Chapter 5

Passive Rendering of the Virtual Wall

5.1 Introduction

5.1.1 Motivation

The virtual wall is the simplest example of a programmable boundary within the workspace of a manipulandum. As such, the virtual wall is a fundamental component of almost all virtual objects. It is set up with an *if* statement:

if beyond a certain position, react; else do nothing.

The wall is the incarnation in *virtual* reality of a unilateral constraint. The wall comprises a simple configuration-dependent changing kinematic constraint. As discussed in previous chapters, such changing constraints are an important consideration in rendering the piano action for haptic display. The fact that the wall fully yet simply encompasses the notion of changing constraint conditions or changing sub-models makes it a natural and worthy topic of study in this thesis. In the present chapter, the virtual wall will be studied systematically with the aim of developing robust algorithms for simulation across constraint discontinuities.

Beyond its position as a fundamental building block of virtual objects, the virtual wall rouses research interest because of the difficulties which its realization presents in practice. Despite its apparent simplicity, the virtual wall usually evades perceptually convincing renderings. When touching

a wall, especially one which is meant to be stiff, undesirable vibratory motion of the manipulandum (often called contact instability or chatter) tends to arise. The manipulandum continually makes and breakes contact with the virtual wall during this behavior. Such non-passive behavior immediately expunges any sense of immersion which the human operator may have been enjoying prior to encountering that chattery wall.

Because it is such a challenge to implement chatter-free, the virtual wall is already finding use as a benchmark for performance comparisons between haptic interfaces. Thomas Massie quotes a maximum wall stiffness attainable using his PHANToM haptic interface of XX N/m, where the criterion is presumably the non-existence of chatter [65]. Colgate has proposed the range of passively displayable impedances as a useful measure of attainable performance by a haptic interface, which he calls Z-range [22]. However, the wall, because of its discontinuous nature which makes it a bigger challenge, should be considered in the suite of objects which can be rendered passively by a haptic interface.

Outline

This chapter will address the problem of chatter associated with stiff virtual walls by developing improved controller designs. These controllers (or virtual wall algorithms), when used in the standard digital implementation for haptic display, will render walls which do not suffer chatter even when the wall stiffness is high and the sampling period long.

In the remainder of this introduction, I will discuss the origins of chatter in the virtual wall. The roles of the human, the haptic interface device, and the controller in the mechanisms whereby mechanical energy is introduced (which exhibits itself as chatter) will each be detailed. Various factors may underlie a tendency toward unstable behavior observed in a controlled, coupled system such as the virtual wall. These include non-colocated sensor and actuator, system dynamics which are unmodeled or otherwise omitted from the controller design, and sensor signal quantization. Some of these mechanisms can be avoided by informed mechanical design, others are more difficult to avoid. Two culprits which are not easily quelled by good design will be identified and singled out for analysis in this chapter. First, the zero-order-hold operator and second, the possible asynchrony of the wall threshold crossings with the sampling times. Both are inevitable consequences of the sampled data implementation of the virtual wall. This introduction will wrap up with an enumeration of claims about two new controllers to be presented which compensate for the ill-effects of the zero order hold and intersample threshold crossing. Section 2 will review the literature pertaining to controller design for haptic display, especially with regard to virtual walls. In section 3, the model of a bouncing ball which will serve as a useful allegory for the development of the controller designs is presented. In section 4, the design of the two improved virtual wall algorithms will be carefully

developed. The first design uses model-based prediction, the second design makes use of standard state-space digital control design techniques. Section 5 will present results from an experimental implementation of both of these candidate virtual walls. Section 6 will discuss and summarize and Section 7 will present extensions.

The next chapter will present a thorough analysis of virtual walls with and without the improvements introduced in the present chapter. The goal in the next chapter will be to produce measures useful for gauging and predicting the performance improvements which result when these new controllers are implemented. Treatments in the next chapter will be of a more theoretical nature and another literature review section will be included.

5.1.2 Origins of chatter in the virtual wall

Assumptions regarding the role of the human

The tendency of chatter to arise is naturally a function of the wall algorithm with its parameters and the physical properties of the haptic interface, but this tendency toward chatter also depends to a large extent on the physical properties of the human user—specifically, the human's driving point mechanical impedance at the interface. (Throughout our treatment, we assume that the human's finger maintains contact with the manipulandum.) The dependence on the human's impedance is not so surprising when we realize that under consideration is the *interaction* behavior of two dynamical systems, the manipulandum and the human limb. But even further, under consideration is the interaction behavior of two *controlled* dynamical systems. Behavioral predictions cannot be made until both systems (manipulandum and human), each with their controller (computer and brain) are brought into the analysis. For example, note that the driving point impedance of the human hand or finger can be modulated (within certain bounds) by the human operator by changing muscle activation levels or by changing hand/finger postures. Thus, by pressing in certain ways, chatter against a virtual wall can be selectively induced and sustained, and sometimes even amplitude-modulated. Another interesting empirical observation to be made regarding walls and the human exploring them is that the same wall may be destabilizable (prone to chatter) under the fingers of one person while always remaining stable under the fingers of another. Presumably this effect is due to the differing impedance properties of the fingers of the two explorers.

The foregoing examples highlight the way in which chatter is usually encountered and points to an important modeling assumption which can be used to greatly simplify the analysis (and design) of the virtual wall—that is to assume constant command on the part of the human. Chatter frequencies are typically 10-30 Hz. An effective strategy for the human to induce chatter is usually not to move back and forth at high frequency but rather to adopt and maintain a certain impedance

while simply hitting the wall a single time (or even gently coming up against the wall). Although not always beyond the command capabilities of the human, typical chatter frequencies are certainly high compared to the frequencies which characterize the wall-strike *intentions* of the human. Therefore, assumptions of constant control output by the human will be made in the following analyses and control designs. Assumptions about the particular human impedance and bias-force command level itself will be left open for as long as possible. The primary culprit in the chattery interactive behavior is assumed to be the discontinuous and digital nature of the manipulandum controller (the digitally implemented virtual wall algorithm) rather than the effects of any varying command from the human. Armed primarily with the observation that chatter remains, both empirically and in simulated settings, when a constant impedance is adopted by the human, we assert that the human is not responsible for introducing energy into the system. We will assume that the human can be modeled with a constant, passive impedance. [1]

In fact, we will assume that the wall-exploring human may be fit with a second order linear time-invariant model and draw justification for this assumption from two items. First we note that contact instability in a virtual wall does not depend on time variance of the human as mentioned above–the problem remains when the human wall explorer refrains from making volitional movements. Second, the literature indicates that second order linear models may be used to model a human finger to a very good degree of fit [39] (and references contained therein), so long as the time durations are short.

The sampled data system

The virtual wall is commonly rendered for haptic display using the very simple algorithm given in Table 5.1. [2]

In natural language, if the sampled position of the manipulandum y_k is beyond a setpoint y_{wall}, exert a restoring force f_k proportional to its distance beyond that setpoint, else do nothing.

Occasionally damping is also used as in the following control law:

$$f_k = K(y_k - y_{wall}) + Bv_k \tag{5.1}$$

[1] We adopt the standard definition of passivity, that energy cannot be extracted from a passive system. Technically: for all possible force/motion trajectories, the running integral of force time velocity is always less than the initial stored energy.

[2] Note that the pair of $if/else$ logical statements in the Pseudocode of Table 5.1 may be replaced by the single statement $if(y > y_{wall})\ y = y_{wall}$, placed before the evaluation of the control law. Such a code structure would be more suggestive of the unilateral operator used in the block diagram figures which follow shortly within this discussion. The $if/else$ structure, however, is more suggestive of a switching control law, conditioned on the sampled position. Subtle distinctions *do* arise for damped virtual walls when a first-difference approximation is made for velocity from the sampled position, and that approximation is considered to be part of the control law. For our purposes there is no difference between hese two code structures.

Table 5.1: **Pseudocode for the Virtual Wall**

```
loop at sample rate {
    sample manipulandum position y.
    if y_k < y_wall
        f_k = K(y_k − y_wall);
    else
        f_k = 0.0;
    display force f_k
}
```

where v_k is the sampled manipulandum velocity. If the manipulandum velocity is not available from a sensor, the position is numerically differentiated to produce v_k. The loop is typically run at servo-rates of 300 to 1000 Hz. Hysteresis is sometimes added in the hopes of dispelling chatter and further improving perceived hardness. These and other approaches, successful to various degrees, will be discussed in the Literature Review section to follow.

The above algorithm is implemented as the digital controller $C(z)$ in a sampled data system as shown in Figure 5.1. The haptic interface device is shown as the continuous plant $P(s)$ and the human as the continuous linear impedance operator $H(s)$. This sampled data system naturally comprises both continuous and discrete elements, linked through the Sample and Hold operator S/H and the sampler of period T, as shown in Figure 5.1. The Sample and Hold operator is responsible for sampling the controller output (at sampling times which are assumed to be synchronous with the sampler T) and holding these constant until the next sampling time (zero order hold). A unilateral nonlinearity, symbolized with an icon of its graph, is used to encapsulate the action of the $if/else$ switching statements in Figure 5.1.

Now that the elements of the virtual wall have been laid out and the human has been acquitted of the energy-introduction crime, it is time to implicate the real offenders. The offenders are the zero-order hold and intersample-threshold crossing. First we discuss the role of the zero-order hold.

Origins of non-passive behavior: the zero order hold

Though the virtual wall implementation does have elements capable of giving rise to active behavior (the motor and motor amplifier), these are not directly responsible for chatter in the virtual wall. If the motor produces only those reaction forces which mimic the reaction forces of a physical unilateral spring, the controlled motor would appear passive, despite the fact that its amplifier is plugged into

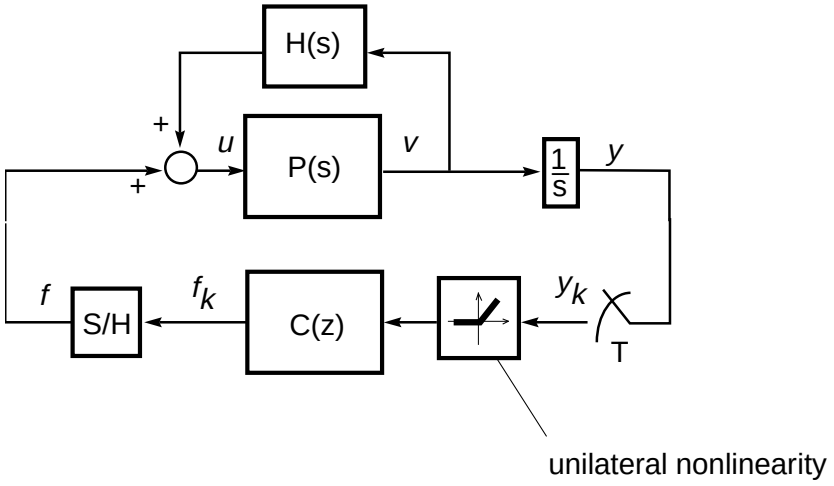

Figure 5.1: *Implementation of a Virtual Wall*

the wall. We cannot assume, however, that a discretely implemented but continuous-time inspired spring-damper control law will cause the motor to behave passively. The offense has been committed while implementing in the discrete domain a wall designed in the continuous domain. Specifically, the sample and hold (zero order hold) operator and the possibility of the crossing of the wall threshold being asynchronous with the sampling times can be identified as the means of introduction of energy.

It is well known in digital control theory that a controller designed in the continuous domain will yield poor results if implemented in the digital domain as a sampled-data controller when the sampling period is long. The standard rule of thumb used in digital control design requires that the sampling rate be 20 times the highest expected system frequency. If this guideline is not met, the closed loop system will likely be effectively destabilized by those designs which are produced with continuous system methods. Now, given sufficient physical damping or (to a possibly different degree) positive virtual damping, the closed loop system may nonetheless exhibit passive behavior. However, such requirements on the design can be considered sub-optimal since they require increased actuator authority. More discussion on this topic will take place in the Literature Review section to follow.

Intuitive explanation

An intuitive explanation of the energy-producing effects of the discrete (sampled-data) implementation of a wall created with the control law for the undamped wall,

$$f_k = K(y_k - y_{wall}) \tag{5.2}$$

is now offered. This description roughly follows that of Colgate in [23].

