPLICIT NONE
      CHARACTER*80  INFILE, TITLE
      REAL          C, CDRAG, CRR, DEGREES, FA, FB1, FB2, FB3, FORCEM,
     &              FX1, FX2, FX3, FX4, FX5, FX6, FY1, FY2, FY3, FY4,
     &              FY5, FY6, FZ1, FZ2, FZ3, FZ4, FZ5, FZ6, GEES, H1,
     &              H2, HC, IPRINT, ITRK33, ITRL33, L1, L1H, L2, MTRK,
     &              MTRL, OUTBUF, PARS, PC, PHEE, Q, QP, S, STEER,
     &              STEP, STOPT, T, T1, T2, T3, THETA, U, UP, X11, X2F,
     &              Z
      INTEGER       IFILE, ILOOP, NCOORD, NPARS, NSPEED, NVARS
C
      PARAMETER     (NCOORD = 4, NSPEED = 4, NVARS = 45)
      DIMENSION     Q(NCOORD), QP(NCOORD), U(NSPEED), UP(NSPEED),
     &              OUTBUF(NVARS)
      DIMENSION     PC(34)
      COMMON        /PRCMP/ PC
      SAVE          /PRCMP/
C
      DIMENSION     C(4), FORCEM(2), S(4), Z(86)
      COMMON        /DYVARS/ C, FORCEM, S, Z, STEER, FA, FB1, FB2, FB3,
     &              FZ1, FZ2, FZ3, FZ4, FZ5, FZ6, PHEE, THETA, FY1, FX1,
     &              FY2, FX2, FY3, FX3, FY4, FX4, FY5, FX5, FY6, FX6
      SAVE          /DYVARS/
C
      PARAMETER     (NPARS = 20)
      DIMENSION     PARS(NPARS)
      COMMON        /INPARS/ PARS, TITLE, INFILE
      SAVE          /INPARS/
C
      EQUIVALENCE   (PARS(1), CDRAG), (PARS(2), CRR), (PARS(3), H1),
     &              (PARS(4), H2), (PARS(5), HC), (PARS(6), IPRINT),
     &              (PARS(7), ITRK33), (PARS(8), ITRL33), (PARS(9), L1),
     &              (PARS(10), L1H), (PARS(11), L2), (PARS(12), MTRK),
     &              (PARS(13), MTRL), (PARS(14), STEP), (PARS(15),
     &              STOPT), (PARS(16), T1), (PARS(17), T2), (PARS(18),
     &              T3), (PARS(19), X11), (PARS(20), X2F)
      PARAMETER     (GEES = 386.2, DEGREES = 57.29577951308232)
C

C
```

```
C  fill buffer with output variables.
C
      OUTBUF(1) = DEGREES*Z(9)
      OUTBUF(2) = DEGREES*Z(11)
      OUTBUF(3) = DEGREES*Z(17)
      OUTBUF(4) = DEGREES*Z(19)
      OUTBUF(5) = DEGREES*Z(35)
      OUTBUF(6) = DEGREES*Z(36)
      OUTBUF(7) = Q(1)
      OUTBUF(8) = Q(2)
      OUTBUF(9) = DEGREES*Q(3)
      OUTBUF(10) = DEGREES*Q(4)
      OUTBUF(11) = U(1)
      OUTBUF(12) = U(2)
      OUTBUF(13) = DEGREES*U(3)
      OUTBUF(14) = DEGREES*U(4)
      OUTBUF(15) = UP(1)/GEES
      OUTBUF(16) = UP(2)/GEES
      OUTBUF(17) = DEGREES*UP(3)
      OUTBUF(18) = DEGREES*UP(4)
      OUTBUF(19) = FY1
      OUTBUF(20) = FX1
      OUTBUF(21) = FY2
      OUTBUF(22) = FX2
      OUTBUF(23) = FY3
      OUTBUF(24) = FX3
      OUTBUF(25) = FY4
      OUTBUF(26) = FX4
      OUTBUF(27) = FY5
      OUTBUF(28) = FX5
      OUTBUF(29) = FY6
      OUTBUF(30) = FX6
      OUTBUF(31) = FORCEM(1)
      OUTBUF(32) = FORCEM(2)
      OUTBUF(33) = DEGREES*STEER
      OUTBUF(34) = DEGREES*PHEE
      OUTBUF(35) = DEGREES*THETA
      OUTBUF(36) = Z(82)/GEES
      OUTBUF(37) = Z(79)/GEES
      OUTBUF(38) = Z(84)/GEES
      OUTBUF(39) = Z(83)/GEES
      OUTBUF(40) = FZ1
      OUTBUF(41) = FZ2
      OUTBUF(42) = FZ3
      OUTBUF(43) = FZ4
      OUTBUF(44) = FZ5
      OUTBUF(45) = FZ6
C
C  The following line writes to an unformatted binary file
      WRITE (IFILE) (OUTBUF(ILOOP), ILOOP=1, NVARS)
C
C--The next 3 lines are for the Macintosh
C
      IF (T .EQ. 0.) WRITE (*, '(/A/7X,A)') ' Progress:','sec'
      CALL TOOLBX (Z'89409000', 0, -11)
      WRITE (*, '(F6.2)') T
      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE PRECMP
C=======================================================================
C  This subroutine defines all constants that can be pre-computed for
C  the Planar tractor/semi truck.  The constants are put into the
C  COMMON block /PRECMP/
C
C  (c) The Regents of The University of Michigan, 1989, 1990. All
C  rights reserved.
C
      IMPLICIT NONE
      CHARACTER*80  INFILE, TITLE
      REAL          CDRAG, CRR, DEGREES, GEES, H1, H2, HC, IPRINT,
     &              ITRK33, ITRL33, L1, L1H, L2, MTRK, MTRL, PARS, PC,
     &              STEP, STOPT, T1, T2, T3, X11, X2F
      INTEGER       NPARS
C
      DIMENSION     PC(34)
      COMMON        /PRCMP/ PC
      SAVE          /PRCMP/
C
      PARAMETER     (NPARS = 20)
      DIMENSION     PARS(NPARS)
      COMMON        /INPARS/ PARS, TITLE, INFILE
      SAVE          /INPARS/
C
      EQUIVALENCE   (PARS(1), CDRAG), (PARS(2), CRR), (PARS(3), H1),
     &              (PARS(4), H2), (PARS(5), HC), (PARS(6), IPRINT),
     &              (PARS(7), ITRK33), (PARS(8), ITRL33), (PARS(9), L1),
     &              (PARS(10), L1H), (PARS(11), L2), (PARS(12), MTRK),
     &              (PARS(13), MTRL), (PARS(14), STEP), (PARS(15),
     &              STOPT), (PARS(16), T1), (PARS(17), T2), (PARS(18),
     &              T3), (PARS(19), X11), (PARS(20), X2F)
      PARAMETER     (GEES = 386.2, DEGREES = 57.29577951308232)
C
      PC(1) = 0.5*T1
      PC(2) = 0.5*T2
      PC(3) = 0.5*T3
      PC(4) = (L1 -X11)
      PC(5) = (X11 -L1H)
      PC(6) = (MTRK + MTRL)
      PC(7) = CRR*GEES*(MTRK + MTRL)
      PC(8) = (L2 -X2F)
      PC(9) = X2F*MTRL
      PC(10) = (ITRK33 + ITRL33)
      PC(11) = (MTRL*X2F**2 + ITRL33)
      PC(12) = 1.0/PC(6)
      PC(13) = GEES*PC(6)
      PC(14) = X2F*GEES
      PC(15) = (H2 -HC)
      PC(16) = 1.0/L2
      PC(17) = X11*GEES
      PC(18) = GEES*MTRL
      PC(19) = 1.0/L1
      PC(20) = X2F**2
      PC(21) = MTRL*PC(14)
      PC(22) = MTRL*PC(15)
      PC(23) = PC(16)*PC(21)
```

