Proceedings of the ASME 2011 International Mechanical Engineering Congress & Exposition
IMECE2011
November 11-17, 2011, Denver, Colorado, USA

IMECE2011-63803

AN EFFICIENT METHOD FOR SIMULATION OF 3D MULTI-BODY DYNAMIC PROBLEMS USING MAPLE AND SIMULINK PACKAGES

Kourosh H.Shirazi
Associate Professor, Department of Mechanical Engineering, Shahid Chamran University
Ahvaz, Khouzestan, Iran
kh_shirazi@yahoo.com

Behrooz Attaran
Graduate student, Department of Mechanical Engineering, Shahid Chamran University
Ahvaz, Khouzestan, Iran
attaranbehrooz@yahoo.com

Reza Zaeri
Graduate student, Department of Mechanical Engineering, Shahid Chamran University
Ahvaz, Khouzestan, Iran
reza.zayeri@yahoo.com

ABSTRACT

In this paper, an efficient method is proposed for modelling and simulation of multi-body dynamic problems. The method employs symbolic computational abilities of Maple as well as graphical environment of Matlab-Simulink to obtain and solve the equations of motion of a multi-body system accurately and rapidly. Considering a typical multi-body dynamical system the governing equations of system including second order equations of motion, first order nonholonomic and holonomic algebraic constraint equations are derived in Maple software. The state variables of the system are defined based on the systems degrees of freedom or generalized velocities. Converting the system’s equations to an algebraic form and combining them together by using a few Maple commands, the simplest form of the system’s equations of motion are obtained in the canonical standard state space form. This form is suitable when an explicit numerical method of integration is used. Then using a few toolboxes of Simulink, the equations are solved and can be studied. To procedural illustration of the method a lateral vehicle dynamic problem having thirty equations is considered. Beside the present method, Maple as well as Matlab is used to solve the problem. The results show the distinction of the method from the points of execution CPU time, accuracy and longer simulation. This method is suitable when investigation of long term behavior of multi-body dynamical systems is needed.

1. INTRODUCTION

There are many numerical and analytical methods to study multi-body dynamical systems. that each method beside its advantages possesses some disadvantages. Most of the method encounters low accuracy and instability. On the other hand it is not possible to derive an analytical solution in nonlinear dynamical [1]. It seems it is necessary to derive a simple and reliable method to solve multi-body dynamic problems accurately. Some researches have tried to utilize the advantages of several computer packages simultaneously. Moosavian and Rastegari [2,3] utilized MAPLE and MATLAB tools to simulating system of three manipulators mounted on a space free-flying robot. The explicit dynamics model of a multiple-manipulator SFFR can be implemented either numerically or symbolically. Moosavian and Papadopoulos [4] used SPACEMAPLE symbolic code to study this model. Garcia de Jalon and Callejo [5] introduced a simple theoretical approach namely the natural or fully cartesian coordinates and high level programming language (MATLAB) to dynamic analysis of 3D multi-body systems and considered two case studies, a closed-chain 3D robot and a McPherson car suspension system.

The aim of this paper is to present a novel methodology to solve algebraic-differential equations of motion of multi-body systems accurately and rapidly. This methodology is useful for widespread multi-body systems. The equations consist of 2th-order equations of motion, first order nonholonomic and holonomic algebraic equations. In the first step an algorithm is introduced which utilizes the symbolic computational abilities of Maple to transform the systems equations into the canonical standard state space form. In the second step to show the accuracy and speed of the method, a typical three degrees of freedom model is considered. The model is used to study the cornering behavior of a vehicle. In the third step, Maple as well as Matlab is used to solve the problem independently. In the final step a comparison is made between the obtained solution by the presented method and solution by the Maple and Matlab. Finally concluding remarks are given.

2. ALGORITHM

In this section, an algorithm to solve a system of second order nonlinear differential equations related to multi-body models is presented. First, utilizing of some MAPLE commands this system of equations is converted to the state space canonical form and after translating to the Matlab codes it is transferred to the Matlab-Simulink environment.

MAPLE

The primary system's equations consist of equilibrium, constitutive and compatibility equations. After determining the system's equations, all compatibility equations are substituted into the equilibrium ones. The number obtained equations is denoted by "f" and the set of equations is follows:

$$\{Eq_1\ ,\ Eq_2\ ,\ \dots\ ,\ Eq_f\} \tag{1}$$

Reduction Order of Equations and Building State Space

The order of variables whose behavior is to be studied in the model is reduced and then transformed in state space. "m" is the number of second order variables in Eq.(1) which need reducing order two times. The order of variables denoted by $\ddot{a}_i(t)$ is reduced.

$$\begin{cases} a_1(t) = U_1(t)\ , \dot{a}_1(t) = U_2(t)\ , \ddot{a}_1(t) = \frac{dU_2(t)}{dt} \\ a_2(t) = U_3(t)\ , \dot{a}_2(t) = U_4(t)\ , \ddot{a}_2(t) = \frac{dU_4(t)}{dt} \\ \dots \\ a_m(t) = U_{2m-1}(t)\ , \dot{a}_m(t) = U_{2m}(t)\ , \ddot{a}_m(t) = \frac{dU_{2m}(t)}{dt} \end{cases} \tag{2}$$

"n" is the number of first order variables in Eq. (1) which are transformed in state space.

$$\begin{cases} a_{m+1}(t) = U_{2m+1}(t)\ , \dot{a}_{m+1}(t) = \frac{dU_{2m+1}(t)}{dt} \\ a_{m+2}(t) = U_{2m+2}(t)\ , \dot{a}_{m+2}(t) = \frac{dU_{2m+2}(t)}{dt} \\ \dots \\ a_{m+n}(t) = U_{2m+n}(t)\ , \dot{a}_{m+n}(t) = \frac{dU_{2m+n}(t)}{dt} \end{cases} \tag{3}$$

The above equations can be generalized for higher order derivatives.
According to Eq. (4), system of *SS* is created using Eqs. (2) and (3).

$$SS := \left\{a_m(t) = U_{2m-1}(t)\ , \dot{a}_m(t) = U_{2m}(t)\ , \ddot{a}_m(t) = \frac{dU_{2m}(t)}{dt}, a_{m+n}(t) = U_{2m+n}(t)\ , \dot{a}_{m+n}(t) = \frac{dU_{2m+n}(t)}{dt}\right\} \tag{4}$$

Using "*subs*" command, the state variables defined in *SS* system are substituted in Eq. (1) according to Eq. (5).

