A SURVEY OF MULTIBODY DYNAMICS FOR VIRTUAL ENVIRONMENTS

R. Brent Gillespie

J. Edward Colgate

Department of Mechanical Engineering
Northwestern University
Evanston, Illinois 60208, USA
b-gillespie@nwu.edu colgate@nwu.edu

ABSTRACT

The field of computational dynamics is surveyed, focusing on issues relevant to the construction of a general purpose simulator with haptic display. The various formalisms available for generating dynamical models will be examined with regard to the form of the equations of motion which they produce. To render the effects of intermittent contact and other transient phenomena as driven by a human haptic explorer, a model is needed which is computationally efficient yet can handle changing kinematic constraints neatly. Models in dependent and independent coordinates produced with the Newton-Euler, Lagrange and Kane formalisms will each be examined in order to provide a vantage point from which an informed selection may be made from among the many tools now available in computational dynamics.

INTRODUCTION

The repertoire of a virtual environment simulator is determined by the modeling and simulation tools upon which it is based. Most virtual environments to date are limited to static objects or systems of dynamic bodies whose motions are restricted to a plane and whose interconnections are modeled by relatively soft springs. This paper addresses the incorporation of modeling and simulation tools from the field of multibody dynamics into a simulator, so that its repertoire may be expanded to include articulated systems free to move in space. Articulated systems are characterized by the presence of constraints, or imposed relationships among the configuration and/or motion variables of the model such as occur in mechanisms, linkages, vehicles, and the human body. The tools of multibody dynamics allow a constrained system to be modeled using a minimal or near-minimal (at the option of the analyst) set of state variables. The dynamics of a multibody constrained system will evolve in a manifold whose dimension is equal to the size of the state vector. This fact can result in significant computational savings and improved stability properties.

In a brief review, let us contrast the state of the art in interactive dynamics with the currently available tools for off-line simulation of dynamical systems.

Interactive dynamics. Most simulators with haptic display to date are based on what we call the 'Coupled Force Balance' scheme (to be described in more detail below). Basically, each link or contact between bodies is modeled as a spring or spring-damper connection. Each interaction force is computed as a function of the spring and damper displacements and the forward dynamics of each body is computed independently according to Newton's laws. Although bodies will in fact interpenetrate, violations can be held low by stiffening the springs. While justifiable in theory, in practice this approach is computationally expensive and the presence of highly disparate time constants creates so called 'stiff' differential equations which require specialized differential equation solvers. The presence of fast time constants is especially insidious with haptic display, where sampled data control of a haptic interface and the coupling of a human operator into the system can further incite nonphysical oscillatory behavior. The simulator with haptic diplay by Cadoz et al. [1] is an example of the Coupled Force Balance.

Though largely the result of unpublished work, numerous planar multibody systems such as double pendulums and pendulums on carts have been developed for demonstration of haptic interfaces. These systems generally do not allow for changing kinematic constraints. Yoshikawa et al. [2] have created virtual objects using an inverse multibody dynamics formulation. A simulator for articulated systems which does accommodate changing constraints has been developed by Gillespie [3] [4] but the set of constraint configurations must be known prior to simulation time. This somewhat restricted set of systems includes devices such as the piano action and other multibody mechanisms, but excludes systems with collisions between asymmetric ballistic objects.

Other human-in-the-loop simulators have been produced

in the fields of flight simulation, computer music and computer graphics. Rather than haptic interfaces, these simulators feature motion, audio or graphical display.

Flight simulators probably have the longest history in interactive dynamics. Even before the advent of the digital computer, the equations of motion of aircraft were solved using analog computers with human input of certain control variables. But of course flight simulators are used to train pilots to avoid changing contact conditions, whereas changing contacts are among the most salient features in haptic exploration of everyday objects. In computer music, the recent sound synthesis technology known as 'physical modeling' uses a musical instrument as the simulacrant [5]. In contrast to prior algorithmic sound synthesis techniques, physical modeling simulates a string, aircolumn, or even membrane using waveguide equations. Due to its basis in the wave equation, however, the domain of this technique is rather limited. The field which has produced the most general of packages for interactive simulation of dynamical objects is computer graphics, where the principles of multibody dynamics are applied to automate the job of the animator. Update rate requirements, however, are significantly less stringent in graphic display than haptic display. The techniques developed in computer graphics will be further reviewed in the body of the paper.

Off-line simulation. In contrast to the real-time simulators which are application specific, dynamicists have at their disposal a number of general purpose and in many cases commercialized packages for the solution of problems in multibody dynamics. These include DADS, ADAMS, WORKING MODEL, AUTOLEV, SD-EXACT, and many others. See [6] for surveys.

It is the intention of this paper to review the architecture of these packages and the methods upon which they are based to pave the way for the creation of a general purpose multibody dynamics package with haptic display. Beyond the challenge of real-time simulation, this survey will pay particular attention to the handling of constraint changes and singularities and the manner in which a haptic interface may be incorporated. As opposed to motion, audio, or graphical display, the issues of stability are of greater concern in haptic display.

Now, despite the fact that they all derive from a few fundamental principles of mechanics, a great number and variety of formalisms have been developed to solve problems in multibody dynamics. Specialized tools exist for the analysis of articulated bodies in open or closed loop chains, systems involving collisions, systems involving unilateral constraints, and systems involving friction. Since haptic exploration often involves all of these phenomena, a wide variety of tools from the field of multibody dynamics must be called upon for haptic display of dynamical objects. This paper will review each of the major formalisms for generating models: Newton-Euler, Lagrange's method, and Kane's method. A major purpose of this paper is to introduce the haptic community to Kane's method by comparing it to more widely known methods. Kane's method yields many advantages in terms of efficiency of formulation, computational efficiency of the resulting model, and ease of handling changing constraints, the latter two of which are quite relevant to real-time simulation.

