MASS-SPRING NETWORKS FOR SOUND SYNTHESIS

MASS-SPRING NETWORKS FOR SOUND SYNTHESIS

by DON MORGAN, B.A.

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the Requirements for the Degree Master of Computer Science

Department of Computing and Software

McMaster University

MASTER OF COMPUTER SCIENCE(2008) McMaster University
Hamilton, Ontario

TITLE: Mass-Spring Networks for Sound Synthesis
AUTHOR: Don Morgan, B.A. (Laurentian University)
SUPERVISOR: Doctor Sanzheng Qiao
NUMBER OF PAGES: xii, 94

Abstract

Physical sound synthesis produces sound by modelling the physics of musical instruments. One way to do this is to model the musical instrument as a network of masses, springs and dampers. This is known as a mass-spring system.

The goal of this thesis is to determine the stability and accuracy of the numerical methods commonly used to implement mass-spring systems for sound synthesis and determine if there are alternate methods that might provide superior results.

We show that the standard method used in mass-spring systems, when used on undamped systems, has no numerical damping, but does have frequency warping and is unstable at frequencies above $1/\pi$ times the sampling rate. As well, we find that for lightly damped systems the damping is accurate, but large damping can cause instability even at low frequencies. We present an algorithm to implement mass-spring systems using implicit numerical methods and show how a mass-spring system can be implemented using a wave digital filter. We compare the standard method implementing mass-spring systems with two higher order numerical methods: the fourth order Runge-Kutta, and the VEFRL algorithm, a fourth order symplectic algorithm. We find that the VEFRL algorithm is much more accurate than the standard method, but that this increase in accuracy does not noticeably affect the quality of the sound produced by the mass-spring system when used to simulate a vibrating string. The increased accuracy of the VEFRL method may, however, be useful for mass-spring spring systems used in physics or engineering requiring high accuracy.

Acknowledgments

I would like to thank Doctor Qiao for his many helpful comments and corrections.

Contents

List of Figures

Acronyms and Abbreviations

ACROE Association for the Creation and Research on Expression Tools. 6, 30, 90

ADSR Attack, Decay, Sustain and Release. 3

API Application Program Interface. 3

D/A Digital to Analog. 3

GPU Graphics Processing Unit. 10

Hz Hertz—cycles per second. 4

LTI Linear Time Invariant. 16

SGI Silicon Graphics, Inc. 89

VCA Voltage Controlled Amplifier. 3

VEFRL Velocity Extended Forest-Ruth Like. 3, 77, 78, 84, 85, 87, 88

WDFs Wave Digital Filters. 68

List of Symbols

a	Acceleration (meters per second^2).
C	Capacitance.
D_i	Damper element i.
F	Force (newtons).
f_s	Sample rate (samples per second).
G	Port conductance.
$\hat{H}(s)$	Transfer function, which is the Laplace transform of the output divided by the Laplace transform of the input.
$\hat{H}(z)$	Discrete transfer function, which is the z-transform of the output divided by the z-transform transform of the input.
h	Length of time step in the discrete system (seconds).
$i(t)$	Current.
k	Spring stiffness constant (newtons per meter).
L	Inductance.
$\mathcal{L}$	The Laplace transform.
M_i	Mass element i.
m	Mass (kilograms).
$\mathbb{N}$	The natural numbers.
$o(n)$	Delayed portion of a system at time step n.
S_i	Spring element i.
$S(s)$	Port reflectance, which is the Laplace transform of the reflected wave divided by the Laplace transform of the incoming wave.
t	Time (seconds).
$u(t)$	Voltage.
v	Velocity (meters per second).
$x(n)$	Position of mass in continuous system at time $t = h \times n$.
x_n	Numerical approximation of position of mass in discrete system at nth time step.
$x_{i:n}$	Numerical approximation of position of mass i in discrete system at nth time step.
$x(t)$	Position of mass in continuous system at time t.
$\hat{Y}(s)$	Laplace transform of function $y(t)$, i.e. $\mathcal{L}[y(t)]$.
$\hat{Y}(z)$	Z-transform of $y(n)$.
Z	Coefficient of viscous damping (kilograms per second).

Z_0	Port impedance.
$\|z\|$	Length of complex number z, which is $\sqrt{real(z)^2 + imaginary(z)^2}$.
γ	Viscous damping damping coefficient divided by the mass $\left(\frac{Z}{m}\right)$.
ϵ	Mass/length (linear density) (kilograms per meter).
μ	Damped frequency ($\mu = (1/2)\sqrt{4\omega_0^2 - \gamma^2}$).
σ	Decay rate on the s-plane.
χ	Pure delay factor of a system.
ω	Radial frequency (radians per second).
ω_0	$\sqrt{\frac{k}{m}}$, equivalent to ω for undamped systems.

Chapter 1

Introduction

1.1 Overview

Physical sound synthesis uses mathematical models based on the physics of sound production to synthesize sound. In this thesis we focus on mass-spring systems: networks of masses, springs and dampers. The mathematical model of mass-spring systems is based on differential equations. To approximate these differential equations on a digital computer, numerical methods are used. An important question to ask when using a numerical methods is: how well does this method approximate the differential equations used in the system? Numerical methods can become *unstable*. This means that the numerical solution can deviate arbitrarily far from the exact solution. In many cases the error can grow without bound, making the the results of the numerical method meaningless.

As well, we want to be able to quantify the accuracy of our approximation. The two most important parameters in the perception of a musical sound are its frequency and its decay rate. Most musical sounds are composed of a number of different frequencies. The lowest of these frequencies is called the *fundamental frequency*. The fundamental frequency determines the perceived pitch of the sound, and an error in the fundamental frequency will cause the sound to be out of tune. The higher frequency components determine the *timbre*, or tone colour, of the sound and errors in these components give the sound a different timbre than it should. An error in frequency caused by a numerical method is known as *frequency warping*.

The decay rate determines how quickly the amplitude (or volume) of the sound decreases. For example, a note on a piano will can be heard for 20 or 30 seconds after it is struck, while on a banjo it becomes imperceptible after only 3 or 4 seconds because the decay rate of a banjo is much larger than that of a piano. Numerical methods may add extraneous damping to vibrating systems that are undamped. This is known as *numerical damping*.

The previous literature on mass-spring systems used in sound synthesis describe how these systems work (i.e. the equations used and the finite difference equations used to approximate them), but have not addressed the issues of the stability and accuracy of the numerical methods used. This has been a important part of the criticism of mass-spring systems in the physical synthesis literature. Mass-spring system have been criticized as

being computationally expensive [3], lacking an analysis of stability [3], and having an unknown accuracy [13]. In this thesis we try to answer two main questions:

1. Using the "standard method" of implementing mass-spring systems, which we show to be equivalent to the *symplectic Euler method*, how accurate are the frequencies and decay rates of the sounds produced and under what conditions are these systems stable?

2. Given the answer to question 1, are there other numerical methods that, when used to implement mass-spring systems, would give superior results?

In the remainder of this chapter, we look at what sound synthesis is and how sounds are generated on a computer. We briefly describe some of the methods used in general sound synthesis and then some of the methods used in physical sound synthesis. We then give an introduction to the mass-spring model. We conclude with some reasons for using mass-spring models in sound synthesis.

In chapter 2, we present some basic numerical methods used in approximating differential equations and show the results of these methods when applied to simple mass-spring systems. In particular, we are interested in 3 side-effects of numerical methods: instability, frequency warping and numerical damping. To analyze these side-effects we introduce the z-transform. We describe the concept of a numeric method as a mapping from the s-plane to the z-plane. We then describe two previous implementations of mass-spring systems in sound synthesis: CORDIS-ANIMA and TAO.

In chapter 3, we use the techniques described in chapter 2 to analyze mass-spring systems. We begin with an analysis of the symplectic Euler method—the standard method used to implement mass-spring systems—when used to implement an undamped mass-spring system containing a single mass. We then expand the analysis to include damped mass-spring systems. We conclude by considering mass-spring systems containing multiple masses.

Chapter 4 looks at alternative methods that could be used to implement mass-spring systems. We look at three approaches for implementing mass-spring systems. The first approach is to use implicit numerical methods by removing the delay free loops. The second approach is to transform the mass-spring system into an equivalent electrical circuit and implement it as a wave digital filter. The third approach is to avoid delay-free loops by using explicit numerical methods. We consider multi-stage methods and in particular symplectic methods—methods which conserve energy.

In chapter 5 we compare the standard method of implementing mass-spring systems with two of the alternative methods discussed in chapter 4 by simulating a vibrating string. We first find the analytical solution of this mass-spring system and then compare it to results of each of the numerical methods.

The last chapter presents conclusions and some ideas for future work.

The contributions of this thesis are mainly in chapters 3, 4 and 5. The analysis of the symplectic Euler method in chapter 3 is new work, since previous papers on mass-spring systems for sound synthesis [33, 41, 8, 10, 9] have not provided an analysis of the stability and accuracy of the numerical methods used. We refer to the method used as

the symplectic Euler method since, although the previous papers in sound synthesis do not use this name, it is equivalent to the symplectic Euler method used in computational physics [19]. Among other properties, symplectic systems conserve energy, which is important in accurately simulating long lasting sounds. Our result in chapter 3, that the symplectic Euler method has no numerical damping for undamped systems, is not new, since it follows from the fact that the method is symplectic. The analysis of the frequency warping, damping and the extension to systems with multiple masses, has, to our knowledge, not been done before.

The analysis of implementing mass-spring system using the bilinear transform, the fourth order Runge-Kutta method, the Velocity Extended Forest-Ruth Like (VEFRL) method and wave digital filters in chapter 4 is also original work, as is the comparison of the symplectic Euler method with the Runge-Kutta and VEFRL methods in chapter 5.

1.2 Sound Synthesis

What is sound synthesis?

Synthesis is creating something new from a combination of pre-existing parts. The name sound synthesis comes from early analog synthesizers, which created sound by combining oscillators, envelope generators and filters. Figure 1.1 shows an example patch, with *VCA* representing a Voltage Controlled Amplifier and *ADSR* the envelope generator, with parameter values for the Attack, Decay, Sustain and Release.

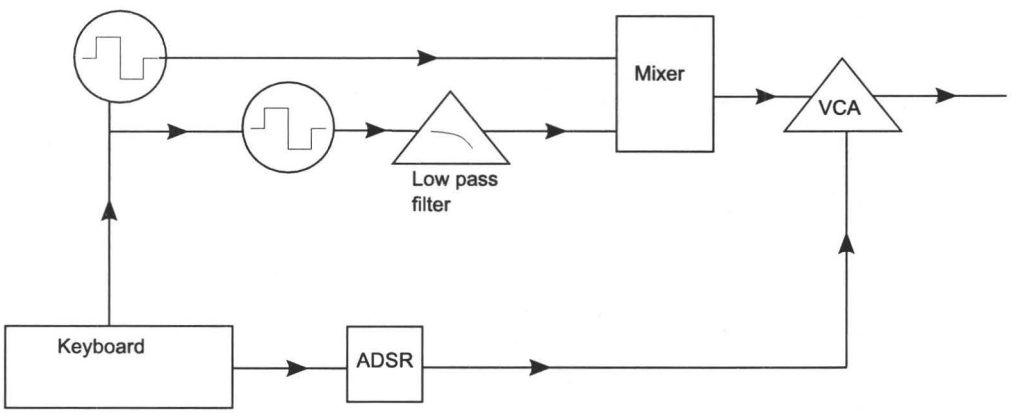

Figure 1.1: Patch on analog synthesizer

Creating sound on a computer

Most personal computers today come equipped with a sound card. A sound card has a Digital to Analog (D/A) converter to convert numbers representing sound pressure (called samples) to analog signals that produce sounds. Since there are many different sound cards, a programmer will usually use an Application Program Interface (API) such as RtAudio or JavaSound to send samples to the sound card. The API will handle the details of communicating with any of the standard sound cards. To produce sounds, the program need to fill a buffer with samples and then send the buffer to the sound card (see figure 1.2). For real-time sound the buffer must be refilled before the sound from

the previous buffer has finished playing — otherwise a gap in the sound is heard. The programmer can choose the sample rate and the size of samples—i.e. doubles or float.

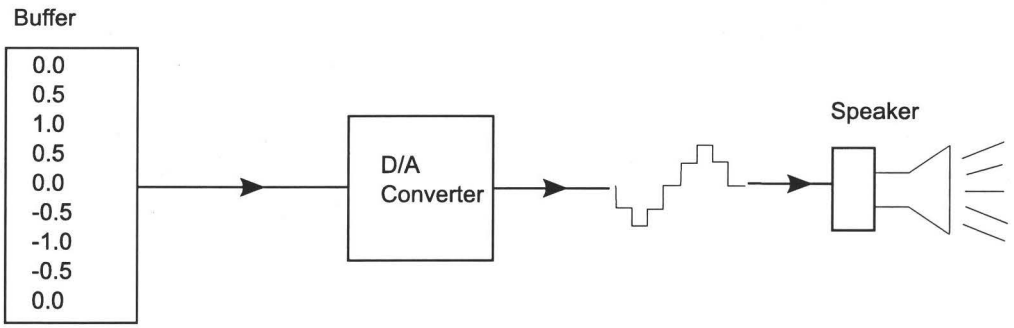

Figure 1.2: Sound on a digital computer

1.2.1 Types of Sound Synthesis

There are many different types of sound synthesis. Some of the more common methods are listed below.

- **Additive synthesis**

  A sound is made up of a number of frequencies. Any sound can be approximated by combining sine waves of different amplitudes and frequencies. A sine wave can be stored in a lookup table. By moving through the table at different rates different frequencies are generated —i.e. by skipping every other sample, a sine wave of twice the frequency of that of using every sample, is generated. Several sine waves can be generated and then added together.

- **Subtractive Synthesis**

  If a sound is made up of a large number of frequencies, additive synthesis is computationally expensive. The opposite approach is to start with a complex sound —e.g. random noise — and remove some of the frequencies by filtering.

- **Modulation Synthesis**

  Modulation synthesis creates more complex sounds by modulating the amplitude, phase or frequencies of basic wave forms. Many synthesizers in the 80's used frequency modulation. In frequency modulation a carrier signal (usually a sine wave) at frequency ω_c has its frequency modulated by a second signal, the modulation signal, at frequency ω_m. If the modulation signal has a low frequency — e.g. $10Hz$.— the result is a vibrato effect. At higher frequencies different timbres are created. Harmonic sounds are heard when the 2 frequencies have simple integer ratios. Non integer ratios produce metallic or bell like sounds.

- **Physical Modelling Synthesis**

  While other synthesis methods model musical sound, physical synthesis models the sound producing mechanisms of musical instruments [34]. This method takes

mathematical models of sound production, developed by physicists and acousticians, and simulates them on computers. Early models were very computationally expensive, but recent improvements in the efficiency of the algorithms coupled with greater processor speeds have resulted in some physical models that can run in real time.

1.2.2 Introduction to Physical Modelling in Sound Synthesis

Why Physical Modelling?

- **Curiosity**

  How do instruments produce sounds? Mathematical models of music go back as far as Pythagoras. How do we know if the models are correct? One way is to simulate them on computers. A model of an instrument which can produce a sound that resembles that of the actual instrument give us confidence that the model is correct.

- **More realistic synthesized sound**

  Many synthesized sounds have an artificial sound. They often sound too uniform because the frequency spectrum does not change over time nor does it vary with different amplitudes. Physical models often result in more complex and interesting sounds. We usually associate sounds with the method of their production—bubbling, scraping, smashing, rubbing etc. [33]. Sound created by physical modelling seem to have an underlying production method, while other methods of synthesis can sound disembodied.

- **Aid to composers**

  Not many composers have a symphony on call to test out their ideas. Physical modelling may make it possible for composers to hear an approximation of what a symphony, string quartet etc. will sound like.

- **Aid to instrument makers**

  Detailed physical models may allow instrument makers to experiment with instrument design on the computer —e.g. how would the sound of a guitar change if a different type of wood is used for the top plate. Software at this level of detail does not yet exist.

- **New Sounds**

  Instruments that would be very difficult to design or play (e.g. a ten foot long flute) could be simulated. Instruments that are impossible physically (e.g. a bell that changes shape) could also be simulated.

History of Physical Modelling in Sound Synthesis and Acoustics

Pythagoras was the first person to relate mathematics with music. He did experiments with the monochord (see figure 1.3) and found that when the string lengths were small integers (e.g. 1:1, 1:2 , 2:3) the sound was harmonious [4]. During the 18th and 19th centuries physicists and mathematicians such as Euler, d'Alembert, Helmholtz and Rayleigh developed mathematical models of vibration and musical instruments.

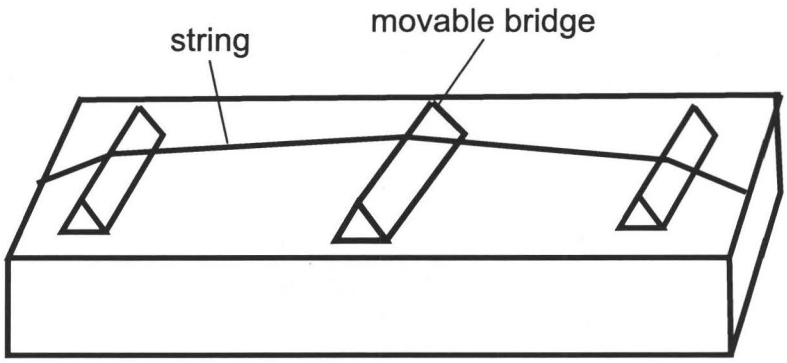

Figure 1.3: Monochord

In 1961 Kelly and Lochbaum [23] created a model of the human vocal tract and used an IBM 704 computer to synthesize the singing of "a bicycle built for two." In the 70's Hiller and Ruiz discretized the wave equation to create a computer model of a vibrating string [25]. In the late 70's and early 80's McIntyre, Woodhouse and Schumacher created the first computer models of musical instruments, such as the violin, clarinet and flute [28].

In the early 80's researchers at Association for the Creation and Research on Expression Tools (ACROE) created the CORDIS system to build virtual musical instruments using networks of masses and springs. They used specially built input devices and transducers to allow gestures by the user to interact with the instruments [8].

In the mid 80's Julius Smith developed the digital waveguide technique. The efficiency of this technique made real time simulations of physical models possible. In 1994 Yamaha used the digital waveguide technique to develop the first synthesizer based on physical modelling.

1.2.3 Methods Used in Physical Synthesis

There are many methods used to implement physical synthesis. *Finite-difference models* use numerical techniques to discretize the differential equations of the mathematical model of the instrument. For example, to simulate a vibrating string, the finite difference method would discretize the wave equation:

$$k\frac{\partial^2}{\partial x^2}y(t,x) = \epsilon\frac{\partial^2}{\partial t^2}y(t,x). \tag{1.1}$$

Here $y(t,x)$ represents the vertical position of the string at time t and horizontal position x, k is the string stiffness and ϵ the mass/length (linear density) of the string. The

resulting difference equation can be run on a digital computer, using parameters for the initial conditions and constants, and will calculate the position of the string at discrete points in time and space.

When a solid object is struck, deformations are caused which propagate through the object. The material and shape of the object determine the possible frequencies—or modes—that the object can vibrate at. Theoretically, there could be an infinite number of modes associated with a continuous structure [1]. In practice, a small number of modes can often be a good approximation for the vibration of an object. *Modal Synthesis* approximates the sound of an instrument by summing up a finite set of modes. Each mode is represented by a sine wave with exponential damping.

Instead of simulating the wave equation itself, as is done in finite difference modelling, *digital waveguides* simulate the solution of the wave equation. Any shape that travels to the left or right at speed $c = \sqrt{T/\epsilon}$, where T is the tension and ϵ the mass/length (linear density), is a solution to this equation. The left and right traveling waves can be simulated with 2 delay lines. We can then get the solution of the wave equation at any point in time by summing the left and right delay lines. The simulation is exact if the waves are bandlimited to half the sampling frequency and it is very efficient. These factors have made digital waveguides the most popular form of physical synthesis.

Source-filter models have been used extensively in speech synthesis and in analog music synthesizers. The source can generate periodic wave forms (i.e. sine waves, square waves etc.), random noise, or glottal noise (vibrations from vocal chords). The sound is then processed by the filter before being output. Both the source and the filter are controlled by time varying parameters.

Wave digital filters were developed by electrical engineers in the late 60's for digitizing analog circuits. Since the equations for inductors, capacitors and resistors have the same form as those of masses, springs and dampers, these filters can be used to model the vibrations of mechanical systems. Wave digital filters can be used to implement mass-spring systems. We explore this topic in section 4.2.

1.2.4 The Mass-spring Model

The *mass-spring* model builds complex musical instruments from simple components: masses, springs and dampers. Each element is discretized using finite difference methods. The behaviour of the system depends solely on the network and the physical equations of each of the componenets. No other physical equations are used.

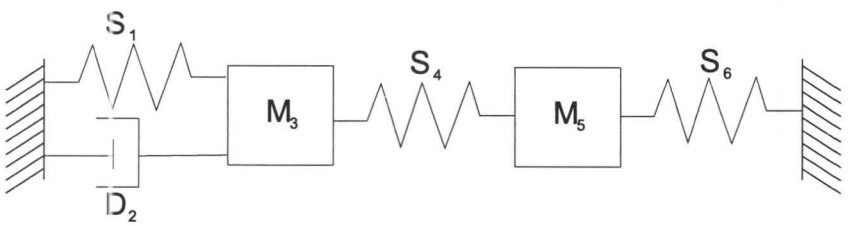

Figure 1.4: Simple mass-spring system

Figure 1.4 shows a simple mass-spring model, where M_3 and M_5 are masses, S_1, S_4 and S_6 are springs and D_2 is a damper.

Mass Spring Discretization

We use finite difference methods to discretize the mass-spring model.

Mass Element

We can derive the behaviour of a mass from Newton's 2nd law:

$$F(t) = ma(t), \tag{1.2}$$

where $F(t)$ is the force acting on the mass at time t, m is the mass, and $a(t)$ is the acceleration of the mass at time t. Since acceleration is the derivative of the velocity we can write equation (1.2) as

$$F(t) = mv'(t). \tag{1.3}$$

We then have a system of two first order differential equations:

$$\begin{pmatrix} x(t) \\ v(t) \end{pmatrix}' = \begin{pmatrix} v(t) \\ \frac{F(t)}{m} \end{pmatrix}. \tag{1.4}$$

We then use the backward Euler approximation to discretize the equations. The backward Euler approximation is:

$$\mathbf{x}_{n+1} \approx \mathbf{x}_n + h\mathbf{f}(\mathbf{x}_{n+1}), \tag{1.5}$$

where h is the length of the time step, $\mathbf{f}(\mathbf{x}_{n+1})$ is the value of the derivative at the $(n+1)$th time step—i.e.

$$\mathbf{f}(\mathbf{x}_{n+1}) \equiv \mathbf{f}(\mathbf{x}(h(n+1))),$$

and the vector $\mathbf{x}_{n+1}$ consists of the position and velocity at time step $n+1$. It is an *implicit* numerical method since $\mathbf{x}_{n+1}$ is on both the left and right sides of the equation. So equation (1.4) is approximated by

$$\begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ v_n \end{pmatrix} + h \begin{pmatrix} v_{n+1} \\ \frac{F_{n+1}}{m} \end{pmatrix}. \tag{1.6}$$

It is sometimes convenient to represent the mass element using a scalar instead of a vector equation. We can write the backwards Euler method as

$$x_{n+1} \approx x_n + hx'_{n+1},$$

where x'_n is the derivative of x with respect to time. The first derivative of the displacement with respect to time is the velocity so

$$v_n = x'_n = \frac{1}{h}(x_n - x_{n-1}).$$

Then the acceleration, which is the second derivative of x, is

$$a = v'_n = \left(\frac{1}{h}(x_n - x_{n-1})\right)' = \frac{1}{h^2}\left(x_n - 2x_{n-1} + x_{n-2}\right). \tag{1.7}$$

Substituting this into equation (1.2) gives us

$$F_n = m\left(\frac{1}{h^2}\left(x_n - 2x_{n-1} + x_{n-2}\right)\right)$$
$$x_n = \frac{h^2}{m}F_n + 2x_{n-1} - x_{n-2}.$$

Spring element The equations for the spring are derived using Hooke's Law $F(t) = -kx(t)$, where k is a constant denoting the spring stiffness. We write them as

$$F_{a:n+1} = k(x_{b:n+1} - x_{a:n+1}) \tag{1.8}$$
$$F_{b:n+1} = -F_{a:n+1}. \tag{1.9}$$

Here we let $x_{b:n}$, and $x_{a:n}$ represent the distance from the equilibrium position of mass M_a and mass M_b at either end of the spring. We use $F_a(n)$ to denote the force acting on mass M_a at one end of the spring at time step n. The force, $F_{b:n}$, acting on mass M_b at the other end of the spring is, according to Newton's third law, equal and opposite to $F_a(n)$.

Note that the equations for the mass and the spring are interdependent: in order to calculate the position of the mass at time $n+1$ we need to know the force of the spring at time $n+1$, but to find the force of the spring at time $n+1$ we need to know the position of the mass at time $n+1$. These are known as *delay-free loops* and they result in systems that are not directly computable, although they can be solved by using a system of simultaneous equations. They are equivalent to implicit difference equations. We will see how this problem is dealt with in section 3.1.

Damper element

The damper element is used to represent viscous friction. This is the object's resistance to motion and is assumed to be proportional to the velocity. The formula for the damper is $F(t) = -Zv_r(t)$, where Z is a constant denoting the coefficient of viscosity, $F(t)$ the force and $v_r(t)$ the relative velocity of the two ends of the damper. This can be written as

$$F_{a:n+1} = Z(v_{b:n+1} - v_{a:n+1})$$
$$F_{b:n+1} = -F_{a:n+1},$$

where F_a and F_b represent the forces acting on the masses at the ends of the damper, and v_a and v_b are the corresponding velocities.

1.2.5 Motivation for Mass Spring Networks

1. **Simplicity, Uniformity, Generality**
   Mass-spring networks represent a method of creating complexity from simple building blocks. While each of the components, the mass, the spring and the damper, are easy to understand, the networks built from these components can become complex. Mass spring networks have been traditionally used in physics and engineering to model vibration and can be used to model a large family of vibration patterns.

2. **Compatibility with Computer Graphics**
   The mass-spring model is also extensively used in computer graphics. This suggests that the same model could be used for both sound and graphics in virtual reality type models such as the CORDIS-ANIMA system described in the next section. One obstacle for this approach is that the time scales for computer graphics and sound synthesis are different. Computer graphics are usually displayed at 50 to 75 Hz, whereas sound synthesis may require sample rates of 40,000 samples per second or more.

3. **Parallelizable**
   Another advantage of mass-spring networks is that they can be computed in parallel. Georgii and Westermann [18] show how the Graphics Processing Unit (GPU) can be used to run a mass-spring computer graphics model in parallel.

Chapter 2

Background

In this chapter, we present some basic numerical methods used in approximating differential equations and show the results of these methods when applied to simple mass-spring systems. In particular, we are interested in 3 side-effects of numerical methods: instability, frequency warping and numerical damping. To analyze these side-effects we introduce the z-transform. We describe the concept of a numeric method as a mapping from the s-plane to the z-plane. We then describe two previous implementations of mass-spring systems in sound synthesis: CORDIS-ANIMA and TAO.

2.1 Numerical Methods

The equations of motion describing how objects react to forces are usually expressed as differential equations. There are many ways of implementing these equations on a digital computer and most of them have side effects. In this section we examine some of the commonly used methods and what their side effects are.

2.1.1 First Order Differential Equations

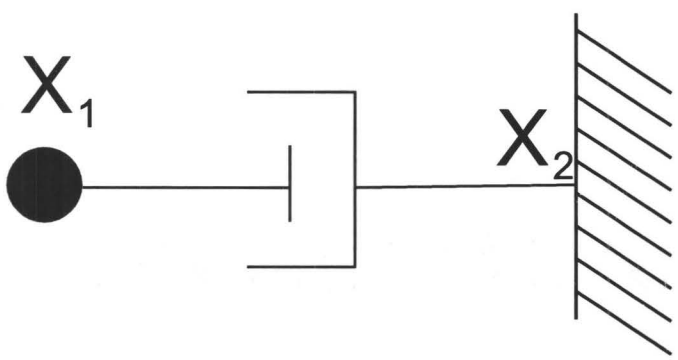

Figure 2.1: Mass damper system

The equation of two point masses, one of which, x_2, is fixed, connected by a damper (figure 2.1) is

$$-F(t) = Zv(t),$$

where F is force exerted by the damper, v is the relative velocity of the ends of the damper and Z is the damping coefficient. From Newton's 2nd law we have

$$ma(t) = -Zv(t).$$

Since the acceleration, a, is the derivative of the velocity

$$m\frac{dv(t)}{dt} = -Zv(t) \quad \text{or} \quad \frac{dv(t)}{dt} = -\frac{Z}{m}v(t) \tag{2.1}$$

This is an example of a first order differential equation, since the highest order derivative is one.

The solution of this equation is

$$v(t) = Ce^{-(Z/m)t},$$

where C is a constant depending on the initial conditions. If $t = 0$ then $v(t) = C$ so C is the velocity at time 0. For each different value of C we get a different solution. These solutions are called integral curves. Some integral curves with $Z/m = 10$ are shown in figure 2.2.

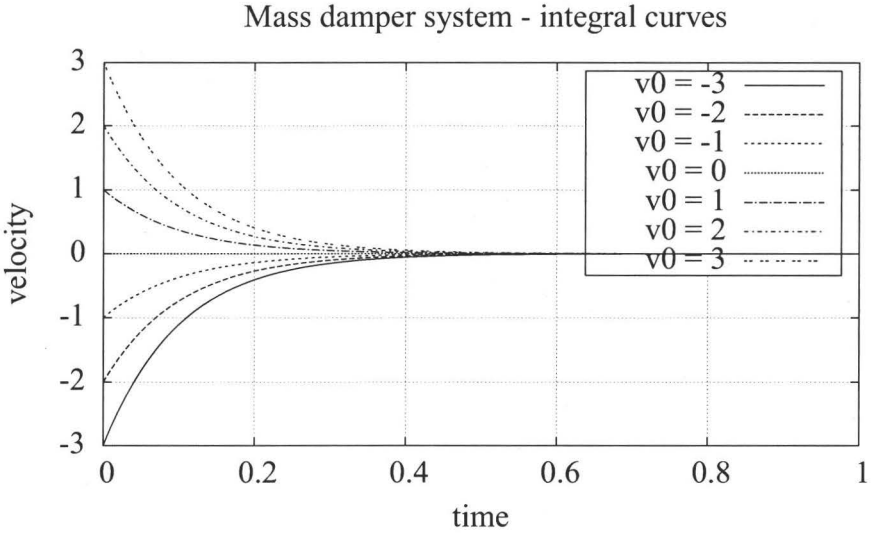

Figure 2.2: Some of integral curves for mass-damper system

The forward Euler Method

The forward Euler method is defined as

$$y_{n+1} \approx y_n + y'_n h \tag{2.2}$$

where y' is the first derivative of y with respect to time and h is the length of each time step. Then the forward Euler approximation of the mass damper system using the derivative from equation (2.1) is:

$$v_{n+1} \approx v_n - (Z/m)v_n h = v_n(1 - (Z/m)h) \tag{2.3}$$

The *local truncation error* is the error caused by the numerical approximation for one time step. It can be shown [7] that, for the forward Euler method, the local truncation error is proportional to h^2. The *global truncation error* is the cumulative error for all time steps and for the forward Euler this error is proportional to h. Because of this, the forward Euler method is called a *first order method.* Figure 2.3 shows the forward Euler approximation of the mass-damper system using a small time step. The Euler method tracks the actual solution very closely.

