Modeling of Plate Impact Dynamics and Noise

by

Jin Qiu
B.S. Electrical Engineering(Acoustics)
Nanjing University 1997

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1999

Author .....
Department of Mechanical Engineering
January 15, 1999

Certified by .....
Zaichun Feng
Associate Professor
Thesis Supervisor

Accepted by .....
Ain A. Sonin
Chairman, Department Committee on Graduate Students

Modeling of Plate Impact Dynamics and Noise

by

Jin Qiu

B.S. Electrical Engineering(Acoustics)

Nanjing University 1997

Submitted to the Department of Mechanical Engineering
on January 15, 1999, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

The noise generation by the impact of thin plates is an example of energy transfer from the low frequency vibration of the fundamental modes to high frequency vibration modes. The noise radiation from the impact is determined by the impact dynamics. This thesis extracts low-dimensional models for the impact dynamics. Detailed analysis is conducted on a single-degree-of-freedom system. Qualitative understanding of the dynamic behavior is achieved through the bifurcation sets which represent the partition of the parameter space into qualitatively different regions. Quantitative results are also obtained on the impact velocity. Experiments are conducted on a Clamped-Free-Clamped-Free thin plate with a striker. Data on both impact velocity and noise level of the plate impact problem are acquired.

Thesis Supervisor: Zaichun Feng
Title: Associate Professor

Acknowledgments

First I want to thank Professor Feng for his patience and support over the past one and half years. Under his guidance, I enjoyed learning lots of Mechanical Engineering knowledge and got valuable experience on how to do research.

I would like to thank Mr. Yu-Hsuan Su for his numerous help to me, academically and personally. Also I want to thank my other officemates Vedantam Srikanth, Napadow Vitaly, Alfred Pettinger for their help to me.

Thanks MIT for being such a wonderful place, where I can meet wonderful professors and students. I really like it.

I would like to thank my parents for their care and support to me. I would like to thank my friends for being my friend.

Finally, thanks goes to Ford Motor Company for funding this project.

This thesis is dedicated to any rattle noise complaintant, and to engineers who work to alleviate the rattle noise problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Rattle noise caused by impact is a major factor in the sound quality control of commercial products especially in automotive industry. The impact often is due to the intermittent contact between adjacent panels. Under external excitation caused by the road roughness or engine vibration, the panel vibration amplitude exceeds the clearance between them; impact thus occurs. See Figure 1-2 for the analog of rattle noise in automobiles.

The approach to study rattle noise can be shown as in figure 1-1.

There have been extensive studies on the noise generation involving impact [1-4]. The main focus in the past has been on the noise generation and radiation. This thesis discusses the noise generation corresponding to steady excitations. Our experimental work indicates that even under constant amplitude sinusoidal excitations, the impact noise can have very complicated characteristics. For example, the impact noise may be intermittent and even chaotic. These observations indicate that while the noise generation and radiation are important issues, the panel *impact dynamics* is essential in determining the noise generation.

In this work, we limit our interest to impact between panels which are separated by small gaps. For small gaps, impact can occur for small panel deflections. Thus a linear theory can predict the clearance required to avoid impact. When panel

