INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

DRUM SOUND SYNTHESIS BY COMPUTER SIMULATION OF THE WAVE EQUATION

BY

JAMES HAWLEY IRWIN, JR.

B.S.E.E., Bradley University, 1980
M.S.E.E., Bradley University, 1981

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University on Illinois at Urbana-Champaign, 1997

Urbana, Illinois

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

UMI Number: 9812640

Copyright 1997 by
Irwin, James Hawley, Jr.

All rights reserved.

UMI Microform 9812640
Copyright 1998, by UMI Company. All rights reserved.

This microform edition is protected against unauthorized
copying under Title 17, United States Code.

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

© Copyright by James Hawley Irwin, Jr., 1997

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

AUGUST 1997

WE HEREBY RECOMMEND THAT THE THESIS BY

JAMES HAWLEY IRWIN, JR.

ENTITLED DRUM SOUND SYNTHESIS BY COMPUTER SIMULATION OF THE WAVE EQUATION

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

James W. Beauchamp
Director of Thesis Research

D. M. Kang
Head of Department

Committee on Final Examination†

James W. Beauchamp
Chairperson

Douglas L. Jones

† Required for doctor's degree but not for master's.

O-517

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

DRUM SOUND SYNTHESIS BY COMPUTER SIMULATION
OF THE WAVE EQUATION

James Hawley Irwin, Jr., Ph.D.
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, 1997
J. Beauchamp, Advisor

This dissertation summarizes research on the problem of generating the sound of a drumhead by computer solution of the wave equation using finite difference methods. The research is novel in that it takes into account the stiffness of the drumhead material, damping due to energy dissipation into the air and rim, mass loading of the air, and changing tension caused by stretching of the drumhead.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

DEDICATION

THIS WORK IS DEDICATED TO
Linda Irwin, my wife,
and
Timothy, Christina, Paul, and Jonathan, my children,
for their love and patience through this long process.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ACKNOWLEDGMENTS

This work would not have been completed without the advice and help of my advisor, James Beauchamp, Ph. D. Thanks to my committee members, Professors Floyd Dunn, Douglas Jones, and Richard Weaver. Special thanks to my colleagues at Bradley University, Professors Santiago Navarro, Donald Schertz, Brian Huggins, and Tom Stewart. Finally thanks to the unseen help of my Lord and heavenly father Jehovah.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

TABLE OF CONTENTS

Page

1. BACKGROUND 1
   - 1.1 Empirical Models 1
   - 1.2 Physical Models 2
2. THEORY OF PERTURBATIONS 4
   - 2.1 Development of the Differential Equation 4
   - 2.2 Changes to the Classical Model 8
     - 2.2.1 Nonlinear (time-variant) tension 8
     - 2.2.2 Effect of shear force 13
     - 2.2.3 Effect of stiffness 13
     - 2.2.4 Effect of drum surface radiation load on modal frequencies 15
     - 2.2.5 Damping, energy loss, and energy transfer 16
   - 2.3 The Final Drumhead Equation 17
3. EXPERIMENTS AND JUSTIFICATIONS 18
   - 3.1 Tuning the Drumhead: circa 1992 18
   - 3.2 Interaction Between the Stick and the Drumhead 19
     - 3.2.1 Striking a real drum: initial velocity 19
     - 3.2.2 Drumstick interaction time experiment: 20 Mar 1991 20
     - 3.2.3 Drumstick-drumhead impact considerations 22
   - 3.3 Energy Lost Through Radiation-- Experiment: 15 Oct 93 23
   - 3.4 Rim Boundary Conditions 24
     - 3.4.1 Hinge boundary justification 25
     - 3.4.2 Effect of drumhead stiffness 26
     - 3.4.3 Effect of rigid Rim - Experiment: 11 Dec 95 26
   - 3.5 Time Varying Tension -- Experiment: 31 May 94 29

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.6 Summary 30
4. SIMULATION OF THE DRUMHEAD DIFFERENCE EQUATION 31
4.1 Finite Element vs. Finite Difference Simulation 31
4.2 Use of Templates for Computation 34
4.3 Mesh Size Determination 34
4.4 Use of Symmetry to Reduce Computation 35
4.5 Boundary and Initial Conditions 35
4.6 Stability of the Simulation 36
4.7 The Simulation Output Signal 38
4.8 Damping: Problems in Modeling 39
4.9 Summary 40
5. SIMULATION RESULTS 41
5.1 Classical Membrane Equation Verification by Center Strike Simulation 41
5.2 Classical Membrane Struck Off-Center 44
5.3 Determination of Simulation Nyquist Frequency 46
5.4 Effects of Radiation Loading in Simulation 48
5.5 Full Simulation of the Difference Equation 53
5.6 Effect of Changing the Monitoring Meshpoint for Output 54
5.7 Effect of Simulating Integrated Radiation Output 55
5.8 Effect of Altering Squareness of Meshpoint Areas 56
5.9 Effect of Stiffness on a Simulation 57
5.10 Effect of Time-Variant Tension on Modal Frequencies of Simulation 59
5.11 Effect of Damping on a Simulation 63
5.11.1 Simple lowpass rim damping 63
5.11.2 Effect of combined rim and radiation damping 64

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.12 Summary 67
6. SUMMARY AND CONCLUSIONS 68
6.1 Accomplishments and Failures 68
6.2 Future Work 68
Appendix A: DERIVATION OF "DEL TO THE FOURTH" 70
Appendix B: FINITE DIFFERENCE EQUATION 71
Appendix C: TEMPLATES 77
Appendix D: SOFTWARE LISTINGS 83
Appendix E: DISCUSSION OF TIME/FREQUENCY ANALYSIS TECHNIQUES 139
Appendix F: HEADER FILES FOR SIMULATIONS AND RECORDINGS 141
REFERENCES 159
VITA 163

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. BACKGROUND

The synthesis of drum sounds has been pursued for a number of years. Two major approaches for synthesizing a drum sound are empirical modeling and physical modeling.

1.1 Empirical Models

The basic idea of empirical models is to attempt mimicry of the time-frequency signal of actual drums by some method of synthesis. For example, with the Shaped-Noise Method, a white or pink noise source is passed through a time-varying filter to produce the percussion-like sound. In general, this technique can imitate a snare or a cymbal fairly well, but does poorly with a tom drum, because its sound spectrum is much less dense than that of the noise.

A second method is based upon the Karplus-Strong technique [1], which recirculates a random number through a delay line with probabilistic feedback and sample pair averaging. By adjusting the probabilities of the feedback and averaging, different percussion-like sounds can be produced. Once suitable parameters are found, this technique can be done in real time.

A third method uses cascaded frequency modulation (Schottstaedt [2], Le Brun [3]). The usual way of doing this is to use a frequency modulation signal to modulate another carrier. This results in a very dense, inharmonic spectrum. This spectrum can be shaped by changing the original modulation properties or the secondary modulation properties. With suitable parameters, percussion-like sounds can be generated. This method is not often used because of the difficulty in determining modulation indices that produce a particular percussion sound type. The parameters used for drums generally produce a spectrum that is too dense.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Other methods exist, such as sampling, additive synthesis, etc., but each has its limitations. The main problem with all of these methods is that they fail to accurately simulate percussion sounds which are neither noise-like nor random. While percussion sounds are probably deterministic, they are very complex. (Some researchers believe percussion sounds result from chaotic behavior (Fletcher et al. [35]), but to this author's knowledge, good evidence does not yet exist to prove drums are chaotic systems.)

1.2 Physical Models

A percussion instrument can be modeled as a large collection of coupled mass-spring-dashpot systems. Adrien and Rodet [4,5] pioneered this approach with strings and made suggestions that this technique could be extended to two-dimensional objects, such as drums. The idea seems to work well with strings, but despite these authors' suggestions not much work has been done to investigate using this method to simulate drums.

An alternative approach is to develop a differential equation for a vibrating object and solve it numerically. In the late 1960s, Pierre Ruiz first explored this topic (working exclusively with strings) under the direction of Lejaren Hiller. Hiller and Ruiz later improved the technique and published their results [6]. Subsequently their work was extended by Kubiak and Crounse [7], who synthesized several string tones through direct computation of the difference equation for a piano string.

The work of Hiller and Ruiz, along with that of Kubiak and Crounse, forms the foundation of this dissertation research. Since this research began, others have made contributions to the field of synthesis by direct computation of difference equations. The following paragraphs will mention those known to this author with explanations on how this research differs, where appropriate.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Several groups have taken the Hiller-Ruiz idea and refined it. The most common refinement takes into account that a string, piano or otherwise, acts like a delay line. Malouf et al. [8] used this approach to make physical models for a clarinet and a string instrument. Julius O. Smith III [9] improved this idea by noting that with strings the sound is transferred to the air mainly by the movement of the "ends" of the string, i.e., where the string connects to some kind of two-dimensional object. This means that all internal activity on the string can be modeled in terms of pure delay with the end points modified to produce the correct sound output. An important note is that it is only possible to make this simplification if it is assumed that the string is linear.

Transferring this theory from a one-dimensional string model to a two-dimensional drumhead model is fairly straightforward, but it does introduce several added complexities. Smith [10] presented an expansion of his waveguide technique to handle the added complexities of the second dimension. The paper is interesting and the idea may work well, but it limits its investigation to a linear membrane without stiffness or damping. This dissertation research considers these three factors: stiffness, nonlinear tension, and damping. If these are included, two dimensional waveguide simulation does not appear to provide a savings in computation time.

Most recently, Abdessamed Ramdane published a dissertation [11] using direct solution of a difference equation to produce timpani tones. His dissertation is similar to this one except that it does not include stiffness, nonlinear tension, or any damping of the drumhead. These features might not be significant for a timpani head due to the diameter of the head, the size of the mallets, and the relatively small deformation (relative to the head diameter) of the head from the mallet impact. (Some evidence of this is noted in two masters theses by Anderson [12] and Mills [13].) However, as will be argued in Chapter 2, these effects are probably significant for a tom or snare head.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2. THEORY OF PERTURBATIONS

Differential equations describing the motion of a drumhead have been known for many years. To my knowledge, Poisson [14] in 1828 and Lamé [15] in 1852 are the earliest authors to publish papers concerning membranes and strings. Rayleigh [16] is probably the best known investigator to publish in this area. Thus, the development of the equation in the next section is not new. It is presented here to provide a baseline from which this dissertation will be developed. It should be noted that this work is not meant to mirror reality in its fullest. Comerford and Kitching [17] and Hall [18] indicate that all real sounds have some disorder that cannot be easily reproduced in any typical mathematical equation. The minor variations and structural complexities within a real drumhead, rim, etc., preclude this research from being perfect. However, this research is intended to advance the field so that future work can more closely approximate reality.

2.1 Development of the Differential Equation

This section of the dissertation is heavily dependent upon several references. They are Kinsler et al. [24], Soedel [27], Morse and Ingard [29], Elmore and Heald [30], and Szilard [31]. Many items are repeated among the references, so to keep the development uncluttered, no specific citations will be mentioned during the development. This in no way belittles the influence on and help these references have given to the author.

Drawing from these sources, a differential equation describing the motion of a membrane with stiffness can be written as:

$$D\nabla^4 w = F_z(x, y, t) + \eta_x \frac{\partial^2 w}{\partial x^2} + \eta_y \frac{\partial^2 w}{\partial y^2} + \eta_{xy} \frac{\partial^2 w}{\partial x \partial y} - \rho_s \frac{\partial^2 w}{\partial t^2} \qquad (1)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Where w is the deflection function, F_z is the external applied force, η_i are the membrane coefficients (x, y normal and xy shear), and ρ_s is the area density (mass/area),

with

$$\nabla^4 w = \nabla^2 \left(\nabla^2 w \right) \tag{2}$$

and

$$D = \text{stiffness} = \frac{Eh^3}{12(1-\nu^2)} \tag{3}$$

where h is the membrane thickness, E is Young's modulus of elasticity, and ν is Poisson's ratio.

Eq. 1 is valid for wavelengths that are much greater than the thickness of the membrane, h.

Classical authors and, except for fleeting references, nearly all present day researchers assume several things concerning the behavior and properties of the drum head in their simulations and explanations. These are: $\eta_x = \eta_y = T$, tension; tension is independent of the deflection; stiffness is zero; damping forces are zero; and $F_z = 0$, except at $t = 0$. This last assumption signifies that the drum sound results from a transient response; thus F_z can be treated as an initial condition. Note that damping is ignored in Eq. 1 and because $\eta_x = \eta_y$, $\eta_{xy} = 0$.

Thus Eq. 1 reduces to the classical membrane equation:

$$\nabla^2 w = \frac{1}{c^2} \frac{\partial^2 w}{\partial t^2} \tag{4}$$

Where c is the speed of sound in the medium given by

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$c = \sqrt{\frac{T}{\rho_s}} \qquad (5)$$

where T is the tension and ρ_S is the surface density.

This equation describes the motion of an ideal drumhead with a rigid rim. It produces some of the frequencies produced by a real drumhead, but mimics nothing else. For comparison purposes, the time-dependent frequency spectrum of a simulation of Eq. 4 and that of a real drum are shown in Figs. 1 and 2. Only the general spectral shape is important in these figures. (Note: in the figure below and several others in this thesis, the sound-file from which the figure is generated has approximately 0.1 seconds of silence Inserted at the beginning in order to make the analysis easier.)

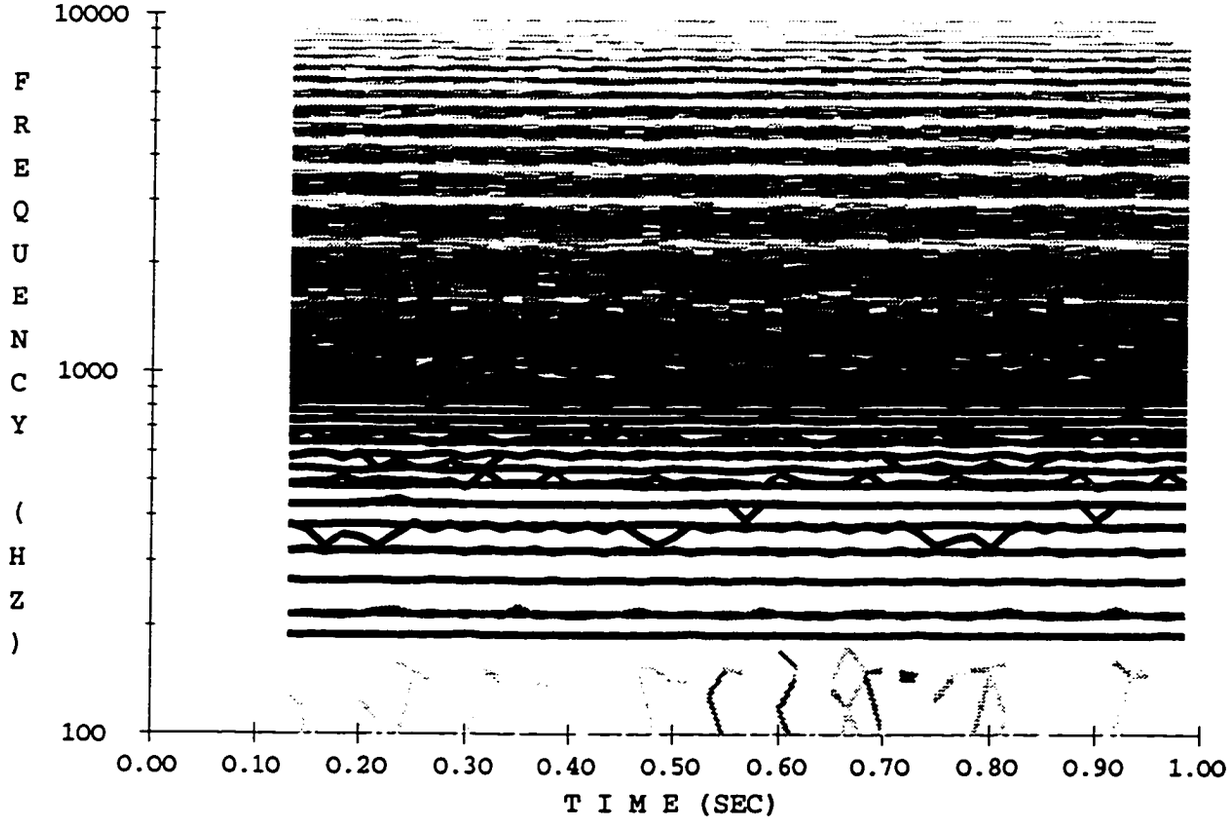

FIGURE 1. Time/frequency analysis of a simulation of Eq. 4

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

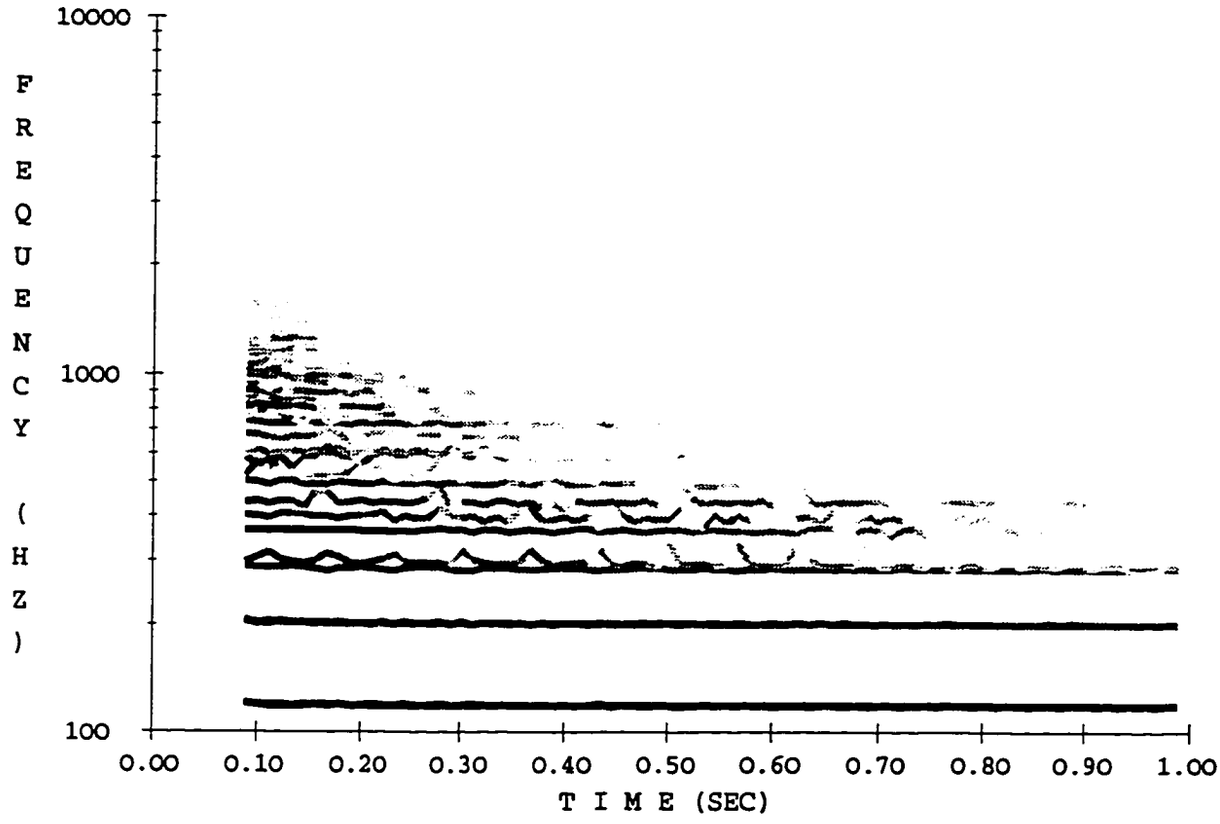

FIGURE 2. Time/frequency analysis of a real drum
(struck with a drumstick near a radius of 0.8, rim=1)

Comparing the two figures shows that because of the excessive energy indicated in the upper right quadrant of Fig. 1, Eq. 4 does not do well at simulating a real drum. (The frequencies and their density are also different, but these are not significant in this case.) This dissertation will investigate the effects of removing some of the assumptions that allow Eq. 1 to be simplified into Eq. 4. Removing all the assumptions would require the use of nonlinear plate theory, which is beyond the scope of this dissertation. Therefore, this dissertation will treat the removal of assumptions as perturbations to Eq. 4. (Note: One assumption not changed from the classical theory is that $\eta_x = \eta_y = T$. This is true when the membrane is uniformly tensioned, an assumption of this dissertation.) The

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

following sections deal with the perturbations to Eq. 4 (the removal of some assumptions) and the resulting differential equation.

2.2 Changes to the Classical Model

The present research addresses the effects of stiffness, nonlinear tension, and damping, all which were assumed negligible in the past. The changes to Eq. 4 caused by removing the assumptions listed above are discussed in the following sections.

2.2.1 Nonlinear (time-variant) tension

As stated earlier, most researchers assume that the tension of a string or membrane remains fixed as its material is deformed, (when struck, plucked, or just vibrating). While this useful assumption keeps the equation linear, it is not realistic. Measurements made on a real drum show that frequencies actually change with time for most hits. Rossing [19] has noted that a changing tension will cause the frequencies to vary with time. This effect has also been noted in other studies by Cahoon [20], Fletcher and Bassett [21], and Bassett [22]. An additional effect, which frequently accompanies frequency-varying tracks resulting from hard hits, is that as time progresses a single track splits into two distinct frequencies. To illustrate these concepts, Fig. 3 shows a time/frequency analysis of a tom drum struck very hard. Here we see a single frequency-varying track which splits into two distinct frequencies. This effect occurred in several recordings, but as yet there is no good explanation for it.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

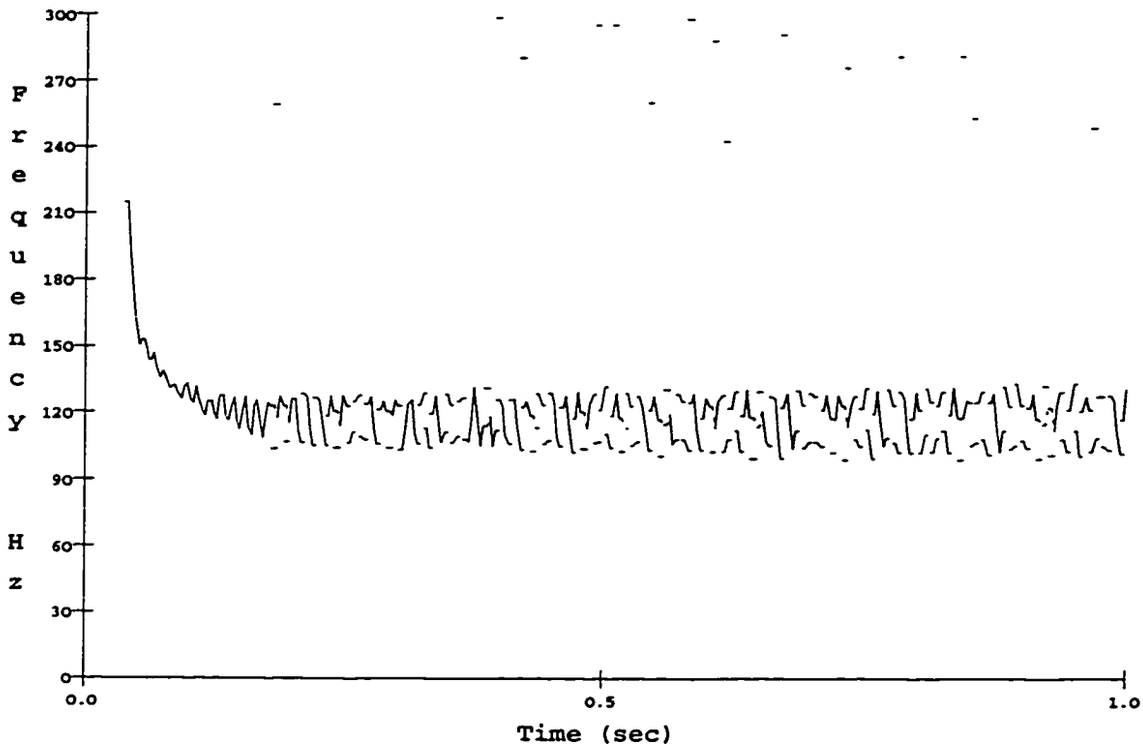

FIGURE 3. Time/frequency analysis of the 01 mode of a tom drum struck very hard (analysis by MQ method)

Cahoon [20] found that a bass drum struck with a reasonably hard blow temporarily changed its fundamental frequency by approximately seven percent, (just over one semi-tone). He computed the increased surface area of the drum head due to the vibrations set up in the drum head by assuming that the surface could be modeled as a cone. (This is a simplification of the Bessel function deflection pattern for the fundamental or 0,1 mode.) This increased area was used to compute a new value of tension which was used to calculate the next surface area, etc. A simulation based upon this procedure roughly matched the actual recorded change in frequency. Complications aside, these results show that changing tension can be a factor in the frequencies produced by a drum. (Cahoon's work influenced the papers by Fletcher and Bassett [21] and Bassett [22] who confirmed this effect and accepted Cahoon's solution.)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

On the other hand Mills [13] found that for a timpani time-varying tension did not cause significant frequency changes. It appears that under ordinary circumstances deflections of a timpani head are much smaller than those of a bass drum. Perhaps the tension effect may have been masked by other phenomena. Nevertheless, while this paper will be noted, this result will be ignored.

The model developed for this dissertation utilizes a tension which depends linearly on the surface area of the drumhead (and nonlinearly on the deflection). Thus

$$T = T_0 + \Delta T \tag{6}$$

where T_0 is the static tension and ΔT is the variable tension. Note that ΔT is always positive so the tension can never drop below the static tension. The form for ΔT is given by

$$\Delta T = \frac{Eh\Delta A}{A_0} \tag{7}$$

where A_0 is the static surface area and ΔA is the change in surface area. Other terms are defined with Eq. 1.

The derivation of the formula used to compute the increased area is given below and is based upon the surface area changing as a function of the cosine of the angle between the normal to a surface element and the z-axis. The deflection of the surface of the drumhead is denoted by w and is indexed in polar coordinates r and θ. The area of this surface is computed by a surface integral.

$$\text{deflection of surface} = w(r, \theta) \tag{8}$$

$$\text{area of surface} = \int_s dA = A \tag{9}$$

where

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$dA = \frac{rdrd\theta}{\bar{z} \bullet \bar{n}} \tag{10}$$

with $\bar{n}$ being the unit vector normal to the deflected surface and $\bar{z}$ being the unit vector normal to the xy-plane.

The differential area *dA* is determined by solving for the dot product which is

$$\bar{z} \bullet \bar{n} = \cos\gamma = \frac{1}{\sqrt{1 + (\nabla w)^2}}$$

Thus, the differential area can be computed by the 'DEL' operator. So that

$$dA = d\alpha\sqrt{1 + (\nabla w)^2} \tag{11}$$

where

$$d\alpha = rdrd\theta \tag{12}$$

Then, the binomial series expansion for small ∇w gives

$$dA = d\alpha\left(1 + \frac{(\nabla w)^2}{2}\right) \tag{13}$$

Substituting this into the surface integral gives

$$A = \int_s d\alpha + \frac{1}{2}\int_s (\nabla w)^2 d\alpha \tag{14}$$

The first integral yields the static or undeflected surface A_0.

$$A = Ao + \frac{1}{2}\int_s (\nabla w)^2 d\alpha \tag{15}$$

Using a vector identity found in many reference books will allow the second integral to be evaluated more easily. The vector identity is shown below, and the integral is expanded through substitution.

$$(\nabla w)^2 = \nabla\bullet\ (w\nabla w) - w\nabla^2 w \tag{16}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Therefore, the area of the surface is given by

$$A = Ao + \frac{1}{2}\int_s (\nabla \bullet (w\nabla w))\, d\alpha - \frac{1}{2}\int_s \left(w\nabla^2 w\right) d\alpha \qquad (17)$$

By Green's theorem the first integral in the previous equation can be changed to a line integral. This is shown in the next equation.

$$A = Ao + \frac{1}{2}\oint_{rim} w\nabla w \bullet ndl - \frac{1}{2}\int_s \left(w\nabla^2 w\right) d\alpha \qquad (18)$$

However, the rim of a drum, in this dissertation anyway, does not move, so this line integral evaluates to zero. In equation form the previous statement is w(rim) = 0.

therefore

$$\frac{1}{2}\oint_{rim} w\nabla w \bullet ndl = 0 \qquad (19)$$

and

$$A = Ao - \frac{1}{2}\int_s \left(w\nabla^2 w\right) d\alpha \qquad (20)$$

Because the static surface area A_0 is a constant, the only term needed is the change in surface area. So removing the static surface area term gives:

$$\Delta A = -\frac{1}{2}\int_s \left(w\nabla^2 w\right) d\alpha \qquad (21)$$

Converting this to a finite difference form gives the required equation.

$$\Delta A = -\frac{1}{2}\sum_m \sum_n (w_{mn}\nabla^2 w_{mn})\, a_{mn} \qquad (22)$$

where m is the radial index, n is the angular index, and a_{mn} is the indexed difference element area $r\Delta r\Delta\theta$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.2.2 Effect of shear force

The shear term η_{xy} handles the coupling of stresses between the *x* and *y* directions. If one side of a rectangle is stretched but confined to stay on its original axis, a side at right angles to the stretched side will be deformed to keep the "rectangle" closed. Its deformation changes the tension in the other axis depending upon the value of η_{xy}. The deformed side will not be on an axis, so its total elongation will not directly affect the perpendicular tension; see Fig. 4. The effect is not explicitly handled in this dissertation but is implicitly handled by the variable tension.

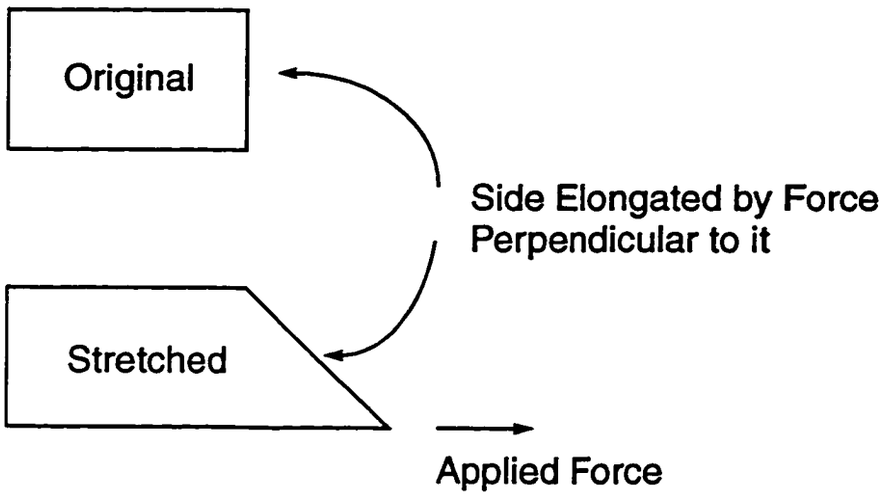

FIGURE 4. Explanation of effect of η_{xy}

2.2.3 Effect of stiffness

Stiffness is a property that distinguishes a sheet of paper from a rubber sheet. The property is easily demonstrated by conforming a circular sheet over a spherical surface. Rubber conforms easily, but paper does not. Young [23] verified that inharmonicity in a piano wire is caused, in part, by the stiffness of the wire. A wire that has tension but no stiffness will produce frequencies in a strict harmonic series, as shown by Kinsler et al. [24]. The

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

fact that the frequencies in a real piano wire are not harmonic is explained by the fact that wire has stiffness. Young found good agreement between the predicted frequencies that depended upon stiffness with the frequencies measured from a vibrating piano wire.

Though it has been mentioned in passing by other researchers, to this author's knowledge, no researcher, except for Mills [13], has investigated the effect of stiffness on the frequencies of a drumhead. He examined stiffness effects in a timpani and found that they were insignificant due to the fact that timpani produce mainly low-order/low-frequency vibration modes corresponding to slight bending of the drumhead. Stiffness only significantly affects high-order/high-frequency vibration modes. (In Mills's timpani model, stiffness would only become significant at frequencies above 17 kHz, but the timpani produces most of its sound output at frequencies below 600 Hz.)

Mills's equation, from page 10 of his paper, is used to determine the *crossover frequency* (f_{cr}), defined as the frequency below which the restoring force due to tension is greater than the restoring force due to stiffness.

$$f_{cr} = \left(\frac{f_{01} 2\pi r}{z_{01}}\right)^2 \frac{1}{2\pi} \sqrt{\frac{12\rho\,(1-\upsilon^2)}{Eh^2}} \qquad (23)$$

where

$$\begin{aligned}
f_{01} &= \text{fundamental frequency of drumhead} = 125 \\
r &= \text{radius of drumhead} = 0.25\,\text{m} \\
z_{01} &= \text{Bessel function (Jo) first zero} = 2.405 \\
\rho &= \text{mylar (drumhead) volume density} = 1390\,\frac{\text{kg}}{\text{m}^3} \\
h &= \text{drumhead thickness} = 2.54\times10^{-4}\,\text{m} \\
E &= \text{Young's modulus} = 5\times10^{9}\,\frac{\text{n}}{\text{m}^2} \\
\upsilon &= \text{Poisson's ratio} = 0.38
\end{aligned} \qquad (24)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This list gives values appropriate for a Mylar head tom drum. Using these, the crossover frequency computes to be approximately 7 kHz. A tom drum has significant amounts of energy in frequencies up to at least 10 kHz. Therefore, some of the frequencies in the tom drum will be affected by the stiffness. The frequencies most affected will be those near or above the crossover frequency. However, while frequencies below 7 kHz will be affected, at these lower frequencies the tension effects will increasingly tend to mask the stiffness effects.

2.2.4 Effect of drum surface radiation load on modal frequencies

Mills [13] found that the resonance frequencies of a timpani were affected by the reactive air loading of the membrane. The air loading has the effect of increasing the effective mass of the drumhead and making the frequencies shift downward. In fact, as confirmed by Christian et al. [25], the frequencies tend to shift towards a near-harmonic series. This apparently is why a timpani sounds so well "tuned." Rossing et. al. [26] give a brief confirmation that this effect is quite important for drum sound quality. Therefore, this dissertation includes the air load mass as a component to the mass of the drumhead. In the simulation the drumhead is divided into small pieces, through a polar mesh, which are indexed according to radii and angles of individual "mesh points," (see Fig. 5). Each meshpoint "area" is treated as a simple rigid piston. Using a polar mesh for the simulation makes the area corresponding to each meshpoint change with the radial distance. Fig. 5 shows a typical area associated with a meshpoint. To calculate the area, the two portions of the circle are transformed to chords, and the area of the resulting trapezoid is computed. This area is used to compute the radius of a circle of the same area. This radius is inserted into Eq. 25, taken from Kinsler et al. [24], with the resulting mass used to modify the area density $\overline{M}$ in the software routine *setupcoef*, described in Appendix D.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$\text{radiation mass} = m_r = \frac{8}{3}\rho_0 a^3 \qquad (25)$$

where ρ_o is the air density and *a* is the piston radius.