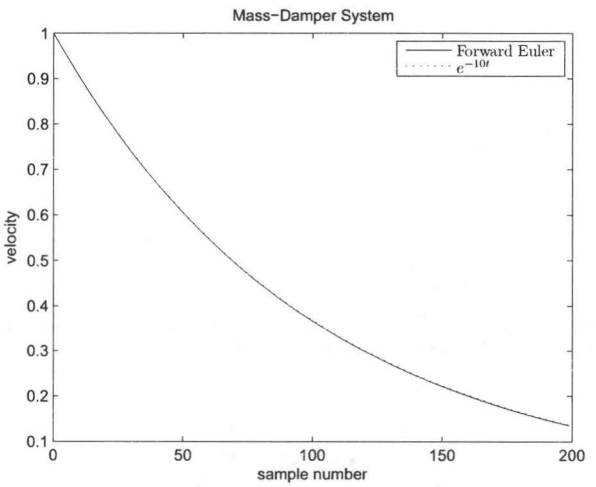

Figure 2.3: Forward Euler solution with small time step ($h = .001$)

Figure 2.4 shows the same system with a larger time step. Since the slope of $v = e^{-10t}$ is constantly decreasing, the forward Euler overestimates the slope. We can see in this diagram that the numerical approximation does not stay on the same integral curve but jumps to other curves with different initial values. Notice that the slope at the start of each line segment is the same as the integral curve at that point.

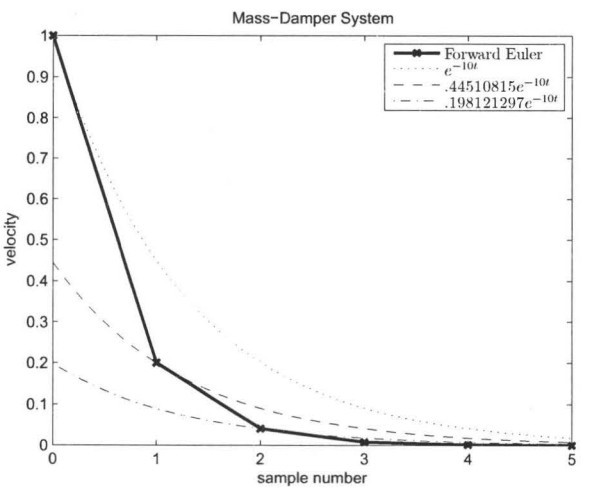

Figure 2.4: Forward Euler solution with larger time step ($h = .08$)

Stability

Differential equations themselves can be unstable. This means that arbitrarily small changes in the initial conditions can cause arbitrarily large changes in the solution as $t \to \infty$. This is shown in figure 2.5. This is also called an *ill-conditioned problem* and it is extremely difficult to model these systems numerically [32].

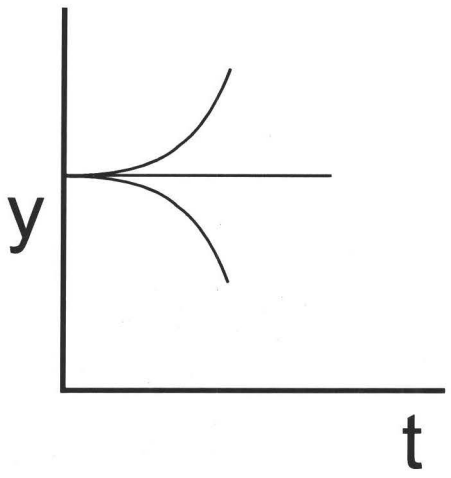

Figure 2.5: Unstable system

Going back to our mass-damper system, we can see from figure 2.2 that it is not an unstable system. In fact, it is *asymptotically stable* since all solutions approach 0 as $t \to \infty$. Numerical methods can also cause instability. Instead of approaching a stable equilibrium point, the computed solution may become arbitrarily large.

The derivative of this equation was $-(Z/M)v$. Using the forward Euler method on this system with $Z/m = 10$ gives us:

$$\begin{aligned} v_{n+1} &\approx v_n + v_n' h = v_n - 10 v_n h \\ &= v_n(1 - 10h) \end{aligned} \tag{2.4}$$

This is a first order difference equation of the form $y_n - Ay_{n-1} = 0$. The solution of this equation is $y_n = y_0 A^n$ [5]. So the solution of equation (2.4) is

$$v_n = v_0(1 - 10h)^n.$$

From this solution we can see some problems that might occur. If $0 > (1 - 10h) > -1$ (i.e. $0.2 > h > 0.1$) , the values of the velocity will oscillate between positive and negative values. Note that in the exact solution there is no oscillation, so the oscillation is produced by the numerical method. If $(1 - 10h) < -1$ (i.e. $h > 0.2$) the oscillations will increase without bound (see figure 2.6). In this case the numerical method has caused a stable system to become unstable.

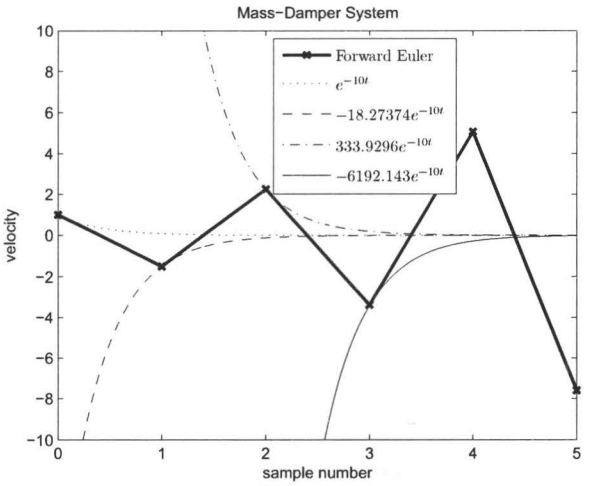

Figure 2.6: Forward Euler becomes unstable ($h = .25$)

Backward Euler

The backward Euler method is defined as

$$y_{n+1} \approx y_n + y'_{n+1}h \tag{2.5}$$

The backward Euler is called an *implicit* method since it uses the derivative at the new point which has not yet been determined. In some cases it may be difficult solve this equation. For the mass-damper system, however, the solution is simple. Again using $V/m = 10$:

$$v_{n+1} \approx v_n + v'_{n+1}h = v_n - 10v_{n+1}h$$
$$v_{n+1} = \frac{1}{1+10h}v_n$$

The solution to this difference equation is $v_n = v_0(\frac{1}{1+10h})^n$. Notice that unlike the forward Euler, this equation will never oscillate or become unstable no matter what the value of h is (since h is the time step it is assumed to always be > 0).

Figure 2.7 shows the backward Euler with a large time step. Notice the the backward Euler underestimates the slope of this curve and that the slope of the line segment matches the integral curve at the end of each segment rather than the start like the forward Euler.

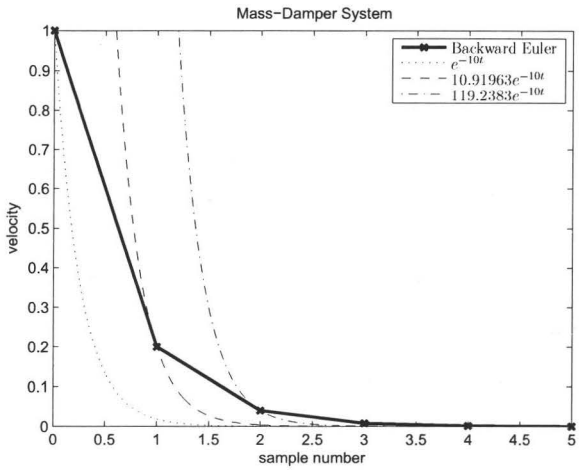

Figure 2.7: Backward Euler with large time step ($h = .4$)

To gain more insight into the stability of numerical methods we introduce the topic of digital filters.

Digital Filters

A *discrete signal*, $x(n)$ is a sequence of *quantized* real numbers indexed by $n \in \mathbb{N}$. *Quantization* is the approximation (rounding) required to represent the number in a finite number of bits in a digital computer. A digital filter is a Linear Time Invariant (LTI) system that transforms an input signal, $x(n)$, into an output signal, $y(n)$ (see figure 2.8). A system is linear iff:

1.

$$\begin{aligned}&\text{for signals } x_1, x_2, y_1, y_2\\ &\text{if } S(x_1(n)) = y_1(n) \text{ and } S(x_2(n)) = y_2(n)\\ &\text{then } S(x_1(n) + x_2(n)) = y_1(n) + y_2(n),\end{aligned}$$

   i.e. the sum of the inputs equals the sum of the outputs. S is the function performed by the LTI system.

2.

$$\begin{aligned}&\text{if } S(x_1(n)) = y_1(n)\\ &\text{then } S(cx_1(n)) = cy_1(n),\end{aligned}$$

   i.e. scaling the input results in scaling the output by the same amount.

A time invariant system is a system whose response does not change over time.

$$\begin{aligned}&\text{if } S(x(n)) = y(n)\\ &\text{then } S(x(n+\delta)) = y(n+\delta),\end{aligned}$$

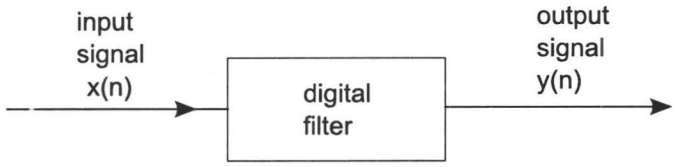

Figure 2.8: A digital filter

i.e. a time shift in the input results in a time shift in the output.

The *impulse response* of a digital filter is its response to an impulse signal, $\delta(n)$, called the Kronecker delta.

$$\delta(n) = \begin{cases} 1 & \text{if } n = 0, \\ 0 & \text{if } n \neq 0. \end{cases}$$

The impulse response is then $h(n)$ where

$$S(\delta(n)) = h(n). \tag{2.6}$$

We can regard the input signal $x(n)$ as the sum of weighted impulses [26]:

$$\forall n \in \mathbb{N}, x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k), \tag{2.7}$$

since $\delta(n-k) = 1$ only when $k = n$, at which point $x(k)\delta(0) = x(n)$. We can then compute the output in terms of the impulse response:

$$\begin{aligned} y(n) = S(x(n)) &= S\left(\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)\right) \quad \text{substitute (2.7)} \\ &= \sum_{k=-\infty}^{\infty} x(k)(S\delta(n-k)) \quad \text{since S is linear} \\ &= \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad \text{by (2.6),} \end{aligned}$$

where $h(n)$ is the impulse response of S.

We can also analyze the filter in the frequency domain by performing the Laplace transform on the input and output. Letting $\hat{X} = \mathcal{L}[x]$ and $\hat{Y} = \mathcal{L}[y]$, we define $\hat{H} = \hat{Y}/\hat{X}$. $\hat{H}$ is called the *transfer function* of the filter. It can be shown [26] that:

$$\mathcal{L}^{-1}[\hat{H}(s)] = h(t),$$

i.e. the inverse Laplace transform of the transfer function is the impulse response in the time domain.

S to z-plane mapping

We can gain insight into the stability of numerical methods by analyzing the system on the s-plane and the z-plane. We map an equation to s-plane by taking its Laplace transform:

$$\mathcal{L}[y(t)] = \int_0^{\infty} e^{st} y(t)dt,$$

where $s = \sigma + j\omega$. We can write e^{-st} as $e^{-\sigma t}e^{j\omega t}$, where $e^{-\sigma t}$ is an exponential damping term and $e^{-j\omega t}$ is a phasor rotating at frequency ω. The horizontal axis on the s-plane is the real part of s represented by σ, and the vertical axis is the imaginary part represented by ω (see figure 2.9) .

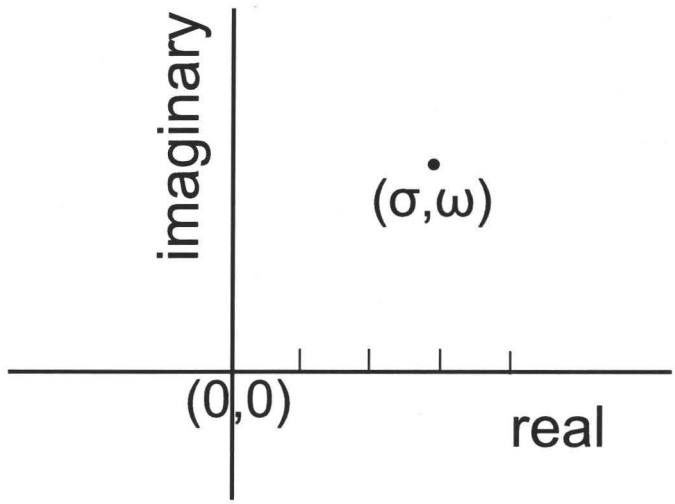

Figure 2.9: The s-plane

We can write the numerator and denominator of the transfer function as polynomials in s:

$$\hat{H}(s) = \frac{(s - q_1)\dots(s - q_M)}{(s - p_1)\dots(s - p_N)}.$$

Since $\hat{Y} = \hat{X}\hat{H}$, whenever $s = q_m$ the output is zero. This is why the values of q_m are called the *zeros* of the filter. Similarly, when $s = p_n$, the output is infinite. The values of p_n are called the poles of the filter. It can be shown [26] that a filter is stable iff all the poles are on the left side of the s-plane—i.e. $\sigma < 0$. A filter is stable if, when the input is bounded, the output is bounded.

The z-transform is defined:

$$\hat{X}(z) = \sum_{m=-\infty}^{\infty} x(m)z^{-m},$$

where $z = re^{j\omega}$. The z-plane uses polar coordinates with $|z| = r$ the distance from the origin and ω as the angle (see figure 2.9).

Analogously to the s domain, the transfer function in the z domain is:

$$\hat{H}(z) = \frac{\hat{Y}(z)}{\hat{X}(z)}.$$

Writing the numerator and denominators of $\hat{H}(z)$ as polynomials in z gives us the poles and zeros of the transfer function. In the z-plane, however, the filter is stable when all the poles are within the unit circle.

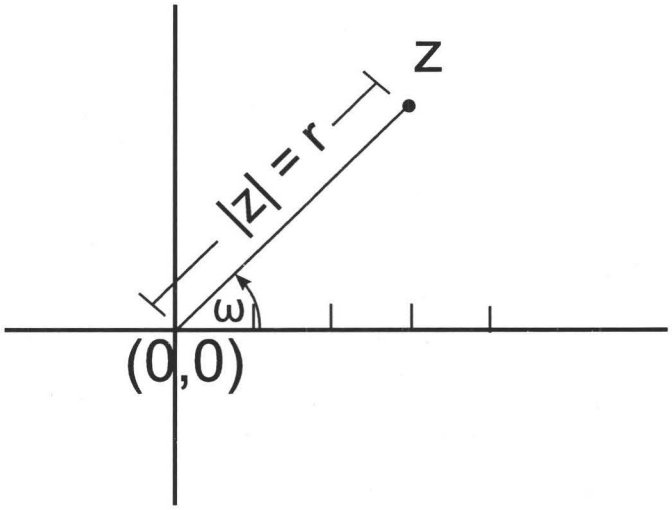

Figure 2.10: The z-plane

An important feature of the z-transform is that multiplying the z-transform of a signal by z^{-1} is the same as delaying the signal by one sample [39].

$$z^{-1}\hat{X}(z) = z^{-1}\sum_{m=-\infty}^{\infty} x(m)z^{-m} = \sum_{m=-\infty}^{\infty} x(m)z^{-(m+1)} = \sum_{k=-\infty}^{\infty} x(k-1)z^{-k},$$

which is the z-transform of signal x delayed by one sample. Similarly, multiplying by z is equivalent to advancing the signal by 1 time step [39].

Mapping the forward Euler from the s to the z-plane

We start by defining an *ideal differentiator* [24] as a system that takes a signal x as an input and outputs its derivative dx/dt. We then take the Laplace transform of both sides and find the transfer function.

$$\begin{aligned} y(t) &= dx(t)/dt \\ \hat{Y}(s) &= s\hat{X}(s) \\ \hat{H}(s) &= \frac{\hat{Y}(s)}{\hat{X}(s)} = s. \end{aligned}$$

Next we find the derivative approximation using the forward Euler method. From equation (2.2), the approximation of the derivative using the forward Euler is

$$\left.\frac{dx}{dt}\right|_n \approx \frac{1}{h}(x_{n+1} - x_n). \tag{2.8}$$

We now take its z-transform and find its transfer function:

$$\begin{aligned} \mathcal{Z}[dx/dt] &= \mathcal{Z}[(1/h)(x_{n+1} - x_n)] \\ \hat{Y}(z) &= (1/h)(\hat{X}(z)z - \hat{X}(z)) \\ \hat{H}(z) &= \frac{\hat{Y}(z)}{\hat{X}(z)} = \frac{z-1}{h}. \end{aligned}$$

We can now map the transfer function from the s-plane to the z-plane using $s \rightarrow \frac{z-1}{h}$. The reverse mapping is then $z \rightarrow 1 + hs$.

We can check to see if, when a system is stable in the s-plane, its transform is also stable in the z-plane; this is equivalent to the forward Euler approximation of a differential equation being stable (absolutely summable) whenever the differential equation is stable (absolutely integrable). The system is stable on the s-plane when all its poles have $\sigma < 0$ — i.e. all poles are to the left of the $j\omega$ axis. Using the reverse mapping we find

$$\text{if } s = j\omega \text{ then } z \rightarrow 1 + h(j\omega) = 1 + (h\omega)j.$$

This is a vertical line going through the point $(1, 0)$. We can see from this that a stable system's poles can be mapped anywhere to the left of this line, and in many cases this will not be inside the unit circle (see figure 2.11). This confirms that the forward Euler may create instability in a stable system.

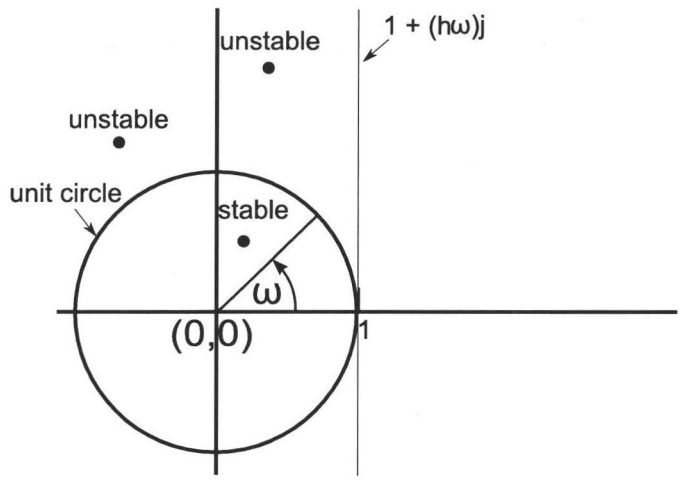

Figure 2.11: Forward Euler on the z-plane

We can analyze the mass-damper system using this transform. We add $F_{ext}(t)$ as the input, which is an external force acting on the mass. F_d is the force produced by the damper. (see figure 2.12).

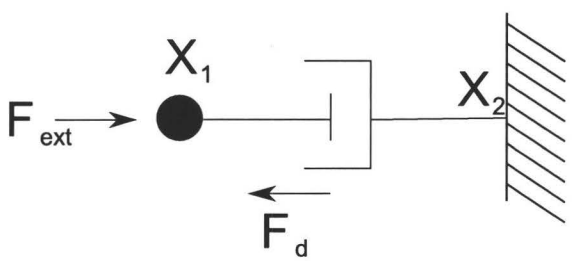

Figure 2.12: Mass damper system with external force

Looking at the forces acting on the mass we see $F_{ext}(t) - F_d(t) = ma$. So:

$$dv/dt = 1/m(F_{ext}(t) - F_d(t)) = 1/m(F_{ext}(t) - Zv(t)).$$

We then take the Laplace transform of both sides:

$$s\hat{V}(s) = (1/m)\hat{F}_{ext}(s) - (Z/m)\hat{V}(s)$$
$$\hat{V}(s)(s + Z/m) = (1/m)\hat{F}_{ext}(s).$$

Since F_{ext} is the input and v the output, the transfer function is:

$$\hat{H}(s) = \frac{1}{m(s + Z/m)} = \frac{1}{ms + Z}$$

The transfer function has one pole at $ms + Z = 0$ or $s = -Z/m$. Since Z and m are always positive, this pole is to the left of the imaginary axis. Therefore the system is always stable.

Now using the the mapping $s \rightarrow \frac{z-1}{h}$ we can map the pole to the z-plane.

$$s = -Z/m \rightarrow \frac{z-1}{h} = -Z/m$$
$$z = \frac{-Zh}{m} + 1.$$

So on the z-plane the system is stable when:

$$\left|\frac{-Zh}{m} + 1\right| < 1$$
$$\text{or } \frac{-Zh}{m} + 1 > -1 \text{ which is true if } h > 2m/Z.$$

If, as in section 2.1.1, we use $Z/m = 10$, we get $h > 0.2$ as the condition for stability. Note that this is exactly the same result that we obtained in section 2.1.1 by solving the difference equation.

Mapping the backward Euler from the s to the z-plane

We can use the same method with the backward Euler method. This results in $s \rightarrow \frac{1-z^{-1}}{h}$ and the reverse mapping is $z \rightarrow \frac{1}{1-hs}$

Now we can check to see if when a system is stable in the s-plane its transform is also stable in the z-plane. The system is stable when $\sigma < 0$ (i.e. all points to the left of the $j\omega$ axis)[16].

$$\frac{1 - z^{-1}}{h} = \frac{z - 1}{hz} = j\omega$$
$$z = \frac{1}{1 - j\omega h} = 1/2 + \frac{2 - (1 - j\omega h)}{2(1 - j\omega h)} = 1/2 + \frac{1}{2}\frac{(1 + j\omega h)}{(1 - j\omega h)}$$
$$|z - 1/2| = \frac{1}{2} \quad (\text{since } |1 + j\omega h| = |1 - j\omega h|).$$

This is the equation of a circle centered at $(1/2, 0)$ with $radius = 1/2$ (see figure 2.13). All poles on the left side of the s-plane map to inside the circle $|z - 1/2| = \frac{1}{2}$ which is inside the unit circle. Therefore, any stable system will still be stable using the backward Euler method no matter what the time step.

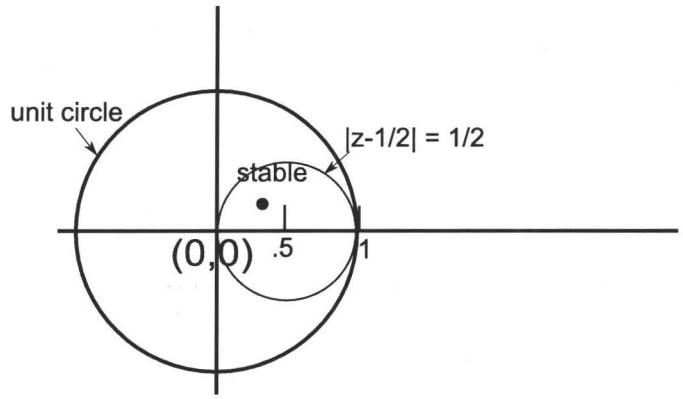

Figure 2.13: Backward Euler on the z-plane

2.1.2 Second Order Differential Equations

Figure 2.14 shows a simple mass spring system.

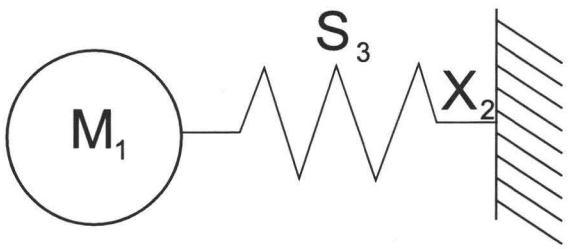

Figure 2.14: Mass spring system

If we regard the equilibrium position of the spring to be $x = 0$, the force of the spring is $F(t) = -kx(t)$.

$$ma(t) = -kx \quad \text{(by Newton's 2nd law)} \tag{2.9}$$

$$d^2x(t)/dt^2 = -(k/m)x(t). \tag{2.10}$$

This is a second order differential equation. The general solution is [7]

$$x(t) = C_1 \cos(\sqrt{k/m}t) + C_2 \sin(\sqrt{k/m}t).$$

Setting initial condition $x(0) = 1$ (the initial position) and $x'(0) = 0$ (the initial velocity)

$$\begin{aligned} x(0) &= C_1 \\ x'(t) &= -\sqrt{k/m}C_1 \sin(\sqrt{k/m}t) + \sqrt{k/m}C_2 \cos(\sqrt{k/m}t) \\ x'(0) &= \sqrt{k/m}C_2 = 0 \; ; C_2 = 0. \end{aligned}$$

So the particular solution is: $x(t) = \cos(\sqrt{k/m}t)$. Writing the frequency $\sqrt{k/m}$ as ω we get:

$$x(t) = \cos(\omega t)$$

The Forward Euler solution of the mass spring system

From the definition of the forward Euler, $x_{n+1} = x_n + x'_n h$, we find that the second derivative can be approximated as:

$$x''_n = \frac{x_{n+2} - 2x_{n+1} + x_n}{h^2}.$$

We can then use the forward Euler to approximate equation (2.10).

$$\frac{x_{n+2} - 2x_{n+1} + x_n}{h^2} = -(k/m)x_n.$$

If we assume the system is time invariant we can shift the indices back by one time step:

$$\frac{x_{n+1} - 2x_n + x_{n-1}}{h^2} = -(k/m)x_{n-1}$$

$$x_{n+1} = 2x_n - \left(\frac{kh^2}{m} + 1\right) x_{n-1}.$$

The frequency, ω_0, is equal to $\sqrt{k/m}$ so:

$$x_{n+1} = 2x_n - \left(\omega_0^2 h^2 + 1\right) x_{n-1}$$

The results of running the forward Euler with $\omega_0 = 125 \times 2\pi$ as the frequency and $h = .001$ as the sample time are shown in figure 2.15.

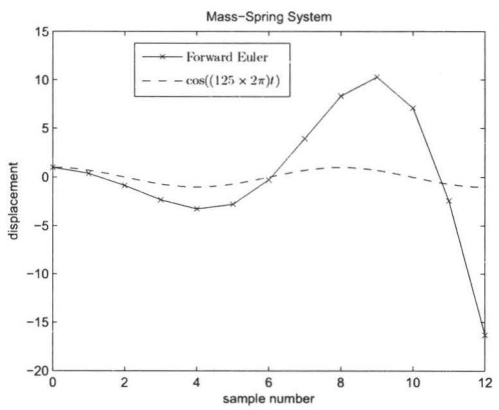

Figure 2.15: Forward Euler approximation of mass spring system ($h = .001$)

We can see that this system is unstable. We can analyze this on the s and z-planes. We first add a forcing input function to equation (2.9) to get $F_{ext} - F_s = ma$. This results in

$$d^2x(t)/dt^2 = (1/m)F_{ext} - (k/m)x(t)$$

Taking the Laplace transform of both sides of the equation yields

$$\hat{X}(s)(s^2 + (k/m)) = \hat{F}_{ext}(1/m).$$

Here, the input is F_{ext} and the output is the position x. So the transfer function is

$$\hat{H}(s) = \frac{1}{m}\frac{1}{s^2 + (k/m)} = \frac{1}{m}\frac{1}{s^2 + \omega_0^2} \tag{2.11}$$

The poles of the system are where $s^2 + \omega_0^2 = 0$, which are $s = \pm j\omega$. These occur on the $j\omega$ axis. The system is stable, but not asymptotically stable, since it does not decay to zero.

Next, we map this to the z-plane with the forward Euler mapping $s \rightarrow (z-1)/h$.

$$s = \pm j\omega_0 \rightarrow \frac{z-1}{h} = \pm j\omega_0$$
$$z = 1 \pm (h\omega_0)j$$

These poles lie on the line $z = 1 + (h\omega)j$ as shown in figure 2.16. We can see that this is unstable for any sample time or frequency—unless the frequency is 0.

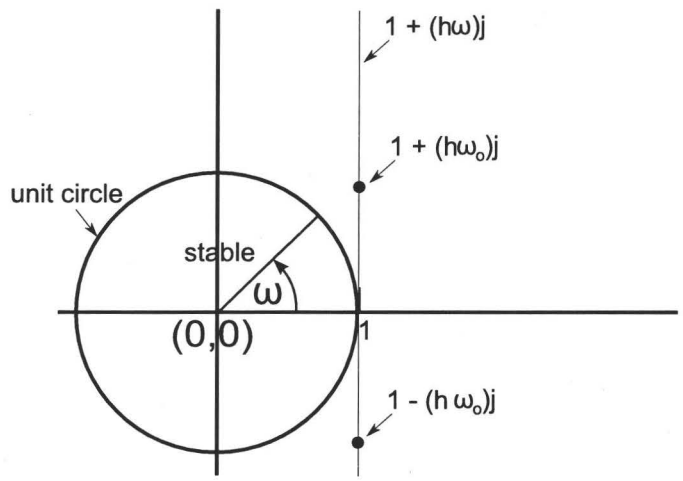

Figure 2.16: Poles on the z-plane of mass-spring system using forward Euler

Due to its problems with stability, we can conclude that the forward Euler is unsuitable for sound synthesis using mass spring networks.