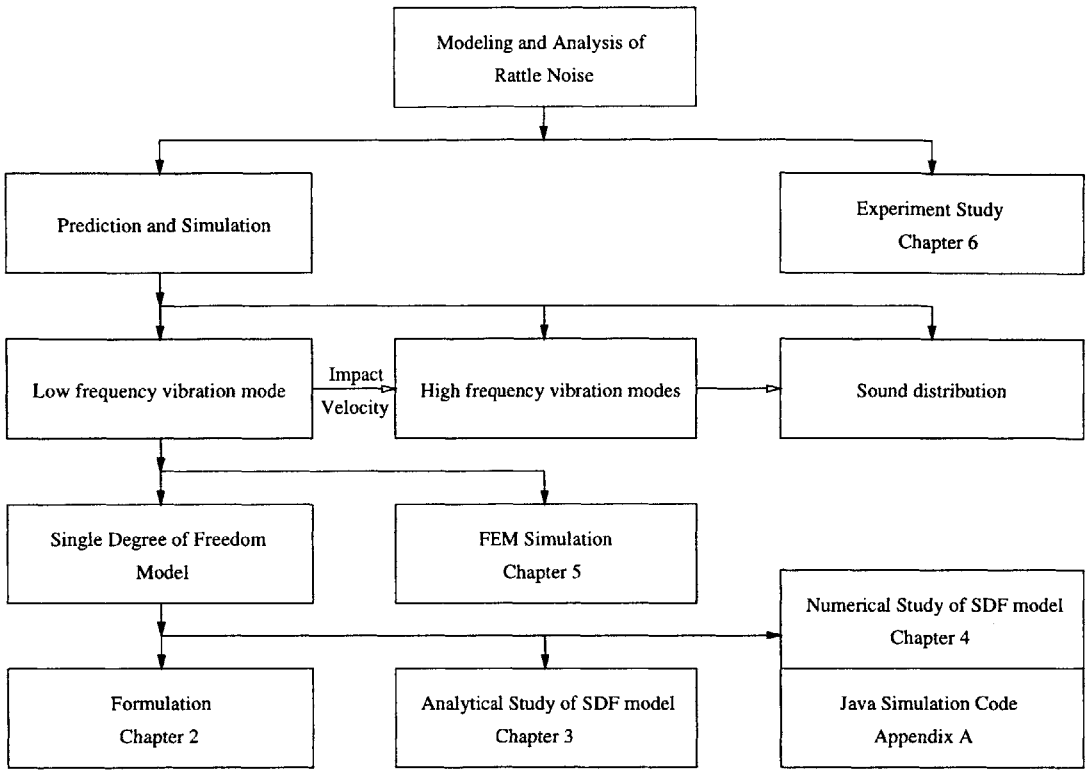


Figure 1-1: The approach to rattle noise study.

dynamics become nonlinear owing to the impact, we will assume that small deflection linear elastic plate equations are valid *between* impacts.

With the above assumptions, it seems that linear plate theory is sufficient to predict the conditions for the onset of impact noise that a designer should avoid. As we will see in this thesis, the linear theory underestimates the regions of potential impact. Furthermore, In cases where it is not feasible to completely avoid impact, certain impact can be tolerated provided the impact noise is within an acceptable bound. For these reasons, a nonlinear analysis of the panel impact dynamics is needed.

The panel *impact dynamics* is very complicated because of the nonlinearity associated with the impact. The complexity is analogous to that seen in a single-degree-of-freedom impact oscillator, which has been studied in the literature [5-7]. In particular, impact oscillators exhibit grazing bifurcation. Chaotic dynamics occurs near grazing bifurcation. This aspect is analogous to those of piecewise linear oscillators [8-9].

1.2 Thesis overview

As shown in Figure 1-1, the outline of this thesis is as follows.

In chapter 2, we first obtain a discrete formulation of the plate impact problem using normal modes. Such a formulation allows us to capture the plate dynamics using just a few modes. We then analyze, in chapter 3, the one-degree-of-freedom model in detail in order to demonstrate some unique characteristics of grazing bifurcations. In particular, at a grazing bifurcation, *many* periodic orbits are born. Most of them terminate at secondary grazing bifurcations, at the end of chapter 3, we get the parameter space characteristics of the periodic dynamic pattern. In chapter 4, we use the numerical integration methods to show the non-periodic pattern of the SDF vibration and more importantly, the impact velocity which we believe is the main factor affecting the rattle noise level. FEM simulation for the plate dynamics is presented in chapter 5. finally in chapter 6, we report the experimental result. In chapter 7, major results of this thesis is stated. The Java program code for the simulation of the single degree of freedom system are included in the appendix.

(When excited, it makes sound.)