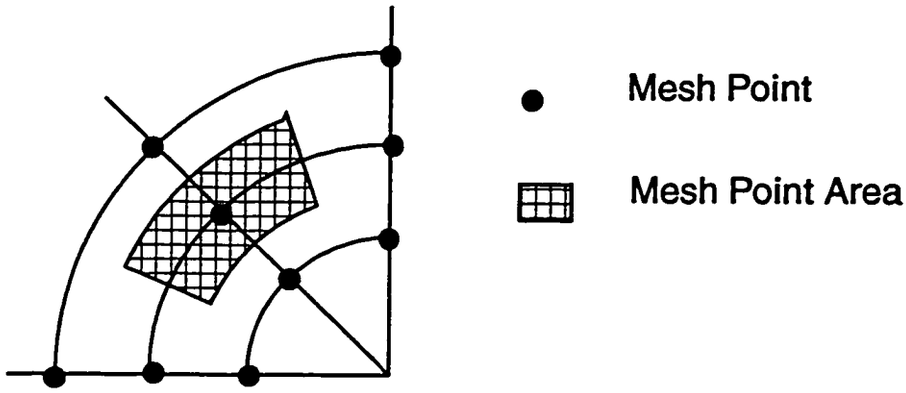

FIGURE 5. Area seen by a meshpoint

Since Mills [13] found that the resistance term from the piston impedance equation did not affect modal frequencies, this dissertation will include the energy loss due to the piston resistance term as radiation damping.

2.2.5 Damping, energy loss, and energy transfer

Any freely vibrating mechanical system loses energy. Because the sound of a drum attenuates as time progresses, losses are very important for sound quality. When making a model, one must address whether a particular energy loss is significant compared to other effects. In general, energy losses to heat, sound radiation, etc., and *energy transfers* to the rim, air, etc., occur. Strictly speaking, while energy losses cannot return to the drumhead and produce more sound, energy transfers can. However, for sake of simplicity both of these will be considered damping (energy loss), (see Sections 3.3, 3.4.3, and 4.8 for more details). Because of the extreme complexity of energy loss in a mechanical

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

system, the equations used are generally empirical, and parameters have been adjusted to fit experimental data or desired results.

There are many types of energy losses to model. In the interest of simplicity, this dissertation will investigate only a few of these. The most notable ones are *the energy transfer to the rim*, *energy lost in bending the drumhead*, and *sound radiation.*

The energy lost to sound is included in the model as a radiation loss to the air. Its defining equation is shown here, and it is applied to the entire drum surface.

$$\text{loss to air} \propto C\frac{\partial w}{\partial t} \tag{26}$$

where the coefficient C is estimated from experimental data.

The energy lost to the rim is modeled in terms of a dashpot at the boundary of the drumhead. Its equation has the same form as Eq. 26, and thus this energy loss modifies the coefficient C at the rim.

2.3 The Final Drumhead Equation

This concludes the perturbations to the classical model that this dissertation will investigate. The augmented equation, which includes the added terms discussed, is given below. The constant C includes the effect of rim damping as a boundary condition in the simulation and, unless variable tension is pertinent to a particular discussion, $T_0 + \Delta T$ will be henceforth written as T, the total tension.

$$(-D)\nabla^4 w + (T_0 + \Delta T)\nabla^2 w = \overline{M}\frac{\partial^2 w}{\partial t^2} + C\frac{\partial w}{\partial t} \tag{27}$$

where D is the stiffness, T_0 is the static tension, ΔT is the time-variant tension, $\overline{M}$ is the area density, C is the damping coefficient, and w is the deflection function.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3. EXPERIMENTS AND JUSTIFICATIONS

The simulation of a struck drumhead vibration developed in this dissertation attempts to model reality using an equation. In order to make this equation as realistic as possible and provide a check on the simulation, it was useful to test some properties/effects of a real drum. The previous chapter ended by stating a theoretical equation describing a drumhead vibration. The boundary conditions of a drumhead imposed by the rim were not described in any detail nor was the interaction of the stick with the drumhead. This chapter will describe these things by reporting the results of some experiments.

3.1 Tuning the Drumhead: circa 1992

In order to facilitate comparisons to the simulation, the tom drum, which the author used in recordings of real drum sounds, was tuned to a particular fundamental frequency F_{01} of 125 Hz. The tuning process for the type of drum used (a Roto-Tom) is as follows: First the rim screws were adjusted by placing salt on the drumhead, bonding a small disk magnet to the center, and placing an electromagnet just above the magnet. The electromagnet was driven by a sinusoidal signal, and its frequency was adjusted until a nearly circular Chladni pattern appeared in the salt. This pattern would be a perfect circle if the tension were uniform in all directions, so the rim screws were adjusted to make the pattern as circular as possible.

Having one circular node on the drumhead meant that the drumhead was actually vibrating at its F_{02} frequency. However, the theoretical ratio of the F_{02} frequency to the F_{01} frequency is the same as the ratio of the first two zeros of the J_0 Bessel function. Because this ratio is 2.295, the main screw was adjusted until the drumhead vibrated in the F_{02} mode at a frequency of 287 Hz. This implied that the F_{01} mode frequency would be at 125 Hz. (Using the same setup the F_{01} and F_{03} resonances were verified. As stated

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

before, this made the comparisons to the simulations straightforward because both the simulation and the real drum (theoretically) had the same F_{01} frequency.

3.2 Interaction Between the Stick and the Drumhead

Sound is produced immediately after a drumstick hits a drumhead. If the moment the drumstick first touches the drumhead is defined as time=0, then the drumstick/drumhead interaction defines the initial conditions for the drumhead differential equation. Because the stick maintains contact with the drumhead for a finite length of time, the simulation model may have to take into account this extended interaction. (The extended interaction can be ignored if the contact time is short enough; see Section 3.2.2.)

3.2.1 Striking a real drum: initial velocity

In the author's experience, a drummer does not apply force to the stick while the stick is in contact with the drumhead. Instead it rests as freely as possible in the drummer's hand. In mathematical terms, when a drumstick strikes a real drumhead (at t=0), the drumstick is at a constant velocity. This justifies the zero external force assumption made in Chapter 2 (see Eq. 1) and means that velocity is an initial condition that must be modeled in the simulation. The second initial condition that must be modeled is initial displacement. Because for t<0, the drumstick has not yet touched the drumhead, the displacement initial condition is zero. No other initial conditions are needed.

The author performed an experiment to determine the extent that the drumhead would deform from a blow by a drumstick. The measurements showed that the hardest hits caused a deflection of approximately one centimeter. This information was used first to develop a formula to compute an initial velocity at the *hitpoint* from given input *hitstrength*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(arbitrary units), and secondly to help justify treating the boundary condition at the rim as a hinge (see Section 4.5).

3.2.2 Drumstick interaction time experiment: 20 Mar 1991

In addition, the author's experience indicated that a striking drumstick remains in contact with a drumhead for an certain length of time. Thus, an experiment was performed to estimate the drumstick-drumhead contact time to determine whether this time is significant.

The interaction time experiment was done using two different methods to provide a cross check on the results. The first method used an accelerometer (<5g mass) attached to the tip of the drumstick, and the second used foil on the stick tip and drumhead to complete a circuit. Because the drumhead can only push the drumstick but not pull it, the interaction time using the accelerometer was measured from the time the acceleration (actually deceleration) started until the sign of the acceleration changed. The interaction time using the foil only was measured as the time that the "switch" was closed. Neither method appeared to disrupt the unloaded action of the stick or drumhead. The drumhead was the batter head of a typical snare drum (material was Mylar with 10 mils thickness). The drumsticks were chosen from several available in the lab, (two sizes of wood-tipped sticks and one size of nylon-tipped stick were used). The drum was struck in the center, halfway to the rim, and one inch from the rim. Each location was struck several times, and a typical strike was recorded. The data are in the table below.

A reading of the table shows that the interaction time is nearly constant unless the strike point is near the rim. (The difference of 1 ms between the accelerometer and the foil switch is probably accounted for by the drumhead following the stick through the change of sign in the acceleration. Contact would then be broken when the drumhead pulls away.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 1: Drumstick-Drumhead Interaction Time

Experiment	Stick Tip	Hit Location	Contact Time (ms)
accelerometer	wood #1	center	4.14
accelerometer	wood #1	mid-way	4.22
accelerometer	wood #1	near rim	3.83
accelerometer	wood #1	mid-way	4.06
accelerometer	wood #2	center	4.90
accelerometer	wood #2	mid-way	4.92
accelerometer	wood #2	mid-way	4.92
accelerometer	wood #2	near rim	3.94
accelerometer	nylon	center	4.84
accelerometer	nylon	mid-way	4.73
accelerometer	nylon	mid-way	4.82
accelerometer	nylon	near rim	3.91
foil switch	nylon	center	5.24
foil switch	nylon	mid-way	5.27
foil switch	nylon	mid-way	5.22
foil switch	nylon	near rim	4.2
foil switch	wood #1	center	5.30
foil switch	wood #1	mid-way	5.37
foil switch	wood #1	mid-way	5.32
foil switch	wood #1	near rim	4.12

This was not easy to measure, so the zero crossing was used.) This experiment shows that the interaction time is small when compared to the overall simulation time (typically three seconds); therefore, no extended interaction time between the drumstick-drumhead was put into the simulation model.

However, setting the interaction time to zero would have made the impact an impulse, which is not useful for an ideal membrane. (This is because an impulse on a membrane produces infinite displacement; see Morse and Ingard [29] and Rossing [32].) To prevent the interaction from being an impulse, the "drumstick" was assumed to interact with a finite (non-zero) area of the drumhead.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Also, a zero interaction time would have resulted in excessive high-frequency terms in the simulation output. This follows from the Fourier transform of the drumstick strike. A longer interaction time reduces the amplitudes of higher frequencies that can be excited in the linear drumhead (Hall [33]).

A future improvement to the model might be to increase the area density of the drumhead in the area of contact, while the stick is in contact with the drumhead (Hall [18, 33]).

3.2.3 Drumstick-drumhead impact considerations

When the drumstick hits the drumhead, profound changes to the drumhead's surface in the region of the hit occur. The fact that the initial condition is a velocity means that the acceleration at t=0 approaches infinity as the simulation time interval (Δt time between computations) approaches zero. Nonlinear effects due to this large acceleration are beyond the scope of this dissertation and will be ignored.

Therefore, the simulation model for the impact was based upon a quasi-static, or slow motion, interaction. For this method, it is important to determine the relative contributions of the membrane and plate portions of the differential equation and the physical size of the drumstick tip. An ideal membrane conforms to the shape of the drumstick tip. This would introduce nonlinear effects into the membrane because "smoothness" assumptions, used in the Chapter 2, would be violated (Eq. 1 stiffness portion). A drumhead not being an ideal membrane, its plate characteristics will affect this conformation to the tip. If a plate is deflected by a point force, it will assume a smooth shape that depends upon the "stiffness" of the plate. This would keep the interaction linear. Thus, if the drumstick tip is small enough or the stiffness large enough, the drumhead has a smooth shape and linear interaction.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Measurements of typical drumstick tips yield a range 5 mm to 9 mm diameter, while the critical size for linear interaction with a mylar drumhead is approximately 7 mm. (This value is the required tip diameter needed to produce convergence in the spatial Fourier transform of the Mylar drumhead used in this dissertation.) This indicates that drumsticks are near the boundary of linear/nonlinear quasi-static interaction. This dissertation uses linear interaction by keeping the stick tip less than 7 mm. In fact, the simulated stick tip is nearly but not quite a point source (see Section 3.2.2).

3.3 Energy Lost Through Radiation– Experiment: 15 Oct 93

This experiment was conducted to determine whether a significant amount of energy is transmitted into the air by a vibrating drumhead. A solenoid with a modified end and spring return was mounted so that it would strike the drumhead upon release. By moving a stop, the drumhead could be struck with an adjustable consistent force. An accelerometer was mounted to the drum surface well away from where the solenoid would strike. Based on comparisons between recordings made with and without the accelerometer, its light mass did not seem to significantly affect the sound of the drum.

The drum was struck in the air and in a 5 mm-mercury vacuum. The resulting instantaneous acceleration-versus-time files were converted to RMS amplitude and plotted on a decibel scale. Typical results are shown in Fig. 6. The downward slopes of the resulting curves correspond to the negative exponents (scaled) of the loss terms. Because both curves would have similar noise floors and rim losses, a significant difference in slope would only be due to a significant radiation loss in the air load case.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

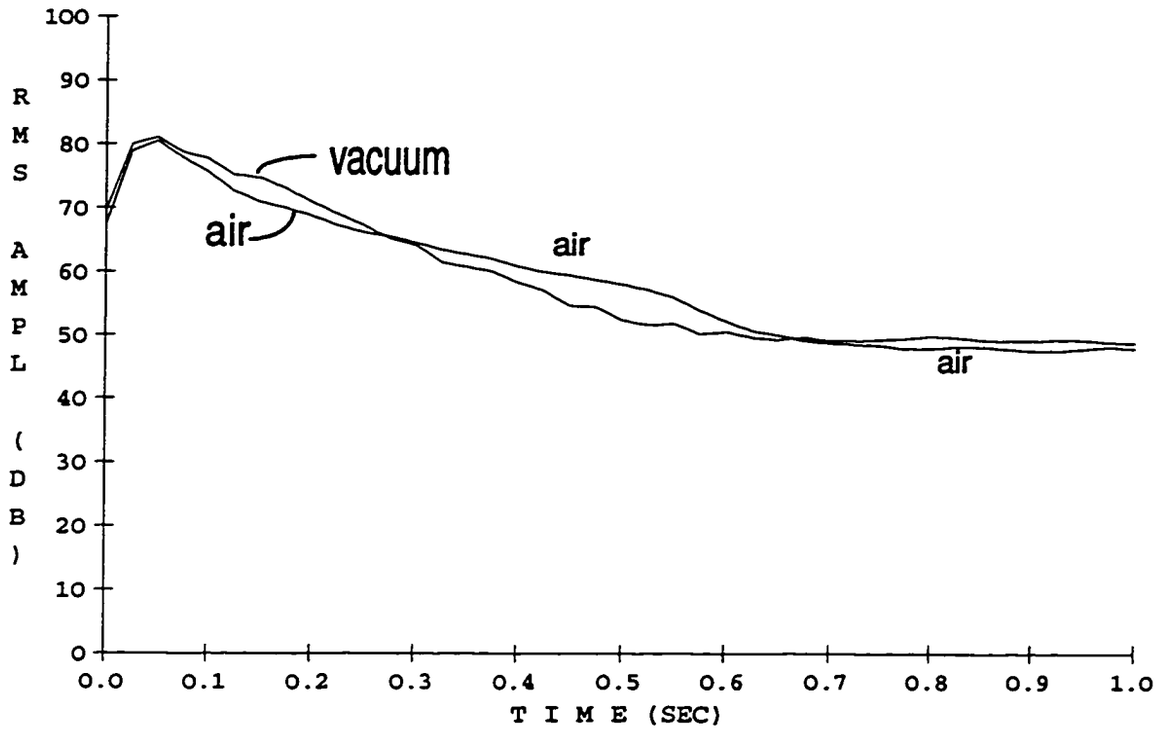

FIGURE 6. Comparison of a drum's acceleration decay in air and vacuum

Since the two curves are very similar, it is clear that the energy loss to air is quite small. The data obtained gave no reliable value of the radiation damping coefficient *C*, but did provide a starting estimate. (Similar graphs were made with the energy split into octave frequency bands, but the results were inconclusive as to whether there were different damping coefficients, *C*, in different octaves.)

3.4 Rim Boundary Conditions

Solving the differential equation requires boundary conditions in addition to the initial conditions. The drumhead's only boundary is the rim.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.4.1 Hinge boundary justification

Real drums are made by stretching the drumhead material over the apex of a triangularly shaped rim top (See Fig. 7). The shape of the rim top is such that the drumhead appears to be free to rotate around the contact point, which imitates a hinge.

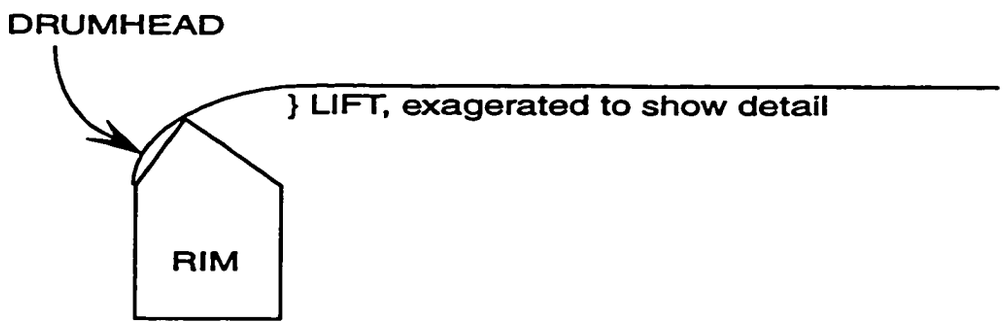

FIGURE 7. Side view of a drum rim and drumhead

One experiment, involving applying extreme pressure to a drumhead near the rim, verified by observation that the drumhead did not touch more than the apex of the rim. In fact no pressure less than that needed to tear the drumhead caused the drumhead to touch other portions of the rim. Another experiment determined that no typical deflection of the drumhead, caused by hitting it with a drumstick, would cause contact anywhere except at the apex.

These experiments led to the choice of a "hinge" to describe the boundary conditions. (Note that a hinge can rotate freely but the "pin" does not deflect.) This means that for

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

computation purposes the slope of the drumhead surface does not change across the rim, (see Appendix C for implementation details).

3.4.2 Effect of drumhead stiffness

The choice of a hinge is not completely accurate because when the edge of the drumhead is pulled down to tension it, the *stiffness* of the drumhead material causes the drumhead to make an angle with the rim and lift up the center of the drumhead. This will impart a spring-like quality to the hinge, much like a saloon door hinge and is illustrated in Fig. 7. This means that the rim is not exactly a hinge, but these non-hinge effects were deemed small enough to ignore.

3.4.3 Effect of rigid Rim - Experiment: 11 Dec 95

While it was hoped that a simple hinge boundary would provide the appropriate boundary conditions, the results of simulations which incorporated it showed little similarity to results from a real drum. Comparison of Fig. 8, a typical simulation using the hinge boundary condition, with that of a real drum, as shown in Fig. 9, indicates that there was insufficient high frequency absorption in the simulation. (Energy absorption is indicated by the frequency tracks growing lighter as time progressed.) This led to an experiment in which the rim of a drumhead was clamped to a very rigid frame, ensuring that the rim could not move. Fig. 10 shows the time-frequency spectrum generated by striking from this drumhead with a drumstick.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

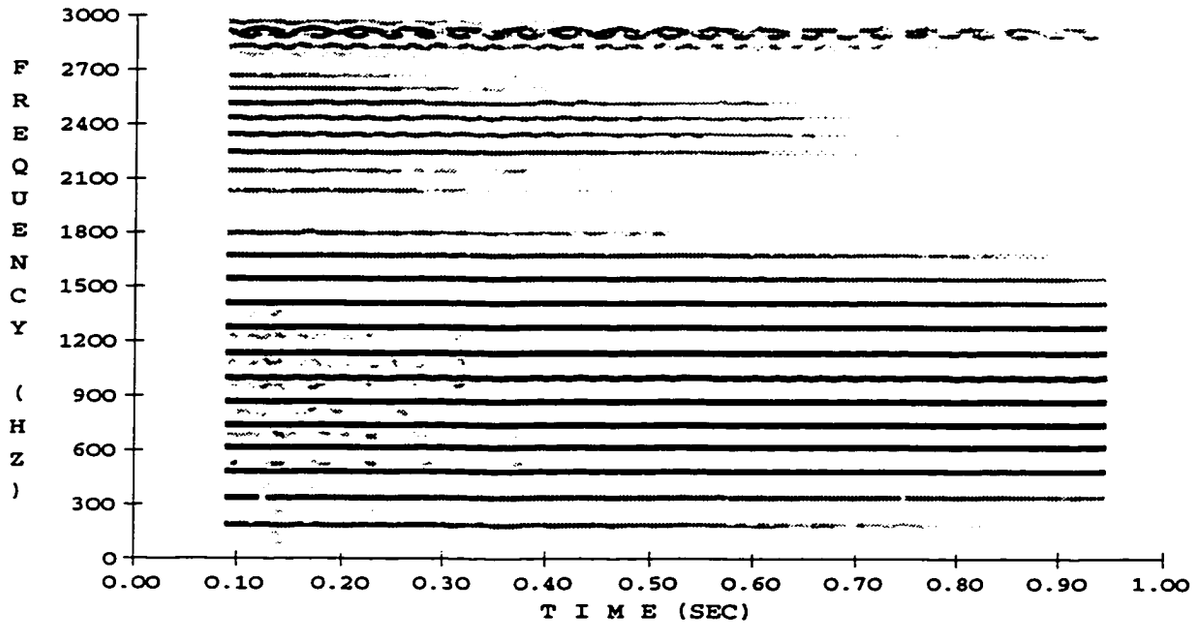

FIGURE 8. Time/frequency spectrum of a simulated drum with radiation damping (Similar to Fig. 1 but with damping)

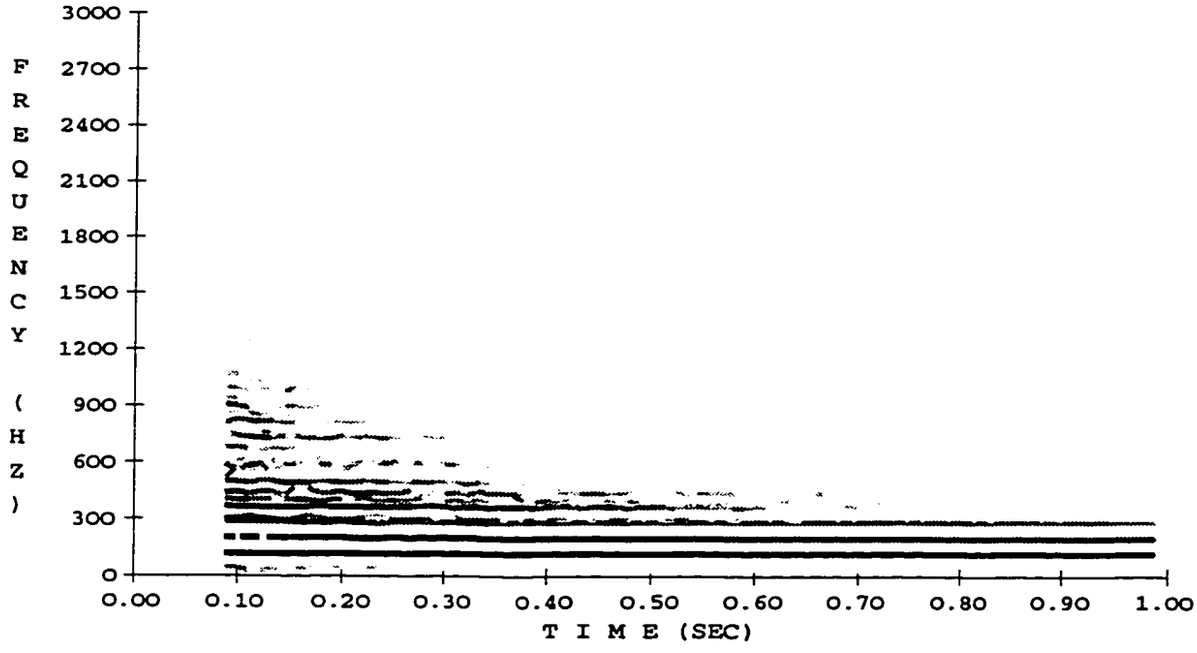

FIGURE 9. Time/frequency spectrum of a real drum (expanded spectrum of Fig. 2)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

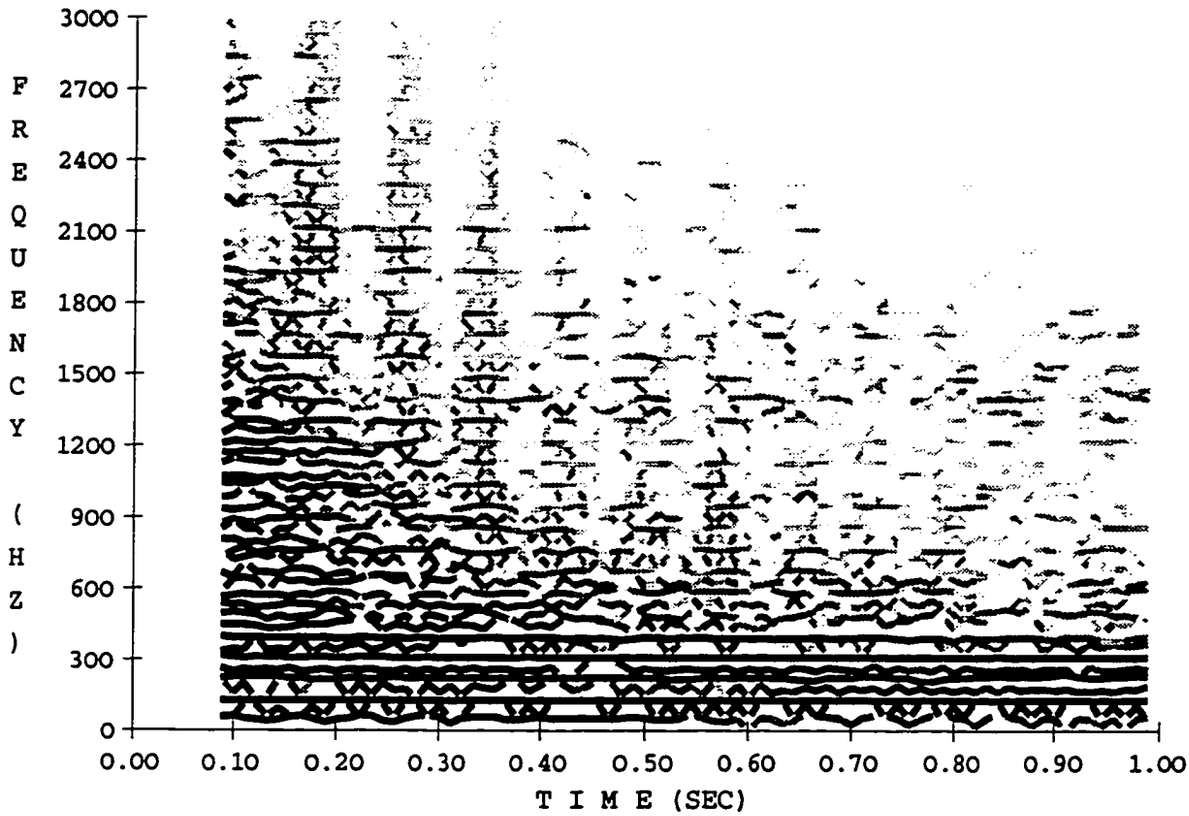

FIGURE 10. Time/frequency analysis of a real drum with a rigid rim

Fig. 10 appears more similar to the results obtained from the simulations (compare with Figs. 8 and 9) because of the "energy" appearing in the upper-right quadrant. This indicates that despite appearances, the rim was actually vibrating, and significant amounts of high-frequency energy were being transferred to the rim. (The earlier "Energy Lost Through Radiation" experiment proved little energy went into the air.) The result was incorporated into the simulation by keeping the rim a hinge and adding an extra resistive "dashpot" to the meshpoints one step in from the rim. Keeping the rim rigid in the simulation made the simulation easier to program. Adding the extra damping next to the rim was trivial and did not introduce much error. Actually, because a rim boundary has finite

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

stiffness (see Section 3.4.2) as well as our assumed damping, the simulation just puts a hypothesized rigid rim one mesh point beyond the "real rim", which has losses.

3.5 Time Varying Tension -- Experiment: 31 May 94

As stated in the previous chapter (see Fig. 3), the tension, and therefore the frequency of vibration, of a drumhead must increase when a deflection occurs. Figs. 11 and 12 show this effect. (Even though the PV analysis routine does not generally work well with large frequency shifts (see Appendix E) it worked better than the MQ method for this case. The initial squiggles in the plots are probably artifacts of the analysis method. These particular figures minimize them, but they could not be removed from all curves simultaneously. The impact time point was shifted to give a clear view of the frequency shifts.)

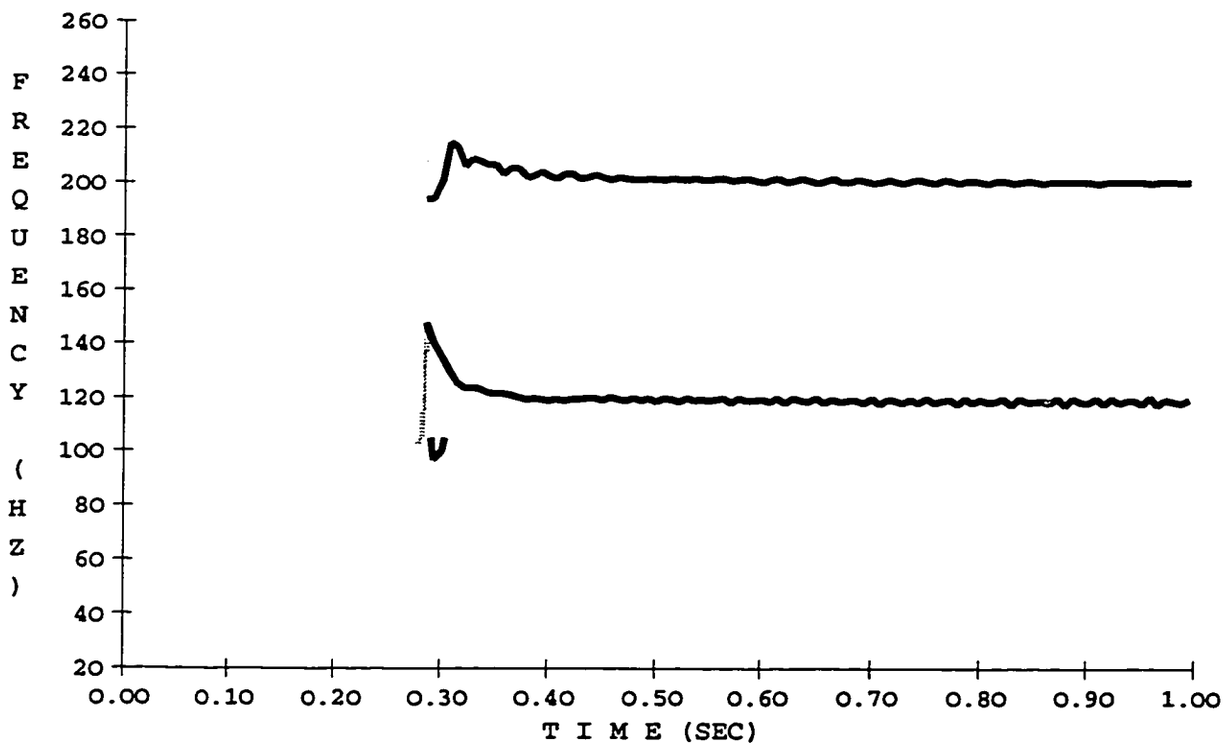

FIGURE 11. Real Drum -- frequency shift due to a forte hit

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In Fig. 11 it is easy to see the frequency shift in the fundamental.

Fig. 12 shows the first four partials shifting frequency. The harder impact of the drumstick helped to clarify the frequency changes in the first four partials. This data verifies the need to model a variable tension.

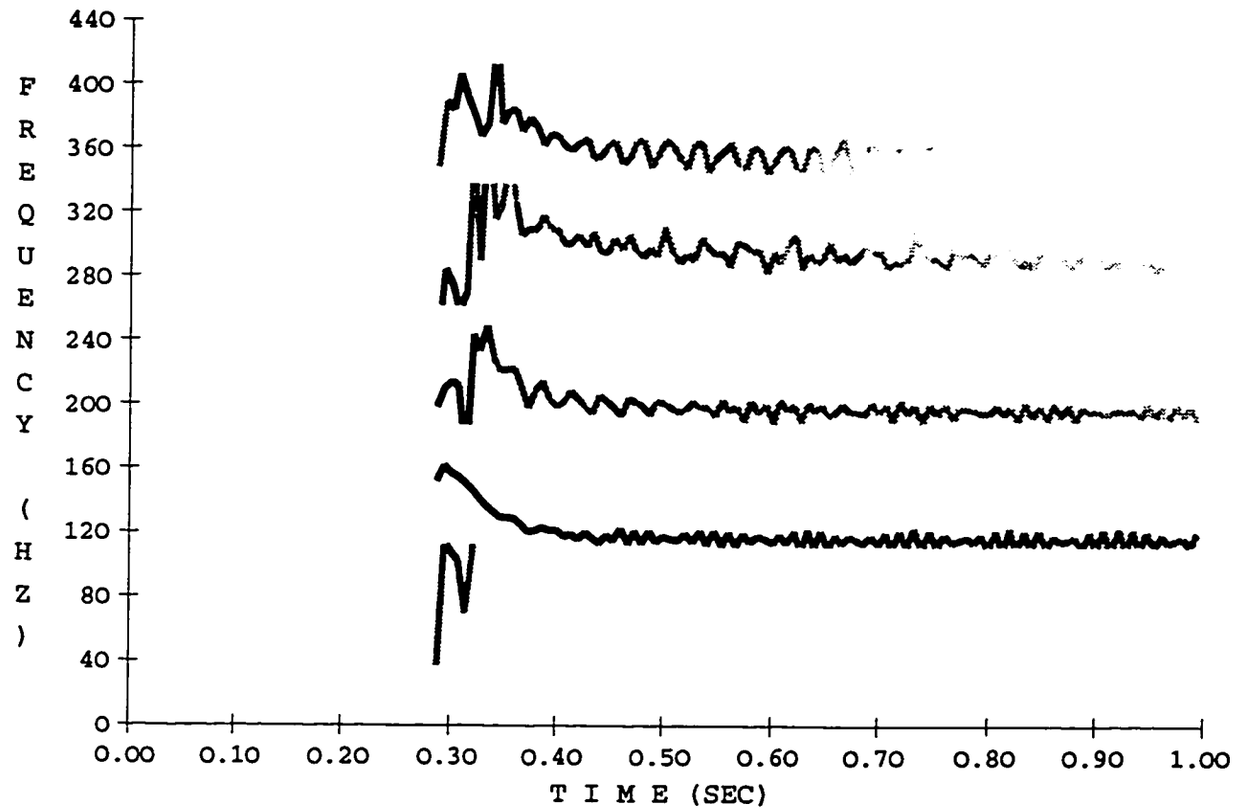

FIGURE 12. Real Drum -- frequency shifts due to a triple forte hit

3.6 Summary

The experiments performed supported ignoring the finite drumstick-drumhead interaction time, using a linear drumstick-drumhead impact, having minimal radiation losses, using nonlinear tension, and using a viscously damped, hinged rim in simulating a drumhead.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4. SIMULATION OF THE DRUMHEAD DIFFERENCE EQUATION

Simulating a differential equation on a computer requires converting from a continuous time/space representation to some form of discrete time/space representation. The mere switch to 'discrete' produces some problems which must be addressed. Other choices must also be made.

4.1 Finite Element vs. Finite Difference Simulation

Both the finite element and finite difference methods are widely used to simulate differential equations. The two methods have different strengths and weaknesses, and on the basis of these, a method was chosen. The major trade-off between the two methods for this dissertation concerned how the method made use of physical modeling.

In talking to colleagues who have done many significant numerical simulations, they generally stated that most simulations using this method make use of shape functions which have boundary constraints imposed upon them. If the boundary constraints are valid, then the object modeled will behave correctly. They stated that the boundary constraints are not difficult to determine, but that knowing them did not necessarily give information as to what physical parameter would alter a particular constraint. Further discussion made it apparent that this method is generally useful only when the object to be simulated is too complex to write a differential equation that describes its overall behavior.

On the other hand, the finite difference method starts from a differential equation, and any effects or changes to the differential equation are directly modeled by the simulation. Because this dissertation deals with a very simple surface whose vibrations are represented by a modified version of a well-known classical differential equation, it was possible to use the method of finite differences for simulations. Note that changes to the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

classic equation are modeled directly so that simulation results can be traced back to parameters in the differential equation. The finite difference method provides a good link between the physics of the vibrating object and the results obtained from the simulation.