The Backward Euler solution of the mass spring system

We next examine the same system using the backward Euler method. The second derivative using the backward Euler is:

$$x''(n) = \frac{1}{h^2}\left(x_n - 2x_{n-1} + x_{n-2}\right).$$

Now we use the backward Euler on the mass spring equation (2.10):

$$\frac{x_n - 2x_{n-1} + x_{n-2}}{h^2} = -(k/m)x_n$$
$$x_n = \frac{1}{1 + (h\omega)^2}(2x_{n-1} - x_{n-2}).$$

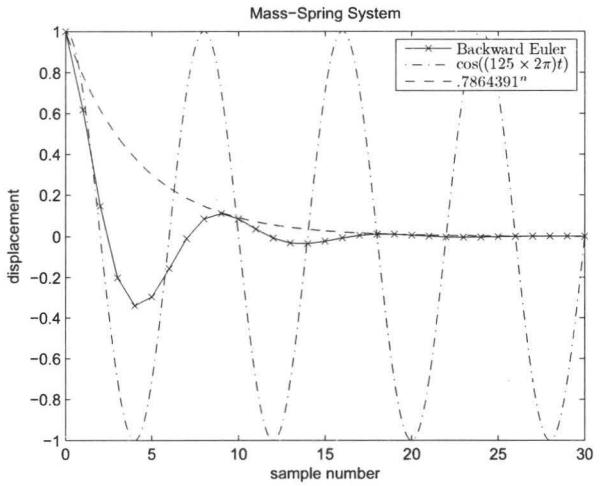

Figure 2.17: Backward Euler approximation of mass spring system

The results are shown in figure 2.17.

From the graph in figure 2.17 we see that the backward Euler has the opposite problem of the forward Euler: instead of exponential growth, we have exponential damping. We can again use the s to z-plane mapping to analyze this problem. We start with the transfer function from equation 2.11 and use the backward Euler mapping $s \rightarrow 1/(1 - hz)$.

$$\hat{H}(s) = \frac{1}{m}\frac{1}{s^2 + \omega_0^2} \rightarrow \frac{1}{m}\frac{1}{\left(\frac{z-1}{hz}\right)^2 + \omega^2}$$

$$\hat{H}(z) = \frac{1}{m}\frac{(hz)^2}{z^2(1 + (h\omega)^2) - 2z + 1}.$$

The poles are at $z^2(1 + (h\omega)^2) - 2z + 1 = 0$. Solving this quadratic equation gives us:

$$z = \frac{1 \pm j(h\omega)}{1 + (h\omega)^2}.$$

For this example, the frequency, f, is $1/8$ the sampling frequency, so

$$f = (1/8)f_s; \quad \omega/(2\pi) = 1/(8h)$$

$$h\omega = \frac{\pi}{4}; \quad (h\omega)^2 = \frac{\pi^2}{16}.$$

Now we can split z into its real and imaginary parts:

$$Re\left[\frac{1 \pm jh\omega}{1 + (h\omega)^2}\right] = \frac{1}{1 + (h\omega)^2} = \frac{1}{1 + \frac{\pi^2}{16}} \approx .618486458$$

$$Im\left[\frac{1 \pm jh\omega}{1 + (h\omega)^2}\right] = \frac{\pm h\omega}{1 + (h\omega)^2} = \frac{\frac{\pi}{4}}{1 + \frac{\pi^2}{16}} \approx \pm .485758128.$$

The results are shown in figure 2.18. As expected the poles lie on the smaller circle. But note that they do not lie on the unit circle. The closer they are to the origin the larger the damping.

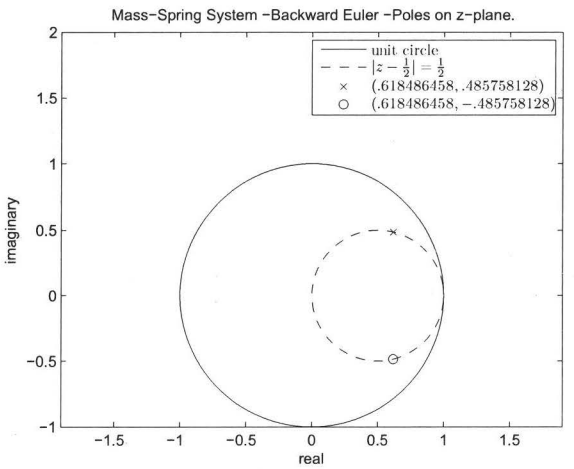

Figure 2.18: Backward Euler approximation on the z-plane

We can use the real and imaginary parts to find the exact damping.

$$|z| = r = \sqrt{Re^2 + Im^2} = \sqrt{.618486458^2 + .485758128^2} = .7864391.$$

This means the backward Euler approximation for this example is decaying at the rate $.7864391^n$, where n is the sample number. This is shown in figure 2.17.

We can solve the backward Euler mapping to the z-plane for r and ω in the general case for the undamped mass spring system.

$$\begin{aligned} r &= \sqrt{\left(\frac{1}{1+(h\omega)^2}\right)^2 + \left(\frac{h\omega}{1+(h\omega)^2}\right)^2} \\ &= \frac{\sqrt{1+(hw)^2}}{1+(hw)^2} = \frac{1}{\sqrt{1+(hw)^2}}. \end{aligned}$$

We see from this that r approaches 1 (no damping) when either the sample time, h, or the frequency, ω, approach zero. The extraneous damping increases with the product of the sample time and the frequency.

$$\omega_d = \tan^{-1}\left(\frac{\frac{h\omega}{1+(hw)^2}}{\frac{1}{1+(hw)^2}}\right) = \tan^{-1}(h\omega). \tag{2.12}$$

Here ω_d represents the digital frequency—the actual frequency produced by the numerical method—and ω is the original frequency. Figure 2.19 shows the arctan function. Note that the entire $j\omega$ axis on the s-plane is mapped to the range $[-\pi/2, \pi/2]$. The part of the graph near the origin—where low frequency or small sample times are mapped—is nearly linear. High frequencies and large sample times are progressively "warped". This is called *frequency warping*. The "x" on this graph shows our

example with $f = (1/8)f_s$. Here the frequency is only slightly warped. The "o" is at $f = (1/2)f_s$, called the Nyquist frequency. At this point there is considerable frequency warping.

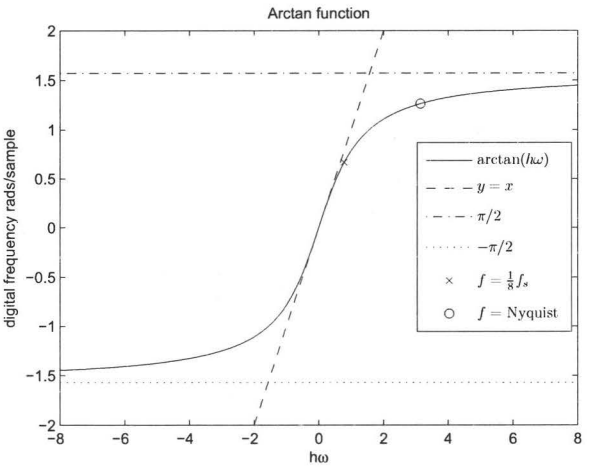

Figure 2.19: Backward Euler -frequency warping

Equation (2.12) shows that for our example the frequency of the system is 105.96 Hz instead of 125 Hz. This can be seen in figure 2.17 where the backward Euler lags somewhat behind $cos(125 \times (2\pi)t)$.

The backward Euler, although superior to the forward Euler, still is not ideal especially at high frequencies.

The Bilinear Transform solution of the mass spring system

The bilinear transform is defined $s \leftarrow \frac{2}{h}\frac{z-1}{z+1}$[16]. This mapping can be used on our undamped mass spring system's transfer function using equation 2.11.

$$\hat{H}(s) = \frac{1}{m}\frac{1}{s^2 + \omega_0^2} = \frac{1}{s^2 m + k}$$

$$\frac{1}{s^2 m + k} \rightarrow \frac{1}{\left(\frac{2}{h}\frac{1+z^{-1}}{1-z^{-1}}\right)^2 m + k}.$$

This simplifies to

$$\hat{H}(s) = \frac{\hat{X}(s)}{\hat{F}(s)} = \frac{\frac{1}{4m+kh^2}\left(h^2 + 2h^2 z^{-1} + h^2 z^{-2}\right)}{1 + \frac{2kh^2 - 8m}{4m+kh^2} z^{-1} + z^{-2}}$$

We can now solve for $\hat{X}(s)$ and calculate the inverse z-transform, resulting in

$$\begin{aligned} x_n &= \frac{h^2}{4m + kh^2}F_n + \frac{2h^2}{4m + kh^2}F_{n-1} + \frac{h^2}{4m + kh^2}F_{n-2} \\ &\quad - \frac{2kh^2 - 8m}{4m + kh^2}x_{n-1} - x_{n-2}. \end{aligned}$$

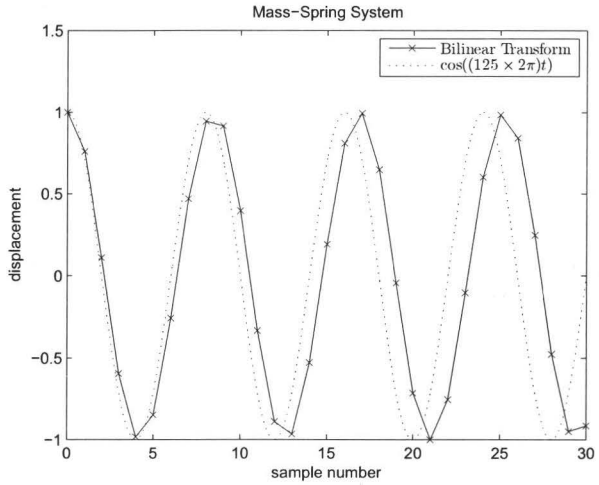

Figure 2.20: Bilinear transform of undamped mass spring system at f = 1/8 sample rate

The results of running this equation with the external force F_n set to 0 are shown in figure 2.20.

We see from figure 2.20 that the bilinear transform, at least in this case, is much more accurate than the backward Euler. There appears to be no numerical damping. As well, although there seems to be some frequency warping, it appears to be much less than the backward Euler. To confirm these observations, let us analyse how the bilinear transform maps $j\omega$ axis on the s to the z-plane.

The inverse mapping for the bilinear transform is

$$z \rightarrow \frac{1 + hs/2}{1 - hs/2}.$$

Now we find the value of z when $s = j\omega$.

$$\text{if } s = j\omega \text{ then } z = \frac{1 + hj\omega/2}{1 - hj\omega/2}$$

$$|z| = \frac{|1 + hj\omega/2|}{|1 - hj\omega/2|} = 1.$$

$|z| = 1$ is the equation a unit circle centered at the origin. This means that poles to the left of the $j\omega$ axis map to inside the unit circle. So any stable system will still be stable after the bilinear transform. Further, since the $j\omega$ axis maps to the unit circle, there is no numerical damping introduced to undamped systems.

Next we look at how the frequencies on the $j\omega$ axis map to the z-plane. We denote the analog frequency on the s-plane as ω_a and the discrete frequency on the z-plane as

ω_d [27]. We let $z = e^{j\omega_d}$.

$$
\begin{aligned}
s &= \frac{2}{h}\frac{z-1}{z+1} = \frac{2}{h}\frac{1-z^{-1}}{1+z^{-1}} = \frac{2}{h}\left(\frac{1-e^{-j\omega_d}}{1+e^{-j\omega_d}}\right) \\
&= \frac{2}{h}\left(\frac{e^{-j\omega_d/2}(e^{j\omega_d/2}-e^{-j\omega_d/2})}{e^{-j\omega_d/2}(e^{j\omega_d/2}+e^{-j\omega_d/2})}\right) \\
&= \frac{2}{h}\frac{e^{-j\omega_d/2}[2j\sin(\omega_d/2)]}{e^{-j\omega_d/2}[2j\cos(\omega_d/2)]} \\
s &= j\omega_a = \frac{j2}{h}\tan(\omega_d/2) \\
\omega_d &= 2\tan^{-1}\left(\frac{\omega_a h}{2}\right).
\end{aligned}
$$

This function is shown in figure 2.21.

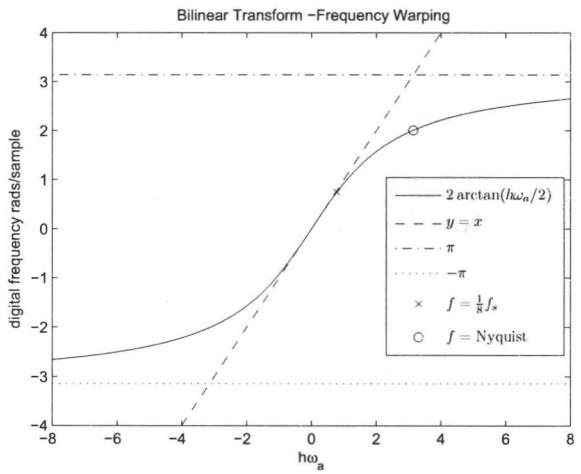

Figure 2.21: Frequency warping by the bilinear transform

Like the frequency mapping of the backward Euler shown in figure 2.19, the bilinear transform maps the entire $j\omega$ axis to a circle—in this case the unit circle. There are, however, some differences. Comparing this function to the frequency warping of the backward Euler, we see that the linear portion is much longer for the bilinear transform since the discrete frequency ranges from $-\pi$ to π radians. For our example the frequency, which should be 125 cycles per second, is actually 119.11 cycles per second. This compares favourably with the backward Euler, which was 105.96 cycles/s.

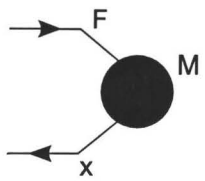

Figure 2.22: M point [10]

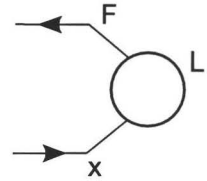

Figure 2.23: L point [10]

2.2 Previous Work

In this section we look two implementations of mass-spring systems: CORDIS-ANIMA and TAO. Several other mass-spring systems have been built including PhyMod and Cymatic [22].

2.2.1 CORDIS-ANIMA

The CORDIS-ANIMA system consists of two main modules: the CORDIS module which does the sound synthesis and the ANIMA module which creates the computer graphics. This software was built in several stages beginning in 1978 [10] by ACROE (Association for the Creation and Research on Expression Tools) in France. The philosophy behind the system was that instead of simulating the sounds themselves, as in signal processing or wave table synthesis, CORDIS-ANIMA would simulate the sound producing object [9] —i.e. simulate the musical instrument rather than just simulating the sound it produces.

Communication Points

CORDIS consists of a collection of intercommunicating objects. The objects communicate through *communication points* which can be of 2 types [10]:

1. **M points** An *M point* takes the input of a force and outputs a displacement (figure 2.22).

2. **L points** An *L point* takes the input of a displacement and outputs a force (figure 2.23).

Both the displacements and force are vectors of n dimensions. Objects must be linked so that an output force in one object is connected to an input force in another object and an output displacement is connected to input displacement. An M point can take any number of forces as input; these forces are summed together. An L point, however, can take only one displacement as an input, since an object can only be at one position at a given time. This restricts the way objects can be linked together. For example, objects containing only one communication point can only be linked in star shaped topologies with the M point at the center (figure 2.24).

Objects containing two L points or one M point can be connected in arbitrary topologies (figure 2.25). This corresponds to the mass-spring model with the masses being represented by the elements with one M point (matter points) and the dampers and springs by elements with 2 L points (link elements).

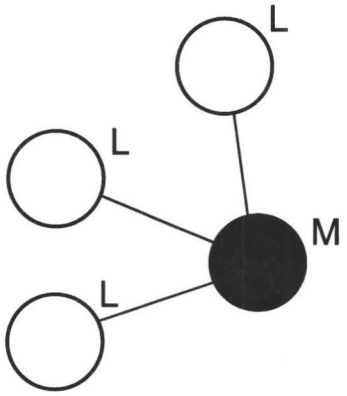

Figure 2.24: Connection of objects with 1 communication point

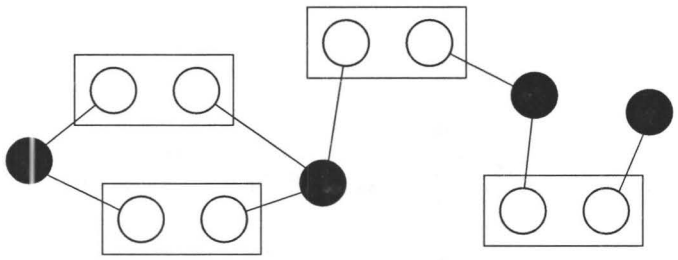

Figure 2.25: Connection of matter points and link elements

Physical Modules

Physical modules are used to represent the physical elements of the system—the masses, springs and dampers. They have force and position as inputs and outputs.

- **Mass element**
  These represent point masses. Each element contains a mass and a position. It receives one or more forces as input and calculates its new position as output. The new position is calculated using the backward Euler scheme as presented in section 1.2.4:

$$x_n = F_{n-1}/m + 2x_{n-1} - x_{n-2}$$

  where x_n is the position at the current time, F_{n-1} is the total force at the previous time step and m is the mass. Notice that F_{n-1} is used instead of F_n — i.e. the new position is calculated using the forces in the previous time step. This is done to make the scheme realizable — to avoid a delay free loop when the elements are linked together.

- **Spring element**
  This is a link element, linking two mass elements. It takes the position of the two masses as inputs and outputs the forces acting on each end of the spring. The equations are:

$$F_{r2} = k(x_{1:n-1} - x_{2:n-1} - L_0)$$
$$F_{r1} = -F_{r2},$$

where F_{r1} and F_{r2} are the forces at each end of the spring, k is the spring's stiffness constant and L_0 is the equilibrium length of the spring. Again, this equation differers from the one developed in section 1.2.4 by using the positions at the previous time step to calculate the current forces.

- **Damper element**
  The damper is also a link element, linking two mass elements. The damping is proportional to the relative velocities of the two masses the damper connects. The equations are:

$$F_{f2} = Z(x_{1:n-1} - x_{1:n-2} - x_{2:n-1} + x_{2:n-2})$$
$$F_{f1} = -F_{f2},$$

  where F_{f1} and F_{f2} are the forces at each end of the damper and Z is the coefficient of viscosity. As with the other elements, the forces of the damper at the current time step are calculated using the velocities from the previous time step.

Figure 2.26 shows an example of the CORDIS model. An external force F_e is acting on the point mass X_1. Mass X_1 is also connected by a spring and a damper to X_2. The three forces are summed to determine the next position of X_1. The vibrations in CORDIS are simplified to act in 1 dimension only, even though the points may be in 2 or 3 dimensional space. So, for example, the spring force acting on X_1 is determined only by the relative vertical displacements of X_1 and X_2; the x and y co-ordinates of the 2 points are not relevant to the vibration.

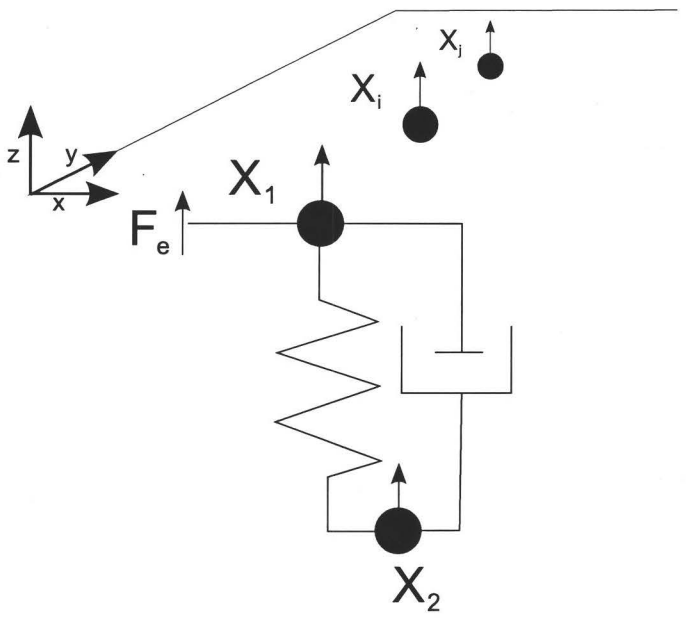

Figure 2.26: CORDIS model

Functional Modules

Functional modules do not represent physical elements, but control their behaviour or relationship with other elements. There are three type of functional modules.

1. **Dynamic Structure Variation Modules** These modules have one or more inputs (M or L points) and their output is boolean. They use the inputs to determine whether a connection is made between two elements. An example of a dynamic structure variation module is shown in figure 2.27.

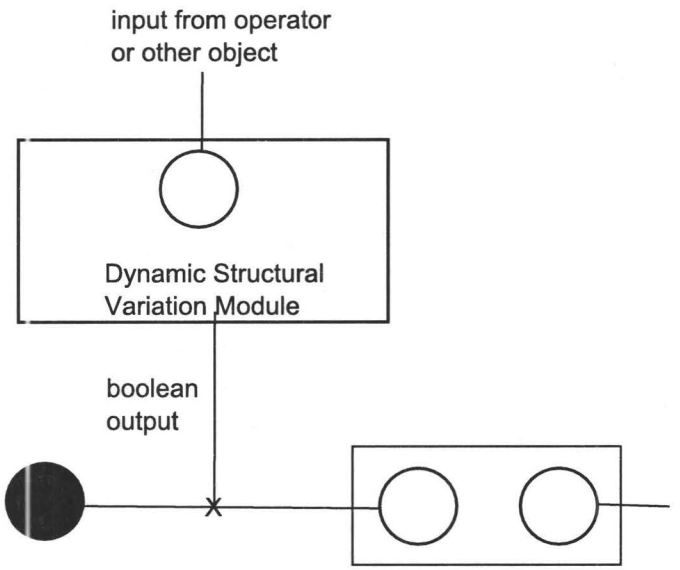

Figure 2.27: Dynamic structure variation module

2. **Dynamic Parameter Variation Modules** These modules allow the parameters of an element to be modified during the simulation.

3. **Relative Viewpoint Modification** These modules allow the scaling of data from communication points.

Instrument Interface

An important aspect of the CORDIS system is the interface with the user. The system uses physical actions (gestures) made by the user to produce the excitations of the simulated instruments. This is done by using *transducers* to convert the gestures into data usable by the system.

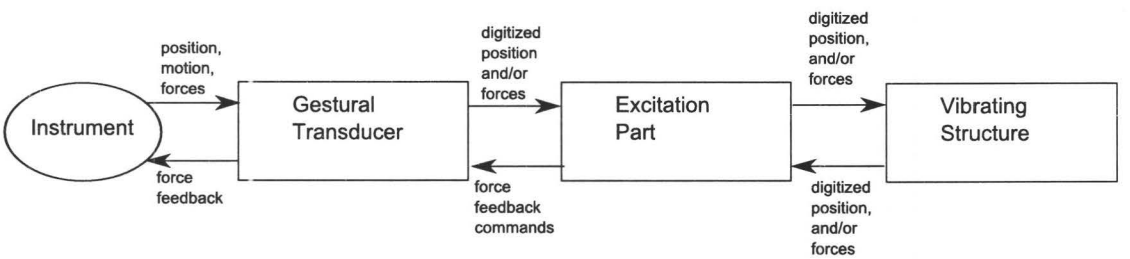

Figure 2.28: Interaction with input devices

Figure 2.28 shows the outline of the interface between the instrument and the system. The user produces motions and forces on the instrument which are converted by the

transducer into digitized position and force information. This information is passed to the excitor part of the simulated instrument and is used to calculate its vibrations. The force and position information from the excitor is in turn passed to the vibrating structure of the simulated instrument and determines its vibrations. Note that all the information is two way. The vibrating structure returns information to the excitor which can affect its vibrations. The excitor passes force feedback instructions to the transducer which provides feedback to the user about the instrument. For example, Florens in [15] gives an example of a force feedback joystick used to bow a virtual stringed instrument. The force feedback allows the user to feel the amount of pressure the bow is applying to the string.

Modelling the Connection of the Excitation and Vibrating Structure

There are three methods of excitation [8]:

- **Percussion**
  The excitation part and vibrating structures are connected for a brief period and then disconnected. This can be accomplished by creating a Dynamic Structure Variation Module that controls the connection between the excitation part and the vibrating structure. For example, the excitation part might be a drum stick and the vibrating structure the drum head. When the position of mass X_1 on the drum stick becomes less than the position of mass X_2 on the drum head a connection is made between the two objects (see figure 2.29). The spring would be set to a high constant and the user would feel the force feedback on contact. The connection between X_1 and X_2 would cause vibration in X_2 which would in turn cause vibration in the vibrating structure. Once the position of X_1 was no longer greater than X_2 the connection would be broken.

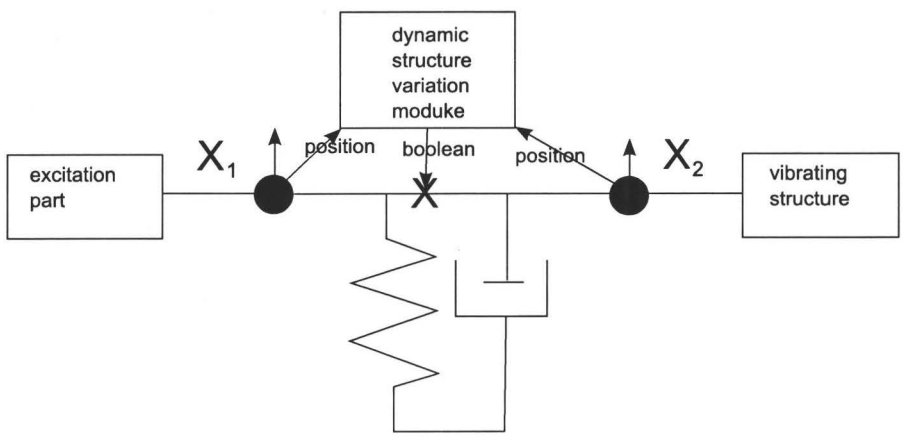

Figure 2.29: Percussive excitation

- **Plucked String**
  The left side of figure 2.30 shows the plucked string before contact is made between the pick and the string. The point mass X_1 is a point on the pick and X_2 a point on the string. When $x_1 < x_2$ there is no connection between them. The middle section of figure 2.30 shows the connection made between X_1 and X_2 when

$x_1 \geq x_2$ but $x_2 < h$. This represents when the pick is in contact with the string. When the string reaches point h the contact is broken, which is shown on the right side of 2.30. This represents the point where the force of the string causes it to slip off the pick.

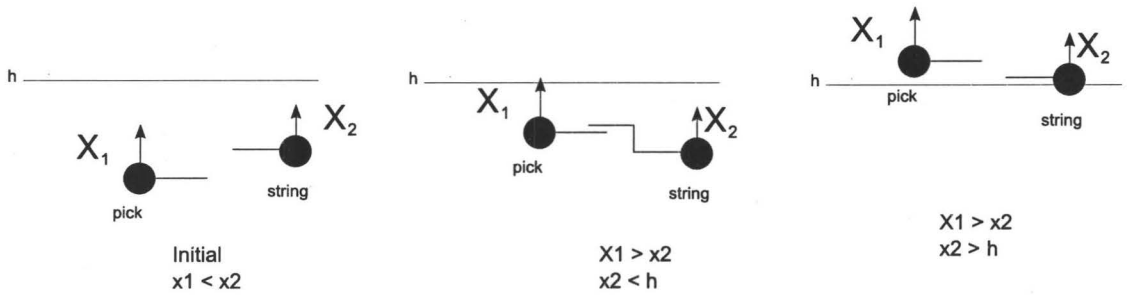

Figure 2.30: Plucked String

- **Bowed String**
  When the relative velocity of the string and the bows is less than a threshold value, the bow "captures" the string causing point X_1 on the string to move with point X_2 on the bow. When the force on the string exceeds a threshold value the string slips — i.e. the connection is broken.

2.2.2 TAO

History

TAO was created by Mark Pearson as part of his PHD thesis [33] at the University of York in 1996 under the supervision of Dr. David M. Howard. It is freely available for download at http://web.ukonline.co.uk/taosynth/.

Philosophy

Pearson's goal in creating TAO was to allow users who are not signal processing experts to create virtual musical instruments that had "organic" sounds —i.e. sounds that seem to come from a physical source [33]. The program uses the concept of cellular automata to produce the sound synthesis. The idea is that from an array of simple cells that interact locally, complex and interesting sounds can be generated.

The Cellular Model of TAO

An instrument in TAO is constructed as a two dimensional cellular automaton, with each cell having 8 immediate neighbours. Each cell contains a point mass and each pair of neighbouring masses are connected by a spring and damper. Figure 2.31 shows the current cell, S_c and its 8 neighbour cells. The arrows mark the direction of vibration: along the z axis.

Pearson demonstrates that the model can generate high level "emergent" behaviour such as reflection, refraction and diffraction of sound waves (see figure 2.32).

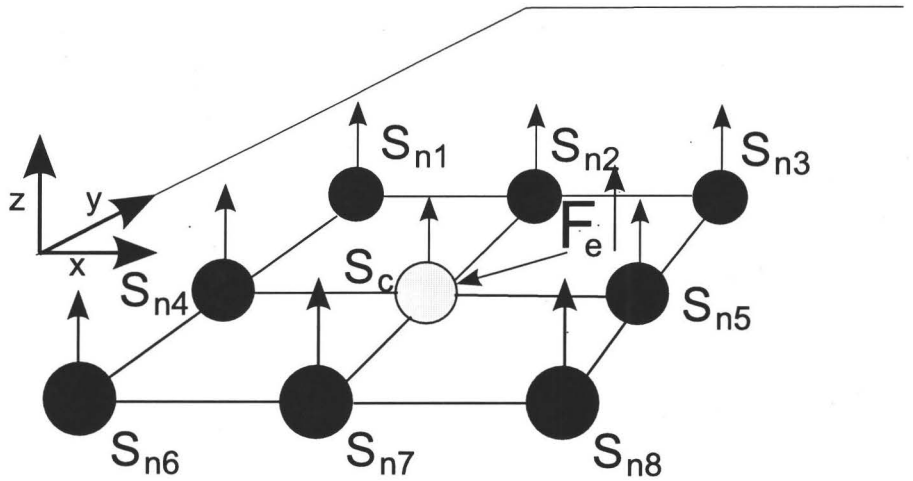

Figure 2.31: The current cell and its neighbours

Virtual microphones are used to capture output. They can be placed anywhere along the instrument and record the vibrations at that position. They can be fed into either the right or left channel for stereo output.

The instruments can be observed visually as they vibrate. Sounds are written to a wave file which can then be heard by using an audio player such as Windows Media Player or Real Audio. The sounds are not heard in real time and, for complex instruments, may take several minutes or even hours to generate 15 or 20 seconds of output.

