INTERACTIVE DYNAMICS WITH HAPTIC DISPLAY

Brent Gillespie
Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University
Stanford, California

Mark Cutkosky
Department of Mechanical Engineering
Stanford University
Stanford, California

ABSTRACT

The simulation of virtual environments which are multi-degree-of-freedom presents unique challenges to the designer of control software for haptic display devices. Once the successfully rendered stiffness, damping, and mass elements are interconnected to form multibody systems, a host of controller implementation choices arise. For one, the control law must manifest the dynamics of the simulated multibody system. Further extensions are required if the interconnection topology is allowed to change, as in the case of changing kinematic constraints. The control law must be able to undergo transformations in order to reflect these. We want to include such capabilities in our simulator repertoire, as these are among the most interesting to explore haptically. This paper describes a combined simulation and experimental apparatus for exploring issues in the haptic display of dynamical models. In particular, we address problems in the simulation of changing kinematic constraints with numerical integration methods which have been specialized for real-time mechanical system simulation. Issues in the software design of a haptic display are addressed. A simplified model of the piano action is used as an illustrative example.

1.0 INTRODUCTION

Humans are admirably equipped to explore and characterize the mechanical properties or behaviors of objects in the environment. Our muscles and articulated limbs allow us to manipulate, and our haptic senses (tactile and kinesthetic) provide us with information about an object's mechanical response to our manipulations. Our goals, however, often go beyond system identification or characterization of the mechanical impedance of an object. We may want to influence an object's behavior. Such is the case when playing a musical instrument. While attempting to exact a desired dynamical behavior (and corresponding sound) from an instrument in our hands, we use not only the sound but also the force/motion response in a feedback sense to modify our manipulation. If the musical event is of a long duration in comparison to human response times (typically greater than 200ms), this information may be used by the musician for real-time feedback control . Otherwise, the response information is used for anticipatory control to modify the manipulation the next time around, as with practice and learning. Examples of musical instrument playing in which haptic information is of relatively obvious and immediate value to the player include ¾sultandoà or ¾iicochetà bowing[1] of a string instrument and use of the repetition feature on a grand piano[2]. When an instrumentàs mechanical behavior has no correspondence to its acoustic behavior or there is no haptic information available, a very valuable channel of communication from instrument to player is lost. This is the case for most synthesizer-based musical instruments. In order to alleviate this deficiency on keyboard synthesizers, yet preserve and even expand their programmability, we are developing a synthesizer keyboard with haptic display.

A virtual piano action (a mathematical model), a simulation algorithm, and a set of 88 single degree-of-freedom haptic display devices constitute a promising means to make a synthesizer keyboard feel like a real grand piano. Other keyboard instruments could be simulated with the same device at the touch of a button. We propose, then, to simulate the feel of the grand piano by numerically integrating the equations of motion of the multibody piano action in real-time in a human-in-the-loop simulation scheme. As the integration proceeds, the finger-key interaction force is computed and

1. The sultando bowing technique involves bouncing the bow on the string with control of the contact/non-contact timing according to a desired rhythm.

2. For a quick re-strike of a note on a grand piano, it is necessary to let the key up only beyond a certain level which allows re-set of the jack under the hammer knuckle. This level is detectable by feel.

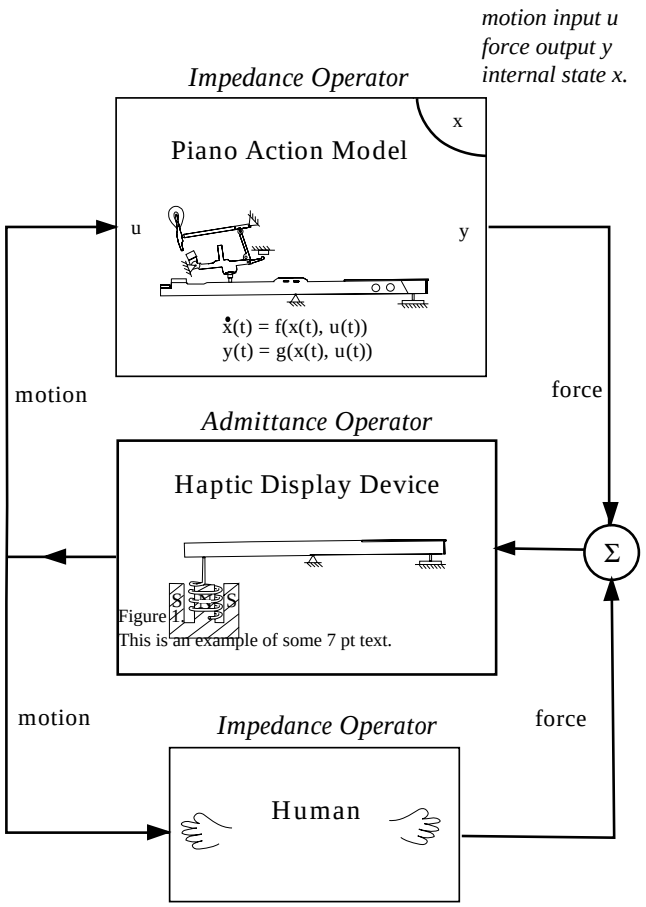


FIGURE 1.
A FEEDBACK LOOP INTERPRETATION OF THE SIMULATOR COUPLED TO THE HUMAN.

generated by a haptic interface. In addition, the motion (needed for use in the integration) is sampled from the haptic interface hardware.