The basic idea of the finite difference method is to replace derivatives with finite differences. Once this is done, the solution to the equation can only be determined at discrete locations, called meshpoints. For the polar coordinates used in this dissertation, the meshpoint locations correspond to line intersections on a polar graph (see Fig. 5). Each meshpoint is selected by a radial index m, an angular index n, and a time index i. The difference equation is set up to produce a deflection w at any particular time and location based on the deflection at neighboring locations and previous times; thus, the total surface is generated one meshpoint at a time.

The final difference equation is shown in Eq. 28, where w is the deflection at a given location indexed by m, n, and i; $\overline{M}$ is the area density; C is the damping coefficient; D is the stiffness; T is the tension; Δt is the time increment; Δr is the radial difference; $\Delta\theta$ is the angular difference; and r is the radius. (A difference is the distance between two adjacent meshpoints.)

(Appendix B gives the difference equation development.)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$w(m,n,i+1)\left[\frac{\bar{M}r^4}{\Delta t^2}+\frac{Cr^4}{2\Delta t}\right] = w(m,n,i-1)\left[\frac{-\bar{M}r^4}{\Delta t^2}+\frac{Cr^4}{2\Delta t}\right]$$

$$+w(m-2,n,i)\left[\frac{-Dr^4}{\Delta r^4}+\frac{Dr^3}{\Delta r^3}\right]$$

$$+w(m-1,n-1,i)\left[\frac{-2Dr^2}{\Delta r^2\Delta\theta^2}-\frac{Dr}{\Delta r\Delta\theta^2}\right]$$

$$+w(m-1,n,i)\left[\frac{4Dr^4}{\Delta r^4}-\frac{2Dr^3}{\Delta r^3}+\frac{Tr^4}{\Delta r^2}+\frac{Dr^2}{\Delta r^2}-\frac{Tr^3}{2\Delta r}+\frac{Dr}{2\Delta r}+\frac{4Dr^2}{\Delta r^2\Delta\theta^2}+\frac{2Dr}{\Delta r\Delta\theta^2}\right]$$

$$+w(m-1,n+1,i)\left[\frac{-2Dr^2}{\Delta r^2\Delta\theta^2}-\frac{Dr}{\Delta r\Delta\theta^2}\right]$$

$$+w(m,n-2,i)\left[\frac{-D}{\Delta\theta^4}\right]$$

$$+w(m,n-1,i)\left[\frac{4Dr^2}{\Delta r^2\Delta\theta^2}+\frac{Tr^2}{\Delta\theta^2}-\frac{4D}{\Delta\theta^2}+\frac{4D}{\Delta\theta^4}\right]$$

$$+w(m,n,i)\left[\frac{-6Dr^4}{\Delta r^4}-\frac{2Tr^4}{\Delta r^2}-\frac{2Dr^2}{\Delta r^2}-\frac{8Dr^2}{\Delta r^2\Delta\theta^2}-\frac{2Tr^2}{\Delta\theta^2}+\frac{8D}{\Delta\theta^2}-\frac{6D}{\Delta\theta^4}+\frac{2\bar{M}r^4}{\Delta t^2}\right]$$

$$+w(m,n+1,i)\left[\frac{4Dr^2}{\Delta r^2\Delta\theta^2}+\frac{Tr^2}{\Delta\theta^2}-\frac{4D}{\Delta\theta^2}+\frac{4D}{\Delta\theta^4}\right]$$

$$+w(m,n+2,i)\left[\frac{-D}{\Delta\theta^4}\right]$$

$$+w(m+1,n-1,i)\left[\frac{-2Dr^2}{\Delta r^2\Delta\theta^2}+\frac{Dr}{\Delta r\Delta\theta^2}\right]$$

$$+w(m+1,n,i)\left[\frac{4Dr^4}{\Delta r^4}+\frac{2Dr^3}{\Delta r^3}+\frac{Tr^4}{\Delta r^2}+\frac{Dr^2}{\Delta r^2}+\frac{Tr^3}{2\Delta r}-\frac{Dr}{2\Delta r}+\frac{4Dr^2}{\Delta r^2\Delta\theta^2}-\frac{2Dr}{\Delta r\Delta\theta^2}\right]$$

$$+w(m+1,n+1,i)\left[\frac{-2Dr^2}{\Delta r^2\Delta\theta^2}+\frac{Dr}{\Delta r\Delta\theta^2}\right]$$

$$+w(m+2,n,i)\left[\frac{-Dr^4}{\Delta r^4}-\frac{Dr^3}{\Delta r^3}\right] \qquad (28)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2 Use of Templates for Computation

The finite difference method for solving differential equations computes its solution by using previously computed values of the difference equation from locations that differ from the location being computed. The previously computed values have indices that are fixed relative to the location being computed. Therefore, the difference equation is put into a form that is independent of the computation location (see Appendix B), so that the computation software can be written as an indexed loop in order to provide a simple calculation method. This form is called a template and is indexed from the mesh location being computed. The template is three-dimensional because there are three derivatives to be calculated: one each with respect to radius, angle, and time. The template must be modified when it intersects any boundary of the object and the rules for handling modifications and the modifications themselves are listed in Appendix C.

The templates must also be modified to take into account the initialization of the simulation; i. e., some provision must be made to handle the time boundary. All simulations start at time zero. The modifications to the template for the time boundary are developed in Appendix C.

4.3 Mesh Size Determination

The number of radial meshpoints needed to simulate a particular maximum frequency was determined by experiment (see Section 5.3). However, the number of angular meshpoints, *N_max*, was determined from the maximum number of radial meshpoints, *M-max*, in a procedure that kept the regions of the mesh as "square" as possible, considering the polar nature of the mesh. This should minimize errors caused by extreme rectangular mesh shapes.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.4 Use of Symmetry to Reduce Computation

The simulation of the drum surface by using the finite difference method is computationally intense. To reduce the number of computations, a symmetry condition is used to decrease the number of meshpoints computed.

If an imaginary diameter is drawn on the drum surface through the point at which the drumstick strikes the surface, the surface has even symmetry with respect to this line. Thus, only one-half of the drum surface needs to be computed to determine the displacement of the entire surface. The diameter described above is called the *strikeline*.

The strikeline introduces a boundary onto the drum surface. Modifications that this boundary makes to the template are described in Appendix C.

4.5 Boundary and Initial Conditions

Besides the strikeline mentioned in the previous section, the rim provides the other spatial boundary. Real drum rims are attached to shells (circular cylindrical surfaces) to make the rim as stiff (immovable) as possible. And as mentioned in Section 3.4, the shape of the rim top is such that the drum surface is free to rotate around the contact point, which imitates a hinge. These two conditions affect the modification of the template at the rim by requiring any rim meshpoint to have a zero deflection and an arbitrary slope (see Appendix C).

As mentioned in Section 3.2.1, the initial conditions of the simulation are that at time zero the drumhead is flat and receives an initial velocity from the drumstick. A single parameter, the *hit strength*, is used to set up the initial conditions. Note that a real drum is struck with a stick that is "free." This means that the stick has a constant velocity and is not constrained in the vertical direction. In actual experiments, it was determined that the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

maximum deflection produced by a drumstick on a typical drumhead was approximately one centimeter. In order to mimic the strike of a drumstick at a constant velocity, the simulation program constructs a velocity profile in the displacement matrix around the point the stick strikes that extends into negative time. Velocity is arbitrarily scaled so that the maximum allowed value of the hit force produces a deflection of one centimeter. (Note that exactly what physical velocity this implies is not important because a human drummer would determine this by experimentation.The drummer would actually correlate loudness and force of hit, rather than velocity and hit force.) This deflection is interpreted as a velocity by the program due to the nature of the coefficients that are used (see Appendix D, *setupcoef.c).*

In order to start the simulation, a special set of coefficients is used (see Appendix D, *setupcoef.c*) so that the initial velocity contour of the drum head, which is stored in the matrix that contains the time slices of the drumhead, is used.

4.6 Stability of the Simulation

If the template terms involving stiffness are ignored, the difference equation simulated in this dissertation reduces to one having hyperbolic form. This form is well known and has particular stability criteria (Al-Khafaji [28]). These criteria are:

$$\Delta t = \frac{\beta \Delta x}{c} \tag{29}$$

where

$$\Delta x = \min(\Delta r, r\Delta\theta) \tag{30}$$

and for convergence the stability factor β is restricted to the range

$$0 < \beta < \frac{1}{2} \tag{31}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Under these criteria the simulation is stable, provided the stiffness is zero. However, when there is nonzero stiffness, the simulation is only conditionally stable. It was found that when stiffness was introduced, there appeared to be two competing sub-solutions; a stable one and an unstable one. By looking at the computations at each time step, it was clear that the stability of the overall solution depended upon whether the stable sub-solution dominated within the first few milliseconds. The difficulty appears to be due to the choice of using polar coordinates for the simulation. Polar coordinates permit very convenient boundary conditions, but a problem develops at the coordinate origin. The difference (and differential) equation includes a $1/r^2$ term in it and a $1/r^4$ term if stiffness is included. This is expected because the solution to the classical membrane equation is based upon Bessel functions. One type of Bessel function, the 'I' type, has a "pole" at zero. Classically, this solution is deemed impossible, so its coefficients are set to zero. The simulation can handle this problem in the classical equation by using L'hospital's theorems for indeterminate forms. However, when stiffness is added to the differential equation, it is unclear that L'Hospital's theorems apply past the second derivative. (The form after the third application of L'Hospital's theorems does not appear to be indeterminate.) This means that the difference equation including stiffness has a singular point at the origin.

This singularity is handled in the simulation by one of two methods. The first is to apply L'hospital's theorems without justifying their use. This leads to a difference equation at the center that appears to be reasonable and gives reasonable simulations even though there is no mathematical justification for doing this. The second method is to not compute the value at the center; i. e., the solution is computed near, but not at, the singularity. This seems defensible because in mathematics one can ignore a finite number of singular points in an otherwise continuous function and still be confident that the solution is valid. Here, physical reasoning validates ignoring this singular point because if another type of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

mesh coordinates, say X-Y, is used, the singularity disappears. A modification of this second method is to set the stiffness to zero at the center and use only the classical difference equation to calculate the center.

Because computations at the origin are so troublesome, why include them? There are two main reasons to include computations at the origin. The first is that the most convenient way to confirm the correctness of the difference equation is to excite it at the origin and look for the classical circular modes in the solution. The second is that the origin is needed to complete templates that are used to compute the mesh values near the origin.

These methods of handling the singularity all work with limited success. It appears that for fine meshes, the near singularity of the equation with stiffness at meshpoints near the center of the drumhead causes the unstable sub-solution to dominate. This remains a problem with this simulation and will be an area for further research beyond this dissertation. (It has been suggested that switching to a rectangular mesh may circumvent this difficulty.)

4.7 The Simulation Output Signal

In order to include the largest number of modal frequencies and the least amount of dispersion, the simulation's output signal is be taken as the displacement of the hitpoint as if the hitpoint were a point radiator. Obviously, this is not how sound from a real drum is typically monitored, but because a new set of coefficients for a summation equation must be formulated for each monitoring location, this method was adopted to keep computations within reason.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.8 Damping: Problems in Modeling

The curves in Fig. 6 were used to estimate the value for the radiation damping coefficient C, but the significance of the similarity of these curves was long overlooked. This similarity implies that very little energy is radiated into the air from the drumhead compared to other losses. Later investigation determined that the energy was primarily being transferred into the rim. This rim loss effect was modeled by adding an extra dashpot at the rim of the simulated drumhead. Its resistance value is controlled separately from the radiation parameter; therefore, the damping coefficient at the rim is $C + C_r$, where C is the radiation damping coefficient and C_r is the rim damping coefficient.

Near the end of the research phase of this project, it was clear that frequency-dependent damping was not being simulated correctly, even when the rim damping was included. The lossy effects of bending and rotation of the areas corresponding to each mesh location might be more important than originally thought. These effects could be included by changing stiffness and tension spatial partial derivatives to include a time derivative as well. However, after modifying the difference equation to include these terms, it was determined that the direct finite difference method used in this dissertation for simulation would not work for the modified equation. The problem was that the extra time derivatives required the template to be solved at multiple locations simultaneously. Therefore, it was decided to not pursue these effects with the modified difference equation at this time. However, the simulation was modified by adding an ad hoc filter with a user chosen cutoff frequency (a three point low pass FIR type) to the drum rim in an attempt to improve the simulated sound by removing excessive high frequency energy.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.9 Summary

The most appropriate simulation for this dissertation used the method of finite differences, with modifications to the difference equation to handle symmetry, stability, and rim damping. It was shown that the choice of the monitoring location at the hitpoint provides an output which adequately represents typical outputs obtained from other locations on the drum surface. The initial and boundary conditions developed from physical reality were translated to template modifications.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5. SIMULATION RESULTS

Because of the interrelatedness of the effects of the various enhancements proposed to the Classical Equation (see Eq. 4), usually only one enhancement was simulated at a time. Combinations of enhancements were also simulated. Damping provided the most useful enhancement. The conditions and particular reasons for the simulations along with the results of these simulations and comparisons to actual drum recordings will be explained. Two different analysis techniques were used for analyzing sound files (see Appendix E for details). If not explicitly stated, the PV method was used.

5.1 Classical Membrane Equation Verification by Center Strike Simulation

A set of simulations called the Equation Verification set was performed to make certain that the simulation produced results that matched known theory. The simulation strike point was in the center of the drumhead in order to produce a well-known set of theoretical frequencies. The PV method worked very well above 300 Hz but was unable to resolve the lower frequencies. The MQ method worked well below 450 Hz. Between the two, the entire range was covered. The entire range from 0 to 1000 Hz is shown in Figs. 13 and 14 to justify the use of each method in its range. Some of the frequencies reported in Figs. 13 and 14 agree, but when they differ one must choose between them.

Notice the well-defined frequencies present at 125, 280, and 450 Hz in Fig. 13. The frequencies greater than these are not so well defined. There is crosstalk between the frequencies present which could not be removed by changing the analysis parameters.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

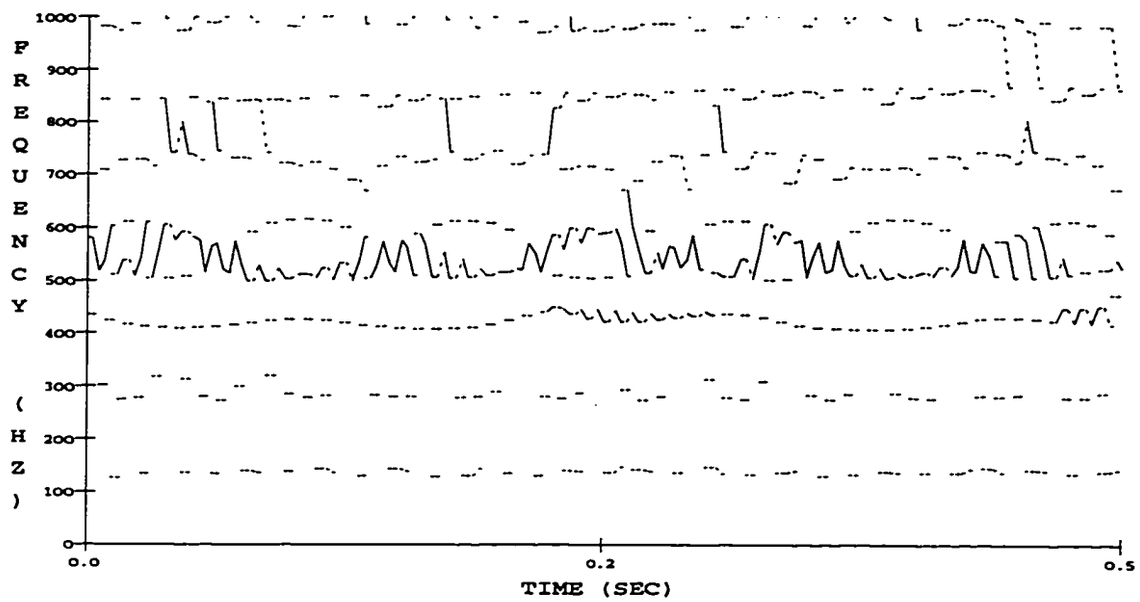

FIGURE 13. Drum simulation, center strikepoint - (MQ method)

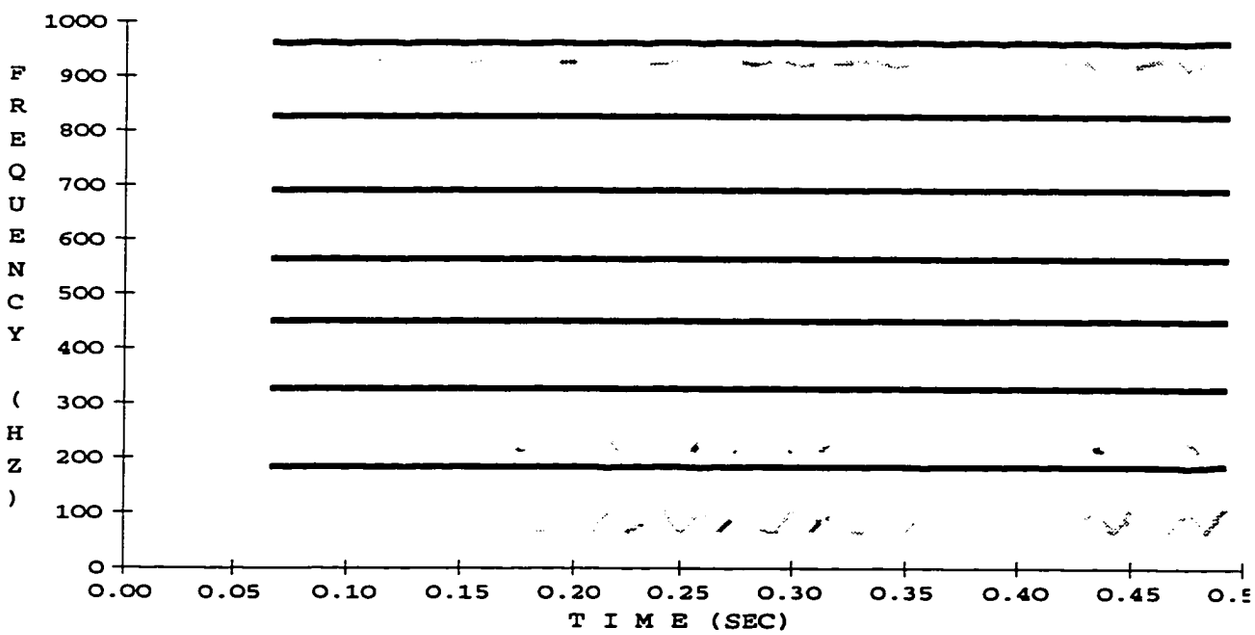

FIGURE 14. Drum simulation, center strikepoint - (PV method)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The PV method was used for Fig. 14 and it resolved the frequencies at 180, 340, 450, 560, 690, 820, and 970 Hz. The lines below 200 Hz show some interference and resolved differently depending upon the analysis parameters chosen.

Table 2 shows the theoretical and simulated frequencies for the simulated drum. Simulated frequencies marked with an asterisk are preferred based upon comments in the previous paragraphs. (The other frequencies are presented to show that in the middle band, 300 to 700 Hz, both techniques give similar answers. This gives some justification for using the better technique at the edges.)

Table 2: Comparison of simulated frequencies to theory

Mode	Theoretical frequency	Simulated frequency MQ analysis	Simulated frequency PV analysis
01	125	125*	?
			180
02	288	280*	320
03	450	450*	450*
		510	
			560
04	612	610*	
			700
05	776	720	820*
		850	
06	940		960*

From Table 2 it can be seen that there is reasonable agreement for four of the first six circular modes. Problems result from extraneous frequencies being present. No particular analysis proved this, but by varying the harmonic filters in the PV analysis, the frequencies between 450 and 800 Hz moved somewhat as if the analysis technique could not resolve them consistently.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The theoretical frequencies listed in Table 2 correspond to circular nodal lines. It is interesting to compare the extraneous frequencies found by analysis with those corresponding to modes including one nodal diameter, i.e., modes 1 1, 1 2, and 1 3. Theoretically these modes should not be excited due to center strikepoint, but due to the nature of the surface mesh and the necessary asymmetry of the initial condition shape, a nodal diameter might have been established. The frequencies for these modes are 200, 360, and 529 Hz. These match up reasonably well with the extraneous frequencies, so that their presence might be attributed to some asymmetry in the mesh. However, enough frequencies matched to assume the simulation was reasonable. Therefore, the difference equation appears to have been correctly implemented. As a further test, the equations were independently checked and implemented using higher-order templates, [Al-Khafaji 28]. No significant differences were noted. The higher-order templates were not used anywhere else because the implementation of the general boundary conditions could not be determined for the fourth-order partial derivative.

5.2 Classical Membrane Struck Off-Center

A set of simulations was performed to produce a baseline for comparisons to the other simulations. Unlike the Equation Verification set, this set used a strike-point three-quarters of the way from the center to the rim, which is more typical of ordinary drum playing. Fig. 15 shows the results of this simulation. The drum was again simulated with an 0 1 mode of 125 Hz, but in this case, all frequencies to 10 kHz were simulated. This simulation was meant to provide a baseline from which to judge whether a particular addition/correction to the classical equation more closely approximated a real drum sound. (Of course, the classical equation's solution does not adequately imitate a real drum except in its modal frequencies. Its main failing is its failure to simulate decay.)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

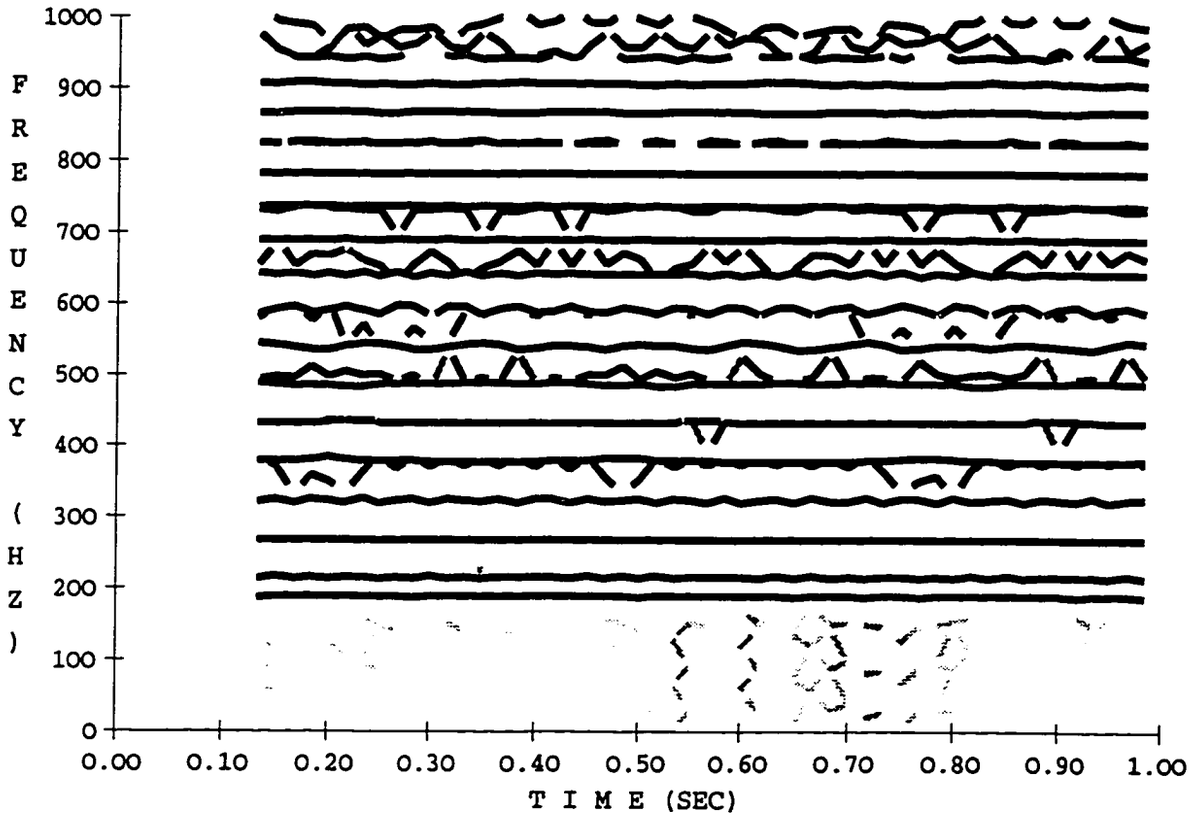

FIGURE 15. Drum Simulation, classical equation with off center strikepoint - (PV method)

The main features to note are the increased density of modal frequencies as compared to Fig. 14, the non-noise like quality of the frequency distribution, and the lack of decay of the signal. The first two characteristics somewhat match a real drum. That the signal does not decay with time violates the behavior of a real drum.

This simulation is what the classical drum equation produces. It is the base from which all the additions/corrections of this dissertation are constructed. The only surprise was that the 0 1 mode at 125 Hz did not register. It is known to exist because it showed up in Fig. 13.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.3 Determination of Simulation Nyquist Frequency

Simulations were made to determine the "spatial-Nyquist" criteria for the simulation method in this dissertation. To produce higher frequencies in the simulation, more meshpoints must be included. This information is necessary to determine the number of radial meshpoints for a particular simulation. Fig. 16 illustrates the theoretical relationship between *M_max*, which is the index of the maximum radial meshpoint, and the frequency simulated.

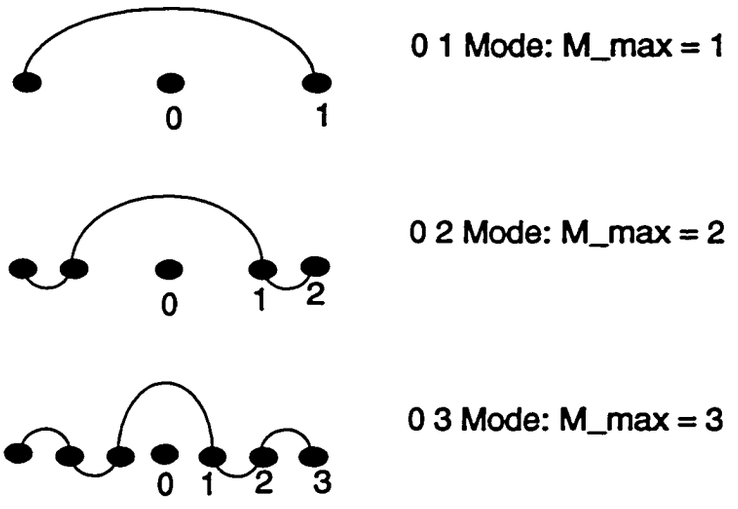

FIGURE 16. Side View of Drumhead - Simplified

From Fig. 16 it is clear that the 01 mode needs three meshpoints to represent one-half cycle, which becomes five meshpoints for one cycle. (Half-cycles overlap by one meshpoint.) Formulating a rule that handles the "extra" meshpoint at the center and the "extra" half-cycle present in each mode gives

$$\text{samples/mode} = \frac{4n+1}{2n-1} \tag{32}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

where *n* is the mode number. By inspection of Fig. 16, *n* is also equal to *M_max*. Taking the limit as *n* becomes large gives a value of 2. This means that for any *M_max*, the Nyquist sampling criteria should be met. Thus, a Spatial-Nyquist criteria based upon this simple model would be

$$\text{maximum simulated frequency} = \text{M_max} \times \text{fundamental frequency} \quad (33)$$

The following table gives the results of five simulations.

Table 3: Radial Meshpoints vs. frequencies simulated (Fund. = 125 Hz)

Number of radial meshpoints (M-max)	Maximum Simulated Frequency (kHz)
10	1.1
50	5.0
100	10.0
200	19.75
225	22.6

Formulating an empirical rule from Table 3 leads to the following

$$\text{maximum simulated frequency} = \text{M_max} \times 100 \quad (34)$$

This is the rule that was followed in other simulations, but this rule did not quite fit the theory because it requires more than the expected number of meshpoints. The explanation for this is that while the radial meshpoints are evenly spaced, the nodal circles on an actual drumhead are not. Therefore, some radial meshpoints are "wasted" in a similar manner as the meshpoint in the center of the drumhead is wasted.

The relationship between the number of angular meshpoints and the maximum simulated frequency was not investigated heavily because the choice was made to determine the number of angular meshpoints, *N_max*, from *M-max*. A test was performed to see what effect changing the angular mesh would produce (see Section 5.8).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.4 Effects of Radiation Loading in Simulation

Simulations were performed to verify the effect of the (mass) air loading on the drum-head. (The true classical equation assumes no air mass, i.e., a vacuum.) The model used for the air mass assumed that each meshpoint area is a simple piston radiator of appropriate radial dimension. The software is set up to modify the air density according to a user-entered coefficient. Any air density relative to normal atmospheric density, from pure vacuum (0) to infinite, is permitted. Spectral analyses of the results for vacuum and normal air density loads for an off-center hit simulation are compared in Figs. 17, 18, and 19. (Note: in Figs. 17, 18, and 19: While the 0 1 mode was predicted to be at 125 Hz, for some unknown reason, the center hit did not excite it enough for its frequency track to resolve.)

Fig. 17 shows the frequencies produced by a classical membrane equation that has the air density set to zero. The frequency information below 120 Hz in this figure is probably an artifact of the analysis routine. Note the three strong frequency components indicated by the dark lines.

Fig 18 shows the a similar result but for an air density of one atmosphere. Again note the three strong frequencies. By comparing Figs. 17 and 18 it is possible to determine the frequency decrease in Fig. 18, but to give better comparison, the two frequency plots are placed together on the Fig. 19.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

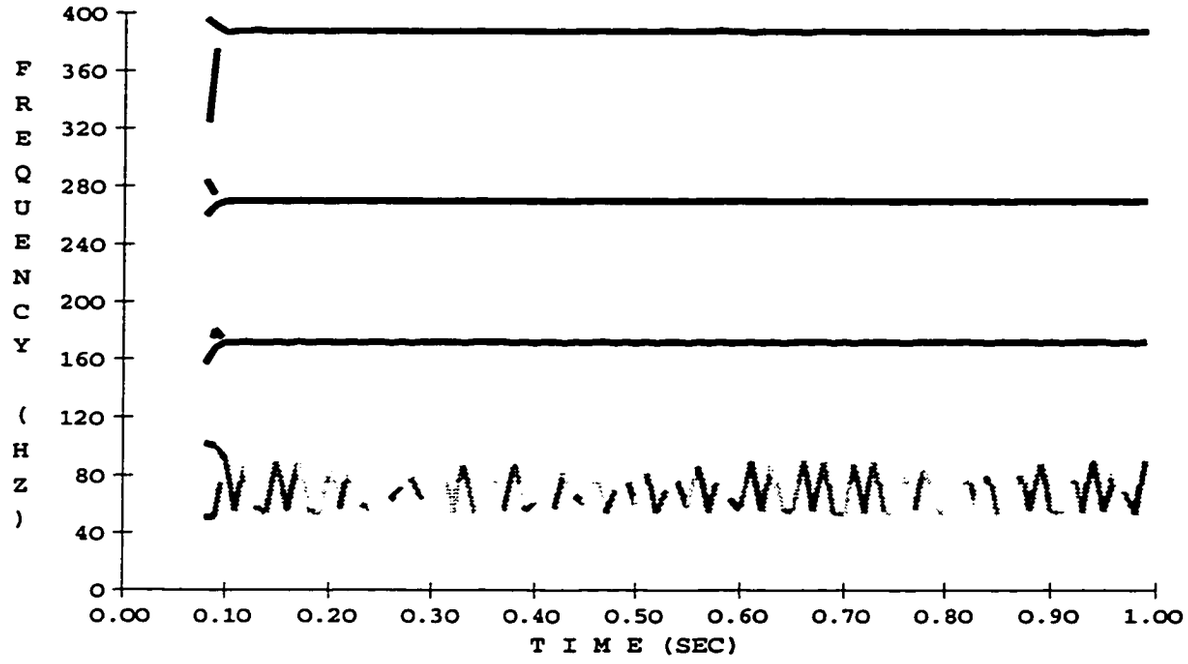

FIGURE 17. Simulated Drum Vacuum Case, center hit

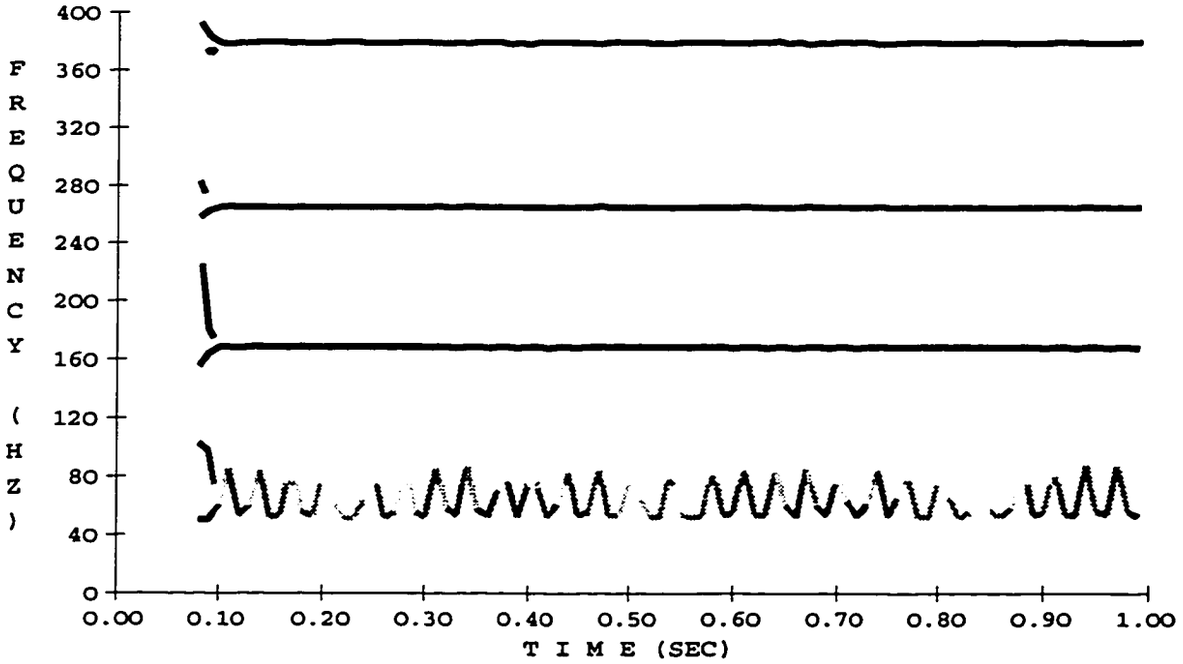

FIGURE 18. Simulated Drum one-atmosphere case, center hit

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

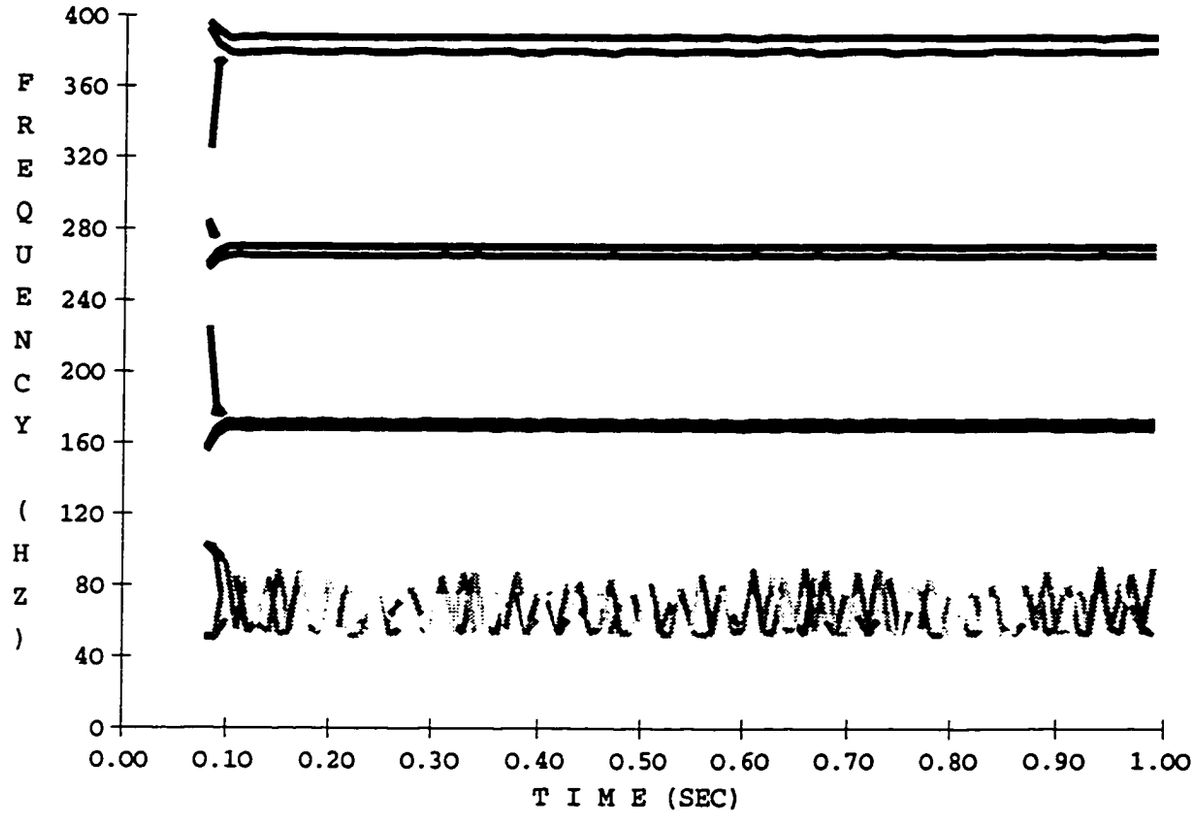

FIGURE 19. Overlay of Figs. 17 and 18

Fig. 19 overlays the vacuum and one-atmosphere time/frequency plots. From Fig. 19 it is calculated that the frequency shift is approximately -2.33% using the vacuum frequency as a reference. (Vacuum frequency = 386 Hz, Air frequency = 377 Hz.)