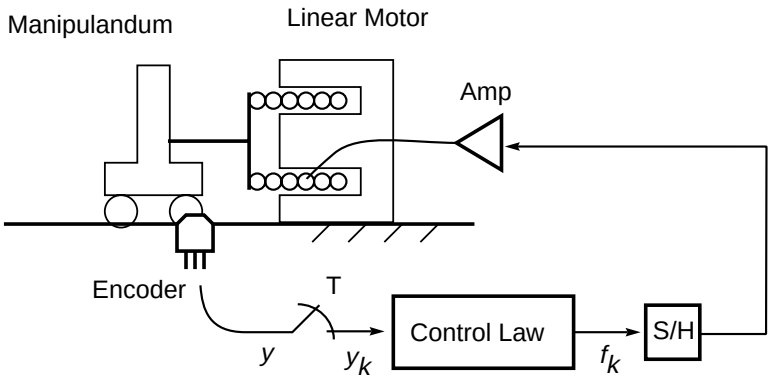

Figure 5.2: *Implementation of a Virtual Wall*

Figure 5.2 shows a very typical virtual wall implementation, into which the control algorithm of Eq. 5.2 would be inserted in the control block. The position of a linear single-axis backdrivable manipulandum is sampled by an encoder (with assumed infinite spatial resolution) and used as y_k in the algorithm. The algorithm output f_k is zero-order held before being amplified and used in the form of current to drive the linear motor attached to the manipulandum.

While moving into the wall, the sampled manipulandum position will necessarily (except at the sampling times themselves) lie closer to the virtual wall surface than the actual position of the manipulandum. Consequently, the force output, while moving into the wall, will (except at the sample times) be lower than it would have been for a continuous wall. By contrast, while moving out of the wall, the sampled position will lie deeper inside the wall (except at the exact sample times, where it is correct) than the actual manipulandum position and consequently the force will be (by comparison to the real wall), too high. Thus as one presses on the virtual wall, one needs to perform less work than one would on the real wall (its referent) to produce the same deformation. As one lets go, one has more work returned by the virtual wall than would have been returned by its real-world counterpart. Thus to simply push on the wall and let go (a common exploratory procedure) is an effective method for extracting energy from the wall. The virtual wall is, obviously, quite non-passive.

Figure 5.3 shows a trace of the zero-order-held force output history versus the position history which produced it overlaid on a graph of the constituent equation $f = Kx$ of the referent wall of

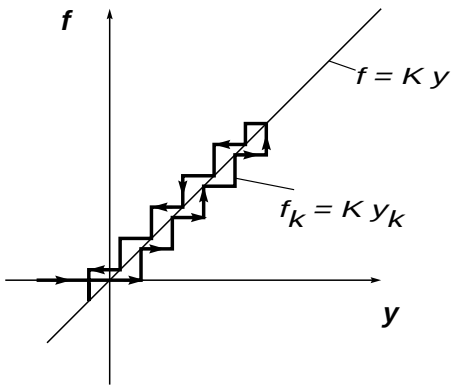

Figure 5.3: *Tracing out the Virtual Wall*

stiffness K. From such a plot, it can easily be seen that the time-average force while moving into the wall is by comparison to the constituent equation too low and on the way out too high. A *negative* hysteresis curve is thus introduced which produces energy when traversed.

Origins of non-passive behavior: asynchronous switching times

From the intuitive discussion above, it is apparent that the zero-order hold is responsible for non-passive behavior of virtual walls, but I will point out a second slightly more subtle energy-instilling aspect of the digital implementation of the virtual wall. This factor only plays a role in unilaterally acting virtual objects such as the wall, it is not present in objects which lack changing constraint conditions.

Briefly, this effect is due to asynchrony of would-be constraint changes in the virtual wall with the sampling times of the controller. Constraint changes should occur when the indicator function (as defined in previous chapters) evaluates to zero. However, because of discrete sampling and the ZOH, changes are not enacted until the next sampling time after the trip of an indicator function. Crossings of the threshold (trips of the indicator) which occur between sample times can effectively create energy.

Considering a wall of stiffness $K = 1$ allows us to view the control signal in an informative time-chart. Figure 5.4 shows the trace of the zero-order held output force resulting from a single strike of the unit-stiffness wall overlaid on top of a time trace of the manipulandum position y. This plot is somewhat idealized, but experimental plots are very similar. See Figure 3.18 for a plot of the virtual wall reaction force overlayed on the manipulandum position derived using a virtual wall implementation.

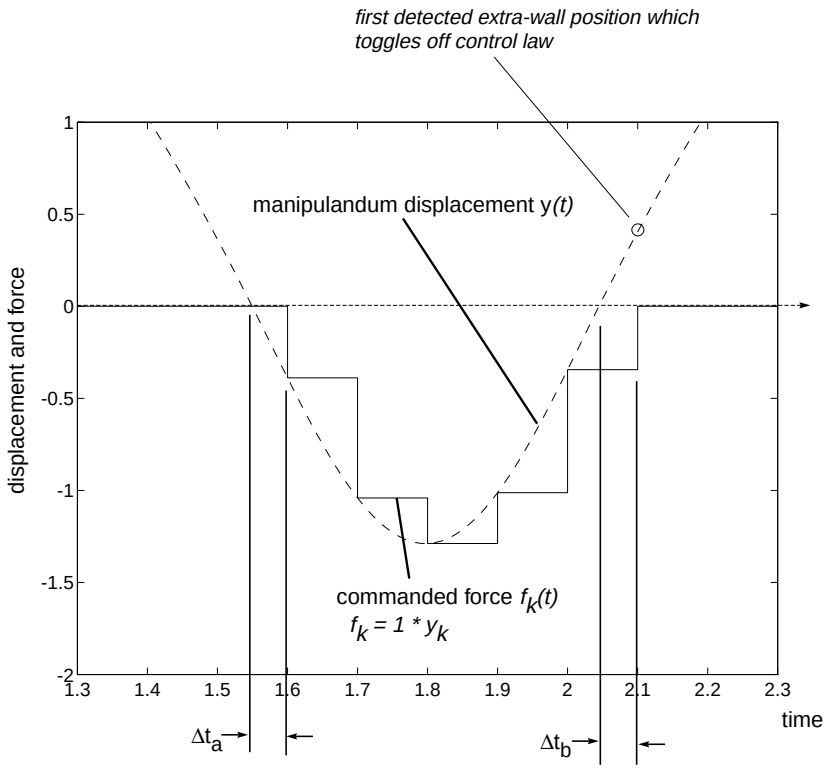

Figure 5.4: *Time-chart of modeled manipulandum position and control signal*

As is depicted in the force-displacement plot of Figure 5.3, the on-average small valued force on the way into the wall and large valued force on the way out of the wall can be seen. But two time periods in particular are highlighted. The first, labeled Δt_a, is the delay in turning on the wall controller and Δt_b is the delay in turning off the wall. If it were not for the discrete implementation of the algorithm, whether the wall was on or off would be strictly a function of configuration. However, because of discrete, constant step size sampling, the switching times become a function of both configuration and time. Because the crossing of the configuration threshold will not in general occur on the sample times, yet model switching is only admitted on the sample times, errors will be committed. For example, the wall will likely first be turned on with the wall's spring in

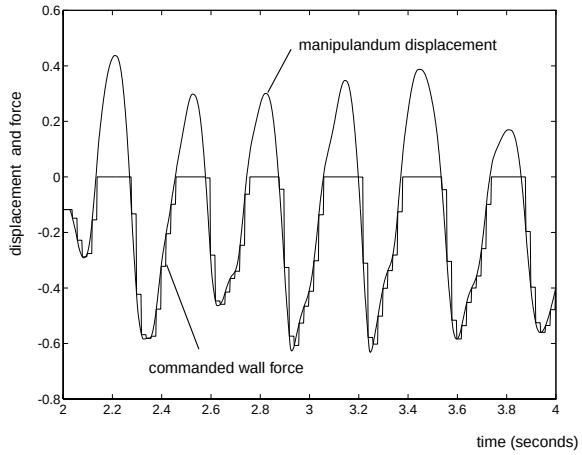

Figure 5.5: *Time-chart of experimental manipulandum position and control signal*

a slightly compressed state because the first sampled position to trip the conditional will not be located precisely on the boundary. Such an error can produce energy since the spring now stores energy without the requisite work having been done on the spring.

Upon leaving the wall, the first sampled manipulandum position will, in all likelihood, lie outside of the wall rather than just on the threshold. In this case, since the wall will immediately be turned off, (assuming no computational delays) by the algorithm of Table 5.1, the wall will no do extra work on the human (the force for the last wall-on sample period will still push away from the wall). Thus we see that the asynchrony of the wall on/off switching times with the sampling times are energy-producing. [3]

5.1.3 Claims

Interaction with virtual objects through a haptic interface imposes two conditions on the simulation algorithm, namely that the simulation be run with a constant step size and that the discrete simulation output be zero-order held before acting through the motor on the interface. These two requirements are simple consequences of the fact that the haptic interface and the human are continuous systems and we would like to implement the virtual object dynamics with a digital computer,

[3] Note that this asynchrony can alternatively be interpreted as misalignment in space of the on/off switching points with the wall location (threshold) itself. These two interpretations are both valid, but neither provides more affordances to the solution since in the final analysis, the switching effects are a function of both position and (sampled) time. A worst-case or averaging assumption must be made before one effect can be treated independently of the other, as will become apparent in the next chapter.

making the whole a sampled data system. The goal is to have the apparent dynamics of the haptic interface change in response to varying configuration of the virtual environment just as the dynamics of a real wall change in response to varying configurations. That is, the wall algorithm should execute without any dependencies on either the step-size or the relative placement of discontinuities and sampling times, despite the sampled-data nature of the linking of the discrete algorithm (simulation) to the continuous interface and human.

We have developed two controllers which render, in sampled data settings, passive virtual walls. The first of these is more easily generalized to other virtual objects, for it approaches the problem from the simulation perspective (to be defined shortly). The second controller is the simpler of the two, but it is not quite as extensible. Its design makes use of tools from digital control.

Both designs begin with modeling assumptions about the human operator and the haptic interface device, as motivated by the discussions above. In the case of the first design, these models are used to make predictions of behavior one half sample ahead so that the approximate half-sample delay of the zero-order hold can be effectively cancelled. Such strategies have been used in flight simulation with motion display [48]. Additionally, these models are used to derive the state at the inter-sample threshold crossing times from state information at the sampling times. These inter-sample states are used to synthesize special control signals which stay within the constraints of their sampled-data implementation yet drive the system to the state appropriate in the corresponding continuous-time system. This is an application of 'deadbeat control' techniques which are available in digital control design but not in continuous control design. Thus the errors of inter-sample switching are fully accounted for.

In the case of the second design, models of the human and the haptic interface device are directly incorporated into a controller design process which takes place in the digital domain. A zero-order hold equivalent model of the human and device is used during the design of a digital wall algorithm (controller) which, both when simulated in the digital domain and implemented in the sampled data system, yields the desired results. Finally, to handle behavior irregularities due to possible intersample switching times, a method identical to that used in the first design (deadbeat control) is used to correct for errors commited becuase of intersample threshold crossing.

Simulator versus Controller

We use the term simulator somewhat interchangeably with the term controller in the present context because both words apply. The digital computer is on the one hand a controller, making the haptic interface exhibit dynamics (in response to user input) which are not its own, and on the other hand a simulator, responsible for maintaining and advancing in time (in response to user input) the state of a virtual object. For the simulation of a static (memoryless) object such as the virtual wall, use

of a state is not necessary, the computer need not encode any dynamics, and the word controller is more appropriate. For a dynamical object, however, a numerical simulation scheme must be used, unless a solution of its equation of motion is available. Certain simulated variables are interpreted as control output and certain others are fed in real-time into the simulator, making its interpretation as a controller complete. The methods developed in this chapter for the virtual wall will be extended to more complex dynamical systems using the simulator viewpoint.