Beyond the need to simulate configuration dependent impedances, the piano action inspires another desired feature. Because the piano action relies on various trip mechanisms for its function, and these have a significant effect on its touch-response, our simulator must be capable of simulating multibody systems with changing kinematic constraints.

Simulation of multibody dynamical systems with visual display has been the subject of much research in the computer graphics field. Simulation of changing kinematic constraints has also been addressed (Barzel, 1992, van Overveld, 1991). However, the use of haptic display with simulation creates a number of unique challenges. For example, to guarantee passive behavior of a system or even just stability would require some rather sophisticated analytical tools (Colgate, 1989). Coupling the force/motion sensors and actuators to a physical display device to allow for interactive dynamics effectively closes a control loop and changes the behavior of the simulation. For example, a virtual system with undamped behavior when uncoupled will demonstrate damped behavior when the powered manipulandum is tied in. (In fact, if the actuator signal is sign-reversed, it will exhibit negative damping!) The mass and damping properties of the manipulandum will figure into the closed-loop behavior and effectively mask the feel of the virtual system to the user. Figure 1 presents a feedback loop interpretation of the electrical signal coupling between the simulator (an impedance operator in this case) and the manipulandum. The mechanical contact between the user and manipulandum can also be considered a feedback connection. If the simulator is implemented as an impedance operator (operating on motion to produce force), then the manipulandum must necessarily be viewed as an admittance. The user, if coupled, appears as an impedance operator as shown in Figure 1. Other analytical viewpoints, including that of scattering operators also exist. See, for example, (Hannaford 1989). This paper does not address questions of coupled stability with analytical tools, rather it suggests the addition of some parameters into the interaction controller which will allow the range of stability to be fully explored.

Several standard forms exist for expressing a multibody model (ordinary differential equations (ODEs), differential algebraic equations (DAEs), coupled force balances). A simulator architechture can be based on any one of these (Haug, 1991). Each approach has its own advantage, be it ease of expression or ease of implementation. We have chosen to formulate the equations of motion in their reduced form (incorporating the kinematic constraints) so that they may be integrated by a standard ODE solver. Thus, the rendering of a system with multiple constraint conditions requires the formulation of multiple models and a passing of the state information from one model to the next at the transition times. Each sub-model governs the behavior only during that time-period which corresponds to the particular kinematic structure for which it was developed. Our simulator must be able to handle models described by ODEs which are only piece-wise continuous, with the discontinuities occurring at times which are themselves functions of the state. A simulation of the piano action will then be able to account for the changing kinematic constraints that occur when the elements of the action make and break contact with one another.

In this paper, we introduce a single-key motorized keyboard and its simulation algorithm which is capable of synthesizing certain aspects of the mechanical impedance of a grand piano. We discuss the application of an ODE solver to the rendering of a driving point impedance which may include changing kinematic constraints in section 2. In section 3, we briefly introduce our software environment and mention some of its features, especially those relevant to the integration scheme. In section 4, a simplified model of the grand piano action and one variant thereof are introduced. We use these models in sections 5 and 6 as specific examples of the methods outlined in section 1. In particular, the advantages of the model variant are discussed along with some experimental results. Section 7 summarizes.

2.0 CONSTRUCTION OF A HUMAN-IN-THE-LOOP SIMULATOR FROM AN ODE SOLVER

The driving point mechanical impedance of a very large class of mechanical systems can be simulated using an ODE solver as the primary computational workhorse. In the following, we describe one possible construction of an impedance operator from an ODE solver, taking note of whatever restrictive assumptions must be made. An impedance operator we define as any causal mapping from a velocity input u to a force output y. In order to allow for non-linear models, we cannot use a transfer function or impulse response description and may not express the impedance operator as a convolution. So while Z(s), the linear impedance transfer function, may not exist, we nevertheless use the term impedance to describe a mapping from motion input to force output.

As our model is dynamical, it will take the form of one or more differential equations. In these differential equations, the force output need not be the variable whose time derivative we choose to express in terms of itself and of the input, so we allow an internal state variable x, where

$$x(t) \in \Re^{2p} \tag{1}$$

The dimension of the state vector is twice the number of the degrees-of-freedom, p. Because the differential equations of motion of a mechanical system are second order, there are p kinematical differential equations and p dynamical differential equations lined up in:

$$\dot{x}(t) = f(x, u(t)) \tag{2}$$

A readout function r then expresses the force output in terms of the state x and input u.

$$y(t) = r(x, u(t)) \tag{3}$$

We do not allow for time varying models; the time t does not appear as an argument in the above description. The solution y is obtained in the usual way by numerical integration. The value of the state x, which the impedance operator maintains, may be viewed as an encapsulation of the action of the input history on the model.

