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

IMECE2011-63613

DERIVATION OF RIGID BODY ANALYSIS MODELS FROM VEHICLE ARCHITECTURE ABSTRACTIONS

Rostyslav Lesiv
University of Louisville,
Louisville, KY, USA,
rmlesi01@louisville.edu

Glen Prater
University of Louisville,
Louisville, KY, USA,
gprater@louisville.edu

Gary Osborne
University of Louisville,
Louisville, KY, USA,
gary.osborne@louisville.edu

David Lamb
U.S. Army RDECOM-TARDEC,
Warren, MI, USA,
david.lamb@us.army.mil

Matthew Castanier
U.S. Army RDECOM-TARDEC,
Warren, MI, USA,
matt.castanier@us.army.mil

ABSTRACT

Vehicle analysis models of every type have their basis in some type of physical representation of the design domain. Rather than describing three-dimensional continua of a collection of components as is done in detail-level CAD models, an architecture-level abstraction describes fundamental function and arrangement, while capturing just enough physical detail to be used as the basis for a meaningful representation of the design, and eventually, analyses that permit architecture assessment. The design information captured by the abstractions is available at the very earliest stages of the vehicle developing process, so the model itself can function as a "design space for ideas". In this paper we describe a generalized process for analysis model extraction from vehicle architecture abstractions, and then apply that process to the specific case of rigid body response models. We also discuss implementation of a rigid body analysis engine that forms part of the analysis suite of a software package supporting all aspects of vehicle architecture design.

INTRODUCTION

Rigid body computer models can help predict the mechanical characteristics and performance of a vehicle design prior to fabricating and testing prototypes, resulting in shortened design cycles and reduced costs. While the engineering value of such analyses is very well established, current vehicle development methodologies have an inherent limitation. High-precision analysis models require topologically accurate geometric descriptions as a basis for abstraction. CAD solid models are generally a product of the detail design phase, and are not available during conceptual design. Architecture design is inherently conceptual in nature, qualitative rather than quantitative, and based upon precedent and designers' experience-based intuition. As a result, computer analyses tend to play a nominal role in vehicle concept development and assessment. Optimization based upon detailed CAD models tends to focus on detail level features rather than the fundamental vehicle architecture. There is a compelling need for vehicle modeling and analysis methodologies supporting the quantitative assessment and optimization of vehicle architecture design early in the design process.

Kojima [1] has proposed "First Order Analysis" (FOA), the analysis tool developed to execute preliminary analyses simultaneously with the model creation. The author has described the evolution of the car design process from the conventional approach to the new development procedure that incorporates concept modeling stage. Comparing to the conventional development procedure, expensive iterative evaluation of prototypes can be avoided, hence significantly reducing the cost and time for making prototypes. Hou at al. [2] has developed the ACD-ICAE (auto-body concept design-intelligent computer aided engineering) software suitable for the concept design phase of vehicle development. The tool for auto-body modeling, analysis and optimization permits quick creation of a geometric model for a conceptual vehicle design, as well as generating the FEM. Many types of auto-body templates have been integrated into the software, and template geometry can be modified by changing the control points. Another concept level analysis tool has been proposed by Schelkle and Eslenhans [3]. A structural optimization procedure is divided into three steps: topology optimization, which serves to find out where material should be located; parametric concept design, used for the layout optimization to

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
17 JUN 2011	**Journal Article**	**17-06-2011 to 17-06-2011**

4. TITLE AND SUBTITLE
DERIVATION OF RIGID BODY ANALYSIS MODELS FROM VEHICLE ARCHITECTURE ABSTRACTIONS

5a. CONTRACT NUMBER
W56HZV-04-C-0314

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Matt Castanier; David Lamb; Rostyslav Lesiv; Glen Prater; Gary Osborne

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Louisville,2301 S. 3rd Street,Louisville,KY,40208

8. PERFORMING ORGANIZATION REPORT NUMBER
; #21932

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army TARDEC, 6501 E.11 Mile Rd, Warren, MI, 48397-5000

10. SPONSOR/MONITOR'S ACRONYM(S)
TARDEC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
#21932

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Paper submitted to ASME 2011 INTERNATIONAL MECHANICAL ENGINEERING CONGRESS AND EXHIBITOIN (IMECE)

14. ABSTRACT

Vehicle analysis models of every type have their basis in some type of physical representation of the design domain. Rather than describing three-dimensional continua of a col-lection of components as is done in detail-level CAD models, an architecture-level abstraction describes fundamental function and arrangement, while capturing just enough physical detail to be used as the basis for a meaningful representation of the design, and eventually, analyses that permit architecture assessment. The design information captured by the abstractions is available at the very earliest stages of the vehicle developing process, so the model itself can function as a ?design space for ideas?. In this paper we describe a generalized process for analysis model extraction from vehicle architecture abstractions, and then apply that process to the specific case of rigid body response models. We also discuss implementation of a rigid body analysis engine that forms part of the analysis suite of a software package supporting all aspects of vehicle architecture design.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**10**	

prepare the body concept model; and stochastic concept assessment, which assists in shape, size and material optimization.

RIGID BODY ANALYSIS MODEL DERIVATION

At the architecture/conceptual level, vehicle abstraction is of necessity fundamentally different than a detail-level abstraction. In general, these abstractions must include general shape and layout, major spatial features (nominal volumes, in particular), component and subsystem connections (which can be quite simple relative to a detailed CAD model), and inertia properties. They should describe critical subsystems, including, but not limited to primary body structure, suspensions and powertrain, energy storage and transfer elements, and the human operator and passengers.

Vehicle abstraction is best represented as a hierarchy ordered according to function, and to a certain extent, projected physical complexity. On the very top level of the abstraction hierarchy are assemblies and assembly connections. Assembly connections store the connectivity information between assemblies. Further, each assembly can include beam, panel, rigid components, and component connections, see Fig. 1. Like assembly connections, component connections provide the connectivity specification between components inside an assembly. Components and connectivity information of each assembly are stored in the graph data structure, where components are represented by nodes and component connection by edges.