```
      PC(24) = HC*PC(16)
      PC(25) = PC(16)*PC(22)
      PC(26) = MTRK*PC(17)
      PC(27) = H1*MTRK
      PC(28) = L1H*PC(18)
      PC(29) = (PC(26) + PC(28))
      PC(30) = PC(19)*PC(29)
      PC(31) = PC(19)*PC(27)
      PC(32) = L1H*PC(19)
      PC(33) = HC*PC(19)
      PC(34) = HC*MTRL*PC(19)
      RETURN
      END
```

```
      SUBROUTINE TIMDAT (TIMEDT)
```

```
C  Get date and time.  On the Mac, this requires the TIME and DATE
C  subroutines from Absoft.
C
C  by M. Sayers, 1986.
C
C  <-- TIMEDT char*24   string containing time & date.
C
      CHARACTER*24 TIMEDT
      CHARACTER*36 MONTHS
      INTEGER*4 M, IDAY, IYEAR, ISEC
      INTEGER*2 YEAR, MONTH, DAY, HOUR, MIN, SEC, I100
      MONTHS = 'JanFebMarAprMayJunJulAugSepOctNovDec'

C--The following 4 lines are for the IBM PC (using Microsoft
C--time and date functions)
*      CALL GETDAT (YEAR, MONTH, DAY)
*      CALL GETTIM (IHOUR, MIN, SEC, I100)
*      WRITE (TIMEDT, 100) IHOUR, MIN, MONTHS (MONTH*3-2:MONTH*3),
*     &       DAY, YEAR

C--get time for MTS version
C      CALL TIME(22, 0, TIMEDT)

C--The following 5 lines are for the Apple Mac
C--(using Absoft time & date functions)
      CALL DATE (M, IDAY, IYEAR)
      CALL TIME (ISEC)
      WRITE (TIMEDT, 100)
     &       ISEC/3600, MOD (ISEC, 3600) / 60, MONTHS (M*3-2:M*3),
     &       IDAY, 1900 + IYEAR

  100 FORMAT (I2,':',I2.2,' on ',A3,I3,',',I5)
      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE UPDATE(T, Q, QP, U, UP)
C=======================================================================
C  This subroutine performs "update" operations once per time step for
C  the Planar tractor/semi truck.
C  (c) The Regents of The University of Michigan, 1989, 1990. All
C  rights reserved.
C
      IMPLICIT NONE
      CHARACTER*80  INFILE, TITLE
      REAL          BPF, C, CDRAG, CHKAX, CRR, DEGREES, FA, FB1, FB2,
     &              FB3, FORCEM, FX1, FX2, FX3, FX4, FX5, FX6, FY1,
     &              FY2, FY3, FY4, FY5, FY6, FZ1, FZ2, FZ3, FZ4, FZ5,
     &              FZ6, GEES, H1, H2, HC, IPRINT, ITRK33, ITRL33, L1,
     &              L1H, L2, MTRK, MTRL, NORMA, PARS, PC, PHEE, Q, QP,
     &              S, STEER, STEP, STOPT, SWA, T, T1, T2, T3, THETA,
     &              THROT, U, UP, X11, X2F, Z
      INTEGER       NCOORD, NPARS, NSPEED
C
      PARAMETER     (NCOORD = 4, NSPEED = 4)
      DIMENSION     Q(NCOORD), QP(NCOORD), U(NSPEED), UP(NSPEED)
      DIMENSION     C(4), FORCEM(2), S(4), Z(86)
      COMMON        /DYVARS/ C, FORCEM, S, Z, STEER, FA, FB1, FB2, FB3,
     &              FZ1, FZ2, FZ3, FZ4, FZ5, FZ6, PHEE, THETA, FY1, FX1,
     &              FY2, FX2, FY3, FX3, FY4, FX4, FY5, FX5, FY6, FX6
      SAVE          /DYVARS/
C
      PARAMETER     (NPARS = 20)
      DIMENSION     PARS(NPARS)
      COMMON        /INPARS/ PARS, TITLE, INFILE
      SAVE          /INPARS/
C
      EQUIVALENCE   (PARS(1), CDRAG), (PARS(2), CRR), (PARS(3), H1),
     &              (PARS(4), H2), (PARS(5), HC), (PARS(6), IPRINT),
     &              (PARS(7), ITRK33), (PARS(8), ITRL33), (PARS(9), L1),
     &              (PARS(10), L1H), (PARS(11), L2), (PARS(12), MTRK),
     &              (PARS(13), MTRL), (PARS(14), STEP), (PARS(15),
     &              STOPT), (PARS(16), T1), (PARS(17), T2), (PARS(18),
     &              T3), (PARS(19), X11), (PARS(20), X2F)
      DIMENSION     PC(34)
      COMMON        /PRCMP/ PC
      SAVE          /PRCMP/
C
      PARAMETER     (GEES = 386.2, DEGREES = 57.29577951308232)
C
C  Update control inputs with GETCON then save new values with STRSYS,
C  BRAKE, and ACCEL.
C
      CALL GETCON(T, SWA, BPF, THROT)
      CALL STRSYS(SWA)
      CALL BRAKE(BPF, FZ1, FZ2, FZ3, FZ4, FZ5, FZ6, T)
      CALL ACCEL(THROT, U(1))
      Z(78) = SQRT(Z(37) + U(2)**2)
      Z(79) = CHKAX((Z(40) + UP(2)), Z(78), FB1, FB2, FB3)
      Z(80) = (PC(5)*UP(3) + UP(2))
      Z(81) = SQRT(Z(20)*(PC(20)*Z(20) -X2F*(Z(23) -Z(24))) + Z(22)
     &        *(U(2) + Z(21) -X2F*Z(20)*C(4)) + U(1)*(U(1) +
     &         X2F*Z(20)*S(4)))
```

```
      Z(82) = CHKAX(-(-U(3)*(Z(27) + Z(28)) + X2F*(UP(3) + UP(4))
     &          -Z(80)*C(4) + UP(1)*S(4)), Z(81), FB1, FB2, FB3)
      Z(83) = CHKAX(-(Z(41) -UP(1)), Z(78), FB1, FB2, FB3)
      Z(84) = CHKAX((-U(3)*(Z(23) -Z(24)) + Z(44) + UP(1)*C(4) +
     &          Z(80)*S(4)), Z(81), FB1, FB2, FB3)
      Z(85) = (PC(23) + PC(24)*Z(53) + PC(25)*Z(84))
      Z(86) = (PC(30) + PC(31)*Z(83) -PC(32)*Z(85) -(PC(33)*(FX5 + FX6)
     &          -PC(34)*Z(84))*C(4) -(PC(33)*(FY5 + FY6)
     &          -PC(34)*Z(82))*S(4))
      CALL LOADS(Z(79), Z(82), Z(83), Z(84), (PC(13) -Z(85) -Z(86)),
     &      Z(86), Z(85), Q(1), U(1), UP(1), QP(1))
      CALL TERMIN(Q(1), STOPT, BPF)
C
C  Normalize angular coordinates
C
      Q(3) = NORMA(Q(3))
      Q(4) = NORMA(Q(4))
C
      RETURN
      END
```

APPENDIX E

AUXILIARY SUBROUTINES

This appendix lists the file that contains the special purpose auxiliary subroutines used in the simulation. These were written manually.