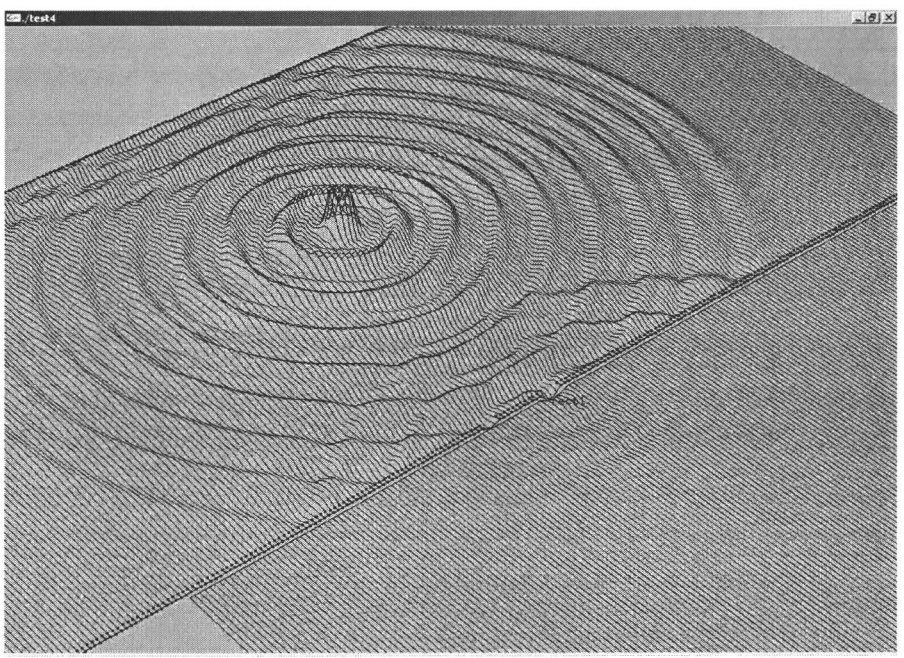

Figure 2.32: Reflection and refraction in TAO

Calculation

The high level algorithm to update the cellular model is:
for each cell

1. Calculate all forces from neighbouring cells and add them together.
2. Add any external forces.
3. Use the total force to calculate the cell's new position and velocity.

The spring force is $F = -k((s_c - s_n) - L_0)$ where k is the spring stiffness, s_c the position of the current cell, s_n the position of the neighbour cell and L_0 the spring's equilibrium length. To simplify the calculation of the force between each pair of cells the spring stiffness, k, is set to 1 and the equilibrium length is set to 0. The speed of the vibration is $\sqrt{k/m}$, so it can still be controlled by adjusting m. The calculation of the force is then:

$$F_{c:n} = s_{n:n-1} - s_{c:n-1},$$

where F_c is the the force of the current cell, s_n the position of the neighbouring cell and s_c the position of the current cell. Notice that, like CORDIS, TAO uses artificial delays — $s_{n:n-1}$ and $s_{c:n-1}$ instead of $s_{n:n}$ and $s_{c:n}$ — to make the system realizable.

The calculations to update the cell's new position and velocity (step 3) are as follows:

1. $a_n = F_n/m$, where a is the acceleration, F is the total force and m the mass.
2. $v_n = Z(v_{n-1} + a_n)$, where v is the velocity and Z the viscous damping coefficient.
3. $s_n = s_{n-1} + v_n$, where s is the position (vertical).

Steps 2 and 3 are just the backward Euler method where the time step is 1. Note that step 2 is different from that used in CORDIS: here the damping uses just the velocity of the cell (mass), where CORDIS uses the relative velocities of the masses at each end of the damper.

Scripting Language

One of the main difference between TAO and CORDIS-ANIMA is the user interaction with the instruments. Whereas CORDIS uses physical actions and transducers to produce the excitations of the instruments, TAO uses a script. This makes TAO's interface resemble a musical score created by composer and given to an orchestra, rather than a musician playing an instrument as in CORDIS.

Scripts allow the user to easily create various virtual instruments and specify how and when they are excited. There are commands to create strings, rectangles, circles and triangles. The size of these objects are determined by their frequency — i.e. a string with frequency 220 Hz is twice as long as a string of 440 Hz. The user can also specify pitch instead of frequency — i.e. $C\sharp 8$ for the $C\sharp$ above middle C. Parts of instruments can be *locked* — they do not move — or set to different damping. Instruments can also be glued together.

Chapter 3

Analysis of Mass-Spring Systems

We begin this chapter with an analysis of the symplectic Euler method—the standard method used to implement mass-spring systems—when used to implement an undamped mass-spring system containing a single mass. This analysis is also presented in [30]. We then expand the analysis to include damped mass-spring systems. We conclude by considering mass-spring systems containing multiple masses.

3.1 Undamped Mass-Spring Systems

We have seen in the introduction that using the backward Euler method to discretize the equations of the components of mass-spring systems leads to delay-free loops that make the system non-computable. A simple way to eliminate this problem is to calculate the positions at time step n based on the forces at time step $n-1$. This is the method used in the TAO system. Equation (1.6) for the mass then becomes

$$\begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ v_n \end{pmatrix} + h \begin{pmatrix} v_{n+1} \\ \frac{F_n}{m} \end{pmatrix}. \tag{3.1}$$

where equation (3.1) now uses F_n instead of F_{n+1}. We will see in section 4.3.2, that this is equivalent to the symplectic Euler method and we will refer to it by this name. We can now calculate the velocity at time $n+1$ using only the velocity and forces at time n, which are known, and can then use the velocity at time $n+1$ to calculate the position at time $n+1$. In other words, we have converted the *implicit* backward Euler method into an *explicit* method.

Figure 3.1 shows a simple mass-spring system without a damper. It shows an external force acting on mass M_1. Spring S_3 has one end connected to M_1 and the other end is fixed at point X_2. If we regard the equilibrium position of the spring to be $x(t) = 0$, where $x(t)$ is the position of M_1, the equation of this system is

$$d^2x/dt^2 = (1/m)F_{ext} - (k/m)x(t). \tag{3.2}$$

Since the symplectic Euler method is equivalent to the backward Euler with the forces delayed by 1 time step, we can approximate equation (3.2) by using the second derivative of the backward Euler. The derivative of the backward Euler is

$$x'(n) \approx \frac{1}{h}(x_n - x_{n-1}).$$

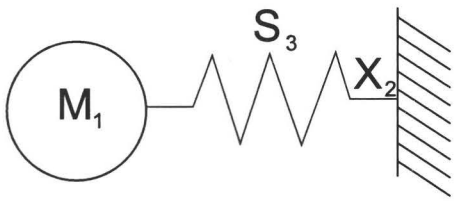

Figure 3.1: Mass-spring system

The second derivative is then

$$x''(n) \approx \left(\frac{1}{h}(x_n - x_{n-1})\right)' = \frac{1}{h^2}(x_n - 2x_{n-1} + x_{n-2}). \tag{3.3}$$

We now approximate equation (3.2) by substituting the second derivate of the backward Euler on the left side and the delayed forces on the right:

$$\frac{x_n - 2x_{n-1} + x_{n-2}}{h^2} = -(k/m)x_{n-1} + m^{-1}F_{ext:n-1}.$$

Then we take the z-transform

$$(h^{-2})\left(\hat{X}(z) - 2\hat{X}(z)z^{-1} + \hat{X}(z)z^{-2}\right) = -(k/m)\hat{X}(z)z^{-1} + m^{-1}\hat{F}_{ext}(z)z^{-1},$$

where we make use of the fact that the z-transform of a unit delay is equal to z^{-1}. The z-transform of x_n is denoted by $\hat{X}(z)$ and that of $F_{ext}(n)$ by $\hat{F}_{ext}(z)$. We then find the transfer function:

$$\hat{H}(z) = \frac{\hat{X}(z)}{\hat{F}_{ext}(z)} = \frac{(h^2/m)z^{-1}}{1 + ((h\omega)^2 - 2)z^{-1} + z^{-2}}, \tag{3.4}$$

where $\omega^2 = k/m$. The poles of the transfer function occur when

$$z^2 + ((h\omega)^2 - 2)z + 1 = 0.$$

Solving this quadratic equation gives us:

$$z = \frac{-(h\omega)^2 + 2 \pm (h\omega)\sqrt{(h\omega)^2 - 4}}{2}.$$

If $(h\omega)^2 < 4$ then

$$z = \frac{-(h\omega)^2 + 2}{2} \pm \frac{jh\omega\sqrt{4 - (h\omega)^2}}{2}.$$

Using the real and imaginary parts, we can calculate r:

$$r = |z| = \sqrt{\left(\frac{-(h\omega)^2 + 2}{2}\right)^2 + \left(\frac{h\omega\sqrt{4 - (h\omega)^2}}{2}\right)^2} = 1.$$

So on this interval, the poles map exactly to the unit circle. This means there is no numerical damping. We can calculate the frequencies in this range. Using the facts that

$\omega = 2\pi f$, where f is the frequency, and $h = 1/f_s$, where f_s is the sample rate, we find that $(h\omega)^2 < 4$ is equivalent to $f < \frac{1}{\pi} f_s$, i.e. all cases in which the frequency is less than $\frac{1}{\pi}$ times the sample rate.

We now find the frequency warping. Using ω_d for the digital angular frequency (the actual angular frequency using the numerical method) and ω_a as the analog angular frequency (the angular frequency of the original continuous system):

$$\omega_d = \tan^{-1}\left(\frac{im(z)}{re(z)}\right) = \tan^{-1}\left(\frac{h\omega_a\sqrt{4-(h\omega_a)^2}}{-(h\omega_a)^2+2}\right). \tag{3.5}$$

Figure 3.2 shows the graph of this function. As $h\omega_a$ approaches 2 (i.e. f_a approaches $(1/\pi)f_s$) the frequency becomes progressively warped. Notice that the frequency warping of symplectic Euler causes the digital frequency to be higher than the original analog frequency. The digital frequency, f_d, is in radians per sample, so to convert to radians per second we need to multiply it by the sample rate f_s.

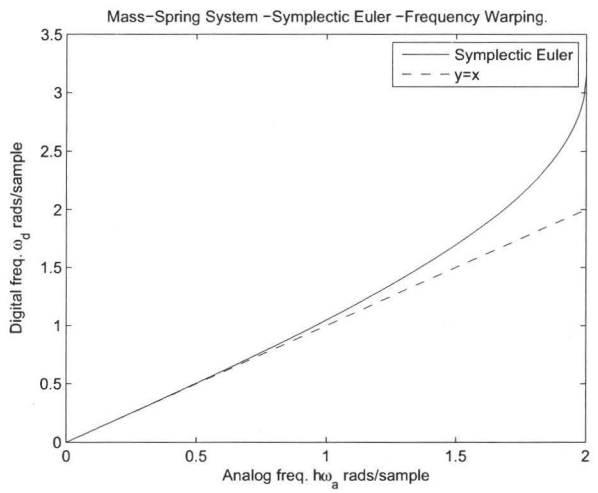

Figure 3.2: Frequency warping for undamped mass-spring system using the symplectic Euler method

Next we examine the interval where $(\omega h)^2 > 4$. We examine the pole

$$z = \frac{-(h\omega)^2 + 2 - (h\omega)\sqrt{(h\omega)^2 - 4}}{2}.$$

First, notice that this is a real number. Looking at the first 2 terms of the numerator, we see that

$$(-(h\omega)^2 + 2) < (-4 + 2) = -2.$$

The radical in the last term of the numerator is positive when $(\omega h)^2 > 4$, so the last term is always positive. Therefore, the numerator is always less than -2 and value of the pole is always less than -1. Since this pole is outside the unit circle, the system is unstable in all cases in which $(\omega h)^2 > 4$. So when the analog frequency is more than $(1/\pi)f_s$, the system in unstable.

The final case is when $(\omega h)^2 = 4$. This results in $z = -1$.

Figure 3.3 shows the plot of the poles on the z-plane. The two sets of conjugate poles trace the upper and lower halves of the unit circle. At frequency $f_a = (1/\pi)f_s$, they meet at $z = -1$. Then one pole continues along the real axis toward the left and the other along the real axis to the right.

So for stability, it is required that

$$f_a \leq (1/\pi)f_s, \tag{3.6}$$

where f_a is the frequency and f_s the sample rate.

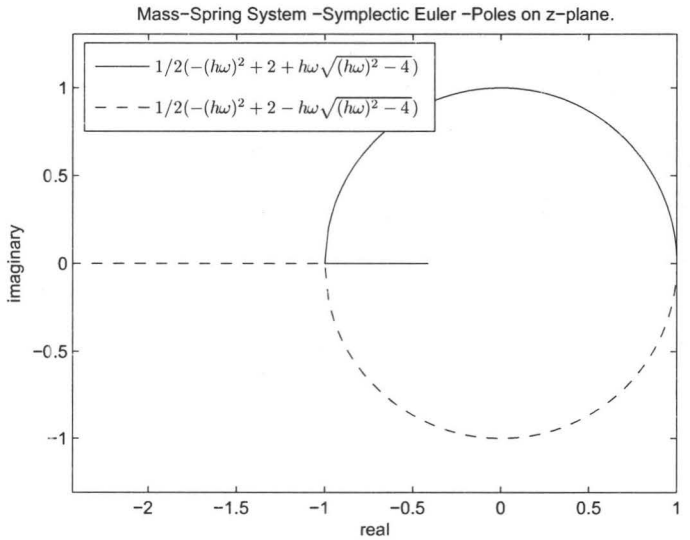

Figure 3.3: Poles on the z-plane for the symplectic Euler

3.2 The Mass-spring System with Damping

In this section we look at the stability and accuracy of the symplectic Euler method when used to simulate damped mass-springs systems. We start by finding the analytical solution of a damped mass-spring system containing a single mass, a single spring and a single damper. We then find the z-transform of this equation when it is approximated by the symplectic Euler method. We use the z-transform to find the damping and the frequency of the transformed equation. We also find the conditions under which this equation is stable.

3.2.1 The Analytical Solution of the Damped Mass-spring System

Figure 3.4 shows a damped mass-spring containing only one mass, M_1, one spring, S_1 and one damper, D_1.

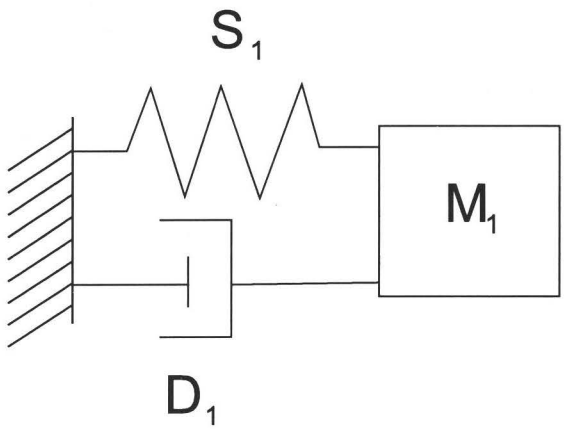

Figure 3.4: Damped mass-spring system

The equation for this system is:

$$mx''(t) + kx(t) + Zx'(t) = 0, \tag{3.7}$$

where $x(t)$ is the distance of the mass from its equilibrium position, m is the mass, k the spring stiffness and Z the viscous damping coefficient.

We can solve this equation by finding the roots of the characteristic equation:

$$r^2 + \frac{Z}{m}r + \frac{k}{m} = 0.$$

We use the substitutions $\gamma = Z/m$ and $\omega_0^2 = k/m$:

$$r^2 + \gamma r + \omega_0^2 = 0.$$

The roots of this characteristic equation are

$$r = \frac{-\gamma \pm \sqrt{\gamma^2 - 4\omega_0^2}}{2}.$$

For the system to vibrate we require that $\gamma^2 < 4\omega_0^2$, so

$$r = \frac{-\gamma \pm i\sqrt{4\omega_0^2 - \gamma^2}}{2}. \tag{3.8}$$

The condition dividing vibrating systems from those that do not vibrate occurs when

$$4\omega_0^2 = \gamma^2 \text{ or } \gamma = 2\omega_0. \tag{3.9}$$

This value is known as *critical damping*.

The general solution of equation (3.7), when $\gamma^2 < 4\omega_0^2$, can be show to be [7]

$$x(t) = Re^{-\gamma t/2}\sin(\mu t + \phi). \tag{3.10}$$

This shows that the damped mass-spring system has a starting amplitude of R. This amplitude is being decreased by the term $e^{-\gamma t/2}$. The frequency (actually the *quasi-frequency* since the system is not strictly periodic) is $\mu = (1/2)\sqrt{4\omega_0^2 - \gamma^2}$. As the damping approaches zero this equation becomes $x(t) = R\sin(\omega_0 t + \phi)$.

3.2.2 The Damped Mass-spring System using the Symplectic Euler Method

We next examine the damping and frequency of the damped mass-spring system, when approximated by the symplectic Euler method, by using the z-transform.

We start by writing the symplectic Euler approximation of damped mass-spring equation

$$mx''(t) + kx(t) + Zx'(t) = F_{ext}(t), \tag{3.11}$$

where F_{ext} is an external force acting on the mass. Using the substitutions from the previous section we can write equation (3.11) as

$$x''(t) = -\omega_0^2 x(t) - \gamma x'(t) + \frac{1}{m}F_{ext}(t) \tag{3.12}$$

Since the symplectic Euler is equivalent to the backward Euler with the forces delayed by one time step, we can approximate equation (3.12) by substituting equation (3.3) for the acceleration and delaying the forces by 1 time step:

$$\begin{aligned}&\frac{1}{h^2}(x_n - 2x_{n-1} + x_{n-2}) = \\ &-\omega_0^2 x_{n-1} - \frac{\gamma}{h}(x_{n-1} - x_{n-2}) + \frac{1}{m}F_{ext:n-1}.\end{aligned} \tag{3.13}$$

The z-transform of equation (3.13) is

$$\begin{aligned}&\frac{1}{h^2}\left(\hat{X}(z) - 2\hat{X}(z)z^{-1} + \hat{X}(z)z^{-2}\right) = \\ &-\omega_0^2\hat{X}(z)z^{-1} - \frac{\gamma}{h}\left(\hat{X}(z)z^{-1} - \hat{X}(z)z^{-2}\right) + \frac{1}{m}\hat{F}_{ext}(z)z^{-1}, \\ &\hat{X}(z)\left(\frac{1}{h^2} + \left(\omega_0^2 + \frac{\gamma}{h} - \frac{2}{h^2}\right)z^{-1} + \left(\frac{1}{h^2} - \frac{\gamma}{h}\right)z^{-2}\right) = \frac{1}{m}\hat{F}_{ext}(z)z^{-1},\end{aligned}$$

and the transfer function is

$$\hat{H}(z) = \frac{\hat{X}(z)}{\hat{F}_{ext}(z)} = \frac{z^{-1}/m}{\frac{1}{h^2} + \left(\omega_0^2 + \frac{\gamma}{h} - \frac{2}{h^2}\right) z^{-1} + \left(\frac{1}{h^2} - \frac{\gamma}{h}\right) z^{-2}}$$
$$= \frac{(h^2/m)z^{-1}}{1 + \left((\omega_0 h)^2 + \gamma h - 2\right) z^{-1} + (1 - \gamma h)\, z^{-2}}.$$

The poles of the transfer function occur when

$$1 + \left((\omega_0 h)^2 + \gamma h - 2\right) z^{-1} + (1 - \gamma h)\, z^{-2} = 0,$$
$$z^2 + \left((\omega_0 h)^2 + \gamma h - 2\right) z + (1 - \gamma h) = 0. \tag{3.14}$$

The roots of equation(3.14) are

$$z = \frac{1}{2}\left(2 - (\omega_0 h)^2 - \gamma h \pm \sqrt{\left((\omega_0 h)^2 + \gamma h - 2\right)^2 - 4(1 - \gamma h)}\right)$$
$$= \frac{1}{2}\left(2 - (\omega_0 h)^2 - \gamma h \pm \omega_0 h \sqrt{(\omega_0 h)^2 + 2\gamma h + \left(\frac{\gamma}{\omega_0}\right)^2 - 4}\right). \tag{3.15}$$

There are 2 cases to consider in equation (3.15): 1) z is complex and 2) z is real.

Case 1) - z is Complex

We look at the case in which z contains an imaginary component — i.e. when

$$\left((\omega_0 h)^2 + 2\gamma h + \left(\frac{\gamma}{\omega_0}\right)^2\right) < 4.$$

We first calculate the range of values for ω_0 of this case. If

$$(\omega_0 h)^2 + 2\gamma h + \left(\frac{\gamma}{\omega_0}\right)^2 = 4 \tag{3.16}$$
$$\text{then } \omega_0^4 + \left(2\frac{\gamma}{h} - \frac{4}{h^2}\right)\omega_0^2 + \left(\frac{\gamma}{h}\right)^2 = 0,$$

by multiplying both side by ω_0^2/h^2 and rearranging. The roots of this equation are

$$\omega_0^2 = \frac{1}{2}\left(\frac{4}{h^2} - \frac{2\gamma}{h} \pm \sqrt{\left(2\frac{\gamma}{h} - \frac{4}{h^2}\right)^2 - 4\left(\frac{\gamma}{h}\right)^2}\right).$$

This simplifies to

$$\omega_0^2 = \frac{2}{h^2} - \frac{\gamma}{h} \pm 2\sqrt{\frac{1}{h^4} - \frac{\gamma}{h^3}}. \tag{3.17}$$

If z is complex, ω_0 falls between these two roots.

What is the maximum value that $\omega_0 = \sqrt{k/m}$ can have in this range? As h approaches 0

$$\begin{aligned}\omega_0^2 &= \lim_{h\to 0}\left(\frac{2}{h^2} - \frac{\gamma}{h} + 2\sqrt{\frac{1}{h^4} - \frac{\gamma}{h^3}}\right) \\ &= \frac{2 - \lim_{h\to 0}(\gamma h) + 2\sqrt{1 - \lim_{h\to 0}(\gamma h)}}{h^2} \\ (\omega_0 h)^2 &= 4; \omega_0 h = 2 \quad \text{(since we assume } h > 0 \text{ and } \omega_0 > 0).\end{aligned}$$

Thus, when the sample rate is high, the maximum value for ω_0 in this range approaches 2 radians per sample, or $(1/\pi)f_s$ samples per second, where f_s is the sample rate. This is the same maximum value we obtained using the undamped mass-spring system. We can also see from equation (3.16) that when γ approaches 0, the maximum value for ω_0 is also 2 radians per sample. As the sample rate decreases or the damping increases, the maximum value is lowered according to the equation

$$\omega_{0,max} = \sqrt{\frac{2}{h^2} - \frac{\gamma}{h} + 2\sqrt{\frac{1}{h^4} - \frac{\gamma}{h^3}}}.$$

This equation depends on both γ and h. Figure (3.5) shows the dependence of the maximum value for ω_0 in this range on the sample rate when γ is fixed. For this figure, γ is set at $1,000$, while the sample rate ranges from $1,000$ samples per second to $40,000$ samples per second. As the sample rate increases, the maximum value for ω_0 approaches 2 radians per sample. Figure (3.6) shows the dependence of the maximum value for ω_0

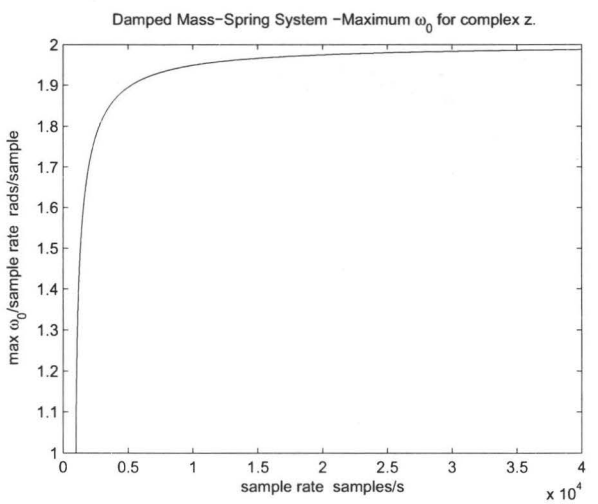

Figure 3.5: Damped mass-spring system -Max ω_0 versus sample rate

in this range on γ when the sample rate is fixed. For this figure, $h = .001$ or $f_s = 1000$, while γ ranges from 0 to $1,000$. As γ approaches 0, the maximum value for ω_0 approaches 2 radians per sample.

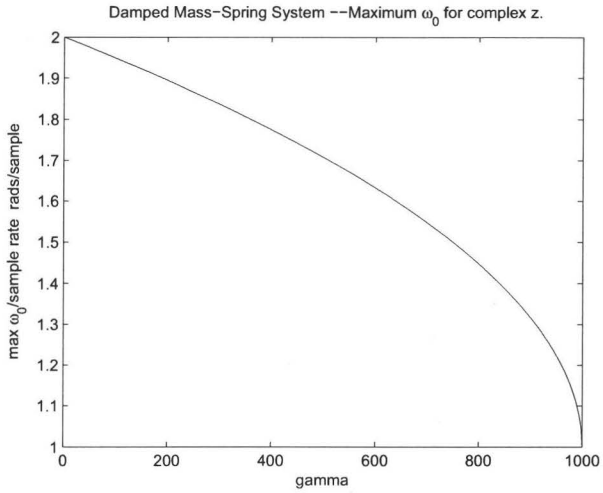

Figure 3.6: Damped mass-spring system -Max ω_0 versus γ

Now we look at the minimum value for ω_0 in this range. As h approaches 0

$$\begin{aligned}\omega_{0,min}^2 &= \lim_{h\to 0}\left(\frac{2}{h^2}-\frac{\gamma}{h}-2\sqrt{\frac{1}{h^4}-\frac{\gamma}{h^3}}\right)\\ &= \frac{2-\lim_{h\to 0}(\gamma h)-2\sqrt{1-\lim_{h\to 0}(\gamma h)}}{\lim_{h\to 0} h^2} = 0/0.\end{aligned}$$

Using L'Hospital's rule twice results in

$$\omega_{0,min} = \frac{\gamma}{2}.$$

We see that this is the *critical damping* value from equation (3.9). So when the value for h is small, the system starts to vibrate near the critical damping value as it should. Figure (3.7) shows a mass-spring system with γ set to $1,000$ while the frequency ranges from $1,000$ to $40,000$ samples per second. As the sample rate increases, the minimum value for ω_0 approaches 500 which is $\gamma/2$. We can solve the smaller root of equation (3.17) for γ to give us a formula for the critical damping. This works out to be

$$\gamma_{cr} = -\omega_0^2 h + 2\omega_0. \tag{3.18}$$

Figure (3.8) shows the critical damping for $\omega_0 = 1,000$ while the sample rate ranges from $1,000$ to $40,000$ samples per second. This shows that at high sample rates the critical damping is close to twice ω_0, while at low sample rates the critical damping becomes less than this value. This means that if the sample rate is too low, a system may stop vibrating with a damping coefficient lower than the critical damping (i.e. it will take less damping to stop the vibration than it should). For example, if the sample rate is $5,000$ samples per second ($h = 1/5000$) and $\omega_0 = 1,000$, the symplectic Euler approximation of the system will not vibrate with $\gamma > 1,800$ although the critical damping for the analog equation is $2,000$.

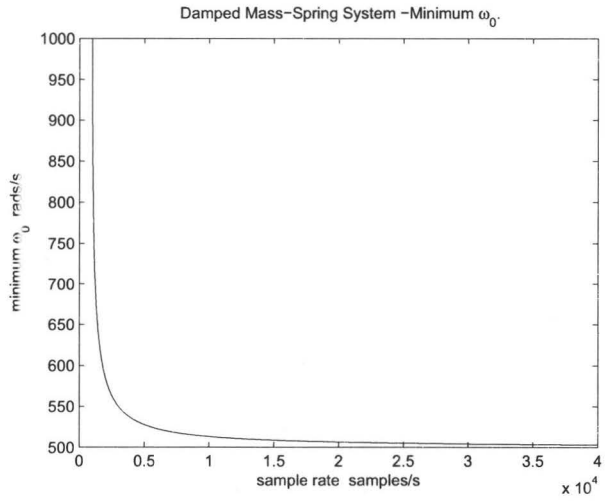

Figure 3.7: Damped mass-spring system -Min ω_0 versus sample rate

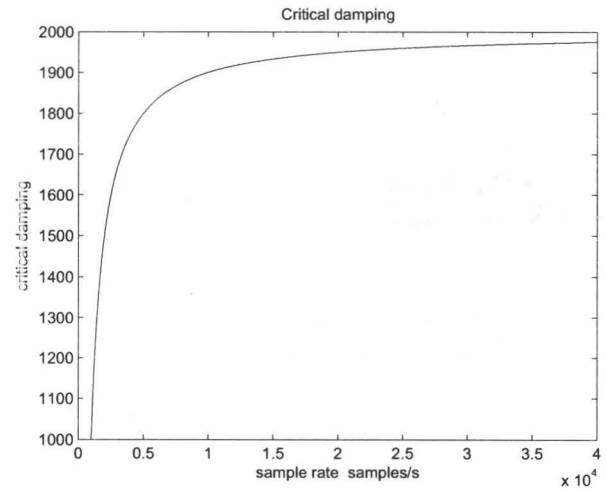

Figure 3.8: Critical damping versus sample rate

We now look at the accuracy of the damping for case 1) (z is complex). In this case equation (3.15) becomes

$$z = \frac{1}{2}\left(2 - (\omega_0 h)^2 - \gamma h \pm i\omega_0 h\sqrt{4 - (\omega_0 h)^2 - 2\gamma h - \left(\frac{\gamma}{\omega_0}\right)^2}\right). \tag{3.19}$$

We can calculate the length of z using the imaginary and reals parts:

$$\begin{aligned} |z| &= \sqrt{re(z)^2 + imag(z)^2} \\ &= \frac{1}{2}\sqrt{(2 - (\omega_0 h)^2 - \gamma h)^2 + \left(\omega_0 h\sqrt{4 - (\omega_0 h)^2 - 2\gamma h - \left(\frac{\gamma}{\omega_0}\right)^2}\right)^2}. \end{aligned}$$

This simplifies to

$$|z| = \sqrt{1 - \gamma h}. \tag{3.20}$$

We see from equation (3.20) that, for this case, the length of z does not depend on the frequency. We can also observe that length decreases with the amount of damping (since $\gamma = Z/m$), and approaches 1 as the damping coefficient, Z, approaches 0.

Figure 3.9 shows the poles on the z-plane of the symplectic Euler approximation of the damped mass-spring equation. The value for γ is 40 and the time step, h, is .001.