$$\begin{cases} SS_1 := subs(SS, Eq_1) \\ SS_2 := subs(SS, Eq_2) \\ \dots \\ SS_f := subs(SS, Eq_f) \end{cases} \tag{5}$$

Considering Eq. (4), the variables of Eq. (6) that are (m+n) in number, are extracted from system of equations given in Eq. (5).

$$\left\{\frac{dU_{2m}(t)}{dt}, \frac{dU_{2(m+n)-1}(t)}{dt}\right\} \tag{6}$$

Now, ξ_i which denotes the state variables of Eq. (6), may be written in the forms of Eqs. (7) and (8).

$$\xi_i = \frac{dU_{2i}(t)}{dt}\ , i = 1, \dots, m \tag{7}$$

$$\xi_i = \frac{dU_{2i-1}(t)}{dt}\ , i = m+1, \dots, m+n \tag{8}$$

Some of the equations of Eq. (5) in which the variables of Eq. (6) exist are put in "*system0*".

$$system0 := \{SS_1, \dots, SS_{m+n}\} \tag{9}$$

The remaining equations of Eq. (5) are creating *system1*. The system given in Eq. (11) contains all the unknowns except those of Eq. (6).

$$system1 := \{SS_1, \dots, SS_{f-(m+n)}\} \tag{10}$$

$$\{x_1, x_2, \dots, x_{f-(m+n)}\} \tag{11}$$

To obtain the unknowns of Eq. 11, the system of equations, "*system1*", is solved using "*solve*" command as given in Eq. (12).

$$answer0 := solve\left(system1, \{x_1, x_2, \dots, x_{f-(m+n)}\}\right) \tag{12}$$

The unknowns obtained by Eq. (12) are then substituted in "*system0*" using "*subs*" command.

$$system2 := map\{simplify, subs(answer0, system0), symbolic\} \tag{13}$$

To obtain the state variables of Eqs. (7) and (8), the system of equations given in *system2* is solved by the use of "*solve*" command (Eq.(14)).

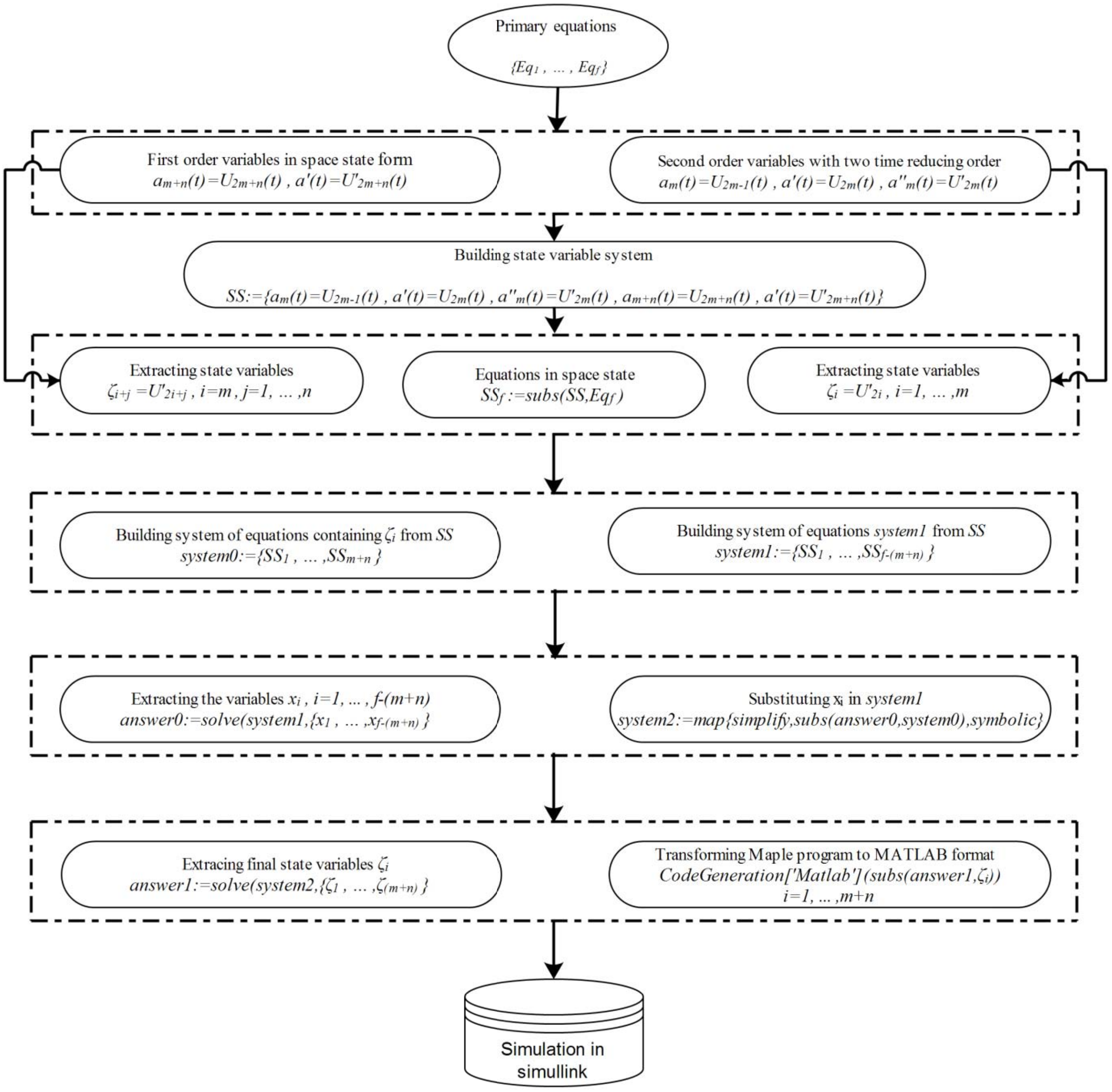
$$answer1 \coloneqq solve(system2, \{\xi_1, \dots, \xi_{m+n}\}) \tag{14}$$

Simulink

Finally, after solving the equations in space state, it is necessary to simulate the model. As MATLAB Simulink is used to simulate the model, the ultimate variables of Eq. (14) should be transformed in MATLAB program. So the "*CodeGeneration*" commands of the Maple (Eq. (15)) which can make MATLAB codes from state variables are used.

$$CodeGeneration['Matlab'](subs(answer1, \xi_i)) \ , \quad i = 1, \dots, m+n \tag{15}$$

Figure (1) shows a schematic of the algorithm.