```
C  This file contains auxiliary functions and subroutines for the
C  yaw-plane tractor-semitrailer model.  They are based on models
C  specified by Paul Fancher at UMTRI.

C  written by M. Sayers and Z. Bareket, Last update: May 19, 1990

C  (c) The Regents of The University of Michigan, all rights reserved.

C ACCEL(THROT, SPEED) - compute "requested" acceleration force for each C
                                    axle based on measured THROT
C BFECHO(IFILE) - echo brake force data
C BFREAD(IFILE) - read table inputs for brake pedal position
C BRAKE(BPF,FZ1, FZ2, FZ3, FZ4, FZ5, FZ6,T) - compute Fx- for each axle C
                                              based on measured BPF
C function CHKA(ALPHA, SPEED) - make slip angle = 0 for low speeds
C function CHKAX(ACC, SPEED, FB1, FB2, FB3)- make acceleration =0 if V=0
C function FRR(F0, V) - compute rolling resistance
C FRREAD(IFILE) - read table input for road friction conditions
C FRECHO(IFILE) - echo road friction conditions data
C GETCON(SWA, BPF, THROT) - get control inputs
C GETFA(FACC) - get FACC from private common
C GETFBS(BF1, BF2, BF3) - get current brake forces from common
C GETFZS(FZ1, FZ2, FZ3, FZ4, FZ5, FZ6) - get Fz's from common
C GETSTR(X) - get steer angle from common
C INROP(X, Y, VINIT) - Initialize roll & pitch angles, roll, and initial
C                                                                    speed
C INVEL(VEL, TITLE, STOPT, CONV) - prompt user for speed, title, stopt
C KFECHO - echo roll and pitch stiffness data
C KFREAD - read roll and pitch stiffness data
C LOADS(Q, QP, U, UP) - compute load transfers
C RESET(Q, QP, U, UP) - Reset all variables to zero.
C STECHO(IFILE) - echo steer data
C STREAD(IFILE) - read table inputs for steer
C STRSYS(SWA) - compute steer of front wheels
C TECHO(ID, IFILE) - echo tire data
C TERMIN(Q, STOPT) - send termination message if rollover, or
C                                                    jackknife occurs
C THECHO(IFILE) - echo throttle data
C THREAD(IFILE) - read table inputs for throttle force
C TINIT(ID, LOAD) - put static loads into common
C TIRE(ID, ALPHA, FZ, FY, FB, FA, FX) - compute shear forces
C TREAD(ID, BUFFER) - put tire data into common
C These subroutines define aspects of the vehicle system not included in
C AUTOSIM.  First, the control inputs are defined by subroutines
C ACCEL, BRAKE, GETCON, and STRSYS.  Mimicking a driving simulator, the
```

```
C subroutine GETCON provides steer, throttle, and brake inputs from the
C "driver."  Then, the routines ACCEL, BRAKE, and STRSYS convert those
C inputs into proper units and save them for later reference in private
C common blocks.  Within the equations of motion, the throttle, brake,
C and steer inputs are obtained from the private common blocks with the
C subroutines GETFA, GETFB, and GETSTR.
C
C The controls are obtained from three simple table lookups.  Each table
C is read by an input routine (BFREAD, STREAD, and THREAD), and echoed
C by an echo routine (BFECHO, STECHO, and THECHO).
C
C The tire model is dealt with in the subroutines TIRE, TINIT, TREAD,
C  and  TECHO.
C
C Changes in tire load are not directly accommodated in the AUTOSIM
C model, which is limited to yaw-plane motions.  However, computations
C are made for the benefit of the tire model with the routines LOADS and
C GETFZS.




C=======================================================================
      SUBROUTINE ACCEL(THROT, SPEED)
C=======================================================================
C  this subroutine computes a "requested" acceleration force for each
C  side of the drive axle(s), based on the measured throttle THROT.
C  The updated force is kept in a private COMMON block.

C  ACCEL is called in MAIN during the update operation.

C  --> THROT  REAL   throttle position  (0 ≤ THROT ≤ 1)
C  --> SPEED  REAL   forward velocity

      IMPLICIT NONE
      REAL THROT, SPEED, VELOCITY, DRIVF
      include throttle.inc

C for acceleration and engine performance, any speed below 82.5 in/sec
C is taken as constant; for that purpose, use local variable:

      VELOCITY = SPEED
      IF(VELOCITY .LE. 82.5)  VELOCITY = 82.5

C the total driving force (6600 = 550*12):

      DRIVF = THROT*HPMAX*6600./VELOCITY

C each wheel gets half:

      FA = DRIVF*.5

      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE BFECHO(IFILE)
C=======================================================================
C  Echo brake force data

      IMPLICIT NONE
      INTEGER  IFILE, J

      include tables.inc
      include brakes.inc

      WRITE(IFILE, '(A)') 'BRAKE'
      WRITE(IFILE, '(T6,G13.6,T20,5A)') BGAIN1*2.,
     & 'Gain for axle 1 (axle brake torque[in-lb]/input pressure[psi])'
      WRITE(IFILE, '(T6,G13.6,T20,5A)') BGAIN2*2.,
     & 'Gain for axle 2 (axle brake torque[in-lb]/input pressure[psi])'
      WRITE(IFILE, '(T6,G13.6,T20,5A)') BGAIN3*2.,
     & 'Gain for axle 3 (axle brake torque[in-lb]/input pressure[psi])'
      WRITE(IFILE, '(T6,G13.6,T20,5A)') RTIRE,
     & 'Rolling radius of the tires [in]'

      WRITE(IFILE, '(T8,I2,T24,A)') NTAB(3),
     & 'Number of points in time vs. pedal position table (number)'
      DO 10, J=1,NTAB(3)
        WRITE(IFILE, '(G13.6,'', '', G13.6, 5X,5A)')
     &       TABLES(J,1,3), TABLES(J,2,3),
     &       '  point for table-lookup: [sec], position'
   10 CONTINUE
      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE BFREAD(IFILE)
C=======================================================================
C  Read table with brake pedal position values (lb)

      IMPLICIT NONE
      INTEGER IFILE
      include tables.inc
      include brakes.inc

      READ(IFILE, *) BGAIN1
      READ(IFILE, *) BGAIN2
      READ(IFILE, *) BGAIN3
      READ(IFILE, *) RTIRE
      BGAIN1 = .5*BGAIN1
      BGAIN2 = .5*BGAIN2
      BGAIN3 = .5*BGAIN3

      CALL TBREAD(TABLES(1,1,3), TABLES(1,2,3), NTAB(3), IFILE)
      CALL INITTB(TABLES(1,1,3), TABLES(1,2,3), TABLES(1,3,3), NTAB(3))
      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE BRAKE(BPF, FZ1, FZ2, FZ3, FZ4, FZ5, FZ6, T)
C=======================================================================
C  this subroutine computes "requested" brake forces from 3 axles (FB
C  each is for one side of the vehicle) based on the measured brake
C  pedal position BPF.  The updated forces are kept in a private COMMON
C  block.
C  the subroutine also checks for too high brake forces and chops it

C  BRAKE is called in MAIN during the update operation.