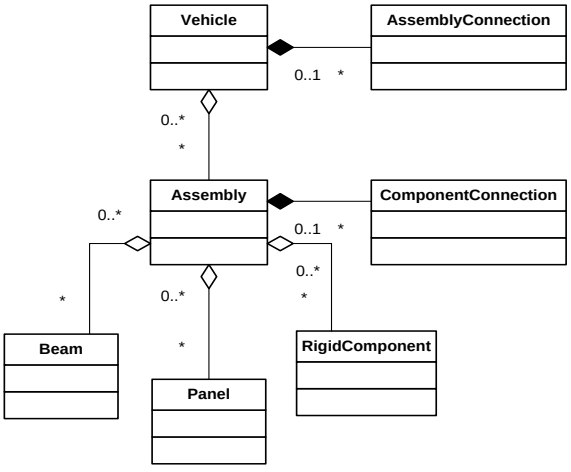

Figure 1. UML class diagram of the vehicle abstraction and aggregation and composition relationships that make up a portion of the vehicle hierarchy (properties and methods not shown).

Provided a suitable design abstraction can be formulated, the architecture model can be used as a starting point for the derivation of a rigid body analysis model. Associated analysis algorithms should be able to determine rigid body structural characteristics, forced time domain response, and frequency response it terms of both natural frequencies and modes, and frequency response transfer functions. Implementation of this list of analysis requirements mandates the use of preprocessing algorithms for determining gross dimensions, areas, and inertia properties (mass, mass centers, mass moments of inertia). Geometrically, vehicle abstraction consists of parametric curves and surfaces, which are later in the design process designated as representations of structural components such as beams and panels. Volumetric enclosures can also be designated as rigid components. During this process, material-related properties are applied for each newly created component in the model. Higher dimensional properties (gross vehicle measurements, in particular) and inertia properties are calculated based on the material type and geometric specifications.[1]

Algorithmic derivation of an architecture-level rigid body model begins with discretization of the features into components, subsystems, and connections included in the geometric abstraction, with nonstructural elements either embodied only through their inertia contributions, or ignored entirely. In the rigid body analysis model assemblies are represented as either rigid bodies or sets of rigid bodies, depending upon the internal connectivity. For example, the cab, frame, and wheels are all single rigid bodies extracted from their corresponding assemblies, whereas most suspensions contain multiple rigid bodies based on the internal rigid components and hinge or ball-joint connections. The RigidBody class data structure contains inertia properties and current information about its position, velocity, acceleration, and active forces. Inertia parameters are extracted from the corresponding structural or rigid component in the model.

Like the vehicle abstraction, the rigid body model is stored in an undirected graph data structure, with each node of the graph representing a rigid component, assembly or subassembly (class RigidBody), and edges representing component or assembly connections (class Edge). Figure 2 illustrates the structure of RBGraph.

Interface- and assembly-related properties provide connectivity specifications between components and assemblies, with appropriate joint models used for the derivation of rigid body constraints. Many common joint types, including spring, damper, ball-joint, hinge, motor, slider, and limit, are supported in the engine. The rigid body graph does not contain a rigid constraint type. Instead, we combine any rigidly connected bodies into a single composite body for the purposes of the simulation. The model formulation algorithm loops through all rigid edges in the graph and combines bodies until no rigid edges remain. For newly created composite bodies, the center of mass and inertial tensor are calculated based on the inertia properties of bodies that were combined. The model formulation algorithm runs during the preprocessing stage,

[1] The classes used to calculate these vehicle dimensions and inertia properties represent the most fundamental of the analysis modules used to evaluate the vehicle architecture being designed.

thus computationally expensive constraint resolution of the six degrees of freedom is not needed for the rigid joints.

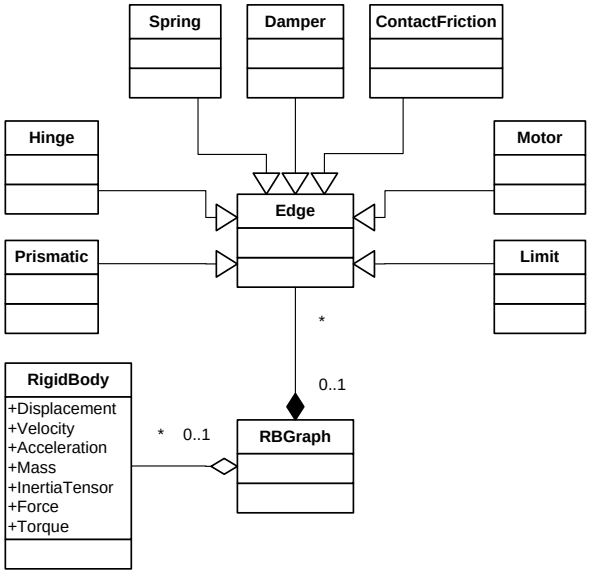

Figure 2. UML diagram of the RBGraph class.

Based on these connection types we have developed a number of predefined templates that can be used to quickly implement common suspension configurations in an architecture model. Templates are available for dependent and independent swing arm suspensions, Macpherson struts, live axles, walking tandem suspension, and several other configurations. We also provide support for generalized dependent and generalized independent suspensions that do not model suspension details, but instead, model basic functionality, including compliance properties.

ANALYSIS ENGINE DESIGN

The rigid body analysis engine is implemented as a module of a larger suite analysis tools, and it follows design patterns common to all analyses included in the package. The UML class diagram of Fig. 3 depicts this shared structure. Analysis, Engine and PostProcessor are all abstract classes, and interaction between them also implemented on the abstract level through their interfaces. This allows us to implement a simulation framework that is independent of the underlying simulation method, and where strategy of the implementation can be easily changed by switching to the different concrete class.

In addition to being capable of rapid solution times, a viable analysis suite should be capable of preparing architecture analysis models for full vehicles, a primary subsystem (a cab or rolling chassis, for example), and in some cases singular features or components. It must have access to visual renderings that depict the vehicle architecture's physical layout while allowing designer to interact with both the physical design space and analysis results. Finally, individual modules of the analysis suite should be algorithmically integrated.