This paper is organized around the following three steps for the creation of a touchable virtual object:

Step 1: A mathematical model is constructed by naming variables and parameters. Initial conditions are assigned to the variables as appropriate and values are assigned to the parameters according to the material or geometric properties for which those parameters stand.

Step 2: A principle of mechanics is used to formulate equations, called the equations of motion, which govern the variables in the model.

Step 3: Those equations are incorporated into an interactive simulator with haptic display.

These three steps may be considered a procedure for using the familiar relationship $\mathbf{f} = m\mathbf{a}$: First, collect together $\mathbf{f}, m$, and $\mathbf{a}$. Second, relate these with an = sign. Finally, wrap everything up in an ODE solver.

Before launching into the survey in earnest, the incorporation of an ODE solver into a haptic interface controller will be described. Section *Step 1* will discuss assembly of the elements of a model—paying particular attention to the description of system configuration, motion, and constraints. In section *Step 2*, the various formalisms for generating equations of motion will be surveyed, with an eye to the manner in which constraint equations are handled. Each form becomes the basis for a simulator design in section *Step 3*, and numerical methods for handling changing constraints will be discussed.

IMPEDANCE DISPLAY: THE VIRTUAL COUPLER

A haptic interface controller may be designed using one of two paradigms: impedance display (sensing motion and producing force) or admittance display (sensing force and producing motion). The limits of performance of each paradigm are still matters of active research, but current lore holds that the optimal implementation depends on the physical characteristics of the haptic interface itself (including its

sensor suite) and whether the virtual object impedance is dominated by inertia or compliance. Most virtual environments to date, whether static or dynamic, are implemented using impedance display—the work of Yoshikawa [2] and MacLean [7] being notable exceptions.

In this paper we treat impedance display: sensing motion and producing force. An image of the manipulandum handle or puck is placed in the virtual environment and attached to a certain virtual object through a spring and damper, together called the virtual coupler. (The attached virtual object is often isomorphic to the handle.) Thus coupled to the interface hardware, an ODE solver may operate on the *forward* dynamics simulation of the virtual environment, computing motion from applied force. As shown in Figure 1, the difference between the sensed hardware position x and attached virtual object position y is taken as the extension of the virtual coupler and used to produce the force f. This force is simultaneously displayed on the interface and used as the applied force in the forward dynamics simulation of the virtual environment. One of the primary advantages of the virtual coupler approach is the clean separation it imposes between hardware and simulator which allows a coupled stability analysis to be performed. In particular, a sampled data passivity analysis may be reduced to a discrete time passivity analysis [8]. Finally, we assume the use of a constant step-size explicit numerical integration routine with computational time reserved at each step for the detection and resolution of any possible collisions and, as discussed at length below, the handling of constraint changes.

STEP 1: MODEL CONSTRUCTION

The basis of a virtual environment model is a description, in parameters and variables, of a physical system of objects. In addition, certain relationships may be imposed among the parameters and variables. These relationships (called constraints) also become part of the system description. The model is complete with the assignment of values to each parameter and initial conditions to each variable. Note that the initial conditions must satisfy the constraints. Of course the model is not viable for simulation until those parameters and variables have been related to one another in a differential equation, but that topic will be reserved for the next section (*Step 2*). In the present section, the parameters, variables and constraint descriptions will simply be assembled in preparation for equation formulation.

The model may be further subdivided into three parts: the dynamic model, the geometric model, and the contact model. The dynamic model will be used to handle the configuration and the rate of change of configuration (motion) of the system. The geometric model will provide for the detection of collisions, and the contact model will provide the means for resolving collisions.

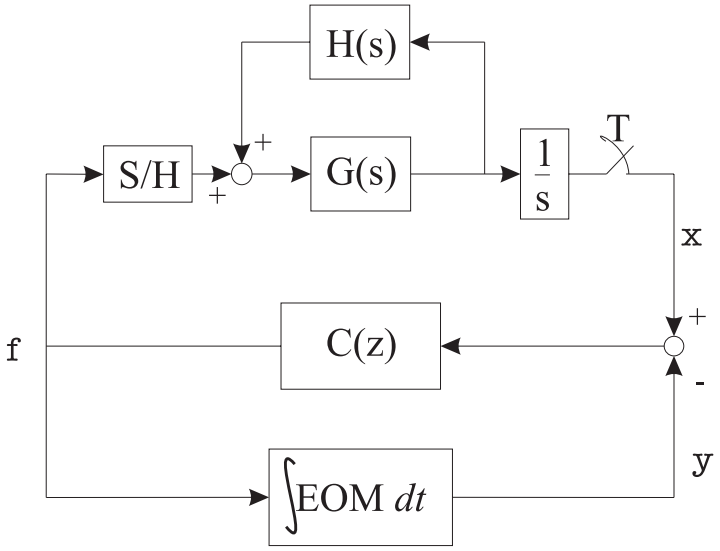

Figure 1: *Block diagram showing the incorporation of an ODE solver operating on the Equations of Motion (EOM) into a haptic interface controller. The interface hardware is denoted $G(s)$, the human user $H(s)$, and the discretely implemented virtual coupler $C(z)$.*

Dynamic Model

Newton's laws of motion will be used in Step 2 (formulate equations) to relate the parameters and variables describing internal and applied forces with those describing mass properties and accelerations. Given that there exist significant differences in the description of acceleration among the various formalisms, we will discuss kinematic descriptions in some detail. Note that Newton's laws produce second order differential equations when written in terms of configuration variables. Alternatively, Newton's laws may be written in terms of configuration *and* motion variables, producing *first* order differential equations. (We use the term 'motion' to refer either to the velocity of points or the angular velocity of bodies.) The configuration and motion variables together make up the state vector. As the reader will soon appreciate, the use of state variables can lead to significant advantages when formulating equations of motion.