Primary equations
$\{Eq_1, \dots, Eq_f\}$
First order variables in space state form
$a_{m+n}(t)=U_{2m+n}(t)$, $a'(t)=U'_{2m+n}(t)$
Second order variables with two time reducing order
$a_m(t)=U_{2m-1}(t)$, $a'(t)=U_{2m}(t)$, $a''_m(t)=U'_{2m}(t)$
Building state variable system
$SS:=\{a_m(t)=U_{2m-1}(t)$, $a'(t)=U_{2m}(t)$, $a''_m(t)=U'_{2m}(t)$, $a_{m+n}(t)=U_{2m+n}(t)$, $a'(t)=U'_{2m+n}(t)\}$
Extracting state variables
$\zeta_{i+j}=U'_{2i+j}$, $i=m$, $j=1, \dots, n$
Equations in space state
$SS_f:=subs(SS,Eq_f)$
Extracting state variables
$\zeta_i=U'_{2i}$, $i=1, \dots, m$
Building system of equations containing ζ_i from *SS*
$system0:=\{SS_1, \dots, SS_{m+n}\}$
Building system of equations *system1* from *SS*
$system1:=\{SS_1, \dots, SS_{f-(m+n)}\}$
Extracting the variables x_i , $i=1, \dots, f-(m+n)$
$answer0:=solve(system1,\{x_1, \dots, x_{f-(m+n)}\}$
Substituting x_i in *system1*
$system2:=map\{simplify,subs(answer0,system0),symbolic\}$
Extracing final state variables ζ_i
$answer1:=solve(system2,\{\zeta_1, \dots, \zeta_{(m+n)}\}$
Transforming Maple program to MATLAB format
$CodeGeneration['Matlab'](subs(answer1,\zeta_i))$
$i=1, \dots, m+n$
Simulation in simullink

Fig.1: Flowchart of an algorithm

3. Algorithm through an example

First kinematic equations (see appendix A. Eqs. (1) - (28) and (43) - (59)), kinetic and auxiliary equations (see appendix A. Eqs. (29) - (32), (39) - (42), (66) and (67)) of the model respectively are transformed in Maple program. Kinematic equations are automatically combined with kinetic and auxiliary. According to Eq. (1) and knowing that f=14, the below primary equations are derived.

$> Eq1 := zigmaMo(1,1) - Hdot1ou(1,1)$: (16)
$> Eq2 := zigmaMo(2,1) - Hdot1ou(2,1)$: (17)
$> Eq3 := zigmaMo(3,1) - Hdot1ou(3,1)$: (18)
$> Eq4 := zigmaFo(1,1) - (mus * (ao(1,1))$: (19)
$> Eq5 := zigmaFo(2,1) - (mus * (ao(2,1))$: (20)
$> Eq6 := zigmaFo(3,1) - (mus * (ao(3,1))$: (21)
$> Eq7 := zigmaMG(1,1) - Hdot3sG(1,1)$: (22)
$> Eq8 := zigmaMG(2,1) - Hdot3sG(2,1)$: (23)
$> Eq9 := zigmaMG(3,1) - Hdot3sG(3,1)$: (24)
$> Eq10 := zigmaFG(1,1) - (ms * (aG(1,1))$: (25)
$> Eq11 := zigmaFG(2,1) - (ms * (aG(2,1))$: (26)
$> Eq12 := zigmaFG(3,1) - (ms * (aG(3,1))$: (27)
$> Eq13 := Nfr - Nfl - \frac{2 * Mf}{te}$: (28)
$> Eq14 := Nrr - Nrl - \frac{2 * Mr}{te}$: (29)

Reducing Order of Equations and Building State Space

Maple is used to transform equations in space state. The main part of this section is how to transform the equations and the method of solving equations. Using Eq. (2) and (3), (n=2, m=2) number of state variables are 2m+n=6. State space can be built using Eq. (4).

$> SS := \{\theta(t) = U1(t), diff(\theta(t), t) = U2(t), diff(\theta(t), t\$2) = diff(U2(t), t), \phi(t) = U3(t), diff(\phi(t), t) = U4(t), diff(\phi(t), t\$2) = diff(U4(t), t), u(t) = U5(t), v(t) = U6(t)\}$: (30)

Eq. (6) is used for substituting final equations with state variables.

$> SS1 := subs(SS, Eq1)$: (31)
$> SS2 := subs(SS, Eq2)$: (32)
$> SS3 := subs(SS, Eq3)$: (33)
$> SS4 := subs(SS, Eq4)$: (34)
$> SS5 := subs(SS, Eq5)$: (35)
$> SS6 := subs(SS, Eq6)$: (36)
$> SS7 := subs(SS, Eq7)$: (37)
$> SS8 := subs(SS, Eq8)$: (38)
$> SS9 := subs(SS, Eq9)$: (39)
$> SS10 := subs(SS, Eq10)$: (40)
$> SS11 := subs(SS, Eq11)$: (41)
$> SS12 := subs(SS, Eq12)$: (42)
$> SS13 := subs(SS, Eq13)$: (43)
$> SS14 := subs(SS, Eq14)$: (44)

Solving Equation in State Space

After inserting all the equations in software, equations are substituted in each other automatically. 12 equations are derived from Newton's second law and Momentum equations of sprung and unsprung mass and 2 other equations are derived from auxiliary equations related to torsional springs. Finally, kinematic Eqs. (31) - (44) in state space form are derived.

Since our goal is to find 4 state variables ($U_2(t) = \frac{d\theta(t)}{dt}$ ، $U_4(t) = \frac{d\varphi(t)}{dt}$ ، $U_5(t) = u(t)$ and $U_5(t) = v(t)$), according to Eq. (9) we consider this 4 equations as a system of equations, "*system0*".

$> system1 := \{SS3, SS4, SS5, SS12\}$: (45)

According to Eq. (11) and knowing that the Eqs. (31) - (44) contain 10 unknowns; "*system1*" is defined as below using equation (10).