2.1 Multiple Constraint Conditions

Thus far, we have not introduced the use of kinematic constraints (either configuration or motion constraints) into our modeling process. These arise if, in constructing our model, we choose to use more generalized coordinates than exist degrees-of-freedom for the model. Then m constraint equations can be written. The integer m is given by

$$m = n - p \tag{4}$$

where n denotes the number of generalized coordinates and p the number of degrees-of-freedom. If all constraint equations are either holonomic or simple non-holonomic constraints, then the entire model is expressible in the form of a set of ODEs. A simple non-holonomic constraint is expressible by a relationship between the generalized speeds u_i in the following form:

$$u_r = \sum_{s=1}^{p} A_{rs} u_s + B_r \qquad (r = 1, \ldots p) \tag{5}$$

where A_{rs} and B_r are functions of $q_1,\ldots q_n$ and possibly time t, but not of $u_1,.u_n$. The method used to obtain the equations of motion in this paper is Kane's method (Kane 1985).

2.2 Piece-wise Continuous ODEs

Finally, to further extend the class of allowable systems to those which contain changing kinematic constraints, we define piecewise continuous ordinary differential equations (PODEs) and specify a numerical algorithm to solve them. In the following, we follow the formalism developed by Barzel (1992). A PODE is made up of a sequence of continuous ODE segments. Discontinuities are allowed in both the specification and in the solution at the transition times t_i. We wrap a standard ODE solver in an algorithm which can locate events during the solution and manage the exchange of the differential functions in and out of the solver, keeping the relevant ODE in place and starting it with the proper initial conditions.

A complete model is described by a sequence of ODEs,

$$\dot{x} = f_i(x_i, u(t)) \qquad (i = 1, 2, \ldots) \tag{6}$$

a sequence of readout equations,

$$y_i(t) = r_i(x_i, u(t)) \qquad (i = 1, 2, \ldots) \tag{7}$$

a sequence of indicator functions,

$$g_i(x_i, u(t)) \qquad (i = 1, 2, \ldots) \tag{8}$$

and a sequence of transition functions.

$$h_i(x_i, u(t)) \qquad (i = 1, 2, \ldots) \tag{9}$$

We step forward in time using our chosen numerical ODE solver on the i^{th} ODE and use the i^{th} readout equation so long as

$$g_i(x_i, u(t)) > 0 \tag{10}$$

Time t_i, the end of the i^{th} segment, is found when we detect

$$g_i(x_i, u(t)) = 0 \tag{11}$$

We then switch from the i^{th} to the $(i+1)^{th}$ ODE. The initial conditions for the next ODE are setup with

$$y_{i+1}(t_i) = y(t_i) + h_i(x_i(t_i), t_i) \tag{12}$$

(See Figure 2.) Of course we cannot find the precise time point at which the event function g_i is identically *0* when we are only sampling the indicator function at each integration step. Therefore, we must use a root finder to find t_i to a specified degree of accuracy, and integrate that particular integration step in which it occurs us-

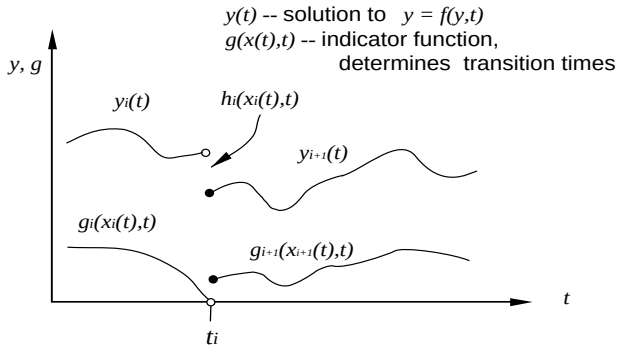


FIGURE 2.

NUMERICAL INTEGRATION OF PIECE-WISE CONTINUOUS DIFFERENTIAL EQUATIONS (AFTER BARZEL (1992)).

ing first the *i*‘‘ ODE then the *(i+1)*‘‘ ODE. Multistep integration routines must be reset after each event. See (Shampine, 1991).

3.0 SOFTWARE AND HARDWARE FEATURES

The above outlined PODE-solver based simulator design has been implemented in a C++ program and some hardware which we briefly describe here.