C  --> BPF    REAL   brake pedal position (0 ≤ BPF ≤ 1)

      IMPLICIT NONE
      REAL    BPF, BF1, BF2, BF3, BF4, BF5, BF6, FZMIN, BPFMAX, T,
     &       TABLE

      INTEGER J, J1
      include brakes.inc
      include tire.inc
      include tables.inc

C   check if chopping the brake force is required:
C     *  skip the process if there's no braking;
C     *  chop if the least loaded wheel has less than 3000 lb. on it;
C     *  chop at a level that will enable all the wheels to lock;

      IF (BPF .EQ. 0.0) GOTO 100
      FZMIN = AMIN1(FZ1, FZ2, FZ3, FZ4, FZ5, FZ6)
      IF (FZMIN .GT. 3000.0) GOTO 100

C  Make sure  not to divide by zero:
      IF (BGAIN1 .EQ. 0.0) THEN
            BF1 = 0.0
            BF2 = 0.0
      ELSE
            BF1 = FZ1*FRICT*1.5/(BGAIN1*BRPRES/RTIRE)
            BF2 = FZ2*FRICT*1.5/(BGAIN1*BRPRES/RTIRE)
      ENDIF

      IF (BGAIN2 .EQ. 0.0) THEN
            BF3 = 0.0
            BF4 = 0.0
      ELSE
            BF3 = FZ3*FRICT*1.5/(BGAIN2*BRPRES/RTIRE)
            BF4 = FZ4*FRICT*1.5/(BGAIN2*BRPRES/RTIRE)
      ENDIF
      IF (BGAIN3 .EQ. 0.0) THEN
            BF5 = 0.0
            BF6 = 0.0
      ELSE
            BF5 = FZ5*FRICT*1.5/(BGAIN3*BRPRES/RTIRE)
            BF6 = FZ6*FRICT*1.5/(BGAIN3*BRPRES/RTIRE)
      ENDIF
      BPFMAX = AMAX1(BF1, BF2, BF3, BF4, BF5, BF6)
```

```
      DO 10 J = 1,NTAB(3)
            IF (T .LT. TABLES(J,1,3)) THEN
              J1 = J
              GO TO 15
            ENDIF
10    CONTINUE

15    TABLES(J1,2,3) = BPFMAX

      CALL INITTB(TABLES(1,1,3), TABLES(1,2,3), TABLES(1,3,3), NTAB(3))
      BPF = TABLE(T, TABLES(1,1,3), TABLES(1,2,3), TABLES(1,3,3),
     &                NTAB(3))

C  use the chopped (or untouched, according to the case) BPF:

100   FB1 = BPF*BGAIN1*BRPRES/RTIRE
      FB2 = BPF*BGAIN2*BRPRES/RTIRE
      FB3 = BPF*BGAIN3*BRPRES/RTIRE


      RETURN
      END
```

```
C=======================================================================
      FUNCTION CHKA(ALPHA, SPEED)
C=======================================================================
C  Since the slip angle is computed with speed in the denominator, for
C  too low speeds (≤ 2 MPH, or 35 in/sec) ALPHA will be too large. This
C  function prevents that, and set the slip angle to zero in such a
C  case. This function appears before calling to TIRE in subroutine
C  DIFEQN.

C  --> ALPHA     REAL   slip angle
C  --> SPEED     REAL   forward speed

      IMPLICIT NONE
      REAL CHKA, ALPHA, SPEED, V0
      PARAMETER (V0 = 35.)

      IF (SPEED .LT. V0) THEN
        CHKA = 0.0
      ELSE
        CHKA = ALPHA
      END IF

      RETURN
      END
```

```
C=======================================================================
      FUNCTION CHKAX(ACC, SPEED, FB1, FB2, FB3)
C=======================================================================
C  With the presence of a braking force, the resultant Fx will cause
C  AUTOSIM to calculate some longitudinal & lateral accelerations, even
C  when the vehicle is stationary, and cause a load transfer in LOADS.
C  This function prevents that, and set the accelerations (Ax & Ay) to
C  zero if the speed ≤ 35.

C  --> ACC       REAL   acceleration
C  --> SPEED     REAL   forward speed
C  --> FB1       REAL   braking in axle #1
C  --> FB2       REAL   braking in axle #2
C  --> FB3       REAL   braking in axle #3

      IMPLICIT NONE
      REAL CHKAX, ACC, SPEED, V0, FB1, FB2, FB3
      PARAMETER (V0 = 35.)

      IF((SPEED .LE. V0) .AND.
     &   ((FB1 .NE. 0.0) .OR. (FB2 .NE. 0.0) .OR. (FB3 .NE. 0.0))) THEN
        CHKAX = 0.0
      ELSE
        CHKAX = ACC
      END IF

      RETURN
      END
```

```
C=======================================================================
      FUNCTION FRR(F0, V)
C=======================================================================
C  compute lumped vehicle rolling resistance.  This function appears
C  in the equations of motion in subroutine DIFEQN.

C  --> F0      REAL   nominal rolling resistance: W*CRR (LB)
C  --> V       REAL   forward speed

      IMPLICIT NONE
      REAL FRR, F0, V, V0
      PARAMETER (V0 = 50.)

      IF (V .GT. V0) THEN
        FRR = F0
      ELSE
        FRR = F0*V/V0
      END IF
      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE FRREAD(IFILE)
C=======================================================================
C  read friction input parameters
C  FRREAD is called from within INPUT
C --> IFILE  INTEGER   file i/o number

      IMPLICIT NONE
      include tire.inc
      INTEGER  IFILE

      READ(IFILE, *) FRICT
      READ(IFILE, *) FRATIO
      RETURN
      END
C=======================================================================
      SUBROUTINE FRECHO(IFILE)
C=======================================================================
C  Echo friction data
      IMPLICIT NONE
      INTEGER  IFILE
      include tire.inc
      REAL          DEGREE
      PARAMETER    (DEGREE = 57.29577951308232)

      WRITE(IFILE, '(A)') 'FRICTION'
      WRITE(IFILE, '(T6,F4.2,T24,A)') FRICT,
     & 'Tire/Road friction value'
      WRITE(IFILE, '(T6,F4.2,T24,A)') FRATIO,
     & 'Ratio between sliding friction and "normal" friction'
      RETURN
      END
C=======================================================================
      SUBROUTINE GETCON(T, SWA, BPF, THROT)
C=======================================================================
C  In this version (UMTRI), GETCON generates control inputs by using
C  tabular values for SWA, BPF, and THROT as functions of T.
C  In a driving simulator, GETCON gets control inputs from the
C  driver by accessing a digitizer and scaling the signal.
C  GETCON is called in MAIN during the update operation.
C  --> T      REAL   time
C  <-- SWA    REAL   steering wheel angle
C  <-- BPF    REAL   brake pedal position (0 ≤ BPF ≤ 1)
C  <-- THROT  REAL   throttle position    (0 ≤ THROT ≤ 1)

      include tables.inc
      REAL   T, SWA, BPF, THROT

C Get controls from table look-up function TABLE
      SWA = TABLE(T, TABLES(1,1,1), TABLES(1,2,1), TABLES(1,3,1),
     &                NTAB(1))
      THROT = TABLE(T, TABLES(1,1,2), TABLES(1,2,2), TABLES(1,3,2),
     &                NTAB(2))
      BPF = TABLE(T, TABLES(1,1,3), TABLES(1,2,3), TABLES(1,3,3),
     &                NTAB(3))
      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE GETFA(FACC)
C=======================================================================
C  Get current "requested" acceleration force from private common
C  GETFA is called at the start of DIFEQN.