Rigid body analysis can be based upon discrete system models with lumped mass/inertia, damping, and stiffness elements. Mathematically, the model will consist of coupled ordinary differential and algebraic equations. Complementary analyses for such models include determination of rigid body natural frequencies chracteristics, and time domain response.

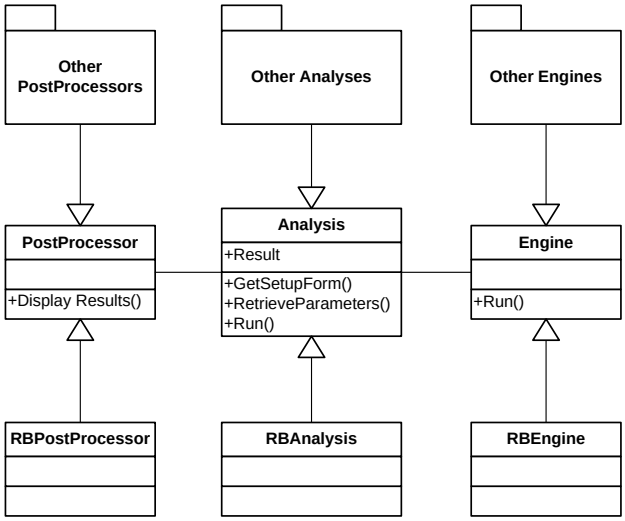

Figure 3. Rigid body module structure and its place in the software.

The entire rigid body module consists of the three main classes: RBAnalysis, RBEngine and RBPostProcessor. RBAnalysis is the class responsible for the interaction of RB module with the rest of the software. Inside RBAnalysis we store input parameters and results of the analysis. The class is also responsible for constructing rigid body graph and combining any rigidly connected components. RBEngine is the actual multibody system dynamics engine that controls all subclasses responsible for the multibody system dynamics mathematical model calculations. This class defines the interaction between its subclasses, see Fig. 4. Position, velocity, and acceleration updates for each body in the system are performed in RBEngine. More details on the structure and the mathematical background of the engine are presented in the next section.

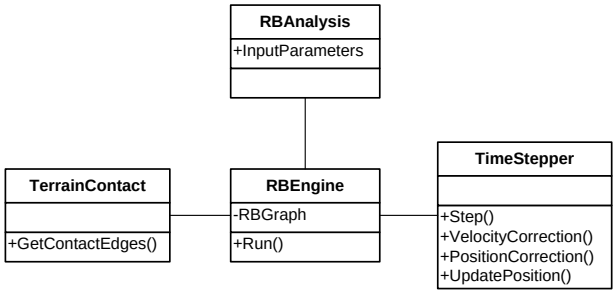

Figure 4. UML diagram of RBEngine interactions with other classes.

RBPostProcessor is the class responsible for the visual representation of results. The simulation results are normally depicted as an animation of the vehicle traversing the input terrain profile. They can also be shown as plots presenting the position, velocity and acceleration of a specified system body as a function of time, or as a state rendering, which represents a rendering of the analyzed vehicle at a given moment in time.

Another important class seen in Fig. 4 is TimeStepper, which provides the algorithmic functionality needed to advance numerical integration in time. The class is responsible only for the data flow within one time step, see Fig. 5.

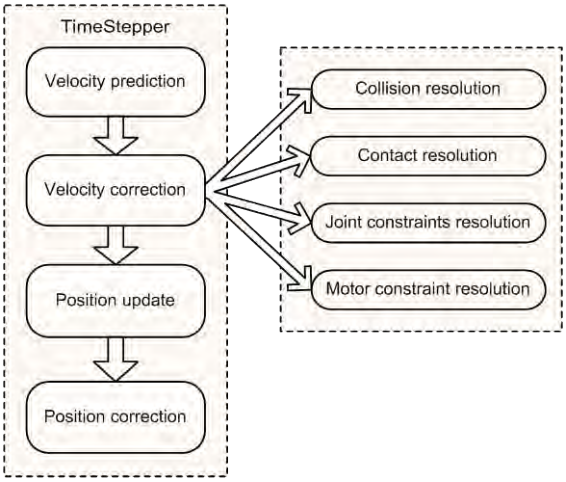

Figure 5. Data flow diagram for the TimeStepper class.

For each time step the algorithm performs checks to see if contact exists between individual wheels and the terrain. Collision detection and time of impact solution methods are implemented in the class TerrainContact. Any such contacts found are stored in RBGraph as ContactFriction type of edges.

It is often useful to understand the rigid body modal characteristics of a vehicle architecture design. Knowing the system natural frequencies, damping ratios, and response patterns allows the designer to optimize the system behavior by tuning spring and damper characteristic properties. Accordingly, a frequency analysis has been implemented as a part of the rigid body analysis engine, see Fig. 6.

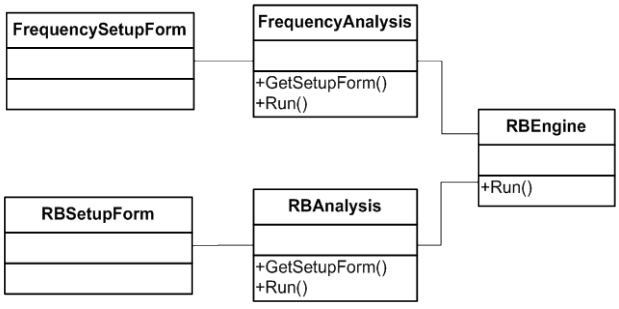

Figure 6. Relationship of the frequency analysis classes to the rigid body analysis.

THEORETICAL BACKGROUND

The main building blocks of the time domain rigid body solution engine are the system equations of motions and associated constraints and assumptions, the numerical solution approach, the time stepping algorithm, and the collision detection scheme, see Fig. 7.

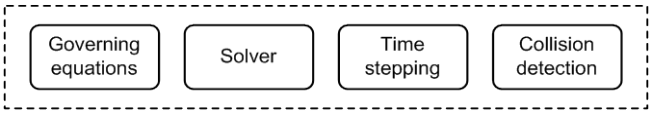

Figure 7. Primary functional blocks of the rigid body engine.