$> system1 := \{SS1, SS2, SS6, SS7, SS8, , SS9, SS10, SS11, SS13, SS14\}$ (46)

Considering Eq. (12) and writing "*answer0*", vertical reaction forces as well as the forces between A and B are extracted from "*system1*" as state variables using "*solve*" command:

$> answer0 := solve(system1, \{Nfl, Nfr, Nrl, Nrr, Ax, Ay, Az, Bx, By, Bz\})$: (47)

Using Eq. (13), "*system2*" is formed from a mixture of "*simplify*" command used for simplifying equations and "*subs*" command is used for substituting extracted unknown from "*answer0*" in the system of equations "*system0*", and "*map*" command to sort the equations:

$> system2 := map(simplify, subs(answer0, system0), symbolic)$: (48)

Using Eq. (14), "*answer1*" is used to extract state variables as follows:

$> answer1 := solve(sys2, \{diff(U2(t), t), diff(U4(t), t), diff(U5(t), t), diff(U6(t), t)\})$: (49)

Finally, after solving the equations in state space, it is necessary to use them for simulating the model. Using "*CodeGeneration*" command of Maple program of Eq. (15), the state variables are transformed in MATLAB format as follows:

$$> CodeGeneration['Matlab']\left(subs\left(answer1, \frac{d}{dt}U2(t)\right)\right) \tag{50}$$

$$> CodeGeneration['Matlab']\left(subs\left(answer1, \frac{d}{dt}U4(t)\right)\right) \tag{51}$$

$$> CodeGeneration['Matlab']\left(subs\left(answer1, \frac{d}{dt}U5(t)\right)\right) \tag{52}$$

$$> CodeGeneration['Matlab']\left(subs\left(answer1, \frac{d}{dt}U6(t)\right)\right) \tag{53}$$

The derived solution for each state variable contains more than 100 terms. To analyse them, MATLAB program is used.

Model Simulation in Simulink

Considering the input data given in Tab. 1 for the Fcn (Function Block Parameters) blocks, the model is simulated according to simulink model given in Figure 2. As an example, the time history of *u(t)* in first 5 seconds is shown in Fig. 3.

Tab.1: Input parameters of Fcn

$K_{\varphi f}$(kN/m)	100	I_z(kg m^2)	60	m_{us}(kg)	400
$K_{\varphi r}$(kN/m)	100	J_x(kg m^2)	60	m_s (kg)	2300
C_{fr}(kNs/m)	40	J_y(kg m^2)	80	W_s (N)	23000
C_{fl}(kNs/m)	40	J_z(kg m^2)	70	δ_{fr}(rad)	0.0698
C_{rr}(kNs/m)	40	b(m)	1.5	δ_{fl}(rad)	0.0524
C_{fl}(kNs/m)	40	c(m)	1.5	t_e(m)	1.5
h_g(m)	0.3	e(m)	0.6	η(rad)	0
h_r(m)	0.5	d(m)	0.4		

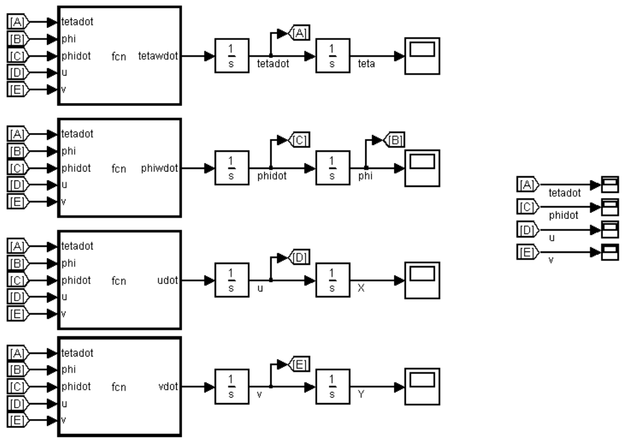


Fig.2: Simulink model

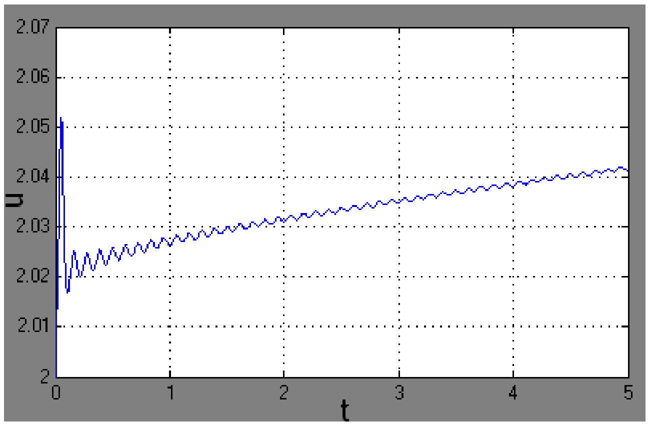


Fig.3: Time history of u(t) by present algorithm

4. Solving an Example Using Maple and Matlab

In this section, the given example in section 3 is solved using Maple and Matlab independently.

Maple

Considering Eqs. (16) - (29), the governing equations of system is solved only by the use of Maple commands. In other words, the unknowns are obtained without simulation in simulink and transforming into state space.
First, the initial conditions are determined. Then, the system of equations is solved using numerical methods. The initial conditions are given to Maple in the format of Eq. (54):

$$> initialconditions \coloneqq \{u(0) = 2, v(0) = 0.5, D(\theta)(0) = -0.1, \theta(0) = 0.01, D(\varphi)(0) = 0, \varphi(0) = 0.001\} \tag{54}$$

To solve this system in Maple environment, it is necessary to use a command which is able to solve algebraic and differential equations together. Therefore, "*mebdfi*" command is used:

$$> dsol \coloneqq dsolve(dsys \cup initialconditions, numeric, method = mebdfi, output = listprocedure): \tag{55}$$

The error given in Eq. (56) is observed during the solution process which shows that Maple is only able to solve the system for 0.09 seconds. The reason is that in this moment, the variable φ is equal to zero and the equations become singular. In this condition, the software is not able to solve the equations and the following error message is returned by Maple:

$$Error, (in\ unknown) unable\ to\ integrate\ past\ 0.92749768e-1: convergence\ could\ not\ be\ achieved\ with\ stepsize\ at\ minimum \tag{56}$$

The command in Eq. (57) is used to plot the time history of $u(t)$. Which can be seen in Fig. 4.

$$> plots[odeplot](dsol, [t, u(t)], 0..5, mumpoints = 10000): \tag{57}$$

As it was mentioned, the Maple can solve the equations for only 0.09 seconds and the below warning is returned:

$$Warning, could\ not\ obtain\ numerical\ solution\ at\ all\ points\ ,plot\ may\ be\ incomplete \qquad (58)$$