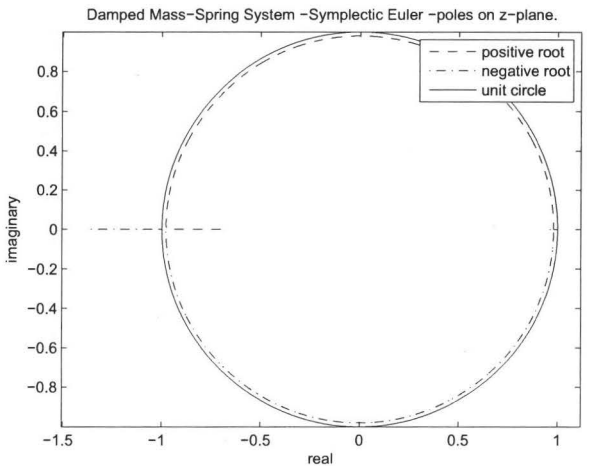

Figure 3.9: Damped mass-spring system using symplectic Euler -poles on z-plane

The radius of the circle containing the poles is $\sqrt{1-\gamma h}$, which depends on both γ and the time step h. If the damping or the time step increase, the radius of the circle will become smaller.

From equation (3.10), the damping of the analog system is $e^{-\gamma t/2}$. From equation (3.20), the damping of the system using the symplectic Euler approximation is $\sqrt{1-\gamma h}^n$. How do these two values compare? We can rewrite t, the time, as nh, where n is the sample number and h is the length of the time step. Then we can write the analog damping as $e^{-\gamma nh/2}$. Figure 3.10 shows the damping for the first 400 samples of the analog mass spring system and the symplectic Euler approximation. The value used for γh was .1. Figure 3.11 shows the absolute error of this system for the first 100 samples. The error reaches a maximum around the 20th sample, which is the $(1/(\gamma h) \times 2)$th sample.

As γh decreases, so does the error. Figures 3.12 and 3.13 show the damping results when $\gamma h = .01$. The difference between the symplectic Euler and the analog system can no longer be distinguished on the graph (figure 3.12). Note that figure 3.13 has the same shape as figure 3.11 with the maximum error occurring around the $(1/(\gamma h) \times 2)$th sample — this time the 200th sample. The magnitude of the error is one tenth that of figure 3.11.

The *time constant* of the decay, τ, is the time it takes for the peak amplitude to decay

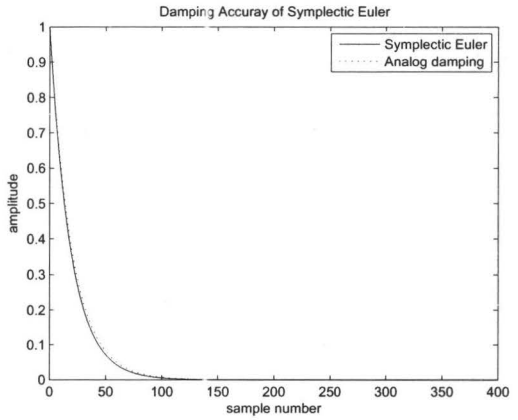

Figure 3.10: Comparison of damping with $\gamma h = .1$

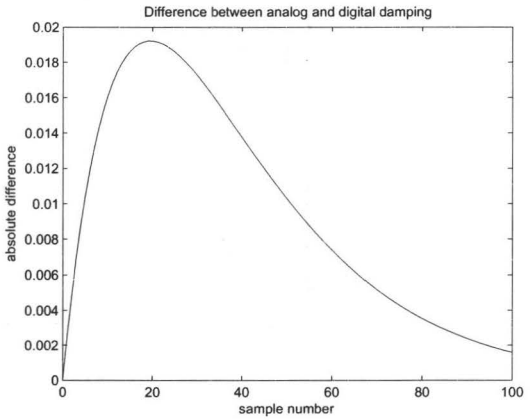

Figure 3.11: Accuracy of damping with $\gamma h = .1$

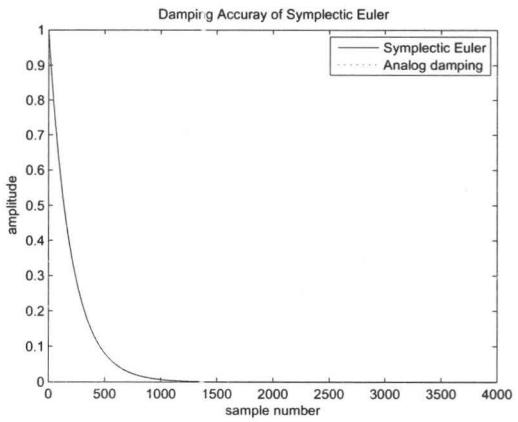

Figure 3.12: Comparison of damping with $\gamma h = .01$

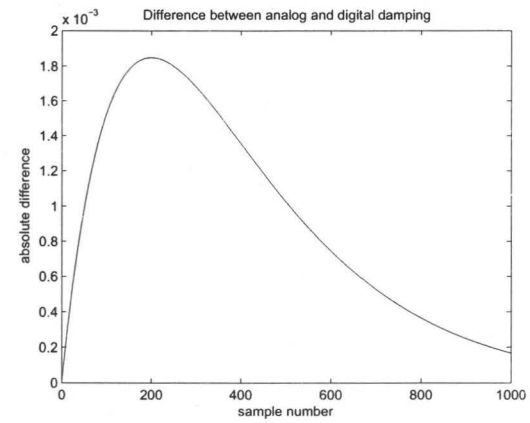

Figure 3.13: Accuracy of damping with $\gamma h = .01$

by $1/e$ [37]. Then τ for the mass spring system with amplitude A is

$$\begin{aligned} Ae^{-\gamma\tau/2} &= A(1/e) \\ ln(e^{-\gamma\tau/2}) &= ln(1/e) \\ -\gamma\tau/2 &= -1 \\ \tau &= 2/\gamma. \end{aligned}$$

Sounds with time constants less than around .005 seconds are not heard as pitches, but merely clicks. If the sample rate is $44,100$ samples per second, for a time constant of .005, γ is 400 and γh is 9.07×10^{-3}. This means that for any musical sound, the damping of the symplectic Euler method will be quite accurate at this sampling rate.

We next look at the frequency warping when z is complex. From equation (3.19), we can calculate the frequency using the real and imaginary components. Using ω_d to

denote the actual frequency obtained using the symplectic Euler,

$$\omega_d = \tan^{-1}\left(\frac{imaginary(z)}{real(z)}\right)$$
$$= tan^{-1}\left(\frac{\omega_0 h\sqrt{4-(\omega_0 h)^2-2\gamma h-\left(\frac{\gamma}{\omega_0}\right)^2}}{2-(\omega_0 h)^2-\gamma h}\right)$$

Figure 3.14 shows the frequency warping of the symplectic Euler method when the damping is quite high: $h = .001$, $\gamma = 500$ and $\gamma h = .5$. The analog frequency is calculated as $\omega_a = (1/2)\sqrt{4\omega_0^2 - \gamma^2}$, and is slightly lower than ω_0. Note that the digital frequency has reached the Nyquist limit of π radians per sample at around $\omega_0 h = 1.7$ radians per sample. This is somewhat less than the frequency warping of the undamped system, which does not reach the Nyquist limit until $\omega_0 h = 2.0$ radians per sample. This is consistent with equation (3.17), which gives .293 as the lower limit and 1.707 as the upper limit for z being complex.

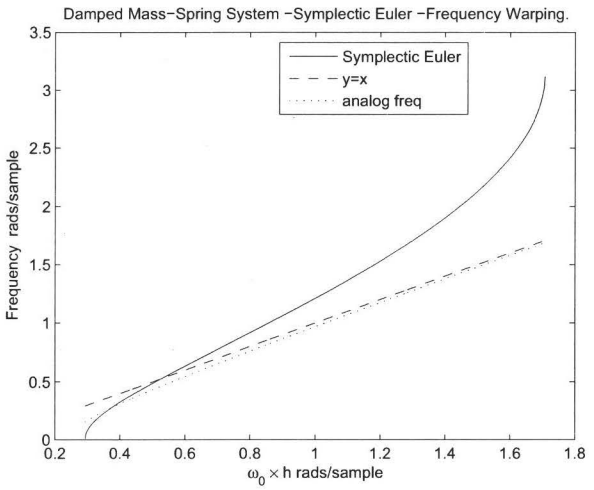

Figure 3.14: Frequency for damped mass-spring system, $\gamma h = .5$

Figure 3.15 shows the frequency warping for values more typical of sound synthesis: $h = 1/44100$, $\gamma = 200$ and $\gamma h = 4.54 \times 10^{-3}$. This time the frequency does not reach the Nyquist limit until very close to $\omega_0 h = 2.0$ radians per sample. The frequency warping for these values is very close to that of the undamped system. Equation (3.17) gives 0.0023 as the lower limit and 1.998 as the upper limit for z being complex.

Case 2) - z is Real

Equation (3.17) gives us 2 conditions for z to be real. If ω_0 is less than the lower root of this equation, the system has a digital frequency of zero, and hence produces no sound. If ω_0 is greater than the upper root, the system has a digital frequency of π radians per sample. As the leftmost pole approaches -1, the damping becomes smaller. The system

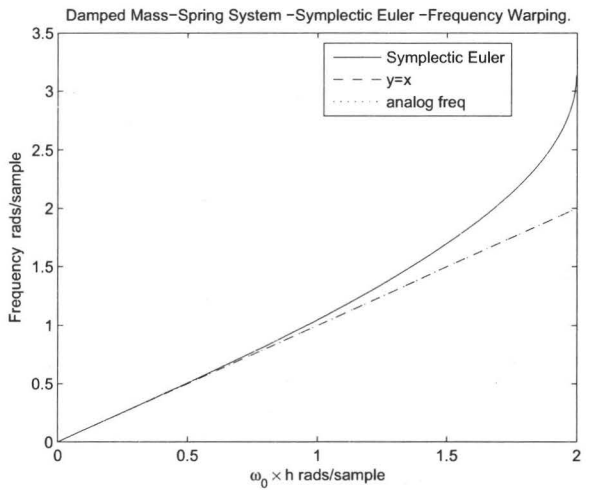

Figure 3.15: Frequency for damped mass-spring system, $\gamma h = 4.54 \times 10^{-3}$

will become unstable when z becomes less than -1. From equation (3.15), the lower root of equation (3.17) equals -1 when

$$\frac{1}{2}\left(2 - (\omega_0 h)^2 - \gamma h - \omega_0 h\sqrt{(\omega_0 h)^2 + 2\gamma h + \left(\frac{\gamma}{\omega_0}\right)^2 - 4}\right) = -1$$

$$4 - (\omega_0 h)^2 - \gamma h = \omega_0 h\sqrt{(\omega_0 h)^2 + 2\gamma h + \left(\frac{\gamma}{\omega_0}\right)^2 - 4}.$$

Squaring both sides and simplifying results in

$$(\omega_0 h)^2 + 2\gamma h - 4 = 0.$$

For a fixed time step, h, we can solve for ω_0

$$\omega_0^2 = \frac{4 - 2\gamma h}{h^2} \tag{3.21}$$
$$\omega_0 = \frac{1}{h}\sqrt{4 - 2\gamma h}.$$

So for stability we require that

$$\omega_0 \leq \frac{1}{h}\sqrt{4 - 2\gamma h}. \tag{3.22}$$

Figure 3.16 shows the maximum stable value for $\omega_0 h$ with γ set to $1,000$ while the sample rate ranges from $1,000$ to $40,000$. As h becomes small the system is stable for values of $\omega_0 h$ approaching 2.

As the value of γh increases, the circular region on the z-plane where z is complex becomes smaller. Figure (3.17) shows the poles of the damped mass-spring system where $\gamma h = .98$. When $\gamma h = 1.0$ the circular region disappears. For $\gamma h \geq 1.0$, there are only two possible digital frequencies: zero and π radians per sample. If the pole

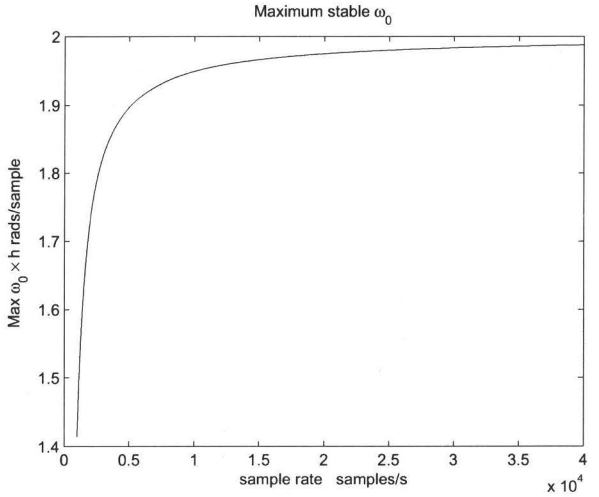

Figure 3.16: Maximum stable ω_0 for damped mass-spring system

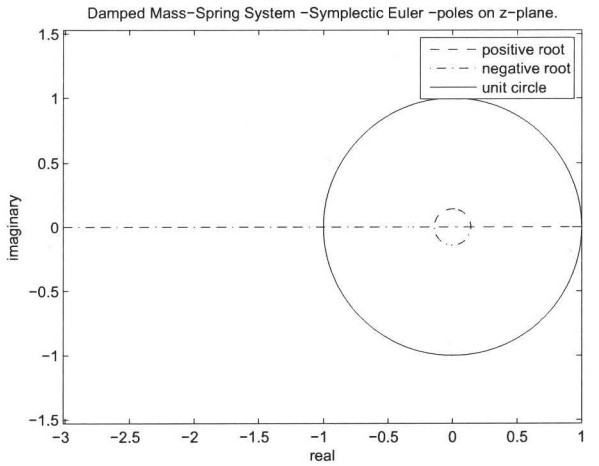

Figure 3.17: Damped mass-spring system -poles on z-plane where $\gamma h = .98$

with the larger magnitude is negative, the digital frequency will be π radians per sample; otherwise it will be zero. From equation (3.15), the pole with the larger magnitude is negative when

$$2 - (wh)^2 - \gamma h < 0,$$

so, when $\gamma h \geq 1.0$, the curve dividing systems that do not vibrate and those that vibrate at π radians per sample is

$$2 - (wh)^2 - \gamma h = 0. \tag{3.23}$$

3.3 Generalizing to Mass-spring Systems with Multiple Degrees of Freedom

So far, we have just analyzed mass-spring systems with a single mass. We now consider systems with multiple masses. Since each mass can move independently of the other

masses, these systems are said to have multiple degrees of freedom. We first look at the analytical solution of the general mass-spring system with n degrees of freedom using the method presented by Meirovitch [29].

3.3.1 Analytical Solution of Mass-Spring Systems

We can find the analytical solution of a mass-spring system by using the state-space method. The state space method uses a vector, $\mathbf{x}(t)$, of state variables. The equation describing the state variables of the state-space system is [2]

$$\frac{d}{dt}\mathbf{x}(t) = A\mathbf{x}(t) + B\mathbf{u}(t),$$

where the vector $\mathbf{u}(t)$ is the input and A and B are matrices. The output is produced from the state variables and input by the equation

$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t).$$

The state variables should contain the information needed to calculate the system's configuration at each point in time, so for a mass-spring system obvious choice is the displacement and velocity of each of the masses. The first step is to determine the matrix A. We start by looking the 2 mass system shown in figure 3.18. The equations of this

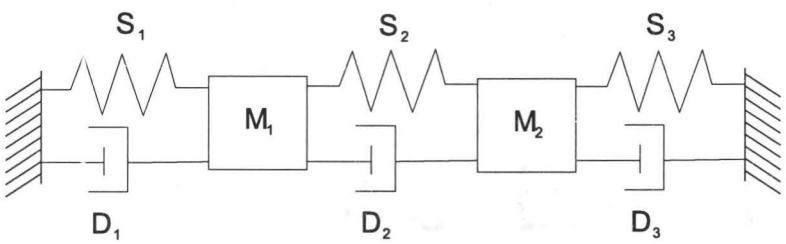

Figure 3.18: Two mass system with damping

system are

$$\begin{aligned} m_1 x_1''(t) &= -k_1 x_1(t) - k_2(x_1(t) - x_2(t)) - Z_1 x_1'(t) - Z_2((x_1'(t) - x_2'(t)), \\ m_2 x_2''(t) &= k_2(x_1(t) - x_2(t)) - k_3 x_2(t) + Z_2(x_1'(t) - x_2'(t)) - Z_3 x_2'(t), \end{aligned}$$

where k_i is the spring stiffness coefficient for spring S_i and Z_i is the damping coefficient for damper D_i. Solving for $x_1''(t)$ and $x_2''(t)$ in terms of the state variables x_1, x_2, x_1' and x_2' gives us

$$\begin{aligned} x_1''(t) &= -x_1(t)\frac{k_1 + k_2}{m_1} + x_2(t)\frac{k_2}{m_1} - x_1'(t)\frac{Z_1 + Z_2}{m_1} + x_2'(t)\frac{Z_2}{m_1}, \\ x_2''(t) &= x_1(t)\frac{k_2}{m_2} - x_2(t)\frac{k_2 + k_3}{m_2} + x_1'(t)\frac{Z_2}{m_2} - x_2'(t)\frac{Z_2 + Z_3}{m_2}. \end{aligned}$$

We can now write the equation for the update of the state vector as

$$\mathbf{x}'(t) = A\mathbf{x}(t),$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_1' \\ x_2' \end{pmatrix}' = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_1+k_2}{m_1} & \frac{k_2}{m_1} & -\frac{Z_1+Z_2}{m_1} & \frac{Z_2}{m_1} \\ \frac{k_2}{m_2} & -\frac{k_+k_3}{m_2} & \frac{Z_2}{m_2} & -\frac{Z_2+Z_3}{m_2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_1' \\ x_2' \end{pmatrix}. \tag{3.24}$$

We can rewrite this in terms of the mass, stiffness and damping matrices. The mass matrix, M, contains each of the masses along its diagonal

$$M = \begin{pmatrix} m_1 & 0 & 0 & \dots & 0 \\ 0 & m_2 & 0 & \dots & 0 \\ \dots & & & & \\ 0 & \dots & 0 & 0 & m_{20} \end{pmatrix}.$$

The stiffness influence coefficients, k_{ij}, are defined as the forces required for a unit displacement of mass i, with all other masses $j \neq i$ having a displacement of zero. On a simulated string such as Figure 3.18, the force required to displace mass M_i one unit to the right is $k_i + k_{i+1}$, since the spring on the left must be expanded by one unit and the spring on the right compressed by one unit. Masses M_{i-1} and M_{i+1} require forces in the opposite direction to keep them in place. The stiffness matrix, K, for a linear string is then a tridiagonal matrix with the ith plus the $(i+1)$th spring's stiffness coefficient on the diagonal, the negative of the ith stiffness coefficient to its left, and the negative of the $(i+1)$th stiffness coefficient to its right:

$$K = \begin{pmatrix} k_1 + k_2 & -k_2 & 0 & \dots & & 0 \\ -k_2 & k_2 + k_3 & -k_3 & \dots & & 0 \\ 0 & -k_3 & k_3 + k_4 & -k_4 & \dots & 0 \\ \dots & & & & & \\ 0 & \dots & & 0 & -k_{20} & k_{20} + k_{21} \end{pmatrix}.$$

The damping matrix, Z, contains damping coefficients, Z_{ij}, that are defined as the forces required for a unit velocity of mass i to the right, with all other masses $j \neq i$ having a velocity of zero [21]. On a simulated string containing n masses, $n+1$ springs and $n+1$ dampers, the force required for a velocity of one to the right for mass M_i is $(Z_i + Z_{i+1})$, since the dampers D_i and D_{i+1} both have a velocity of one. The masses M_{i-1} and M_{i+1} both need forces of $-Z_i$ and $-Z_{i+1}$ respectively to keep them from moving. The matrices for the system shown in figure 3.18 are then

$$M = \begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix}, \; K = \begin{pmatrix} (k_1 + k_2) & -k_2 \\ -k_2 & (k_2 + k_3) \end{pmatrix}, \; Z = \begin{pmatrix} (Z_1 + Z_2) & -Z_2 \\ -Z_2 & (Z_2 + Z_3) \end{pmatrix}.$$

We can write the equations of this system in matrix form as

$$M \begin{pmatrix} x_1'' \\ x_2'' \end{pmatrix} + K \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + Z \begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = 0$$

$$\begin{pmatrix} x_1'' \\ x_2'' \end{pmatrix} = -M^{-1} K \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - M^{-1} Z \begin{pmatrix} x_1' \\ x_2' \end{pmatrix}.$$

We can then write equation (3.24) as

$$\begin{pmatrix} x_1 \\ x_2 \\ x_1' \\ x_2' \end{pmatrix}' = \left(\begin{array}{c|c} \mathbf{0} & I \\ \hline -M^{-1}K & -M^{-1}Z \end{array}\right) \begin{pmatrix} x_1 \\ x_2 \\ x_1' \\ x_2' \end{pmatrix}.$$

This is the general form of matrix A for a state space mass-spring system. If there are n masses, A is an $2n \times 2n$ matrix, $\mathbf{0}$ is a $n \times n$ matrix of zeros, I is the $n \times n$ identity matrix, and $-M^{-1}K$ and $-M^{-1}Z$ are both $n \times n$ matrices.

For a mass-spring system the input $\mathbf{u}$ is an external force acting on each mass. If the system has no external force $\mathbf{u}$ is equal to zero. The matrix B is then the $n \times n$ identity matrix. Since we want the state vector $\mathbf{x}$ as output, the matrix C is the $n \times n$ identity matrix and $D = \mathbf{0}$.

The solution to the state space system is [2]

$$\mathbf{x}(t) = e^{tA}\mathbf{x}(0) + \int_0^t e^{t-\tau}B\mathbf{u}\, d\tau$$

$$\mathbf{y}(t) = C\left(e^{tA}\mathbf{x}(0) + \int_0^t e^{t-\tau}B\mathbf{u}\, d\tau\right) + D\mathbf{u}.$$

The matrix exponential, e^{tA} is defined as

$$e^{tA} = I + \frac{t}{1!}A + \frac{t^2}{2!}A^2 \ldots + \frac{t^k}{k!}A^k \ldots.$$

3.3.2 Mass-spring Systems with Multiple Masses using the Symplectic Euler Method

The poles of the analog system on the s-plane are the eigenvalues the system matrix (the matrix A described in section the previous section). For each mass in the mass-spring system, we have a conjugate pair of poles. We can view numerical methods as mapping for the s-plane to z-plane, so if any poles on the s-plane is mapped outside the unit circle on the z-plane, the system will be unstable. The poles on the s-plane have the abscissa of $\sigma = -\gamma/2$ and the ordinate of $\mu = (1/2)\sqrt{4\omega_0^2 - \gamma^2}$. We can find the region on the s-plane that maps to stable poles on the z-plane by using equation (3.22). In this case, however, we need it terms of μ and σ. First we find ω_0^2 in terms of μ and σ:

$$\begin{aligned} \mu &= \frac{1}{2}\sqrt{4\omega_0^2 - \gamma^2} \\ \mu^2 &= \frac{1}{4}(4\omega_0^2 - 4\sigma^2) \\ \omega_0^2 &= \mu^2 + \sigma^2. \end{aligned} \tag{3.25}$$

We then substitute equation (3.25) in equation (3.21)

$$\begin{aligned} \mu^2 + \sigma^2 &= \frac{1}{h^2}(4 - 2\gamma h) \\ \mu &= \pm\sqrt{\frac{1}{h^2}(4 + 4\sigma h) - \sigma^2}. \end{aligned} \tag{3.26}$$

The system is then stable if

$$-\sqrt{\frac{1}{h^2}(4+4\sigma h)-\sigma^2} \leq \mu \leq \sqrt{\frac{1}{h^2}(4+4\sigma h)-\sigma^2}.$$

Figure 3.19 shows a damped string made with 20 masses. The sampling frequency is $20,000$ samples per second, the spring stiffness coefficients, k, are all $18,497.2439^2$, the masses are all 1 and the damping coefficients are $1,000$. Since all the poles lie between the two roots of equation (3.26), this system is stable.

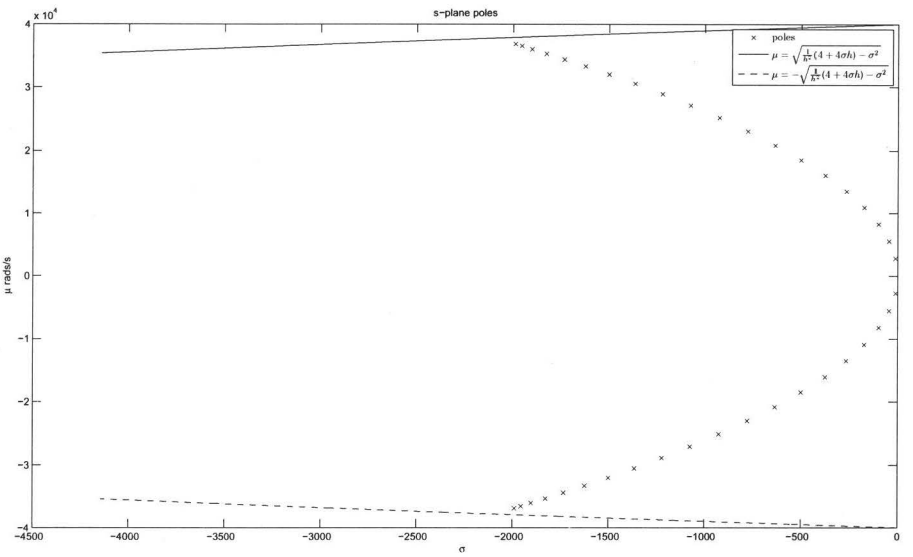

Figure 3.19: Poles of damped string on the s-plane

We can also solve equations (3.17) and (3.23) in terms of μ and σ. Equation (3.17) divides poles that have complex components from those that do not. Substituting equation (3.25) into ((3.17)) we get

$$\mu^2+\sigma^2 = \frac{2}{h^2}-\frac{\gamma}{h} \pm 2\sqrt{\frac{1}{h^4}-\frac{\gamma}{h^3}}$$

$$\mu = \pm\sqrt{-\sigma^2+\frac{2}{h^2}+\frac{2\sigma}{h} \pm 2\sqrt{\frac{1}{h^4}+\frac{2\sigma}{h^3}}}.$$

Similarly, solving equation (3.23)—the curve dividing vibrating poles from non-vibrating poles—for μ in terms of σ results in

$$\mu = \pm\sqrt{\frac{-(\sigma h)^2+2\sigma h+2}{h^2}}.$$

Using these equations, we can divide the left half on the s-plane into sections. These regions represent the qualitative properties that any pole on the s-plane within the region

will have when the systems is approximated with the symplectic Euler method. Figure 3.20 shows the regions of the s-plane for positive frequencies. The negative frequencies are mirror images of the positive ones. The sampling frequency used for this graph was $1,000$ samples per second, but the shape of the graph is the same for all sampling frequencies. Making the sampling frequency larger dilates all parts of the graph uniformly.

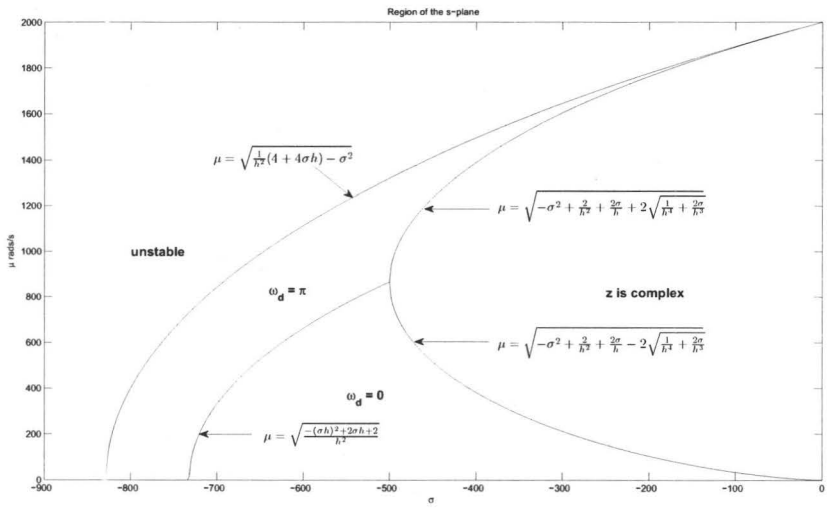

Figure 3.20: Regions of the s-plane

Figure 3.21 shows a damped string with 3 masses using a sampling rate of 1,000 samples per second. The highest frequency pole is very close to the region of instability. This means there is almost no damping in the symplectic Euler simulation of this system even though $\sigma = -307$ on the s-plane. The frequency on the s-plane is $1,634$ radians per second, but since it lies in the region that the symplectic Euler maps to π radians per sample, the actual frequency is $3,142$ radians per second — almost twice what it should be. The sound produced by this simulation in a sustained pitch of 500 cycles per second (since 3,142 radians = 500 cycles), which is approximately the G above middle C. The other frequencies are damped so quickly they are barely heard. Figure 3.22 shows the same simulation, but this time using a sample rate of $2,000$ samples per second. The poles stay in the same positions, but all the regions are expanded so all the poles are now well within the region where z is complex. The sound of this simulation is a short thud, as it should be because of the high damping.

Figure 3.23 shows the errors vectors for the symplectic Euler. The arrows show, for points on the s-plane, the direction and magnitude of the error that the symplectic Euler method will produce. For example, if the arrow points upward and to the left, it means the symplectic Euler will have excessive damping and a frequency that is too high. Notice that the lower right part of graph has small errors.