There are a number of possible approaches to the rigid body simulations. The earliest formulations were developed using an acceleration-based approach in the works of Pfeiffer et al. [4], Featherstone [5], and Baraff [6]. In the acceleration-based approach, all equations are written in terms of forces and accelerations. The main drawback to this method is that constraints are usually imposed in terms of position of a body, not on acceleration. Converting position constraints to acceleration constraints is complex, often producing large equations, and computationally expensive.

An alternative to the acceleration approach is the use of velocity-based equations. In a velocity-based approach, equations are formulated in terms of impulses and velocities. A velocity-based approach was selected as the best option for this application, as it combines a relatively simple set of equations, a number of well-proven solution methods, and a fast run time. Two families of algorithms typically in use with velocity-based approach are constraint-based [7] and impulse-based [8]. The most popular and one used in our work is constraint-based paradigm.

Having established a formulation approach for the equations of motion, we must now choose a solver. It is well know that the formulation of the constrained rigid body problem represents an example of a mixed linear complementary problem (MLCP). There are number of solution methods available for solving MCLPs, and these can be divided into two groups: exact and iterative algorithms. For this application, an iterative solver is the best option, since exact algorithms are much slower. A scheme for the numerical integration of the equations of motion also has to be chosen. Popular methods used in rigid body simulations are Verlet or explicit, semi-implicit, and implicit Euler schemes. Euler methods are first order techniques and converge slowly, therefore, we chose to use a Verlet method. Finally, collision detection between wheels and terrain is handled using a Ray casting method.

Mixed Linear Complementary Problem Formulation

In three-dimensional space, a rigid body position can be uniquely specified by 6 coordinates. Convenient reference fame uses three variables to provide the location of the body's center

of mass and three others to provide successive Euler angles. The Newton-Euler equations of motion of a single rigid body may be stated as follows:

$$\mathbf{M}\dot{\vec{\mathrm{v}}} = \vec{f}_e, \tag{1a}$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{m} \\ \mathbf{I} \end{bmatrix}, \quad \vec{\mathrm{v}} = \begin{bmatrix} \vec{v} \\ \vec{\omega} \end{bmatrix}, \quad \vec{\mathrm{f}}_e = \begin{bmatrix} \vec{f} \\ \vec{\tau} - \vec{\omega} \times \mathbf{I}\vec{\omega} \end{bmatrix} \tag{1b-d}$$

Here **m** and **I** are the mass matrix and inertia tensor, $\vec{f}$ and $\vec{\tau}$ are the vector of the resultant force and torque acting on the mass center, and $\vec{v}$ and $\vec{\omega}$ are the linear and angular velocity vectors for the mass center. In addition to external forces, a rigid body is often subjected to contact forces $\vec{f}_c$, joint reaction forces $\vec{f}_j$, and friction forces $\vec{f}_f$. Equation 1 then becomes:

$$\mathbf{M}\dot{\vec{\mathrm{v}}} = \vec{f}_e + \vec{f}_c + \vec{f}_j + \vec{f}_f \tag{2}$$

The contact, joint, and friction forces are modeled using Lagrange multipliers:

$$\vec{f}_c = \mathbf{J}_c^T \vec{\lambda}_c, \quad \vec{f}_j = \mathbf{J}_j^T \vec{\lambda}_j, \; \vec{f}_f = \mathbf{J}_f^T \vec{\lambda}_f, \tag{3}$$

where **J** and $\vec{\lambda}$ are the Jacobians and vectors of the Lagrange multipliers, respectively.

Using the above defined notations, the rigid body system with joints, contact and friction can be formulated as a mixed linear complimentary problem (MLCP):

$$\mathbf{M}\left(\vec{\mathrm{v}}_1 - \vec{\mathrm{v}}_0\right) = \mathbf{J}_u^T \vec{\lambda}_u + \mathbf{J}_b^T \vec{\lambda}_b + \vec{f}_e \tag{4}$$

$$\mathbf{J}_b \vec{\mathrm{v}} = 0, \; -\infty \geq \vec{\lambda}_b \geq +\infty \tag{5a}$$

$$\mathbf{J}_u \vec{\mathrm{v}} \geq 0, \; \vec{\lambda}_u \geq 0 \tag{5b}$$

Equations (4) and (5) represent a set of differential algebraic equations (DAE) that governs the dynamics of constrained multibody system. Here $\mathbf{J}_b$ and $\mathbf{J}_u$ are the Jacobians representing bilateral and unilateral constraints, respectively. Solving for velocity $\vec{\mathrm{v}}_1$ and satisfying constraint Eq. (5a) and (5b) yields

$$\mathbf{J}_b \vec{\mathrm{v}}_0 + \mathbf{J}_b \mathbf{M}^{-1} \mathbf{J}_u^T \vec{\lambda}_u + \mathbf{J}_b \mathbf{M}^{-1} \mathbf{J}_b^T \vec{\lambda}_b + \mathbf{J}_b \mathbf{M}^{-1} \vec{f}_e = 0 \tag{6a}$$

$$\mathbf{J}_u \vec{\mathrm{v}}_0 + \mathbf{J}_u \mathbf{M}^{-1} \mathbf{J}_u^T \vec{\lambda}_u + \mathbf{J}_u \mathbf{M}^{-1} \mathbf{J}_b^T \vec{\lambda}_b + \mathbf{J}_u \mathbf{M}^{-1} \vec{f}_e \geq 0 \tag{6b}$$

The Projected Gauss-Seidel (PGS) method is the most popular iterative solver used in rigid-body simulation, offering a reasonable compromise between computational speed and accuracy [9]. PGS has a number of advantages relative to exact algorithms, including ease of implementation and accuracy that can be controlled by changing the number of iterations. Significant performance improvement is achieved when using constraint reaction caching, also called "warmstarting" [9]. In this scheme, the final constraint reaction set from the previous time step is used as the starting point for the current solution step. We chose to implement the PGS method in a matrix-free fashion as described by Catto [10]. In this formulation PGS is called a Sequential Impulse solver.