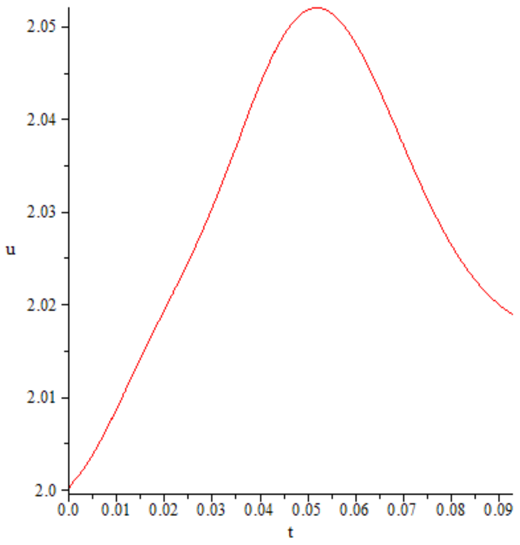


Fig.4. Time history of u(t) by maple

Matlab

Considering Eqs. (16) - (29) the governing equations of system the system is solved by the use of Matlab commands and unknown parameters are obtained.
The commands "*ode23t*" and "*ode15s*" are used in Matlab to solve the systems containing both differential and algebraic equations. In this method, the equations are must be in the form of Eq. (59) [6]:

$$M\dot{y} = f(t,y) \qquad (59)$$

Where "M" is the singularity matrix.

The equations must be transformed into state space and their order has to be reduced. The complexity of the equations given in section 3 causes the differential terms to appear in the right side of the equations and the form given in Eq. (59) is not satisfied. As a result, the Matlab commands in M-file are not able to solve the system by themselves.

5. Comparison

In this section, some tables are given as a comparison between methods given in sections 3 and 4.

To show cutoff simulation of system behavior, in Tab. 2 the present algorithm is compared with MAPLE and MATLAB independently solution. From this viewpoint the present algorithm has a running time over than 45000 second unlike the MAPLE execution time it stop at 0.09 second and MATLAB which is not performance for this problem.

Tab. 2: Cutoff simulation of system behavior

Method	Max Run Time (sec)
Present Algorithm	More than 45000
MAPLE	0.09
MATLAB	×

Tab. 3: CPU time for 0.09sec simulation of system behavior

Method	CPU time for 0.09sec simulation
Present Algorithm	0.56
MAPLE	5.203
MATLAB	×

In Tab. 3 the rapidity of an algorithm for 0.09 second simulation is compared than MAPLE and MATLAB. The results shown that the present algorithm is much more rapid compared to the MAPLE. Also MATLAB is not performance for this problem.

Tab. 4: A comparison on relative error and CPU time between fixed as well as variable step size method and ode5 of simulink for φ

Method		%E with respect to ode5	CPU time for 10^3sec simulation
Fixed-step type with size 10^{-4} sec	ode5	0	30019.27
	ode4	0	27129.41
	ode3	0.00005	16517.89
Variable-step type with relative tolerance 10^{-3} sec	ode45	0.00188	1420
	ode23	2.30607	972.39
	ode113	0.06125	1607.06

ode5:	Dormand-Prince	ode45:	Dormand-Prince
ode4:	Runge-Kutta	ode23:	Bogacki-Shampine
ode3:	Bogacki-Shampine	ode113:	Adams

Three methods of numerical solution with *fixed time-step* are compared in Tab. 4. The default numerical method considered in simulink is *ode5* which is compared with the methods *ode4* and *ode3*. From the execution CPU-time point of view for run-time of 1000 seconds, the *ode5* method is more rapid. But the *ode4* method is more accurate.

Also results are given in Tab. 4 are compared using *variable time-steps*. In this condition, the default numerical method considered in simulink is *ode5* which is compared with the methods *ode45*, *ode23* and *ode113*. From the execution CPU-time point of view for run-time of 1000 seconds, the *ode23* method is more rapid. But the *ode45* method is more accurate.

6. CONCLUSION

In this work a method for simulation of multi-body dynamic problems containing second order equations of motion, first order nonholonimic, and algebraic holonomic constraint equations was presented. The method combines benefits of both Maple and Matlab-Simulink packages to rapid as well as accurate simulation of the dynamic behavior. A comparison between the present method and the standard methods for numerical solutions of algebraic-differential equations in Maple as well as Matlab showed that the presented method is remarkably more efficient in speed, CUP consumption time and longer simulation time. These advantages make it suitable for investigation of long term dynamical behavior which is needed in chaotic and nonlinear dynamical systems.

REFERENCES

[1] Fan. Jianping, Huang.Tao, Tang.Chak-yin, Wang. Cheng, a predict-correct numerical integration scheme for solving nonlinear dynamic equations. Acta Mechanica Solida Sinica, Vol. 19, No. 4, December (2006).

[2] S. Ali A. Moosavian, Rambod Rastegari, Multiple-arm space free-flying robots for manipulating objects with force tracking restrictions, Journal of Robotics and Autonomous Systems 54 (2006).

[3] Rambod Rastegari, S.Ali A. Moosavian, Multiple impedance control of space free-flying robots via virtual linkages, Journal of Acta Astronautica 66 (2010).

[4] S.A.A. Moosavian, E. Papadopoulos, Explicit dynamics of space free-flyers with multiple manipulators via SPACEMAPLE, Journal of Advanced Robotics 18 (2004).

[5] J. García de Jalón & A. Callejo, A straight methodology to include multibody dynamics in graduate and undergraduate subjects, journal of Mechanism and Machine Theory (2011).

[6] Help Matlab Version R2009a

ANNEX A
Vehicle lateral motion in turn

The dynamic multi-body model of the Vehicle is a three dimensional model with three degree of freedom for Vehicle lateral motion in turn. It contains sprung mass m_s and unsprung mass m_u. The points G and O are the centre of sprung and unsprung mass respectively and point R is the cross section of line OG and roll axis. These three points are in a line when the roll does not move (according to fig. 1). The point G turns around roll axis with respect to point R and the point R is constant with respect to O. The distance between R and O is h_R and the distance between G and R is h_g. Four coordinate systems are considered for the model. The coordinate $x_0y_0z_0$ jointed to ground, $x_1y_1z_1$ jointed to unsprung mass in point O, $x_2y_2z_2$ jointed to unsprung mass in point R and $x_3y_3z_3$ jointed to sprung mass in point R. The goal is to find kinematic and kinetic equations of the system $x_1y_1z_1$.