The equations of motion are derived from an input model description and expressed in reduced form via computerized symbol manipulation using Kane’s method (Schaechter 1988). These are then incorporated (see details below) into a simulator program which features graphical, haptic and audio (MIDI--musical instrument digital interface) interfaces. The user may select from a menu of virtual objects. Screen-based sliders allow for the modification of various model parameters during run-time, which we have found to be an invaluable asset during development as have others (Minsky 1990). A ¾486 50 MHz PC with a Delta-Tau Co. motor control card form the heart of the system. An electro-mechanical apparatus employing voice-coil motors attached to keys or paddles provides for the haptic display. Virtual bouncing rubber balls, pendulums on carts colliding with walls, simplified piano actions, and other dynamical systems with configuration-dependent kinematic constraints are examples of the simulations currently operational.

3.1 Some Details of the Software Design

The simulator class acquires its functionality by composition rather than by inheritance (See Figure 3.) Upon selection of say, the virtual piano, each of the pertinent piece-part objects is constructed and pointers to these passed as arguments to the constructor of a simulator object. Construction of a piano simulator object is shown. Because the members of the graphical objects, models, and indicator functions are all written as virtual member functions, the simulator object can take advantage of the run-time polymorphism features of

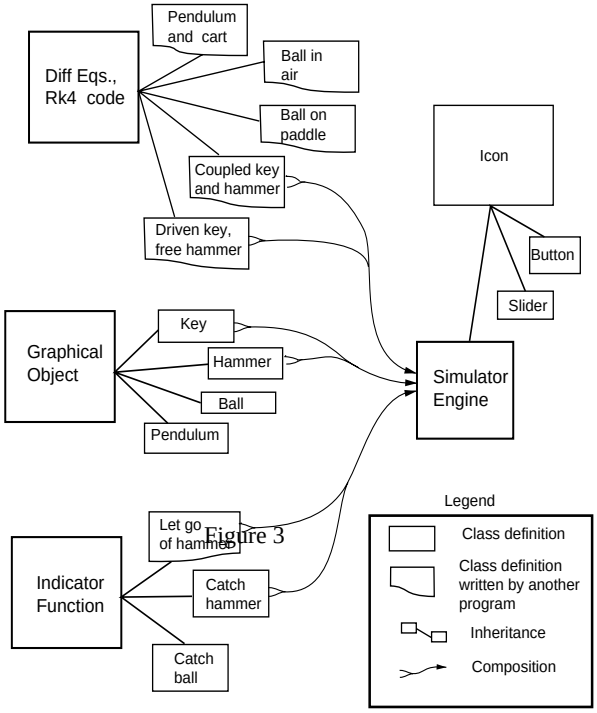


Figure 3

FIGURE 3.

CLASS HIERARCHY FOR THE SIMULATOR PROGRAM (C++ LANGUAGE).

C++. Thus, the simulator engine itself is completely generic code, able to operate on any of the models. Once created, a simulator object (of class Simulator Engine) will be called upon at each iteration of the event loop to execute the following: poll the sensors for current readings, use the current *(i^{th})* model to integrate ahead one time-step, check the indicator function, send a force value per the readout equation to the D/A converters, and finally, perform graphic display functions. One more software feature is worth noting. The class definitions highlighted with non-square boxes in Figure 3 are themselves written by another program. Thus, a new model may be added without actually having to write C-code.

4.0 MODELS OF THE SIMPLIFIED PIANO ACTION

4.1 Model A

A highly simplified schematic representation of the grand piano action is shown in Figure 4. With such a model, we wish to allow the user to manipulate (and feel), through the leverage of a key, a hammer either resting on the key or swinging like a pendulum free of the key. Both the key and hammer are rigid bodies subject to the action of gravity, and their motion is confined to a vertical plane. A successful rendering of the feel of throwing and catching the hammer will be a prelude to the rendering of the repetition feel of the piano action. A more detailed model of the piano action designed to render the letoff behavior has also been doc-

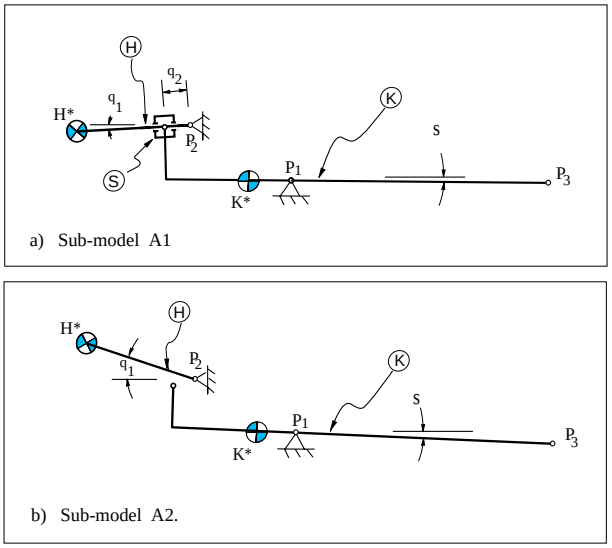


FIGURE 4.