Table 1 organizes three sets of configuration and motion variables into columns. Each column represents an option for a full set of state variables and corresponds to a particular equation formulation method: Coupled Force Balance, Lagrange's Method, and Kane's Method.

As indicated in the left column, position and orientation coordinates for each of ν bodies may be used as configura-

Table 1. Options for the state vector
ν: # bodies; M: # holonomic constraints; m: # nonholonomic constraints;
$n = 6\nu - M$: # generalized coordinates; $p = n - m$: # independent generalized speeds.

	Coupled Force Balance	Lagrange's method	Kane's method
Configuration Variables	position and orientation coordinates $(x, y, z, \theta_1, \theta_2, \theta_3)_i,\ (i = 1, ..., \nu)$	generalized coordinates $q_i,\ (i = 1, ..., n)$	generalized coordinates $q_i,\ (i = 1, .., n)$
Motion Variables	position and orientation coordinate derivatives $(\dot{x}, \dot{y}, \dot{z}, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3)_i,\ (i = 1, ..., \nu)$	generalized velocities $\dot{q}_i,\ (i = 1, ..., n)$	generalized speeds $u_i,\ (i = 1, ..., p)$
number of constraints	$M + m$	m	0

tion variables. For example, one may use the Cartesian coordinates x, y, z to locate each mass center and orientation angles $\theta_1, \theta_2, \theta_3$ to orient the principle axes of inertia of each body. When configuration is restricted to a plane, the description of orientation is trivial: it requires a single angular displacement. Orientation in space, however, requires three angular displacements, and the order in which these displacements are carried out and about which unit vectors must be specified explicitly. To complete the state description, the time derivatives of the position and orientation coordinates are commonly used.

To describe the position and orientation of ν bodies in space, 6ν variables will be needed. Likewise, to describe the system motion, 6ν motion variables will be needed, bringing the size of the state vector to 12ν.

Bodies of a system S which are articulated or in contact with one another over a period of time are subject to restrictions on the positions or orientations which they may occupy. These restrictions can be expressed as relationships among the position and orientation variables enumerated above and are called holonomic constraint equations or configuration constraints.

Generalized coordinates. In place of the position and orientation variables for each body, a set of configuration variables called generalized coordinates, which encapsulate the holonomic constraints, can be defined for the system. For example, if two bodies are connected by a revolute joint, only the joint angle is needed to describe their relative orientation. Let us use n to designate the smallest number of generalized coordinates needed to unambiguously describe the configuration of a system S of ν bodies in a reference frame N. The number n for S is then less than 6ν by the number M of holonomic constraint equations in effect: $n = 6\nu - M$.

Often, however, it is convenient to use a set of generalized coordinates which is not independent and to carry along a holonomic constraint equation as part of the system description. For example, a minimal set of generalized coordinates may be difficult to find, as in the case of closed kinematic chains. The use of a non-minimal set is also useful when a constraint is subject to change, as we shall see later. Let us continue to use n to designate the number of generalized coordinates, whether they make up an independent set or a dependent set (a set containing at least one dependency).

In Lagrange's method, the generalized coordinate derivatives, called generalized velocities, make up the remainder of the state vector as indicated in the second column of Table 1.

In addition to the holonomic constraints, there may exist nonholonomic constraints. Nonholonomic constraints are relationships among the generalized *velocities* and may arise for certain physical reasons such as rolling. The word 'nonholonomic' signifies that these constraints cannot be integrated to produce relationships among generalized coordinates. It is often convenient to treat a holonomic constraint by differentiating it with respect to time to produce a relationship among the generalized velocities. To refer to either a differentiated holonomic constraint or a nonholonomic constraint, we will use the term 'motion constraint'.

Generalized speeds. Kane's method also uses generalized coordinates to describe system configuration, but new variables known as generalized speeds are used rather than generalized velocities to describe system motion. On the way to generating expressions for the velocities of relevant points (and angular velocities of bodies) the analyst has the opportunity to define generalized speeds as independent linear combinations of the generalized velocities.

$$u_r = \sum_{s=1}^{n} Y_{rs}\dot{q}_s + Z_r,\ (r = 1, ..., n) \qquad (1)$$

where Y_{rs} and Z_r are functions of $q_i,\ (i = 1, ..., n)$ and possibly the time t. Reciprocal relations express the generalized

velocities in terms of the generalized speeds:

$$\dot{q}_s = \sum_{r=1}^{n} W_{sr} u_r + X_s, \ \ (s = 1, ..., n) \qquad (2)$$

where W_{sr} and X_s are functions of q_i, $(i = 1, ..., n)$ and possibly the time t. The equations of motion will take on a particularly compact (and thus computationally efficient) form with the effective use of generalized speeds. Note again, however, that the definition of generalized speeds is up to the analyst. Mitiguy [9] has developed some directives for the optimal choice of generalized speeds for articulated bodies in open chains.

Motion constraints may be expressed as relationships among the generalized speeds. Given m motion constraints, there will be m dependent generalized speeds, which may be expressed in terms of the remaining $p = n - m$ independent generalized speeds. Ordering the independent generalized speeds first, the motion constraints may be expressed :

$$u_r = \sum_{s=1}^{p} A_{rs} u_s + B_r, \ \ (r = p + 1, ..., n) \qquad (3)$$

where A_{rs} and B_r are functions of q_i, $(i = 1, ..., n)$ and possibly the time t.