Figure 1-2: Analog of rattle noise in automobiles

Chapter 2

Modeling the Rattle Vibration as a Single Degree of Freedom Motion

2.1 Modeling of the vibration parts and excitation

The rattle noise is usually generated by a transversely-excited panel impacting on some thing near it, such as a "striker" or another flexible panel. In this thesis, we only consider the plate-striker system. To formalize the problem, we model the impacting parts, excitation as follows:

- The plate is an even-thickness rectangular one. It can have different boundary conditions.
- The striker, to which the plate impacts, is located at a small distance away from the center of the plate. Compared to the plate, the striker is a very rigid one, so we assume it is fixed at the other end.
- The excitation to the plate comes from the ground acceleration of the car body, which in turn comes from either road toughness or engine vibration. Because the

car chassis and tire system serve as some kind of filter to the original excitation, the ground acceleration as viewed by the plate-striker system is assumed to be a sinusoidal one,

We begin our analysis upon the above system and will make further assumptions later about the impact process.

2.2 Normal mode formulation for the plate vibration

The plate has infinitely many modes that can be generated during the rattle noise vibration. Generally the low modes have larger displacement and velocity amplitude, thus having the main contribution to the plate overall dynamics; the high modes have relatively small displacement and velocity, but they are the main factor in producing noise. In this section, we formulate the equation of motion for all the modes when the plate is away from the striker.

Usually the panel vibration is excited by some ground motion which is transverse to the plate, the excitation can be modeled as an equivalent transversal pressure exerted on the panel.

We consider a thin rectangular plate with dimensions given in Figure 2-1. If the The plate is subjected to transversal ground acceleration

$$a_e(t) = A_e cos(\omega_e t - \phi) \tag{2.1}$$

then the equivalent transversal loading is

$$p_e(t) = P_e cos(\omega_e t - \phi) \tag{2.2}$$

where

$$P_e = \rho h_p A_e \tag{2.3}$$

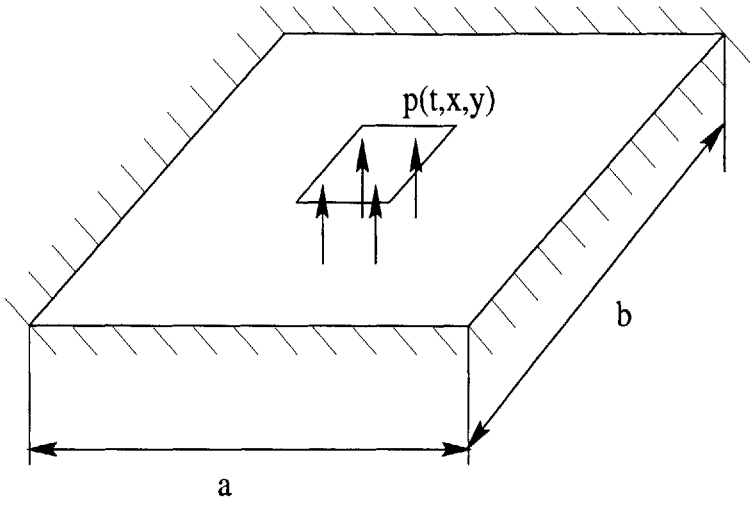


Figure 2-1: Plate geometry

and ρ is the density of the plate material and h_p is the thickness of the plate. We use normal modes to describe the small amplitude vibration of the thin plate:

$$w(x,y,t) = \sum_{i=1}^{n} x_i(t) W_i(x,y) \tag{2.4}$$

where $W_i(x,y)$ is the ith mode shape and x_i is the ith mode coordinate.

Let's consider one particular mode,

$$X_i(x,y,t) = x_i(t) W(x,y) \tag{2.5}$$

We use the Variation method to deduce the single degree of freedom equation for this mode.

The Lagrangian for the system

$$\mathrm{L} = T^* - V = \frac{1}{2} \int_R \rho h \dot{x_i}^2 dx dy - x_i^2(t) V^* \tag{2.6}$$

where V^* is a constant determined by $W(x,y)$.