MODEL A: SIMPLIFIED PIANO ACTION IN TWO PHASES OF MOTION

a) COUPLED MOTION b) HAMMER FREE FLIGHT

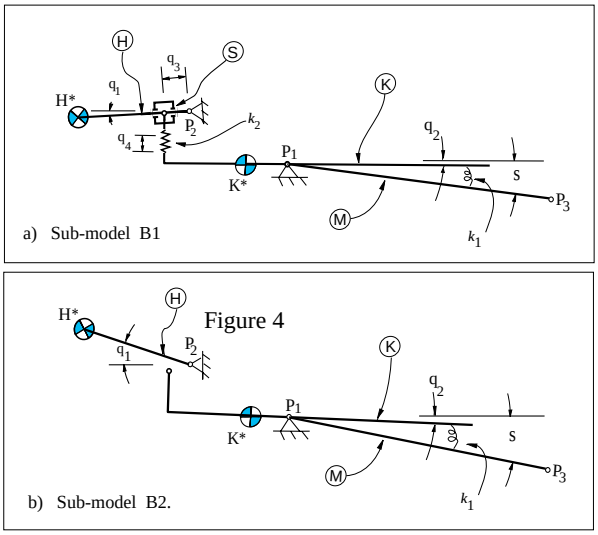


Figure 4

FIGURE 5.

MODEL B: SIMPLIFIED PIANO ACTION WITH SPRINGS IN TWO PHASES OF MOTION

a) COUPLED MOTION b) HAMMER FREE FLIGHT

TABLE 1.

COMPARISON OF MODELS A AND B.

Model name	Sub-model name	Acitve sliders	DOF *(p)*	Number of constraint equations *(m)*	Generalized coordinates (of number *n)*
A	1 coupled motion	S	0	2	q1, q2
	2a hammer flight	x	1	0	q1
	2b key	x	0	0	-
B	1 coupled motion	S	2	2	q1, q2, q3, q4
	2a hammer flight	x	1	0	q2
	2b key	x	1	0	q1

umented (Gillespie 1992). The simple model described here is similar to a bouncing ball simulation, with the important extra requirement that trigonometric constraint conditions must be considered while the hammer rests on the key.

Two bodies comprise this model, which we shall call model **A**: the key **K** and Hammer **H**. In sub-model $\mathbf{A_1}$, the motions of **K** and **H** are coupled through a frictionless slider **S** which pivots on **K** and slides along **H**. In sub-model $\mathbf{A_2}$, **K** and **H** move independently. Let **N** denote a Newtonian reference frame. Body **K** pivots about point $\mathbf{P_1}$ fixed in **K** and **N**; body **H** pivots about point $\mathbf{P_2}$ fixed in **H** and **N**. Points **K*** and **H***represent the mass centers of **K** and **H** respectively. Generalized coordinates q_1 and q_2 and the specified variable scompletely characterize the instantaneous configuration of sub-model $\mathbf{A_1}$. Displacement q_2 locates slider **S** on **H** whereas s and q_1 are the radian measure of angles which locate **K** and **H** with respect to the horizontal. Since the motion of **K** will by prescribed by the user, sub-model $\mathbf{A_1}$ has zero degrees-of-freedom; it is a strictly kinematical model. Two constraint equations can be used to relate q_1 and q_2 to s. Sub-model $\mathbf{A_2}$ is itself made up of two models: a single degree-of-freedom model for the free-flying hammer and a zero degree-of-freedom model of the key. Generalized speeds are formed as functions of the generalized coordinates simply by setting

$$u_i = \frac{dq_i}{dt} \qquad (i = 1, 2) \tag{13}$$

An auxiliary generalized speed u_3 is used to bring the interaction force between **K** and **H** at **S**, called $\boldsymbol{F_{KH}}$ into evidence. An auxiliary generalized speed u_4 is used to bring the interaction force between **K** and the user who specifies the variable s into evidence; call this force $\boldsymbol{F_{KU}}$.

4.2 Model B

Alternatively, model **B** of the piano action (with linear springs in place wherever expressions for interaction forces will be needed) can be constructed as shown in Figure 5. Use of this model in the simulator carries one very important advantage, namely, the interaction force trajectories during simulation can be made arbitrarily smooth if the spring constants are set sufficiently low, even in the face of very noisy sensed velocity and acceleration from the manipulandum. However, this advantage may come at the expense of an unrealistically compliant keyboard sensation.

In model **B**, an additional body **M** (the manipulandum) is pivoted about point $\mathbf{P_1}$. A torsional spring of stiffness $\boldsymbol{k_1}$ is attached

between **M** and **K**. Also, a spring of stiffness $\boldsymbol{k_2}$ which always lies perpendicular to the long axis of **K** attaches **S** to **K**. Let $(s\text{-}q_2)$ and q_4 denote the extensions of springs $\boldsymbol{k_1}$ and $\boldsymbol{k_2}$. The use of two additional generalized coordinates in model B_1 is necessary since there are two degrees-of-freedom. Sub-model $\mathbf{B_2}$ is made up of two one degree-of-freedom models as shown in Figure 5b. See Table 1 for a comparison of models **A** and **B**.