5.2 Literature Review

5.2.1 Robotics Literature

Contact instability observed in the virtual wall is closely related to contact instability observed between a robot, especially a force-controlled robot, and its environment. The destabilizing factors at play in the robotic problem have been addressed both analytically and with improved controller designs in numerous papers during the last decade. Indeed, contact instability is still considered one of the holy grails in the field of robotics. I will briefly review this literature here before reviewing the literature in the field of haptic interface design itself. First those papers from the robotics literature concerned with control design, then those that deal with the problem of contact instability with experimental investigations, and finally those containing analytical treatments will be reviewed.

Hogan has discussed the application impedance control [46] for the stable execution of contact tasts. stiffness control [], passive and active damping controls.

Xu, Hollerbach, and Ma present a nonlinear PD controller for contact transition in [107] Their controller features a set of PD gains which are a function of the force error and force error rate. During periods of robot motion away from the target, gains are increased with the aim of suppressing chatter.

Lin and Yae [60] present an improved impedance control design where contact force is extracted from a force sensor signal. A unified controller results, with no switching terms. The force feedback signal simply kicks in upon feedback of a non-zero force signal.

Mills [67], Mills and Lockhorst, [69] and Mills and Goldenberg [68] have presented a suite of discontinuous controllers for contact transition control along with stability proofs to guarantee asymptotic tracking of the commanded force upon contact, even given inadvertant bouncing. These papers call upon some of the russian litterature on discontinuous control [88], [87], [1].

Hyde and Cutkosky [50] conducted a comparative experimental study into the performance of five control designs which had appeared in the literature, each aimed at executing smooth, transition between motion and force control. The five controllers were simple discontinuous control, impedance

control, active impact damping, active nonlinear damping, and input preshaping. Input preshaping was in fact a new introduction to the available contact control schemes. Noise sensitivity and ease in parameter selection and performance was compared.

Teleoperation

Hannaford and Anderson [41] discuss an experimental and simulation study into hard contact through a bilateral teleoperator. Forces sensed at the slave site are displayed at the master and positions sensed at the master are fed forward to the slave. A sixth order nonlinear but time invariant model was used for the human in the simulated system. The simulated system demonstrated behaviors very similar to the experimental setup. The effects of a heavier grasp on the handle were noted in experiment and duplicated in simulation with increased damping in the human operator model. Hannaford and Anderson mention on-line estimation of the parameters of the human operator as a possible extension for more robust control.

Robotics Literature: Analysis

These are controllers designed to execute tracking or minimize bouncing, in a discrete controlled robot. None of them address the destabilizing effects of the ZOH or intersample threshold crossing. These authors are primarilty interested in robot behavior, not emulation or exhibition of the dynamics of changing kinematic constraints. Controllers explore many methods to attain transition in minumum time (including adding physical compliance, virtual damping, switching laws, and so on), motivated in part by the extremely robust performance in such tasks which humans can demonstrate.

Our aims in coming up with virtual wall controllers are somewhat different than those of the robotics community interested in contact transition control. We actually want to keep the bouncey behavior, insofar as it represents the dynamics of the referent wall. But we want to extinguish the bouncey behavior which arises from the wall implementation in a haptic display device with a discrete controller. To the extent that destabilizing the effects shared, however, we are interested in the same issues as the robotics community. We are both interested in the dynamics of a discontinuous system, where making and breaking contacts is at play.

A number of papers in the spirit of identifying destabiling effects and designing controllers which compensate for those effects have appeared. For example, Eppinger and Seering treat the destabilizing effects of sensor/actuator non-collocation in force-controled robots in [29]. Effects of the robot and workpiece dynamics on the stability of a simple force-controlled system are considered when these dynamics intervene between sensor the point of application of control effort. Unstable

behavior on the part of a robot is predicted using continuous-domain lumped-parameter models. Although dicontinuities were not analyzed within this paper, this unstable behavior often exhibits itself as a limit cycle of repeated contact and loss of contact with a workpiece. In the present chapter, by contrast, we have chosen not to treat the destabilizing effects of non-collocation. We assume that system dynamics do not intervene between the human/manipulandum contact and the sensors.

Regarding the half sample delay introduced by the zero order hold, the field of robotics has certainly acknowledged its destabilizing effect, and some robotic controllers have enjoyed the application of digital control design techniques which account for this delay. Neuman, for example, has contructed a fully discrete model of a robot manipulator [77]. Interestingly, though, as computer speeds increase and half sample delays grow shorter, attention to the benefits of design with digital techniques has waned. Most analyses and design efforts use continuous-domain methods again these days. Growing interest in the contact instability problem in robotics seems to have coincided with waning attention to sampled-data effects. Therefore, very little effort has been applied to analyze the effects of sampling on contact instability. I have not seen the sampling operator or zero order hold singled out for treatment in either design efforts or analysis efforts with regard to its effect on contact instability in the robotics literature. The effects of intersample threshold crossing have not been addressed in the design or analysis of robotic controllers to date.

Bartolini? [8] Utkin [96]

5.2.2 Haptic Interface Design Literature

In Chapter 4 of his thesis [86], Rosenberg presents the results of a human subject study on the rigid wall percept. The standard virtual wall algorithm, Eq. 5.2 was presented to human subects along with many variants (which incorporated exponential springs, thresholded dampers, unidirectional dampers, and position-offset dampers) for subjective ratings of "hardness", "crispness" (initial contact), "cleanness" (final release) and overall "wallness". Rather than defining these terms in physical variables, Rosenberg had chosen them to allow his subjects to fully characterize and also somewhat decompose the manifest behaviors of the various virtual wall algorithms. Among these behaviors was of course the same non-passive behavior underlying chatter, which Rosenberg describes as 'bouncy' contact. 'Sticky' release was also encountered in Rosenberg's damped walls. The various dampers were specifically designed to quell this undesirable and un-wall-like behavior. Rosenberg advocates a design methodology which he calls *design for perception*, in which one attempts to create percepts pertaining to virtual objects through perceptual decomposition rather than physical modeling. Perceptual decomposition involves human subject testing of various algorithms, some physically inspired, others simply tricks, to ascertain the optimum algorithm.

Although the *design for perception* approach may yield initial promise and point to some efficient shortcuts in algorithm design, we believe that perceptual decomposition is actually a more difficult problem than physical modeling. See Gillespie [34] for further discussion on this interesting topic. There exist many unanswered fundamental questions in psychophysics, and trial-and-error approaches become less attractive once the initial hurdles are overcome. The work in this thesis lies in the area of physical modeling, but certainly approaches like *design for perception* (where the *percept* rather than the algorithm take center stage) are useful to gauge the severity or consideration-worthiness of a problem. We take the work pertaining to the virtual wall of Rosenberg as further motivation for our own work. The supposition that contact instability is indeed a problem worth addressing with new controller designs is underlined by works like that of Rosenberg.

One of the earliest analytical treatments of contact instability associated with the virtual wall was in a paper by Minsky and Ouh-Young *et al.* [71]. This paper will be further reviewed in the next chapter —here I will just point out that the destabilizing effects of delay were addressed analytically. Interestingly, these authors attributed the delay to computational delays rather than the zero order hold operator. Virtual wall designs other than the standard controller were not explored, and design in the digital domain was not undertaken, though mention of digital analysis was made.

Colgate *et al.* present the results of an analysis of the passivity of certain virtual wall implementations to the virtual reality community in [23]. The analysis itself is covered in a pair of journal articles to be discussed in detail in the next chapter. The goals underlying this paper (and the supporting papers) are very much the same as the goals of [71] by Minsky *et al.*: "to delineate regions in parameter space that lead to suitable wall implementations". Colgate, however, formulates the problem as a question of passivity rather than one of stability.

Colgate endorses the use of a passivity criterion to characterize virtual walls rather than stability because passivity is a property of the wall alone, non-inclusive of the unpredictable human properties, making it possible to express passivity criteria without reference to properties of the human operator. Furthermore, the observed sustained or growing oscillations can be taken to be evidence of *active* walls since the human cannot be the source of energy as discussed above (because these oscillations are outside the range of voluntary motion, and sustained oscillations are not observed with physical walls). The main interpretation of the results of Colgate's passivity analysis is that an implementation of a virtual wall must contain some inherent physical damping if it is to be behave passively.

Colgate *et al.* cite the zero order hold as the path of energy introduction into virtual walls, and give an intuitive explanation similar the one above in section 5.1. Colgate *et al.* point out that, although coupled stability and isolated stability imply contact stability for continuous-time systems, these conclusions cannot be drawn for discrete time systems. Intersample threshold crossing and

attendant simulation errors may emerge in sampled data systems. In a discrete analysis, Tsai and Colgate [95] treat the unilateral nonlinearity explicitly, but do not treat intersample effects.

Colgate's passivity analysis may be viewed as a very thorough and elegant treatment of the effects of the zero-order hold in this sampled data system. The elegance lies in the manner in which the specific dynamics of the human operator are excluded from the analysis and the fact that the end result may be applied to controller design. Colgate's contributions, however, lie in the area of analysis rather than design. New controllers or controller design methods which directly account for the destabilizing effects of the zero order hold are not suggested.

5.2.3 Simulation Literature

Numerous researchers in the field of numerical simulation are concerned with simulation across discontinuities, especially discontinuities embodied by changing kinematic constraints. Now that numerical simulation of multi-body dynamical systems is finding so much application in computer graphics, certain papers have appeared which address the problems directly. The desire to run these simulations in real-time has grown strong of late with emerging interest in *interactive* systems, bringing this literature close in spirit to our concerns.

Researchers in this field, however, have the luxury of being able to neglect the dynamics of the human in the consideration of accurate simulation across discontinuities since there is no loop closed through mechanical variables when the human is coupled through a unmotorized interface device. Furthermore, the effects of instability using only visual display are far less disturbing than in the case of haptically displayed instability. Thus less attention has been paid to difficulties in simulating across discontinuities in this field to date. Non-physical behavior due to limited update-rate is often dismissed as less important since it can be effectively treated with more computing power. Such treatment is less readily applied in haptic interface because of hardware interfacing requirements.

Howe [48] presents a technique which essentially amounts to half-sample prediction to account for the equivalent half-sample delay introduced by a zero-order hold in real-time flight simulation with motion display. Howe also prevents the effects of computational delay from surfacing with similar simulate-ahead strategies. Given that motion display does not actually close the loop through mechanical variables as does haptic display (unless feed-through dynamics are present), Howe does not need to include human dynamics in the model which is used for half-sample prediction.

A paper by Lin and Howe [59] addresses the issues involved in real-time simulation of a discontinuous system with a discrete constant step-size simulation algorithm. Their approach to the real-time simulation of systems with discontinuities takes care of errors arising from an occurrence

of a discontinuity between simulation steps. The dynamic equations of a control subsystem with discontinuities are integrated off-line with a sufficiently small step-size, repeated over a matrix of initial conditions and inputs, to produce a function table which may be used during run time to efficiently account for intersample placement of a discontinuity.

In summary, although applied in robotics, it appears that digital control design techniques have not been applied in the field of haptic interface to date. While appearing to some extent in numerical simulation, compensation for the effects of intersample threshold crossing in simulation across discontinuities has not been applied in haptic display. The application of deadbeat control to correct for errors of intersample threshold crossing in a discontinuous system is new.

Our approach to the problem of contact instability in virtual walls can be contrasted with that of most other researchers to date. Rather than setting out to delineate regions in parameter space which will ensure passivity of a virtual wall using a particular (somewhat standardized) controller [25] [23] [24], we embarked on another effort —to design an altogether new controller which would meet some special performance criteria. These were in fact a rather stringent set of performance criteria: that the controlled system (despite its sampled data structure) behave exactly as another continuous but switching system, as discussed above. Under the virtual wall application, the criteria consisted of non-introduction of energy into the system upon contacting the wall.

Although methods for the design of sampled data controllers have appeared widely in the literature, and methods for handling switching models in constant step-size simulation algorithms also exist [48], application of both methods to sampled data controller design has apparently not been made. The switching sampled-data controllers developed in this chapter do not have precedent in the literature. Although contact instability has received much research attention in the Robotics literature, authors have not ascribed this instability to the sampled-data implementation of robot controllers.