C  <-- FACC  REAL   current value of acceleration force (1 side)

      IMPLICIT NONE
      include throttle.inc
      REAL    FACC

      FACC = FA
      RETURN
      END
C=======================================================================
      SUBROUTINE GETFBS(BF1, BF2, BF3)
C=======================================================================
C  Get current "requested" brake forces from private common
C  GETFBS is called at the start of DIFEQN.

C  <-- BF1   REAL   brake force, axle 1 (1 side)
C  <-- BF2   REAL   brake force, axle 1 (1 side)
C  <-- BF3   REAL   brake force, axle 1 (1 side)

      IMPLICIT NONE
      include brakes.inc
      REAL    BF1, BF2, BF3

      BF1 = FB1
      BF2 = FB2
      BF3 = FB3
      RETURN
      END
C=======================================================================
      SUBROUTINE GETFZS(FZ1, FZ2, FZ3, FZ4, FZ5, FZ6)
C=======================================================================
C  Get current wheel loads from private common
C  GETFZS is called at the start of DIFEQN.
C          (*** AS OF 3/16/90 IT'S NOT IN USE AT ALL !!!  ***)

C  <-- FZi   REAL   vertical force, wheel i

      IMPLICIT NONE
      include loads.inc
      REAL   FZ1, FZ2, FZ3, FZ4, FZ5, FZ6

C Replace this with something more interesting

      FZ1 = VERTF(1)
      FZ2 = VERTF(2)
      FZ3 = VERTF(3)
      FZ4 = VERTF(4)
      FZ5 = VERTF(5)
      FZ6 = VERTF(6)
      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE GETSTR(X)
C=======================================================================
C  Get current wheel steer angle from private common

C  GETSTR is called at the start of DIFEQN.

C  <-- X   REAL   front wheel steer angle (rad)

      IMPLICIT NONE
      REAL X
      include steer.inc
      X = STEER
      RETURN
      END
C=======================================================================
      SUBROUTINE INROP(X, Y, VNOT)
C=======================================================================
C  Initialize flag, roll and pitch angles to be zero, and registers the
C  initial velocity:

C  INROP is called in subroutine INIT, during the initialization.

C  --> X     REAL  Initial value of Pitch angle (rad)
C  --> Y     REAL  Initial value of Roll  angle (rad)
C  --> VNOT  REAL  Initial velocity

      Include loads.inc
      CHARACTER*80  INFILE, TITLE
      REAL          C, CDRAG, CHKA, CRR, DEGREES, FA, FB1, FB2, FB3,
     &              FORCEM, FRR, FX1, FX2, FX3, FX4, FX5, FX6, FY1,
     &              FY2, FY3, FY4, FY5, FY6, FZ1, FZ2, FZ3, FZ4, FZ5,
     &              FZ6, GEES, H1, H2, HC, IPRINT, ITRK33, ITRL33, L1,
     &              L1H, L2, MTRK, MTRL, PARS, PC, PHEE, Q, QP, S,
     &              STEER, STEP, STOPT, T, T1, T2, T3, THETA, U, UP,
     &              X11, X2F, Z
      INTEGER       NCOORD, NPARS, NSPEED
C
      PARAMETER    (NCOORD = 4, NSPEED = 4)
      DIMENSION    Q(NCOORD), QP(NCOORD), U(NSPEED), UP(NSPEED)
      DIMENSION    C(4), FORCEM(2), S(4), Z(86)
      COMMON       /DYVARS/ C, FORCEM, S, Z, STEER, FA, FB1, FB2, FB3,
     &             FZ1, FZ2, FZ3, FZ4, FZ5, FZ6, PHEE, THETA, FY1, FX1,
     &             FY2, FX2, FY3, FX3, FY4, FX4, FY5, FX5, FY6, FX6
      SAVE         /DYVARS/

      IMPLICIT NONE

      REAL  X, Y, VNOT

      THETA = X
      PHEE = Y
      VINIT = VNOT
      FLAG = 0

      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE INVEL(VEL, TITLE, STOPT, CONV)
C=======================================================================
*  This prompts the user for a speed, title, and stop time.
*  It is called in subroutine INIT, during the initialization.
*  <-- VEL    real     forward speed (in/sec)
*  <-- TITLE  char     title for run
*  <-- STOPT  real     stop time for simulation
*  --> CONV   real     conversion for units.  1=inches, .0254 = m

      IMPLICIT NONE
      CHARACTER*32  STR1
      CHARACTER*(*) TITLE
      INTEGER       LENSTR, L, L1
      REAL*4        VEL, STOPT, CONV, REAL4

*  Get simulation speed.
      REAL4 = 55.
      CALL GETX(' Simulated vehicle speed (mi/h):', REAL4)
      VEL = REAL4 * 88 * 12 * CONV / 60.

* Modify title.
      L = LENSTR(TITLE)
      CALL STRX(REAL4, STR1, L1)
      TITLE(L + 1:) = '; V = ' // STR1

      CALL GETST(' Title for this run:', TITLE, 1)

*  Get final time and set simulation parameters.

      REAL4 = STOPT
      CALL GETX(' Length of time for simulation (sec):', REAL4)
      STOPT = REAL4

   40 RETURN
      END
C=======================================================================
      SUBROUTINE KFREAD(IFILE)
C=======================================================================
C  Read roll and pitch stiffness parameters
C  KFREAD is called from within INPUT
C --> IFILE  INTEGER   file i/o number

      include loads.inc

      IMPLICIT NONE
      INTEGER  IFILE, II

      READ(IFILE, *) KFI
      READ(IFILE, *) NN

      DO 10 II=1,NN
            READ(IFILE, *) KFIR(II)
10    CONTINUE

      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE KFECHO(IFILE)
C=======================================================================
C  Echo roll and pitch stiffness parameters

C --> IFILE  INTEGER   file i/o number

      include loads.inc

      IMPLICIT NONE
      INTEGER  IFILE, II

      WRITE(IFILE, '(A)') 'ROL&PTCH'

      WRITE(IFILE, '(T1,F10.1,T21,A)') KFI,
     & 'Longitudinal (Pitch) Stiffness of Tractor'
      WRITE(IFILE, '(T1,I2,T21,A)') NN,
     & 'Total number of lumped axles'
      WRITE(IFILE, '(T1,F10.1,T21,A,T2,I2)') KFIR(1),
     & 'Lateral (Roll) Stiffness of Axle #1'
      WRITE(IFILE, '(T1,F10.1,T21,A,T2,I2)') KFIR(2),
     & 'Lateral (Roll) Stiffness of Axle #2'
      WRITE(IFILE, '(T1,F10.1,T21,A,T2,I2)') KFIR(3),
     & 'Lateral (Roll) Stiffness of Axle #3'

10    CONTINUE

      RETURN
      END


C=======================================================================
      SUBROUTINE LOADS(AYT, AYTR, AXT, AXTR, A1, A2, A3, Q, U, UP, QP)
C=======================================================================
C  Compute load transfers due to lateral and longitudinal acceleration
C     (For now, do longitudinal load transfer):

C  LOADS is called from the main program in the update process.