These simulation results were compared to an experiment on a real drum. The drum was struck off-center in air and in a vacuum using a solenoid to mimic a drumstick. (Hitting a real drum in the center is very difficult.) The spectral analysis for the vacuum case is shown in Fig. 20. The spectral analysis for the air loaded case is shown in Fig. 21. Table 4 lists some frequencies and their deviations. The frequency tracks used in Table 4 were chosen by overlaying the two charts. Where the two simulations produced similar strong

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

modes free from apparent interference, a comparison was made. Lining up the corresponding frequency tracks of the two figures was difficult due to the amplitude decay of the tracks. (Lower frequencies were ignored because the frequency deviations were too small to be measured precisely.)

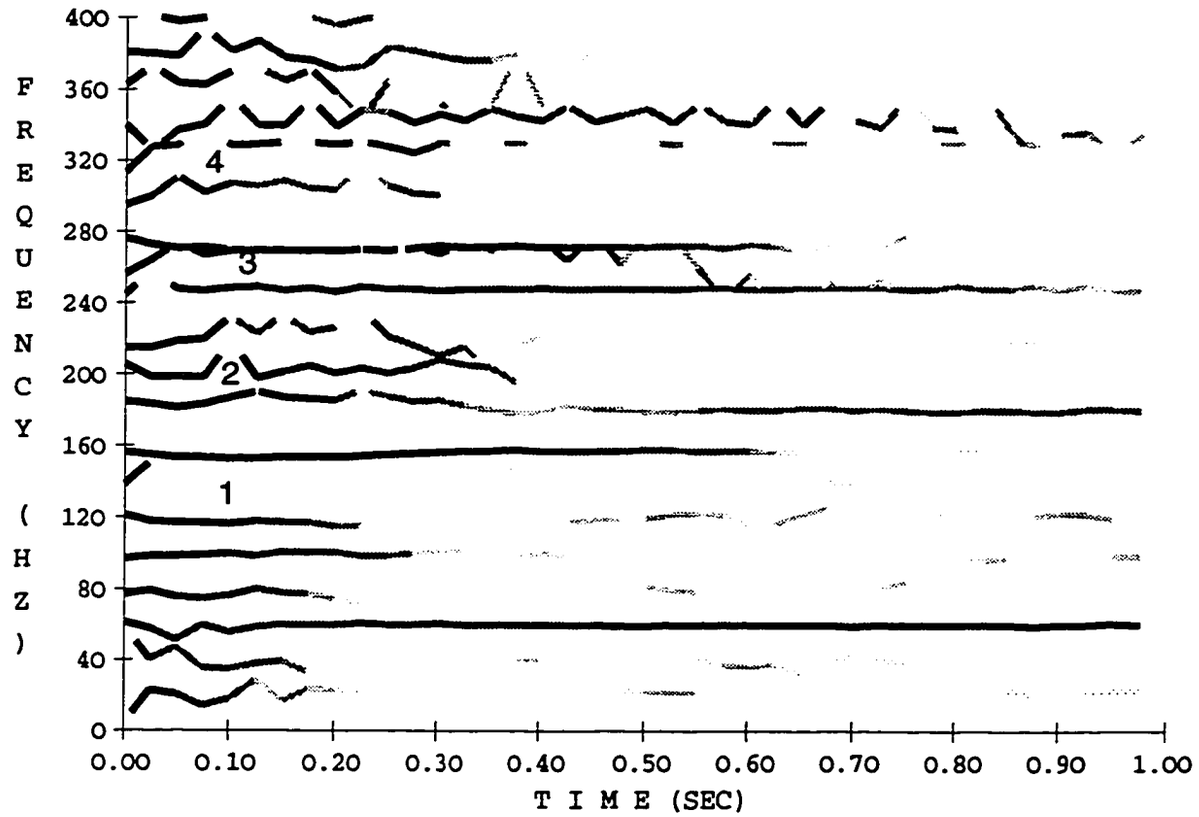

FIGURE 20. Time/frequency spectrum of a real drum struck in a vacuum

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

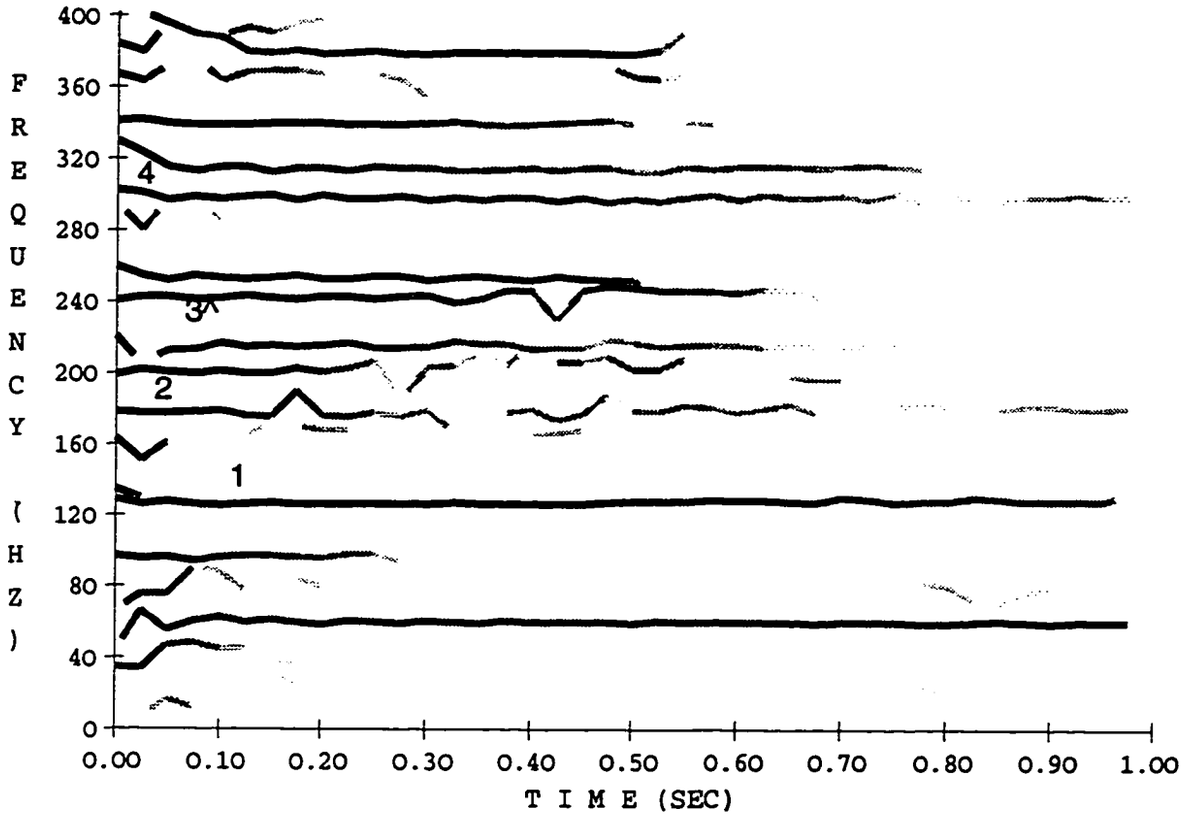

FIGURE 21. Time/frequency spectrum of the same drum struck in air

Table 4: Air Vacuum frequency deviation

Index	Air Frequency	Vacuum Frequency	Percent Deviation (vac. ref.)
1	128	118	+8.47
2	178	184	-3.26
3	242	248	-2.42
4	298	302	-1.32

The deviations shown in Table 4 match those found from Fig. 19, so this aspect of the simulations is verified. Note: Theory tells us that the deviation, using the vacuum as a

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

reference, must be negative, so the result for index 1 must be in error (source unknown) and will be ignored.

5.5 Full Simulation of the Difference Equation

A complete simulation of the difference equation, covering frequencies up to 20 kHz and time up to three seconds, was performed early in the data-gathering phase in order to see how much time was required to complete the simulation. (All simulations were run on a NeXT -040 running Mach-Unix.) Unfortunately this simulation did not complete. The simulation was approximately one-third complete at 720 hours of computer time when it was interrupted by a power failure. Because of the excessive time required to run a full 20 kHz simulation, the maximum frequency was reduced to 10 kHz, and the simulation was repeated. This simulation required 67 hours of computer time. It was obvious from the amount of time these two simulations took that this simulation on this computer would not generally be practical for music generation. Its main use would be the investigation of the physics of a particular drumhead. (There are "hooks" in the simulation to allow very different drumheads and conditions to be modeled.) Fig. 22 shows the spectrum analysis of the 10 kHz fully simulated drum sound. Comparing this to Fig. 2, the frequency density was reasonable throughout the 10 kHz range, but it is obvious that the damping is not at all correct. (Damping is addressed in Section 5.11.)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

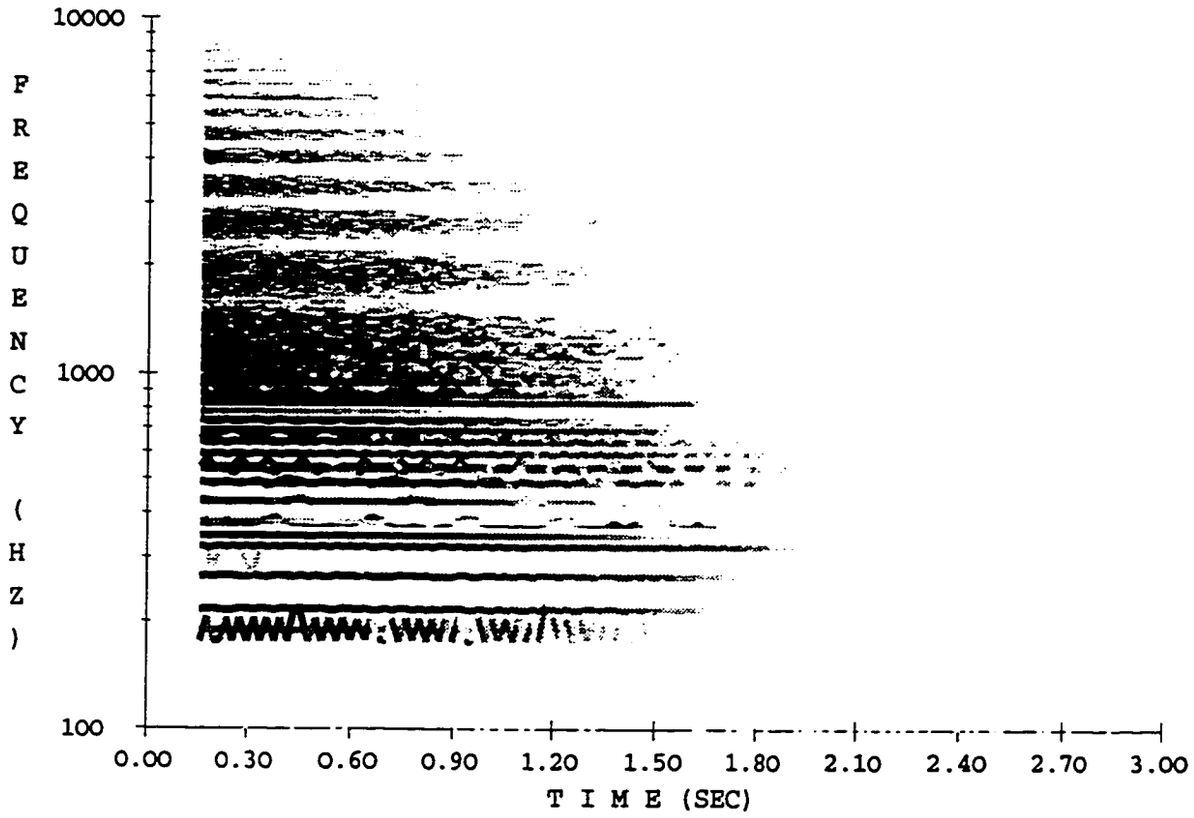

FIGURE 22. Time/frequency spectrum of a 10 kHz simulation

5.6 Effect of Changing the Monitoring Meshpoint for Output

This set of simulations tried a different tack, that of striking the simulated drum at one point and monitoring the drumhead at several other locations. The drum was struck at radial meshpoint 40, and sounds were generated by monitoring at the radial meshpoints 0, 10, 25, 40, and 45. Because rim damping was not used, the sounds produced are more like struck springs rather than drums. Fig. 23 shows the results monitoring the five different meshpoints. Note how the lighter (weaker) frequency tracks line up in each slice. This indicates that the monitoring point is not crucial to the overall structure of the sound, but the differing darkness (strength) of the other tracks indicates that the different

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

meshpoints produce different sounds. From informal listening tests the sounds all have a similar character. (Of course, case 0 has a much lower frequency density.)

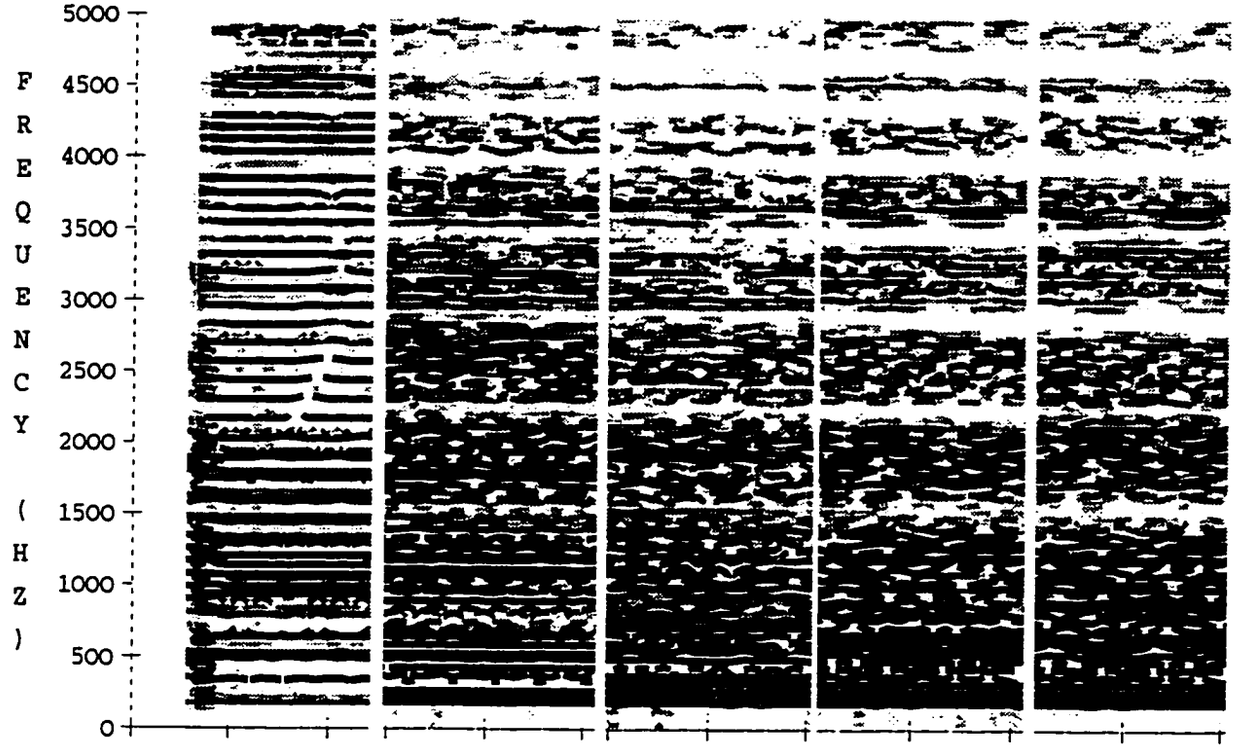

FIGURE 23. Effect of monitoring different meshpoints
slice order (meshpoint): 0, 10, 25, 40, 45

5.7 Effect of Simulating Integrated Radiation Output

When a person listens to a real drum, the sound is a summation or integration of all the energy radiated from the entire drum surface, which depends on the listener/drum orientation. The simulations so far have only monitored the displacement at the hitpoint. This was reasoned to be appropriate because it should have the greatest frequency content. However, the results did not turn out to be completely realistic, so a set of simulations was performed to see if some form of average waveform sampling over the drumhead would make the simulated drum sound better.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

However, no form of average, whether using a few meshpoints, all the meshpoints, adding displacements, or adding accelerations seemed to make much difference in the sound quality. Each of the trials sounded no better or worse than when using only the hitpoint. It is unclear why this is, but because this was not central to the dissertation, it was not pursued further.

5.8 Effect of Altering Squareness of Meshpoint Areas

A set of simulations tested the effect of deviating from the standard "square" value of a general simulation by increasing the angular meshpoint density (see Appendix D program *DSTop.c* for details). Fig. 24 compares the spectra from the standard *N_max* value of the angular mesh with those of a doubled *N_max* value and a quadrupled *N_max* value. In all cases, the number of radial meshpoints corresponded to a spatial Nyquist frequency of about 3 kHz. It is clear from Fig. 24 that increasing the number of angular meshpoints does not increase the spatial Nyquist frequency, nor does it change the general nature of the center-hit spectrum. (Perhaps decreasing the number of angular meshpoints would change things, but corrupting the simulation was not the purpose; it was to make sure the simulation was valid.) Therefore, using the standard angular meshpoint value is justified.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

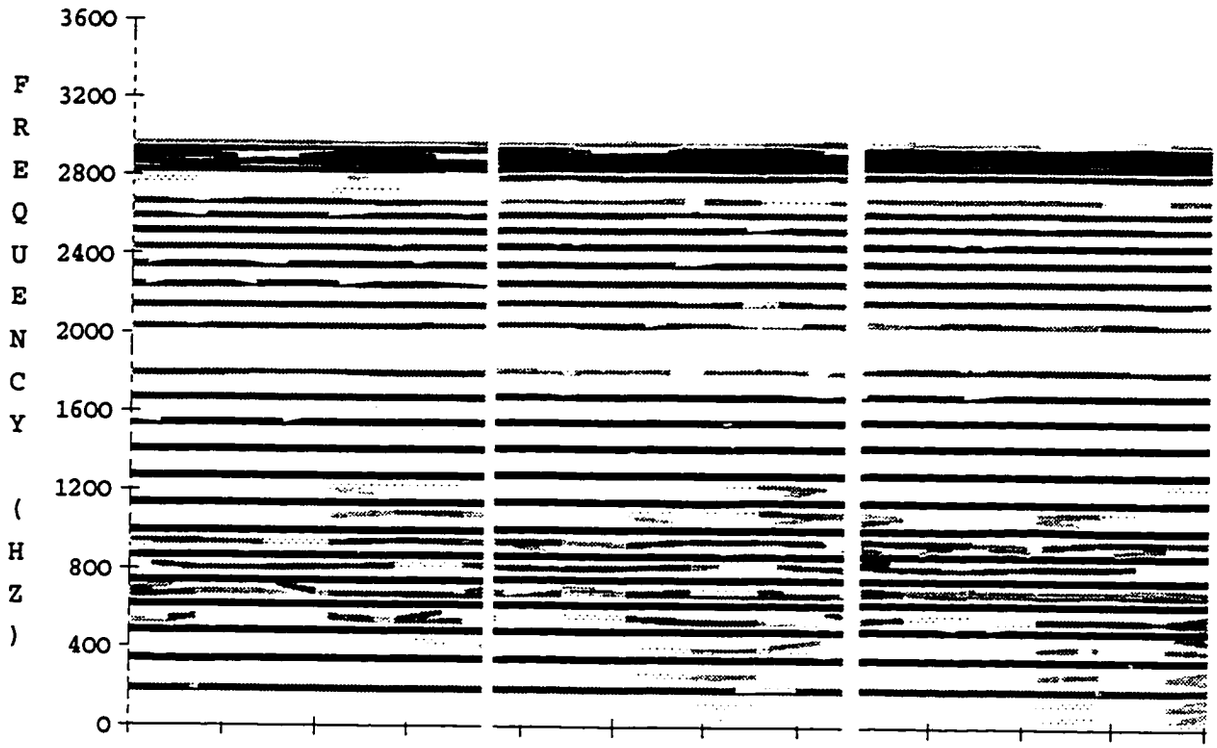

FIGURE 24. Angular mesh value comparison, relative mesh size: 1, 2, 4

5.9 Effect of Stiffness on a Simulation

A set of simulations was performed to verify the effect of stiffness upon the frequencies produced by the simulated drumhead. All the simulations used a center hit to limit the frequencies produced to the 0 n modes. This set was not completed because the simulation became unstable when the stiffness coefficient was set to 0.7 or greater. (This, among other things, such as the inability to produce an actual center hit, prevented a direct comparison with a real drum.) No choice of time increment or other modification seemed to help. It appears that the choice of a polar grid for the simulation resulting in a 1/r problem at the center (Appendix B) may be the culprit, but this has not been confirmed or disproved. However, useful results still occurred.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

What was expected and what generally happened was that increased stiffness caused the membrane frequencies to shift upward. The tables shown below compare the frequencies measured from a simulation to those predicted by theory. The theory comes from Mills [13] and is repeated here in a form using the symbols from this dissertation. (In Tables 5 through 7 stiffness is referenced to that of Mylar. Mylar is the typical material used to make drumheads, and its properties were used as references for constants in the simulation software.)

$$f_s = f_u\left(1 + \frac{f_u^2 4\pi^2 D}{c^4 \rho_s}\right)^{1/2} \qquad (35)$$

where f_S is the shifted frequency, f_U is the unshifted frequency, D is the stiffness, c is the speed of sound in the drumhead given by $c = (T/\rho_S)^{1/2}$, and ρ_S is the surface density.

Table 5: Comparison of membrane frequencies to 10% stiffness

Membrane frequencies	Simulated		Theoretical		Match?
	Shifted Frequency	Ratio	Shifted Frequency	Ratio	
865	872	1.008	877	1.014	Y
998	1012	1.014	1017	1.019	Y
1130	1158	1.025	1157	1.024	Y
1270	1312	1.033	1308	1.030	Y

Table 6: Comparison of membrane frequencies to 25% stiffness

Membrane frequencies	Simulated		Theoretical		Match?
	Shifted Frequency	Ratio	Shifted Frequency	Ratio	
865	879	1.016	895	1.035	nY
998	1040	1.042	1044	1.046	Y
1130	1207	1.068	1196	1.059	nY
1270			1363	1.073	

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 7: Comparison of membrane frequencies to 50% stiffness

Membrane frequencies	Simulated		Theoretical		Match?
	Shifted Frequency	Ratio	Shifted Frequency	Ratio	
865	914	1.057	924	1.068	nY
998	1088	1.090	1088	1.090	Y
1130	1277	1.130	1259	1.114	N
1270			1451	1.142	

The membrane frequencies were first measured (using PV analysis) from a simulation that had a stiffness set to zero. These frequencies, as well as other measured frequencies, are subject to an estimation uncertainty of +/- 7.5 Hz due to the line widths in the measurement plots. The theoretical shifted frequencies are based upon the membrane frequencies.

Tables 5 - 7 show good agreement between the theory and the simulations. The last column in each table merely indicates whether the +/- 7.5 Hz deviation in the simulated frequencies is enough to bring the simulated frequencies to match the theoretical frequencies. A 'Y' indicates that this is so, an 'nY' indicates that a match occurs with the theoretical frequency after it has been adjusted by applying the +/- 7.5 Hz uncertainty to the membrane frequencies, and 'N' indicates no match.

Because nearly all frequencies match, this aspect of the simulation is well proven.

5.10 Effect of Time-Variant Tension on Modal Frequencies of Simulation

A simulation was performed to determine whether the variable tension term, discussed in Section 2.2.1, would produce frequency shifts that would match those produced by real drums. Figs. 25 and 26 show analyses of real drums hit hard, and Fig. 27 shows the analysis of a simulation where time-variant tension has been introduced.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

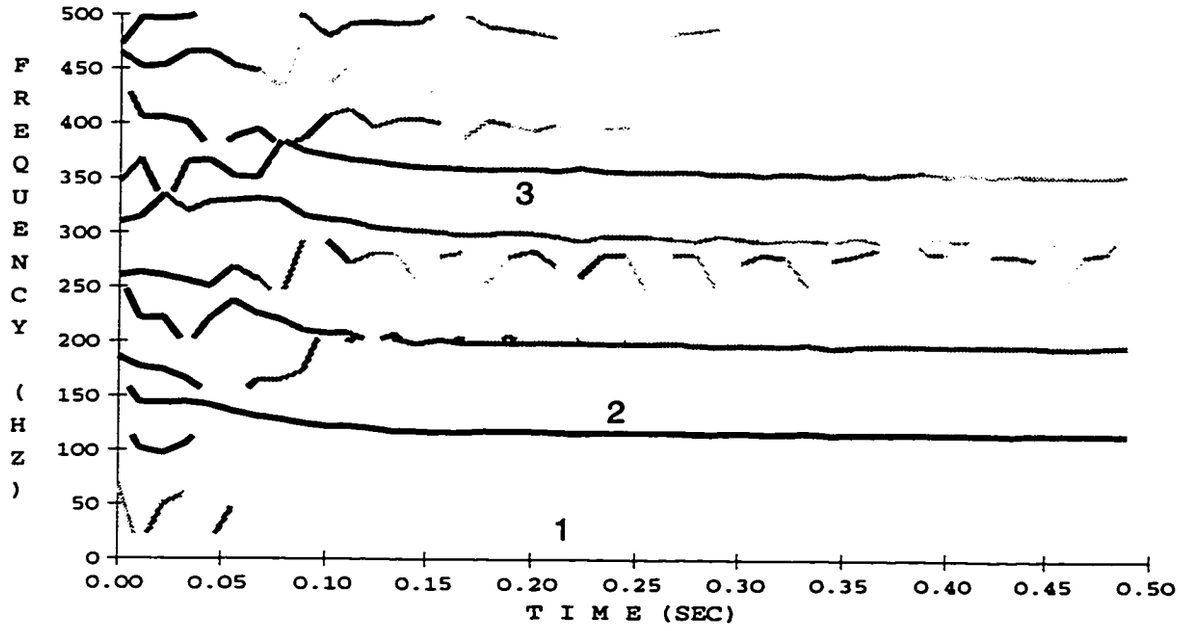

FIGURE 25. Time/frequency spectrum of a real drum struck hard (same recording as Fig. 3)

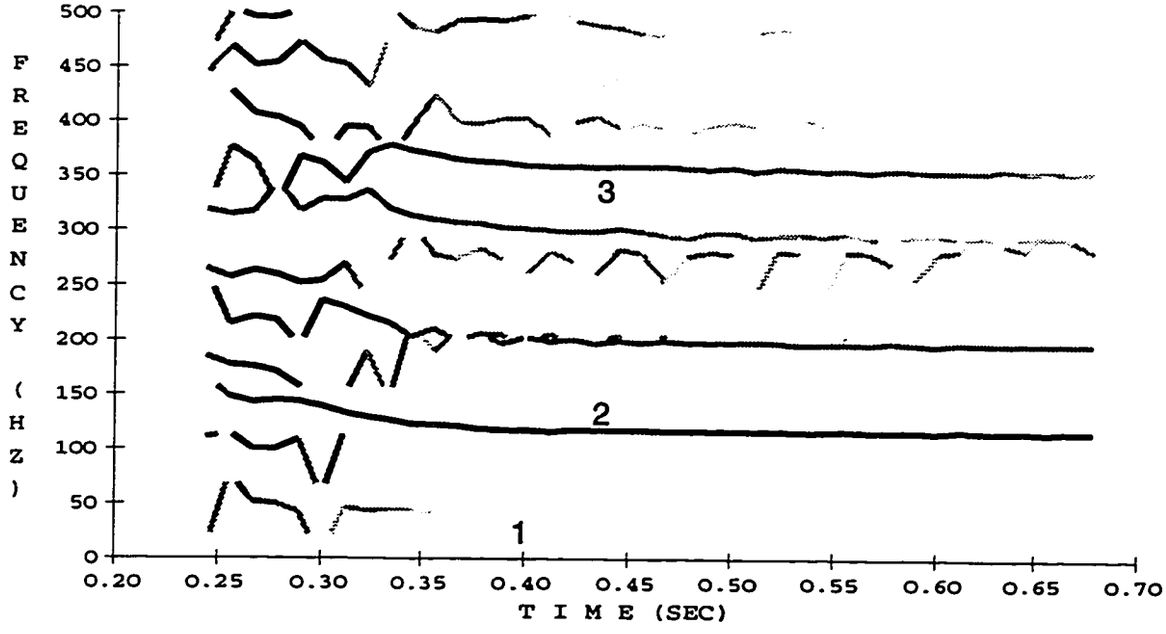

FIGURE 26. Time/frequency spectrum of the same drum for another hard strike

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

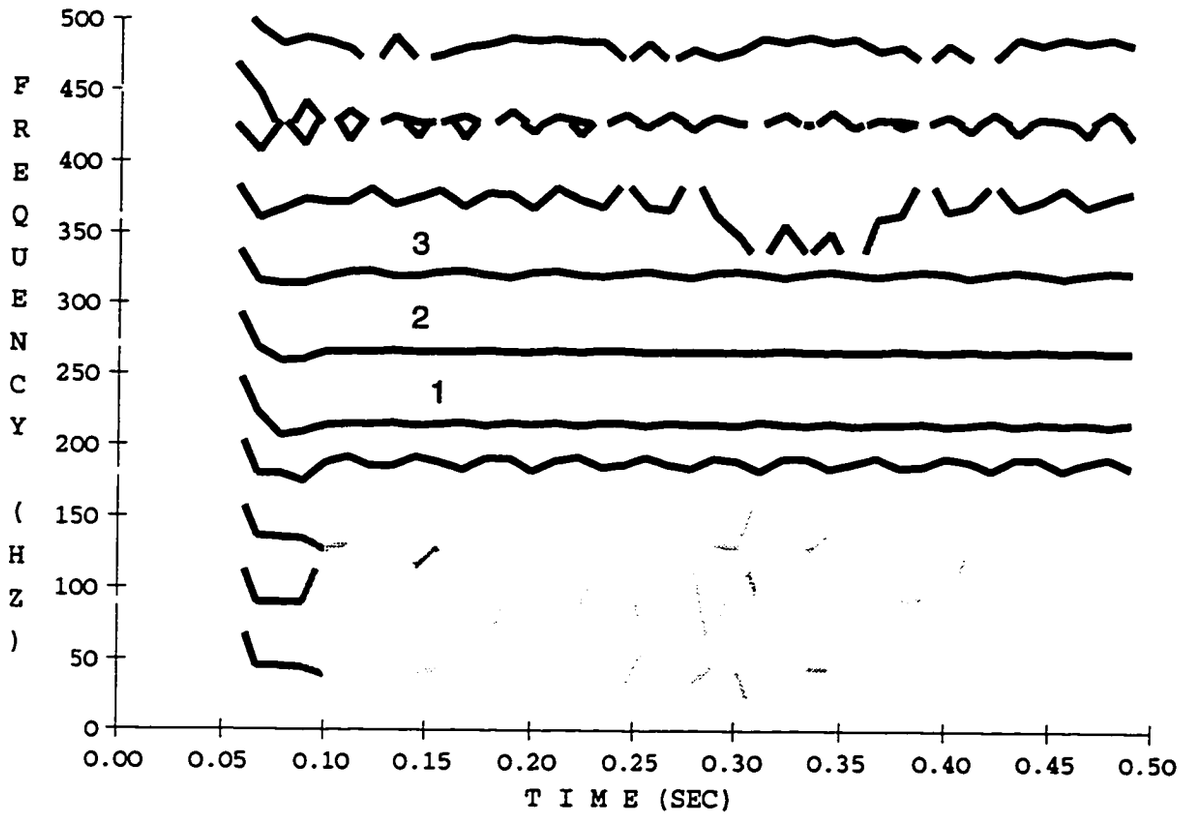

FIGURE 27. Time/frequency spectrum of a simulated drum struck hard

Table 8 is generated from information gathered from Figs. 25, 26 and 27. The labels 1, 2, 3 indicates which frequency tracks are used in Table 8. (The tracks were chosen based upon information from a MQ analysis, but the MQ analysis did little but to identify which tracks showed frequency change. The analysis was repeated using the PV method whose base frequency was chosen to emphasize the tracks identified by the MQ method.)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 8: Frequency Shift Comparison

Key	Figure 25			Figure 26			Figure 27		
	Start Freq.	End Freq.	Delta Freq.	Start Freq.	End Freq.	Delta Freq.	Start Freq.	End Freq.	Delta Freq.
1	186	116	70	184	116	67	248	215	33
2	248	198	50	248	197	52	295	266	28
3	429	357	72	429	354	75	336	318	18
settle time	0.2 sec			0.2 sec			0.1 sec		

The similarity of the first two sections of Table 8 indicate that the data from the real drum is reliable. (The data were taken approximately two years apart.) The third section gives the frequencies from the analysis of the simulation. The variable tension did not make measurable frequency shifts at the lower frequencies, as in real drums, nor was the frequency shift as great. However, adding the variable tension to the model did produce some frequency shift in the data and thus appears to be needed. The variable tension did not account for all of the frequency shift, but without the variable tension there was no frequency shift present (see Fig. 15).