5.3 Modeling the sampled data system

To expound the controller designs, I discuss another discontinuous system, simpler than, but still representative of the virtual wall: the lossless bouncing ball. Shortly I will defend the bouncing ball as a suitable model of human exploration of a virtual wall through a haptic interface, but first, I discuss our use of the bouncing ball model as a kind of work bench for the design of virtual wall controllers.

Basically, we seek a simulation algorithm for the elastic floor upon which a ball bounces which yields 'realistic' bouncing behavior, yet which adheres to certain 'structural restrictions' placed on the algorithm itself. We know that a lossless ball bouncing on a perfectly elastic floor should bounce

forever, never gaining or losing height. Any simulated behavior in which the ball bounces higher, lower, or even irregularly shall be deemed 'non-physical'. For a ball with damping, any simulated ball motion which deviates from the motion of its referent (continuous) damped bouncing ball will indicate problems, or a failed floor simulator design. Note that the motion of the continuous bouncing ball is easily produced with simple simulations, or even with the careful use of two switching analytical solutions.

The structure of the simulator itself will be further elucidated below, but stated simply, the simulator structure is modeled after the very sampled data system which is the haptic interface displaying a virtual wall. Thus, upon coming up with a suitable floor simulation, that simulation algorithm may be reinterpreted as a wall controller and implemented directly in the actual physical hardware. So the immediate goal which directs the controller design is simply to eradicate 'non-physical' simulated behavior which arises in the bouncing ball simulation because of the 'structural restrictions' placed on the simulator.

5.3.1 The Bouncing Ball as Allegory for the Virtual Wall

The bouncing ball, I shall now argue, is in fact a rather good model of a human interacting with a virtual wall through a haptic interface. Again, the tendency of the manipulandum to chatter or iteratively bounce against a virtual wall is the behavior which we aim to study. Presumably, if the bouncing ball is a good model, its unstable *simulated* behavior using a certain floor simulation algorithm will predict chattery *controlled* behavior when that 'floor simulator' is implemented as a 'wall controller'. In order to generate a particularly simple bouncing ball model (and likewise a simple work bench), we model the ball and floor without any dissipative elements. We work under the premise that bouncing ball instability or unbounded growth (due to the introduction of energy from an unsuitable simulation algorithm) will indicate unfitness of the corresponding wall controller (where the introduced energy will possibly only cause sustained oscillations because of damping inherent in the physical system).

Figure 5.6 shows a massive ball B in two configurations. The configuration in Figure 5.6 a) corresponds to the *floor off* condition. Here, the ball is being acted on solely by the force of gravity. In Figure 5.6 b), corresponding to the *floor on* condition, the ball is being acted on by the force of gravity and the force of a special spring $k(y,t)$. The rest position of the wall spring is taken to be zero ($y_{wall} = 0$).

The ball represents the manipulandum and the hand or finger of the human operator. For now, elastic and dissipative effects in the human and manipulandum are not modeled. The force of gravity represents a constant force exerted by the human on the manipulandum. Backup for such a crude

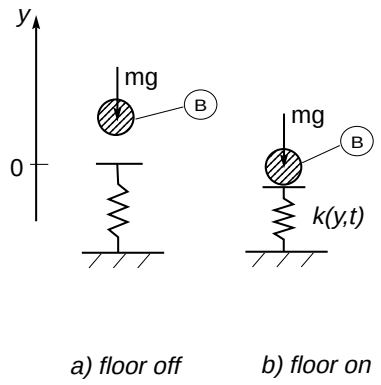

Figure 5.6: *Modeling the Bouncing Ball*

approximation of the human in the virtual wall system under study is provided, as mentioned in section 5.1, by noting that the observed chatter is much higher than the frequencies characterizing the intentions of the human and that a human does not need to do anything once oscillations begin in order to sustain them except passively maintain that hand impedance found to be destabilizing. Various gravity fields will be used to represent various forces exerted by the human. A representative impedance (inertial, damping, and spring forces) for the human is also neglected for the present for simplicity.

The spring force of the floor depicts the virtual wall, but, as mentioned above, in order to allow for the subsequent reinterpretation of the 'floor' as a controller in a sampled-data system, the floor simulator is specially structured as follows. Rather than by Newton's impact law (with a coefficient of restitution), the floor is modeled as a compressible spring with a constituent law f=k(y,t) to be determined by the designer. Simulations of this model are allowed to communicate with simulations of the ball only at certain time points (the sampling times). Furthermore, the force response of the floor shall be held constant between sampling times to depict the zero-order-hold. The floor is thus simulated as a discrete system and the ball as a continuous system. While the ball depicts the manipulandum and human, (plant) the floor depicts the discrete controller.

Figure 5.7 shows this discrete floor as the feedback controller in a block diagram with the ball as plant. The discrete floor controller $C(z)$ is shunted into the loop only during the *floor on* periods of simulation by the unilateral nonlinearity.

Our simulations thereby include the zero order held forces and appropriately compute the response of the continuously modeled ball to these discontinuous (staircase-shaped) forces. The switching times are also constrained in this simulation as they are in the sampled data system, to lie on

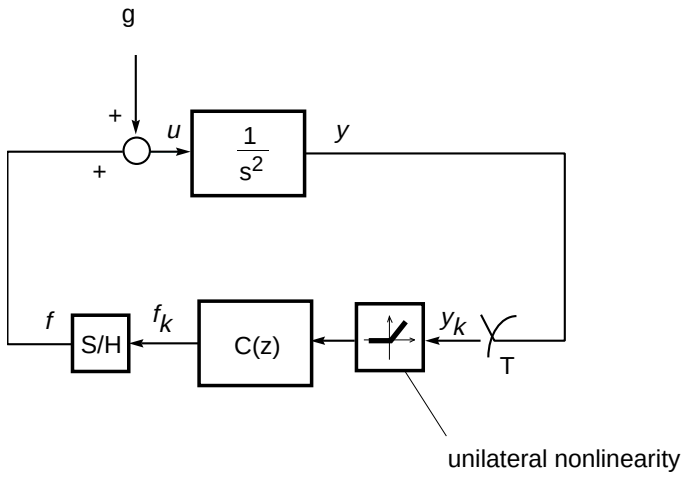

Figure 5.7: *Sampled Data System-inspired Block Diagram for the Bouncing Ball Simulator*

the sampling times.

We begin with a floor control law for simulation like that most commonly used in virtual walls (the control law inspired by its continuous time counterpart), namely $f_{spring} = k(y_k - y_{floor})$. For the floor, we set $y_{floor} = 0$ as in Figure 5.6. The differential equation (model) for a unit mass responding to gravity and the force of a spring is simply:

$$\ddot{y} = -g - f_{spring} \tag{5.3}$$

which has the following equivalent form as state-space model, using $x = [\ y \quad \dot{y}\ ]'$:

$$\dot{x} = Ax + b(-g - f_{spring}) \tag{5.4}$$

where

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \text{and} \quad b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Note that the reaction force of the spring is a forcing term (on the right-hand side) in this model. The motion of the ball is simulated in intervals, each the length of one sampling period T, with an ODE solver. At each sampling point (between intervals) an indicator function is checked (whether the ball is inside the domain of the floor). If the indicator function evaluates true, the reaction force of the spring, f_{spring}, is computed according to the control law, and held constant for the duration of the next sampling period. If the indicator function evaluates false, f_{spring} is set to zero.

Pseudocode for our simulator is given in Table 5.2. Appendix A contains MATLAB code for this algorithm.

Table 5.2: **Pseudocode for the Sampled Data Bouncing Ball**

$k = 0$

loop {

apply ODE solver to (5.3) from $t = kT$ to $t = (k+1)T$

append interval to stored solution vector $x(t)$

if $(y < 0)$ then $f_{spring} = K * y$

else $f_{spring} = 0$

$k = k + 1$

}

plot $x(t)$.

In order to produce the motion of a continuous bouncing ball for comparison, several methods can be used, (including simple evaluation of model solutions). The method most parallel in structure to the above uses an ODE solver on time-intervals which are pre-determined from the solution. The following will demonstrate determination of the switching times.

Starting from a state outside the floor, the time to floor strike is computed using the model of a free unit mass flying ball:

$$\ddot{y} = -g \tag{5.5}$$

The solution to this simple model is of course:

$$y(t) = y_0 + v_0 t - \frac{1}{2} g t^2 \tag{5.6}$$

The time to floor strike is simply its root, given by:

$$\Delta t_I = -v_0 - \sqrt{v_0^2 + 2 g y_0} \tag{5.7}$$

where y_0 and v_0 are the initial state of the free-flying ball.

The *floor off* model is used to simulate the motion for the time period Δt_I. From the time of wall entry (on threshold), the time to exit the floor is computed using the solution of the *floor on* model given the floor entry state. The *floor on* model is simply a sprung mass (this time with the

spring incorporated into the left hand side of the equation):

$$m\ddot{y} + ky = -g \tag{5.8}$$

which has an equivalent from as a state-space model:

$$\dot{x} = Ax + b(-g) \tag{5.9}$$

where

$$A = \begin{bmatrix} 0 & 1 \\ -k/m & 0 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The solution to this differential equation is:

$$x(t) = C_1 cos(\omega t) + C_2 sin(\omega t) - g/w_0^2 \tag{5.10}$$

where

$$\begin{aligned} C_1 &= y_0 + g/w_0^2 \\ C_2 &= v_0/w_0 \\ w_0 &= \sqrt{k/m} \end{aligned}$$

The first root of this equation, or time Δt_{II} which yields $y = 0$ is:

$$\Delta t_{II} = \frac{1}{w_0}\left[\pi + atan2(C_1, -C_2) + sin^{-1}\left(\frac{g/w_0^2}{\sqrt{C_1^2 + C_2^2}}\right)\right] \tag{5.11}$$

where $atan2$ is the four-quadrant arc tangent function.

The *floor on* model is then used in the ODE solver for the time period Δt_{II}. Then the the process starts over.

Pseudocode for this algorithm is given in Table 5.3. Appendix A contains MATLAB code for the same algorithm.

Figure 5.8 shows the simulation results using the "Sampled Data" simulator (Table 5.2) along with the results of the continuous bouncing ball simulator (Table 5.3) for reference, which is shown with a dashed line. The staircase-shaped trace of the reaction force from a unit-stiffness spring,

Table 5.3: **Pseudocode for the Continuous Bouncing Ball**

```
loop {
    compute time to floor strike Δt_I using (5.7)
    apply ODE solver to (5.5) from t to t + Δt_I
    append interval to stored solution vector x(t)
    compute time to floor exit Δt_II using (5.11)
    apply ODE solver to (5.8) from t to t + Δt_II
    append interval to stored solution vector x(t)
}
plot x(t).
```

f_{spring} is also ploted.

Figure 5.9 shows a close-up of the boxed portion of Figure 5.8.

We very quickly note that this floor produces non-physical behavior; the ball bounces higher and higher. At this point, the design of improved controllers is underway.