To summarize the components of a dynamic model, we may choose 6ν configuration and orientation variables and identify M holonomic constraints and m motion constraints to be carried along as part of the description. Alternatively, we may use generalized coordinates q_i $(i = 1, ..., n)$ where $n = 6\nu - M$ and carry along m motion constraints. As yet another alternative, we may use generalized coordinates q_i $(i = 1, ..., n)$ and generalized speeds u_r $(r = 1, ..., p)$ where $p = n - m$ in which case we have embedded all the constraints and formed what is refered to as a set of 'independent coordinates'. For virtual environments, changing contact conditions are of interest for rendering, and these will most effectively be handled as constraint equations accompanying a nonminimal set of coordinates.

The modeling of gravitational forces and forces arising from the extension of springs or motion of dampers is relatively simple for objects commonly manipulated by a human, at least after capacitive and dissipative effects have been lumped into springs and dampers and mass properties have been described. The expression of friction forces, on the other hand, deserves special attention. Coulomb's law will often render the equations of motion implicit or give rise to existence and uniqueness problems. The familiar modeling issue of indeterminacy and over determination are closely related to the existence and uniqueness of a solution for the differential equations of motion. Existence and uniqueness of solutions for models with friction are topics of active research [10]. These issues may be side-stepped by adding spring damper couplers at the frictional contacts.

STEP 2: FORMULATE EQUATIONS

Each equation formulation method produces equations of motion in a particular form. Despite the fact that in the long run the simulated behavior will be the same, there exist large discrepancies in computational efficiency and ease of handling changing constraints among the various forms. Furthermore, the form of the equations of motion dictate to large extent the software architecture of the simulator. Figure 2 presents an overview of the various forms which may be taken on by equations of motion, categorizing these forms according to three general headings: 'Coupled Force Balance', 'Model in Dependent Coordinates', or 'Model in Independent Coordinates'. Roughly, models produced by application of the Newton-Euler equations are treated as Coupled Force Balances, Lagrange's method produces models in dependent coordinates, and Kane's method produces models in independent coordinates. Modifications can be made to each formulation procedure, however, in order that models belonging to other categories are produced, and this point will be treated in detail in section *Step 3* below. Let us investigate each standard form with regard to constraint handling and discuss its preparation for a numerical differential equation solver to be run in real-time.

The Coupled Force Balance

The forces and torques called for in the Newton-Euler equations include not only applied forces, but also contact forces between touching bodies and joint forces between articulated bodies. Each such interaction force must be computed by some means and then made available to the force/torque balance of each contacting body so that their resulting motion may be computed by solving the ODEs produced with the Newton-Euler equations. Essentially, the state of each body is evolved forward in time independently of the others; with interaction among bodies taking place only through the communication of forces. The Coupled Force Balance models are further broken down in Figure 2 by the method for computing the interaction forces.

Coupling through Spring-Damper Pairs. The simplest of the methods for determining the interaction force between bodies in the coupled force balance scheme consists of placing intervening springs or spring-damper couplers between contacting bodies. During those times interference between two bodies is detected, an interaction force is communicated to each according to the constituent equation of the intervening spring-damper coupler. This is often called the 'penalty method' since interference is penalized. To

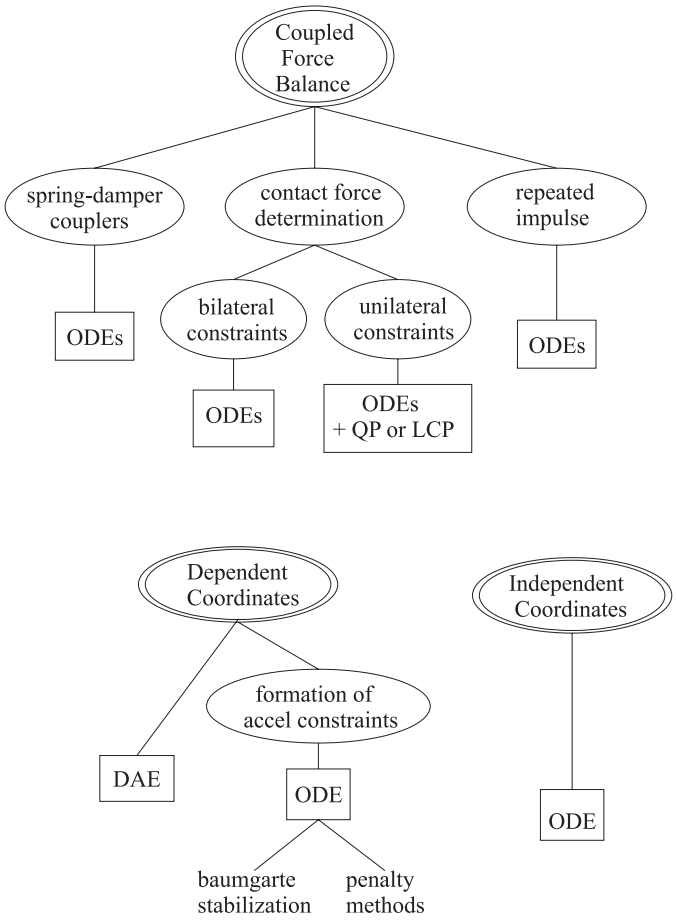

Figure 2: *Overview of the Formalisms for Equation of Motion Production*

mimic unilateral constraints, the spring-damper pair is removed when a change from compressive to tensile interaction force is detected. Many examples of this approach may be drawn from the computer graphics community [11], [12], [13]. Most demonstrations of touchable virtual objects to date fall into this category.