There are two forces exerted on the plate, the excitation pressure $p(x,y,t)$ and the damping pressure $p_{damping}(x,y,t)$. We assume the damping pressure is proportional

to the plate velocity, so

$$p_{damping}(x,y,t) = C\sum_{i=1}^{n}\dot{x}_i(t)W_i(x,y) \tag{2.7}$$

where C is a constant.

By consideration of the virtual work principle, the generalized force exerted upon the generalized coordinate x is

$$F(t) = \frac{\int_R (p(x,y,t) - p_{damping}(x,y,t))W(x,y)dxdy}{W(x_0,y_0)} \tag{2.8}$$

here (x_0, y_0) is the coordinate where $X_i(x_0, y_0, t) = x_i(t)$.

The variation method gives:

$$\frac{d}{dt}(\frac{\partial \mathrm{L}}{\partial \dot{x}_i}) - \frac{\partial \mathrm{L}}{\partial x_i} = F(t) \tag{2.9}$$

Combining equation (2.6) (2.8) and (2.9) we arrive at the equations in the normal coordinates [11]

$$\ddot{x}_i + 2\zeta_i\omega_i\dot{x}_i + \omega_i^2 x_i = \frac{L_i}{m_i}p_e(t) \tag{2.10}$$

where ζ_i and ω_i are the modal damping and modal frequency of the i-th mode,

$${\omega_i}^2 = \frac{2V^*}{\rho h\int_R W^2(x,y)dxdy} \tag{2.11}$$

$$2\zeta_i = \frac{\int_R CW(x,y)dxdy}{W(x_0,y_0)\rho h\int_R W^2(x,y)dxdy} \tag{2.12}$$

$$m_i = \rho h W(x_0,y_0)\int\int_R W_i(x,y)^2 dxdy \tag{2.13}$$

$$L_i = \int\int_R W_i(x,y)dxdy. \tag{2.14}$$

The mode frequency ω_i and mode shape $W(x,y)$ can be determined, in some cases, analytically, and in general, by experiment or finite element methods. The damping ratio $2\zeta_i$ is usually got from experiment.

We choose the plate deflection at $(x_0,\ y_0)$ to nondimensionalize the mode shape

so that $W(x_0, y_0) = 1$. Usually we can use the center of a symmetric plate as the coordinate (x_0, y_0). But when the central point location happens to be a node, we use the maximum mode shape value at some other location. Therefore, in dimensionless form, we define

$$W(\bar{x}, \bar{y}) = W(\frac{x}{a}, \frac{y}{b}). \tag{2.15}$$

Therefore we can evaluate m_i and L_i to obtain

$$L_i = a^2 Q_1 \tag{2.16}$$

$$m_i = \rho h_p a^2 Q_2 \tag{2.17}$$

where

$$Q_1 = \int_0^1 \int_0^{\frac{b}{a}} W(\bar{x}, \bar{y}) d\bar{x} d\bar{y} \tag{2.18}$$

$$Q_2 = \int_0^1 \int_0^{\frac{b}{a}} W^2(\bar{x}, \bar{y}) d\bar{x} d\bar{y}. \tag{2.19}$$

Consider the case of a simply-supported rectangular plate with the dimension of $a \times b$. The normal modes are identified by two mode numbers m and n:

$$W_{mn} = \sin(m\pi x/a) \sin(n\pi y/b) \tag{2.20}$$

The natural frequencies of the normal modes are

$$\omega_{mn} = \sqrt{\frac{D}{\rho}} \pi^2 \left[(\frac{m}{a})^2 + (\frac{n}{b})^2 \right] \tag{2.21}$$

where

$$D = \frac{E {h_p}^3}{12(1 - \nu^2)}. \tag{2.22}$$

Under base excitation, we have

$$Q_1 = \frac{b/a}{mn\pi^2} [1 - (-1)^m][1 - (-1)^n] \tag{2.23}$$

and

$$Q_2 = \frac{b}{4a} \tag{2.24}$$

For other boundary conditions, the eigen-functions can be obtained using the series solutions in [12]. Alternatively, we can use the finite element method. Using ADINA, we have obtained the corresponding values for several types of boundary conditions aspect ratios.