C  --> AYT   real  truck's lateral acc.
C  --> AYTR  real  trailer's lateral acc.
C  --> AXT   real  truck's longitudinal acc.
C  --> AXTR  real  trailer's longitudinal acc.
C  --> A1    real  first axle vertical load
C  --> A2    real  second axle vertical load
C  --> A2    real  third axle vertical load
C  --> Q  real  array of 4 generalized coordinates
C  --> U  real  array of 4 generalized speeds
C  --> UP real  array of derivatives of U
C  --> QP real  array of derivatives of Q

      include loads.inc
      include tire.inc

      IMPLICIT NONE
```

```
C   This section (to the next comment), is copied/pasted as  is from
C   from the AUTOSIM generated subroutine DIFEQN:

      CHARACTER*80  INFILE, TITLE
      REAL          C, CDRAG, CHKA, CRR, DEGREES, FA, FB1, FB2, FB3,
     &              FORCEM, FRR, FX1, FX2, FX3, FX4, FX5, FX6, FY1,
     &              FY2, FY3, FY4, FY5, FY6, FZ1, FZ2, FZ3, FZ4, FZ5,
     &              FZ6, GEES, H1, H2, HC, IPRINT, ITRK33, ITRL33, L1,
     &              L1H, L2, MTRK, MTRL, PARS, PC, PHEE, Q, QP, S,
     &              STEER, STEP, STOPT, T, T1, T2, T3, THETA, U, UP,
     &              X11, X2F, Z
      INTEGER       NCOORD, NPARS, NSPEED
C
      PARAMETER    (NCOORD = 4, NSPEED = 4)
      DIMENSION    Q(NCOORD), QP(NCOORD), U(NSPEED), UP(NSPEED)
      DIMENSION    C(4), FORCEM(2), S(4), Z(86)
      COMMON       /DYVARS/ C, FORCEM, S, Z, STEER, FA, FB1, FB2, FB3,
     &             FZ1, FZ2, FZ3, FZ4, FZ5, FZ6, PHEE, THETA, FY1, FX1,
     &             FY2, FX2, FY3, FX3, FY4, FX4, FY5, FX5, FY6, FX6
      SAVE         /DYVARS/
C
      PARAMETER    (NPARS = 20)
      DIMENSION    PARS(NPARS)
      COMMON       /INPARS/ PARS, TITLE, INFILE
      SAVE         /INPARS/
C
      EQUIVALENCE  (PARS(1), CDRAG), (PARS(2), CRR), (PARS(3), H1),
     &             (PARS(4), H2), (PARS(5), HC), (PARS(6), IPRINT),
     &             (PARS(7), ITRK33), (PARS(8), ITRL33), (PARS(9), L1),
     &             (PARS(10), L1H), (PARS(11), L2), (PARS(12), MTRK),
     &             (PARS(13), MTRL), (PARS(14), STEP), (PARS(15),
     &             STOPT), (PARS(16), T1), (PARS(17), T2), (PARS(18),
     &             T3), (PARS(19), X11), (PARS(20), X2F)

C  Following are types that are "local" in the subroutine:


      REAL       AXT, AXTR, AYT, AYTR, AA, BB, X1R, A1, A2, A3
      INTEGER    ILOOP
      PARAMETER  (GEES = 386.2, DEGREES = 57.29577951308232)
      DIMENSION  AA(5), BB(4)


C  in case the vehicle stopped (speed ≤ 6), reset all the variables:
      IF((ABS(U(1)) .LE. 6.0) .AND. ((FB1 .GT. 0.0).OR.(FB2 .GT. 0.0))
     &                  .AND. (ABS(U(2)) .LE. 6.0)) THEN
            CALL RESET(Q(1), QP(1), U(1), UP(1))
            THETA = 0.0
            PHEE = 0.0
            AXT = 0.0
            AYT = 0.0
            AXTR = 0.0
            AYTR = 0.0
      ENDIF
```

```
C  Reset all FZ's to their static values if there are no accelerations
C              due to the function CHAX:

      IF ((AXT .EQ. 0.0) .AND. (AXTR .EQ. 0.0) .AND.
     &    (AYT .EQ. 0.0) .AND. (AYTR .EQ. 0.0))      THEN
            FZ1 = FZSTAT(1)*NTIRE(1)
            FZ2 = FZSTAT(2)*NTIRE(2)
            FZ3 = FZSTAT(3)*NTIRE(3)
            FZ4 = FZSTAT(4)*NTIRE(4)
            FZ5 = FZSTAT(5)*NTIRE(5)
            FZ6 = FZSTAT(6)*NTIRE(6)
            RETURN
      ENDIF
C
C  To be used in the subroutine:
      X1R = L1H - X11

C  Truck's PITCH angle:
C     (The 2.5 deg. condition stands for the pitch limit due to front
C     axle bump-stops).

      IF (THETA .GT. -(2.5/DEGREES)) THEN
            THETA = (-(A1-VERTF(1)*2)/(KFI*L1))
      ELSE
            THETA = -(2.5/DEGREES)
      ENDIF

C  VERTICAL LOADS due to lateral load transfer (ROLL):
C     AA and BB: temporary, intermediate variables;
C     KFIR(I)  : roll stiffness of each axle;
C     The truck and trailer are treated as one rolling body;

      AA(1) = MTRK * AYT * H1
      AA(2) = MTRL * AYTR * H2

      BB(1) = ABS((KFIR(3)*PHEE/T3))
      BB(2) = .5*A3
      BB(3) = ABS((KFIR(2)*PHEE/T2))
      BB(4) = .5*A2

C     Trailer's wheels are kept on the ground as long as:
      IF (BB(1) .LT. BB(2)) THEN
            AA(3) = KFIR(1)+KFIR(2)+KFIR(3)
            AA(4) = MTRK*GEES*H1 + MTRL*GEES*H2+LS
            PHEE = (-AA(1) - AA(2))/(AA(3) - AA(4))
C             NOTE:  under severe maneuvers, and due to the quasi-static
C               approach, the rate of change in roll angle (PHEE), might
C               be too big for the time step.  this will cause  negative
C               wheel loads. hence, double check:
            BB(1) = ABS((KFIR(3)*PHEE/T3))
          IF (BB(1) .GE. BB(2)) GO TO 15
            FZ1 = .5*A1 - KFIR(1)*PHEE/T1
            FZ2 = .5*A1 + KFIR(1)*PHEE/T1
            FZ3 = .5*A2 - KFIR(2)*PHEE/T2
            FZ4 = .5*A2 + KFIR(2)*PHEE/T2
            FZ5 = .5*A3 - KFIR(3)*PHEE/T3
            FZ6 = .5*A3 + KFIR(3)*PHEE/T3
```

```
C     If the above does not hold, trailer's wheels lifted, and now check
C     for rear tractor axle:

15    ELSE IF ((BB(1) .GE. BB(2)) .AND. (BB(3) .LT. BB(4))) THEN
            AA(3) = .5*A3 * T3 * SIGN(1,AYTR)
            AA(4) = KFIR(1) + KFIR(2)
            AA(5) = MTRK*GEES*H1 + MTRL*GEES*H2
            PHEE = (-AA(1) - AA(2) + AA(3)) / (AA(4) - AA(5))
C             NOTE:   again, to avoid negative wheel loads under  severe
C                  maneuvers, this time for the trailer read axle, double
C                  check:
            BB(3) = ABS((KFIR(2)*PHEE/T2))
          IF (BB(3) .GE. BB(4)) GO TO 25
            FZ1 = .5*A1 - KFIR(1)*PHEE/T1
            FZ2 = .5*A1 + KFIR(1)*PHEE/T1
            FZ3 = .5*A2 - KFIR(2)*PHEE/T2
            FZ4 = .5*A2 + KFIR(2)*PHEE/T2
            IF (PHEE .GT. 0.0) THEN
                  FZ5 = 0.0
                  FZ6 = A3
            ELSE
                  FZ5 = A3
                  FZ6 = 0.0
            ENDIF
      ELSE