Further enhancing its simplicity, the penalty method alleviates the need for separate impulse resolution. The compliant coupler smoothes out impulsive forces. If the value of the stiffness coefficient of the couplers is increased in an attempt to approximate rigid body behavior, however, the system equations may become 'stiff' and numerically ill-conditioned.

Coupling through Bilateral Constraint Forces. By assuming that a bilateral constraint is locked into place when two bodies make contact, an inverse dynamics model may be set up and used to solve for the unknown interaction forces. Since the kinematic state is known (relative acceleration between contacting bodies assumed nil), the interaction forces may be determined from a linear system of equations [14] [15].

Coupling through Unilateral Constraint Forces. The inverse dynamics problem may be formulated together with inequality conditions stipulating non-interpenetration and repulsive contact forces as a linear complementarity problem (LCP) or quadratic programming problem (QP) [16] [17]. These unilateral contact force computation methods have been introduced to the computer graphics community by Baraff [18]. Treatments of friction between contacting bodies are presented by Lötstedt in [19] and Baraff in [17]. Lee, Ruspini and Khatib [20]. have applied the methods of Baraff to robotic simulation.

Coupling through Repeated Impulse. As an alternative to solving for and imposing contact forces to prevent interpenetration of contacting bodies, Hahn [21] has suggested using impulses, found through the solution of impulse-momentum equations. To handle resting contact, repeated impulses are used. Mirtich and Canny have extended the impulse technique [22] [23] and also combined it with constraint techniques [24]. Chang and Colgate explore the efficacy of an impulse technique for haptic display in [25].

With the advent of object-oriented programming techniques, the coupled force balance modeling approach is receiving a fair amount of attention since it is by nature modular. Most importantly from our viewpoint, changing kinematic constraints are handled quite easily in such models. A communication line between objects (over which the interaction force is relayed to each force balance equation) is simply toggled on and off when interference or clearance is detected. Resolution of impulses may be performed upon detection of a collision.

The number of second order differential equations to be integrated in the coupled force balance scheme is 6 times the number of bodies. Accordingly, the number of independent coordinates is large compared to a formulation involving constraints, and computational efficiency must be regarded as poor.

Models in Dependent Coordinates

The Newton-Euler equations may not be the most expedient starting point for the construction of a interactive simulator since certain contact forces are of no intrinsic interest, especially the constraint forces between articulated bodies. The generalized coordinates used in Lagrange's equations and Kane's equations facilitate writing equations of motion on a lower dimension state space—the space of allowable motions. That is, the equations consider only those motions which satisfy those holonomic constraints encapsulated by

the choice of generalized coordinates.

After performing the operations indicated in Lagrange's equations, the terms may be arranged in the form so familiar from the robotics literature:

$$M(q)\ddot{q} + h(q,\dot{q}) = \tau \tag{4}$$

where $M(q)$ is called the inertia matrix, $h(q,\dot{q})$ consists of centripetal, Coriolis, and gravity terms, and τ contains the generalized input forces and torques. This second order differential equation contains n rows.

Further constraints, including those subject to change, or arising through intermittent contact between bodies are handled with the method of Lagrange Multipliers (also called undetermined multipliers). Holonomic constraints may be expressed

$$\Phi(q,t) = 0. \tag{5}$$

Differentiating this equation with respect to time produces:

$$\Phi_q \dot{q} = -\Phi_t. \tag{6}$$

Nonholonomic constraints may also be expressed in this form, so let Equation 6 contain both the holonomic and nonholonomic constraints. The method of Lagrange multipliers entails *adjoining* these constraints to the n differential equations, resulting in a set of n equations in $(n+m)$ unknowns. Adjoining means appending the Jacobian of the constraint matrix, Φ_q with a pre-multiplying vector of m undetermined coefficients (Lagrange multipliers) λ to the n Lagrange equations:

$$M(q)\ddot{q} + h(q,\dot{q}) + \Phi_q^T(q,t)\lambda = \tau \tag{7}$$

The m constraint equations themselves may then be used together with the n dynamical equations to bring the number of equations up to the number of unknowns in one of two ways. Firstly, the algebraic constraint equations may be used directly with the differential equations if a Differential Algebraic Equation (DAE) solver is available. DAE simulators are favored in the real-time flight simulation and automobile simulation communities, where changing kinematic constraints are occasionally of interest [26]. DAE solvers, however, are not the most numerically efficient and are not free of numerical stability problems [26] [27].

The second manner in which the constraint equations may be incorporated is by differentiating them twice (in the case of configuration constraints) or once (in the case of motion constraints) to produce m *acceleration* constraint equations which may be integrated along with the dynamical differential equations in an ODE solver. Differentiating Equation 6 again yields the acceleration constraint equations:

$$\Phi_q \ddot{q} = -(\Phi_q \dot{q})_q \dot{q} - 2\Phi_{qt}\dot{q} - \Phi_{tt} \triangleq \gamma, \ (q,\dot{q},t) \tag{8}$$

Equations 8 and 7 may be expressed together in matrix form:

$$\begin{bmatrix} M & \Phi_q^T \\ \Phi_q & 0 \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \lambda \end{bmatrix} = \begin{bmatrix} Q \\ \gamma \end{bmatrix} \tag{9}$$

There is, however, one problem remaining with the model expressed in Equation 9. The acceleration constraint equations are unstable. Through two differentiations of the holonomic constraints, information about displacements has been lost. To stabilize the system, the motion constraints are appended with penalizing factors in either the Baumgarte stabilization [28] or penalty stabilization methods. (See [27], p. 162). The penalty method is closely related to the method of the same name discussed above in connection with the Coupled Force Balance—and subject to the same short-comings. That is, numerical ill-conditioning.