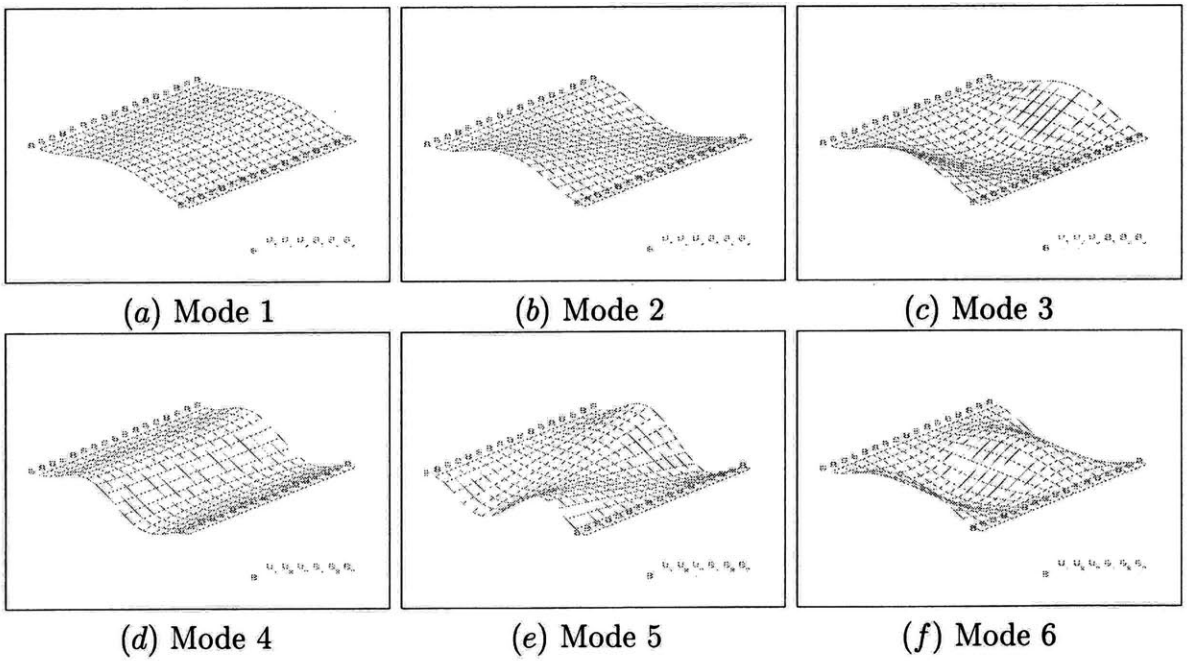
See Table 2.1 and Table 2.2 for two boundary conditions of clamped-free-clamped-free and clamped-clamped-clamped-clamped. The tables also give the natural frequencies of the normal modes in dimensionless form

$$\lambda = \omega a^2 \sqrt{\frac{\rho h_p}{D}}. \tag{2.25}$$

The corresponding mode shapes are shown in Figure 2-2 and Figure 2-3 for two types of boundary conditions.