Contact and Friction Modeling

Contact and friction are among the MCLP constraints included in Eq. (5). Specifically, the requirement that

$$\mathbf{J}_c \vec{\mathrm{v}} \geq 0, \vec{\lambda}_c \geq 0, \text{ and } \mathbf{J}_f \vec{\mathrm{v}} \geq 0, \vec{\lambda}_f \geq 0, \tag{7a,b}$$

means that the normal velocity of a body cannot be negative, thus precluding penetration. Also, the friction force is limited by the normal contact reaction:

$$-\mu\vec{\lambda}_c \leq \vec{\lambda}_f \leq \mu\vec{\lambda}_c, \tag{8}$$

where μ is the coefficient of friction. The term $\mathbf{J}_c\vec{\mathrm{v}}$ is projection of the body contact point velocity on the surface normal $\vec{n}$:

$$\mathbf{J}_c \vec{\mathrm{v}} = \vec{n} \cdot \left(\vec{v} + \vec{r}^{\,g} \times \vec{\omega}\right) \tag{9}$$

Here $\vec{r}^{\,g}$ is the vector of the global moment arm of the body, connecting center of the mass of a body and its point of contact with a surface. Analogously, friction Jacobians have the following form:

$$\mathbf{J}_f^1 \vec{\mathrm{v}} = \vec{t}_1 \cdot \left(\vec{v} + \vec{r}^{\,g} \times \vec{\omega}\right), \tag{10a}$$

$$\mathbf{J}_f^2 \vec{\mathrm{v}} = \vec{t}_2 \cdot \left(\vec{v} + \vec{r}^{\,g} \times \vec{\omega}\right), \tag{10b}$$

where $\vec{t}_1$ and $\vec{t}_2$ are two mutually perpendicular vectors in the tangent plane of the terrain contact point such that $\vec{t}_1 \perp \vec{n}$ and $\vec{t}_2 \perp \vec{n}$.

The contact and friction Jacobians in matrix form are 6×1 row vectors:

$$\mathbf{J}_c = \begin{bmatrix} \vec{n} \\ \vec{n} \times \vec{r}^{\,g} \end{bmatrix} \tag{11}$$

$$\mathbf{J}_f^1 = \begin{bmatrix} \vec{t}_1 \\ \vec{t}_1 \times \vec{r}^{\,g} \end{bmatrix} \tag{12a}$$

$$\mathbf{J}_f^2 = \begin{bmatrix} \vec{t}_2 \\ \vec{t}_2 \times \vec{r}^{\,g} \end{bmatrix} \tag{12b}$$

Now it will be demonstrated how to solve separately for contact and friction.

$$\vec{v}_1 = \vec{v}_0 + \mathbf{M}^{-1} \mathbf{J}_c^T \vec{\lambda}_c \tag{13}$$

$$\mathbf{J}_c \vec{v}_0 + \mathbf{J}_c \mathbf{M}^{-1} \mathbf{J}_c^T \vec{\lambda}_c = 0 \tag{14}$$

The Lagrange multipliers that assure no penetration, are calculated as

$$\vec{\lambda}_c = -\mathbf{m}_c \left(\mathbf{J}_c \vec{v}_0 \right), \quad \mathbf{m}_c = \frac{1}{\mathbf{J}_c \mathbf{M}^{-1} \mathbf{J}_c^T}, \tag{15}$$

where $\mathbf{m}_c$ is often referred to as the effective mass. Similarly, in case of friction we have

$$\vec{\lambda}_f = -\mathbf{m}_f \left(\mathbf{J}_f \vec{v}_0 \right), \quad \mathbf{m}_f = \frac{1}{\mathbf{J}_f \mathbf{M}^{-1} \mathbf{J}_f^T} \tag{16}$$

The one should notice that this formulation only approximates the Coulomb friction law, while the original formula is non-linear: $\lambda_{f1}^2 + \lambda_{f2}^2 \le (\mu \lambda_c)^2$. On the practice also more accurate linearized models can be used, for example as in [11].

Impulse Solver

Wheel to terrain surface collision can be modeled using Newton's Law of Impact:

$$v_a = -e \cdot v_b, \tag{17}$$

where v_b and v_a are the wheel velocities normal to the surface of contact velocities before and after collision, and e is the restitution coefficient (e = 0 corresponds to fully plastic contact; e = 1 corresponds to fully elastic contact). To calculate normal velocity before impact, the contact Jacobian can be used. The vector of Lagrange multipliers is calculated as for the case of contact and friction, though the desired velocity b = v_a - v_b must be taken into account. Lagrange multipliers guaranteeing this velocity increment are calculated as

$$\vec{\lambda}_c = -\mathbf{m}_c \left(\mathbf{J}_c \vec{v}_0 + b \right) \tag{18}$$

Substitution in the Eq. (13) will now give the velocity $\vec{v}_1$ of the body after the impact.

Joint Modeling

The basic equation for a bilateral constraint is formulated as follows:

$$\mathbf{J}\vec{v} = 0 \tag{19}$$

As an example, we will demonstrate constraint modeling for a revolute joint, also known as a hinge, or pin joint. Hinge joints allow only one degree of freedom, specifically rotation about an arbitrary axis l. Thus we have five equations constraining the motion of two bodies. Three of these describe translational constraints (one vector equation for the X, Y, Z degrees of freedom):

$$\vec{v}_1 + \vec{r}_1^{\,g} \times \vec{\omega}_1 - \vec{v}_2 - \vec{r}_2^{\,g} \times \vec{\omega}_2 = 0 \tag{20}$$

The remaining two scalar equations preclude rotation:

$$\vec{k}_1 \cdot \vec{\omega}_1 - \vec{k}_1 \cdot \vec{\omega}_2 = 0 \tag{21a}$$

$$\vec{k}_2 \cdot \vec{\omega}_1 - \vec{k}_2 \cdot \vec{\omega}_2 = 0 \tag{21b}$$

Here $\vec{r}^{\,g}$ is the vector of the global moment arm of the body connecting center of the mass and anchor point, $\vec{v}_1$, $\vec{v}_2$, $\vec{\omega}_1$, $\vec{\omega}_2$ are the linear and angular velocities of body 1 and 2, respectively, and $\vec{k}_1$ $\vec{k}_2$ are orthogonal vectors such that $\vec{k}_1 \perp \vec{l}$ and $\vec{k}_2 \perp \vec{l}$. The entire Jacobian in matrix form becomes

$$\mathbf{J}_h = \begin{bmatrix} \mathbf{I} & \vec{r}_1^{\,g\times} & -\mathbf{I} & -\vec{r}_1^{\,g\times} \\ \mathbf{0} & \vec{k}_1^{\,T} & \mathbf{0} & -\vec{k}_1^{\,T} \\ \mathbf{0} & \vec{k}_2^{\,T} & \mathbf{0} & -\vec{k}_2^{\,T} \end{bmatrix}, \tag{22}$$

where $\mathbf{I}$ is a 3×3 identity matrix.