One very good reason for formulating a model in dependent coordinates is to facilitate handling of changing constraints.

Models in Independent Coordinates

Although models employing Lagrange multipliers can be converted into independent coordinate models using an 'embedding technique' discussed in the next section, let us use Kane's equations to introduce independent coordinate formulations. Kane's method in its traditional application always produces models in independent coordinates. Intimately related to this capability is the production of the equations of motion as first order differential equations involving a state vector of generalized coordinates and generalized speeds. Note that the dependent coordinates may be found during integration either by solving the position problem at each time step, or, more conveniently, by integrating the constraint equations 6, which are only first order and stable, along with the differential equations.

Using a model in independent coordinates, the following must occur with each change in contact condition (change of constraint): Integration of the equations of motion must be stopped, the impulses resolved, a new set of independent coordinates must be found, the entire model reformulated, and integration re-started. Such a simulator construction which involves re-formulation of the equations of motion on the fly each time a change in constraint occurs is indeed feasible, since both symbol manipulation and numerical integration may be computerized, but this proposition remains a topic for future consideration. We will instead consider these as two separate steps: the formulation of equations to be performed prior to simulation and the solution to be performed in real time. If all possible constraint configurations are known prior to simulation time, all models may be pre-formulated and sequenced together during run-time interactively using a finite state machine [3].

We shall assume in this paper that constraints which are subject to change shall be carried along with a model in dependent coordinates as supplemental constraint equations. Kane's method can also be used to produce efficient formulations in dependent coordinates and routines for handling changing constraints numerically during simulation time will be described in the next section.

STEP 3: SIMULATOR DESIGN

Now that we have the equations of motion for our target multibody system in hand, we are ready to incorporate them into a real-time simulator with haptic display. We will now discuss the means by which each of the various models discussed in the previous section may be used to create virtual multibody systems, including those with changing kinematic constraints.

The discussion accompanying Figure 2 suggests that in choosing a model formulation method (and by implication a model form), there exists a tradeoff between computational efficiency and ease of handling changing constraints. The Coupled Force Balance is simple and accommodates changing constraints easily, but it is quite computationally intensive. On the other side of the spectrum are the models in independent coordinates which make no provisions for handling changing constraints (unless the simulator can formulate models on the fly), but are maximally computationally efficient.

Let us discuss the compromises: models formulated in dependent coordinates either with Lagrange's method or Kane's method and accompanied by constraint equations. These models will be simulated with numerical differential equation solvers outfitted with numerical constraint handling routines. Note that the constraint handling methods discussed in the present section are to be implemented *numerically* whereas the ODE formulation methods discussed in the previous section are to be implemented *symbolically*. We assume that symbolic manipulation shall take place offline, prior to simulation time, and those constraints used to eliminate dependent generalized coordinates or generalized speeds are not subject to change.

Figure 3 shows a pair of models in dependent coordinates, one produced with the method of Lagrange Multipliers and expressed as a second order ODE (Equation 9), and the other a first order ODE produced with Kane's method but without having used certain constraint equations to eliminate dependencies. These models, the product of symbolic manipulation, shall be operated upon by the various procedures to be discussed presently and indicated by arrows in Figure 3. The end-product of these numerical procedures is in certain instances a model in dependent

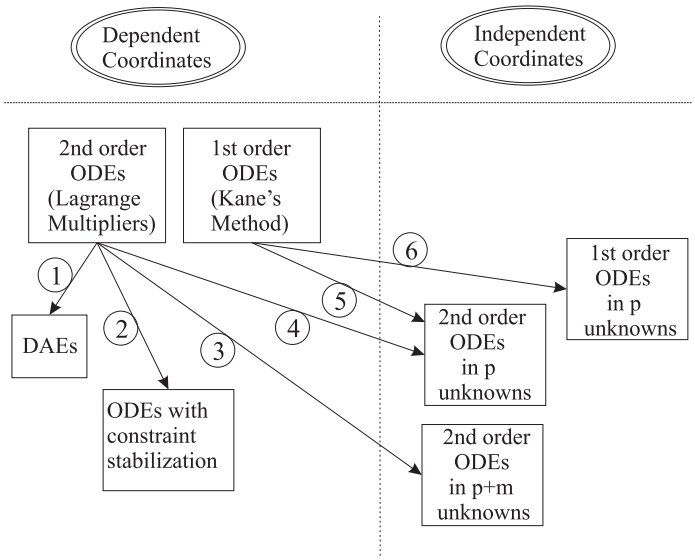

Figure 3: *Coordinate reduction technques*

coordinates and in other instances a model in independent coordinates as indicated by the organization of the termination points of each arrow in Figure 3.

The arrows numbered 1 and 2 have already appeared in Figure 2 and indicate two means by which a model produced by the method of Lagrange multipliers may be simulated. This model is ready to go if a DAE solver is available. Alternatively, after differentiating the constraint equations to form acceleration constraints and adding constraint stabilization, an ODE solver may be used as discussed above in connection with Figure 2.

Changing kinematic constraints are handled by swapping out the adjoined algebraic constraint equations at the transition times. The differential equations derived symbolically persist throughout the simulation. Gilmore and Cipra [29] [30] use an 'incidence matrix' containing the continually updated system topology to automate the handling of changing constraints in the simulation of planar systems expressed in dependent coordinates. See the work of Haug et al. [31], [32] [33], in which changing constraints, impact, and friction are treated in one framework based on a model in dependent coordinates with Lagrange multipliers.