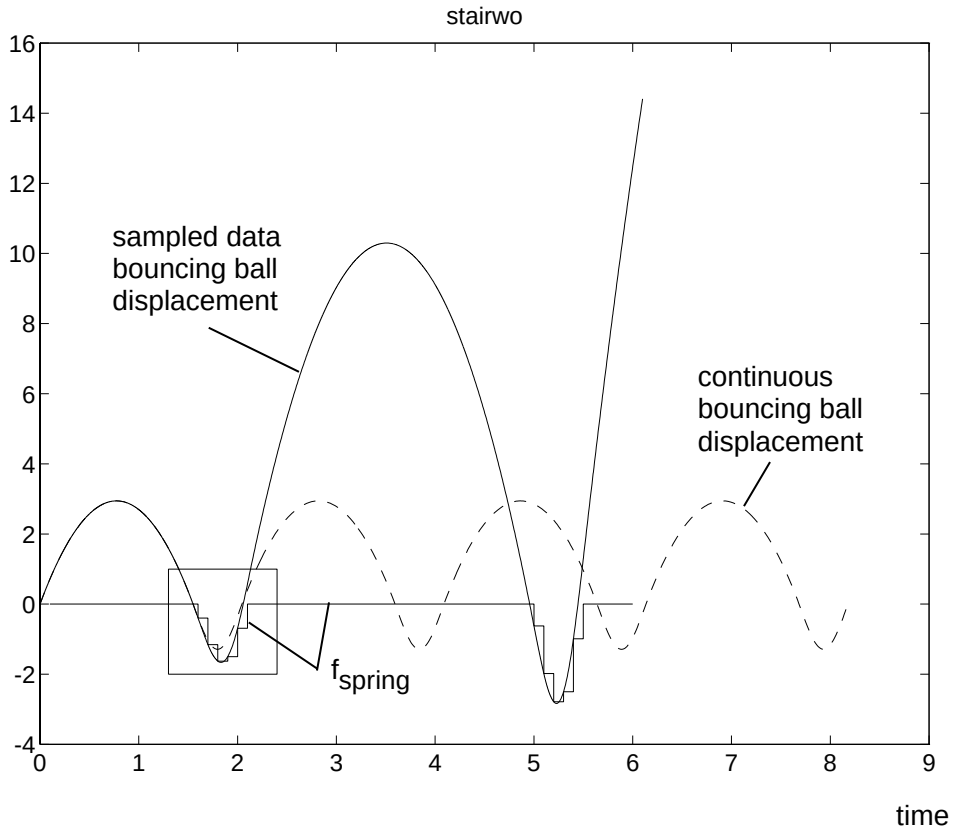

Figure 5.8: *Sampled Data Algorithm Simulation Results*

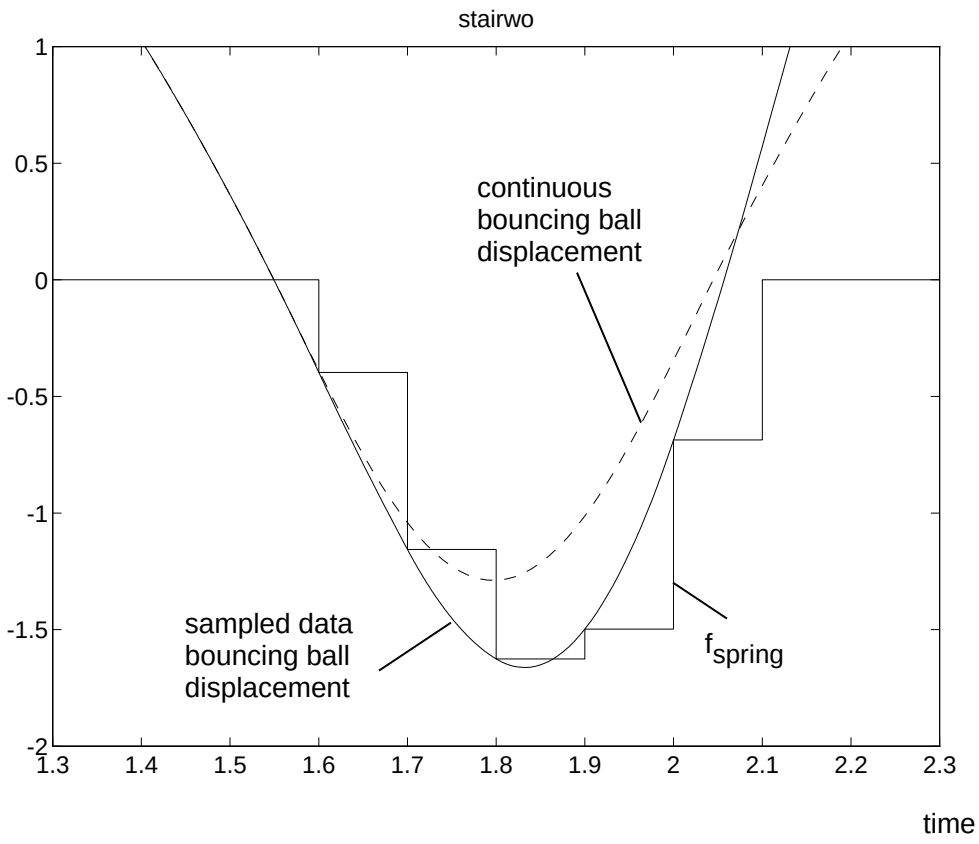

Figure 5.9: *Sampled Data Algorithm, Close-up*

5.4 Controller design

Two controller designs will be presented, the first based on half-sample prediction and the second on design in the digital domain. Both of these controllers are intended to compensate for the destabilizing effects of the zero order hold within our sampled data system. An enhancement to both of these controllers, which compensates for the effects of intersample threshold crossing, will be presented in the third part of this section.

5.4.1 Half Sample Prediction Controller

Our first improved controller is inspired by noting that the effect of the zero order hold can be approximated by a half-sample time delay. An improved controller will be constructed by adding half-sample prediction to the algorithm in the hopes of cancelling the effect of the zero order hold.

We already have a model of the target system (a sprung mass) in hand in the form of Eq 5.8. At each sample time $t = kT$, we can simulate ahead using this model or, even easier, evaluate its solution, Eq. 5.10, at $t = kT + T/2$ and starting from the present state $[y_0 v_0] = x(kT)$, to predict the ball's position a half sample ahead. This predicted position is then used in the standard control law (Eq 5.2).

Note that the model whose response is the target (the sprung mass, Eq. 5.8), rather than the model of the mass responding to the zero-order held force (the mass on wall, Eq. 5.4) is used to make the predictions.

Our simulation pseudocode using this half-sample prediction now looks like the following:

Figure 5.10 shows the simulation results of the above half-sample prediction pseudocoded algorithm. The spring stiffness is unity $K = 1$.

Figure 5.11 shows a close-up of the boxed area in Figure 5.10. Here one sees how the trace of the zero-order held spring force intersects the continuous floor position trace approximately midway between sample times. This propitious intersection leaves half of the inscribed area above and half below in contrast to the staircase plot which the reader may recall from Figure 5.3. Our new algorithm does not climb "uphill both ways" as that of Figure 5.3.

The other interesting thing to note in Figure 5.11 is that, for the last wall-on sample period, the wall is actually exerting a tensile force, pulling on the operator. During this period, since the manipulandum is moving in the direction away from the wall, the operator is doing work on the wall. During motion away from maximum wall penetration, the wall is for the most part returning work to the operator, except for this last sample period. This fact will become important and will be further discussed in the analyses of the next chapter.

Though the half-sample prediction method outlined above yields vastly improved results over

Table 5.4: **Pseudocode for the Half Sample Prediction Bouncing Ball**

$k = 0$

loop {

apply ODE solver to 5.3 from $t = kT$ to $t = (k+1)T$

append interval to stored solution vector $x(t)$

if $(y < 0)$ then {

$A = (y0 + g/w_0^2)$; $B = v_0/w_0$;

$y_{predict} = Acos(t + T/2) + Bsin(t + T/2) - g/w_0^2$

$f_{spring} = K * y_{predict}$

}

else $f_{spring} = 0$

$k = k + 1$

}

those given by algorithm 5.2, the results are still not perfect. Deviations from the desired bouncing path can already be seen in Figures 5.10 and 5.11, but excursions become especially apparent if the algorithm is allowed to continue for some time, as in Figure 5.12. Here we see that the bouncing height is irregular, sometimes higher, sometimes lower than the target height.

Reasons for this erratic behavior are twofold. Firstly, the ZOH is only approximated by a T/2 sample delay; its full effect is more complex. See, for example [31]. The next section will present a design using digital controller design tools which fully accounts for the effects of the ZOH.

Secondly, the wrong control law will be used to compute the reaction force for certain portions of those sample periods which contain the entry and exit, thus exerting a force inappropriate to that portion of the time period. Stated another way, turning on and turning off of the floor control law (entry into *if* block of pseudocode) will not necessarily occur when the ball height is $y = 0$. A fix for this second phenomenon which we call intersample threshold crossing will be presented in the section following the next.