5.0 IMPLEMENTATION OF THE VIRTUAL PIANO ACTION.

In the following, we discuss the setup of the components of the simplified piano action models **A** and **B** for the PODE solver.

5.1 Simulation with model A

The formulation of model $\mathbf{A_1}$ described in section 4.1 produces the differential equation *d/dt(x)=f1 (Eq 6, i =1)* upon which the ODE solver is first invoked. We use the expression of the interaction force at point $\mathbf{P_3}$, $\boldsymbol{F_{KU}}$ for the readout equation r_1 *(Eq. 7, i =1)*. For the event function g_1 *(Eq. 8, i =1)*, the expression for the interaction force between the key and hammer $\boldsymbol{F_{KH}}$ is used. So long as the interaction force $\boldsymbol{F_{KH}}$ remains compressive, that is, $g_1 > 0$, the simulation continues with the motions of the hammer and key coupled as guaranteed by Model $\mathbf{A_1}$. The transition function h_1 is constructed to ensure that the initial conditions for Model A_1 satisfy its constraint conditions. Generalized coordinates q_1, q_2, and generalized speeds u_1 and u_2 are determined from *s* and *d/dt(s)* according to the constraint equations.

Upon first detecting $g_1 < 0$, integration with Model $\mathbf{A_2}$ begins. The hammer now flies freely towards the string. During this time, along with integration using the differential equation *d/dt(x)=f2 (Eq. 6, i =2)*, and use of readout function r_2 *(Eq. 7, i =2)* from model $\mathbf{A}_2$, a new event function g_2 *(Eq. 8, i =2)* is used. Event function g_2 is an interference checker which evaluates to a number less than *0* when **K** intersects the line segment $\mathbf{P_2H^*}$. No special transition function h_2 is used for Model $\mathbf{A_2}$. The final conditions of $\mathbf{A_1}$ are used as initial conditions for Model $\mathbf{A_2}$. Note that the two independent sub-models of Model $\mathbf{A_2}$ can be integrated independently using a multi-state integrator by uncoupling 2 states. The hammer will either fall back towards the key under the action of gravity or reach a height at which it strikes a virtual string. A struck string event is demarcated by the sounding of a tone by a synthesizer hooked into our hardware setup. At the time of contact, the sign of u_2 is reversed to effect a perfectly elastic collision between hammer and string. [1] At the end of the simulation epoch with model A_2, that is, at time t_2 detected by $g_2 < 0$, we return to simulation with model $\mathbf{A}_1$ (catch the hammer)

The most obvious errors in the algorithm outlined so far are introduced because the roots of the event functions are not found exactly but instead crossed over due to the finite step size h. If computational time allows, a root finder can be used once a crossover is detected to find t_i to within some prescribed bounds. Model $\mathbf{A_1}$ is used to simulate up to time t_i, then model $\mathbf{A_2}$ is used for the remainder of that step. Portions of this computational burden can be shared by prior integration steps if the location of a root can be predicted in advance. This requires even more stringent smoothness (Lipschitz) conditions on g_i. Multistep codes are especially well suited for such applications. These techniques are the subject of ongoing work.

5.2 Simulation with model B

Because the acceleration of the manipulandum appears in the expressions for the interaction forces $\boldsymbol{F_{KU}}$ and $\boldsymbol{F_{KH}}$, these force signals will be in error if that acceleration signal is beset with noise. Such noise is a serious performance limiting characteristic of most hardware setups. Temporal filtering may help, but will introduce de-stabilizing lags into the control loop.

Instead, use of Model **B** for simulation provides tolerance to noisy input signals. Expressions for the interaction forces are very simple in model **B**:

$$F_{KH} = k_2 \cdot q_4(t) \qquad (14)$$

$$F_{KU} = k_1 \cdot (s(t) - q_2(t)) \qquad (15)$$

These expressions are used for the indicator function and readout equation, respectively. Otherwise, the setup of model **B** for the PODE solver is quite analogous to the setup of model **A**. During simulation with model B, the signals F_{KH} and F_{KU} will be very smooth in comparison to those for model **A**. The more compliant the springs $\boldsymbol{k_1}$ and $\boldsymbol{k_2}$ are made, the smoother these signal will become. Essentially, the resonant frequencies of the structure are lowered as a result of lowering the stiffness. Thus, the use of Eq. 15 for readout r_1 (force display) will produce a signal of limited bandwidth which is well suited for force-display on a manipulandum. Likewise, Eq. 14, used for the indicator function g_1, will be more trustworthy because of its smoothness. This smoothness could be expressed as a Lipschitz condition and used to guarantee the ability to find the transition times with a finite sampling rate of the indicator function. Most importantly, the extra degrees of freedom in model **B** provide filtering of high-frequency noise on the input signals.

Unfortunately, piano action **B** will feel spongy. Extra dynamics have been added which we are not interested in feeling. Model **B**, however, does provide a means of measuring the capabilities of the haptic display hardware for dynamical simulation. The spring stiffnesses may be raised incrementally until the associated natural frequencies are high enough, or outside of the capabilities of the display.