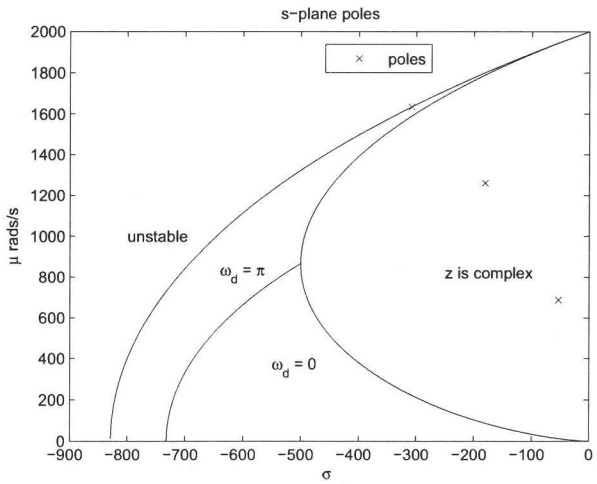

Figure 3.21: Damped string with 3 masses - sample rate is 1,000 samples per second

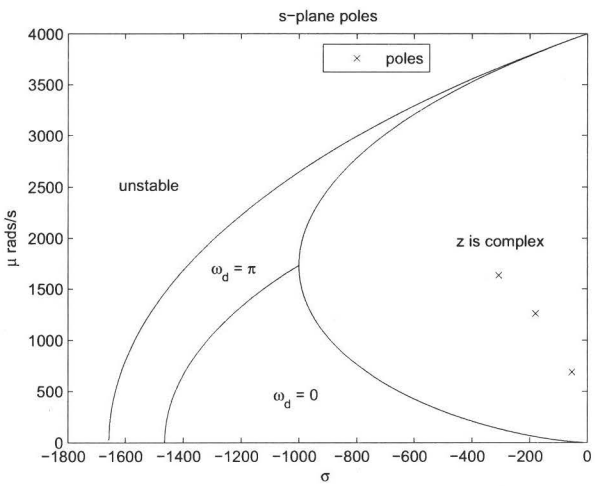

Figure 3.22: Damped string with 3 masses - sample rate is 2,000 samples per second

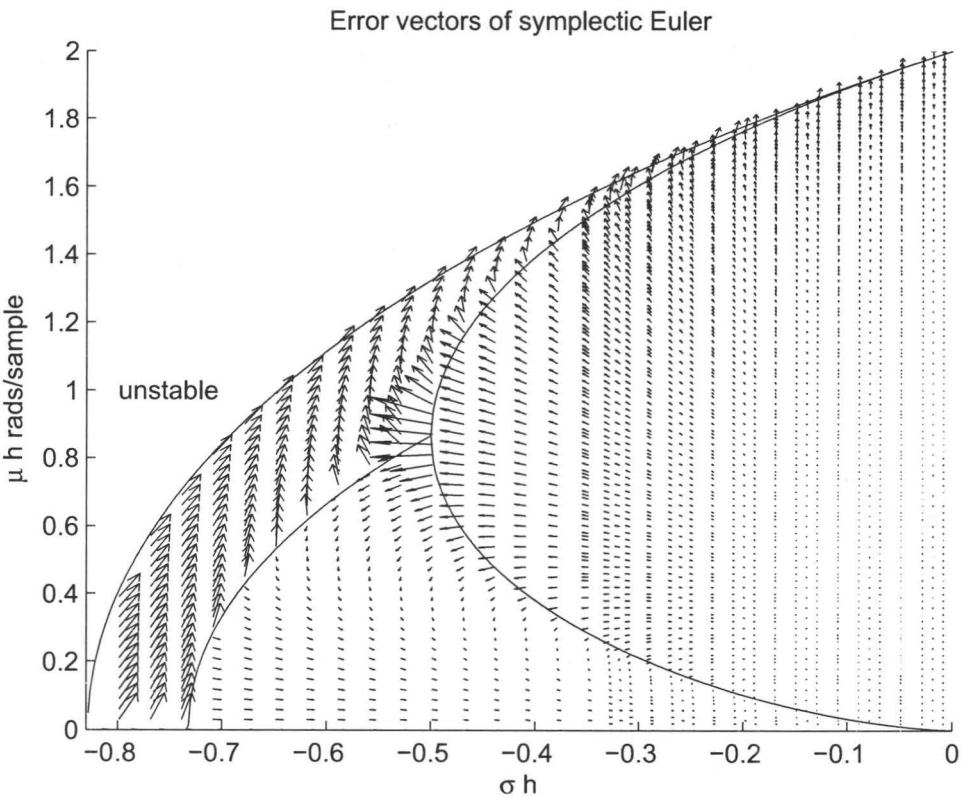

Figure 3.23: Error vectors of symplectic Euler

Chapter 4

Alternate Methods of Implementing Mass-Spring Systems

In this section we look at three other methods of implementing mass-spring systems. The first approach is to solve implicit methods by removing the delay-free loops. The second approach is to transform the mass-spring system into an equivalent electrical circuit and implement it as a Wave Digital Filter. The third approach is to avoid delay-free loops by using explicit numerical methods. We consider multi-stage methods and in particular symplectic methods—methods which conserve energy.

4.1 Implementing Implicit Numerical Methods

As we have seen in chapter 2, using the backward Euler method or the bilinear transform results in a stable system whenever the analog system is stable. This means that, if we implement a mass-spring system with one of these methods, it will be stable for any frequency and sample rate. However, both these methods are implicit, and implicit methods cannot be implemented directly. In this section we show how to implement implicit numerical methods.

4.1.1 Removing Delay-Free Loops

What are Delay-Free Loops?

Figure 4.1 shows a simple example of a delay-free loop. We can write an equation for $y(n)$ as:

$$y(n) = x(n) + o(n) = x(n) + P(y(n)).$$

This will create a problem in the implementation of this system since the calculation of y at time step n depends on $y(n)$, the very value we are trying to determine. Such systems are called *unrealizable*. A delay-free loop is then a loop whose output cannot be directly evaluated at time step n since it depends on its own value at that time step. It is often possible to replace a system with one or more delay-free loops with an equivalent system that does not contain any delay-free loops. For example, if the system P merely

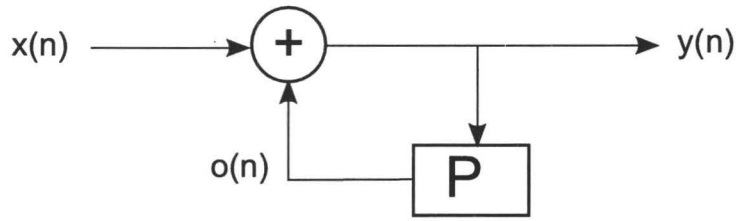

Figure 4.1: A simple system with delay-free loop [20]

multiplies $y(n)$ by a constant k, we can write:

$$y(n) = x(n) + P(y(n)) = x(n) + ky(n),$$
$$y(n) = \frac{1}{1-k}x(n).$$

This removes the delay-free loop, since the output $y(n)$ is now defined solely in terms of the input $x(n)$, which is known at the time we calculate $y(n)$.

In this section we look at a general method of eliminating delay-free loops and then show how this method can be used in mass-spring systems using the bilinear transform.

4.1.2 Eliminating Delay-Free Loops in Digital Filters

A method of eliminating delay-free loops in digital filters is presented by Harma [20]. The idea is to split the component contained in the delay-free loop into 2 parts: the *pure delay-free structure* and the output from the delay units. The *pure delay-free structure* is the part of the component that goes directly from input to output without any delays. This is the output the filter would give if all the delay units contained zeros. The rest of the output comes from the delay units and is therefore known at the current time step. This is the output the filter would give if the input at time n was 0.

Figure 4.2 shows a generic digital filter. The filter has L delay units for previous input values and M delay units for previous output values. The values $b_0 ... b_L$ are the coefficients for inputs $x(n) ... x(n-L)$. Similarly, the values $a_1 ... a_M$ are the coefficients for outputs $y(n-1) ... y(n-M)$. From the diagram it can be seen that the *pure delay-free structure*—the path directly from input to output—includes only b_0. The rest of the output comes from the delay units. The filter can be represented mathematically by:

$$y(n) = \sum_{i=0}^{L} (b_i x(n-i)) + \sum_{j=1}^{M} (a_j y(n-j)).$$

The *pure delay* is $b_0 x(n)$. The pure delay factor, b_0, is denoted as χ. The delayed portion is denoted as $o(n)$:

$$o(n) = \sum_{i=1}^{L} (b_i x(n-i)) + \sum_{j=1}^{M} (a_j y(n-j)).$$

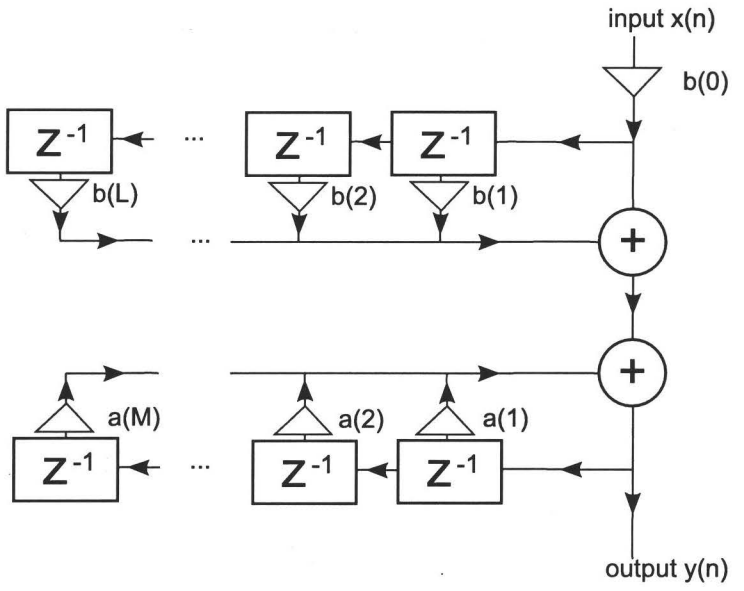

Figure 4.2: A generic digital filter

Thus

$$y(n) = \chi x(n) + o(n).$$

Figure 4.3 shows the delay-free loop we looked in the last section.

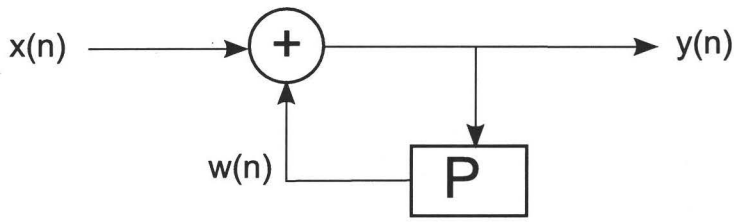

Figure 4.3: A simple system with delay-free loop [20]

The equation for this system is:

$$y(n) = w(n) + x(n) = x(n) + P(y(n)). \tag{4.1}$$

We add to P a function which returns $o(n)$ without making any changes to its state variables. The input to P is $y(n)$. We then have:

$$w(n) = P(y(n)) = \chi y(n) + o(n).$$

Substituting the right side into equation (4.1) gives us:

$$y(n) = x(n) + \chi y(n) + o(n)$$
$$y(n) = \frac{x(n) + o(n)}{1 - \chi}.$$

Once $y(n)$ has been calculated, P is updated.

Removing Delay-Free Loops in Mass-Spring Systems

We start by looking at the approximation of a mass using the bilinear transform. Using $F = ma$ as the equation for the mass and $s \rightarrow \frac{2}{h}\frac{1-z^{-1}}{1+z^{-1}}$ for the bilinear transform:

$$\begin{aligned}
\mathcal{L}\{F\} &= \mathcal{L}\{mx''\} \\
\hat{F}(s) &= ms^2\hat{X}(s) \\
\hat{H}(s) &= \frac{\hat{X}(s)}{\hat{F}(s)} = \frac{1}{ms^2} \\
\hat{H}(z) &= \frac{1}{m\left(\frac{2}{h}\frac{1-z^{-1}}{1+z^{-1}}\right)^2} = \frac{1}{m\left(\frac{4}{h^2}\frac{1-2z^{-1}+z^{-2}}{1+2z^{-1}+z^{-2}}\right)} \\
&= \frac{h^2+2h^2z^{-1}+h^2z^{-2}}{4m-8mz^{-1}+4mz^{-2}} = \frac{(\frac{1}{4m})(h^2+2h^2z^{-1}+h^2z^{-2})}{1-2z^{-1}+z^{-2}} = \frac{\hat{Y}(z)}{\hat{X}(z)} \\
y_n &= \frac{h^2}{4m}x_n + \frac{h^2}{2m}x_{n-1} + \frac{h^2}{4m}x_{n-2} + 2y_{n-1} - y_{n-2}.
\end{aligned}$$

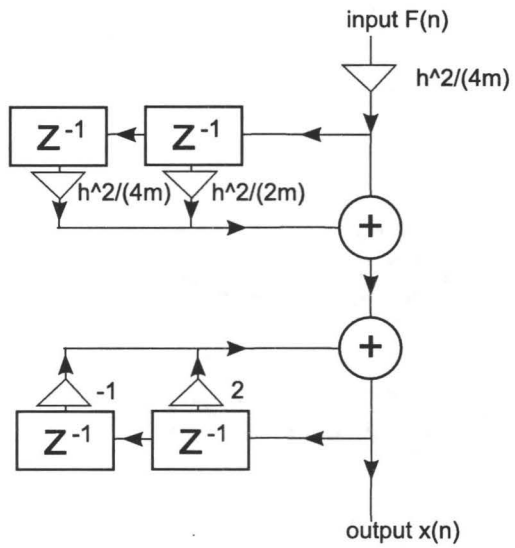

Figure 4.4: Signal flow diagram for mass using bilinear transform

Figure 4.4 shows the signal flow diagram for the mass. We can observe from the diagram that the pure delay part is

$$\chi = h^2/(4m). \tag{4.2}$$

Since the equation for the spring is $F = k(x_2 - x_1)$, which does not contain a derivative, the bilinear transform is simply

$$y_n = k(x_{2:n} - x_{1:n}),$$

where y_n is the spring force and x_1 and x_2 the displacement of the two ends of the spring from their equilibrium positions.

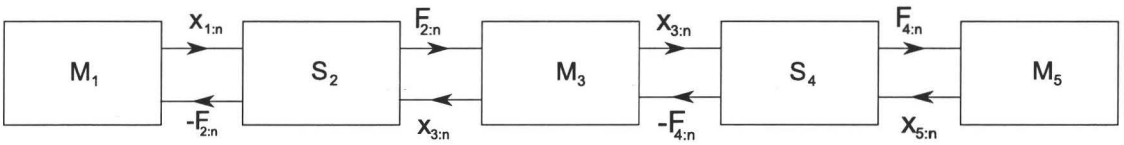

Figure 4.5: Mass spring system

For an example, we use the mass-spring system shown in figure 4.5. This is a system with 3 masses—M_1, M_2 and M_3—connected to 2 springs—S_1 and S_2. We designate the pure delay for element i as χ_i. So from equation (4.2), for each mass

$$\chi_i = h^2/(4m_i).$$

The output — the position — of each mass can be written as

$$x_i = \chi_i F_{i:n}^{tot} + o_{i:n},$$

where $F_{i:n}^{tot}$ is the total force acting on mass i at time n.

We denote the force of each spring acting to the right by $F_{i,a}$ and the force acting to left by $F_{i,b}$. So

$$F_{i,a} = -F_{i,b}.$$

We can now write the output for the 3 masses as:

$$\begin{aligned} x_{1:n} &= \chi_1 F_{2,a:n} + o_{1:n} \\ x_{3:n} &= \chi_3[F_{2,b:n} + F_{4,a:n}] + o_{3:n} \\ x_{5:n} &= \chi_5 F_{4,b:n} + o_{5:n}. \end{aligned}$$

We can then substitute these into the spring equations. For Spring S_2

$$\begin{aligned} F_{2,a} &= k_2(x_3 - x_1) \\ &= k_2\left(\chi_3[F_{2,b} + F_{4,a}] + o_3 - [\chi_1 F_{2,a} + o_1]\right), \\ F_{2,a}(1 + k_2\chi_3 + k_2\chi_1) &- F_{4,a}(k_2\chi_3) = k_2(o_3 - o_1). \end{aligned}$$

(We have dropped the argument n —i.e. x_1 for $x_{1:n}$— to simplify the notation.)

Similar calculations for spring S_4 result in

$$F_{4,a}(1 + k_2\chi_5 + k_2\chi_3) - F_{2,a}(k_2\chi_3) = k_2(o_5 - o_3).$$

We now have two equations and two unknowns —the forces $F_{2,a}$ and $F_{4,a}$. We can create a matrix equation in the form $\mathbf{Ax} = \mathbf{b}$.

$$\begin{pmatrix} 1 + k_2\chi_3 + k_2\chi_1 & -k_2\chi_3 \\ -k_2\chi_3 & 1 + k_2\chi_5 + k_2\chi_3 \end{pmatrix} \begin{pmatrix} F_{2,a} \\ F_{4,a} \end{pmatrix} = \begin{pmatrix} k_2(o_3 - o_1) \\ k_2(o_5 - o_3) \end{pmatrix}$$

Note that matrix $\mathbf{A}$ contains only constant terms —recall that $\chi_i = h^2/4m_i$. On the other hand, the vector $\mathbf{b}$ normally changes at each time step. This means that we can calculate the $\mathbf{LU}$ decomposition of $\mathbf{A}$ ($\mathbf{L}$ is a lower triangular matrix and $\mathbf{U}$ an

upper triangular matrix) once and then at each time step the forces, $\mathbf{x}$, can be quickly calculated using the upper and lower triangular matrices.

The calculations for dampers are a little more complicated than those of springs. The bilinear transform of the damper is

$$\begin{aligned} y_n =& (-2Z/h)x_{r:n} + (2Z/h)x_{r:n-1} - y_{n-1} \\ =& (-2Z/h)x_{a:n} + (2Z/h)x_{b:n} + (2Z/h)x_{a:n-1} \\ & - (2Z/h)x_{b:n-1} - y_{n-1}, \end{aligned}$$

where $x_r = x_a - x_b$, with x_a and x_b representing the positions of the two masses, M_a and M_b, connected to the damper. For the damper, y_n is the force and x_n the position. We can then write an equation for the $damper_i$ using $x_i = o_i + \chi_i F_i^{tot}$:

$$\begin{aligned} F_{i,a:n} =& (-2Z/h)(o_{a:n} + \chi_a F_{a:n}^{tot}) + (2Z/h)(o_{b:n} + \chi_b F_{b:n}^{tot}) \\ & + (2Z/h)x_{a:n-1} - (2Z/h)x_{b:n-1} - F_{i,a:n-1}. \end{aligned}$$

Since we know that F_i is one of the forces acting on the masses M_a and M_b we can separate F_a^{tot} and F_b^{tot} into:

$$\begin{aligned} F_a^{tot} &= F_{i,a} + F_a^{other} \\ F_b^{tot} &= F_{i,b} + F_b^{other} = -F_{i,a} + F_b^{other}, \end{aligned}$$

where F_a^{other} denotes all the forces acting on mass M_a except F_i, and likewise for F_b^{other}. Using this substitution and substituting $\alpha = (2Z/h)$ gives us

$$\begin{aligned} & F_{i,a:n}(1 + \alpha(\chi_a + \chi_b)) + \alpha\chi_a F_a^{other} - \alpha\chi_b F_b^{other} \\ & = \alpha(o_{b:n} - o_{a:n}) + \alpha(x_{a:n-1} - x_{b:n-1}) - F_{i,a:n-1}. \end{aligned}$$

We now look at an algorithm to calculate the forces for a general mass-spring system using the bilinear transform. We define a *connector* to be either a spring or a damper. Each mass can be connected to an arbitrary number of connectors, while each connector connects to exactly 2 masses, M_a and M_b. Each row in the matrix $\mathbf{A}$ represents one connector. So, if there are n connectors in the system, $\mathbf{A}$ is an $n \times n$ matrix. If the ith connector is a spring, the diagonal element in the ith row is

$$1 + k_i\chi_{M_a} + k_i\chi_{M_b}.$$

Here k_i is the spring stiffness for that spring, and χ_{M_a} and χ_{M_b} are the pure delay coefficients for the masses connected to each end of the spring.

If the connector is a damper the diagonal element is

$$1 + (2Z_i/h)(\chi_{M_a} + \chi_{M_b}),$$

where Z_i is the damping coefficient. By using $\alpha = k_i$ if the connector is a spring and $\alpha = (2Z_i/h)$ if it is a damper we get:

$$\mathbf{A}_{i,i} = 1 + \alpha(\chi_{M_a} + \chi_{M_b}).$$

Each entry, (i, j), in the matrix not on the diagonal represents the effect of connector j that is connected to the same mass that the connector for row i is connected to. If connector j is not connected to either of the masses that connector i connects, the (i, j) entry is 0. If both connectors are pulling the same way—i.e. mass w is connected to the left sides of both connectors i and j, or mass w is connected to the right sides of both connectors i and j, and $connector_j$ is a spring —we *add* $k_j\chi_{M_w}$ to the (i, j) element. If the connectors are pulling in opposite directions—i.e. mass w is connected to the left side of connector i and the right side of j, or vice versa— we *subtract* $k_j\chi_{M_w}$ from the i, j element. If the connector j is a damper, then $(2Z_j/h)\chi_{M_w}$ is either added or subtracted from the (i, j) element.

Vector $\mathbf{b}$ also has one entry for each connector. If the connector for the ith entry is a spring

$$\mathbf{b}_i = k_i(o_{a:n} - o_{b:n}).$$

If the connector for the ith entry is a damper

$$\mathbf{b}_i = (2Z_i/h)(o_{b:n} - o_{a:n}) + (2Z_i/h)(x_{a:n-1} - x_{b:n-1}) - F_{i,a:n-1}.$$

Here $F_{i,a:n-1}$ is the previous output of damper i.

Algorithm 1 (Create Matrix to Solve Implicit System)**.**

1. Create $n \times n$ matrix $\mathbf{A}$,where n is the number of connectors. Set all entries to 0.
2. For $i = 1$ to n
  3. If ($connector_i$ is a spring)
    4. $\alpha = k_i$
  5. Else ($connector_i$ is a damper)
    6. $\alpha = (2Z_i/h)$
  7. Endif
  8. Let M_a and M_b be the masses at either end of $connector_i$
  9. $\mathbf{A}_{i,i} = 1 + \alpha(\chi_{M_a} + \chi_{M_b})$
  10. For each $M_w \in \{M_a, M_b\}$
    11. For each other $connector_j$, $j \neq i$, connected to M_w
      12. If ($connector_j$ is a spring)
        13. $\beta = k_j$
      14. Else ($connector_j$ is a damper)
        15. $\beta = (2Z_j/h)$
      16. Endif
      17. Let M'_a and M'_b be the masses at either end of $connector_j$
      18. If (($M_w = M_a$) and ($M_a = M'_a$)) or (($M_w = M_b$) and ($M_b = M'_b$))
        19. $\mathbf{A}_{i,j} = \mathbf{A}_{i,j} + \beta\chi_{M_w}$
      20. Else
        21. $\mathbf{A}_{i,j} = \mathbf{A}_{i,j} - \beta\chi_{M_w}$
      22. Endif
    23. Endfor
  24. Endfor
25. Endfor

Once the matrix $\mathbf{A}$ is calculated, we find its $\mathbf{LU}$ decomposition. The $\mathbf{LU}$ decomposition decomposes the matrix into an upper triangular matrix and a lower triangular matrix. Then at each time step we calculate the vector $\mathbf{b}$. The new forces, $\mathbf{x}$, are found by solving

$$\mathbf{Ly} = \mathbf{b}, \text{ for } \mathbf{y} \text{ and}$$
$$\mathbf{Ux} = \mathbf{y}, \text{ for } \mathbf{x}.$$

Once we have updated the forces for the current time step we can calculate the new positions of the masses based on the forces of the current time step.

Algorithm 2 (Implicit Mass-Spring System)**.**

1. Matrix $\mathbf{A}$ = *Create_Matrix_to_Solve_Implicit_System()*
2. $[\mathbf{L}, \mathbf{U}] = \mathbf{LU}decomposition(A)$
3. For $n = 1$ *to numberOfSamples*
  4. For each connector i
    5. $\mathbf{b_i} = \begin{cases} k_i(o_{a:n} - o_{b:n}) & \textit{if spring} \\ (2Z_i/h)(o_{b:n} - o_{a:n} + x_{a:n-1} - x_{b:n-1}) & \\ \quad -F_{i,a:n-1} & \textit{if damper} \end{cases}$
  6. EndFor
  7. solve $\mathbf{Ly} = \mathbf{b}$, *for* $\mathbf{y}$ *and* $\mathbf{Ux} = \mathbf{y}$, *for* $\mathbf{x}$.
  8. For each connector $conn_i$
    9. $conn_i.Force(n) = \mathbf{x}_i$
  10. EndFor
  11. For each mass m_j
    12. $m_j.updatePosition(n)$
  13. EndFor
  14. writeSample(n)
15. EndFor

These algorithms are based on the bilinear transform, but could be easily adapted to other numerical methods.

There are, however, some drawbacks to this method:

1. If the mass-spring system is not static —i.e. if the configuration of the mass-spring system or any mass, spring stiffness or damping factor changes— the matrix $\mathbf{A}$ and its $\mathbf{LU}$ decomposition must be recalculated. For a continuously changing system, this would mean solving the matrix equation at each time step.

2. This method assumes the elements of the system are linear. If non-linear elements are introduced it will no longer work.

4.2 Converting Mass-Spring Systems to Wave Digital Filters

Wave Digital Filters (WDFs) were developed by electrical engineers to digitize analog filters. An important part of the design of WDFs was the elimination of delay-free loops. For every mechanical system composed of masses, springs and dampers, there is an equivalent electrical system composed of inductors, capacitors and resistors [12]. In this section, we first show the analogy between mass-spring systems and electrical circuits. We then look at converting mass-spring systems to their electrical equivalents and how to implement them as WDFs.

4.2.1 Introduction to Wave Digital Filters

An *N port* is a black box with N ports for interaction with other elements. Each port has 2 terminals; the port voltage is difference between them. Each port has an associated current and voltage.

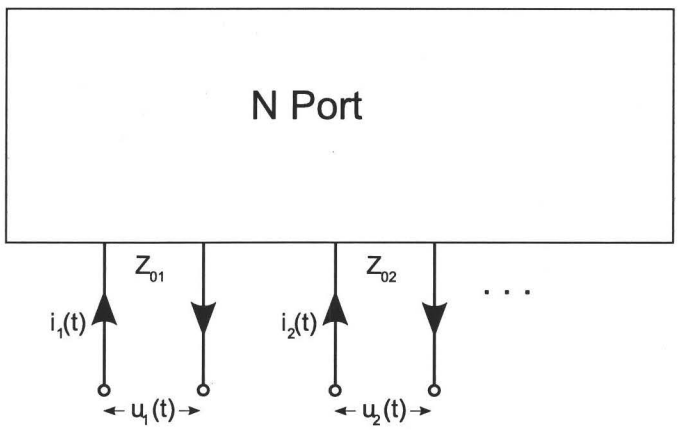

Figure 4.6: N Port

Figure 4.6 shows an N port. The voltage is denotes by $u(t)$, the current by $i(t)$ and the impedance by Z_0.

Wave digital filters use the concept of *wave variables*. Wave variable $a = u(t) + Z_0 i(t)$ represents the incoming wave and $b = u(t) - Z_0 i(t)$ represents the reflected wave. The reflectance of a port, $S(s)$, is the Laplace transform of the reflected wave divided by the Laplace transform of the incoming wave or:

$$S(s) = \frac{\mathcal{L}\{b\}}{\mathcal{L}\{a\}} = \frac{\mathcal{L}\{u(t) - Z_0 i(t)\}}{\mathcal{L}\{u(t) + Z_0 i(t)\}}$$
$$= \frac{\mathcal{L}\{\frac{u(t)}{i(t)} - Z_0\}}{\mathcal{L}\{\frac{u(t)}{i(t)} + Z_0\}} = \frac{\mathcal{L}\{Z(t) - Z_0\}}{\mathcal{L}\{Z(t) + Z_0\}} = \frac{Z(s) - Z_0}{Z(s) + Z_0}$$

One Port Elements

Next we look at some one port elements used in electronics.

- **Capacitor**

  The formula for capacitors is: $i(t) = C\frac{\partial u(t)}{\partial t}$, where C is the capacitance. Its Laplace transform is: $I(s) = \mathcal{L}\{i(t)\} = CsU(s)$. The impedance is defined as $U(s)/I(s)$, so the impedance of a capacitor, $Z_c(s)$, is $Z_c(s) = 1/C_s$. The reflectance, $S_C(s)$, is: $\frac{Z(s)-Z_0}{Z(s)+Z_0} = \frac{\frac{1}{Cs}-Z_0}{\frac{1}{Cs}+Z_0} = \frac{1-CsZ_0}{1+CsZ_0}$

- **Inductor**

  For inductors, we start from the formula $u(t) = L\frac{\partial i(t)}{\partial t}$, where L is the inductance. The Laplace transform is $U(s) = LsI(s)$ and $Z_L(s) = Ls$. The reflectance, $S_L(s)$, is: $\frac{s-Z_0/L}{s+Z_0/L}$

- **Resistor**

  For resistors the formula is $u(t) = Ri(t)$, where R is the resistance. We get $U(s) = RI(s)$ and $Z_R(s) = R$. The reflectance, $S_R(s)$, is: $\frac{1-Z_0/R}{1+Z_0/R}$

Bilinear transform

The next step is to use the bilinear transform to discretize the one port elements. The bilinear transform does the mapping $s \rightarrow \frac{2}{h}\frac{1-z^{-1}}{1+z^{-1}}$ where $z = e^{sh}$.

- **Inductor**

  To discretize the inductor with the bilinear transform we use the mapping

  $$S_L(s) \rightarrow S_L\left(\frac{2}{h}\frac{1-z^{-1}}{1+z^{-1}}\right).$$

  Since $S_L(s) = \frac{s-Z_0/L}{s+Z_0/L}$ the transform is:

  $$\frac{\frac{2}{h}\frac{1-z^{-1}}{1+z^{-1}} - Z_0/L}{\frac{2}{h}\frac{1-z^{-1}}{1+z^{-1}} + Z_0/L} = \frac{2/h - Z_0/L - (2/h + Z_0/L)z^{-1}}{2/h + Z_0/L + (2/h - Z_0/L)z^{-1}}.$$

  Z_0 is an arbitrary parameter, so we can choose any value. If we choose $Z_0 = 2L/h$ this simplifies to $-z^{-1}$.

- **Capacitor**

  Similarly, performing the bilinear transform and choosing $Z_0 = h/(2C)$ gives $S_C = z^{-1}$

- **Resistor**

  For the resistor $Z_0 = R$ gives $S_R = 0$

Connecting Elements

Units that connect elements are called *adaptors*. We can connect elements either in series or in parallel. Using Kirchhoff's current law, we can represent a parallel adaptor as [6]

$$b_k = -a_k + \frac{2}{\sum_{j=1}^{M} G_j}\sum_{j=1}^{M} a_j G_j, \tag{4.3}$$

where $G = 1/Z_0$. Note that we have each reflected wave, b_k, defined in terms of a_j, the incoming waves.

Similarly, series adaptors are represented by the equation [6]:

$$b_k = a_k - \frac{2Z_{0k}}{\sum_{j=1}^{M} Z_{0j}} \sum_{j=1}^{M} a_j. \tag{4.4}$$

More information on wave digital filters can be found in [41, 6, 38].