Torsional stiffness of the rear and front axis are, $K_{\varphi f}$ and $K_{\varphi r.}$ The angle between role axis and the horizon is η. φ is the angle of body rolling and the angle of vehicle turning is θ.

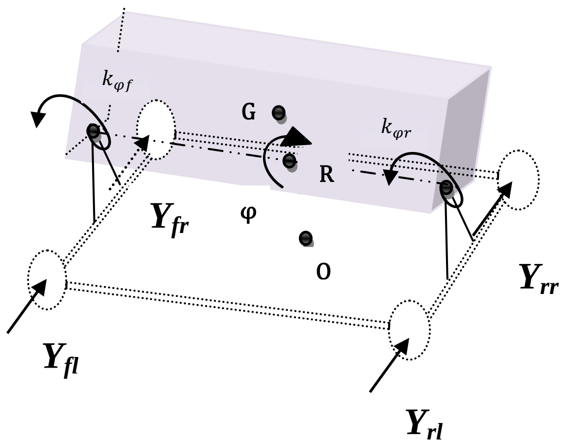


Fig. 1: Sprung and unsprung mass cordinte

Kinematic analysis of the Vehicle

Considering coordinate system 1 of unsprung centre of mass (Fig. 1 and 2), velocity and one dimensional acceleration of unsprung centre of mass can be found from Eqs. (17) and (18).

$$V_o = ui_1 + vj_1 \tag{1}$$
$$a_o = (\dot{u} - vr)i_1 + (\dot{v} + ur)j_1 \tag{2}$$

Because the Vehicle is turning on a turn, considering variation of rolling angle as angular velocity with respect to parameter r, the acceleration and angular velocity of the Vehicle in unsprung system, system 1, can be found from Eqs. (19) and (20).

$$\omega_u = rk_0 \tag{3}$$
$$\alpha_u = \ddot{\theta}\, k_o = \dot{r}\, k_0 \tag{4}$$

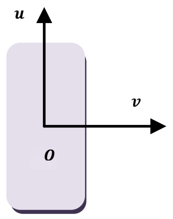


Fig. 2: Absolute velocity of unsprung mass

$$\omega_s = \omega_u + \omega_{\frac{s}{u}} = rk_0 + \dot{\varphi} i_2 \tag{5}$$

The Eq. (21) is defined for relative angular velocity of sprung mass with respect to unsprung mass. Considering Fig. 3 an equation for angular velocity in system one is derived again.

$$\omega_s = \dot{\varphi} \cos\eta \, i_1 + (r - \dot{\varphi} \sin\eta) k_1 \tag{6}$$

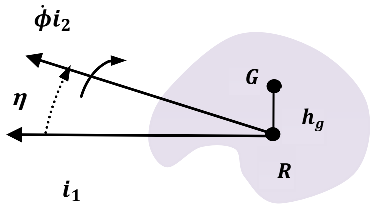


Fig. 3: Transformation from frame 2 to frame 1

Taking derivative from Eq. (22) absolute angular acceleration of body system is derived.

$$\alpha_s = \ddot{\varphi} \cos\eta \, i_1 + \dot{\varphi} r \cos\eta \, j_1 + (\dot{r} - \ddot{\varphi} \sin\eta) k_1 \tag{7}$$

The velocity of sprung mass with respect to point R can be found in Eq. (24).

$$V_G = V_R + V_{rel} + V_{trac} \tag{8}$$

The velocity, V_{trac}, can be found from Eq. (25) considering Fig. 3.

$$V_{trac} = \omega_s \times r_{\frac{G}{R}} = \omega_s \times \left(-h_g\right) k_3 \tag{9}$$

Since the Eq. (25) is in system 3, it is not necessary to define it in system 1. So the transform matrixes 26, 27 and 28 (transform from system 0 to 1, 1 to 2 and 2 to 3 respectively) are used.

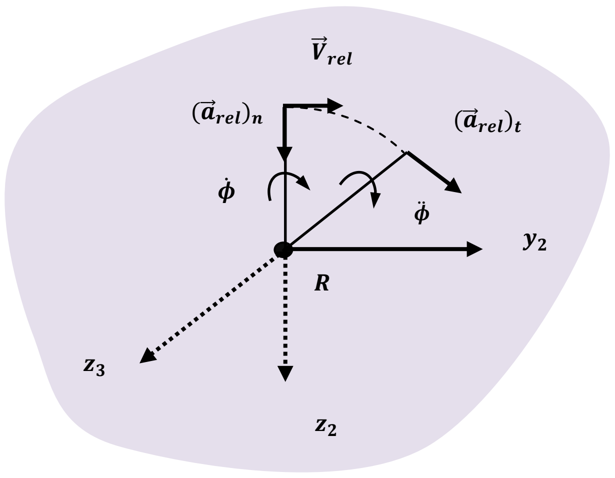


Fig. 4: Relative velocity and acceleration

Transform matrix of vehicle turning is defined as below.

$$T_0^1 = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Transform matrix of vehicle pitching is defined as below.

$$T_1^2 = \begin{bmatrix} \cos\eta & 0 & -\sin\eta \\ 0 & 1 & 0 \\ \sin\eta & 0 & \cos\eta \end{bmatrix} \tag{11}$$

Transform matrix of vehicle rolling is defined as below.

$$T_2^3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \tag{12}$$

Eq. (9) can be modified using transform matrixes (11) and (12). Before that Eq. (13) which show transform form system 3 to 1 is written.

$$T_3^1 = (T_1^3)^{-1} = (T_2^3 T_1^2)^{-1} = T_2^1 T_3^2 \tag{13}$$

$$V_{trac} = \omega_s \times \left(T_2^1\, T_3^2 \begin{bmatrix} 0 \\ h_g \sin\varphi \\ -h_g \cos\varphi \end{bmatrix} \right) \tag{14}$$

The relative velocity is obtained from Eq. 15 in system 2 considering Fig. 4.

$$V_{rel_2} = \left(\dot{\phi}\, i_2\right) \times \left(-h_g\, k_3\right) \tag{15}$$

The transform matrix (27) is used to obtain relative velocity in system 1.

$$V_{rel} = T_2^1\left(V_{rel_2}\right) = T_2^1 \left(\begin{bmatrix} \dot{\varphi} \\ 0 \\ 0 \end{bmatrix} \times T_3^2 \begin{bmatrix} 0 \\ h_g \sin\varphi \\ -h_g \cos\varphi \end{bmatrix} \right) \tag{16}$$