1. The hammer/string interaction is the subject of future work. For now, we assume that the duration of this simulation epoch is zero.

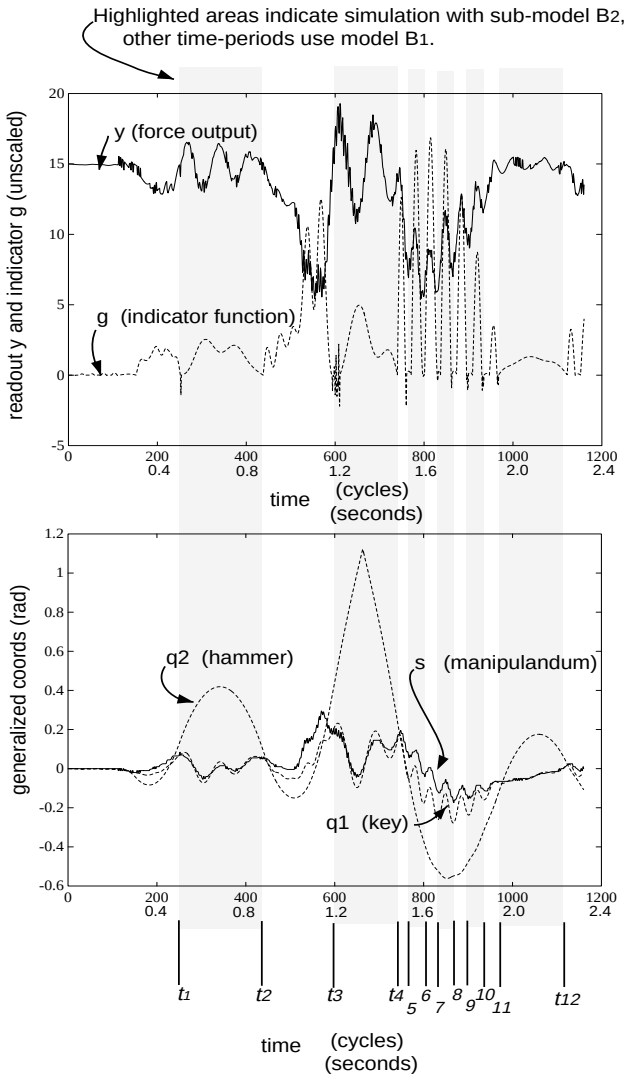


FIGURE 6.
SIMULATION USING MODEL B

6.0 RESULTS AND DISCUSSION

Figure 6 shows the results of an experiment using model **B** (with springs). The time-histories of the output force y(t) and the indicator function are plotted in the upper graph of Figure 6 in a manner analogous to that of Figure 2. Directly below, the angular displacements of the manipulandum, key, and hammer are plotted. The model switch times t_i as found by the indicator function are noted with prominent vertical lines. The simulation epochs of model $\mathbf{B_1}$ and of $\mathbf{B_2}$ are noted by highlights. The simulation step size was .002 seconds but the closed loop servo step size was .007 seconds, so the simulation was not synchronized to real-time. The numerical integrator used is of fourth-order Runge-Kutta type. However, as we sample the inputs only once per step instead of three times as required by such codes, the numerical accuracy lies somewhere between that of the Euler method and the fourth-order Runge-Kutta method. The model parameters used for the experiment are shown in Table 2.

For this particular simulation, the manipulandum was held still for 0.3 seconds, the hammer was then thrown upwards, it flew free, and was caught at 0.9 seconds into the simulation. It was then thrown again, this time harder, so that the hammer hit the virtual string, landed on the key again at 1.5 seconds and bounced several times. Note from Figure 6 that the virtual piano action expresses different dynamics during differing epochs. In particular, the key pushes up on the user (compressive F_{KU}) with more force during simulation with model $\mathbf{B_1}$. It is during these times that the weight of the hammer is felt through the key. At present, we do not consider the possible loss of contact between user and key. In fact, the user holds the key on both top and bottom, so that both compressive and tensile values for $\boldsymbol{F_{KU}}$ are possible when viewed as a force on one side only.

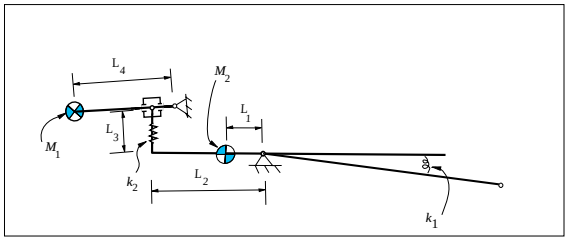
Figure 6 depicts what we consider a relatively successful simulation of a (rather soft) hammer and key. One certainly gets the feeling of manipulating a bouncing object through a lever. In a preliminary experiment, our subjects were able to bounce the hammer on the key so as to either make a sound or not (as desired or called upon) without watching the graphic display. Readers familiar with synthesizer keyboards will recognize that this virtual piano action already overcomes many of the limitations of the direct mapping from key velocity to loudness used on MIDI keyboards.