4.2.2 The Analogy between Mass-Spring Systems and Electrical Circuits

The equations for the inductor, capacitor and resistor using u for voltage and i for current are:

- *Inductor*

$$u(t) = L\frac{di(t)}{dt} \tag{4.5}$$

- *Capacitor*

$$i(t) = C\frac{du(t)}{dt}; \quad u(t) = \frac{1}{C}\int_0^t i(t)dt + u(0) \tag{4.6}$$

- *Resistor*

$$u(t) = i(t)R \tag{4.7}$$

The equation for a mass is:

$$F(t) = ma(t) = m\frac{dv(t)}{dt}, \tag{4.8}$$

where $v(t)$ is the velocity.

If we compare equation (4.8) with equation (4.5) we see that they have the same form. If we substitute $u(t)$ for $F(t)$ —voltage for force— and $i(t)$ for $v(t)$ —current for velocity— the equations are identical if $C = m$.

The equation for the spring is:

$$F(t) = kx(t) = k\int_0^t v(t)dt + F(0). \tag{4.9}$$

Comparing equations (4.9) and (4.6) and again substituting voltage for force and current for velocity, we see that they are the same if $k = 1/C$.

The equation for the damper is:

$$F(t) = Zv(t). \tag{4.10}$$

Comparing equations (4.10) and (4.7) and again substituting voltage for force and current for velocity, we see that they are the same if $Z = R$.

The analogies between the mass-spring system and an electrical circuit are summarize in the table 4.1.

Mass-spring	Electrical
force $F(t)$	voltage $u(t)$
velocity $v(t)$	current $i(t)$
mass	inductor $L = m$
spring	capacitor $C = 1/k$
damper	resistor $R = Z$

Table 4.1: Analogy between Mass-spring system and electrical circuit

4.2.3 Converting Mass-Spring systems to Equivalent Electrical Circuits

We start by looking at the example mass-spring system shown in figure 4.7.

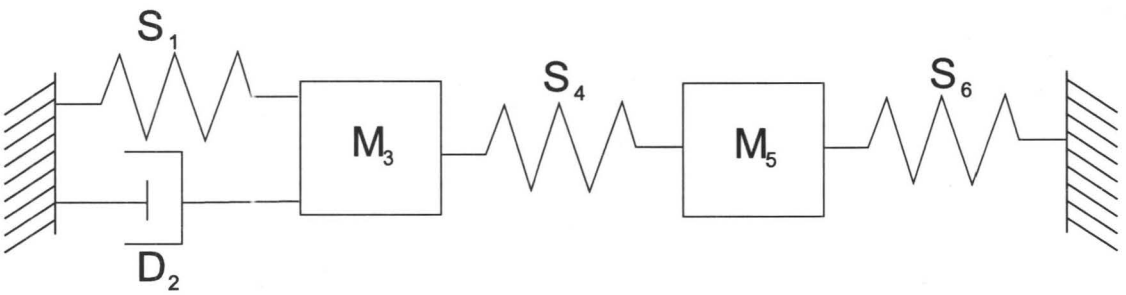

Figure 4.7: Mass spring system

The first thing to notice is that there are two masses, M_3 and M_5. Each mass must have exactly one velocity, and the velocities of the 2 masses are independent. Since the velocity in the mass-spring system is equivalent to the current in the electrical circuit, we must have 2 independent currents in the equivalent circuit. This means we need 2 loops in the circuit, one for M_3 and one for M_5.

Next, notice that the position of mass M_3, the length of spring S_1 and the length of damper D_2 are constrained to be equal at all times. This means their velocities are always identical. Therefore the current in the equivalent electrical circuit is the same for these elements, requiring that they be connected in series on the first loop. The same applies to mass M_5 and spring D_6 which are connected in series on the second loop.

The length of spring D_4 is the position of M_5 minus the position of M_3. So the velocity of D_4 is the velocity of M_5 minus the velocity of M_3. This means D_4 must be on the intersection of the 2 loops. Since the forces at each end of the spring must be equal and opposite, the forces across D_4, the first loop and the second loop are all the same. This means the voltage drop in the equivalent circuit is the same so D_4 must be connected in parallel with the two loops.

The resulting electrical circuit is shown in figure 4.8.

In general, each mass requires a separate loop on the electrical circuit. A spring or a damper that has one fixed end is connected in series to the mass at the free end. A spring or damper with a free mass at each end needs to be connected in parallel between the two loops for two masses.

We then convert the circuit into a WDF. This is shown in figure 4.9. The blocks in

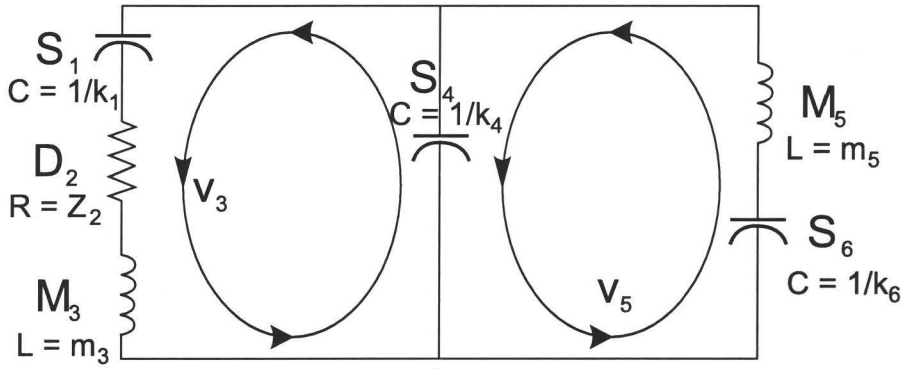

Figure 4.8: Equivalent electrical circuit

the diagram are called adaptors. An adaptor with a single vertical line represents a series connection, while two vertical lines represent a parallel connection. The first adaptor connects S_1, D_2 and M_3 in series. This adaptor has 4 ports, each with an input a_i and an output b_i. The elements—the capacitor, the inductor and the resistor—are implemented using the techniques discussed in section 4.2.1. This adaptor is connected in parallel to D_4 and the series connection of M_5 and D_6.

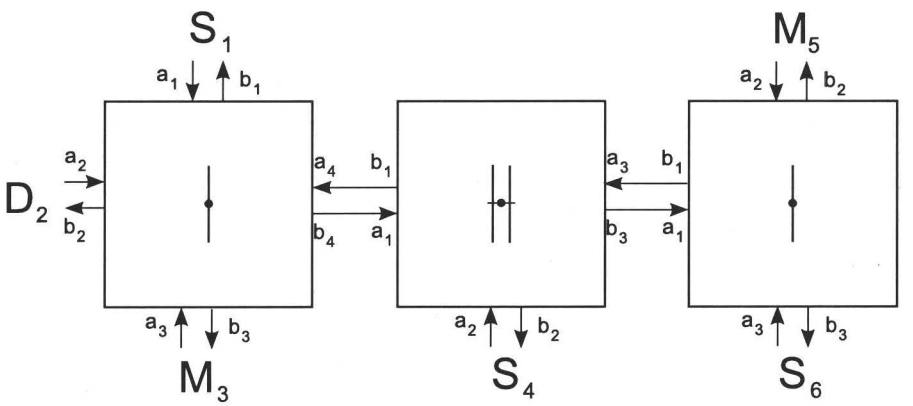

Figure 4.9: Wave digital filter for circuit

4.2.4 Removing Delay-Free Loops from Wave Digital Filters

Recall from section 4.2.1 that the reflectance of the capacitor and the inductor contain a delay. As well, the reflectance of the resistor is always zero. This means that there is no problem in calculating the output of each element if we are only dealing with a single adaptor. But looking at Figure 4.9, we appear to have a problem when adaptors are interconnected, since the output of one adaptor depends on the input of the other adaptor and vice versa. This results in a delay-free loop between adaptors.

The delay-free loop between adaptors can be eliminated by using a *reflection-free port* [14]. The port that connects two adaptors has an associated port impedance, Z_0. This is a free parameter. Suppose port q is the port on a series adaptor that connects

another adaptor. We can set the impedance, Z_{0q}, so that it equals the sum of the other port impedances on the adaptor:

$$Z_{0q} = \sum_{j=1, j\neq q}^{M} Z_{0j}.$$

Note that

$$\sum_{j=1}^{M} Z_{0j} = \sum_{j=1, j\neq q}^{M} Z_{0j} + Z_{0q} = 2Z_{0q}. \tag{4.11}$$

Using equation (4.4) and substituting the left side of equation (4.11) for $2Z_{0q}$, we can write b_q as

$$\begin{aligned} b_q &= a_q - \frac{2Z_{0q}}{\sum_{j=1}^{M} Z_{0j}} \sum_{j=1}^{M} a_j = a_q - \frac{\sum_{j=1}^{M} Z_{0j}}{\sum_{j=1}^{M} Z_{0j}} \sum_{j=1}^{M} a_j \\ &= a_q - \sum_{j=1}^{M} a_j = - \sum_{j=1, j\neq q}^{M} a_j. \end{aligned}$$

This means we can calculate the output of port q without knowing the input, a_q. This is called a *reflection-free port*.

We can use the same method on a parallel adaptor. In this case we set the port conductance, G_q, to the sum of the other port conductances:

$$G_q = \sum_{j=1, j\neq q}^{M} G_j.$$

The output for port q can then be calculated as

$$b_q = \frac{2}{\sum_{j=1}^{M} G_j} \sum_{j=1, j\neq q}^{M} G_j a_j.$$

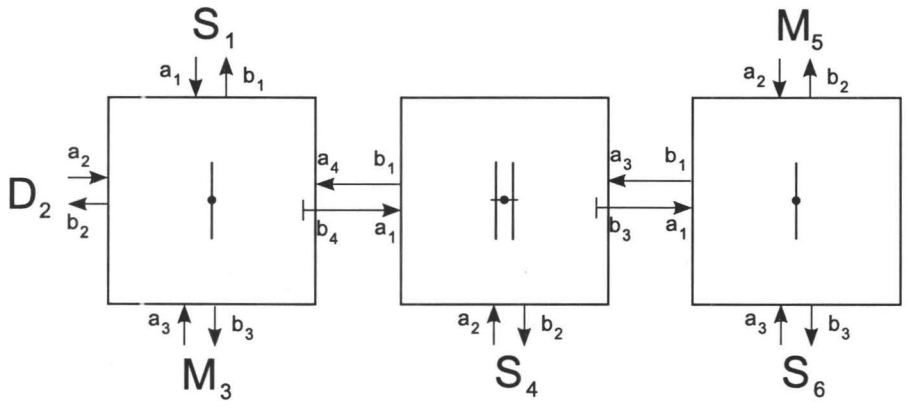

Figure 4.10: Wave digital filter with refection-free ports

Figure 4.10 shows our example circuit with the reflection-free ports indicated by "caps" on the output arrow. We have made b_4 on the left adaptor and b_3 on the middle adaptor reflection-free.

Even with refection free ports, we still need to be careful of the order in which the new output calculations are made. Examining equation (4.4), we see that we can not immediately update the left adaptor in our example, since the ports 1, 2 and 3 depend input a_4 which is not known. We can update the refection-free port (port 4) on the left adaptor since it depends on a_1, a_2 and a_3, which are all known since they contain delays. Once b_4 on the left adaptor is updated, a_1 on the middle adaptor is known. This allows us to updated port 3 on the middle adaptor, which depends on a_1 and a_2 which are now both known. After b_3 on the middle adaptor is updated, a_1 on the right adaptor is known. Since all the inputs for the right adaptor are now known, we can update all the elements in it including b_1. This allows us to update the rest of the middle adaptor and then the left adaptor.

Converting mass-spring systems to wave digital filters allow us to eliminate delay-free loops. But they add a considerable amount of complexity since we need do the conversion, create refection-free ports, and make sure the updates are done in the right order.

4.3 Multi-stage Numerical Methods

We have seen in the previous section that although the symplectic Euler method used in mass-spring systems has no numerical damping, it does have frequency warping and does become unstable. The Euler method is a first order method, meaning that its error at each step (local truncation error) is order(n^2) and the overall error (global truncation error) order(n). We next look at more accurate methods.

4.3.1 Runge-Kutta Methods

Runge-Kutta methods are a family of methods used to solve ordinary differential equations. They may be either implicit or explicit, and work by sampling the derivative at several points in the interval between time n and time $n + 1$. These methods are called multi-stage methods. The most popular of these—usually just referred to as *the* Runge-Kutta method—has four stages and is explicit. The derivatives are found at four points in the interval—k_1 to k_4—and then these four derivatives are averaged to get an estimate of the derivate for the $n + 1$ step. For the differential equation $\mathbf{x}'(t) = \mathbf{f}(t, x)$, the function $\mathbf{f}(t_n, \mathbf{x}(n))$ determines the value of the derivative at time step n using the time t_n and the vector $\mathbf{x}(n)$, which consists of the position x_n and the velocity v_n. The fourth

order Runge-Kutta method is then [7]:

$$\begin{aligned}
\mathbf{k_1} &= \mathbf{f}(t_n, \mathbf{x}_n) \\
\mathbf{k_2} &= \mathbf{f}(t_n + (1/2)h, \mathbf{x}_n + (1/2)h\mathbf{k_1}) \\
\mathbf{k_3} &= \mathbf{f}(t_n + (1/2)h, \mathbf{x}_n + (1/2)h\mathbf{k_2}) \\
\mathbf{k_4} &= \mathbf{f}(t_n + h, \mathbf{x}_n + h\mathbf{k_3}) \\
\mathbf{x}_{n+1} &= \mathbf{x}_n + h\frac{\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4}}{6}.
\end{aligned}$$

The method has a local truncation error of order h^5 and a global truncation error of order h^4 [7]. This should make it much more accurate than the Euler method.

There are, however, some complications in using the Runge-Kutta method on mass-spring systems. The Runge-Kutta method assumes we can calculate the derivative using the function $\mathbf{f}(t_n, \mathbf{x}_n)$ at any point. This is not straightforward in a mass-spring system. Recall equation (1.4) for the mass element:

$$\begin{pmatrix} x(t) \\ v(t) \end{pmatrix}' = \begin{pmatrix} v(t) \\ F(t)/m_i \end{pmatrix}.$$

Here $F(t)$ represents the total forces acting on the mass at time step t. Calculating $\mathbf{k}_1$ is not a problem, since the forces at time step n have already been calculated. But, calculating $\mathbf{k_2}$ is a problem, since we need the forces when $x = x_n + (1/2)hk_{1,1}$ and $v = v_n + (1/2)hk_{1,2}$, where $k_{1,1}$ and $k_{1,2}$ are $\mathbf{k_1}$'s components corresponding to the position and velocity respectively. The force for spring S_j, connected to mass M_i, is based on the positions of the masses at either end of the spring, so we need to know not just the position of M_i but the position of the mass M_k connected to M_i by spring S_j. Since a mass can be connected to an arbitrary number of springs, we need to know the positions of all these masses to calculate the force acting on mass M_i. The situation is similar for the damper, except that it needs the velocities of the 2 masses instead of the positions. Since each of these masses may be connected to other masses, we need to know the positions and velocities of all the masses at $x = x_n + (1/2)hk_{1,1}$ and $v = v_n + (1/2)hk_{1,2}$ to compute the forces acting on each mass. After computing all the forces, we can finally compute the derivative $\mathbf{k_2}$.

This means, for each stage, $s = 1 \ldots 4$, we must first calculate the positions of all the masses and then use these positions to calculate the forces of all the springs and dampers. After this we can calculate $\mathbf{k_s}$ for each mass. After stage 4 is completed, we can calculate the new positions and velocities of all the masses and finally, the new forces of all the springs and dampers. This means that instead of one loop through each of the masses, springs and dampers as in the Euler method, we need 5 loops through each of the components. This means that the Runge-Kutta method will take about 5 times longer than the Euler method.

4.3.2 Symplectic Numerical Methods

Some numerical methods, such as the backward Euler method, can cause numerical damping—damping caused by the numerical methods that is not contained in the equation itself. This has proved to be a problem, especially in fields such as molecular and

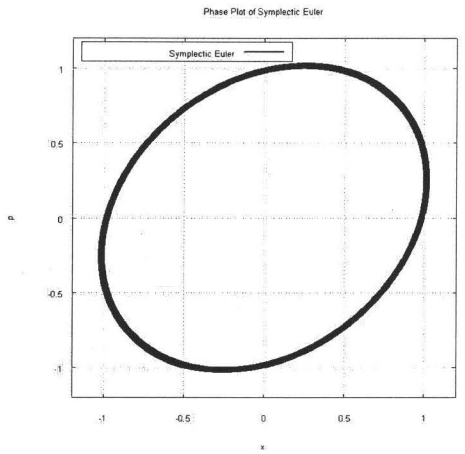

Figure 4.11: Phase plane of symplectic Euler

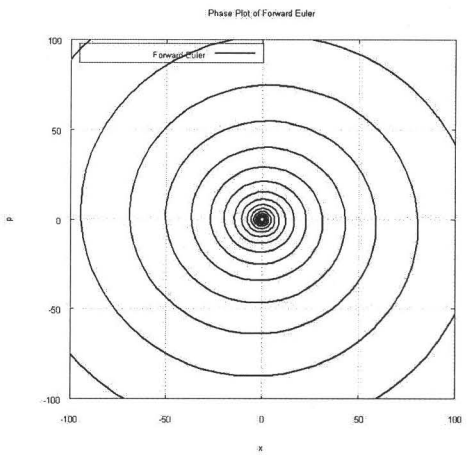

Figure 4.12: Phase plane of forward Euler

planetary simulation. In such systems conservation of energy over long periods of time is very important. Although all real systems that produce sound have some damping, many are very lightly damped, so a numerical method that preserves energy is important in physical sound synthesis. A numerical method is called symplectic if it preserves area on the position/momentum phase plane. Among other properties, symplectic systems conserve energy.

The Euler-Cromer method, also called the symplectic Euler, is a first order symplectic method defined as [19]:

$$\begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ v_n \end{pmatrix} + h \begin{pmatrix} v_{n+1} \\ a_n \end{pmatrix}. \tag{4.12}$$

If we replace a with F/m, we see that this equation is the same as (3.1).

Next, we plot the energy of the mass-spring system. The total energy of a mass-spring system is the sum of the kinetic energy of the masses, $K = \frac{1}{2}mv^2$, and the potential energy stored in the springs, $U = \frac{1}{2}kx^2$. Momentum is mass times velocity so the total energy is

$$E = \frac{1}{2}mv^2 + \frac{1}{2}kx^2 = \frac{1}{2}pv + \frac{1}{2}kx^2 = \frac{1}{2m}p^2 + \frac{1}{2}kx^2.$$

Setting $k = 1$ and $m = 1$, we get

$$E = \frac{1}{2}p^2 + \frac{1}{2}x^2; \quad \left(\sqrt{2E}\right)^2 = p^2 + x^2.$$

This is a circle of radius $\sqrt{2E}$. If the starting conditions are $x = 1$ and $v = 0$, the potential energy is $\frac{1}{2}kx^2 = \frac{1}{2}$ and the kinetic energy is 0. The radius is then $\left(\sqrt{2 \times \frac{1}{2}}\right)^2 = 1$. The phase plot, therefore, should be a unit circle.

Figure 4.11 shows a plot of the position/momentum phase plane of the symplectic Euler. The plot covers a thousand oscillations. This would be 1 second of sound at

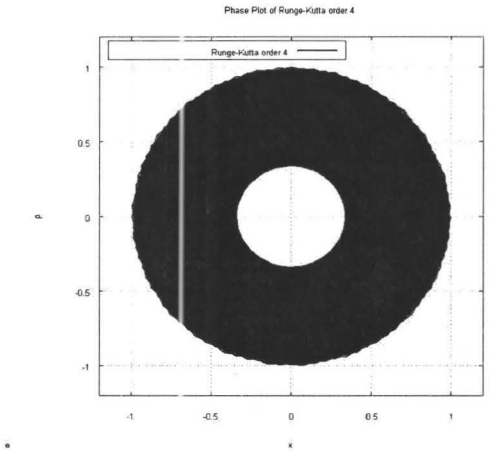

Figure 4.13: Phase plane of fourth order Runge-Kutta

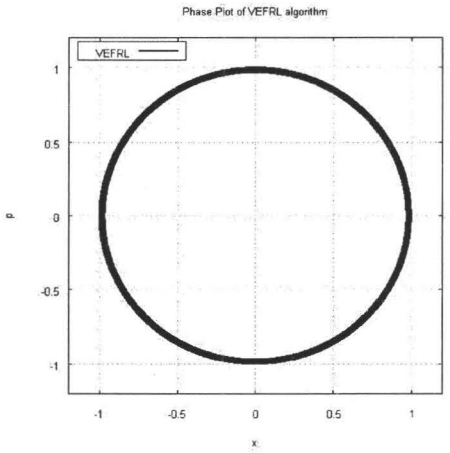

Figure 4.14: Phase plane of the VEFRL algorithm

1000 Hz or just .1 seconds at 10,000 Hz, which is still within the range of hearing. So it is important that a numerical method be able to maintain relatively constant energy over this many cycles. From figure 4.11 we can notice that the shape is somewhat elliptical due to inaccuracies of the first order method. But, the symplectic Euler is able to maintain the energy over a thousands cycles.

In contrast, figure 4.12 shows the phase plot of the forward Euler. The energy of the forward Euler increases without bound.

Next we try a higher order algorithm—the fourth order Runge-Kutta—which is not symplectic. The results are shown in figure 4.13. The more accurate Runge-Kutta method results in a circle, not an ellipse. The energy, however, does not stay constant, which can be seen by the fact the the graph slowly spirals inwards toward the center of the circle.

Recently, several higher order symplectic methods have been developed. One of these methods is called the VEFRL algorithm, which is a fourth order method defined

as [31]

$$
\begin{aligned}
v_1 &= v(t) + \frac{1}{m} f[r(t)]\xi h \\
r_1 &= r(t) + v_1(1 - 2\lambda)h/2 \\
v_2 &= v_1 + \frac{1}{m} f[r_1]\chi h \\
r_2 &= r_1 + v_2\lambda h \\
v_3 &= v_2 + \frac{1}{m} f[r_2](1 - 2(\chi + \xi))h \\
r_3 &= r_2 + v_3\lambda h \\
v_4 &= v_3 + \frac{1}{m} f[r_3]\chi h \\
r(t+h) &= r_3 + v_4(1 - 2\lambda)h/2 \\
v(t+h) &= v_4 + \frac{1}{m} f[r(t+h)]\xi h,
\end{aligned} \tag{4.13}
$$

where r is the position, v the velocity, m the mass, f the force. The constants are:

$$
\begin{aligned}
\xi &= +0.1644986515575760E + 00 \\
\lambda &= -0.2094333910398989E - 01 \\
\chi &= +0.1235692651138917E + 01.
\end{aligned}
$$

The phase plot for this algorithm is shown in figure 4.14. This plot shows that the algorithm is both accurate—since it plots a circle—and symplectic—since it maintains a constant energy over a thousand cycles.

This algorithm can be implemented similarly to the Runge-Kutta, with each stage requiring an update of the velocities and positions of all the masses and then the force of each spring and damper is calculated based on the new velocities and positions. The last step has an additional problem, since for mass-spring systems the force calculation is a function of both the position and the velocity (since viscous damping is a function of the velocity). This makes the last equation in (4.13) implicit. A simple solution is to calculate the positions of all the masses for $r(t+h)$ and calculate an estimated velocity for time $t+h$ as $v_{est} = v(t) + (f(t)/m)h$; then, update all the forces of the springs and dampers using the new positions and estimated velocities; now we can calculate the velocity at time $t+h$ using $f(r(t+h), v_{est})$; finally we use $r(t+h)$ and $v(t+h)$ to calculate the force of the springs and dampers at time $t+h$.

Algorithm 3 (VEFRL for Mass-Spring System)**.**

1. For stage s = 1 to 4
  2. For each mass
    3. calculate position and velocity for stage s according to equation (4.13).
  4. end for
  5. For each spring or damper
    6. calculate the force for stage s.

7. end for
8. end for
9. For each mass
10. calculate the position of the mass for time step $n + 1$.
11. calculate estimated velocity $v_{est} = v_n + (f(n)/m)h$.
12. end for
13. For each spring or damper
14. calculate force using positions at time step $n + 1$ *and estimated velocities.*
15. end for
16. For each mass
17. calculate velocity of the mass for time step $n + 1$ *using forces from step 14.*
18. end for
19. For each spring or damper
20. calculate force at time step $n + 1$ *using positions and velocities at time step* $n + 1$.
21. end for

Chapter 5

Simulating a String using a Mass-Spring System

In this chapter we look at the stability and accuracy of mass-springs systems when used to simulate a vibrating string. We simulate a vibrating string using 20 masses and compare the analytical solution described in section 3.3.1 with the results obtained from three numerical methods: the symplectic Euler, the fourth order Runge-Kutta and the VEFRL algorithm. This comparison is also presented in [30].

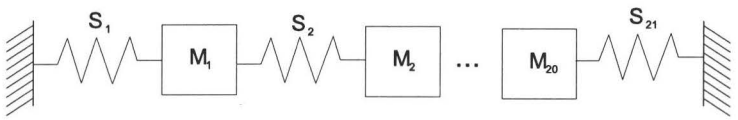

Figure 5.1: String created from 20 masses

5.1 Simulating a Vibrating String with 20 Masses

In this section, we simulate a string by connecting springs between 20 masses as shown in figure 5.1. Each of the masses and the spring stiffness coefficients are set to the same value. The constants were set as follows:

$$k_i = 18,497.2439^2 = 342,148,031.9, \quad m_j = 1.0, \; sample\_rate = 44,100\ Hz.$$

The initial conditions were that the $6th$ mass was displaced by one unit and all the other masses had a displacement of zero.

Figure 5.2 compares the analytical solution of this system with that of a mass-spring system using the symplectic Euler. The first 200 samples are shown using the displacement of mass 1. Notice that at the start the symplectic Euler is fairly close to the analytical solution, but that it quickly diverges from it. Notice also that the frequency of the Symplectic Euler is a little higher than that of the analytical solution. This is consistent with the frequency warping noted in section 3.1.

Figure 5.3 shows the same system comparing the fourth order Runge-Kutta and the VEFRL algorithm with the analytical solution. For the first 200 samples both methods

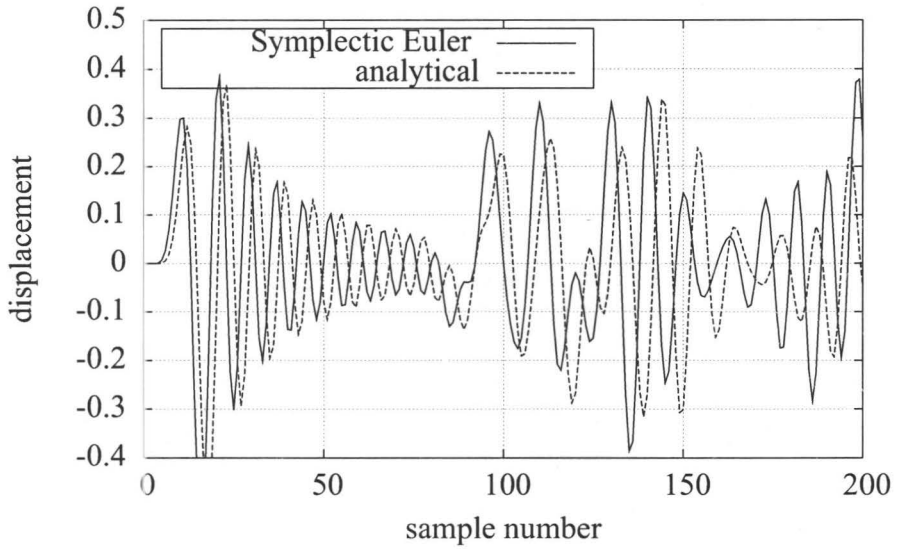

Figure 5.2: Simulated string comparing symplectic Euler with analytical solution

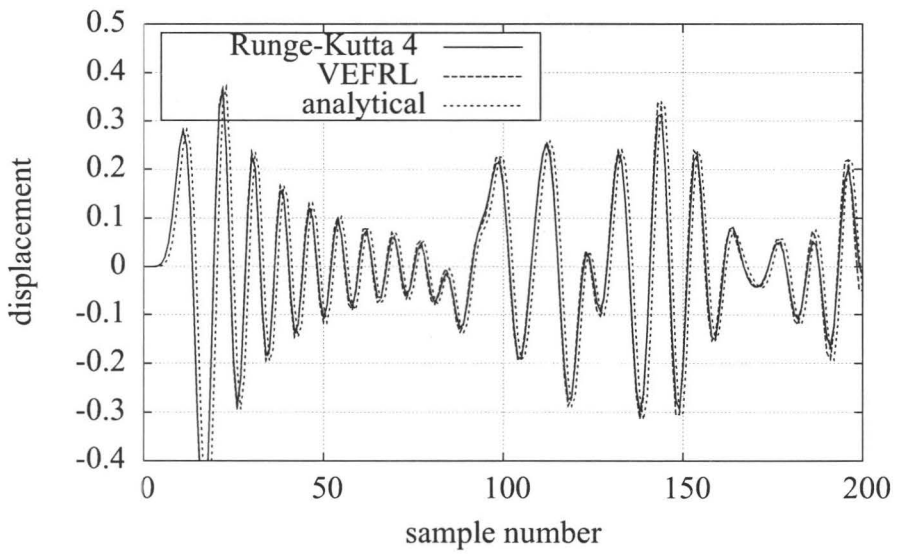

Figure 5.3: Simulated string comparing Runge-Kutta and the VEFRL with the analytical solution

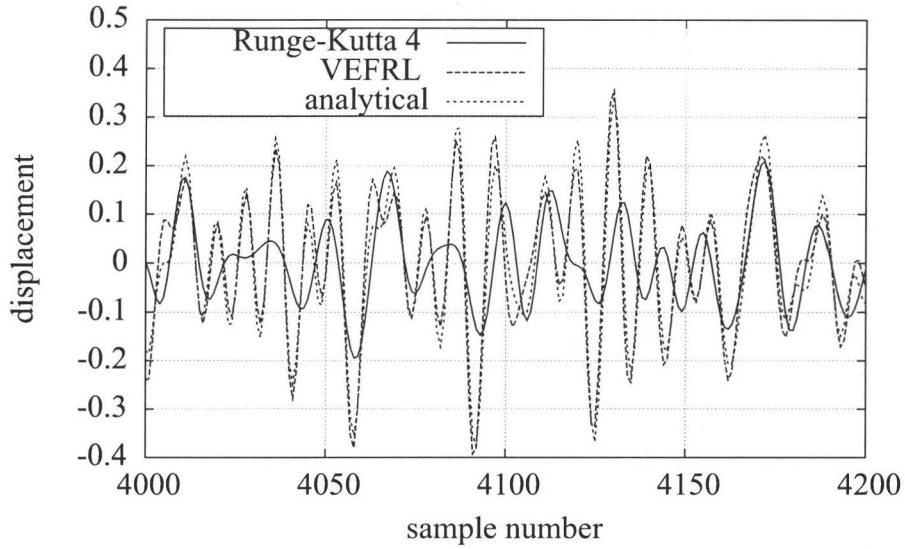

Figure 5.4: Simulated string comparing Runge-Kutta and VEFRL with the analytical solution after 4,000 samples

match the analytical solution almost perfectly. The situation, however, changes over time. Figure 5.4 shows the same simulation after $4,000$ samples. The VEFRL algorithm is still quite close to the analytical solution, but the Runge-Kutta is showing the effects of numerical damping.