The embedding technique indicated in Figure 3 by arrow number 4 involves partitioning the generalized coordinates into an independent and a dependent set. The acceleration constraint equation may be partitioned:

$$[\,\Phi_{q_i}|\Phi_{q_d}\,]\begin{bmatrix}\ddot{q}_i\\ \ddot{q}_d\end{bmatrix} = , \tag{10}$$

and thus the column matrix of generalized accelerations

may be expressed in terms of the independent set:

$$\ddot{q} = \begin{bmatrix} \ddot{q}_i \\ \ddot{q}_d \end{bmatrix} = \begin{bmatrix} I \\ -\Phi_{q_d}^{-1}\Phi_{q_i} \end{bmatrix} \ddot{q}_i + \begin{bmatrix} 0 \\ \Phi_{q_d}^{-1}, \end{bmatrix} \triangleq A\ddot{q}_i + b \quad (11)$$

which, when substituted into Equation 9 produces the second order differential equation in $n = p + m$ unknowns:

$$\begin{bmatrix} MA & -\Phi_q^T \end{bmatrix} \begin{bmatrix} \ddot{q}_i \\ \lambda \end{bmatrix} = \tau - h - Mb. \quad (12)$$

which is ready for an ODE solver. This method has been espoused by Wehage and Haug [34].

Two further methods for handling the Lagrange multiplier formulation involve reducing this dependent coordinate model to a model in independent coordinates. These methods are attractive because they produce a more computationally efficient model, minimizing the variables handled by the ODE solver. These methods are known as the embedding technique (also called coordinate partitioning) and the orthogonal complement technique. Both of these methods require that the coordinates be partitioned into an independent set and a dependent set, which can be a non-trivial task. For example, the Jacobian matrix of constraint relations may be solved for the independent rows at each time step, and used to direct and maintain a set of well-conditioned (maximally independent) coordinates.

The coordinate reduction (also called orthogonal complement method) indicated by arrow number 4 requires that a matrix T be found whose columns are orthogonal to the constraint Jacobian matrix Φ_q. Methods for constructing said matrix include SVD and triangularization. Once found, it may be used to pre-multiply Equations 7 and knock out the λ term, leaving a second order ODE in only p coordinates.

A first order model produced with Kane's method can be converted to a model in independent coordinates (arrow numer 5) using the methods of Wampler [35] and further developed by Mitiguy [9].

A first order model can also be converted to a second order model in independent coordinates as indicated by arrow number 6 [36].

Collision detection and impulse resolution. One of the largest challenges in multibody dynamics simulation is the detection of interference and the determination of contact points when two bodies collide (called the collision detection problem), for this determines the points of application of interaction forces. When interacting bodies are each articulated to a common base, indicator functions which take a form very similar to constraint equations may be constructed and used to detect collisions [3].

The impact restitution problem must be handled upon detection of a collision since the bodies have been assumed rigid and the time intervals over which the impact forces act will be very short, giving rise to discontinuous jumps in certain velocities. Recall that a contact may be established between bodies only when a collision occurs and the collision resolver indicates a null departure velocity. Deletion of a contact, however, may happen in one of three ways: the contact breaks because of insufficient closing force (unilateral condition), the contact breaks due to relative sliding (one body slides off of the edge of another), or the contact breaks due to the restitution of an impact (possibly occurring at a location remote to the contact under consideration). Figure 4 illustrates the sequencing of events. The integration routine must be stopped upon the detection of a collision, the momentum-impulse equations solved for the subsequent velocities, and the integration routine re-started. This issue is not a consequence of discrete simulation, but an essential part of analysis using rigid body models. Note that collision response problems can be formulated as ODEs with many of the same tools used to derive the equations of motion and certainly solved with the same solvers. Note that the worst-case simulation timestep, including collision detection and response, and model reduction or constraint equation swapping must fit into one servo period.

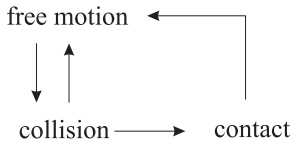

Figure 4: *Sequencing events through the Collision detector and Collision Resolver*

SUMMARY

This paper has surveyed the field of computational dynamics, contrasting the Newton-Euler, Lagrangian and Kane approaches for formulating models. In particular, the form of the model and the implications of that form on the design of a simulator for virtual environments with changing kinematic constraints were reviewed. Models produced with Kane's method were suggested as computationally efficient yet modular candidates for the development of a general purpose simulator. The ability to handle constraints (including nonholonomic constraints) using Kane's method without having to introduce additional unknowns affords significant advantages in computational efficiency. Symbolic derivation of the equations of motion of articulated structures would be supplemented with numerical handling of coordinate partitioning and automated model reduction.

REFERENCES

[1] C. Cadoz, A. Luciani, and J.-L. Florens, "Responsive input devices and sound synthesis by simulation of instrumental mechanisms: the cordis system," *Computer Music Journal*, vol. 8, pp. 60–73, Fall 1984.

[2] T. Yoshikawa, Y. Yokokohji, T. Matsumoto, and X.-Z. Zheng, "Display of feel for the manipulation of dynamic virtual objects," *JDSMC*, vol. 117, pp. 554–558, December 1995.

[3] B. Gillespie, *Haptic Display of Systems with Changing Kinematic Constraints: The Virtual Piano Action*. PhD thesis, Stanford University, 1996.

[4] B. Gillespie, "Interactive dynamics with haptic display," in *Advances in Robotics, Mechatronics and Haptic Interfacesm ASME WAM*, vol. DSC-Vol 49, pp. 65–72, ASME, 1993.

[5] J. O. Smith III, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, pp. 74–87, Winter 1992.

[6] A. Erdman and J. P. Sadler, *Modern Kinematics: Developments in the last forty years*. John Wiley and sons, 1992.

[7] K. E. MacLean and W. K. Durfee, "An apparatus to study the emulation of haptic feedback," in *Dynamic Systems and Controls*, vol. 57-2, pp. 615–621, 1995.