In contrast, use of model A does not produce a successful simulation. No matter what parameter values are used, the system quickly goes unstable. Simulation with model A is unsuccessful in large part because of noise on the only velocity and acceleration signals available in our present system. We are using a 50-line-per-inch linear optical encoder for position and taking first and second backward differences of this to obtain velocity and acceleration signals. The range of motion is about 3 inches, corresponding to 45 degrees on the manipulandum. The current hardware setup is only capable of simulating springs of up to 100 N/m and dashpots of 1 N/m/s, and cannot simulate masses. Hardware improvements are underway so that we may stiffen up the springs in model B and eventually use model A.

Forthcoming integration routine improvements include the use of a root finder to locate the transition times t_i within a step along with the intra-step switching of sub-models. The mis-estimation of the transition times in the present system will cause constraint violations in the hammer/key model, which will quickly lead to instability. The design of indicator functions is another area which could use improvement. Note that both of the indicator functions g_1 and g_2 used for Figure 6 start at zero. Therefore, they run a high risk of evaluating to a number less than zero and causing faulty transitions at early periods in each epoch.

TABLE 2.

MODEL PARAMTERS USED FOR EXPERIMENTS



Dimensions (m)	L1	4
	L2	10
	L3	1
	L4	0.5
Masses (kg)	M1	6.5
	M2	.01
Stiffnesses (N/m)	k1	620
	k2	15

7.0 SUMMARY

We have presented a modeling and simulation algorithm which accommodates dynamical systems with changing kinematic constraints and provides for the re-creation of their mechanical impedance by simulation and haptic display. The method involves modeling the system in each of its constraint conditions and lining these differential equation descriptions up in a sequence along with the requisite readout equations. Indicator functions which signal the end of applicability accompany each model as do transition functions which ensure the proper set-up of initial conditions. The sequence of models can be considered a piece-wise continuous ordinary differential equation. An ODE solver together with some managing routines constitutes a means of creating an impedance operator for use in the real-time simulator. Although the method is intended only for systems for which all constraint conditions are known ahead-of-time, it is conceivable that model formulation could be accomplished in real-time and general non pre-determined systems accommodated.

Our hardware and software setup was briefly described. A simplified piano action model was presented and used as an illustrative example and basis for reporting some initial experimental results. A model with additional compliant elements was introduced and suggested as a first-cut approach to simulating dynamical systems. Such a ¼ess-stiff½set of equations produces smooth output signals despite noisy input velocity and acceleration signals. The extra dynamics and associated spongy feel can be incrementally tuned out by increasing the stiffness of the added springs as hardware improvements reduce sensor noise.

8.0 REFERENCES

Barzel, R., *Physically-Based Modeling for Computer Graphics*, Academic Press, Boston, 1992.

Colgate, J. E., 1988, ¼The Control of Dynamically Interacting Systems,½Ph.D. Thesis, Massachusetts Institue of Technology, Cambridge, MA.

Gillespie, B., ¼Dynamical Modeling of the Grand Piano Action,½*International Computer Music Conference Proceedings*, Held: San Jose, CA 1992, pp. 77-80.

Gotow, J. K., et al. ¼Controlled Impedance Test Apparatus for Studying Human Interpretation of Kinesthetic Feedback,½*Proceedings of the 1989 American Control Conference*, vol. 1, 1989, pp. 332-7.

Hannaford, B., ¼A Design Framework for Teleoperators with Kinesthetic Feedback,½*IEEE Transactions on Robotics and Automation*, vol. 5, no. 4, Aug. 1989, pp. 426-34.

Haug, E. J., and Deyo, R. C., eds. *NATO Advanced Workshop on Real-Time Integration Methods for Mechanical System Simulation*, Springer-Verlag, Berlin, 1991.

Kane, T. R., *Dynamics, Theory and Applications*, McGraw-Hill, New-York, 1985.

Minsky, M., et al. ¼Feeling and Seeing: Issues in Force Display,½*Computer Graphics*, vol. 24, no. 2, March 1990, pp. 235-43.

van Overveld, C. W. A. M., ¼An Iterative Approach to Dynamic Simulation of 3-D Rigid-Body Motions for Real-Time Interactive Computer Animation,½*Visual Computer*, vol. 7, no. 1, 1991, pp. 29-38.

Schaechter, D. B., and Levinson, D. A., ¼Interactive Computerized Symbolic Dynamics for the Dynamicist,½*Journal of Astronautical Sciences* vol. 36, no. 4, Oct-Dec 1988, pp. 365-88.

Shampine, L. F., and Gladwell, I., ¼Reliable Solution of Special Event Location Problems for ODEs,½*ACM Transactions on Mathematical Software*, vol. 17, no. 1, march 1991, pp. 11-25.