Mathematical formulations for many other types of constraints may be found in the open literature [12, 13]. In addition to hinge joints, our rigid body analysis engine supports several other lower pair joints, including limits, motors, prismatic (slider), and spherical joints. To deal with the problem of constraint "drifting" (the phenomenon when anchor points that connect two bodies drift apart due to the numerical error) we have used a post stabilization method described in the work of Cline and Pai [14].

Time Stepping Scheme

Implicit integration can provide computational savings by allowing large time steps without risk of instability; however, such large increments may result in a loss of accuracy and an inability to detect higher frequency response characteristics of a system. One result of the large step sizes is overly damped behavior of the rigid bodies being simulated. While it not as important for "qualitative" animation purposes, accuracy is quite important for mechanical simulation of a vehicle model. It is certainly possible to decrease the time step of an implicit method so it approaches the size required for stability of an explicit method, but this will make implicit method unusable, since an implicit method requires matrix inversion and is thereby more complex in implementation. Thus, for reason of accuracy and simplicity we prefer explicit methods.

The approach chosen for our analysis engine, Verlet integration, is a conditionally stable [15] second order explicit method with computational efficiency comparable to that of first order methods. The Courant stability criterion must be satisfied in order for the method not to diverge:

$$\Delta t \le \frac{2}{\omega_d} \tag{23}$$

Here ω_d is the highest damped modal natural frequency of the system.

Collision Detection and TOI Solver

The fundamental objectives of architecture concept modeling justifies using a somewhat simplified approach to rigid body collision detection. Specifically, only the collision of wheels with the terrain surface is considered. Ignoring

collision detection between the terrain and other bodies in the vehicle system considerably improves the simulation response time.[2] We used ray casting against a Minkowski sum of the wheel radius and the radius of the larger object involved in the collision, which in our case is the terrain [16]. Collision detection is now simplified to determination of the smallest distance from the center of the wheel to the above defined Minkowski sum.

A more complicated problem involves establishing the time of impact (TOI) for colliding bodies. Consider a situation where at the beginning of a time step there was no contact, and at the end of step Δt, penetration is detected. The algorithm must go back to a safe position at time $t_i \le t \le t_i+\Delta t$ where no contact has occurred. This algorithm is sometime called Conservative Advancement [13], since it iteratively comes closer to the contact until a certain threshold is reached.

Algorithm Implementation

Along with the rest of the vehicle architecture design software, the methods and algorithms embodied in this rigid body response engine have been implemented using the C# programming language and Microsoft Visual Studio .NET compiler. Microsoft's XNA Framework has been used as a 3D graphics API, permitting animation of the time domain response simulations.

RESULTS AND DISCUSSION

The fundamental result from the rigid body analysis module is the kinematic response of the rigid bodies making up the system. Specifically, the simulation yields translational and rotational velocities, positions, and accelerations for the cab, frame, and suspension mass centers. These results can be displayed as time domain response plots (cab velocity magnitude versus time, for example), or represented simultaneously as an animated rendering of the entire vehicle. The response of important non-centriodal points can also be calculated and displayed, as can connection forces. The instantaneous response can be compared with quaisi-static responses to emphasize the effects of system compliance and contact. Vehicles can be excited by specifying a torque-speed relationship from the powertrain. Transient or steady-state response can be evaluated. In addition to time-dependent powertrain inputs, the vehicle models can be excited through terrain profile inputs.

To demonstrate the rigid body analysis results we have built concept models for two different vehicles. The first of these models has architecture typical of a Class 3[3] (3,850 - 4,540 kg GVWR) light-duty commercial truck: four-wheel ladder frame, engine-forward crew cab and open cargo box, see Fig. 8a. The second model represents the architecture of a Class 7 (11,800 - 14,970 kg GVWR) heavy-duty commercial truck with three axles (two driven), a cab-over-engine layout, and no payload module, see Fig. 8b.[4]

For the light-duty truck we used Macpherson struts for the front suspension and dependent swing arm as a rear suspension[5], see Fig. 8a. Suspension compliance is controlled by tuning the spring stiffness coefficient and damping coefficient. The cab is modeled as single rigid body with rigidly attached seats. It is connected to the frame by mean of springs, dampers and prismatic joints. The cargo box is connected rigidly to the frame, thus these two bodies are analyzed as a single rigid body. In addition to the cab, frame, cargo box and payload, suspensions, and wheels, we included a compliantly mounted connected engine and a rigidly mounted fuel tank. For the Class 7 truck, we used a longitudinally-constrained live axle for the front suspension, and swing arm rear suspensions for both of the rear axles.