What is the aural effect of these different methods? The numerical damping of the Runge-Kutta method is clearly audible. But, the symplectic Euler and the VEFRL are virtually indistinguishable. To understand why this is the case, we consider a continuous string. The frequencies that make up the continuous string are all integer multiples of the fundamental frequency [4]. For example, if the fundamental frequency is 440 Hz, the other frequencies (or modes) that make up the sound will be 880 Hz, 1320 Hz etc. In a string simulated with lumped masses, if the all the masses and spring constants are the same, the frequencies are [17]

$$\omega_n = 2\omega_0 \sin\left(\frac{n\pi}{2(N+1)}\right), \tag{5.1}$$

where ω_n is the nth frequency, $\omega_0 = \sqrt{k/m}$, and N is the number of masses. We can solve this equation for ω_0 with $n = 1$ (the fundamental frequency)

$$\omega_0 = \frac{\omega_1}{2\sin\left(\frac{\pi}{2(N+1)}\right)}. \tag{5.2}$$

This allows us to calculate ω_0 for a given frequency and number of masses. So, for 20 masses and a fundamental frequency of 440 Hz, ω_1 is $440 \times 2\pi$ and ω_0 is $18,497.2439$ (the value used in the simulation). We can now calculate the frequencies for our simulated string. The second frequency is 877.54 Hz as opposed to 880 Hz for the continuous string. The third frequency is 1310.17 Hz as opposed to 1320 Hz. The twentieth frequency is $5,822.09$ Hz as opposed to $8,800$ Hz. We can see that the low frequencies are

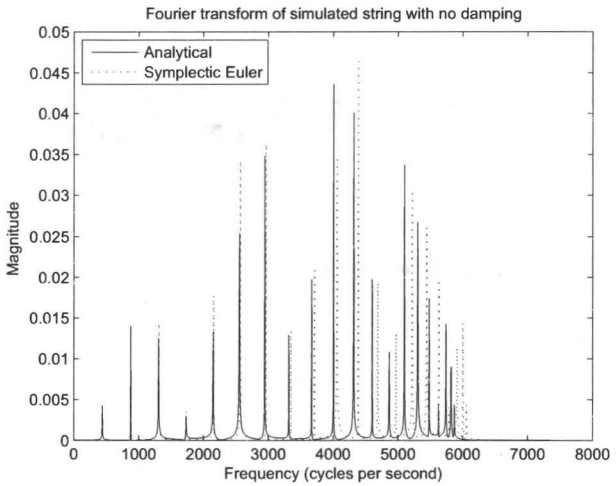

Figure 5.5: Fourier transform of undamped simulated string

close to those of the continuous string, while the higher frequencies are quite inaccurate. All the frequencies of the mass-spring approximation are *lower* than those of the continuous string. When we use the symplectic Euler method these frequencies become warped according to equation (3.5) with $h = 1/44100$. The fundamental is warped from 440 Hz to 440.07 Hz, the second frequency from 877.54 Hz to 878.11 Hz and the third from 1310.17 Hz to 1312.08 Hz. The change of frequency in the fundamental is almost imperceptible. The other frequencies are shifted by a small amount upwards—toward the frequencies that a continuous string would have. So if anything, the symplectic Euler should sound a little more like a continuous string than the more accurate methods.

Computing the Fourier transform of the output of the simulated string produces results that are consistent with the frequency warping of equation (3.5). Figure 5.5 shows the Fourier transform of the analytical solution and the symplectic Euler simulation. Each spike in the graph represents the frequency of one of the modes. We used 8192 "buckets", so, using a sample rate of $44,100$ Hz, each bucket is about 5.4 Hz wide. This means the results should be accurate within 2.7 Hz. As expected, the low frequency modes of the symplectic Euler are very close to those of the analytical solution. As the frequency becomes higher the frequency of the symplectic Euler is noticeably higher than that of the analytical solution. For the 20th mode, the spike for the analytical solution occurs at 5873 Hz, which is close to the expected 5971 Hz. The spike for the 20th mode in symplectic Euler simulation is at 6056 Hz, which is close to the calculated 6058 Hz.

If a lower sampling rate is used, the frequency warping of the symplectic Euler becomes more noticeable. For example, at $8,000$ samples per second, the symplectic Euler warps the fundamental frequency, 440 Hz, to 442.19 Hz, which is noticeably higher. However, since the VEFRL method takes around 6 times longer to run, it is much more efficient to use the symplectic Euler and compensate for the frequency warping.

We can do this by using the inverse of equation (3.5), which comes out to

$$w_a = h^{-1}\sqrt{\frac{2\left(\alpha+1+\sqrt{\alpha+1}\right)}{\alpha+1}}, \tag{5.3}$$

$$\tag{5.4}$$

where $\alpha = \tan^2(h\omega_d)$, ω_a is the analog frequency and ω_d the actual frequency resulting from the symplectic Euler method. So if we set w_d to the desired frequency (i.e. $440 \times 2\pi$) we get back the frequency we should use in equation (5.2) to calculate ω_0. This will correct the fundamental frequency so that it is exactly 440 Hz.

The more masses used in the simulated string, the more accurate the simulation becomes. We note in section 3.1 that the symplectic Euler becomes unstable at $1/\pi$ times the sampling frequency. This puts a limitation on the number of masses we can use to simulate a string vibrating at frequency ω and sample rate f_s. Consider for example, the simulation of a string vibrating at 440 Hz and using a sample rate of $44,100$ Hz. The maximum frequency that the symplectic Euler can have at this sample rate is $44,100/\pi = 14,037.5$ Hz. If we list the highest frequencies by using equation (5.1) with $n = N$, where N is the number of masses, we find that at 44 masses the highest frequency is $13,957$ Hz and with 45 masses the highest frequency is $14,045$ Hz. This tells us that the maximum number of masses we can use for a string vibrating at 440 Hz with a sample rate of $44,100$ is 44. If we want to use more masses we have to go to a higher sampling rate. The Runge-Kutta and VEFRL can go up to near the Nyquist limit, $f_s/2$, before becoming unstable. Even if a numerical method can go above the Nyquist limit without becoming unstable, aliasing will occur, resulting in an inaccurate sound. It is computationally more efficient to use the symplectic Euler at a higher sample rate, than it is to use the VEFRL algorithm. At some point, though, the sampling rate of the sound card will be exceeded. At that point the results of the simulation must be put through a low pass filter to remove the frequencies above Nyquist limit, and then downsampled to the maximum sample rate of the sound card.

5.2 Simulating a Damped String

Next we simulate a string using 20 masses with a damper and a spring between each consecutive pairs of masses. All masses, springs and dampers are set to the same values. The constants were set as follows:

$$k_i = 18,497.2439^2 = 342148031.9,\ m_i = 1.0,\ Z_j = 5.0,\ sample\_rate = 44,100\ Hz.$$

The initial conditions were that mass 6 had a displacement of 1, while the other masses had no displacement. The initial velocities were all zero. The displacement of mass 1 is used as the output. Figure 5.6 compares the analytical solution of this system with the symplectic Euler simulation. As with the undamped string, the two result start off very similar but quickly diverge.

Figure 5.7 shows the same simulation, this time comparing the fourth order Runge-Kutta and the VEFRL algorithm with the analytical solution. For the first 200 samples

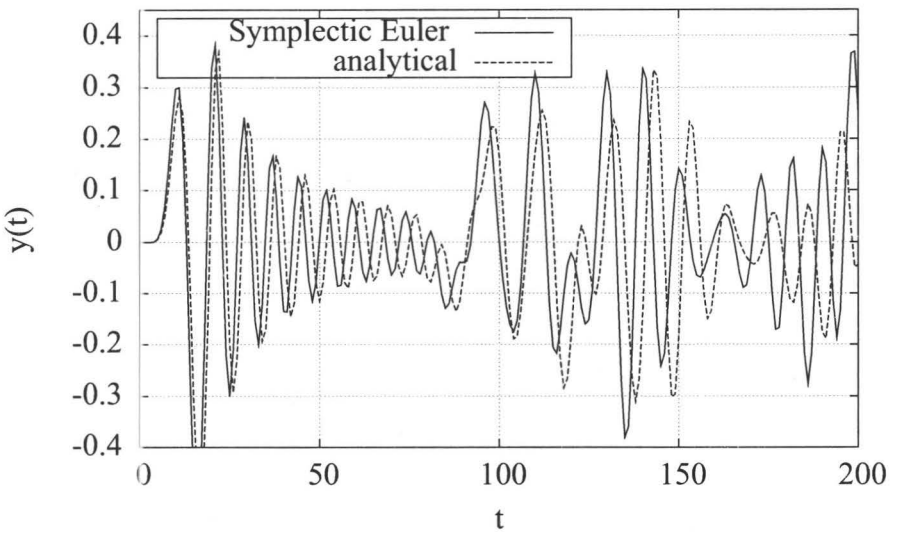

Figure 5.6: Simulated damped string comparing symplectic Euler with analytical solution

that are show in this figure, the Runge-Kutta and the VEFRL match the analytical solution almost exactly. Figure 5.8 shows the same simulation after $4,000$ samples. While the VEFRL algorithm has become a little inaccurate, it is still much more accurate than the Runge-Kutta at this point. The sound output of the damped string gives a much more realistic sound than the undamped string. But, as in the undamped string, the sound produced by the symplectic Euler and that of the VEFRL method are virtually indistinguishable. The extra damping of the Runge-Kutta can be noticed, but the difference is not as great as for the undamped string.

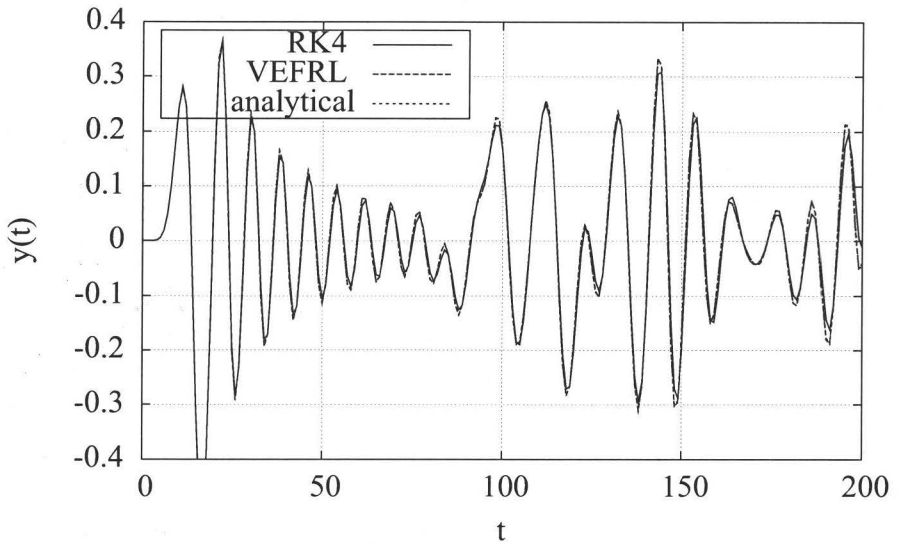

Figure 5.7: Simulated damped string comparing symplectic RK4 and VEFRL with analytical solution

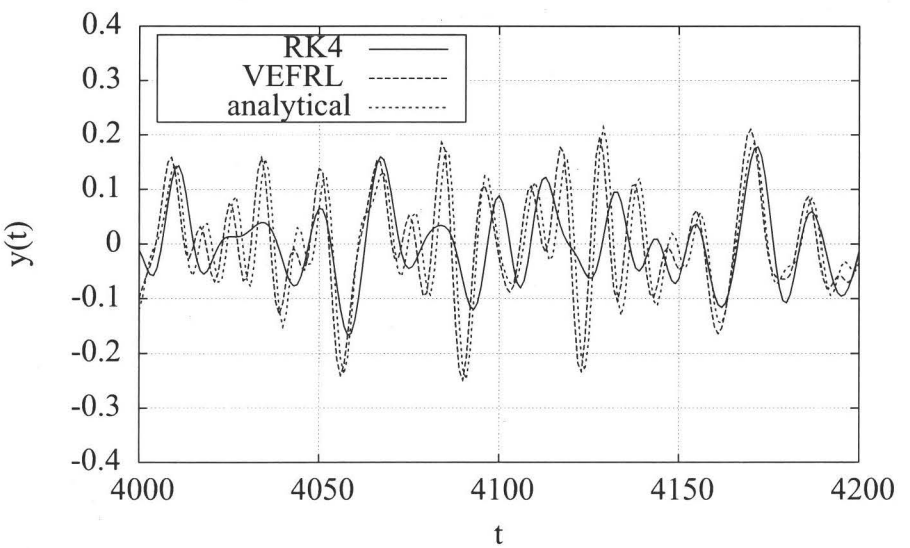

Figure 5.8: Simulated damped string after 4,000 samples

Chapter 6

Conclusions and Future Work

6.1 Conclusions

We have addressed the issues of stability and accuracy of mass-spring systems in sound synthesis. We find that, using the symplectic Euler method (the standard method used in implementing mass-spring systems), undamped mass-spring systems are stable up to frequencies of $1/\pi$ times the sample rate. When simulating a vibrating string, the highest mode of string, therefore, must be less than $1/\pi$ times the sample rate for the simulation to be stable. Damped mass-spring systems become unstable at a lower frequencies and large damping values may cause even low frequency vibration to become unstable. The general condition for stability is

$$\omega_0 \leq \frac{1}{h}\sqrt{4 - 2\gamma h},$$

where ω_0^2 is the spring stiffness coefficient divide by the mass, γ is the damping coefficient divided by the mass, and h is the time step.

We find that, for undamped systems, the symplectic Euler method has no numerical damping, but that it is not accurate in frequency, warping frequencies upward. For large damping values the symplectic Euler warps low frequency vibrations downward and high frequency vibrations upward. We compare the symplectic Euler method with two higher-order methods: the fourth order Runge Kutta, and the VEFRL algorithm, a fourth order symplectic algorithm. We find that the Runge-Kutta method has numerical damping which makes it become increasingly inaccurate in long lasting sounds. The VEFRL algorithm, on the other hand, conserves energy, and is much more accurate than either the Symplectic Euler or the Runge-Kutta. We find, however, that perceptually—the way the simulation sounds—there is little or no difference between the symplectic Euler and the VEFRL algorithm for the simulation of a vibrating string. This is because, when a small number of masses are used, the difference between the mathematical model used by the mass-spring system and an actual vibrating string is much greater than the difference in accuracy of the two numerical methods. If a large number of masses are used the model becomes much more accurate, but a high sample rate is required to avoid aliasing, and so both methods are quite accurate in the frequency range of human hearing.

Figure 6.1 shows a model space. We are trying, ultimately, to model the actual physical system (marked with and x), but to do this we first create a mathematical model

of the system (marked with an m). The distance between the models and the physical system on this figure represents the accuracy of the model. To realize the mathematical model, we use a numerical method (marked with an n). So the numerical method is approximating the mathematical model and only indirectly the actual physical system. We must remember that the continuous model of the string, based on the wave equation, is also a mathematical model. Since the string is actually made up of molecules, with enough masses, the lumped model should be more accurate than the continuous model. But our simulations use only a small number of masses, and so the mathematical model is not very accurate. So, even though the VEFRL method is a much better approximation of the mathematical model than the symplectic Euler, it does not follow that it will be a more accurate model of the physical system. In our case, it turns out that the inaccuracies of the symplectic Euler actually make it more accurate, in some aspects, with respect to the physical system.

We have also looked at using the bilinear transform to implement mass-spring systems. The bilinear transform has the advantage that for any stable system on the s-plane, the bilinear transform approximation of the system results in a stable system on the z-plane. As well, the bilinear transform is has no numerical damping for undamped systems. The disadvantage of this method is that it is implicit, and requires that a system of equations be solved at each time step. Wave digital filters also use the bilinear transform and mass-spring systems can be converted to wave digital filters. If a large number of masses is used to simulate a vibrating string, the symplectic Euler and the VEFRL algorithm require very high sampling rates to remain stable and avoid aliasing, whereas the bilinear transform does not.

In summary, we can divide applications using mass-spring systems into 3 categories:

1. If the system being simulated is lumped system or a close approximation of a lumped system, and a high degree of accuracy is need without high sampling rates, the VEFRL algorithm should be used to implement the mass-spring system.

2. If the simulation requires a large number of masses without using a very high sampling rate, the bilinear transform or a wave digital filter should be used to implement the mass-spring system.

3. In most other cases, the symplectic Euler method, due to its simplicity and efficiency, should be used to implement the mass-spring system.

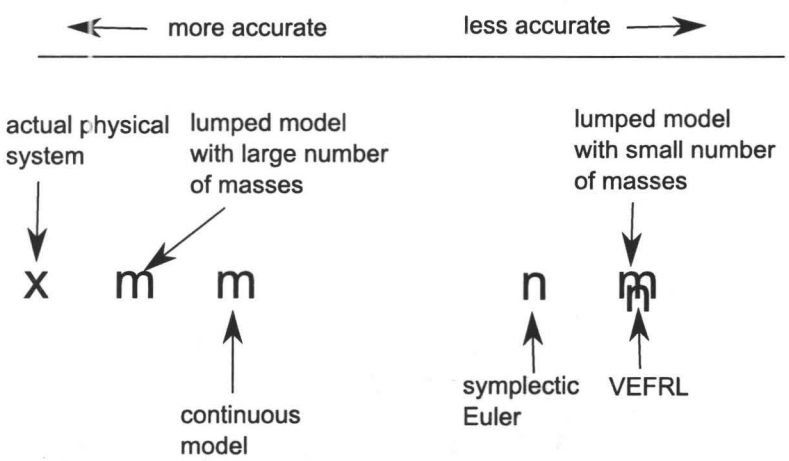

Figure 6.1: Model space

6.2 Future Work

- **Numerical Analysis**

  The analysis of the symplectic Euler presented in this thesis looks at stability, frequency warping and numerical damping. One other parameter of vibration that we have not considered is the phase. We have not determined whether the symplectic Euler alters the phase of the vibration.

- **Efficiency**

  How efficient can mass-spring systems be? We would like to run systems with hundreds or thousands of elements in real time. The system we have built is written in Java and can run simulations with around 50 masses in real time. What gains in efficiency could be made by writing the system in optimized C code? Mass-spring systems are inherently parallel, so it would be interesting to see if there are parallel computing techniques that could improve the efficiency of these systems and what hardware would be required for this. We note that our limit of 50 masses in real time is roughly consistent with the CORDIS system, which ran a real time simulation of a bowed string on a Silicon Graphics, Inc. (SGI) workstation in 2002, using 25 to 60 masses [15].

- **Acoustical Analysis**

  We have looked at the simulation of strings in this thesis. Mass-spring systems can also be used to simulate vibrating systems with any geometry: plates, shells, beams etc. How accurate are the mass-spring models of these geometries? We have done a simulation of a very simple model of a guitar. The complete model of a guitar, requires not just the simulation of the strings, but the body of the instrument. Very little of the sound produced by a stringed instrument comes from the strings themselves; most of the sound is produced by the instrument body [35]. Little research has been done on using mass-spring systems to attempt to accurately model musical instruments. Theoretically, by using enough masses,

mass-spring systems could model vibration patterns of specific types of wood. Whether this is practical on today's computers is an interesting research question.

- **User Interface**

  Playing a virtual musical instrument requires a large number of parameters. Specifying these parameters is an important issue in physical sound synthesis. For real time instruments the user may produce the input parameters physically by using *transducers*. The transducers convert physical motion produced by the user into digital signals that can be used as the parameters for the virtual instrument. This area has already been studied at ACROE using the CORDIS system [11], however a simulation of a guitar using a mass-spring system with input from two data gloves does not seem to have been attempted yet.

- **Using more than 1 Dimension of Vibration**

  In all of the mass-spring models we looked at, the vibration was in one dimension only. Even though the model of the instrument can be considered to have points in two or three dimension, these co-ordinates only affect how the sound propagates through the instrument. For true two dimensional vibration, we have to consider both transverse and longitudinal vibrations (see figure 6.2). If the mass is displaced along the x axis, the natural frequency of the longitudinal vibration is [36]

  $$f_l = \frac{1}{2\pi}\sqrt{\frac{2k}{m}},$$

  where each spring has a spring stiffness of k and m is the mass. The transverse vibration, however, is more complex. If the mass is displaced along the y axis, the force acting in the y direction is [36]

  $$F_y \approx -2ky\left(1 - \frac{l_0}{l}\right) - \frac{kl_0}{a^3}y^3,$$

  where l_0 is the equilibrium length of the spring and l is the current length. We see that the force contains a cubic term, which results in a nonlinear differential equation. This would likely make the analysis of mass-spring systems with 2 or more dimensions much more difficult than for those with only one dimension.

  It should be noted that real strings exhibit non-linear vibration [40]. Mass spring system would seem to be a promising approach to studying nonlinear behaviour in real strings and other nonlinear aspects of the sound production of musical instruments. This would require a modification to the algorithms of the mass-spring system to change the positions, velocities and forces from scalar to vector quantities.

- **Combining Mass-spring Systems with other Sound Synthesis Methods**

  In addition to simulating a musical instrument, we might also want to simulate the environment surrounding the musical instrument to produce reverberation. But to do this solely using a mass-spring system would involve a very large number of masses (i.e. we would use masses, springs and dampers to simulate the air and

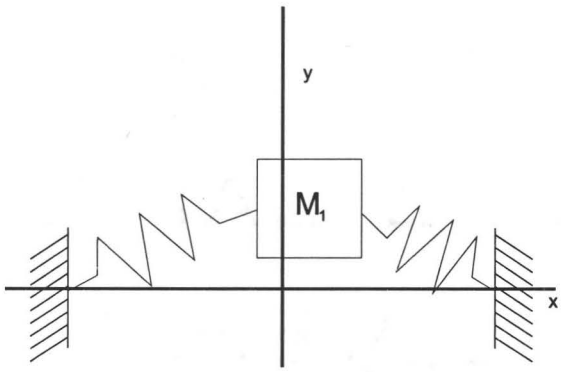

Figure 6.2: Two dimensional mass-spring system

other materials in the room). A more practical approach might be to use digital wave guides to simulate the environment around the instrument, since, unless the system is massively parallel, digital waveguides are much more efficient than mass-spring systems. Thus in this simulation, the mass-spring system would produce the sound of the instrument, and the digital waveguide would produce the reverberation of the room.

- **Converting Mass-spring systems to Wave Digital Filters**

  We have seen in section 5 that in a simulated string, the more masses used, the more accurate the simulation is. We noted that, for a given fundamental frequency, the more masses used, the greater the sampling rate required to keep the system stable. We also noted that if the bilinear transform was used instead of the symplectic Euler, the system would be stable at any sampling rate. The bilinear transform is problematic to implement, since it is an implicit numerical method and would require solving a system of equations at each time step. However, wave digital filters also use the bilinear transform and so are also stable at any sampling rate. To implement complex mass-spring systems using wave digital filters would require an algorithm to covert the mass-spring system into a wave digital filter. This might be an efficient method of implementing mass-spring systems using a large number of masses.

Bibliography

[1] Jean-Marie Adrien. The Missing Link: Modal Synthesis. In Giovanni DePoli, Aldo Piccialli, and Curtis Roads, editors, *Representations of Musical Signals*. MIT Press., Cambridge, MA, 1991.

[2] J. Dwight Aplevich. *The Essentials of Linear State Space Systems.* John Wiley & Sons Inc., 2000.

[3] Federico Avanzini. *Computational Issues in Physically-based Sound Models*. PhD thesis, Universit'a degli Studi di Padova Dipartimento di Elettronica ed Informatica, 2001.

[4] John Backus. *The Acousitcal Foundations of Music.* W.W. Norton and Company Inc., 1969.

[5] Mario Benedicty and Frank R. Sledge. *Discrete Mathematical Structures.* Harcourt Brace Janovich, Publishers, 1987.

[6] Stefan Bilbao. *Wave and Scattering Methods for Numerical Simulation.* John Wiley and Sons Ltd., 2004.

[7] William E. Boyce and Richard C. DiPrima. *Elementary Differential Equations and Boundary Value Problems.* John Wiley and Sons, Inc., 2001.

[8] C. Cadoz, A. Luciani, and J.L. Florens. Responsive input devices and sound synthesis by simulation of instrumental mechanisms: the CORDIS system. *The Computer Music Journal*, 8(3):60–73, 1983.

[9] Claude Cadoz and Jean Loup Florens. The Physical Model: Modeling and Simulating the Instrumental Universe. In Giovanni DePoli, Aldo Piccialli, and Curtis Roads, editors, *Representations of Musical Signals*. MIT Press., Cambridge, MA, 1991.

[10] Claude Cadoz, Annie Luciani, and Jean Loup Florens. CORDIS-ANIMA: A Modeling and Simulation System for Sound and Image Synthesis: The General Formalism. *Computer Music Journal*, 17(1):19–29, 1993.

[11] Claude Cadoz, Annie Luciani, Jean-Loup Florens, and Nicolas Castagné. Artistic Creation and Computer Interactive Multisensory Simulation Force Feedback Gesture Transducers. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression (NIME-03), Montreal, Canada*, 2003.

[12] Robert H. Cannon. *Dynamics of Physical Systems.* McGraw-Hill Book Company, 1967.

[13] Giovanni DePoli and Davide Rocchesso. Physically based sound modelling. *Organised Sound*, 3(1):61–76, 1998.

[14] Alfred Fettweis. Wave Digital Filters: Theory and Practice. *Proceedings of the IEEE*, 74(2):270–327, Feb. 1986.

[15] Jean Loup Florens. Real time Bowed String Synthesis with Force Feedback Gesture Interaction. *Invited paper. Forum Acousticum.*, 2002.

[16] Gene F. Franklin and J. David Powell. *Digital Control of Dynamic Systems.* Addison Wesley Publishing Company, 1980.

[17] A.P. French. *Vibrations and Waves.* W.W. Norton and Company, 1971.

[18] Joachim Georgii and Rüdiger Westermann. Mass-Spring Systems on the GPU. *Simulation Modelling Practice and Theory*, 13(8):693–702, 2005.

[19] Harvey Gould, Jan Tobochnik, and Wolfgang Christian. *An Introduction to Computer Simulation Methods.* Addison Wesley, 2007.

[20] A. Harma. Implementation of frequency-warped recursive filters. *Signal Processing*, 80:543–548(6), March 2000.

[21] Michael R. Hatch. *Vibrations Simulation Using MATLAB and ANSYS.* Chapman and Hall/CRC, 2001.

[22] David M. Howard, Stuart Rimell, and Andy D. Hunt. Force Feedback Gesture Controlled Physical Modelling Synthesis. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression (NIME-03), Montreal, Canada*, 2003.

[23] J. L. Kelly and C. C. Lochbaum. Speech Synthesis. In *Proceeding of the Fourth International Congress on Acoustics*, volume 4, pages 1–4, Copenhagen, June 1962.

[24] Peter Kraniauskas. *Transforms in Signals and Systems.* Addison Wesley Publishing Company, 1992.

[25] Hiller L. and Ruiz P. Synthesizing musical sounds by solving the wave equation for vibrating objects: Part 1. *Journal of the Audio Engineering Society*, 19(6):462–470, 1971.

[26] Edwards A. Lee and Pravin Varaiya. *Structure and Interpretation of Signals and Systems.* Addison Wesley, 2003.

[27] Richard G. Lyons. *Understanding Digital Signal Processing.* Prentice Hall, 2004.

[28] M.E. McIntyre, R. T. Schumacher, and J. Woodhouse. On the oscillations of musical instruments. *The Journal of the Acoustical Society of America*, 74(5):1325–1345, 1983.

[29] Leonard Meirovitch. *Fundamentals of Vibrations.* McGraw Hill, 2001.

[30] Don Morgan and Sanzheng Qiao. Accuracy and Stability in Mass-Spring Systems for Sound Synthesis. In *C3S2E*, pages 69–80, 2008.

[31] I.P. Omelyan, I.M. Mryglod, and R Folk. Optimized Forest-Ruth and Suzuki-like algorithms for integration of motion in many-body systems. *Computer Physics Communications*, 146:188–202, 2002.

[32] James M. Ortega and William G. Jr. Poole. *An Introduction to Numerical Methods for Differential Equations.* Pitman Publishing, Inc., 1981.

[33] Mark Pearson. *Synthesis of organic sounds for electroacoustic music.* PhD thesis, Department of Electronics, University of York, 2000.

[34] Curtis Roads. *The Computer Music Tutorial.* The MIT Press, 1996.

[35] Thomas D. Rossing. *The Science of Sound.* Addison-Wesely Publishing Company, 1982.

[36] Thomas D. Rossing and Nevill H. Fletcher. *Principles of Vibration and Sound.* Springer-Verlag, 1995.

[37] Julius O. Smith. *Mathematics of the Discrete Fourier Transform (DFT).* `http://ccrma.stanford.edu/~jos/mdft/Exponentials.html`, accessed (March 14/2008). online book.

[38] Julius O. Smith III. *Physical Audio Signal Processing: for Virtual Musical Instruments and Digital Audio Effects.* 2006.

[39] Ken Steiglitz. *A Digital Signal Processing Primer.* Addison Wesley Publishing Company, Inc., 1996.

[40] N. B. Tufillaro. Nonlinear and chaotic string vibrations. *American Journal of Physics*, 57(5):408–414, 1989.

[41] Vesa Välimäki, Jyri Pakarinen, Cumhur Erkut, and Matti Karjalainen. Discrete-time modelling of musical instruments. *Reports on Progress in Physics*, 69(1):1–78, 2006.