C     That means we have passed the rollover threshold. Before quitting
C     with a rollover message, make sure the last wheel load points in
C     the plotting array are not negative:

25          IF (PHEE .GT. 0.0) THEN
                  FZ5 = 0.0
                  FZ6 = A3
                  FZ3 = 0.0
                  FZ4 = A2
            ELSE
                  FZ5 = A3
                  FZ6 = 0.0
                  FZ3 = A2
                  FZ4 = 0.0
            ENDIF


            FLAG = 1
      ENDIF

C   In order to make the main program quit if the vehicle had
C    rolled-over, the stop-time is artificially changed to a
C    small value. The type of termination is transferred via
C   "loads.inc" to the subroutine TERMIN:

      IF (FLAG .EQ. 1)  STOPT=0.02

100   CONTINUE

      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE RESET(Q, QP, U, UP)
C=======================================================================

C  Reset all variables to zero.

      IMPLICIT NONE

      INTEGER     NC, NS, ILOOP
      REAL        Q, QP, U, UP
      PARAMETER   (NC = 4, NS = 4)
      DIMENSION   Q(NC),QP(NC), U(NS), UP(NS)

      DO 10 ILOOP = 1, NC
            QP(ILOOP) = 0.0
10    CONTINUE

      DO 20 ILOOP = 1, NS
            U(ILOOP) = 0.0
            UP(ILOOP) = 0.0
20    CONTINUE


      RETURN
      END


C=======================================================================
      SUBROUTINE STECHO(IFILE)
C=======================================================================

C  Echo steer data

      IMPLICIT NONE

      INTEGER  IFILE, J
      include tables.inc

      REAL          DEGREE
      PARAMETER    (DEGREE = 57.29577951308232)

      WRITE(IFILE, '(A)') 'STEER'

      WRITE(IFILE, '(T8,I2,T24,A)') NTAB(1),
     & 'Number of points in time vs. front-axle steer (number)'
      DO 10, J=1,NTAB(1)
        WRITE(IFILE, '(G13.6,'', '', G13.6, 5X,5A)')
     &       TABLES(J,1,1), TABLES(J,2,1)*DEGREE,
     &       '  point for table-lookup (sec), (deg)'

   10 CONTINUE

      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE STREAD(IFILE)
C=======================================================================
C  Read table with steering angle values (at road wheel)

      IMPLICIT NONE
      include tables.inc
      REAL          DEGREE
      PARAMETER    (DEGREE = 57.29577951308232)
      INTEGER       J

      CALL TBREAD(TABLES(1,1,1), TABLES(1,2,1), NTAB(1), IFILE)

      DO 10 J=1, NTAB(1)
        TABLES(J,2,1) = TABLES(J,2,1)/DEGREE
   10 CONTINUE

      CALL INITTB(TABLES(1,1,1), TABLES(1,2,1), TABLES(1,3,1), NTAB(1))

      RETURN
      END


C=======================================================================
      SUBROUTINE STRSYS(SWA)
C=======================================================================
C  this subroutine computes the steer angle of the front wheels from
C  the steering wheel angle.

C  STRSYS is called in MAIN during the update operation.

C  --> SWA  REAL   steering wheel angle

      IMPLICIT NONE

      REAL SWA
      include steer.inc

      STEER = SWA

      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE TECHO(ID, IFILE)
C=======================================================================

C  Echo data for tire # ID

C  TECHO is called from within subroutine ECHO

C --> ID     INTEGER   ID number of tire (1, 2, or 3)
C --> IFILE  INTEGER   file id for echo
C --> FZ     REAL      static tire load

      IMPLICIT NONE

      include tire.inc

      REAL          DEGREE, FZ
      PARAMETER    (DEGREE = 57.29577951308232)
      INTEGER      ID, IFILE


      WRITE(IFILE, '(A,I1,T24,5A)') 'TIRE', ID,'Tire model of form: ',
     & 'Calph = A*Fz - B*Fz**2'

      WRITE(IFILE, '(G13.6,T24,5A)') ATIRE(ID)/DEGREE,
     & 'Coefficient A in tire model (1/deg)'

      WRITE(IFILE, '(G13.6,T24,5A)') BTIRE(ID)/DEGREE,
     & 'Coefficient B in tire model (1/(lb-deg))'

      WRITE(IFILE, '(F5.0,T24,5A)') NTIRE(ID),
     & 'Number of tires at location (count)'

      FZ = FZSTAT(ID)

      WRITE(IFILE, '(G13.6,T24,5A)') FZ,
     & 'nominal static load (lb)'

      WRITE(IFILE, '(G13.6,T24,5A)')
     &  (FZ*ATIRE(ID) - BTIRE(ID)*FZ*FZ)/DEGREE,
     & 'nominal cornering stiffness (lb/deg)'


      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE TERMIN(Q, STOPT, BPF)
C=======================================================================
C  Sends a termination message to the screen according to the case:
C  FLAG = 1       Vehicle had rolled-over
C     (FLAG is determined in LOADS):
C  --> Q      real    array of 4 generalized coordinates
C  <-> STOPT  real    "modified" quitting time, if necessary
C  --> BPF    real    brake pedal position (0 ≤ BPF ≤ 1)

      include loads.inc
      IMPLICIT NONE
      REAL        Q, STOPT, BPF
      DIMENSION   Q(4)

      IF (FLAG .EQ. 1) THEN
              PAUSE 'VEHICLE HAD ROLLED-OVER.  HIT  CR  TO  EXIT.'

C  the following condition is for a case where braking is applied, and
C  the tractor-trailer angle is more than 45 deg.:
      ELSEIF ((ABS(Q(4)) .GE. 0.785) .AND. (BPF .GT. 0.0)) THEN
              PAUSE 'JACKKNIFE LOCK OCCURED.  HIT  CR  TO  EXIT.'
              STOPT=0.02

C  the following condition is for a case where a non-realistic maneuver was
C  attempted ( the trailer is more than 90 deg. with the tractor):
      ELSEIF (ABS(Q(4)) .GE. 1.571) THEN
              PAUSE 'TRAILER IS MORE THAN 90 Deg. WITH TRACTOR.
     &   HIT CR TO  EXIT.'
              STOPT=0.02
      ENDIF

      RETURN
      END
C=======================================================================
      SUBROUTINE THECHO(IFILE)
C=======================================================================
C  Echo throttle position and horsepower data.
      IMPLICIT NONE
      INTEGER  IFILE, J
      include tables.inc
      include throttle.inc

      REAL          DEGREE
      PARAMETER    (DEGREE = 57.29577951308232)
      WRITE(IFILE, '(A)') 'THROTTLE'
      WRITE(IFILE, '(T6,I4,T24,A)') HPMAX,
     & 'Maximum road Horsepower of the engine'
      WRITE(IFILE, '(T8,I2,T24,A)') NTAB(2),
     & 'Number of points in time vs. pedal position (number)'
      DO 10, J=1,NTAB(2)
        WRITE(IFILE, '(G13.6,'', '', G13.6, 5X,5A)')
     &       TABLES(J,1,2), TABLES(J,2,2),
     &       '  point for table-lookup [sec], position'
   10 CONTINUE
      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE THREAD(IFILE)
C=======================================================================
C  Read table with throttle values (total drive force, lb)

      IMPLICIT NONE
      INTEGER IFILE
      include tables.inc
      include throttle.inc

      READ (IFILE,*) HPMAX
      CALL TBREAD(TABLES(1,1,2), TABLES(1,2,2), NTAB(2), IFILE)
      CALL INITTB(TABLES(1,1,2), TABLES(1,2,2), TABLES(1,3,2), NTAB(2))
      RETURN
      END


C=======================================================================
      SUBROUTINE TINIT(ID, LOAD)
C=======================================================================
C  Save static tire load during initialization.  Also, initialize the
C  COMMON block /LOADSC/ used for possible in-the-loop load
C  computations.