The only other parameter known to shift a frequency is *stiffness*, which can change values during an impact. (Impact stiffness can be imagined by thinking about hitting a closed-cell foam sheet, or a mixture of corn starch and water. Both are quite hard under a quick impact, but are soft under a gentle pressure.) A drumhead may well exhibit impact stiffness, which would account for the extra frequency shift and perhaps the longer relaxation time. This effect was outside of the scope of this dissertation and so was not pursued.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.11 Effect of Damping on a Simulation

Three sets of simulations were performed in order to test the effects of three different types of damping: radiation, membrane bending, and rim absorption. Unlike stiffness or tension, it was impossible to separately infer the various damping types from the analysis of recordings of real drumheads.

The first enhancement, the original one for the dissertation, uses the model of radiation damping. It models the energy lost to the air, which produces the sound heard. From theory, it is a combination of several effects, but their aggregation can be modeled as a single term.

The second enhancement was an empirical attempt to model the damping due to the bending of a stiff membrane, (see Chapter 2). The way this was done was to place a lowpass filter at the rim of the drum. Its cutoff frequency was set low enough to have the attenuation region decrease the energy in the sound produced.

The third enhancement was written after a sound recording showed the unexpected major effect of the rim. While the rim was known to absorb energy, the large amount of energy absorbed was not expected. This effect was handled by placing dashpots, radiation type, one meshpoint in from the rigid rim. These were used to model the energy flow into the rim. This set had to be tested in conjunction with the radiation damping, and so could be viewed as a third enhancement to the damping.

5.11.1 Simple lowpass rim damping

Before discussing the simulations, which were successful, a brief presentation will be made of a method that, while unsuccessful, is still included in the simulation code, in order to warn others from treading down this apparently useless path. After much disap-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

pointment in creating a realistic damping effect, it was decided to try an ad hoc approach. This was done by attaching a low-pass filter to the rim of the simulation (see Appendix D under the *DSTop.c* program). The filter's cutoff frequency was set so that higher frequencies produced by the drum, those above 1 kHz or so, were attenuated at the rim at a rate of -20 dB/decade. This method causes the higher frequencies to damp out more rapidly than the lower frequencies. Depending upon the user-chosen filter cutoff frequency, the sound produced by the simulation changed, but all simulations sounded like a "quick pop" followed by "weak ringing." Fig. 28 shows a typical spectrum plot. The bulges at 1200, 2200, 2700, 3500, and 4000, and the long tail at 4500 (the spacial nyquist frequency) perhaps cause the ringing sound. Because this method did not produce good results, it was ignored in other simulations; i.e., this filter was switched out of the simulation.

The long tail at the spatial Nyquist frequency of 4500 Hz is not likely to be aliasing because the digital filter was designed to prevent this, but it probably is caused by an unexpected limitation of the three point FIR filter. (The extensive changes needed in the simulation software to implement a longer filter precluded using such a filter but a longer filter would probably have removed this effect.)

5.11.2 Effect of combined rim and radiation damping

The use of two types of damping together, radiation and rim damping, produced a significant improvement in the quality of the simulation. The decay slope of the RMS amplitude (see Fig. 6) was used as a starting point for computing the radiation damping coefficient C. The value for C, derived from that graph, proved to be too large so it was reduced and the rim coefficient C_r was increased until the simulated drum sounded as similar as possible to a real drum sound. The two coefficients were chosen by comparing (by ear) the simulated drum sound to a real drum sound.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

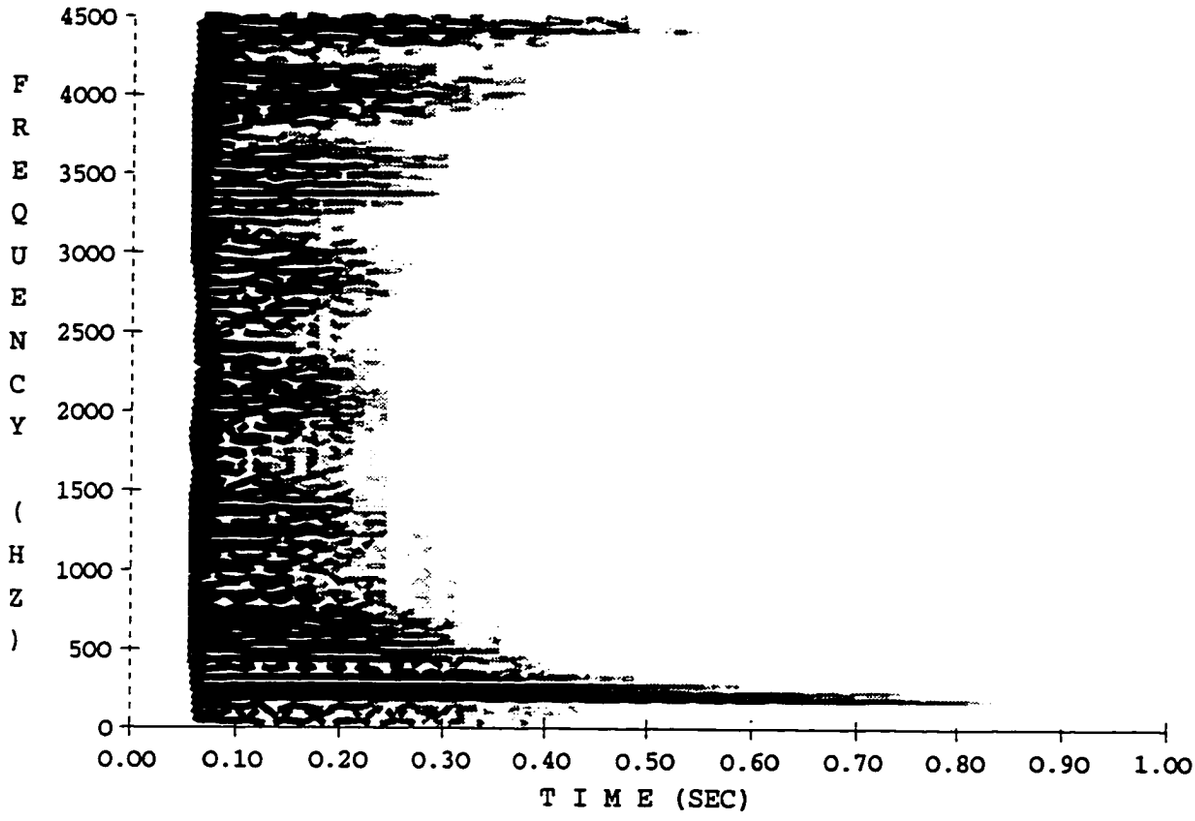

FIGURE 28. Time/frequency spectrum for a simulation using empirical damping by placing a low pass filter at the rim

The spectrum of the combination which produced the "best results," in the sense of giving the most drum-like sound, is graphed in Fig. 29. Compare the overall energy envelope of Fig 29. with that of Fig. 30.

Note that there are amplitude "bulges" in Fig. 29, but they damp out more quickly than the previous case where the low-pass rim filter alone was used (see Fig. 28), and the high frequencies do not persist. Surprisingly, various simulations showed that the sound produced was very sensitive to the radiation coefficient value while not to the rim coefficient value.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

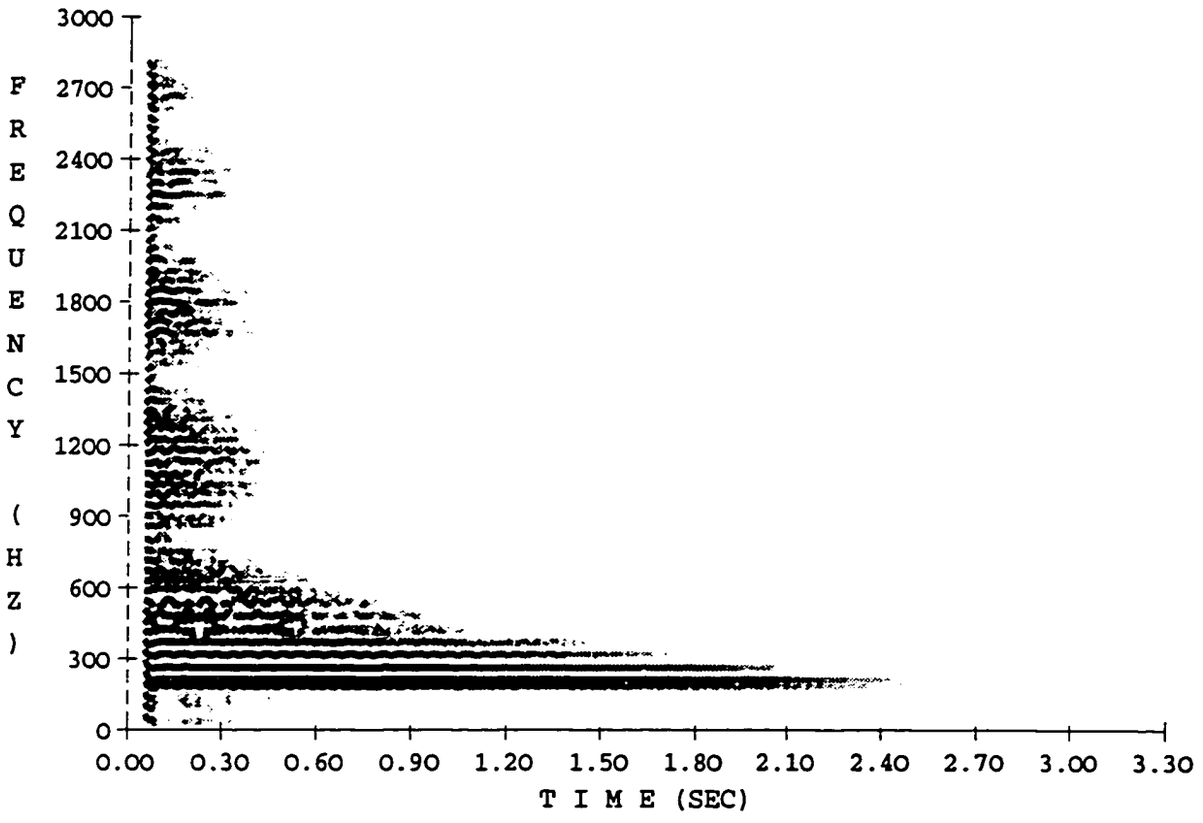

FIGURE 29. Time/frequency spectrum for a simulation using radiation and rim damping

A possible cause of the bulges is that the waveform is sampled at the hitpoint. Sections 5.6 and 5.7 show the results of changing the sampling location.

All in all, damping did not produce sounds that accurately resembled a real drum, but they were a huge improvement over no damping at all. In fact, damping produced the best single improvement to the monitored sound over all the other perturbations investigated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

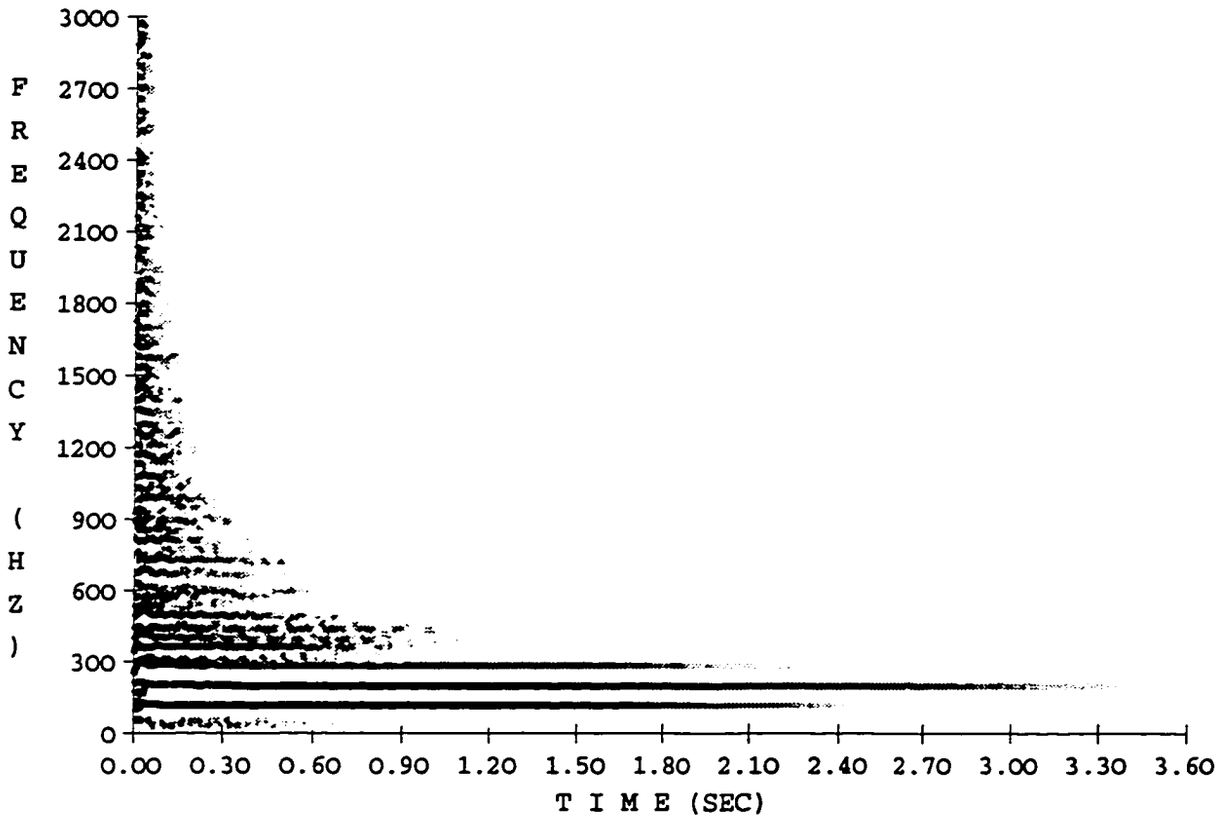

FIGURE 30. Time/frequency spectrum for a real drum - for comparison to Fig. 29 (a subsection of Fig. 2)

5.12 Summary

All simulations were generally successful with the most successful being those related to the stiffness, tension and damping.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6. SUMMARY AND CONCLUSIONS

6.1 Accomplishments and Failures

As can be construed from Chapter 5, the simulation of the wave equation described in this dissertation met with general success as follows:

- Adding air loading matched the predictions of theory and experiment.
- Adding stiffness matched the predictions of theory.
- Time-variant tension produce results similar to a hard hit drum.
- Adding radiation and rim damping qualitatively matched physical experiments.
- Monitoring the simulation output at a meshpoint corresponding to the strike point was proven valid.
- The choice of the density of the simulation mesh was proven valid.

6.2 Future Work

The software developed in this dissertation should prove useful in the simulation of different types of drumheads - ones more complex than the one used in this dissertation. To this end, some improvements and further work need to be done as follows:

- Determine the cause of the instability due to stiffness and correct it. This will require determining, if possible, the actual stability criteria for a fourth-order partial differential equation in the finite difference domain. (Correcting the damping will increase the equation order to five.)
- There are also some minor improvements to the code that will make other drumheads more easy to simulate. These generally take the form of converting

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

constants to user-supplied variables, and changing the coefficient matrix to permit unsymmetrical values in the surface.

- Modify the difference equation or its initialization so that its simplest embodiment, corresponding to the simple two-dimensional wave equation, yields exactly the correct frequencies.
- Determine methods for substantial computational speed enhancement.
- Improve the simulation of damping effects. The energy lost to bending heats up the drumhead. This heat then radiates to the surrounding air. Sharper bends generate more heat and thus cause greater energy loss. No references could be found that describe this. The only information came from Abrahams [34]. He offered the equations below from his current research. They have the correct features to mimic an energy loss that is dependent upon frequency. The final equation shows how Eq. 33 would be modified to include these terms.

$$\text{bending damping} = R\frac{\partial}{\partial t}(\nabla^4 w) \tag{36}$$

$$\text{stretching damping} = S\frac{\partial}{\partial t}\left(\nabla^2 w\right) \tag{37}$$

$$(-D + R\frac{\partial}{\partial t})\nabla^4 w + (T_0 + \Delta T + S\frac{\partial}{\partial t})\nabla^2 w = \bar{m}\frac{\partial^2 w}{\partial t^2} + C\frac{\partial w}{\partial t} \tag{38}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix A: DERIVATION OF "DEL TO THE FOURTH"

This derivation is given because no references could be found to show it. It is used in the stiffness (plate) portion of the drumhead equation.

$$\nabla^4 w = \nabla^2 \left(\nabla^2 w \right) \tag{39}$$

Expanding this using polar coordinates and dropping the "w" gives:

$$\nabla^2 \left(\nabla^2 w \right) = \left[\frac{\partial^2}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} + \frac{1}{r}\frac{\partial}{\partial r} \right] \left(\frac{\partial^2}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} + \frac{1}{r}\frac{\partial}{\partial r} \right) \tag{40}$$

$$= \frac{\partial^2}{\partial r^2}\left(\frac{\partial^2}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} + \frac{1}{r}\frac{\partial}{\partial r} \right) + \left(\frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} \right) \left(\frac{\partial^2}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} + \frac{1}{r}\frac{\partial}{\partial r} \right) + \left(\frac{1}{r}\frac{\partial}{\partial r} \right) \left(\frac{\partial^2}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} + \frac{1}{r}\frac{\partial}{\partial r} \right) \tag{41}$$

$$= \frac{\partial^2}{\partial r^2}\frac{\partial^2}{\partial r^2} + \frac{\partial^2}{\partial r^2}\left(\frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} \right) + \frac{\partial^2}{\partial r^2}\left(\frac{1}{r}\frac{\partial}{\partial r} \right)$$

$$+ \left(\frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} \right)\frac{\partial^2}{\partial r^2} + \left(\frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} \right) \left(\frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} \right) + \left(\frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} \right) \left(\frac{1}{r}\frac{\partial}{\partial r} \right)$$

$$+ \left(\frac{1}{r}\frac{\partial}{\partial r} \right)\frac{\partial^2}{\partial r^2} + \left(\frac{1}{r}\frac{\partial}{\partial r} \right) \left(\frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} \right) + \left(\frac{1}{r}\frac{\partial}{\partial r} \right) \left(\frac{1}{r}\frac{\partial}{\partial r} \right) \tag{42}$$

Using the chain rule and collecting terms gives the final equation.

$$\nabla^4 w = \left[\frac{\partial^4}{\partial r^4} + \frac{2}{r^2}\frac{\partial^4}{\partial r^2 \partial \theta^2} + \frac{1}{r^4}\frac{\partial^4}{\partial \theta^4} + \frac{2}{r}\frac{\partial^3}{\partial r^3} - \frac{2}{r^3}\frac{\partial^3}{\partial r \partial \theta^2} - \frac{1}{r^2}\frac{\partial^2}{\partial r^2} + \frac{4}{r^2}\frac{\partial^2}{\partial \theta^2} + \frac{1}{r^3}\frac{\partial}{\partial r} \right] (w) \tag{43}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix B: FINITE DIFFERENCE EQUATION

The basic finite-difference equation solved in the simulations done for this dissertation is derived below. Use of this equation is described in Chapter 4. For clarity, some information from the main body of the dissertation will be repeated here. Eq. 33 is the starting point and is repeated here with $T = T_0 + \Delta T$.

$$-D\nabla^4 w + T\nabla^2 w = \overline{M}\frac{\partial^2 w}{\partial t^2} + C\frac{\partial w}{\partial t}$$

where D is the stiffness, T_0 is the static tension, ΔT is the time variant tension, $\overline{M}$ is the area density, C is the damping coefficient, and w is the deflection function.

To transform the continuous-time equation above into a discrete-time equation, it is necessary to provide an index for each variable. The radius will be indexed by a lower case m with $m=0$ being the center of the drum and $m=M$ being the rim. (Do not confuse $\overline{M}$ with M.) The angle will be indexed by a lower case n with $n=0$ being zero degrees and $n=N$ being 180 degrees. (As mentioned in the Section 4.4 the drum has even symmetry across a line through the strikepoint and the center of the drum. For the finite difference equation, zero degrees is chosen to be the radial line from the center of the drum through the strikepoint.) The time is indexed by a lower case i with $i=0$ being the present value of time, $i-\#$ being previous time and $i+\#$ being future time.

The differential quantities become delta (Δ) quantities with actual numerical values being determined from input arguments. These become

$$\partial r \rightarrow \Delta r = \frac{\text{radius}}{M} \qquad \text{and} \qquad \partial\theta \rightarrow \Delta\theta = \frac{\pi}{N}$$

The radius and angle of a particular meshpoint are

$$r = m\Delta r \qquad \text{and} \qquad \theta = n\Delta\theta$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The Δt value is chosen to give stability (convergence).

Substituting for the ∇ operators of Eq. 33 (see Eqs. 40 and 43) gives

$$-D\left(\frac{\partial^4}{\partial r^4}+\frac{2}{r^2}\frac{\partial^4}{\partial r^2\partial\theta^2}+\frac{1}{r^4}\frac{\partial^4}{\partial\theta^4}+\frac{2}{r}\frac{\partial^3}{\partial r^3}-\frac{2}{r^3}\frac{\partial^3}{\partial r\partial\theta^2}-\frac{1}{r^2}\frac{\partial^2}{\partial r^2}+\frac{4}{r^2}\frac{\partial^2}{\partial\theta^2}+\frac{1}{r^3}\frac{\partial}{\partial r}\right)w$$

$$+T\left(\frac{\partial^2}{\partial r^2}+\frac{1}{r^2}\frac{\partial^2}{\partial\theta^2}+\frac{1}{r}\frac{\partial}{\partial r}\right)w=\bar{M}\frac{\partial^2 w}{\partial t^2}+C\frac{\partial w}{\partial t} \qquad (44)$$

Combining and folding in terms involving w gives

$$-D\frac{\partial^4 w}{\partial r^4}-\frac{2D}{r}\frac{\partial^3 w}{\partial r^3}+(T+\frac{D}{r^2})\frac{\partial^2 w}{\partial r^2}+(\frac{T}{r}-\frac{D}{r^3})\frac{\partial w}{\partial r}-\frac{2D}{r^2}\frac{\partial^4 w}{\partial r^2\partial\theta^2}+\frac{2D}{r^3}\frac{\partial^3 w}{\partial r\partial\theta^2}$$

$$+(\frac{T}{r^2}-\frac{4D}{r^4})\frac{\partial^2 w}{\partial\theta^2}-\frac{D}{r^4}\frac{\partial^4 w}{\partial\theta^4}=\bar{M}\frac{\partial^2 w}{\partial t^2}+C\frac{\partial w}{\partial t} \qquad (45)$$

Multiplying through by r^4 yields

$$-Dr^4\frac{\partial^4 w}{\partial r^4}-2Dr^3\frac{\partial^3 w}{\partial r^3}+(Tr^4+Dr^2)\frac{\partial^2 w}{\partial r^2}+(Tr^3-Dr)\frac{\partial w}{\partial r}-2Dr^2\frac{\partial^4 w}{\partial r^2\partial\theta^2}+2Dr\frac{\partial^3 w}{\partial r\partial\theta^2}$$

$$+(Tr^2-4D)\frac{\partial^2 w}{\partial\theta^2}-D\frac{\partial^4 w}{\partial\theta^4}=\bar{M}r^4\frac{\partial^2}{\partial t^2}(w)+Cr^4\frac{\partial w}{\partial t} \qquad (46)$$

Before substituting finite difference templates for the partial derivatives, some mixed derivatives need to be derived. To be complete, all the templates for the partial derivatives are listed below. The plain templates and the procedure for deriving the mixed templates come from Al-Khafaji & Tooley [28]. All the templates have errors of order Δ^2 and use 'x' and 'y' as dummy variables. The overbar indicates the point in the template corresponding to the present value of the index. (Left-to-right and bottom-to-top the index values are previous, present, next.)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$\frac{\partial}{\partial x} = \frac{1}{2\Delta}\left[-1\ \bar{0}\ 1\right]$$

$$\frac{\partial^2}{\partial x^2} = \frac{1}{\Delta^2}\left[1\ -\bar{2}\ 1\right]$$

$$\frac{\partial^3}{\partial x^3} = \frac{1}{2\Delta^3}\left[-1\ 2\ \bar{0}\ -2\ 1\right]$$

$$\frac{\partial^4}{\partial x^4} = \frac{1}{\Delta^4}\left[1\ -4\ \bar{6}\ -4\ 1\right]$$

$$\frac{\partial^4}{\partial x^2 \partial y^2} = \frac{1}{\Delta^4}\begin{bmatrix} 1 & -2 & 1 \\ -2 & \bar{4} & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

$$\frac{\partial^3}{\partial x \partial y^2} = \frac{1}{2\Delta^3}\begin{bmatrix} 1 & -2 & 1 \\ 0 & \bar{0} & 0 \\ -1 & 2 & -1 \end{bmatrix} \qquad (47)$$

In substituting in the templates, some notation must be used to refer to which mesh location is being used. For convenience, only the subscripts are used to locate a mesh point, and all mesh points are referenced relative to the mesh point presently being computed. The equations below should clarify matters.

$$w(r, \theta, t) \rightarrow w(r_m, \theta_n, t_i) = w(m, n, i) \qquad (48)$$

m ⇒ present radial mesh location

m + # ⇒ # mesh points toward the rim

m − # ⇒ # of mesh points toward the center

n ⇒ present angular mesh location

n + # ⇒ # mesh points toward 180 degrees

n − # ⇒ # mesh points toward zero degrees

i ⇒ present time value/location

i + 1 ⇒ future time

i − 1 ⇒ previous time (49)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Substituting in the templates with the revised notation

$$-\frac{Dr^4}{\Delta r^4}(w(m-2,n,i)-4w(m-1,n,i)+6w(m,n,i)-4w(m+1,n,i)+w(m+2,n,i))$$

$$-\frac{Dr^3}{\Delta r^3}(-w(m-2,n,i)+2w(m-1,n,i)-2w(m+1,n,i)+w(m+2,n,i))$$

$$+\frac{Tr^4+Dr^2}{\Delta r^2}(w(m-1,n,i)-2w(m,n,i)+w(m+1,n,i))$$

$$+\frac{Tr^3-Dr}{2\Delta r}(-w(m-1,n,i)+w(m+1,n,i))$$

$$-\frac{2Dr^2}{\Delta r^2\Delta\theta^2}\begin{pmatrix}(w(m-1,n-1,i)-2w(m-1,n,i)+w(m-1,n+1,i))\\-2w(m,n-1,i)+4w(m,n,i)-2w(m,n+1,i)\\+w(m+1,n-1,i)-2w(m+1,n,i)+w(m+1,n+1,i)\end{pmatrix}$$

$$+\frac{Dr}{\Delta r\Delta\theta^2}\begin{pmatrix}-w(m-1,n-1,i)+2w(m-1,n,i)-w(m-1,n+1,i)\\+w(m+1,n-1,i)-2w(m+1,n,i)+w(m+1,n+1,i)\end{pmatrix}$$

$$+\frac{Tr^2-4D}{\Delta\theta^2}(w(m,n-1,i)-2w(m,n,i)+w(m,n+1,i))$$

$$-\frac{D}{\Delta\theta^4}(w(m,n-2,i)-4w(m,n-1,i)+6w(m,n,i)-4w(m,n+1,i)+w(m,n+2,i))$$

$$=\frac{\overline{M}r^4}{\Delta t^2}(w(m,n,i-1)-2w(m,n,i)+w(m,n,i+1))$$

$$+\frac{Cr^4}{2\Delta t}(-w(m,n,i-1)+w(m,n,i+1)) \qquad (50)$$

Next, the equation is rearranged to put all the terms with "i+1" in them on the left-hand side of the equation. The "i+1" or future term is computed from past and present data. The representation below forms the basis for the *setuptemplate.c* subroutine in Appendix D and is the final form of the difference equation.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$w(m,n,i+1)\left[\frac{\bar{M}r^4}{\Delta t^2}+\frac{Cr^4}{2\Delta t}\right] = w(m,n,i-1)\left[\frac{-\bar{M}r^4}{\Delta t^2}+\frac{Cr^4}{2\Delta t}\right]$$

$$+w(m-2,n,i)\left[\frac{-Dr^4}{\Delta r^4}+\frac{Dr^3}{\Delta r^3}\right]$$

$$+w(m-1,n-1,i)\left[\frac{-2Dr^2}{\Delta r^2\Delta\theta^2}-\frac{Dr}{\Delta r\Delta\theta^2}\right]$$

$$+w(m-1,n,i)\left[\frac{4Dr^4}{\Delta r^4}-\frac{2Dr^3}{\Delta r^3}+\frac{Tr^4}{\Delta r^2}+\frac{Dr^2}{\Delta r^2}-\frac{Tr^3}{2\Delta r}+\frac{Dr}{2\Delta r}+\frac{4Dr^2}{\Delta r^2\Delta\theta^2}+\frac{2Dr}{\Delta r\Delta\theta^2}\right]$$

$$+w(m-1,n+1,i)\left[\frac{-2Dr^2}{\Delta r^2\Delta\theta^2}-\frac{Dr}{\Delta r\Delta\theta^2}\right]$$

$$+w(m,n-2,i)\left[\frac{-D}{\Delta\theta^4}\right]$$

$$+w(m,n-1,i)\left[\frac{4Dr^2}{\Delta r^2\Delta\theta^2}+\frac{Tr^2}{\Delta\theta^2}-\frac{4D}{\Delta\theta^2}+\frac{4D}{\Delta\theta^4}\right]$$

$$+w(m,n,i)\left[\frac{-6Dr^4}{\Delta r^4}-\frac{2Tr^4}{\Delta r^2}-\frac{2Dr^2}{\Delta r^2}-\frac{8Dr^2}{\Delta r^2\Delta\theta^2}-\frac{2Tr^2}{\Delta\theta^2}+\frac{8D}{\Delta\theta^2}-\frac{6D}{\Delta\theta^4}+\frac{2\bar{M}r^4}{\Delta t^2}\right]$$

$$+w(m,n+1,i)\left[\frac{4Dr^2}{\Delta r^2\Delta\theta^2}+\frac{Tr^2}{\Delta\theta^2}-\frac{4D}{\Delta\theta^2}+\frac{4D}{\Delta\theta^4}\right]$$

$$+w(m,n+2,i)\left[\frac{-D}{\Delta\theta^4}\right]$$

$$+w(m+1,n-1,i)\left[\frac{-2Dr^2}{\Delta r^2\Delta\theta^2}+\frac{Dr}{\Delta r\Delta\theta^2}\right]$$

$$+w(m+1,n,i)\left[\frac{4Dr^4}{\Delta r^4}+\frac{2Dr^3}{\Delta r^3}+\frac{Tr^4}{\Delta r^2}+\frac{Dr^2}{\Delta r^2}+\frac{Tr^3}{2\Delta r}-\frac{Dr}{2\Delta r}+\frac{4Dr^2}{\Delta r^2\Delta\theta^2}-\frac{2Dr}{\Delta r\Delta\theta^2}\right]$$

$$+w(m+1,n+1,i)\left[\frac{-2Dr^2}{\Delta r^2\Delta\theta^2}+\frac{Dr}{\Delta r\Delta\theta^2}\right]$$

$$+w(m+2,n,i)\left[\frac{-Dr^4}{\Delta r^4}-\frac{Dr^3}{\Delta r^3}\right] \qquad (51)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Eq. 51 will not work at the center of the drum because the r^4 term in the left-hand side of the equation goes to the denominator on the right-hand side during evaluations. Applying L'Hospital's rule four times removes the problem giving

$$w(m,n,i+1)\left[\frac{2\bar{M}+C\Delta t}{\Delta t^2}\right] = w(m,n,i-1)\left[\frac{-\bar{M}}{\Delta t^2}+\frac{C}{2\Delta t}\right]$$

$$+w(m-2,n,i)\left[\frac{-D}{\Delta r^4}\right]$$

$$+w(m-1,n,i)\left[\frac{4D}{\Delta r^4}+\frac{T}{\Delta r^2}\right]$$

$$+w(m,n,i)\left[\frac{-6D}{\Delta r^4}-\frac{2T}{r^2}+\frac{2\bar{M}}{\Delta t^2}\right]$$

$$+w(m+1,n,i)\left[\frac{4D}{\Delta r^4}+\frac{T}{\Delta r^2}\right]$$

$$+w(m+2,n,i)\left[\frac{-D}{\Delta r^4}\right] \qquad (52)$$

(This method does not appear to be strictly valid, but no other correction is obvious.) Another method, that of setting D = zero at the center of the drum, was also tried. It has a better physical basis, but both methods produced reasonable results.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix C: TEMPLATES

When the difference equation developed in Appendix B is evaluated, it is easier to understand the evaluation by using a template analogy. That is, the deflection value being computed depends upon nearby values, all indexed relative to the mesh location of the value being computed. The value being computed is in the "i+1" level of the template. Its location is w(m, n, i+1). The values used to compute this value reside in the "i" level and the "i-1" level. (Refer to the end of Appendix B for details on the particular values of *m*, *n*, and *i*.)

Computing the values on the "i+1" level requires moving this template to different *m* and *n* values. As the *m* and *n* values change, the edges of the template can go past the physical edge, the symmetry line, or the center point of the drum surface. When this happens, the template must be modified to take these facts into account. This appendix briefly describes the rules used to modify the template and explains each modified template that is used in this simulation.

With no modifications, the template is called the "Standard Template." All the other templates are labeled based upon the position of the center point of the template to whatever edge is encountered. The standard template varies in space, *m* and *n*, by plus or minus two meshpoints. Thus, when the center of the template comes within two mesh points of an edge, the template must be modified. (Because the only value from the "i-1" level is at the center of the template and the center is always on the drum surface, the "i-1" level is not modified.)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Rules:

- Any template location corresponding to the rim of the drumhead has a zero magnitude (no deflection at the rim).
- Any template location one meshpoint past the rim of the drumhead is multiplied by negative one and added to the location one meshpoint inside the rim on the same radial line. (The rim acts as a hinge.)
- Any template location past the symmetry line is added to its mirror location.
- Any template locations that correspond to the center of the drum are added together and treated as one point.

Some explanations will make the following paragraphs easier to understand. (These paragraphs explain the sections in the subroutine *value* in Appendix D.) The name or title of each template will be left-justified followed by a number used to identify the template in software. The modifications will be listed next to bullets. The previous paragraphs talk of changing the value of the meshpoint and adding it to another meshpoint. To accomplish this in a simpler mathematical way, the coefficient of the meshpoint is modified and added to another appropriate coefficient. If the meshpoint value is zero because it is on the rim, its coefficient is removed from the template. As an example, the first bullet below this paragraph means: take the coefficient of w(m+2, n, i) change its sign and add it to the coefficient of w(m, n, i). Also, the second bullet means: set the coefficient of w(m+2, n, i) to zero. These two examples should be enough to understand the ones that follow.