Since while the Vehicle is moving the plane x_1y_1 is parallel to plane x_oy_o and z_1 is parallel to $z_{o,}$ the below relation exist for velocity and acceleration.

$$V_R = V_o \tag{17}$$
$$a_R = a_o \tag{18}$$

Substituting Eqs. (14) - (17) in Eq. (8), the final equation for the velocity of centre of mass in system 1 is derived.

$$V_G = ui_1 + vj_1 + T_2^1 \left(\begin{bmatrix} \dot{\varphi} \\ 0 \\ 0 \end{bmatrix} \times T_3^2 \begin{bmatrix} 0 \\ h_g \sin\varphi \\ -h_g \cos\varphi \end{bmatrix} \right) + \omega_s \times \left(T_2^1\, T_3^2 \begin{bmatrix} 0 \\ h_g \sin\varphi \\ -h_g \cos\varphi \end{bmatrix} \right) \tag{19}$$

The absolute acceleration, $a_{G,}$ in system 3 (body) is derived from Eq. (20).

$$a_G = a_R + (a_{rel})_n + (a_{rel})_t + (a_{trac})_n + (a_{trac})_t + a_{cor} \tag{20}$$

Vertical and tangent relative accelerations are derived from Eqs. (21) and (22) considering Fig. 4 in system 3.

$$(a_{rel})_n = \begin{bmatrix} 0 \\ 0 \\ h_g\,\dot{\varphi}^2 \end{bmatrix} \tag{21}$$

$$(a_{rel})_t = \begin{bmatrix} 0 \\ h_g\,\ddot{\varphi} \\ 0 \end{bmatrix} \tag{22}$$

Using Eq. 13, the Eqs. (21) and (22) in system 1 are modified.

$$(a_{rel})_n = T_3^1 . \begin{bmatrix} 0 \\ 0 \\ h_g\,\dot{\varphi}^2 \end{bmatrix} \tag{23}$$

$$(a_{rel})_t = T_3^1 . \begin{bmatrix} 0 \\ h_g\,\ddot{\varphi} \\ 0 \end{bmatrix} \tag{24}$$

Vertical and tangent trac accelerations are derived from Eqs. (25), (26) and Eq. (13) in system 1.

$$\left(a_{trac_1}\right)_n = \omega_u \times \left(\omega_u \times T_3^1 \begin{bmatrix} 0 \\ h_g \sin\varphi \\ -h_g \cos\varphi \end{bmatrix} \right) \tag{25}$$

$$\left(a_{trac_1}\right)_t = \alpha_u \times T_3^1 \begin{bmatrix} 0 \\ h_g \sin\varphi \\ -h_g \cos\varphi \end{bmatrix} \tag{26}$$

Coriolis acceleration is derived from Eq. (27) in system 3 and then from Eq. (28) in system 1.

$$a_{cor} = \left(2(T_1^3\ \omega_s) \times T_2^3 V_{rel_2}\right) \tag{27}$$

$$a_{cor_1} = T_3^1\left(a_{cor_3}\right) = T_3^1\left(2(T_1^3\ \omega_s) \times T_2^3 V_{rel_2}\right) \tag{28}$$

Kinetic Analysis of sprung and unsprung mass

Considering Newton and momentum equations for angular motion, the correlations for sprung and unsprung mass are given below.

$$\sum F_G = m_s\, a_G \tag{29}$$

$$\sum F_o = m_u\, a_o \tag{30}$$

$$\sum M_G = \dot{H}_G^s \tag{31}$$

$$\sum M_o = \dot{H}_o^u \tag{32}$$

Where

$$H_o^u = [I_o^u]\ \omega_u \tag{33}$$

$$\dot{H}_o^u = [I_o^u]\ \alpha_u + \omega_u \times H_o^u \tag{34}$$

$$H_G^s = [I_G^s]\ T_1^3 \omega_s \tag{35}$$

$$\dot{H}_G^s = [I_G^s]\ T_1^3 \alpha_s + T_1^3 \omega_s \times H_G^s \tag{36}$$

$$[I_o^u] = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{37}$$

$$[I_G^s] = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \tag{38}$$

As the geometric characteristics have a great effect on wheel controllability, it is necessary to derive the correlations between Vehicle geometric characteristics and wheel angles. So, according to Fig. 5 and the angle between the x- axis and each wheel, Eq. (28) through (31) is derived.

$$\theta_{fl} = \tan^{-1}\left|\frac{(V_{fl})_y}{(V_{fl})_x}\right| \tag{39}$$

$$\theta_{fr} = \tan^{-1}\left|\frac{(V_{fr})_y}{(V_{fr})_x}\right| \tag{40}$$

$$\theta_{rl} = \tan^{-1}\left|\frac{(V_{rl})_y}{(V_{rl})_x}\right| \tag{41}$$

$$\theta_{rr} = \tan^{-1}\left|\frac{(V_{rr})_y}{(V_{rr})_x}\right| \tag{42}$$

Also, according to Fig. 5, the velocity of each wheel can be obtained.

$$V_{fr} = V_o + \omega_u \times \left(bi_1 + \frac{t}{2}j_1\right) \tag{43}$$

$$V_{fl} = V_o + \omega_u \times \left(bi_1 - \frac{t}{2}j_1\right) \tag{44}$$

$$V_{rr} = V_o + \omega_u \times \left(-ci_1 + \frac{t}{2}j_1\right) \tag{45}$$

$$V_{rl} = V_o + \omega_u \times \left(-ci_1 - \frac{t}{2}j_1\right) \tag{46}$$

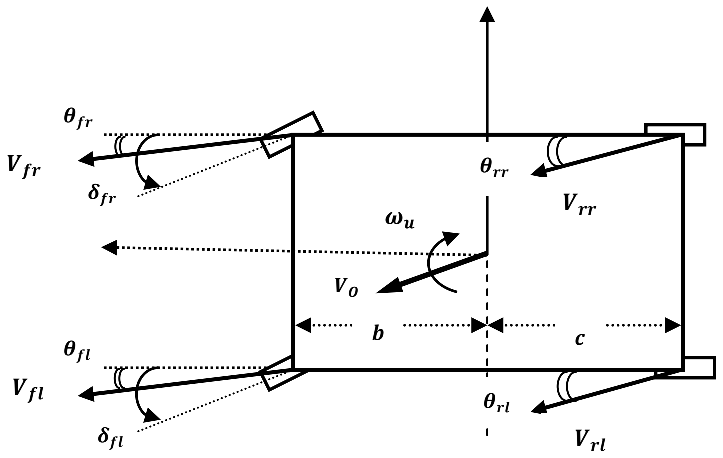


Fig. 5: Kinematics of a vehicle with yaw rotation

Considering the stir angels and also the angels of wheel velocity direction and a combination of Eq. (43) through (51), some correlations are driven for the slip angel of the wheels Eq. (52) through (55). Slip angels are defined as below.