C  TINIT is called from within INPUT during the initialization

C --> ID     INTEGER   ID number of tire (1, 2, 3, 4, 5 or 6)
C --> LOAD   REAL      static tire load

C  This block is for transferring the static loads as initialization of
C  the FZ's:

      IMPLICIT NONE

      CHARACTER*80  INFILE, TITLE
      REAL          C, CDRAG, CHKA, CRR, DEGREES, FA, FB1, FB2, FB3,
     &              FORCEM, FRR, FX1, FX2, FX3, FX4, FX5, FX6, FY1,
     &              FY2, FY3, FY4, FY5, FY6, FZ1, FZ2, FZ3, FZ4, FZ5,
     &              FZ6, GEES, H1, H2, HC, IPRINT, ITRK33, ITRL33, L1,
     &              L1H, L2, MTRK, MTRL, PARS, PC, PHEE, Q, QP, S,
     &              STEER, STEP, STOPT, T, T1, T2, T3, THETA, U, UP,
     &              X11, X2F, Z
      INTEGER       NCOORD, NPARS, NSPEED
C
      PARAMETER    (NCOORD = 4, NSPEED = 4)
      DIMENSION    Q(NCOORD), QP(NCOORD), U(NSPEED), UP(NSPEED)
      DIMENSION    C(4), FORCEM(2), S(4), Z(86)
      COMMON       /DYVARS/ C, FORCEM, S, Z, STEER, FA, FB1, FB2, FB3,
     &             FZ1, FZ2, FZ3, FZ4, FZ5, FZ6, PHEE, THETA, FY1, FX1,
     &             FY2, FX2, FY3, FX3, FY4, FX4, FY5, FX5, FY6, FX6
      SAVE         /DYVARS/

C  These were in the subroutine originally:

      REAL          LOAD, DEGREE
      PARAMETER     (DEGREE = 57.29577951308232)
      INTEGER ID
```

```
      include tire.inc
      include loads.inc

      NTIRE(ID) = MAX(1.,NTIRE(ID))
      FZSTAT(ID) = LOAD/NTIRE(ID)
      VERTF(ID) = LOAD
      ATIRE(ID) = ATIRE(ID)*DEGREE
      BTIRE(ID) = BTIRE(ID)*DEGREE

C  Initialize the vertical loads:

      IF (ID .EQ. 1) THEN
            FZ1 = LOAD
      ELSE IF (ID .EQ. 2) THEN
            FZ2 = LOAD
      ELSE IF (ID .EQ. 3) THEN
            FZ3 = LOAD
      ELSE IF (ID .EQ. 4) THEN
            FZ4 = LOAD
      ELSE IF (ID .EQ. 5) THEN
            FZ5 = LOAD
      ELSE IF (ID .EQ. 6) THEN
            FZ6 = LOAD
      ENDIF

      RETURN
      END

C=======================================================================
      SUBROUTINE TIRE(ID, ALPHA, FZ, FY, FB, FA, FX, SPEED)
C=======================================================================
C  compute tire shear forces

C  TIRE is called from within DIFEQN

C --> ID     INTEGER   ID number of tire (1, 2, 3, 4, 5, or 6)
C --> ALPHA   REAL     Slip angle (rad)
C --> FZ      REAL     vertical load, one side (lb)
C <-- FY      REAL     side force (lb)
C --> FB      REAL     attempted braking force, one side (lb)
C --> FA      REAL     attempted acceleration force, one side (lb)
C <-- FX      REAL     longitudinal force (lb)
C --> SPEED   REAL     Forward velocity


      IMPLICIT NONE
      include tire.inc
      include steer.inc
      INTEGER ID, I
      REAL    ALPHA, FZ, FY, FB, FA, FX, FZ0, TEMP, SLIDFRIC, ALBAR,
     &        DEGREES, SPEED
      PARAMETER    (DEGREES = 57.29577951308232)

C     Initialize sliding friction value (NOTE: This should be moved into
C                                                  TINIT subroutine).
      SLIDFRIC = FRICT * FRATIO
```

```
C     First step, check and see whether we deal with sliding friction
C     due to braking:

      TEMP = ABS( FRICT*FZ*COS(ALPHA) )
      IF (FB .GE. TEMP) THEN
            FX = -SLIDFRIC*FZ*COS(ALPHA)
            FY = -SLIDFRIC*FZ*SIN(ALPHA)
            GO TO 100
      ENDIF

C     Second step, now that we are not sliding, determine the lateral
C     force  produced by the tire in the absence  of any longitudinal
C     force:

      FZ0 = FZ / NTIRE(ID)
      TEMP = ATIRE(ID) - BTIRE(ID)*FZ0

      ALBAR=TEMP*ALPHA/FRICT

      IF (ABS(ALBAR) .GE. (3.0)) THEN
           FY = -FRICT*FZ*(SIGN(1,ALPHA))

      ELSE
           FY = FRICT*FZ*(-ALBAR+(ALBAR*ABS(ALBAR))/3-(ALBAR**3)/27)

      ENDIF

C  Longitudinal driving force can be applied only to a driving wheel.
C  Distinguishing between driving and non-driving wheels was done in the
C  AUTOSIM input file. So, the argument FA is 0.0 if ID refers to a non- C
driving wheel.

C     Third step, determine if the friction can support  both the
C     lateral and longitudinal forces:

      TEMP = FY**2 + (FA - FB)**2

      IF (TEMP .LE. (FRICT*FZ)**2) THEN
            FX = FA - FB
            GO TO 100
      ELSE
            TEMP = SQRT(FY**2 + (FA - FB)**2)
            IF (TEMP .EQ. 0) THEN
                  FX = 0.0
                  FY = 0.0
                  GO TO 100
            ENDIF
            FY = SLIDFRIC*FZ*FY/TEMP
            FX = SLIDFRIC*FZ*(FA - FB)/TEMP
            GO TO 100
      ENDIF
100   CONTINUE

      RETURN
      END
```

```
C=======================================================================
      SUBROUTINE TREAD(ID, IFILE)
C=======================================================================
C  read tire input parameters.

C  TREAD is called from within INPUT

C --> ID     INTEGER   ID number of tire (1, 2, or 3)
C --> IFILE  INTEGER   file i/o number

      IMPLICIT NONE
      include tire.inc
      INTEGER ID, IFILE

      READ(IFILE, *) ATIRE(ID)
      READ(IFILE, *) BTIRE(ID)
      READ(IFILE, *) NTIRE(ID)
      RETURN
      END
```