- w(m+2, n, i) >chs and add to w(m, n, i)
- w(m+2, n, i > zero

Standard template - 1:

- no modification

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Two from rim - 2:

- w(m+2, n, i) > zero

One from rim - 3:

- w(m+2, n, i) > chs and add to w(m, n, i)
- w(m+1, n-1,i) > zero
- w(m+1, n, i) > zero
- w(m+1, n+1,i) > zero

one from zero-degree symmetry axis - 4:

- w(m, n-2,i) > w(m-2,n,i) +w(m, n, i)

On zero-degree symmetry axis - 5:

- w(m+1, n-1,i) > add to w(m+1, n+1, i)
- w(m, n-1,i) >add to w(m, n+1, i)
- w(m, n-2,i) >add to w(m, n+2, i)
- w(m-1, n-1,i) >add to w(m-1, n+1, i)

One from Π symmetry axis - 6:

- w(m, n+2, i) > add to w(m, n, i)

On Π symmetry axis - 7:

- w(m+1, n+1,i) >add to w(m+1, n-1, i)
- w(m, n+1,i) >add to w(m, n-1, i)
- w(m, n+2,i) >add to w(m, n-2, i)
- w(m-1, n+1,i) > add to w(m-1, n-1, i)

One from center - 8:

- w(m-2, n, i) > w(1, n_max - n, i)
- add together w(m-1, n-1, i), w(m-1, n, i), w(m-1, n+1, i)

At center - 9:(computed for convenience at m=n=0)

- w(m+1, n-1, i) > add to w(m+1, n+1, i)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- add together w(m, n-2, i), w(m, n-1,i), w(m, n, i), w(m, n+1, i), w(m, n+2, i)
- w(m-1, n-1, i) > w(1, n_max -1, i)
- w(m-1, n, i) > w(1, n_max, i)
- w(m-1, n+1, i) > w(1, n_max - 1, i) add to same location above
- w(m-2, n, i) > w(2, n_max, i)

Two from the rim, one from the zero-degree symmetry axis - 10:

- w(m+2, n, i) > zero
- w(m, n-2, i) > add to w(m, n, i)

Two from rim and on the zero-degree symmetry axis - 11:

- w(m+2, n, i) > zero
- w(m+1, n-1, i) > add to w(m+1, n+1, i)
- w(m, n-2, i) > add to w(m, n+2, i)
- w(m, n-1, i) > add to w(m, n+1, i)
- w(m-1, n-1, i) > add to w(m-1, n+1, i)

One from rim and one from zero-degree symmetry axis - 12:

- w(m+2, n, i) > chs and add to w(m, n, i)
- w(m+1, n-1, i) > zero
- w(m+1, n, i) > zero
- w(m+1, n+1, i) > zero
- w(m, n-2, i) > add to w(m, n, i)

One from rim and on zero-degree symmetry axis - 13:

- w(m+2, n, i) > chs and add to w(m, n, i)
- w(m+1, n-1, i) > zero
- w(m+1, n, i) > zero
- w(m+1, n+1, i) > zero
- w(m, n-2,i) > add to w(m, n+2, i)
- w(m, n-1, i) > add to w(m, n+1, i)
- w(m-1, n-1, i) > w(m-1, n+1, i)

Two from rim and one from Π symmetry axis - 14:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- w(m+2, n, i) > zero
- w(m, n+2, i) > add to w(m, n, i)

Two from rim and on Π symmetry axis - 15:

- w(m+2, n, i) > zero
- w(m+1, n+1, i) > add to w(m+1, n-1, i)
- w(m, n+2, i) > add to w(m, n-2, i)
- w(m, n+1, i) > add to w(m, n-1, i)
- w(m-1, n+1, i) > add to w(m-1, n-1, i)

One from rim and one from Π symmetry axis - 16:

- w(m+2, n, i) > chs and add to w(m, n, i)
- w(m+1, n-1, i) > zero
- w(m+1, n, i) > zero
- w(m+1, n+1, i) > zero
- w(m, n+2, i) > add to w(m, n, i)

One from rim and on Π symmetry axis - 17:

- w(m+2, n, i) > chs and add to w(m, n, i)
- w(m+1, n-1, i) > zero
- w(m+1, n, i) > zero
- w(m+1, n+1, i) > zero
- w(m, n+2,i) > add to w(m, n-2, i)
- w(m, n+1, i) > add to w(m, n-1, i)
- w(m-1, n+1, i) > w(m-1, n-1, i)

One from the center and one from the zero-degree symmetry axis - 18:

- w(m, n-2, i) > add to w(m, n, i)
- add together w(m-1, n-1, i), w(m-1, n, i), w(m-1, n+1, i)
- w(m-2, n, i) > w(1, n_max - 1, i)

one from the center and on zero-degree symmetry axis - 19:

- w(m+1, n-1, i) > add to w(m+1, n+1, i)
- w(m, n-2, i) > add to w(m, n+2, i)
- w(m, n-1, i) > w(m, n+1, i)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- add together w(m-1, n-1, i), w(m-1, n, i), w(m-1, n+1, i)
- w(m-2, n, i) > w(1, n_max, i)

One from the center and one from the Π axis - 20:

- w(m, n+2, i) add to w(m, n, i)
- add together w(m-1, n-1, i), w(m-1, n, i), w(m-1, n+1, i)
- w(m-2, n, i) > w(1, 0, i)

One from the center and on the Π axis - 21:

- w(m+1, n+1, i) > add to w(m+1, n-1, i)
- w(m, n+2, i) > add to w(m, n-2, i)
- w(m, n+1, i) > w(m, n-1, i)
- add together w(m-1, n-1, i), w(m-1, n, i), w(m-1, n+1, i)
- w(m-2, n, i) > w(1, 0, i)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix D: SOFTWARE LISTINGS

The code used to do the simulation for this dissertation is listed below. It contains sufficient comments to understand it. The bold-faced titles are used to delineate the different routines.

The following sections describe each of the software routines that make up the simulation program for this dissertation. While these sections give a general explanation of each routine, the routines themselves contain extensive comments that explain the detailed operation of the code.

Some comments and sections in the actual code are commented out. Sometimes the code lines are no longer needed, but usually the lines are there for occasional trouble-shooting use or future expansion. Any comments, including commented-out lines of code, can be removed from the program and it will function as described below.

The software listings in the appendices are the actual ones used for this dissertation research.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

 # This shell file is used to start the DSG program interactively

This shell script is used to construct a set of command lines to be executed sequentially, which contain the appropriate parameters chosen for each simulation. It is not required to be used, but the large number of parameters makes not using this program tedious. It prompts the user to enter a value for each parameter. The prompt gives a brief explanation of each parameter and the parameters' minimum, default, and maximum values if they exist. (See the DSinfo.txt listing for details on the parameters.)

```
#
#uncomment the next line if the main shell is not csh
#! /bin/csh -f
echo "DSG interactive entry. Enter a carriage return for default values"
echo " value possibilities -- (minimum:default:maximum)."
echo "Entering values outside the limits will cause DSG to terminate."

echo "#this comment tells csh to run this under csh not sh" > DSG.run
echo "#uncomment the next line if the main shell is not csh" >> DSG.run
echo "#! /bin/csh -f " >> DSG.run
echo "" >> DSG.run
echo "" >> DSG.run

echo ""
echo ""

set TEST = y

while($#TEST != 0)

echo -n "Enter the filestem (DEFAULTSTEM): "
set STEM = ($<)
if ($#STEM == 0) set STEM = DEFAULTSTEM
echo "$STEM"
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
echo -n "enter the drum diameter in meters, (0.05: 0.25: none): "
set DIA = ($<)
if ($#DIA == 0) set DIA = 0.25
echo "$DIA"

echo -n "enter the first circular harmonic, (1: 125: none): "
set F01 = ($<)
if ($#F01 == 0) set F01 = 125
echo "$F01"

echo -n "enter the stiffness coefficient, (0: 7.98e-3: none): "
set STIFF = ($<)
if ($#STIFF == 0) set STIFF = 7.980e-3
echo "$STIFF"

echo -n "enter the variable stiffness switch, (0 = off: 1 = on): "
set VSTIFF = ($<)
if ($#VSTIFF == 0) set VSTIFF = 0
echo "$VSTIFF"

echo -n "enter the center point stiffness switch, (0 = off: 1 = on): "
set CSTIFF = ($<)
if ($#CSTIFF == 0) set CSTIFF = 0
echo "$CSTIFF"

echo -n "enter the stability multiplier, (0.0: 1.0: 1.0): "
set STABLM = ($<)
if ($#STABLM == 0) set STABLM = 1.0
echo "$STABLM"

echo -n "enter the damping coefficient, (0.0: 5.0: none): "
set DAMP = ($<)
if ($#DAMP == 0) set DAMP = 5.0
echo "$DAMP"

echo -n "enter the rim damping coefficient, (0.0: 5.0: none): "
set RDAMP = ($<)
if ($#RDAMP == 0) set RDAMP = 5.0
echo "$RDAMP"
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
echo -n "enter the frequency dependent damping switch, (0 = off: 1 = on): "
set FDAMP = ($<)
if ($#FDAMP == 0) set FDAMP = 0
echo "$FDAMP"

echo -n "enter the H0 (central) term of the damping filter, (none: 0.0: none): "
set VDA = ($<)
if ($#VDA == 0) set VDA = 0.0
echo "$VDA"

echo -n "enter the H1 (peripheral) term of the damping filter, (none: 0.0: none): "
set VDB = ($<)
if ($#VDB == 0) set VDB = 0.0
echo "$VDB"

echo -n "enter the air density coefficient, (0.0: 1.0: none): "
set AIRC = ($<)
if ($#AIRC == 0) set AIRC = 1.0
echo "$AIRC"

echo -n "enter the membrane density coefficient, (0.0: 1.0: none): "
set MEMC = ($<)
if ($#MEMC == 0) set MEMC = 1.0
echo "$MEMC"

echo -n "enter the variable tension switch, (0 = off: 1 = on): "
set TENC = ($<)
if ($#TENC == 0) set TENC = 0
echo "$TENC"

echo -n "enter the length of the simulation in seconds, (0.0: 3.0: 600.0): "
set LENGTH = ($<)
if ($#LENGTH == 0) set LENGTH = 3.0
echo "$LENGTH"

echo -n "enter the strength of the hit, (0.0: 100.0: 150.0): "
set HIT = ($<)
if ($#HIT == 0) set HIT = 100.0
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
echo "$HIT"

echo -n "enter the hit location, 0 is center, and 1.0 is rim, (0.0: 0.8: 1.0): "
set LOC = ($<)
if ($#LOC == 0) set LOC = 0.8
echo "$LOC"

echo -n "enter the desired number of radial mesh points, (10: 10: 250): "
set MESH = ($<)
if ($#MESH == 0) set MESH = 10
echo "$MESH"

echo "/bin/time DSG$1 $STEM $DIA $F01 $STIFF $STABLM $DAMP $LENGTH $HIT $LOC $MESH
$RDAMP $VSTIFF $AIRC $MEMC $FDAMP $TENC $VDA $VDB $CSTIFF>& $STEM.log; " >> DSG.run

echo -n "Do you want to enter more runsets? (yes=y: no=CR) "
set TEST = ($<)
end

echo ""
echo ""
echo "setup parameters are stored in STEM.hdr"
echo "hit point displacement sound output is stored in STEM _d.snd"
echo "hit point acceleration sound output is stored in STEM _a.snd"
echo "averaged (farfield accelerations) sound output is stored in STEM _A.snd"
echo "symmetry line output is stored in STEM.amp"
echo ""
echo ""
echo "Type: DSG.run & to start the sequential run asynchronously"
echo "Type: tail -f STEM.log to check on job progress"
echo ""
echo ""

exec chmod 744 DSG.run
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

/* DSTop By Jim Irwin: written 24 Jan 1994

This file contains the main program. It starts out by declaring variables and saving appropriate space for matrices. Reading through the variables will help in understanding the terminology and structure of the program. The program computes the output in two major steps. The first step is to compute the movement and acceleration of the drumhead in floating-point format. This precludes the need for any scaling in the computation routines. This datum is stored in temporary files. After the floating-point simulation is complete the data is scaled to 16-bit integer format in order to be played through an audio system.

After setting up all the variables, the program reads in the various parameters needed to perform a simulation. These parameters can be placed on a command line by a user, but due to the many parameters it is usually easier to use the routine startDSG, described later, to construct the command line by answering questions.

Next, some matrices are set to zero to prevent garbage from contaminating the program. (Some compilers set matrices to zero automatically, some do not.) The initial conditions and soft constants are computed from the given parameters. Soft constants are defined as a number that is a constant for a particular simulation, but is actually a variable of the input parameters. These include: speed of sound in the drumhead; surface area of the drumhead; rest, or default tension of the drumhead; differential radius; differential angle; and differential time.

The three differential soft constants are computed from the parameter giving the number of radial mesh points. The differential angle is first computed to make the differential area on the drumhead mesh as "square" as possible. (Here, "square" implies that the ratio of the circular edge to the radial edge at the center and the rim should be as similar as possible.) This constraint may not be needed, but seemed useful to prevent very narrow,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

long mesh areas where the disparity between the differential radius and angle could cause problems. The differential time value is first computed to be just smaller than the one required by the hyperbolic stability criteria. If the drumhead has a nonzero stiffness, this stability factor is incorrect and so another parameter is used to reduce the value of differential time. Finally, in order to eliminate the need for interpolation when computing the sound file, the differential time is reduced to the next submultiple of the standard audio playback data rate, i.e. the sampling frequency is a multiple of 44,100 Hz. This means that the data points eventually played are ones actually computed. If the differential time is small enough, some data points will be computed but not played.

The *hitstr* parameter, strength of the hit, is used to set up the initial conditions. This explanation will require some background. A real drum is stuck with a stick that is "free." This means that the stick has a constant velocity and is not constrained in the vertical direction. In actual experiments, it was determined that the maximum deflection produced by a drumstick on a typical drumhead was approximately one centimeter. In order to mimic the strike of a drumstick at a constant velocity, the program constructs a velocity profile in the displacement matrix around the point the stick strikes that extends into negative time. The velocity is arbitrarily scaled so that the maximum allowed value for the hit force, *hitstrength*, will produce a deflection of one centimeter. (Note that exactly what physical velocity this implies is not important because a human drummer would determine this by experimentation.The drummer would actually correlate loudness and force of hit, rather than velocity and hit force.) This "displacement" is interpreted as a velocity by the program due to the nature of the coefficients that are used, see *setupcoef.c*

The matrix that describes the coefficients of the difference equation, or coefficient matrix, is three-dimensional and will be described in detail in *setupcoef.c*. The coefficients are used to compute the value of the deflection of the drumhead at a mesh location.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Various mesh points on the drumhead require modified templates. The templates are set up next and described in Appendix C.

The main program now does some final bookkeeping to finish the preparations for the actual simulation. The program now enters the loop that computes the meshpoint deflection. It uses the *value.c* routine to do this.

The main loop contains a section that was used to attempt to model the frequency dependent damping by placing a low-pass filter near the rim, (see Appendix D, *DSinfo*, for details on how this works).

The main loop is where the nonlinear tension is modeled. If nonlinear tension is part of the simulation, the program computes the excess area of the drumhead due to deflections of the drumhead. This surface area is compared to the "at rest" or default surface area. If the difference is great enough, presently ten parts per billion, the tension is recomputed using the increased surface area. This requires the coefficient matrix to be recomputed before the next use.

Both the main loop and the tension recomputation contain tests to correctly use the coefficient matrix. Please see the actual program for an explanation of how these work.

After the main loop is finished, some bookkeeping is done in preparation for converting the floating point format files into integer *NeXT soundfile* format. A second loop is entered where the data are converted. After the conversion is complete, the program cleans up by deleting unnecessary files and exits.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
See DSinfo.txt for information and modifications.

This is the top file in the drum sound simulation package.

*/

#include <stdio.h>
#include <string.h>
#include <math.h>
#include <fcntl.h>
#include <stdlib.h>
#include "DSsound.h"

/* these arrays are external because of their size */
int template[M_MAX + 1][N_MAX + 1];/*this matrix will hold the index
number of the correct template to use for each mesh point point in the surface. Special templates are needed near the rim, along the symmetry axis, and at the center of the drum, r=0. */
double coef[3][M_MAX + 1][15];/* template coefficient matrix - each row of the matrix will hold the computed value for each coefficient in the standard template. The coefficient order will be determined in the routine that computes them. This array is needed because the coefficients depend upon the radius of the mesh point point; thus the row corresponds to the radial mesh point value */
double s[3][M_MAX + 1][N_MAX + 1];/*This is the surface matrix. */

main(int argc, char *argv[])
{

unsigned long deci;/*used in the decimate routine */
unsigned long decimate = 0;/*used to set dt to a convenient value for decimation*/

short int sound_data;/*used to scale and transfer the output to a sound file*/
short int siZERO=0; /*used to set some things to zero*/

int fdO_d; /*file descriptor for the hitpoint displacement output soundfile */
int fdO_a; /*file descriptor for the hitpoint acceleration output soundfile */
int fdO_A; /*file descriptor for the far-field average acceleration output soundfile*/
int f01 = F01;/*frequency of the first circular harmonic */
int m_max = M_MAX; /*largest radial mesh point number, corresponds to the rim*/
int n_max = N_MAX; /*largest angular mesh point number, corresponds to 180 degrees */
int m,n; /*mesh index variables */
int templatenumber; /*index used by value to pick the appropriate template*/
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
int i,j,k,ii,jj;/* index variables, used many places */
int hitpt;/*value of radial index where strike occurs */
int coefndx = 0;/*used to choose the correct set of coefficients for the difference equation. 0 => the first iteration when t-1 values are not known, 1=> all other iterations. */
int now = 0;/*value of the first array index; it indicates where the present surface data is, as opposed to past data */
int tencount = 0;/*this is used to keep track of the number of times in a row that the tension is at the default value. After several times at the default value I will use this variable to prevent the length of the strikeline from being computed. If it is greater than one the coef matrix will not be recomputed either. */
int csS; /*centerstiffSwitch used to turn the center stiffness to zero independently of the variable stiffness*/
int vsS;/* variablestiffnessSwitch used to turn the variable stiffness off =0, or on =1*/
int FreqSwitch;/*used to turn the frequency dependent damping off =0, or on =1*/
int TenSwitch;/*used to turn the variable tension off =0, or on =1*/

double dZERO=0.0; /*used to send zeros to a file*/
double Acoef;/* used to modify the simulated air density */
double Mcoef;/* used to modify the simulated membrane density */
double beta;/*used for testing. stability factor: delete when works */
double stabilitymultiplier = 1.; /*used to modify beta to force stablity */
double PI=4.0*atan(1.0); /* the constant */
double t_max = T_MAX;/*simulation length, in seconds */
double elapse = 0.0;/*elapsed time in the simulation */
double tmax_amp=0.0; /*time point of hitpoint displacement maximum amplitude */
double tmax_amp_a=0.0; /*time point of hitpoint acceleration maximum amplitude */
double tmax_amp_A=0.0; /*time point of far-field average acceleration maximum amplitude */
double hitstr = HITSTRENGTH;/*hit strength, velocity of the drum stick, in meters per second */
double drdia = DRDIA;/*drum diameter, in meters */
double hitloc = HITLOCATION;/* radial location of the impact point, 0 is the center of the drumhead, 1 is the rim */
double dr;/*radial infinitesimal */
double da;/*angular infinitesimal */
double dt;/*time infinitesimal */
double c;/*speed of sound in the drum head material */
double dttemp;/*used to set dt to a convenient value for decimation*/
double tensionP;/*present tension of the drumhead */
double tensionD;/*default tension of the drumhead */
double deltaTcoef; /*coefficent used to compute the change in tension*/
double s_area;/* surface area, used to compute change in tension*/
double deltaA; /*change in surface area, used to compute change in tension*/
double symlen;/*length of the symmetry line, used to compute the tension of the drumhead*/
double max_amp = 0.0;/*maximum hitpoint displacement amplitude*/
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
double max_amp_a = 0.0;/*maximum hitpoint acceleration amplitude*/
double max_amp_A = 0.0;/*maximum far-field average acceleration amplitude*/
double accel_total = 0.0; /*sum of all accelerations on the drumhead */
double accel_hitpt = 0.0;/*accelation of the hitpoint */
double accel_scale=pow((double)SAMPLE_RATE,2); /*acceleration scale factor */
double amplitude;/*floating point sound data, used in scaling*/
double scalefactor;/*used to scale the data from floating point to short_int */
double D;/*the coefficient of stiffness */
double Cr;/*the coefficient of damping in terms of mass per unit area */
double RCr;/*the coefficient of rim damping in terms of mass per unit area */
double vdampa=0.0; /*filter coefficient used in the frequency damping section*/
double vdampb=0.0; /*filter coefficient used in the frequency damping section*/

FILE *fpT_d;/*file pointer for the displacement temporary file - floating point soundfile data before conversion to integer format */
FILE *fpT_a;/*file pointer for the hitpoint acceleration temporary file - floating point soundfile data before conversion to integer format */
FILE *fpT_A;/*file pointer for the far-field average acceleration temporary file - floating point soundfile data before conversion to integer format */
FILE *fpS;/*file pointer for the symmetry line data */
FILE *fpH;/*file pointer for the header file*/

char *nameO_d;/*pointer to the name of the hitpoint displacement output soundfile */
char *nameO_a;/*pointer to the name of the hitpoint acceleration output soundfile */
char *nameO_A;/*pointer to the name of the far-field average acceleration output soundfile */
char *nameT_d;/*pointer to the name of the hitpoint displacement temporary file */
char *nameT_a;/*pointer to the name of the hitpoint acceleration temporary file */
char *nameT_A;/*pointer to the name of the far-field average acceleration temporary file */
char *nameS;/*pointer to the name of the symmetry line data file */
char *nameH;/*pionter to the name of the header file */

NXT_HDR *header;/*pointer to a header for a NeXT soundfile */

/* check arguments and set parameters */

/* a general check */
if (argc <20)
        error(" Please read DSinfo.txt");
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
/*This section contains the major parameter entry code. Some parameters are entered later because of the
iterative choice for those parameters. In order to print default values the variables must be set to the default
value before the question is asked. */

/*use the filestem parameter, and try to find space for the names */
        if ((nameO_d = (char*)malloc(strlen(argv[1]) + 7)) == NULL)
                error(" no space for nameO_d");
        strcpy(nameO_d, argv[1]);

        if ((nameO_a = (char*)malloc(strlen(argv[1]) + 7)) == NULL)
                error(" no space for nameO_a");
        strcpy(nameO_a, argv[1]);

        if ((nameO_A = (char*)malloc(strlen(argv[1]) + 7)) == NULL)
                error(" no space for nameO_A");
        strcpy(nameO_A, argv[1]);

        if ((nameT_d = (char*)malloc(strlen(argv[1]) + 7)) == NULL)
                error(" no space for nameT_d");
        strcpy(nameT_d, argv[1]);

        if ((nameT_a = (char*)malloc(strlen(argv[1]) + 7)) == NULL)
                error(" no space for nameT_a");
        strcpy(nameT_a, argv[1]);

        if ((nameT_A = (char*)malloc(strlen(argv[1]) + 8)) == NULL)
                error(" no space for nameT_A");
        strcpy(nameT_A, argv[1]);

        if ((nameS = (char*)malloc(strlen(argv[1]) + 5)) == NULL)
                error(" no space for nameS");
        strcpy(nameS, argv[1]);

        if ((nameH = (char*)malloc(strlen(argv[1]) + 5)) == NULL)
                error(" no space for nameS");
        strcpy(nameH, argv[1]);

/* in either case add the suffixes to the file names */
strcat(nameO_d, "_d.snd");
strcat(nameO_a, "_a.snd");
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
strcat(nameO_A, "_A.snd");
strcat(nameT_d, "_d.tmp");
strcat(nameT_a, "_a.tmp");
strcat(nameT_A, "_A.tmp");
strcat(nameS, ".amp");
strcat(nameH, ".hdr");

/* attempt to open the files */
if ((fdO_d = creat(nameO_d, 0644)) == -1)
        error(" cannot create %s", nameO_d);

if ((fdO_a = creat(nameO_a, 0644)) == -1)
        error(" cannot create %s", nameO_a);

if ((fdO_A = creat(nameO_A, 0644)) == -1)
        error(" cannot create %s", nameO_A);

if ((fpT_d = fopen(nameT_d, "w")) == NULL)
        error(" cannot open %s", nameT_d);

if ((fpT_a = fopen(nameT_a, "w")) == NULL)
        error(" cannot open %s", nameT_a);

if ((fpT_A = fopen(nameT_A, "w")) == NULL)
        error(" cannot open %s", nameT_A);

if ((fpS = fopen(nameS, "w")) == NULL)
        error(" cannot open %s", nameS);

if ((fpH = fopen(nameH, "w")) == NULL)
        error(" cannot open %s", nameH);

fprintf(fpH, "The filestem is %s\n", argv[1]);

/* fill in and write the soundfile header to fd */
wrNxtHdr(fdO_d, SAMPLE_RATE, NUMBER_OF_CHANNELS);
wrNxtHdr(fdO_a, SAMPLE_RATE, NUMBER_OF_CHANNELS);
wrNxtHdr(fdO_A, SAMPLE_RATE, NUMBER_OF_CHANNELS);
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
/*drum diameter */
drdia = atof(argv[2]);
fprintf(fpH, "The drum diameter is %f\n", drdia);
if (drdia < 0.05)
        error(" the drum diameter is too small");

/*first circular harmonic */
f01 = atoi(argv[3]);
fprintf(fpH, "The first circular harmonic is %d\n", f01);
if (f01 < 0)
        error(" the first circular harmonic must be positive.");

/*stiffness */
D = atof(argv[4]);
fprintf(fpH, "The stiffness coeffecient is %f\n", D);

/*stability multiplier*/
stabilitymultiplier = atof(argv[5]);
fprintf(fpH, "The stability multiplier is %f\n", stabilitymultiplier);

/*damping */
Cr = atof(argv[6]);
fprintf(fpH, "The damping coefficient is %f\n", Cr);

/*rim damping */
RCr = atof(argv[11]);
fprintf(fpH, "The rim damping coefficient is %f\n", RCr);

/*variable stiffness */
vsS = atoi(argv[12]);
if (vsS != 0 && vsS != 1)
        error(" the variable stiffness coef must be 0 or 1.");
if (vsS)
        fprintf(fpH, "The variable stiffness is ON\n");
else
        fprintf(fpH, "The variable stiffness is OFF\n");

/*center stiffness */
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
csS = atoi(argv[19]);
if (csS != 0 && csS != 1)
        error(" the variable stiffness coef must be 0 or 1.");
if (csS)
        fprintf(fpH, "The center stiffness is ON\n");
else
        fprintf(fpH, "The center stiffness is OFF\n");

/*frequency dependent damping */
FreqSwitch = atoi(argv[15]);
if (FreqSwitch != 0 && FreqSwitch != 1)
        error(" the frequency dependent damping coef must be 0 or 1.");
if (FreqSwitch)
        fprintf(fpH, "The frequency dependent damping is ON\n");
else
        fprintf(fpH, "The frequency dependent damping is OFF\n");

/*frequency dependent damping coefficients*/
vdampa = atof(argv[17]);
vdampb = atof(argv[18]);
fprintf(fpH, "The frequency dependent damping coefficients are\n\t h0= %f,\th+/-1= %f\n", vdampa,
vdampb);

/*air density coefficient */
Acoef = atof(argv[13]);
if (Acoef< 0)
        error(" the air density coefficient must be positive.");
fprintf(fpH, "The air density coefficient is %f\n", Acoef);

/*membrane density coefficient */
Mcoef = atof(argv[14]);
if (Mcoef < 0)
        error(" the membrane density coefficient must be positive.");
fprintf(fpH, "The membrane density coefficient is %f\n", Mcoef);

/*variable tension */
TenSwitch = atoi(argv[16]);
if (TenSwitch != 0 && TenSwitch != 1)
        error(" the variable tension coef must be 0 or 1.");
if (TenSwitch)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        fprintf(fpH, "The variable tension is ON\n");
else
        fprintf(fpH, "The variable tension is OFF\n");

/* simulation length */
t_max = atof(argv[7]);
fprintf(fpH, "The simulation length is %f\n", t_max);
if (t_max > T_MAX)
        error(" the simulation length is too long.");

/* hit strength */
hitstr = atof(argv[8]);
fprintf(fpH, "The hit strength is %f\n", hitstr);
if (hitstr > HITSTRENGTH)
        error(" The hitstrength is too large");

/* hit location */
hitloc = atof(argv[9]);
fprintf(fpH, "The hit location is %f\n", hitloc);
if (hitloc <0 || hitloc >= 1.0)
        error(" The hit location must be > 0.0 and < 1.0");

/* just to be sure - set all matrix values to zero */

for (i=0; i<3; i++)
        {
        for (j=0; j<=M_MAX; j++)
                {
                for (k=0; k<=N_MAX; k++)
                        {
                        template[j][k] = 0;
                        s[i][j][k] = 0.0;
                        }
                for (k=0; k<15; k++)
                        {
                        coef[0][j][k] = 0.0;
                        coef[1][j][k] = 0.0;
                        coef[2][j][k] = 0.0;
                        }
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                }
        }

/* Set up the initial conditions and soft constants*/

c = f01 * PI * drdia/Z01;
fprintf(fpH, "The speed of sound, c, is %f\n", c);

s_area = (PI * drdia * drdia)/4;
deltaTcoef =(EY * THICKNESS)/s_area;

tensionD = RHOMYLAR * c * c * THICKNESS;
fprintf(fpH, "The tension is %f\n", tensionD);
tensionP = tensionD;

/* this section will be used to set dt to a "multiple" of 44100 Hz so that the decimation routine is easy to
implement. Then using beta and some constraint equations meant to keep the polar regions as "square" as
possible dr, and da will be computed. m_max is entered here. */

m_max = atoi(argv[10]);
fprintf(fpH, "The user entered value of m_max is %d\n", m_max);
if(m_max >= M_MAX)
        error(" m_max is too large");

/*compute the value for dr, based upon m_max */
dr= drdia/(2*m_max);

/*da is initially computed using the "square" constraint from m_max. Next n_max must be computed and
because it is an integer da must be adjusted to be correct for the integer chosen. The integer chosen is the
largest possible that will keep da from being smaller when it is recomputed. */

da= sqrt(1.0/m_max); /*initial value for da*/
n_max= floor(PI/da); /*set n_max to the largest possible integer*/
fprintf(fpH, "The computed value of n_max is %d\n", n_max);

/*test n_max*/
        if(n_max > N_MAX)
                {
                fprintf(fpH,"\nDANGER: n_max is greater than N_MAX; n_max is set to N_MAX");
                n_max= N_MAX;
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                }
fprintf(fpH, "The actual value of n_max is %d\n", n_max);

da = PI/n_max; /*this recomputes da to its final value.*/

/*Based upon dr, and da, the maximum value of 'beta', the stability factor, can be computed. This value of
beta is used to compute an initial value of dt. A loop is used to choose a value of dt that will decimate con-
veniently. */

beta= (PI*PI)/(PI*PI + n_max*n_max);
fprintf(fpH, "The maximum value of beta is %f\n", beta);

/*If the stiffness coefficient is not zero beta must be reduced to force
stability. This next line does that. */
beta *= stabilitymultiplier;
fprintf(fpH, "The scaled maximum value of beta is %f\n", beta);

dt= sqrt((beta*dr*dr)/(c*c));
for(decimate=0; decimate <32000; decimate++)
        {
        dttemp = 1.0/(decimate * (double)44100);
        if (dttemp<dt)
                break;
        }
dt= dttemp;
beta= (c*c*dt*dt)/(dr*dr);
fprintf(fpH, "The final value of beta is %f\n", beta);


/*This next section computes the ratio of the edges of the grid at the rim and the center of the drum. The
final value of beta is computed from the final values of dr and da so that it can be compared to the user
supplied value. Small differences between these betas are probably not significant. These and several
other pieces of data are printed out.*/

fprintf(fpH,"\n\nThe final parameters are:\n");
fprintf(fpH,"decimate= %d\n", decimate);
fprintf(fpH,"dt= %e,\t\tsample rate is %lu*44100 Hz.\n", dt, decimate);
fprintf(fpH,"dr= %e,\t\tm_max= %d\n", dr, m_max);
fprintf(fpH,"da= %e,\t\tn_max= %d\n", da, n_max);
fprintf(fpH,"real beta= %f\n", beta);
fprintf(fpH,"rim ratio= %f\n", (da*m_max/2));
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
fprintf(fpH,"center ratio= %f\n\n", (1.0/da));

hitpt = floor(m_max * hitloc);
fprintf(fpH,"hitpt= %d\n",hitpt);
if(hitpt == m_max - 1)
        hitpt--; /* my program cannot handle a hit next to the rim. I cure it by moving the hit point inward
one position. */
fprintf(fpH, "the final hitpt= %d\n",hitpt);
fclose(fpH); /*close the header file */

/*In order to keep the energy, and velocity, relatively constant with changing m_max, and dt, the enetered
value of histr must be modified. This keeps the enetered value reasonably constant. The value 0.1 comes
from a quick test and is used to modify m_max. Remember that hitstr is a velocity not a displacement.*/
hitstr *= 0.1 * m_max/decimate;

switch (hitpt)
        {
        case 0:
                s[2][0][0] = hitstr;
                for(i=0; i<=n_max; i++)
                {
                        s[2][1][i] = hitstr * 2./3.;
                        s[2][2][i] = hitstr * 1./3.;
                }
                break;
        case 1:
                s[2][1][0] = hitstr;

                s[2][hitpt + 1][0] = hitstr * 2./3.;
                s[2][hitpt - 1][0] = hitstr * 2./3.;
                s[2][hitpt][1] = hitstr * 2./3.;
                s[2][hitpt + 1][1] = hitstr * 1./3.;

                break;

        default:

                s[2][hitpt][0] = hitstr;

                s[2][hitpt + 1][0] = hitstr * 2./3.;
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                s[2][hitpt - 1][0] = hitstr * 2./3.;

                s[2][hitpt][1] = hitstr * 2./3.;
                s[2][hitpt + 1][1] = hitstr * 1./3.;
                s[2][hitpt - 1][1] = hitstr * 1./3.;

                break;
        }
/*for (i=0; i <3; i++)
        for (j=0; j<=m_max; j++)
                for (k=0; k <n_max; k++)
                        printf("s[%d][%d][%d]= %e\n", i, j, k, s[i][j][k]);
*/
/* Set up the coefficient matrix */

setupcoef(coef, tensionP, m_max, dr, da, dt, D, Cr, RCr, vsS, csS, Acoef, Mcoef);

/* Set up the template matrix */

setuptemplate(template, m_max, n_max);

/*
for (i=0; i <15; i++)
        {
        printf("coef[0][0][%d] = %e\n", i, coef[0][0][i]);
        printf("coef[1][0][%d] = %e\n", i, coef[1][0][i]);

        printf("coef[0][5][%d] = %e\n", i, coef[0][5][i]);
        printf("coef[1][5][%d] = %e\n", i, coef[1][5][i]);
        }
fflush(stdout);
*/

/*for (i=0; i<=m_max; i++)
        for (k=0; k<=n_max; k++)
                printf("template[%d][%d]= %d\n", i, k, template[i][k]);*/

/* set up to save only decimated data. The value of deci is set to record the first computed value of the output. From there on the values are spaced out*/
deci= decimate - 1;
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
k= 0;
j= 999;

/* save the t=0 symmetry line data; all values should be zero so s[0][][] could be replaced in the next several lines by zero. It is left this way in case the initial conditions are changed so that s[0][][] is not zero. */
        for (i=m_max; i>=0; i--)
                fprintf(fpS,"%e ",s[now][i][0]);
        for (i=1; i<=m_max; i++)
                fprintf(fpS,"%e ",s[now][i][n_max]);
        fprintf(fpS,"\n");

/* the t=0 output data NOTE at present with only an initial velocity at time zero the displacement and the acceleration (past - Since the drum head is at rest before t=0.0 the equation for acceleration is different looking; y(n-1)=y(n-2)=0.0) will be zero these equations will need to be changed if the initial conditions are changed to include an initial displacement */

fprintf(fpT_d,"%e\n",dZERO);
fprintf(fpT_a,"%e\n",dZERO);
fprintf(fpT_A,"%e\n",dZERO);

/* do the computations and save the appropriate data */

elapse = dt;
while (elapse <= t_max )
        {
        /*printf("now= %d\t", now);*/
/*routine that computes one complete time layer; though the center only once */

        templatenumber= template[0][0];
        s[(now+1)%3][0][0]= value(now, templatenumber, s, coef, coefndx, 0, 0, n_max);

        for (m=1; m<m_max; m++)
                {
                for (n=0; n<=n_max; n++)
                        {
                        templatenumber= template[m][n];
                        s[(now+1)%3][m][n]= value(now, templatenumber, s, coef, coefndx, m, n, n_max);
                        }
                }
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
/*to ease the template formulations n can be any number if m=0 so the following loop sets all s[][0][n] values to s[][0][0] */
        for (n=1; n<=n_max; n++)
                s[(now+1)%3][0][n]= s[(now+1)%3][0][0];

/*This section is an attempt to add frequency dependent damping to the rim of the membrane. A three point FIR filter will be used. The coefficients have been calculated by frequency sampling and are empirically adjusted to test whether frequency dependent damping is important to the drum sound. The physical method of frequency dependent damping would require great modifications to setupcoef, and value, and will not be done at this time.
The filtering will be done at m_max -1 around the entire rim. Remember that now+1 is the new value and now is the past value. */

        if(FreqSwitch)
                {
                for (n=0; n<=n_max; n++)
                        {
                        s[(now+1)%3][m_max-1][n]= vdampb*s[(now+1)%3][m_max-1][n] +vdampa*s[(-
now)%3][m_max-1][n] +vdampb*s[(now+2)%3][m_max-1][n];
                        }
                }

/* Stability constraints may require calculating more time detail than can be replicated at 44100 sample/ second playback. The following lines save only those surface values that will be used in the actual sound output. Because of space constraints the symmetry line data will be saved less often. */

        deci++;
        if(deci%decimate == 0)
                {
                deci=0;
                k++;
                j++;

/*The following lines cause the program to abort if the amplitude of the vibration at the hitpoint exceed MAX_DEFLECT. */
                if (s[(now +1)%3][hitpt][0] > MAX_DEFLECT)
                        {
                        printf("time= %e\tdata= %e\n",elapse,s[(now +1)%3][hitpt][0]);
                        fflush(stdout);
                        error(" The simulation is unstable!");
                        }
                if (s[(now +1)%3][hitpt][0] == NAN)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                {
                printf("time= %e\tdata= %e\n",elapse,s[(now +1)%3][hitpt][0]);
                fflush(stdout);
                error(" The simulation is unstable!");
                }


/* save the symmetry line data every %__ time only, presently 25 per second */
        if(k%(1764*decimate) == 0)
                {
                k=0;
                /*printf("\n\nSymmetry line\n");*/
                for (i=m_max; i>=0; i--)
                        {
                        /*printf("%e ",s[(now +1)%3][i][0]);*/
                        fprintf(fpS,"%e ",s[(now +1)%3][i][0]);
                        }
                for (i=1; i<=m_max; i++)
                        {
                        /*printf("%e ",s[(now +1)%3][i][n_max]);*/
                        fprintf(fpS,"%e ",s[(now +1)%3][i][n_max]);
                        }
                /*printf("\n\n");*/
                fprintf(fpS,"\n");
                }

/* save the output data. The routine must be altered for the first acceleration computation since the initial
conditions are in the oldest layer of s[][][], and they are velocities not displacement. */

/*first the hitpoint displacement data*/
        fprintf(fpT_d,"%e\n",s[(now +1)%3][hitpt][0]);

/*next the hitpoint accelation data*/
        if(coefndx == 1)
                accel_hitpt = s[(now +2)%3][hitpt][0] - 2*s[(now)%3][hitpt][0] + s[(now
+1)%3][hitpt][0];
/*
        else
                accel_hitpt = -2*s[(now)%3][hitpt][0] + s[(now +1)%3][hitpt][0];
*/
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
accel_hitpt *= accel_scale;
fprintf(fpT_a,"%e\n",accel_hitpt);

/*finally add up the contributions of the whole drumhead. First the center point. Then the symetry line ,only
once, and finally the other points, doubled, since they have symetrical counterparts. */
if(coefndx == 1)
        {
        accel_total = s[(now +2)%3][0][0] - 2*s[(now)%3][0][0] + s[(now +1)%3][0][0];
        for (ii=1; ii<m_max; ii++)
                accel_total += s[(now +2)%3][ii][0] - 2*s[(now)%3][ii][0] + s[(now
+1)%3][ii][0] + s[(now +2)%3][ii][n_max] - 2*s[(now)%3][ii][n_max] + s[(now +1)%3][ii][n_max];
        for (ii=1; ii<m_max; ii++)
                for (jj=1; jj<n_max; jj++)
                        accel_total += 2 * (s[(now +2)%3][ii][jj] - 2*s[(now)%3][ii][jj] +
s[(now +1)%3][ii][jj]);
        }
/*
else
        {
        accel_total = -2*s[(now)%3][0][0] +s[(now +1)%3][0][0];
        for (ii=1; ii<m_max; ii++)
                accel_total += -2*s[(now)%3][ii][0] + s[(now +1)%3][ii][0] - 2*s[(now)%3][i-
i][n_max] + s[(now +1)%3][ii][n_max];
        for (ii=1; ii<m_max; ii++)
                for (jj=1; jj<n_max; jj++)
                        accel_total += 2*(-2*s[(now)%3][ii][jj] + s[(now +1)%3][ii][jj]);
        }
*/


accel_total *= accel_scale;
fprintf(fpT_A,"%e\n",accel_total);


/*print out the data only every %___ time; the %___ value should be one greater than the j init line */
if(j%1000 == 0)
        {
        j=0;
        printf("time= %e\tdata= %e\n",elapse,s[(now +1)%3][hitpt][0]);
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                    fflush(stdout);
                    }

/*save the maximum decimated amplitudes to scale the data */

/*first the hitpoint displacement */
            if(fabs(s[(now +1)%3][hitpt][0]) > max_amp)
                    {
                    max_amp= fabs(s[(now +1)%3][hitpt][0]);
                    tmax_amp= elapse;
                    }

/*next the hitpoint acceleration*/
                    if(fabs(accel_hitpt) > max_amp_a)
                    {
                    max_amp_a= fabs(accel_hitpt);
                    tmax_amp_a= elapse;
                    }

/*finally the far-field average acceleration*/
                    if(fabs(accel_total) > max_amp_A)
                    {
                    max_amp_A= fabs(accel_total);
                    tmax_amp_A= elapse;
                    }


            }
```