$$\alpha_{fl} = \delta_{fl} - \theta_{fl} \tag{48}$$

$$\alpha_{fr} = \delta_{fr} - \theta_{fr} \tag{49}$$
$$\alpha_{rl} = -\theta_{rl} \tag{50}$$
$$\alpha_{rr} = -\theta_{rr} \tag{51}$$

When the slip angels are obtained, the lateral forces exerting on the wheels can be calculated. These forces are so important in stability and instability of the Vehicle in turns.

$$y_{fl} = C_{\alpha l}\alpha_{fl} \tag{52}$$
$$y_{fr} = C_{\alpha r}\alpha_{fr} \tag{53}$$
$$y_{rl} = C_{\alpha l}\alpha_{rl} \tag{54}$$
$$y_{rr} = C_{\alpha r}\alpha_{rr} \tag{55}$$

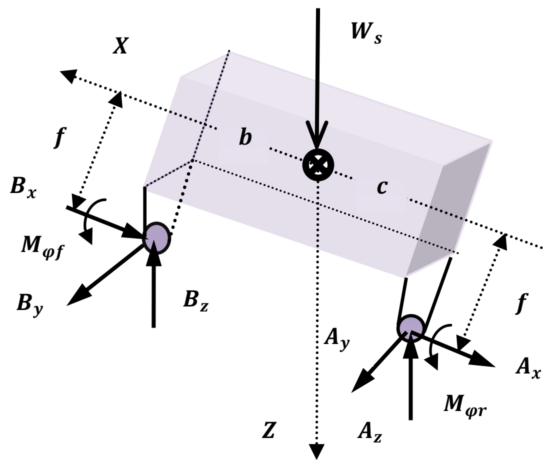


Fig. 6: sprung mass free body diagram

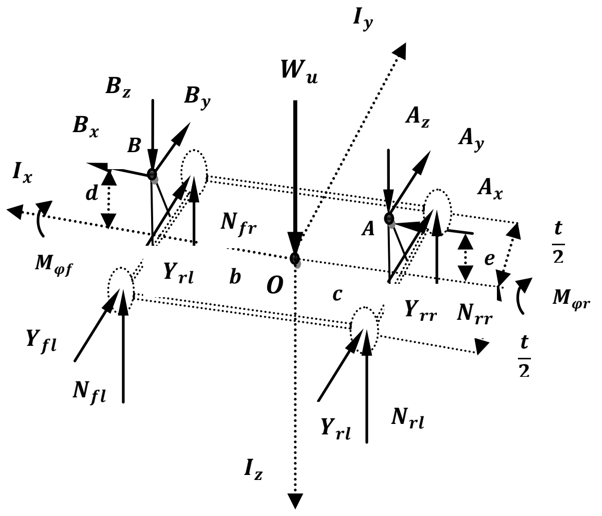


Fig. 7: unsprung mass free body diagram

Using the free body diagrams given in Figs. 6, 7 and Eqs.(28) - (31), the below correlations are driven for forces and torque with respect to points O and G for sprung and unsprung mass.

$$\sum F_o = (A_x + B_x)i_1 + \left(A_y + B_y + y_{fl} + y_{fr} + y_{rl} + y_{rr}\right)j_1 + (W_u - N_{fl} - N_{fr} - N_{rl} - N_{rr} + A_z + B_z)k_1 \quad (56)$$

$$\sum M_o = \left((N_{fl} + N_{rl} - N_{fr} - N_{rr})\frac{t}{2} + B_y d + A_y e + M_{\varphi f} + M_{\varphi r}\right)i_1 + \left((N_{fl} + N_{fr} - B_z)b - (N_{rl} + N_{rr} + A_z)c - A_x e - B_x d\right)j_1 + \left((y_{fl} + y_{fr} + B_y)b - (y_{rl} + y_{rr} + A_y)c\right)k_1 \quad (57)$$

$$\sum F_G = (-A_x - B_x)i_3 + \left(-A_y - B_y\right)j_3 + (W_s - A_z - B_z)k_3 \quad (58)$$

$$\sum M_G = \left((A_z + B_z)h_g \sin\varphi(t) + A_y\left(h_r + h_g\cos\varphi(t) - e\right) + B_y\left(h_r + h_g\cos\varphi(t) - d\right) - M_{\varphi f} - M_{\varphi r}\right)i_3 + \left(-A_z c + B_z b - A_x\left(h_r + h_g\cos\varphi(t) - e\right) - B_x\left(h_r + h_g\cos\varphi(t) - d\right)\right)j_3 + \left(A_y c - B_y b + (-A_x + B_x)h_g \sin\varphi(t)\right)k_3 \quad (59)$$

Considering the known and unknown parameters of above equations, it seems that more equations are required. Therefore, each axel is considered separately and the correlation between its longitudinal and torsional springs is derived by finding the torque with respect to its rolling center. Considering torsional springs, the vertical vibrations and motions may be ignored. So, the summation of forces exerting on each wheel can be considered equal to zero. Writing Newton second law for a case in which longitudinal springs are used will result in below equations.

$$\sum F_z = 0 \quad (60)$$

$$N_{rl} = F + \Delta F_r \quad (61)$$

$$N_{rr} = F - \Delta F_r \quad (62)$$

The below equation can be obtained using the Eq. (61) and (62).

$$N_{rl} - N_{rr} = 2\Delta F_r \quad (63)$$

If torsional springs is used instead of longitudinal spring, the torque form them should be the same as longitudinal springs. So taking torque around roll axis and equating their effects we have:

$$(F + \Delta F_r - F + \Delta F_r)\frac{t}{2} = M_{\varphi r} \quad (64)$$

$$\Delta F_r = \frac{M_{\varphi r}}{t} \quad (65)$$

The Eq. (66) shows the relation between vertical forces exerted in each wheel and torsional torque from torsional spring.

$$N_{rl} - N_{rr} = 2\frac{M_{\varphi r}}{t} \quad (66)$$

The same procedure is used for front axis.

$$N_{fl} - N_{fr} = 2\frac{M_{\varphi f}}{t} \quad (67)$$