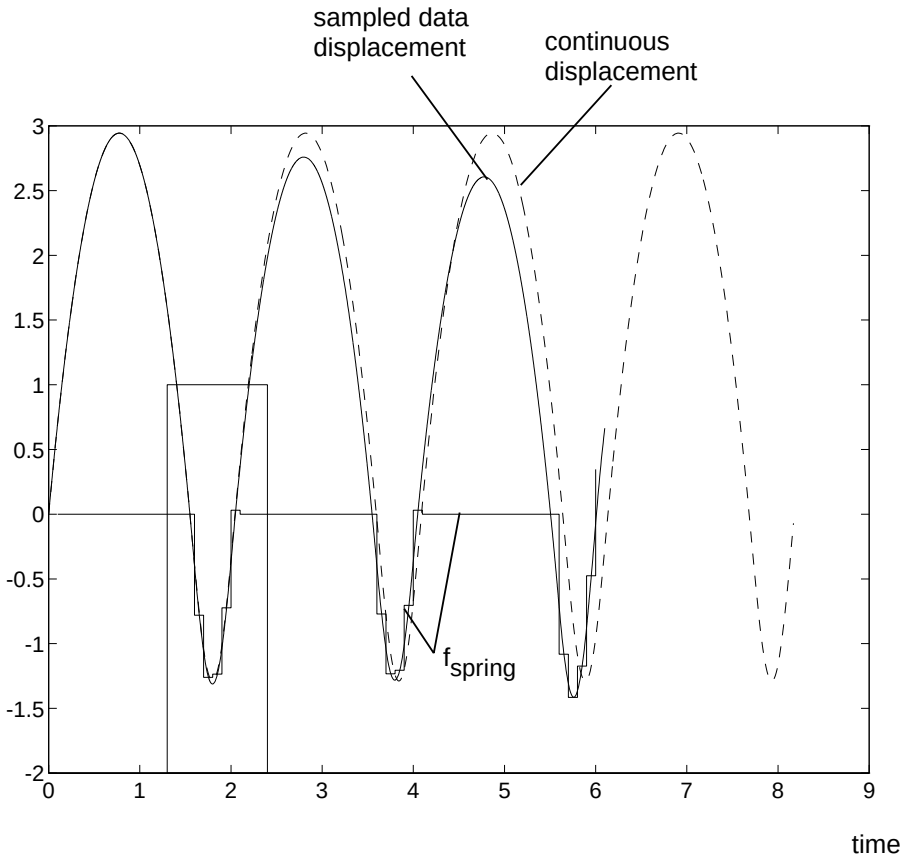

Figure 5.10: *Half Sample Prediction*

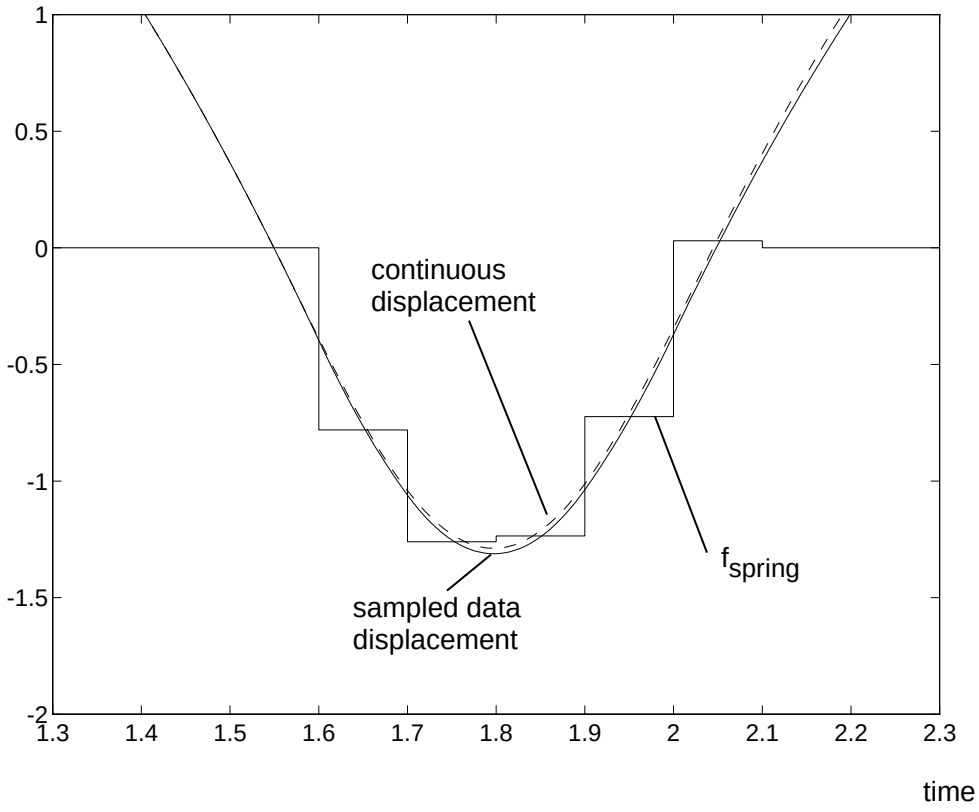

Figure 5.11: *Half Sample Prediction, Close-up*

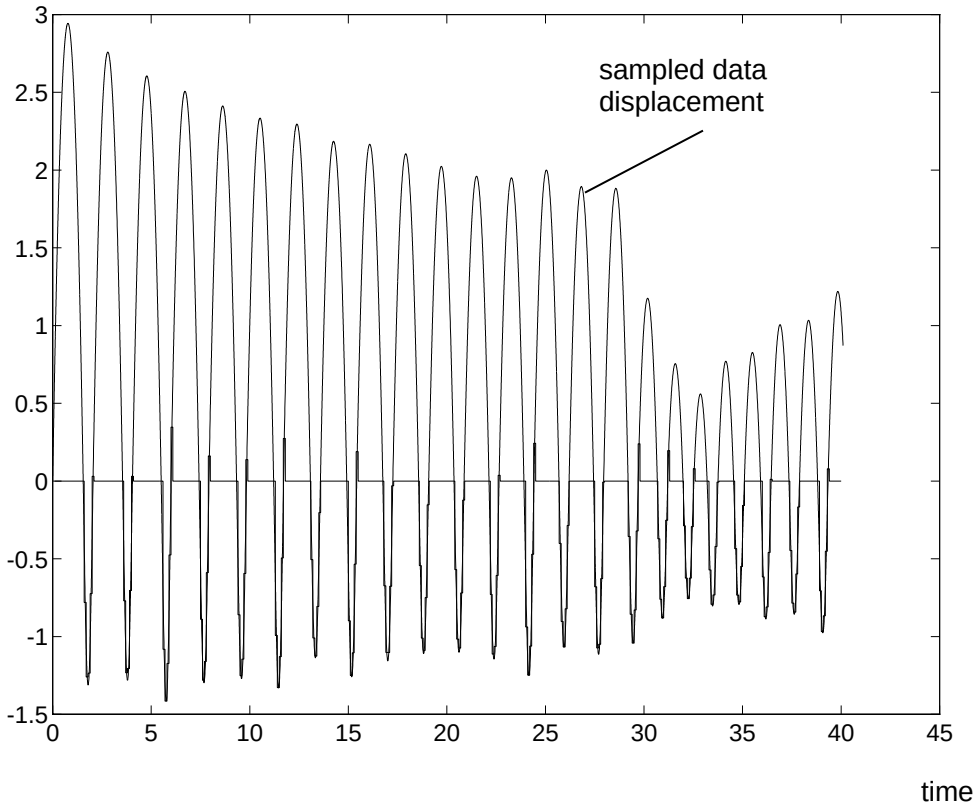

Figure 5.12: *Half Sample Prediction, Long-term Simulation*

5.4.2 Design in the digital domain

Another approch to the design of a controller for our bouncing ball simulator (an in turn for the haptic display of a virtual wall) exists. It is controller design in the digital domain. Our interim goal is to design a controller for a discretized plant, such that the response of the closed loop discrete system is the same as the desired (continuous bouncing ball) response on the sampling times.

First the ZOH discrete equivalent of the desired dynamics (a sprung mass) are found using a table of $\mathcal{Z}$ transforms.

$$\mathcal{ZOH}\left\{\frac{1}{s^2+\omega_0^2}\right\} = \frac{(1/\omega_0)(1-cos\omega_0 T)(1+z)}{z^2-(2cos\omega_0 T)z+1} \tag{5.12}$$

Where $\mathcal{ZOH}\{\cdot\}$ denotes the zero-order hold discrete equivalent of the bracketed expression. Note that $\mathcal{ZOH}\{\cdot\} = (1-z^{-1})\mathcal{Z}\{\cdot\}$.

This discrete equivalent has two complex poles and one zero. The pole locations (roots of the characteristic equation), λ_1 and λ_2, as shown in Figure 5.13, are identified as the desired root locations for the closed loop system comprising the controller being designed and the simple plant $H(s) = 1/s^2$. These root locations correspond to the response of the referent system, a sprung mass, expressed in the digital domain.

We will perform the design of that controller in the digital domain. For this purpose, the ZOH discrete equivalent of the plant $1/s^2$ is found.

$$\mathcal{ZOH}\left\{\frac{1}{s^2}\right\} = \frac{T^2(z+1)}{2(z-1)^2} \tag{5.13}$$

Using full state feedback control, the poles of this system may be placed arbitrarily. We simply choose to place the roots of this controlled system at the root locations λ_1 and λ_2 using pole placement. The full state feedback gains $k = [k_1 k_2]$ which place the closed loop dynamics of the sytem at these target locations are available from a pole placement algorithm such as that known as Aizerman's method.

The response using this controller is essentially the same as that of the previous section, the half-sample prediction controller. Figure 5.14 shows the response of the sampled data bouncing ball with the new discrete controller overlaid on top of the target displacement trajectory. The system is still not perfect, due to errors in switching the controller at the wall threshold (intersample threshold crossing). The next section will address this very problem, producing results in the end from the sampled data controller which are indistinguishable from the continous target system.

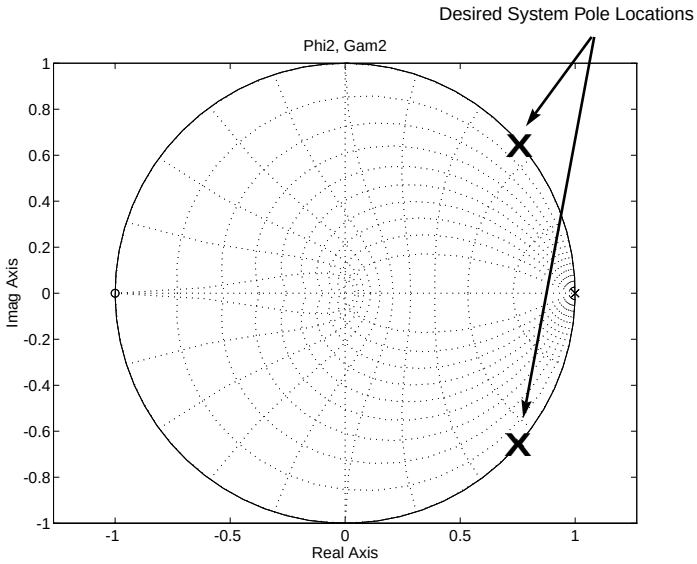

Figure 5.13: *Desired root locations on the unit circle*

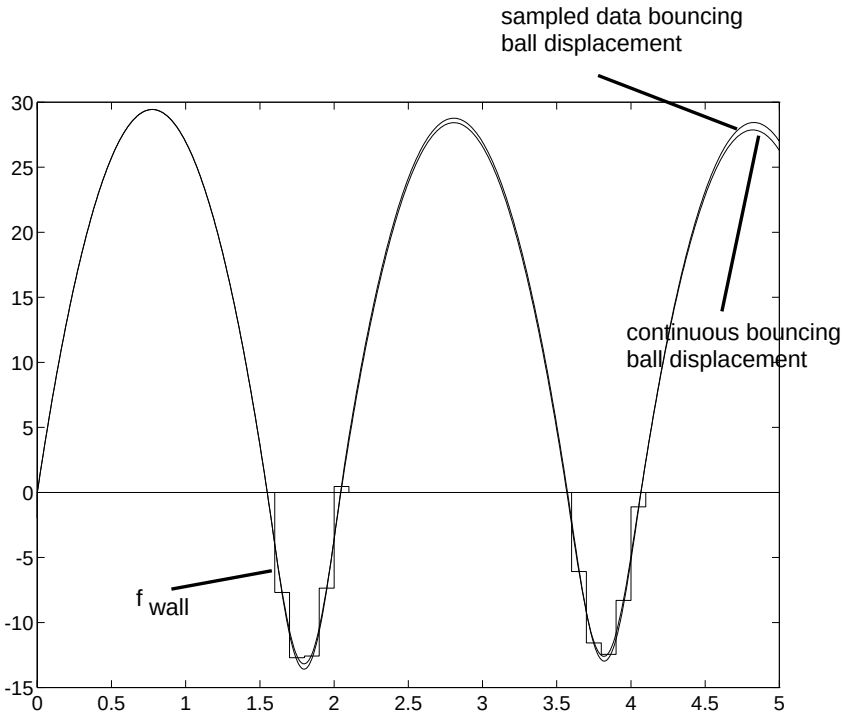

Figure 5.14: *Performance of Controller designed in the digital domain*

5.4.3 Compensation for Asynchrony

The effects of inter-sample threshold crossing can be fully accounted for by using solutions of the two models, the free-flying mass and the sprung mass. For clarity, the approach is first outlined: First, we deduce the inter-sample entry and exit times from information available at the sampling times. We then predict, again using the model solutions, what state the ball would be expected to attain, if the system were continuous, at the first sampling time outside of the floor. Finally, we drive it to that desired state (as shown below) with the last two (zero-order-held) force values inside the floor.

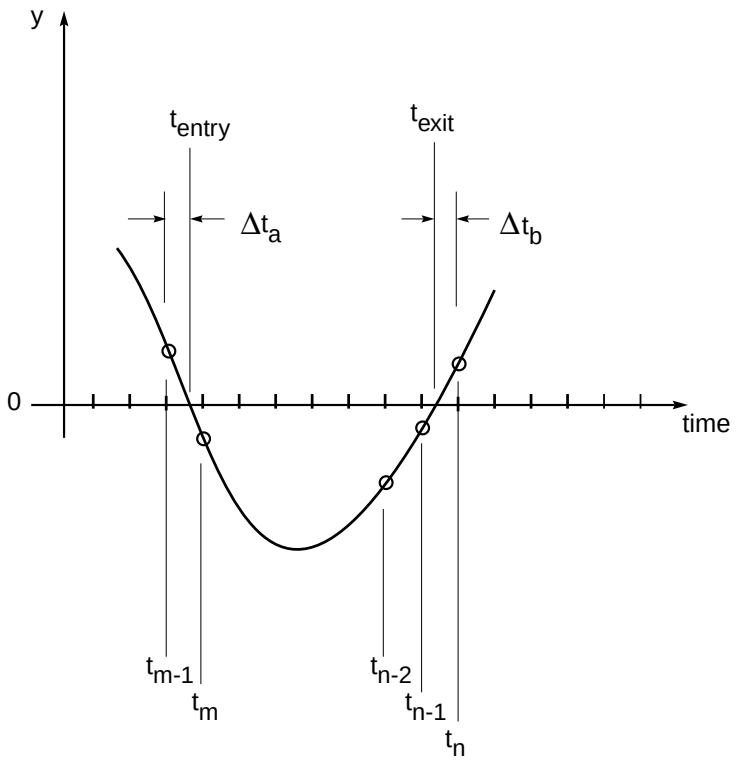

Figure 5.15: *Sampling Points in a Typical floor Strike*

Figure 5.15 shows the arbitrary placement of the sampling times on the motion path of a simulated strike of the floor. The first sampling time for which the ball is inside the floor is designated t_m and the first outside of the floor is designated t_n. Reference to these time points will be made in the following discussion.