/*compute the new value of the tension: If the tension count is less than 20 then recompute the area of the drumhead. If it is significantly greater than the drum area at rest then recompute the tension, and set the tension count to zero. The tension count is used to prevent wasted time during the latter part of the simulation where the tension essentially remains at the static tension level. The count is triggered at 20 to prevent a situation where the tension takes some time to build up from the initial strike.
*/

```
    if(TenSwitch)
            {
            if(tencount < 20)
                    {
```

/* the value at the center computes to zero since r=0 ??? I will include the lines but comment them out until I can think this through.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                templatenumber= template[0][0];
                deltaA = s[(now+1)%3][0][0] * value(now, templatenumber, s, coef, 2, 0, 0, n_max)
*XXX;
*/


                for (m=1; m<m_max; m++)
                        {
                        for (n=0; n<=n_max; n++)
                                {
                                templatenumber= template[m][n];
                                deltaA += s[(now+1)%3][m][n] * value(now, templatenumber, s,
coef, 2, m, n, n_max) *m;
                                }
                        }
                deltaA *= (-dr*dr*da/2);
printf("the delta area = %e\n", deltaA);
                if(deltaA > (0.00000001 * s_area))
                        {
                        tensionP= deltaTcoef * deltaA + tensionD;
/*printf("\tthe tension is = %f\n", tensionP);
fflush(stdout);
*/
                        tencount = 0;
                        printf("reset tension\n");
                        }
                else
                        {
                        tensionP = tensionD;
                        ++tencount;
                        }
printf("the tension= %f tencount= %d time= %e data= %e\n",tensionP, tencount,elapse,s[(now
+1)%3][hitpt][0]);
fflush(stdout);
                }

/*recompute coeffcient array: If the tension changes, then the coefficient array must be recomputed. If tension count is 0 the tension is greater than static and has just changed. If the tension count is 1 the tension is static but the coefficient array still has the modified values in it, so it needs to be recomputed using tensionD. Once the tension count is 2 or more the coefficient array will not need to be recomputed.*/

        if(tencount<2)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                        {
                        printf("\treset coef \t\ttencount= %d\n",tencount);
                        fflush(stdout);
setupcoef(coef, tensionP, m_max, dr, da, dt, D, Cr, RCr, vsS, csS, Acoef, Mcoef);
/*
for (i=0; i <15; i++)
        {
        printf("coef[0][0][%d] = %e\n", i, coef[0][0][i]);
        printf("coef[1][0][%d] = %e\n", i, coef[1][0][i]);

        printf("coef[0][5][%d] = %e\n", i, coef[0][5][i]);
        printf("coef[1][5][%d] = %e\n", i, coef[1][5][i]);
        }
fflush(stdout);
*/
                        }
                }
/* increment the placeholder variables and in preparation for the next iteration.*/
        elapse += dt;
        now = (++now)%3;
        coefndx= 1;
        }

/*close the temporary files, and the symmetry line file */
fclose(fpS);
fclose(fpT_d);
fclose(fpT_a);
fclose(fpT_A);

/*open the temporary files for reading and scale the data into appropriate sound files*/
if ((fpT_d = fopen(nameT_d, "r")) == NULL)
        error("unable to open file %s", nameT_d);
printf("Max hitpoint displacement amplitude= %e at time= %e\n", max_amp, tmax_amp);

if ((fpT_a = fopen(nameT_a, "r")) == NULL)
        error("unable to open file %s", nameT_a);
printf("Max hitpoint acceleration amplitude= %e at time= %e\n", max_amp_a, tmax_amp_a);

if ((fpT_A = fopen(nameT_A, "r")) == NULL)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        error("unable to open file %s", nameT_A);
printf("Max far-field ave acceleration amplitude= %e at time= %e\n", max_amp_A, tmax_amp_A);
fflush(stdout);


/* This section saves onto the sound files one-tenth of a second of zeros to give some room for editing the
sound files.*/
for(i=0; i<4410; i++)
        {
        if (write(fdO_d, &siZERO, 2) != 2)
                error("cannot write %s", nameO_d);
        if (write(fdO_a, &siZERO, 2) != 2)
                error("cannot write %s", nameO_a);
        if (write(fdO_A, &siZERO, 2) != 2)
                error("cannot write %s", nameO_A);
        }

/*convert the hitpoint displacement data*/
scalefactor = MAXINTT/max_amp;
while (fscanf(fpT_d, "%le", &amplitude) != EOF)
        {
        sound_data = amplitude * scalefactor;
        if (write(fdO_d, &sound_data, 2) != 2)
                error("cannot write %s", nameO_d);
        }

/*convert the hitpoint acceleration data*/
scalefactor = MAXINTT/max_amp_a;
while (fscanf(fpT_a, "%le", &amplitude) != EOF)
        {
        sound_data = amplitude * scalefactor;
        if (write(fdO_a, &sound_data, 2) != 2)
                error("cannot write %s", nameO_a);
        }

/*convert the far-field average acceleration data*/
scalefactor = MAXINTT/max_amp_A;
while (fscanf(fpT_A, "%le", &amplitude) != EOF)
        {
        sound_data = amplitude * scalefactor;
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        if (write(fdO_A, &sound_data, 2) != 2)
                error("cannot write %s", nameO_A);
        }

/* This section saves onto the sound files one-tenth of a second of zeros to give some room for editing the
sound files.*/
for(i=0; i<4410; i++)
        {
        if (write(fdO_d, &siZERO, 2) != 2)
                error("cannot write %s", nameO_d);
        if (write(fdO_a, &siZERO, 2) != 2)
                error("cannot write %s", nameO_a);
        if (write(fdO_A, &siZERO, 2) != 2)
                error("cannot write %s", nameO_A);
        }

/*clean up and exit*/
close(fdO_d);
close(fdO_a);
close(fdO_A);
fclose(fpT_d);
fclose(fpT_a);
fclose(fpT_A);
remove(nameT_d);
remove(nameT_a);
remove(nameT_A);
printf("\nProcess Exited Correctly\n");
fflush(stdout);
return;
}
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

/* DSsound.h, by Jim Irwin

This is a file that contains all the templates and any global defines.

```
These are the default program parameters */

#define DEFAULTSTEM "DrumSoundTest"
#define DRDIA .25
#define F01 125
#define M_MAX 250
#define N_MAX 100
#define T_MAX 600.0
#define MAX_DEFLECT 1
#define HITSTRENGTH 150
#define HITLOCATION 0.80
#define Z01 2.5040 /* first bessel zero (Z02 5.520 second bessel zero) */
#define THICKNESS 254.0e-6 /* meters 0.01 inches */
#define MAXINTT 32760
#define SAMPLE_RATE 44100
#define NUMBER_OF_CHANNELS 1
#define MAGIC_NUMBER_SND 0x2e736e64
#define EY 5e9 /*Young's modulus for mylar. */
#define Ddf 7.980e-3
#define CRdf 5
#define RHOMYLAR 1390 /* kg per meter cubed */
#define RHOAIR 1.21 /*kg per meter cubed */

enum boolean {FALSE, TRUE};

double value(int now, int templatenumber, double s[3][M_MAX +1][N_MAX +1], double coef[3][M_MAX
+1][15], int coefndx, int m, int n, int n_max);
void setupcoef( double coef[3][M_MAX +1][15], double T, int m_max, double dr, double da, double dt, dou-
ble D, double Cr, double RCr, int Dvc, int Dcc, int Acoef, int Mcoef);
/*double compute(int now, double s[3][M_MAX +1][N_MAX +1], int m_max, int n_max, int template[M_-
MAX +1][N_MAX +1], double coef[3][M_MAX +1][15]); */
void setuptemplate(int template[M_MAX + 1][N_MAX + 1], int m_max, int n_max);
double strikelength(int now, double s[3][M_MAX +1][N_MAX +1], int m_max, int n_max, double dr);
void wrNxtHdr(int, int, int);
void error(char *fmt, ...);
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
enum boolean reply_scan(char *s);

typedef struct
{
int magic, headerlength, datasize, datatype, samprate, nochans;
char info[4];
} NXT_HDR;
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

SetupCoef.c

This routine computes the coefficients that are in the standard template for the difference equation. The coefficients depend upon the tension and the radial distance. This dependence upon the tension means the coefficients need to be recomputed at times. (See *DSTop.c.*) The radial effect is caused by the choice of a polar mesh and therefore the radius figures in nearly every coefficient. The coefficients are stored in a three-dimensional array indexed by A, B, and C.

The A-dimension has the following meaning:

- A=0 is the set of coefficients used for the first iteration of the main loop in the main program. It assumes that the information given by the initial conditions exists as a velocity and thus the coefficients take on a special form.
- A=1 is the set of coefficients used in the rest of the iterations.
- A=2 is the set of coefficients used to compute the surface area of the drumhead.

The B-dimension is used to index the coefficients correctly for each different radius used in the simulation.

The C-dimension is used to indicate which coefficient is which in the actual template. The numbering scheme follows the coefficient numbers listed in Appendix B.

There are a few other tasks performed in this routine. The following switches operate in this routine: rim damping, radiation damping, variable stiffness, center point stiffness, air density coefficient, and membrane density coefficient, (see Appendix D, *DSinfo* for details).

```
/*
this function sets up the coefficient array so that the function "value" may compute a node magnitude. The
coefficients are dependent upon the inter-node spacing, and thus dependent upon r=m*dr. They are also
dependent upon tension, a semi-constant, and other constants. The coef array will be accessed by the
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

radial node number, and an arbitrary number. The arbitrary number comes from my notebook pages III:8-10

for easy reference

T is the tension
D is the stiffness coefficient
M is the total mass per unit area
Cr is the velocity damping coefficient
RCr is the rim velocity damping coeffecient
Dvc is the coefficient that turns on or off the variable stiffness
Dcc is the coefficient that turns on or off the center stiffness
Acoef is the coefficient that adjusts the density of air 0 - 1
Mcoef is the coefficient that adjusts the density of the membrane */

```
#include <math.h>
#include "DSsound.h"

void setupcoef( double coef[3][M_MAX +1][15], double T, int m_max, double dr, double da, double dt, double D, double Cr, double RCr, int Dvc, int Dcc, double Acoef, double Mcoef)
{

int m;   /* radial index */
int i;   /* index variable used in testing */

double M;/* total mass per unit area*/
double massA;/* unit radiation mass, due to air per unit area*/
double massM;/* unit mass of the Mylar film per unit area*/
double PI=4.0*atan(1.0);/* the constant */
double m2;/* m squared */
double m3;/* m cubed */
double m4;/* m fourthed */
double cr;/* damping coefficient */
double dr2 = pow(dr,2);/* dr squared */
double da2 = pow(da,2);/* da squared */
double dt2 = pow(dt,2);/* dt squared */
double dr4 = pow(dr,4);/* dr fourthed */
double da4 = pow(da,4);/* da fourthed */
double c0, c1;/*the coefficient of t+1 common to all other coefficients */

massA = Acoef * RHOAIR * 8.0 * dr * pow((da/2),0.50)/(3.0 * pow(PI, 1.50));
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
massM = Mcoef * RHOMYLAR * THICKNESS;

/* special case for m=zero, the center of the drum; those not listed are zero - set in main.*/

        M = massM + (Acoef * RHOAIR) * 8 * dr/ (PI * 3.0);


        c0 = dt2/(2*M);

        coef[0][0][1] = c0*(2*M/dt -Cr*M); /*s[i-1][][] is Vo*/

        coef[0][0][2] = -c0*D*Dcc*(1 - Dvc)/dr4;

        coef[0][0][4] = c0*(4*D*Dcc*(1 - Dvc)/dr4 + T/dr2);

        coef[0][0][8] = c0*(-6*D*Dcc*(1 - Dvc)/dr4 -2*T/dr2 +2*M/dt2);

        coef[0][0][12] = coef[0][0][4];

        coef[0][0][14] = coef[0][0][2];


        c1 = 2*dt2/(2*M + Cr*M*dt);

        coef[1][0][1] = c1*(-M/dt2 + Cr*M/(2*dt));

        coef[1][0][2] = -c1*D*Dcc*(1 - Dvc)/dr4;

        coef[1][0][4] = c1*(4*D*Dcc*(1 - Dvc)/dr4 + T/dr2);

        coef[1][0][8] = c1*(-6*D*Dcc*(1 - Dvc)/dr4 -2*T/dr2 +2*M/dt2);

        coef[1][0][12] = coef[1][0][4];

        coef[1][0][14] = coef[1][0][2];

/*used to compute the surface area */
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        coef[2][0][4] = 1.0/dr2;

        coef[2][0][8] = -2.0/dr2;

        coef[2][0][12] = coef[2][0][4];


/* now for the rest */
for (m=1; m<m_max; m++)
        {
        m2 = pow(m,2);
        m3 = pow(m,3);
        m4 = pow(m,4);
        M = massM + massA * m; /*given radius find mass*/

/*This sets up the damping to be greater at the rim than in the interior of the drum head.*/
        if(m == m_max-1)
                cr = Cr + RCr;
        else if(m == m_max-2)
                cr = Cr + RCr/2.;
        else if(m == m_max-3)
                cr = Cr + RCr/4.;
        else
                cr = Cr;


        c0 = dt2/(2*M*m4*dr4);

        coef[0][m][1] = c0*(2*M*m4*dr4/dt -cr*M*m4*dr4); /*s[i-1][][] is Vo*/

        coef[0][m][2] = c0*(-D*(1.0 - Dvc*exp(-3*m/m_max))*m4 +D*(1.0 - Dvc*exp(-3*m/m_max))*m3);

        coef[0][m][3] = c0*(-2*D*(1.0 - Dvc*exp(-3*m/m_max))*m2/da2 -D*(1.0 - Dvc*exp(-3*m/
m_max))*m/da2);

        coef[0][m][4] = c0*(4*D*(1.0 - Dvc*exp(-3*m/m_max))*m4 -2*D*(1.0 - Dvc*exp(-3*m/m_max))*m3
+T*m4*dr2 +D*(1.0 - Dvc*exp(-3*m/m_max))*m2 -T*m3*dr2/2 +D*(1.0 - Dvc*exp(-3*m/m_max))*m/2
+4*D*(1.0 - Dvc*exp(-3*m/m_max))*m2/da2 +2*D*(1.0 - Dvc*exp(-3*m/m_max))*m/da2);

        coef[0][m][5] = coef[0][m][3];
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
coef[0][m][6] = -c0*D*(1.0 - Dvc*exp(-3*m/m_max))/da4;

coef[0][m][7] = c0*(4*D*(1.0 - Dvc*exp(-3*m/m_max))*m2/da2 +T*m2*dr2/da2 -4*D*(1.0 -
Dvc*exp(-3*m/m_max))/da2 +4*D*(1.0 - Dvc*exp(-3*m/m_max))/da4);

coef[0][m][8] = c0*(-6*D*(1.0 - Dvc*exp(-3*m/m_max))*m4 -2*T*m4*dr2 -2*D*(1.0 - Dvc*exp(-3*m/
m_max))*m2 -8*D*(1.0 - Dvc*exp(-3*m/m_max))*m2/da2 -2*T*m2*dr2/da2 +8*D*(1.0 - Dvc*exp(-3*m/
m_max))/da2 -6*D*(1.0 - Dvc*exp(-3*m/m_max))/da4 +2*M*m4*dr4/dt2);

coef[0][m][9] = coef[0][m][7];

coef[0][m][10] = coef[0][m][6];

coef[0][m][11] = c0*(-2*D*(1.0 - Dvc*exp(-3*m/m_max))*m2/da2 +D*(1.0 - Dvc*exp(-3*m/
m_max))*m/da2);

coef[0][m][12] = c0*(4*D*(1.0 - Dvc*exp(-3*m/m_max))*m4 +2*D*(1.0 - Dvc*exp(-3*m/
m_max))*m3 +T*m4*dr2 +D*(1.0 - Dvc*exp(-3*m/m_max))*m2 +T*m3*dr2/2 -D*(1.0 - Dvc*exp(-3*m/
m_max))*m/2 +4*D*(1.0 - Dvc*exp(-3*m/m_max))*m2/da2 -2*D*(1.0 - Dvc*exp(-3*m/m_max))*m/da2);

coef[0][m][13] = coef[0][m][11];

coef[0][m][14] = c0*(-D*(1.0 - Dvc*exp(-3*m/m_max))*m4 -D*(1.0 - Dvc*exp(-3*m/m_max))*m3);



c1 = 2*dt2/((m4*dr4)*(2*M + cr*M*dt));

coef[1][m][1] = c1*(-M*m4*dr4/dt2 +cr*M*m4*dr4/(2*dt));

coef[1][m][2] = c1*(-D*(1.0 - Dvc*exp(-3*m/m_max))*m4 +D*(1.0 - Dvc*exp(-3*m/m_max))*m3);

coef[1][m][3] = c1*(-2*D*(1.0 - Dvc*exp(-3*m/m_max))*m2/da2 -D*(1.0 - Dvc*exp(-3*m/
m_max))*m/da2);

coef[1][m][4] = c1*(4*D*(1.0 - Dvc*exp(-3*m/m_max))*m4 -2*D*(1.0 - Dvc*exp(-3*m/m_max))*m3
+T*m4*dr2 +D*(1.0 - Dvc*exp(-3*m/m_max))*m2 -T*m3*dr2/2 +D*(1.0 - Dvc*exp(-3*m/m_max))*m/2
+4*D*(1.0 - Dvc*exp(-3*m/m_max))*m2/da2 +2*D*(1.0 - Dvc*exp(-3*m/m_max))*m/da2);

coef[1][m][5] = coef[1][m][3];
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        coef[1][m][6] = -c1*D*(1.0 - Dvc*exp(-3*m/m_max))/da4;

        coef[1][m][7] = c1*(4*D*(1.0 - Dvc*exp(-3*m/m_max))*m2/da2 +T*m2*dr2/da2 -4*D*(1.0 -
Dvc*exp(-3*m/m_max))/da2 +4*D*(1.0 - Dvc*exp(-3*m/m_max))/da4);

        coef[1][m][8] = c1*(-6*D*(1.0 - Dvc*exp(-3*m/m_max))*m4 -2*T*m4*dr2 -2*D*(1.0 - Dvc*exp(-3*m/
m_max))*m2 -8*D*(1.0 - Dvc*exp(-3*m/m_max))*m2/da2 -2*T*m2*dr2/da2 +8*D*(1.0 - Dvc*exp(-3*m/
m_max))/da2 -6*D*(1.0 - Dvc*exp(-3*m/m_max))/da4 +2*M*m4*dr4/dt2);

        coef[1][m][9] = coef[1][m][7];

        coef[1][m][10] = coef[1][m][6];

        coef[1][m][11] = c1*(-2*D*(1.0 - Dvc*exp(-3*m/m_max))*m2/da2 +D*(1.0 - Dvc*exp(-3*m/
m_max))*m/da2);

        coef[1][m][12] = c1*(4*D*(1.0 - Dvc*exp(-3*m/m_max))*m4 +2*D*(1.0 - Dvc*exp(-3*m/
m_max))*m3 +T*m4*dr2 +D*(1.0 - Dvc*exp(-3*m/m_max))*m2 +T*m3*dr2/2 -D*(1.0 - Dvc*exp(-3*m/
m_max))*m/2 +4*D*(1.0 - Dvc*exp(-3*m/m_max))*m2/da2 -2*D*(1.0 - Dvc*exp(-3*m/m_max))*m/da2);

        coef[1][m][13] = coef[1][m][11];

        coef[1][m][14] = c1*(-D*(1.0 - Dvc*exp(-3*m/m_max))*m4 -D*(1.0 - Dvc*exp(-3*m/m_max))*m3);



/*used to compute the surface area */

        coef[2][m][1] = 0;

        coef[2][m][2] = 0;

        coef[2][m][3] = 0;

        coef[2][m][4] = m4*dr2 -m3*dr2/2;

        coef[2][m][5] = coef[2][m][3];

        coef[2][m][6] = 0;

        coef[2][m][7] = m2*dr2/da2;
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
            coef[2][m][8] = -2*m4*dr2 -2*m2*dr2/da2;

            coef[2][m][9] = coef[2][m][7];

            coef[2][m][10] = coef[2][m][6];

            coef[2][m][11] = 0;

            coef[2][m][12] = m4*dr2 +m3*dr2/2;

            coef[2][m][13] = coef[2][m][11];

            coef[2][m][14] = 0;


            }

return;
}
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

SetupTemplate.c

This routine sets up a two-dimensional array that tells the routine value which template to use at each particular mesh location on the drum surface. The particular templates are listed in Appendix C.

```
/* This function puts an integer in each valid location. A valid location is one where a computation is permitted to occur. The main program initializes all values to zero. Zero will indicate to the computation routine that no template is valid, therefore an error has occurred somewhere. The templates are numbered according to the arbitrary system used in setting them up. (See pages III:11-18). Templates 34, 37, 40, 43, 46, and 49 are redundant because at the center all the templates collapse to the same one. (I will use #31 for the center.) To ease the setup I will over-write nodes starting at the largest set and working to the special case corners. Remember that m_max, and n_max must be at least 7. */

#include "DSsound.h"

void setuptemplate(int template[M_MAX + 1][N_MAX + 1], int m_max, int n_max)
{
int m,n; /* counters */

/* set up the center of the drum, r=0. Since the center of the drum is a condensation of all values of 'n' I need to set all the members of this row to the center template. This will mean that the center of the drum could get calculated in a wierd fashion (multiple, separate calculations of the same mesh point.) but this is easier that making compute re-index every time the 'm' index goes to zero.*/
for (n=0; n<=n_max; n++)
        template[0][n] = 9;

/* set up all active nodes to 1, standard template, except the center */
for (m=1; m<m_max; m++)
        {
        for (n=0; n<=n_max; n++)
                template[m][n] = 1;
        }

/* set up the default radial nodes */
for (m=1; m<m_max; m++)
        {
        template[m][0] = 5;
        template[m][1] = 4;
        template[m][n_max] = 7;
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        template[m][n_max -1] = 6;
        }

/* set up the default angular nodes */
for (n=0; n<=n_max; n++)
        {
        template[m_max -1][n] = 3;
        template[m_max -2][n] = 2;
        template[1][n] = 8;
        }

/* set up the center: zero-axis corner */
template[1][0] = 19;

template[1][1] = 18;

/* set up the rim: zero-axis corner */
template[m_max -1][0] = 13;
template[m_max -2][0] = 11;

template[m_max -1][1] = 12;
template[m_max -2][1] = 10;

/* set up the rim: PI-axis corner */
template[m_max -1][n_max] = 17;
template[m_max -2][n_max] = 15;

template[m_max -1][n_max -1] = 16;
template[m_max -2][n_max -1] = 14;

/* set up the center: PI-axis corner */
template[1][n_max] = 21;

template[1][n_max -1] = 20;

return;
}
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Value.c

This is the main computational routine. It takes the coefficient array, the template array, the displacement array, the mesh location, etc., and uses them to compute a new value for the drumhead at the particular mesh location. The major case statement uses the template number to select which version of the difference equation to evaluate. The case numbers are indexed to the template number.

```
/* this function takes the computation node, template number, and coefficient matrix; and couputes the new
node value. */

#include <math.h>
#include "DSsound.h"

double value(int now, int templatenumber, double s[3][M_MAX +1][N_MAX +1], double coef[3][M_MAX
+1][15], int coefndx, int m, int n, int n_max)
{

double magnitude;/*the new magnitude of the node */

switch (templatenumber)
        {
        case 0:
        default:
                error("error: Invalid template number = %d\nm= %d\tn=%d\n",templatenumber,m,n);
                break;

        case 1: /* standard template */
                magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
+s[now][m][n-2]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n+2]*coef[coefndx][m][10]

+s[now][m+1][n-1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n+1]*-
coef[coefndx][m+1][13]

+s[now][m+2][n]*coef[coefndx][m+2][14];
                    break;

          case 2: /* 2 from rim */
                    magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n-2]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n+2]*coef[coefndx][m][10]

+s[now][m+1][n-1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n+1]*-
coef[coefndx][m+1][13]

+s[now][m+2][n]*0;
                    break;

          case 3: /* 1 from rim */
                    magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n-2]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n+2]*coef[coefndx][m][10]

+s[now][m+1][n-1]*0 +s[now][m+1][n]*0 +s[now][m+1][n+1]*0
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
-s[now][m][n]*coef[coefndx][m][14];
                    break;

          case 4: /* 1 from 0 axis */
                    magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[coefndx][m][8]
+s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n+2]*coef[coefndx][m][10]

+s[now][m+1][n-1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n+1]*-
coef[coefndx][m+1][13]

+s[now][m+2][n]*coef[coefndx][m+2][14];
                    break;

          case 5: /* on 0 axis */
                    magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n+1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n+2]*coef[coefndx][m][6] +s[now][m][n+1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n+2]*coef[coefndx][m][10]

+s[now][m+1][n+1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n+1]*-
coef[coefndx][m+1][13]

+s[now][m+2][n]*coef[coefndx][m+2][14];
                    break;

          case 6: /* 1 from PI axis */
                    magnitude =
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n-2]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n]*coef[coefndx][m][10]

+s[now][m+1][n-1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n+1]*-
coef[coefndx][m+1][13]

+s[now][m+2][n]*coef[coefndx][m+2][14];
                    break;

          case 7: /* on PI axis */
                    magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n-1]*coef[-
coefndx][m-1][5]

+s[now][m][n-2]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n-1]*coef[coefndx][m][9] +s[now][m][n-2]*coef[coefndx][m][10]

+s[now][m+1][n-1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n-
1]*coef[coefndx][m+1][13]

+s[now][m+2][n]*coef[coefndx][m+2][14];
                    break;

          case 8: /* 1 from center */
                    magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][1][n_max -n]*coef[coefndx][m-2][2]
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n-2]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n+2]*coef[coefndx][m][10]

+s[now][m+1][n-1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n+1]*-
coef[coefndx][m+1][13]

+s[now][m+2][n]*coef[coefndx][m+2][14];
                break;

        case 9: /* center */
                magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][2][n_max]*coef[coefndx][2][2]

+s[now][1][n_max-1]*coef[coefndx][1][3] +s[now][1][n_max]*coef[coefndx][1][4] +s[now][1][n_max-1]*coef[-
coefndx][1][5]

+s[now][m][n+2]*coef[coefndx][m][6] +s[now][m][n+1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n+2]*coef[coefndx][m][10]

+s[now][m+1][n+1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n+1]*-
coef[coefndx][m+1][13]

+s[now][m+2][n]*coef[coefndx][m+2][14];
                break;

        case 10: /* 2 from rim, 1 from 0 axis */
                magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[coefndx][m][8]
+s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n+2]*coef[coefndx][m][10]
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
+s[now][m+1][n-1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n+1]*-
coef[coefndx][m+1][13]

+s[now][m+2][n]*0;
                break;

        case 11: /* 2 from rim, on 0 axis */
                magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n+1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n+2]*coef[coefndx][m][6] +s[now][m][n+1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n+2]*coef[coefndx][m][10]

+s[now][m+1][n+1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n+1]*-
coef[coefndx][m+1][13]

+s[now][m+2][n]*0;
                break;

        case 12: /* 1 from rim, 1 from 0 axis */
                magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[coefndx][m][8]
+s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n+2]*coef[coefndx][m][10]

+s[now][m+1][n-1]*0 +s[now][m+1][n]*0 +s[now][m+1][n+1]*0

-s[now][m][n]*coef[coefndx][m][14];
                break;
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        case 13: /* 1 from rim, on 0 axis */
                magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n+1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n+2]*coef[coefndx][m][6] +s[now][m][n+1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n+2]*coef[coefndx][m][10]

+s[now][m+1][n-1]*0 +s[now][m+1][n]*0 +s[now][m+1][n+1]*0

-s[now][m][n]*coef[coefndx][m][14];
                break;

        case 14: /* 2 from rim, 1 from PI axis */
                magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n-2]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n]*coef[coefndx][m][10]

+s[now][m+1][n-1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n+1]*-
coef[coefndx][m+1][13]

+s[now][m+2][n]*0;
                break;

        case 15: /* 2 from rim, on PI axis */
                magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n-1]*coef[-
coefndx][m-1][5]

+s[now][m][n-2]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n-1]*coef[coefndx][m][9] +s[now][m][n-2]*coef[coefndx][m][10]

+s[now][m+1][n-1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n-
1]*coef[coefndx][m+1][13]

+s[now][m+2][n]*0;
                    break;

          case 16: /* 1 from rim, 1 from PI axis */
                    magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n-2]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n]*coef[coefndx][m][10]

+s[now][m+1][n-1]*0 +s[now][m+1][n]*0 +s[now][m+1][n+1]*0

-s[now][m][n]*coef[coefndx][m][14];
                    break;

          case 17: /* 1 from rim, on PI axis */
                    magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][m-2][n]*coef[coefndx][m-2][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n-1]*coef[-
coefndx][m-1][5]

+s[now][m][n-2]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n-1]*coef[coefndx][m][9] +s[now][m][n-2]*coef[coefndx][m][10]
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
+s[now][m+1][n-1]*0 +s[now][m+1][n]*0 +s[now][m+1][n+1]*0

-s[now][m][n]*coef[coefndx][m][14];
                break;

        case 18: /* 1 from center, 1 from 0 axis */
                magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][1][n_max-1]*coef[coefndx][1][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[coefndx][m][8]
+s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n+2]*coef[coefndx][m][10]

+s[now][m+1][n-1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n+1]*-
coef[coefndx][m+1][13]

+s[now][m+2][n]*coef[coefndx][m+2][14];
                break;

        case 19: /* 1 from center, on 0 axis */
                magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][1][n_max]*coef[coefndx][1][2]

+s[now][m-1][n+1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n+2]*coef[coefndx][m][6] +s[now][m][n+1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n+2]*coef[coefndx][m][10]

+s[now][m+1][n+1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n+1]*-
coef[coefndx][m+1][13]

+s[now][m+2][n]*coef[coefndx][m+2][14];
                break;
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        case 20: /* 1 from center, 1 from PI axis */
                magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][1][1]*coef[coefndx][1][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n+1]*coef[-
coefndx][m-1][5]

+s[now][m][n-2]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n+1]*coef[coefndx][m][9] +s[now][m][n]*coef[coefndx][m][10]

+s[now][m+1][n-1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n+1]*-
coef[coefndx][m+1][13]

+s[now][m+2][n]*coef[coefndx][m+2][14];
                break;

        case 21: /* 1 from center, on PI axis */
                magnitude =
s[(now+2)%3][m][n]*coef[coefndx][m][1]

+s[now][1][0]*coef[coefndx][1][2]

+s[now][m-1][n-1]*coef[coefndx][m-1][3] +s[now][m-1][n]*coef[coefndx][m-1][4] +s[now][m-1][n-1]*coef[-
coefndx][m-1][5]

+s[now][m][n-2]*coef[coefndx][m][6] +s[now][m][n-1]*coef[coefndx][m][7] +s[now][m][n]*coef[-
coefndx][m][8] +s[now][m][n-1]*coef[coefndx][m][9] +s[now][m][n-2]*coef[coefndx][m][10]

+s[now][m+1][n-1]*coef[coefndx][m+1][11] +s[now][m+1][n]*coef[coefndx][m+1][12] +s[now][m+1][n-
1]*coef[coefndx][m+1][13]

+s[now][m+2][n]*coef[coefndx][m+2][14];
                break;

        }

return magnitude;
}
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

/* this function creates and writes the soundfile header for a NeXT soundfile. */

This program is used to place the appropriate header information at the beginning of a NeXT-formatted sound file. If another computer were used to reproduce the sound, this file would need to be converted to one that produced the correct format for that computer.