[8] M. Brown and J. E. Colgate, "Passive implementation of multibody simulations for haptic display," in *in these proceedings*, 1997.

[9] P. Mitiguy, *Efficient formulation and solution of equations of motion*. PhD thesis, Stanford University, 1995.

[10] P. E. Dupont and S. P. Yamajako, "Jamming and wedging in constrained rigid-body dynamics," in *International Conference on Robotics and Automation*, 1994.

[11] M. Moore and J. Wilhelms, "Collision detection and response for computer animation," *Computer Graphics*, vol. 22, pp. 289–298, August 1988.

[12] C. Cadoz, A. Luciani, and J.-L. Florens, "Cordis-anima: a modeling and simulation system for sound and image synthesis —the general formalism," *Computer Music Journal*, vol. 17, pp. 19–29, Spring 1993.

[13] A. Luciani, S. Jimenez, J.-L. Florens, C. Cadoz, and O. Raoult, "Computational physics: a modeler-simulator for animated physical objects," in *International Computer Music Conference*, ICMA, 1991.

[14] R. Barzel and A. H. Barr, "A modeling system based on dynamic constraints," in *Computer Graphics*, vol. 22 no. 4, pp. 179–88, ACM, Aug 1988.

[15] P. M. Isaaks and M. F. Cohen, "Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics," *Computer Graphics*, vol. 21, pp. 215–224, July 1987.

[16] P. Lötstedt, "Mechanical systems of rigid bodies subject to unilateral constraints," *SIAM Journal of Applied Math.*, vol. 42, pp. 281–296, April 1982.

[17] D. Baraff, "Fast contact force computation for nonpenetrating rigid bodies," in *Computer Graphics*, pp. 23–34, ACM, July 1994.

[18] D. Baraff, "Analytical methods for dynamic simulation of nonpenetrating rigid bodies," in *Computer Graphics*, vol. 23-3, pp. 223–232, ACM, July 1989.

[19] P. Lötstedt, "Numerial simulation of time-dependent contact and friction problems in rigid body mechanics," *SIAM Journal of Sci. Stat. Comput.*, vol. 5, pp. 370–393, June 1984.

[20] P. U. Lee, D. C. Ruspini, and O. Khatib, "Dynamic simulation of interactive robotic environment," in *International Conference on Robotics and Automation*, vol. 2, pp. 1147–1152, IEEE, May 1994.

[21] J. K. Hahn, "Realistic animation of rigid bodies," *Computer Graphics*, vol. 22, pp. 299–308, August 1988.

[22] B. Mirtich and J. Canny, "Impulse-based dynamic simulation," in *Proceedings of Workshop on Algorithmic Foundations of Robotics*, February 1994.

[23] B. Mirtich and J. Canny, "Impulse-based simulation of rigid bodies," in *Proceedings of 1995 Symposium on Interactive 3D Graphics*, April 1995.

[24] B. Mirtich, "Hybrid simulation: Combining constraints and impulses," in *Proceedings of First Workshop on Simulation and Interaction in Virtual Environments*, July 1995.

[25] B. Chang and J. E. Colgate, "Real-time impulse-based simulation of rigid body systems for haptic display," in *in these proceedings*, 1997.

[26] E. J. Haug and R. C. Deyo, *NATO Advanced Research Workshop on Real-Time Integration Methods for Mechanical System Simulation (1989 : Snowbird, Utah)*. Springer-Verlag, 1991.

[27] J. Garcia de Jalón and E. Bayo, *Kinematic and Dynamic Simulation of Multibody Systems: the Real-Time Challenge*. Springer-Verlag, 1994.

[28] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Comput. Methods Appl. Mech. Engrg.*, vol. 1, pp. 1–16, 1972.

[29] B. J. Gilmore and R. J. Cipra, "Simulation of planar dynamic mechanical systems with changing topologies: Part 1– characterization and prediction of the kinematic constraint changes," *Journal of Mechanical Design*, vol. 113, pp. 70–76, March 1991.

[30] B. J. Gilmore and R. J. Cipra, "Simulation of planar dynamic mechanical systems with changing topologies: Part ii – implementation strategy and simulation results for example dynamic systems," *Journal of Mechanical Design*, vol. 113, pp. 77–83, March 1991.

[31] E. J. Haug, S. C. Wu, and S. M. Yang, "Dynamics of mechanical systems with coulomb friction, stiction, impact and constraint addition-deletion –i," *Mechanism and Machine Theory*, vol. 21, no. 5, pp. 401–406, 1986.

[32] S. C. Wu, S. M. Yang, and E. J. Haug, "Dynamics of mechanical systems with coulomb friction, stiction, impact and constraint addition-deletion –ii," *Mechanism and Machine Theory*, vol. 21, no. 5, pp. 407–416, 1986.

[33] S. C. Wu, Y. S. M., and E. J. Haug, "Dynamics of mechanical systems with coulomb friction, stiction, impact and constraint addition-deletion –iii," *Mechanism and Machine Theory*, vol. 21, no. 5, pp. 417–425, 1986.

[34] R. A. Wehage and E. J. Haug, "Dynamic analysis of mechanical systems with intermittent motion," *Transactions of the ASME*, vol. 104, pp. 778–784, October 1982.

[35] C. Wampler, K. Buffington, and J. Shu-hui, "Formulation of equations of motion for systems subject to constraints," *Journal of Applied Mechanics*, vol. 52, pp. 465–470, June 1985.

[36] K. J. Reckdahl, *Dynamics and control of mechanical systems containing closed kinematic chains*. PhD thesis, Stanford University, 1997.