The state of the ball at the first sampling time outside of the floor, $x(t_n)$ encapsulates the action

of the floor simulator. The state of the ball at t_n resulting from a continuous floor is obviously not the same as the state resulting from either the typical algorithm floor (see Figure 5.9) or the improved half-sample prediction floor algorithm (see Figure 5.11). But, having assumed models for both the *floor on* and *floor off* conditions, the state of the ball resulting from a continuous floor (which we shall call x_d, (d for *desired*) can be extracted from the state information available on the sampling times. To find this state is a multi-step process as follows:

First, the root of the free-flying ball model is found, using the known state at the last sampling time before entry $x(t_{m-1})$ in Eq. 5.7. This time interval is designated Δt_a (see Figure 5.15).

The full state at floor entry $x(t_{entry})$ is found using the solution to the free-flying ball model, Eq. 5.6 and its derivative,

$$\dot{y}(t) = v_0 - gt \tag{5.14}$$

by evaluating these at $t = \Delta t_a$ and $[\ y_0 \quad v_0\ ]' = x(t_{m-1})$.

Now the time at which the exit from the floor is made, t_{exit}, is found using the initial condition $[\ y_0 \quad v_0\ ]' = x(t_{entry})$ in the root of the sprung mass solution (Eq. 5.11),

Note that the state at t_{exit} is already known, quite simply, because it is an undamped wall:

$$x_{exit} = \begin{bmatrix} y_{entry} \\ -\dot{y}_{entry} \end{bmatrix} \tag{5.15}$$

where the shorthand x_{entry} stands for $x(t_{entry})$.

Knowing the time of exit t_{exit}, the time remaining to the first sampling period outside of the floor Δt_b (see Figure 5.15) is available:

$$\Delta t_b = t_n - t_{exit} \tag{5.16}$$

where $t_n = kT$ and k is the smallest integer such that $y(kT) > 0$ since the last floor encounter.

Finally the desired state at time t_n can be computed by evaluating the solution to the free flying mass model, Eq. 5.6, starting at the exit state $[y_0 v_0]' = x_{exit}$ and $t = \Delta t_b$. This state $x(t_n)$ we shall call the desired state x_d.

Driving the system to the desired state

We now know where this system is to be driven if it is to behave as the continuous bouncing ball. Now the problem has become how to shape the control force f so that, at t_n, the state will arrive at x_d.

This is a problem of controller design in the digital domain. We start by discretizing the continuous part of the system, which is of course only seen by the discrete controller at the sampling times kT.

We find the ball's zero-order hold equivalent so that the effects of the zero-order hold is included in its discrete representation. Since our model of the display device and human are quite simple, their zero-order hold discrete equivalents are easily found by an application of the definitions of Φ and , (See, for example: [31]):

$$\Phi(T) = I + AT + \frac{(AT)^2}{2!} + ... = exp(AT) \tag{5.17}$$

$$, = \left[\int_0^h \left(I + \tau A + \frac{\tau^2 A^2}{2!} + ...\right) d\tau\right] b \tag{5.18}$$

$$= \left[T + \frac{T^2 A}{2!} + \frac{T^3 A^2}{3!} + ...\right] b \tag{5.19}$$

But for our model Eq. 5.3, A^2 is a matrix of zeros, so in terms of the sampling period T, Φ and , are simply:

$$\Phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \quad , = \begin{bmatrix} T^2/2 \\ T \end{bmatrix} \tag{5.20}$$

Given that the discretized system is only second order and that it is fully controllable, it will only take two steps to drive it to any desired state.

The response of a discrete system $\{ \Phi , c \}$ to the input sequence $u(kT)$, $k = 0, ...n$ can be expressed:

$$x(n) = \Phi^n x(0) + \mathcal{C} \begin{bmatrix} u_{n-1} \\ u_{n-2} \\ \vdots \\ u(0) \end{bmatrix} \tag{5.21}$$

Where the controllability matrix $\mathcal{C}$ is given by:

$$\mathcal{C} = \begin{bmatrix} b & \Phi b & ... & \Phi^{n-1} b \end{bmatrix} \tag{5.22}$$

Given controllability ($det(\mathcal{C}) \neq 0$), this equation can be inverted for the control sequence

$$\begin{bmatrix} u_{n-1} \\ u_{n-2} \\ \cdots \\ u(0) \end{bmatrix} = \mathcal{C}^{-1}(x_d - \Phi^n x(0)) \tag{5.23}$$

Since our system is only second order, it will only take two inputs to drive the system to state x_d:

$$\begin{bmatrix} u_{(n-2)} \\ u_{(n-1)} \end{bmatrix} = \mathcal{C}^{-1}(x_d - \Phi^2 x0) \tag{5.24}$$

where $x0$ is the state $x(t_{n-2})$, two samples before the first sample outside of floor, with n pertaining to Figure 5.15.

The control value u_1 is used at t_{n-2} and u_2 is used at t_{n-1}

Deadbeat Control results

Figure 5.16 shows the simulation results of the above T/2 prediction pseudocoded algorithm, 5.4.1. The spring stiffness is $K = 1$.

Figure 5.17 shows a close-up of the boxed area in Figure 5.16.

5.5 Experimental Results

The half-sample prediction controller described in the forgoing section was coded in C++ and tested experimentally using our haptic interface. To facilitate collection of intersample position, the control law was evaluated only every tenth servo cycle, while data was collected every servo cycle. Thus a wall controller was mimicked whose implementation had a sampling rate one-tenth that of the actual servo rate. A virtual wall was displayed in the center of the workspace of one key. Sequential comparisons of virtual walls rendered either with the standard controller or the new controller were made. Controllers were exchanged, sampling rate varied in increments of 50 Hz from 100 to 1000 Hz, and target stiffness was varied during run-time through a simple interface.

The new virtual wall controllers performed significantly better than the old. Walls rendered with the new controllers did not support sustained oscillations for those parameter values under which the old virtual walls in fact did support sustained oscillations.

Figure 5.18 shows the bouncing behavior of the old wall. Four strikes were made against the same wall, but with alternating controllers in action. Approximately the same posture and muscle activation levels were adopted by the human subject. The first and thrid strikes were against a

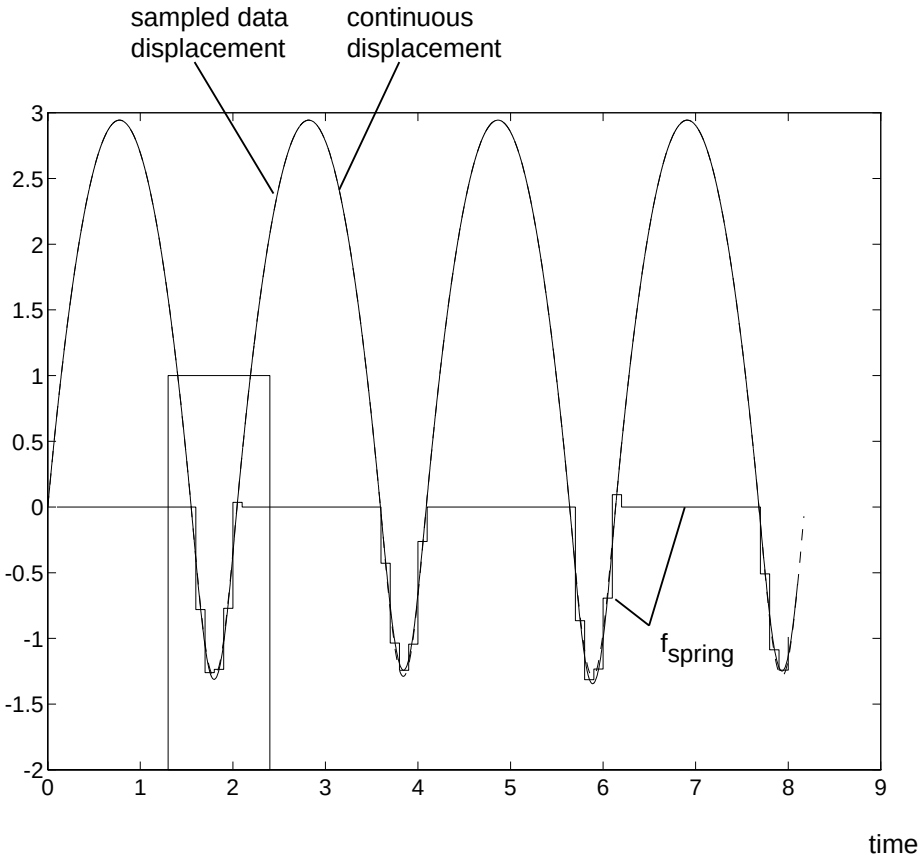

Figure 5.16: *Full Control Algorithm*

virtual wall using the old controller. The second and fourth strikes were against a wall using the new controller. The controller out-performs the old, as evidenced by the lack of chatter.

Figure 5.19 shows a closed -up of the boxed portion shown in Figure 5.18. Using a wall stiffness scaled to unity, the control effort (force for display) from both controllers is shown overlayed on the position trajectory of the manipulandum. The solid line shows the old control effort (displayed during this time period) and the dashed line shows the new control effort (not displayed during this time period). The intersection of the new control effort with the position at the half-sample times is evident.

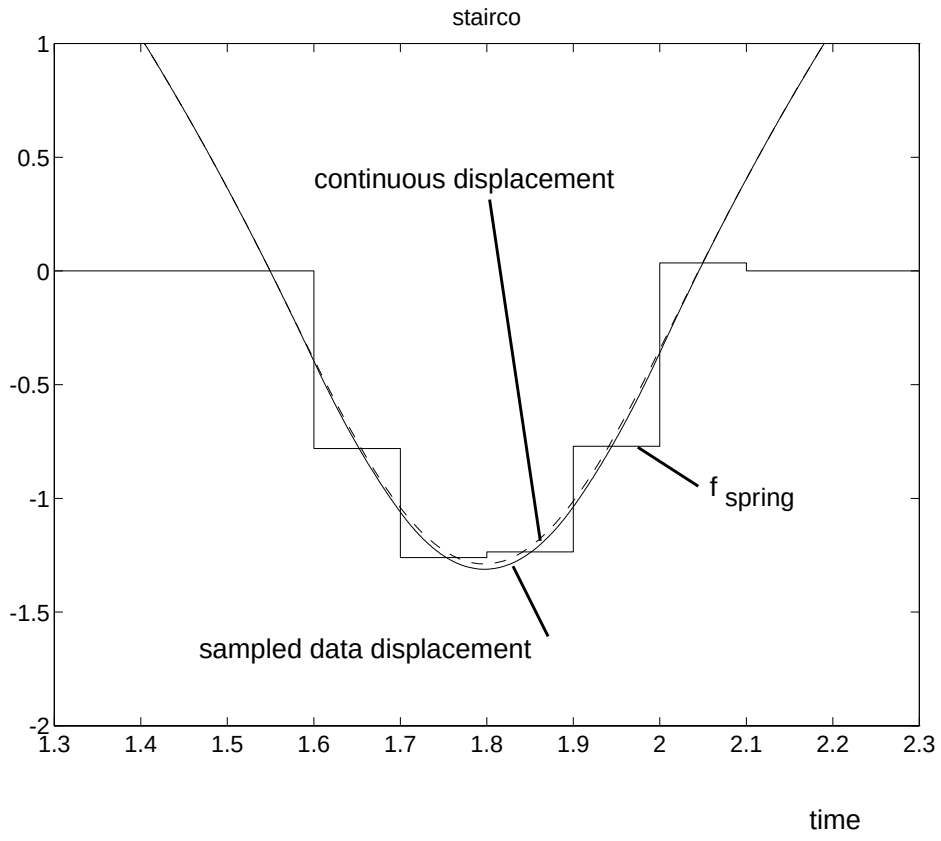

Figure 5.17: *Full Control Algorithm, Close-up*

5.6 Discussion

The forgoing controller exposition and experimental presentation have concentrated solely on the rendering of undamped virtual walls. New controllers were developed which prevent the chatter commonly observed in undamped walls from arising. Yet the new controllers use full state feedback, with gains on both sensed position and sensed velocity. Perhaps it is not fair to compare a wall control law which uses both position and velocity feedback gains with wall controllers which only use position feedback gains. Comparisons between the new controllers and *damped* virtual walls would be more fair. Damping is often added to virtual walls to enhance their stability, usually by trial and error. This trial and error process in the design of damped walls makes their direct comparison difficult. It can indeed be said, however, that the significant improvements exhibited by the new controllers can be attributed to the addition of positive velocity feedback.