```
#define MAGIC_NUMBER_SND 0x2e736e64

typedef struct
{
int magic, headerlength, datasize, datatype, samprate, nochans;
char info[4];
} NXT_HDR;

void wrNxtHdr(int fd, int sr, int nchans)
{
NXT_HDR *head;
head=(NXT_HDR *)calloc(1, sizeof(NXT_HDR));
head->magic = MAGIC_NUMBER_SND;
head->headerlength = 28;
head->datasize = 0;
head->datatype = 3;
head->samprate = sr;
head->nochans = nchans;

if (write(fd, head, sizeof(NXT_HDR)) != sizeof(NXT_HDR))
        error("cannot write header to file");
}
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

/* this function prints an error message and dies */

This file is used to print any error messages that occur from improper conditions within the program. It also terminates the program appropriately and sends an error exit status to the screen file.

```
#include <stdio.h>
#include <stdarg.h>

void error(char *fmt, ...)
{
va_list args;
va_start(args, fmt);
fprintf(stderr, "error: ");
vfprintf(stderr, fmt, args);
fprintf(stderr, "\n");
va_end(args);
exit(1);
}
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

/* DSinfo.txt - contains the information for the program

This file contains information and definitions that are useful in understanding the parameters that are needed to run a simulation. It is used as a help file.

DrumGen

By Jim Irwin: written 24 Jan 1994

This program will run a simulation of a mylar drumhead and produce a NeXT format sound file from this simulation. It will also store the amplitude vs. time data for the symmetry line of the drumhead.

Please use the program startDSG to enter the needed parameters to run this program. The parameters are explained below.

filestem: the base name for all the files used by this program

filestem_d.snd = hitpoint displacement sound output file in NeXT format

filestem_a.snd = hitpoint acceleration sound output file in NeXT format

filestem_A.snd = farfield acceleration sound output file in NeXT format

filestem.amp = the amplitude vs. time along the axis of symmetry - floating point format [with a newline between line data-sets].

filestem.tmp = the temporary file used to store the floating point format sound information

filestem.hdr = user entered parameters, and generated values, that will allow a user to duplicate a simulation.

drum_diameter: (floating point) the drum diameter in meters. The drum diameter must be greater than 0.05

F01: (integer) the frequency of the first circular harmonic for the drumhead. This and the drum_diameter are used to calculate the membrane tension at rest.

Stiffness Coefficient: This indicates the plate characteristics of the drum head. It is calculated from the Young's modulus and the poisson's ratio for real drumheads, but can be set to any value for simulations. A pure membrane has a stiffness coefficient of zero. If the stiffness coefficient is not zero the stability multiplier must be set to a value less than 1. (The stiffer the drumhead the smaller the multiplier needed to keep the system stable.

Variable Stiffness Switch: This controls whether the stiffness is uniform across the drumhead, or follows an embedded profile of zero at the center and two time the Stiffness Coefficient at the rim. (The present profile is exponential.)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Center Point Stiffness Switch: Independent of the previous switch, and the Stiffness Coefficient, the stiffness at the center point can be set to zero.

Stability multiplier: There exists a closed-form equation to compute dt, dr, and da for a membrane such that the simulation is stable. None exists for a stiff membrane. Stability can be achieved by reducing dt. This factor is used internally to alter dt by scaling the stability factor "beta."

Damping Coefficient: This is used to set the "dashpot" damping at each node. A larger value will cause the sound to decay faster. This is radiation damping and values less than five are useful.

Rim Damping Coefficient: This is used to set a higher value of damping near the rim by adding RimDamp to Damp. At present the Rim damping at m_max-1 is RimDamp, and at m_max-2 RimDamp/2. Values between 100 and 1000 are useful.

Frequency Dependent Damping Switch: This controls the empirical low-pass filter attached to the rim. The filter is a three-point FIR digital filter. The coefficient values are entered next.

Air Density Coefficient: This multiplies the air mass for each meshpoint. A value of one is atmospheric pressure; zero is pure vacuum.

Membrane Density Coefficient: This multiplies the drumhead density to allow different materials to be simulated.

Variable Tension Switch: This controls whether the simulation will use constant or variable tension in its computations.

simulation_length: (floating point) The number of seconds the simulation will run. radial_mesh_-points, angular_mesh_points, and the tension will determine the time increment, based upon a stability criterion. The simulation length must be less than or equal to T_MAX.

hit_strength: (floating point) The velocity of the drumstick, in meters per second, when it hits the drumhead. The hitstrength must be less than or equal to HITSTRENGTH.

hit_location: (floating point) The radial location of the impact point, with 0 being the center of the drum and 1.0 being the rim.

radial_mesh_points: (integer) The number of mesh points on each radial line. The center of the drum is mesh point 0, the rim is mesh point "radial_mesh_point". The radial mesh point number must be greater than six and less than or equal to M_MAX. The mesh determines the highest frequency produced from the program, and through several equations: dt, da, stability, and rum time. The run time increases drastically for increased mesh sizes. The maximum frequency produced is approximately -- number of radial mesh points times the frequency of the first circular harmonic times ten.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

angular_mesh_points: (integer) This is not entered by the user but calculated from radial_mesh_-points. The number of mesh points along a semicircle. The strike point is at mesh point 0; 180 degrees is at mesh point "angular_mesh_points". The angular mesh point number must be greater than six and less than or equal to N_MAX. It is computed internally to keep the grid as square as possible.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Makefile

When the program is compiled, this file makes certain that all files are current and updates them whenever a change is made in any of the files.

```
VPATH = /workspace/Irwin/SourceCode
DrumSoundgen: DSTop.o next_sound_header.o error.o strikelength.o setuptemplate.o setupcoef.o value.o
        cc DSTop.o next_sound_header.o error.o strikelength.o setuptemplate.o setupcoef.o value.o -lm -o
DSG

DSTop.o: DSsound.h

next_sound_header.o:

error.o:

strikelength.o: DSsound.h

setuptemplate.o: DSsound.h

setupcoef.o: DSsound.h

value.o: DSsound.h
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix E: DISCUSSION OF TIME/FREQUENCY ANALYSIS TECHNIQUES

This dissertation uses two time-varying spectrum analysis techniques, the Phase-Vocoder (PV) and the McAulay-Quatieri (MQ), [Maher 36], to analyze real and simulated drum sounds. Both use a version of the Short-Time Fourier Transform (STFT) in a digital format, a DFT. The PV technique uses phase information to compute the frequency. The MQ technique computes frequency by interpolation of "peaks" in the magnitude spectrum. The particular information required for each individual analysis governed which technique was chosen. It should be noted that a drum produces a rapidly damped set of inharmonic frequencies. This caused some trouble with the PV version of the STFT method because it assumes that the input signal contains harmonic frequencies.

Nevertheless, the technique used most often was the PV method, which generates coefficients for a harmonic series of frequencies. Choosing a fundamental for this series was a matter of trial and error. The idea is to choose a fundamental such that each vibrational mode appears at or near one of the fundamental's harmonics. This worked well for most of the analyses performed for this dissertation. In general, the frequencies produced by the difference equation did not change and this technique accurately represented them. The display routines for this technique permit time, frequency, and amplitude to be displayed simultaneously.

In the particular case where time-variant tension caused frequency shifts with time, the PV technique did not always perform well because the technique can fail when the frequency shift is large enough to cross "bin" boundaries. The MQ technique helped in this regard, as the requirement for a harmonic series was removed. This technique assumes that each time segment analyzed contains a finite number of sinusoids, not necessarily harmonically related, and uses an STFT to reveal them. The DFT is zero padded to provide greater frequency resolution, and "peaks" in the DFT are located by interpolation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and assigned frequency, phase, and amplitude. These peaks are matched to ones in the adjacent time slices to produce frequency "tracks," which are computed over time. The technique is very useful for systems that have non-stationary modes, like a membrane with time-variable tension. However, the major limitation with this technique is with the robustness of its tracking. For high-level signals it works well, but for low-level signals the tracks can appear chaotic. For this reason, the MQ technique was used only where the signal level was high and the frequency shift was large.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix F: HEADER FILES FOR SIMULATIONS AND RECORDINGS

This appendix lists the parameters used for each simulation analyzed to produce a particular figure in the dissertation.

Note: Due to the broad span of time over which the simulations were made, the simulation parameters and their order changed. If a parameter is missing from a list it can be assumed that it was not used (generally this means "set to zero"). This information is needed for using the present simulation program to repeat any of the previous simulations. A few simulations would require altering and recompiling the present simulation software. This will be noted as it occurs.

Recording information is also given, when a recording from a real drum is analyzed to produce a particular figure. Most recordings were made using the following:

- microphone - Shure SM75
- A/D converter - Singular Solutions A/D 64x
- computer - Next running Mach 3.0, using Soundworks software
- drum - Remo Roto Tom with a Ludwig Striders clear head
- drumstick - promark 11A
- performer - James Irwin
- studio - Bradley University laboratory and office, Jobst 248, 340

Recordings made with the vacuum chamber used the same equipment but for the following: the striking mechanism - a solenoid operated drumstick, and the signal transducer - an accelerometer Atochem model ACH-01.

The parameters are listed according to the appropriate *Figure number*, or *Table number*. All units are standard MKS and can be determined by context or reading the software listings.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Information for Figure 1:

The filestem is classicalA
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The stability multiplier is 1.000000
The damping coefficient is 0.000000
The rim damping coefficient is 0.000000
The variable stiffness is OFF
The center stiffness is OFF
The frequency dependent damping is OFF
The frequency dependent damping coefficients are
h0= 0.000000,h+/-1= 0.000000
The air density coefficient is 1.000000
The membrane density coefficient is 1.000000
The variable tension is OFF
The simulation length is 1.000000
The hit strength is 50.000000
The hit location is 0.800000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 100
The computed value of n_max is 31
The actual value of n_max is 31
The maximum value of beta is 0.010166
The scaled maximum value of beta is 0.010166
The final value of beta is 0.007904

The final parameters are:
decimate= 8
dt= 2.834467e-06, sample rate is 8*44100 Hz.
dr= 1.250000e-03, m_max= 100
da= 1.013417e-01, n_max= 31
real beta= 0.007904
rim ratio= 5.067085
center ratio= 9.867606

hitpt= 80
the final hitpt= 80

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Information for Figure 2:

This is a recording of a real drum struck in the normal playing position with strength sufficient to produce a mezzo-forté loudness.

Information for Figure 3:

This is a recording of a real drum struck in the normal playing position with strength sufficient to produce a triple-forté loudness.

Information for Figure 4:

Not a recording

Information for Figure 5:

Not a recording

Information for Figure 6:

This is an overlay of information from two recordings. Each recording was performed by striking the drumhead using a solenoid drumstick, set to produce a mezzo-forté loudness. The recordings were made from a signal from an accelerometer attached to the drumhead. It was attached about an inch from the rim, on the opposite side of the drumhead, (and opposite face), but not directly opposite the hitpoint. One recording was in air and the other in a vacuum chamber. This figure shows the RMS amplitude of the two signals.

Information for Figure 7:

Not a recording

Information for Figure 8:

The filestem is damp5.hdr
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The stability multiplier is 1.000000
The damping coefficient is 5.000000
The simulation length is 1.000000
The hit strength is 50.000000
The hit location is 0.000000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 30

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The computed value of n_max is 17
The actual value of n_max is 17
The maximum value of beta is 0.033023
The scaled maximum value of beta is 0.033023
The final value of beta is 0.011382

The final parameters are:
decimate= 2
dt= 1.133787e-05, sample rate is 2*44100 Hz.
dr= 4.166667e-03, m_max= 30
da= 1.847996e-01, n_max= 17
real beta= 0.011382
rim ratio= 2.771994
center ratio= 5.411268

hitpt= 0
the final hitpt= 0

Information for Figure 9:

same as figure 2.

Information for Figure 10:

This recording was made by attaching the rim to a large rigid table, without affecting the drumhead itself, to prevent energy from being coupled into the rim. The drumhead was then struck in the ordinary playing position with strength sufficient to give a mezzo-forté loudness.

Information for Figure 11:

This is a recording of a real drum struck in the normal playing position, with strength sufficient to produce a mezzo-forté loudness.

Information for Figure 12:

This is a recording of a real drum struck in the normal playing position, with strength sufficient to produce a triple-forté loudness.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Information for Figure 13:

The filestem is freqC2.hdr
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The damping coefficient is 0.000000
The simulation length is 1.000000
The hit strength is 25.000000
The hit location is 0.000000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 20
The computed value of n_max is 14
The actual value of n_max is 14
The maximum value of beta is 0.047941
The final value of beta is 0.020235

The final parameters are:
decimate= 1
dt= 2.267574e-05, sample rate is 1*44100 Hz.
dr= 6.250000e-03, m_max= 20
da= 2.243995e-01, n_max= 14
real beta= 0.020235
rim ratio= 2.243995
center ratio= 4.456338

hitpt= 0
the final hitpt= 0

Information for Figure 14:

Same as for Figure 14.

Information for Figure 15:

Same as Figure 1.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Information for Figure 16:

Not a recording.

Information for Figure 17:

The filestem is AirVacSim0
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The stability multiplier is 1.000000
The damping coefficient is 0.000000
The rim damping coefficient is 0.000000
The variable stiffness is OFF
The air density coefficient is 0.000000
The membrane density coefficient is 1.000000
The simulation length is 1.000000
The hit strength is 50.000000
The hit location is 0.000000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 10
The computed value of n_max is 9
The actual value of n_max is 9
The maximum value of beta is 0.108613
The scaled maximum value of beta is 0.108613
The final value of beta is 0.005059

The final parameters are:
decimate= 1
dt= 2.267574e-05, sample rate is 1*44100 Hz.
dr= 1.250000e-02, m_max= 10
da= 3.490659e-01, n_max= 9
real beta= 0.005059
rim ratio= 1.745329
center ratio= 2.864789

hitpt= 0
the final hitpt= 0

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Information for Figure 18:

The filestem is AirVacSim1
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The stability multiplier is 1.000000
The damping coefficient is 0.000000
The rim damping coefficient is 0.000000
The variable stiffness is OFF
The air density coefficient is 1.000000
The membrane density coefficient is 1.000000
The simulation length is 1.000000
The hit strength is 50.000000
The hit location is 0.000000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 10
The computed value of n_max is 9
The actual value of n_max is 9
The maximum value of beta is 0.108613
The scaled maximum value of beta is 0.108613
The final value of beta is 0.005059

The final parameters are:
decimate= 1
dt= 2.267574e-05, sample rate is 1*44100 Hz.
dr= 1.250000e-02, m_max= 10
da= 3.490659e-01, n_max= 9
real beta= 0.005059
rim ratio= 1.745329
center ratio= 2.864789

hitpt= 0
the final hitpt= 0

Information for Figure 19:

Overlay of Figures 18 and 19.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Information for Figure 20:

This recording was performed by striking the drumhead in a vacuum chamber using a solenoid drumstick, set to produce a mezzo-forté loudness. The recording was made from a signal from an accelerometer attached to the drumhead. It was attached about an inch from the rim, on the opposite side of the drumhead, (and opposite face), but not directly opposite the hitpoint.

information for Figure 21:

This recording was performed by striking the drumhead in air using a solenoid drumstick, set to produce a mezzo-forté loudness. The recording was made from a signal from an accelerometer attached to the drumhead. It was attached about an inch from the rim, on the opposite side of the drumhead, (and opposite face), but not directly opposite the hitpoint.

Information for Figure 22:

The filestem is fullmeml.hdr
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The stability multiplier is 1.000000
The damping coefficient is 5.000000
The simulation length is 3.000000
The hit strength is 100.000000
The hit location is 0.800000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 100
The computed value of n_max is 31
The actual value of n_max is 31
The maximum value of beta is 0.010166
The scaled maximum value of beta is 0.010166
The final value of beta is 0.007904
The final parameters are:
decimate= 8
dt= 2.834467e-06, sample rate is 8*44100 Hz.
dr= 1.250000e-03, m_max= 100
da= 1.013417e-01, n_max= 31
real beta= 0.007904
rim ratio= 5.067085
center ratio= 9.867606
hitpt= 80 the final hitpt= 80

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Information for Figure 23:

This figure is a combination of five soundfiles, but all were generated from one simulation - note the listen points at the end of this section.

The filestem is Listen1
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The stability multiplier is 1.000000
The damping coefficient is 5.000000
The rim damping coefficient is 0.000000
The variable stiffness is OFF
The center stiffness is OFF
The frequency dependent damping is OFF
The frequency dependent damping coefficients are h0= 0.000000, h+/-1= 0.000000
The air density coefficient is 1.000000
The membrane density coefficient is 1.000000
The variable tension is OFF
The simulation length is 3.000000
The hit strength is 100.000000
The hit location is 0.800000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 50
The computed value of n_max is 22
The actual value of n_max is 22
The maximum value of beta is 0.019984
The scaled maximum value of beta is 0.019984
The final value of beta is 0.014052

The final parameters are:
decimate= 3
dt= 7.558579e-06, sample rate is 3*44100 Hz.
dr= 2.500000e-03, m_max= 50
da= 1.427997e-01, n_max= 22
real beta= 0.014052
rim ratio= 3.569992
center ratio= 7.002817
listen 0= 0 listen .1= 5 listen .2= 10 listen .5= 25listen .9= 45
hitpt= 40 the final hitpt= 40

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Information for Figure 24:

This figure is a combination of three simulations: ntest1, ntest2, and ntest3.
The filestem is ntest1.hdr
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The stability multiplier is 1.000000
The damping coefficient is 0.000000
The simulation length is 0.250000
The hit strength is 100.000000
The hit location is 0.000000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 30
The computed value of n_max is 17
The actual value of n_max is 17
The maximum value of beta is 0.033023
The scaled maximum value of beta is 0.033023
The final value of beta is 0.011382

The final parameters are:
decimate= 2
dt= 1.133787e-05 sample rate is 2*44100 Hz.
dr= 4.166667e-03, m_max= 30
da= 1.847996e-01, n_max= 17
real beta= 0.011382
rim ratio= 2.771994
center ratio= 5.411268
hitpt= 0
the final hitpt= 0

The filestem is ntest2.hdr
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The stability multiplier is 1.000000
The damping coefficient is 0.000000
The simulation length is 0.250000
The hit strength is 100.000000

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The hit location is 0.000000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 30
The computed value of n_max is 34
The actual value of n_max is 34
The maximum value of beta is 0.008465
The scaled maximum value of beta is 0.008465
The final value of beta is 0.005059

The final parameters are:
decimate= 3
dt= 7.558579e-06, sample rate is 3*44100 Hz.
dr= 4.166667e-03, m_max= 30
da= 9.239978e-02, n_max= 34
real beta= 0.005059
rim ratio= 1.385997
center ratio= 10.822536
hitpt= 0
the final hitpt= 0

The filestem is ntest4.hdr
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The stability multiplier is 1.000000
The damping coefficient is 0.000000
The simulation length is 0.250000
The hit strength is 100.000000
The hit location is 0.000000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 30
The computed value of n_max is 68
The actual value of n_max is 68
The maximum value of beta is 0.002130
The scaled maximum value of beta is 0.002130
The final value of beta is 0.001821

The final parameters are:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

decimate= 5
dt= 4.535147e-06, sample rate is 5*44100 Hz.
dr= 4.166667e-03, m_max= 30
da= 4.619989e-02, n_max= 68
real beta= 0.001821
rim ratio= 0.692998
center ratio= 21.645072
hitpt= 0
the final hitpt= 0

Information for Tables 5, 6, 7: Membrane Frequencies

The filestem is stiff.0
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The stability multiplier is 1.000000
The damping coefficient is 0.000000
The rim damping coefficient is 0.000000
The variable stiffness is OFF
The center stiffness is OFF
The frequency dependent damping is OFF
The frequency dependent damping coefficients are: h0= 0.000000, h+/-1= 0.000000
The air density coefficient is 1.000000
The membrane density coefficient is 1.000000
The variable tension is OFF
The simulation length is 1.000000
The hit strength is 25.000000
The hit location is 0.000000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 30
The computed value of n_max is 17
The actual value of n_max is 17
The maximum value of beta is 0.033023
The scaled maximum value of beta is 0.033023
The final value of beta is 0.011382

The final parameters are:
decimate= 2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

dt= 1.133787e-05, sample rate is 2*44100 Hz.
dr= 4.166667e-03, m_max= 30
da= 1.847996e-01, n_max= 17
real beta= 0.011382
rim ratio= 2.771994
center ratio= 5.411268
hitpt= 0
the final hitpt= 0

Information for Table 5:

The filestem is stiff.1.hdr
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000800
The stability multiplier is 1.000000
The damping coefficient is 0.000000
The simulation length is 1.000000
The hit strength is 25.000000
The hit location is 0.000000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 30
The computed value of n_max is 17
The actual value of n_max is 17
The maximum value of beta is 0.033023
The scaled maximum value of beta is 0.033023
The final value of beta is 0.011382

The final parameters are:
decimate= 2
dt= 1.133787e-05, sample rate is 2*44100 Hz.
dr= 4.166667e-03, m_max= 30
da= 1.847996e-01, n_max= 17
real beta= 0.011382
rim ratio= 2.771994
center ratio= 5.411268
hitpt= 0
the final hitpt= 0

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Information for Table 6:

The filestem is stiff.25.hdr
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.002000
The stability multiplier is 1.000000
The damping coefficient is 0.000000
The simulation length is 1.000000
The hit strength is 25.000000
The hit location is 0.000000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 30
The computed value of n_max is 17
The actual value of n_max is 17
The maximum value of beta is 0.033023
The scaled maximum value of beta is 0.033023
The final value of beta is 0.011382

The final parameters are:
decimate= 2
dt= 1.133787e-05, sample rate is 2*44100 Hz.
dr= 4.166667e-03, m_max= 30
da= 1.847996e-01, n_max= 17
real beta= 0.011382
rim ratio= 2.771994
center ratio= 5.411268

hitpt= 0
the final hitpt= 0

Information for Table 7:

The filestem is stiff.5.hdr
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.004000
The stability multiplier is 1.000000

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The damping coefficient is 0.000000
The simulation length is 1.000000
The hit strength is 25.000000
The hit location is 0.000000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 30
The computed value of n_max is 17
The actual value of n_max is 17
The maximum value of beta is 0.033023
The scaled maximum value of beta is 0.033023
The final value of beta is 0.011382

The final parameters are:
decimate= 2
dt= 1.133787e-05, sample rate is 2*44100 Hz.
dr= 4.166667e-03, m_max= 30
da= 1.847996e-01, n_max= 17
real beta= 0.011382
rim ratio= 2.771994
center ratio= 5.411268

hitpt= 0
the final hitpt= 0

Information for Figure 25:

This is a recording of a real drum struck in the normal playing position, with strength sufficient to produce a triple-forté loudness.

Information for Figure 26:

This is a recording of a real drum struck in the normal playing position, with strength sufficient to produce a triple-forté loudness.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Information for Figure 27:

The filestem is Tension5B
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The stability multiplier is 1.000000
The damping coefficient is 5.000000
The rim damping coefficient is 0.000000
The variable stiffness is OFF
The center stiffness is OFF
The frequency dependent damping is OFF
The frequency dependent damping coefficients are
h0= 0.000000, h+/-1= 0.000000
The air density coefficient is 1.000000
The membrane density coefficient is 1.000000
The variable tension is ON
The simulation length is 3.000000
The hit strength is 140.000000
The hit location is 0.800000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 50
The computed value of n_max is 22
The actual value of n_max is 22
The maximum value of beta is 0.019984
The scaled maximum value of beta is 0.019984
The final value of beta is 0.014052

The final parameters are:
decimate= 3
dt= 7.558579e-06, sample rate is 3*44100 Hz.
dr= 2.500000e-03, m_max= 50
da= 1.427997e-01, n_max= 22
real beta= 0.014052
rim ratio= 3.569992
center ratio= 7.002817

hitpt= 40
the final hitpt= 40

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Information for Figure 28:

The filestem is vdamp6
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The stability multiplier is 1.000000
The damping coefficient is 3.000000
The rim damping coefficient is 0.000000
The variable stiffness is OFF
The frequency dependent damping is ON
The air density coefficient is 1.000000
The membrane density coefficient is 1.000000
The variable tension is OFF
The simulation length is 3.000000
The hit strength is 75.000000
The hit location is 0.800000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 50
The computed value of n_max is 22
The actual value of n_max is 22
The maximum value of beta is 0.019984
The scaled maximum value of beta is 0.019984
The final value of beta is 0.014052

The final parameters are:
decimate= 3
dt= 7.558579e-06, sample rate is 3*44100 Hz.
dr= 2.500000e-03, m_max= 50
da= 1.427997e-01, n_max= 22
real beta= 0.014052
rim ratio= 3.569992
center ratio= 7.002817

hitpt= 40
the final hitpt= 40

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Information for Figure 29:

The filestem is rimtest10
The drum diameter is 0.250000
The first circular harmonic is 125
The stiffness coefficient is 0.000000
The stability multiplier is 1.000000
The damping coefficient is 2.000000
The rim damping coefficient is 250.000000
The variable stiffness is OFF
The frequency dependent damping is OFF
The air density coefficient is 1.000000
The membrane density coefficient is 1.000000
The tension coefficient is 1.000000
The simulation length is 3.000000
The hit strength is 50.000000
The hit location is 0.800000
The speed of sound, c, is 39.207177
The tension is 542.724787
The user entered value of m_max is 30
The computed value of n_max is 17
The actual value of n_max is 17
The maximum value of beta is 0.033023
The scaled maximum value of beta is 0.033023
The final value of beta is 0.011382

The final parameters are:
decimate= 2
dt= 1.133787e-05, sample rate is 2*44100 Hz.
dr= 4.166667e-03, m_max= 30
da= 1.847996e-01, n_max= 17
real beta= 0.011382
rim ratio= 2.771994
center ratio= 5.411268
hitpt= 24
the final hitpt= 24

Information for Figure 30:

This is a recording of a real drum struck in the normal playing position, with strength sufficient to produce a mezzo-forté loudness.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

REFERENCES

[1] K. Karplus and A. Strong, "Digital synthesis of plucked-string and drum timbres," *Computer Music Journal*, vol. 7, no. 2, pp. 43-55, 1983.

[2] B. Schottstaedt, "The simulation of natural instrument tones using frequency modulation with a complex modulation wave," *Computer Music Journal*, vol. 1, no. 4, pp. 46-50, 1977.

[3] M. Le Brun, "A derivation of the spectrum of frequency modulation with a complex modulating wave," *Computer Music Journal*, vol. 1, no. 4, pp. 51-52, 1977.

[4] J.-M. Adrien, and X. Rodet, "Physical models of instruments, a modular approach, application to strings," in *Proc. 1985 International Computer Music Conference*, 1985, pp. 85-89.

[5] J.-M. Adrien, R. Causse, and X. Rodet, "Sound synthesis by physical models, application to strings," in *Proc 1987 International Computer Music Conference*, 1987, pp. 264-269.

[6] P. Ruiz and L. Hiller, "Synthesizing musical sounds by solving the wave equation for vibrating objects, parts I and II," *Journal of the Audio Engineering Society*, vol. 19, nos. 6, 7, pp. 462-470, 542-551, 1971.

[7] K. Kubiak and K. Crounse, "Digital synthesis of realistic string sounds," *Computer Music Association, ARRAY*, vol. 11, no. 1, pp. 14-21, 1991.

[8] F. Malouf, M. Lentezner, and C. Chafe, "A real time implementation of physical models," in *Proc. 1990 International Computer Music Conference*, 1990, pp. 191-193.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[9] J. Smith III, "Efficient yet accurate models for strings and air columns using sparse lumping of distributed losses and dispersion," presented at Colloquium on Physical Modeling, Grenoble, Switzerland, 1990.

[10] S. Duyne and J. Smith III, "The 2-D digital waveguide mesh," Center for Computer Research in Music and Acoustics (CCRMA) Department of Music, Stanford University, Stanford, CA 94305, 1994.

[11] A. Ramdane, "Modeles numeriques de structures vibrantes mono- et dedimensionnelles pour la synthese sonore par modele physique," Ph.D. dissertation (in French) de l'Ecole Nationale Supérieure des Télécommunications, 1991.

[12] C. A. Anderson, "The acoustics of timpani: an analysis of vibrating circular membranes," M.S. thesis, Department of Physics, Northern Illinois University, DeKalb, IL, 1978.

[13] R. I. Mills, "The timpani as a vibrating circular membrane: an analysis of several secondary effects," M.S. thesis, Department of Physics, Northern Illinois University, DeKalb, IL, 1980.

[14] M. Poisson, "Addition au mémoire," letter to the acadamy of Paris, published in the academy's *Journal*, 1828.

[15] M. G. Lamé, *Lecons sur la Théorie Mathématique de L'Élasticité des Corps Solides.* Paris, France: Bachelier, Imprimeur-Libraire du Bureau des Longitudes et de L'École Polytechnique,1852.

[16] J. W. Strutt, Baron Rayliegh, Sc.D., F.R.S., *The Theory of Sound.* Mineola, NY: Dover Press [originally Macmillan, 1894], LCCCN 45-8804, 1945.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[17] P. J. Comerford and L. K. Kitching, "Disorder sounds in order," in *Proc. 1990 International Computer Music Conference*, 1990, pp. 60-62.

[18] D. E. Hall, "Piano string excitation II: general solution for a hard narrow hammer," *Journal of the Acoustical Society of America*, vol. 81, no. 2, pp. 535-546, 1987.

[19] T. D. Rossing, "Nonlinear effects in percussion instruments," *Percussionist*, vol. 19, no. 3, pp. 68-72, 1982.

[20] D. E. Cahoon, "Frequency-time analysis of the bass drum sound," M.S. thesis, Department of Physics, Brigham Young University, Salt Lake City, Utah, 1970.

[21] H. Fletcher and I. G. Bassett, "Some experiments with the bass drum," *Journal of the Acoustical Society of America*, vol. 64, no. 6, pp. 1570-1576, 1978.

[22] I. G. Bassett, "Vibration and sound of the bass drum," *Percussionist*, vol. 19, no. 3, pp. 50-58, 1982.

[23] R. W. Young, "Inhanmonicity of plain wire piano strings," *Journal of the Acoustical Society of America*, vol. 24, no. 3, pp. 267-273, 1952.

[24] L. Kinsler, A. Frey, A. Coppens, and J. Sanders, *Fundamentals of Acoustics*. third edition, New York, NY: John Wiley & Sons, Inc., 1982.

[25] R. S. Christian, R. E. Davis, and A. Tubis, "Effects of air loading of timpani membrane vibrations," *Journal of the Acoustical Society of America*, vol. 76, no. 5, pp. 1336-1345, 1984.

[26] T. Rossing, C. Anderson, and R. Mills, "Acoustics of timpani," *Percussionist*, vol. 19, no. 3, pp. 18-31, 1982.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[27] W. Soedel, *Vibration of Shell and Plates.* 2nd ed. New York, NY: Marcel Dekker, Inc., 1993.

[28] A. W. Al-Khafaji and J. R. Tooley, *Numerical Methods in Engineering Practice.* Troy, MO: Holt, Rinehart, and Winston, Inc., 1986.

[29] P. M. Morse and K. U. Ingard, *Theoretical Acoustics.* Princeton, NJ: Princeton University Press, 1968.

[30] W. C. Elmore and M. A. Heald, *Physics of Waves.* Mineola, NY: Dover Publications, Inc., 1969.

[31] R. Szilard, *Theory and Analysis of Plates: Classical and Numerical Methods.* Upper Saddle River, NJ: Prentice-Hall, Inc., 1974.

[32] T. D. Rossing, "Acoustics of percussion instruments -- part II," *The Physics Teacher,* May, 1977.

[33] D. E. Hall, "Piano string excitation in the case of small hammer mass," *Journal of the Acoustical Society of America,* vol. 79, no. 1, pp. 141-147, 1986.

[34] I. D. Abrahams (private conversation) Dec. 1995 .

[35] N. Fletcher, R. Perrin and K. Legge, "Nonlinearity and chaos in acoustics," *Acoustics Australia,* vol. 18, no. 1, pp. 9-13, 1990.

[36] R. C. Maher, "An approach for the separation of voices in composite musical signals," Ph. D. dissertation, Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL, 1989.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

VITA

James Hawley Irwin, Jr. was born on 12 August, 1957, in Springfield Illinois. He attended public schools in the Pleasant Plains school district.

In 1980, he received his Bachelor of Science degree in Electrical Engineering, summa cum laude, from Bradley University in Peoria, Illinois. While working towards this degree he was inducted into the following honor societies: Tau Beta Pi, Eta Kappa Nu, Phi Kappa Phi, and Sigma Pi Sigma.

In 1981, he received his Master of Science degree in Electrical Engineering from Bradley University in Peoria, Illinois.

In 1981, he accepted a job at the National Security Agency in Maryland, USA, where he worked for nearly seven years. His general assignments included radiation damage to semiconductors, battery-powered store-and-forward devices, security code algorithms, and team management.

In 1987, he returned to Bradley University as an instructor in the Electrical Engineering Department. His research includes physical (musical) acoustics and analog circuit design.

He is a member of the following organizations: ASA, ICMA, AES, and the AAUP.

He was married in March of 1981 to Linda Kieliszewski and now has four children.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.