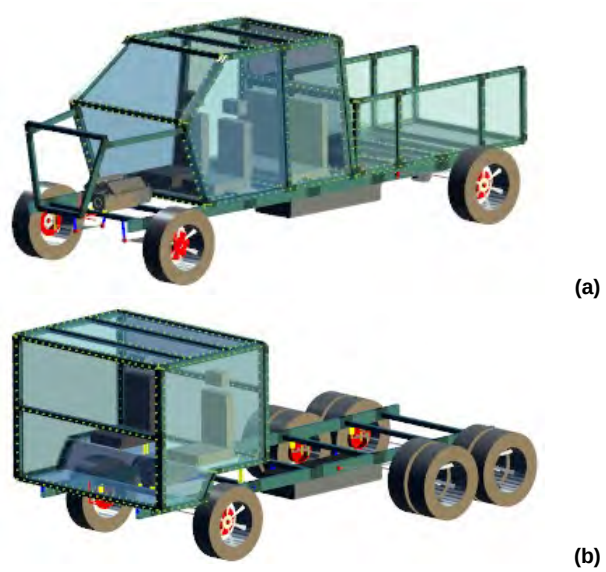

Figure 8. Vehicle concept model renderings for the example cases.[6] (a) Two-axle Class 3 truck with an engine-forward architecture, and (b) three-axle Class 7 truck with a cab-over-engine architecture.

[2] This approximation means what the simulation will not detect and respond to collision cases where the vehicle frame or other structure "bottoms-out" on a terrain feature. However, such interference will be visually apparent on an animation, and will show up on kinematic response plots.

[3] US Department of Transportation Federal Highway Administration (FHWA) commercial truck classifications based on the vehicle's gross vehicle weight rating (GVWR).

[4] The architecture models used for these examples and reproduced in Figure 8 do not represent the full range of architecture features that can be included in the architecture concept model and the derived rigid body analysis model. These models could have included closures, driver/passenger objects, additional geometric detail, and mass/inertia corrections to account for build-out weight.

[5] These suspension configurations were chosen for model validation purposes only, and are not typical of Class 3 trucks.

[6] Like the abstractions themselves, abstraction displays used with vehicle concept models use rendering queues that represent function, rather than topological geometry that does not exist at the concept level. In many cases, functional rendering results in depictions that are reasonably representative of physical appearance. Other visual queues, particularly those related to structural layout (location and configuration of assembly and major compliant joints, for example) are not as intuitive.

Figure 9. Class 3 truck traversing a 50 m long terrain profile with superimposed periodic bumps.

To investigate the rigid body response characteristics of the vehicle, we specified a two-dimensional (longitudinal) terrain profile, see Fig. 9[7], and applied a desired angular velocity (equivalent to a target speed) and maximum torque to the driving wheels. The maximum torque is calculated based on the properties of the engine defined during the architecture design/modeling process. Braking is simulated by specifying a maximum braking torque and minimum wheel speed of zero (to model the case of lock-up). Braking performance also depends on the road surface, road condition, and tire type, parameters that can also be specified. Simulation rate and accuracy is controlled by defining the time step, number of iterations for the solver, and number of sub-steps iterations for the springs and dampers. The last option is helpful in case of very stiff springs that can lead to solution stability due to the explicit time integration.

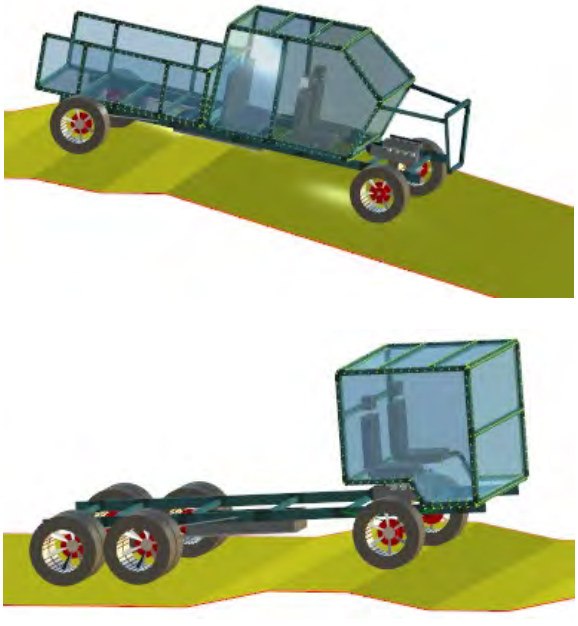

Figure 10. State rendering for the terrain response analysis showing (a) the Class 3 truck model clearing a dip in the terrain surface while driving down a 15 degree incline, and (b) the Class 7 model moving over a bump on a generally flat surface.

[7] Terrain profiles are currently specified using elevation singularity functions of various orders; however, the simulation algorithm can be adapted to support other, more complex approaches.

Figure 10 shows animation screen captures depicting vehicle state renderings of the example case architecture models as they traverse portions of different terrain profiles. While a multitude of time domain kinematic response plots can be generated, the plot of Class 3 cab vertical displacement in Fig. 11 is representative.

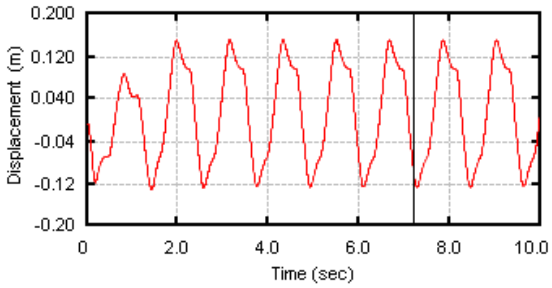

Figure 11. Class 3 truck cab displacement response while traversing a terrain profile periodic.

While a designer's understanding the rigid body time domain response of these models is critical, the frequency domain response can be important as well. Unfortunately, the system of differential equations governing the vehicle response, Eq. (4)-(5), can be highly nonlinear, and formulating an appropriate eigenproblem is quite complicated. The best way of acquiring modal characteristics or frequency response functions for such systems would be to linearize the system equations of motion, express them in standard matrix form, $\mathbf{M}\ddot{y}+\mathbf{C}\dot{y}+\mathbf{K}y=\mathbf{Q}$, and apply appropriate harmonic input/response assumptions. For more details on linearization methods, the interested reader is referred to the work of Negrut and Ortiz [17].

In our application we chose not to directly linearize the system equations; instead, we characterize the frequency domain response by using a discrete Fourier transformation (DFT). After running the rigid body simulation, acceleration responses of each body are used as input data for the DFT algorithm, which yields frequency response spectra for the bodies. DFT response spectra can permit identification of natural frequencies as high as the Nyquist frequency, particularly when transient inputs are specified. The frequency response spectrum (magnitude only) for the Class 3 light duty truck model is shown in Fig. 12.