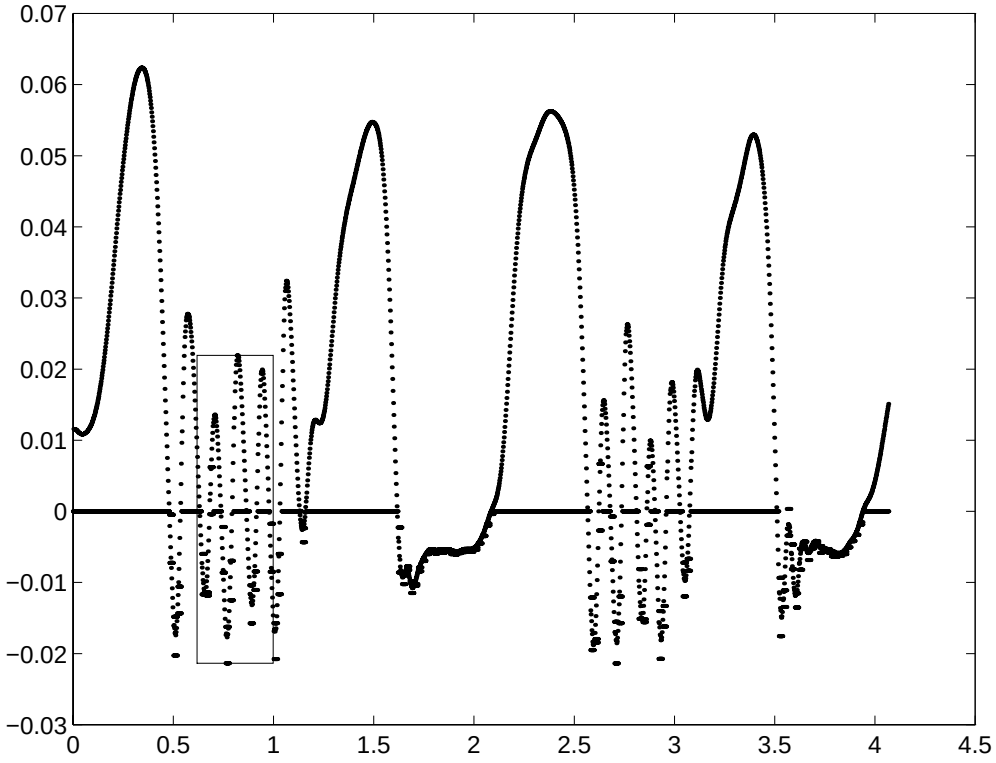

Figure 5.18: *Four stikes of a virtual wall, with two controllers*

The point of the new controllers, then, is not so much their highly improved performance as promulgated in the previous section, but rather that a method for designing feedback gains has been presented which, when implemented in the sampled data setting, will exhibit the desired stiffness. Damped walls with desired stiffness and damping can also be designed with the same method. The model to be used in the half-sample prediction controller would then be a damped second order oscillator rather than the undamped oscillator used in the exposition of this chapter. The desired pole locations in the design-in-the-digital-domain procedure would turn up inside the unit circle. The same Aizerman pole placement algorithm, however, would come up with appropriate feedback gains on position and velocity just as in the undamped case shown. The risk of coming up with a wall which feels damped because excessive damping was chosen (providing more stabilizing influence than necessary) is not a problem with these new design techniques.

The half-sample prediction controller features another important advantage over the old damped virtual wall design. That is, when a first-difference approximation is made for the velocity from a position signal, appropriate model-based filtering will automatically be added. Essentially, the

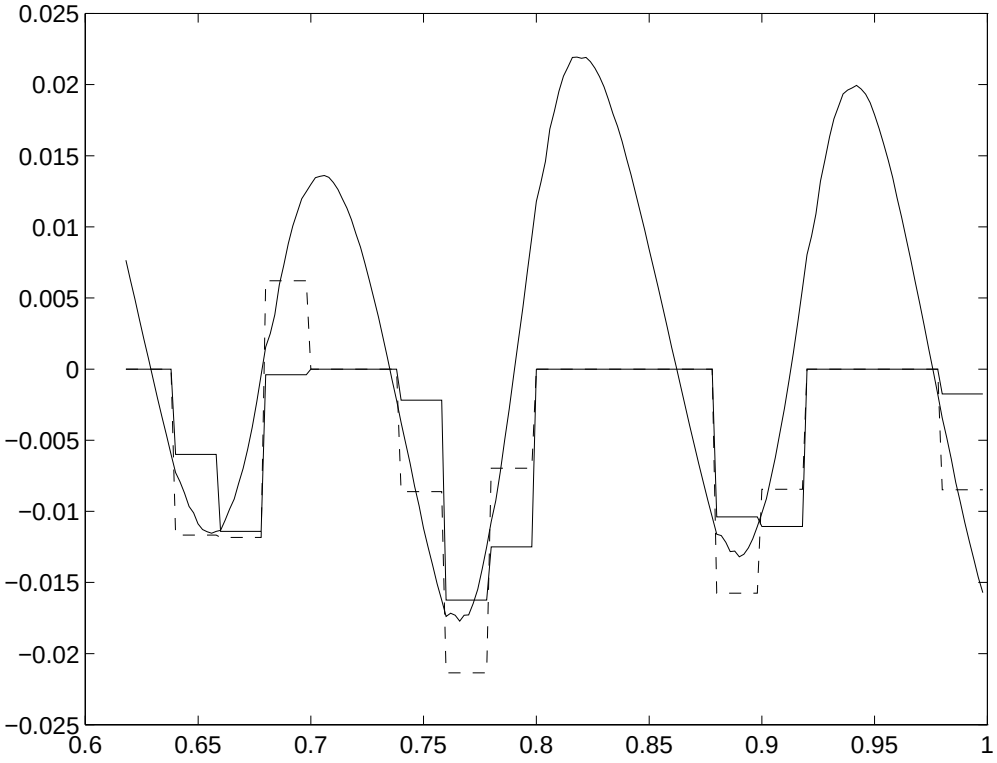

Figure 5.19: *Four stikes of a virtual wall, with two controllers*

use of the present sensed velocity in a model to produce a future state is like a Kalman filter. Many authors have observed and Colgate has shown that increased damping coefficient can lead to unstable behavior from the virtual wall when that damping coefficient is used on a first-difference approximated velocity.

These comments pertain to the compensation for the zero order hold. The dead-beat control techniques may of course also be used in the design of damped walls. This technique for the quelling of effects from intersample threshold crossing does not have precedent in the literature.

5.7 Summary

This chapter has addressed the formulation of controllers designed to create the illusion, for a human operator, of a passive wall which opposes motion with spring forces when the operator drives a manipulandum past a threshold. A set of virtual wall controllers has been presented which are immune to two certain destabilizing factors which otherwise play a large role in the implementation

of virtual walls, causing chattery behavior. Both of these factors are a consequence of the sampled data setting within which virtual wall controllers must operate. First, the zero order hold effectively introduces loop delays which, under position feedback (as called for by a stiff wall) introduce energy into the closed loop system. The effects of the zero order hold are quelled in the new virtual wall controllers using two design techniques: half-sample prediction and design in the digital domain. Both of these controller enhancements were shown to substantially improve virtual wall performance (decrease chatter) in simulation and in experiment. Close scrutiny of simulation, however, shows that something is still amiss in these sampled-data virtual walls: the issue of intersample threshold crossing, the second factor which we address.

The controller for the virtual wall is a *switching* controller, which attempts to cause the manipulandum to take on the dynamics of a two-mode system —an object alternately in contact with a compliant wall and in free motion, as driven by the human operator. Because the on/off switching is based on a signal which is only discretely sampled, timing errors are introduced into the switching behavior of this controller. Turn-on and turn-off times are not synchronous with the wall encounter times, but rather with the next available sampling times, occurring quite independently of the threshold crossings. The effects of intersample threshold crossing may be fully counteracted, however, by model-based deduction of the timing errors from state information collected on the sample times and with an application of dead-beat control. The fact that the controller is discrete works to our advantage this time, since deadbeat control is able to perfectly compensate for the errors of intersample threshold crossing and deadbeat control is *only* available in discrete controller implementations.

Perhaps the most noteworthy aspect of these new controllers is that, in their design and operation, a simple time-invariant model of the human operator has been assumed and utilized. Justification for this rather bold move was drawn from fact that the dynamics in question are outside the range of voluntary movement for humans. Further backup is provided by system simulations which model the human input as a constant bias force that exhibit the same chattery behavior which is seen with the old (un-improved) virtual wall controllers. Also inspiring confidence is the fact that these new controllers work so well. Virtual walls implemented with the new controller do not support sustained oscillations where those with the standard controller under the same wall stiffness and sample rate parameters will. In fact, the sampling rate may be substantially reduced or the stiffness substantially increased before the new controllers break down to exhibit irregular or chattery behavior.

The half-sample prediction controller also has an interpretation as a simulation method. It is a constant step-size numerical simulation scheme designed to simulate a discontinuous (two mode) dynamical system without sensitivity to the step size or relative placement of the steps to switching

times. Rather than making use of backstepping to locate the switching times between sampling points, it relies on a model of the system (virtual object, interface, and human) in each of its constraint configurations to deduce the switching times from state information available at the sampling points and then account for the effects of a change in configuration occurring at an arbitrary time between the sampling points with an application of deadbeat control.

In summary, although implemented in the sampled data setting, the end effects of the new controller are those of a continuous switching controller–the consequences of the zero-order hold on the control input are not apparent and the effects of intersample switching are quelled.

5.8 Extensions and future work

Natural extensions to the method include the following:

- Lookup tables can be substituted for the functions mentioned above to facilitate speed (ease computational overhead).
- Non-autonomous systems can be handled with minimal error if the external independent agent can be assumed to have bandlimited behavior that is, be predicted 1/2 to 1 sample ahead. To reasonable accuracy with say, a polynomial fit.
- Interaction force could be sensed and used in the prediction
- Improved models of the human might yield better controllers.
- More complex kinematic constraint changes (existing in dynamical models rather than the simple static wall) could be handled by simulation rather than model solution evaluation as used above.
- Perhaps cover constraint changes between human and manipulandum (loss of contact) using similar methods.
- sensor resolution
- use deadbeat control techniques to drive system to known appropriate state when known is derived by other methods, perhaps by examnimation of an integral of the motion, energy conservation, or power exchanges with user, as sensed or calculated.
- computational delays can also be compensated out.

The use of data from a human-characterizing experiment to personalize the rendering of a virtual environment has precedent in the technology of audio environments. A person's head related transfer function (HRTF), which describes the sound filtering properties of their pinnae (outer ears), head, and shoulders as a function of sound source location, can be obtained by comparing a recording made with a small microphone placed in the ear canal with a known sound source in a known location with respect to the person's head. By then filtering a synthesized sound according to that HRTF, that sound can be effectively placed in a position in space (with repsect ot the person's head). Thus, with a particular person's HRTF in hand, an important cue used by humans for localization can be synthesized for that person. These spectral cues are especially imporant for localizing sounds which emanate in the median plane (the plane normal to a line betwen the ears) where inter-aural intensity differences and inter-aural time of arrival differences (the other two cues for localization) play no role.

Depending on the (sometimes) minute topological differences between the pinnae of two persons, a virtual sound environment synthesized with the HRTF of one person may or may not provide successful localization cues for another person. Also, the fact that the HRTF may only be used to place virtual sound sources with repsect to the person's head necessitates real-time head-tracking and real-time filtering. The Convolvotron from Crystal River Corp. are among the best known commercially available products in this field. An associated product is available for the acquisition of HRTFs. See also the work of [] and [] for treatements of this subject.

We have introduced the incorporation of mechanical impedance properties of a person's finger or limb in controller designs for the particularization of the rendering of a virtual haptic environment. Guaranteed performance and optimum perceptual fidelity of synthesized mechanical properties are now possible. We look forward to on-line identification of the human mechanical impedance so that, when a user changes limb posture, properties of the haptic interface would continue to provide maximum impedance-range yet guaranteed passive behavior.