The validly of our DFT/signal proccessing algorithm has been tested in three different ways. For the first test we compared the DFT scheme against the simple functions that have known sinusoidal components. Next, we built a very simple vehicle configuration consisting of a frame and four wheels connected to the frame with vertically positioned springs. For this configuration eigenvalues were calculated

analytically. For both cases we achieved good correlation between analytical results and numerical experiments. Finally, for a few very complex vehicle configurations we have visually inspected the cab and frame vibrations on the slowly animated models. The resulting lowest frequencies from the plot matched all the frequencies obtained by mean of visual inspection.

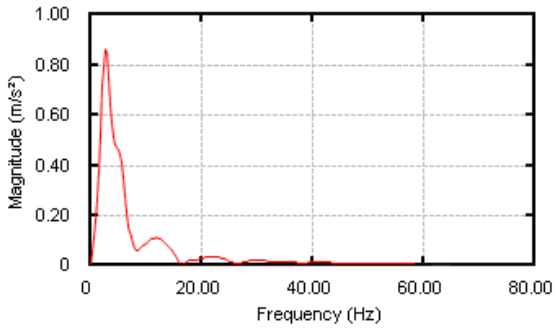

Figure 12. Cab acceleration spectrum magnitude for the Class 3 truck model.

CONCLUDING REMARKS

This paper has presented a description of the methodologies, basic data structures, and implementation algorithms for a rigid body analysis engine that makes up one portion of a comprehensive software package for vehicle concept modeling design and analysis.[8] As noted, the hierarchical organization of the vehicle architecture abstraction is based on the component and feature functionality. These abstractions contain geometric information sufficient to enable direct derivation of parametric models for many analyses appropriate for the conceptual design phase, including rigid body analysis.[9] Like the original vehicle abstraction, derived analysis models encompass the advantages (focus on primary functional attributes, small model size, direct coupling with the design process) and limitations of architecture concept models.

The multibody response engine is fully automated and capable of deriving the rigid body model from the vehicle model abstraction; however, it does have limitations. Flexible body elements are not supported, as is appropriate for rigid body analyses. The collision detection module identifies only contact between the wheels and terrain, while ignoring other interference. These restrictions aside, the rigid body analysis engine in its initial form is complete and validated. Future enhancement efforts will focus on:

- Continued improvement in computational and rendering speeds.
- Implementation of advanced terrain modeling routines, including support for transversely asymmetric profiles, and profiles with spatially varying friction and restitution coefficients.
- Development of standardized input functions and sampling rates optimized for generation of frequency response functions and modal parameters extraction (including use of standard signal processing techniques).
- Formulation of methods to characterize system linearity using homogeneity and additivity tests.
- Adding support for localized restitution coefficients greater than unity to represent vehicle response to explosions triggered by wheel contact.

[8] The working title of this package is Concept Modeling Tool Suite (CMTS).

[9] Other analysis types supported in the software include mass, kinematic, and geometric (MKG) properties, structural finite element analysis (including standard NVH calculations), powertrain performance, and ergonomic characteristics, to name just a few.

ACKNOWLEDGEMENTS

This research was sponsored in part by U.S. DoD contract no. W56HZV-04-C-0314, administered through the U.S. Army U.S. Army Tank-Automotive and Armaments Command (TARDEC). The authors wish to acknowledge the contributions of R.E. Meyers and Y. Yu for their assistance with response animation routines.

REFERENCES

[1] Kojima Y., 2000, "Mechanical CAE in automotive design," R&D Review of Toyota CRDL, 35, pp. 1-10.

[2] Hou W., Zhang H., Chi R., and Hu P., 2009, "Development of an intelligent CAE system for auto-body concept design," Int. J. Automotive Technology, 10, pp. 175-180.

[3] Schelkle E., and Elsenhans H., 2002, "Virtual vehicle development in the concept stage – current status of CAE and outlook on the future," Conference Proceedings for the 3rd Worldwide MSC.Software Aerospace Conference & Technology Showcase.

[4] Pfeiffer F., and Glocker C., 2004, "Multibody dynamics with unilateral contacts," Wiley-VCH.

[5] Featherstone R., 2008, "Rigid body dynamics algorithms," Springer.

[6] Baraff D., 1996, "Linear-time dynamics using Lagrange multipliers," ACM Transactions on Graphics (SIGGRAPH 1996).

[7] Potra F., Anitescu M., Gavrea B., and Trinkle J., 2006, "A linearly implicit trapezoidal method for integrating stiff multibody dynamics with contact, joints and friction," Int. J. Numer. Meth. Engng, 66, pp. 1079-1124.

[8] Mirtich B., 1996, "Impulse-based dynamic simulation of rigid body systems," Ph.D. thesis, University of California, Berkeley.

[9] Catto E., 2005, “Iterative dynamics with temporal coherence,” GDC 2005.

[10] Catto E., 2009, “Modeling and solving constraints,” GDC 2009.

[11] Stewart D., and Trinkle J., 1996, “An implicit time-stepping scheme for rigid body dynamics with Coulomb friction,” Int. J. Numer. Meth. Engng, 39, pp. 2673-2691.

[12] Shabana A., 2001, “Computational dynamics,” Wiley-Interscience.

[13] Erleben K., 2004, “Stable, robust, and versatile multibody dynamics animation,” Ph.D. thesis, University of Copenhagen.

[14] Cline M., and Pai D., 2003, “Post-stabilization for rigid body simulation with contact and constraints,” In Proceedings of IEEE International Conference on Robotics and Automation.

[15] Kacic-Alesic Z., Nordenstam M., and Bullock D., 2003, “A practical dynamics system,” ACM Transactions on Graphics (SIGGRAPH 2003).

[16] Bergen G., 2005, “Ray casting against general convex objects with application to continuous collision detection,” GDC 2005.

[17] Negrut D., and Ortiz J., 2006, “A practical approach for the linearization of the constrained multibody dynamics equations,” ASME J. Comput. Nonlin. Dyn, 1, pp. 230-239.