			mode 1	mode 2	mode 3	mode 4	mode 5	mode 6
$\frac{b}{a} =$ 0.5	λ	FEM	90.0	94.2	110.5	143.6	199.4	253.5
		Leissa	89.0	93.4	110.0	142.0	194.4	245.6
	Q_1		0.283	0.000*	0.016	0.000*	0.005	0.000*
	Q_2		0.232	0.064*	0.107	0.054*	0.102	0.120*
$\frac{b}{a} =$ 0.9	λ	FEM	27.7	31.7	48.7	78.0	83.7	85.6
		Leissa	27.4	31.7	48.8	75.6	81.6	84.4
	Q_1		0.500	0.000*	0.022	0.000*	0.000*	0.000*
	Q_2		0.402	0.119*	0.215	0.255*	0.128*	0.097*
$\frac{b}{a} =$ 1	λ	FEM	22.4	26.4	43.6	63.1	68.7	81.0
		Leissa	22.2	26.4	43.6	61.2	67.2	79.8
	Q_1		0.553	0.000*	0.023	0.000*	0.000*	0.000*
	Q_2		0.443	0.133*	0.244	0.290*	0.144*	0.108*
$\frac{b}{a} =$ 2	λ	FEM	5.55	8.88	15.6	20.8	27.4	31.8
		Leissa	5.51	8.99	15.2	20.5	27.3	29.9
	Q_1		1.073	0.000*	0.000*	0.000*	0.030	0.527
	Q_2		0.832	0.275*	0.673*	0.298*	0.572	1.029
Values with * are normalized by the maximum modal displacement, which is not located at the center of the plate								

Table 2.1: λ, $Q1$, $Q2$ for the first 6 modes of a clamped-free-clamped-free (length a sides clamped) thin plate with different aspect ratios



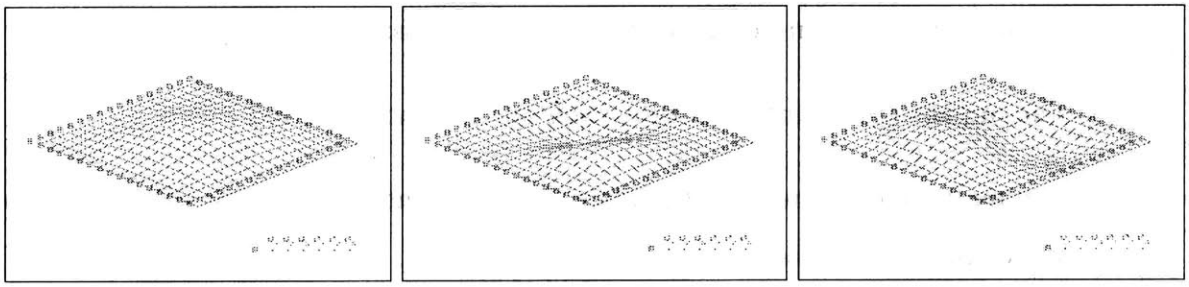
(*a*) Mode 1 (*b*) Mode 2 (*c*) Mode 3

(*d*) Mode 4 (*e*) Mode 5 (*f*) Mode 6

Figure 2-2: First 6 mode shapes of a clamped-free-clamped-free thin plate with the aspect ratio of 0.9. longer sides clamped

			mode 1	mode 2	mode 3	mode 4	mode 5	mode 6
$\frac{b}{a}=0.5$	λ	FEM	99.4	128.4	181.3	258.5	264.1	292.2
		Leissa	97.3	126.5	178.3	252.7	256.1	284.1
	Q_1		0.145	0.000*	0.062	0.000*	0.000*	0.000*
	Q_2		0.084	0.090*	0.097	0.091*	0.098*	0.101*
$\frac{b}{a}=0.9$	λ	FEM	40.7	78.9	89.8	124.5	142.7	169.6
		Leissa	no	no	no	no	no	no
	Q_1		0.254	0.000*	0.000*	0.000*	0.121	0.124
	Q_2		0.145	0.165*	0.167*	0.183*	0.181	0.176
$\frac{b}{a}=1$	λ	FEM	36.4	75.4	75.4	111.2	139.3	140.0
		Leissa	36.0	73.4	73.4	108.2	131.6	132.2
	Q_1		0.282	0.000*	0.000*	0.000*	0.000*	0.136
	Q_2		0.161	0.157*	0.157*	0.203*	0.161*	0.099
Values with * are normalized by the maximum modal displacement, which is not located at the center of the plate								

Table 2.2: λ, $Q1$, $Q2$ for the first 6 modes of a clamped-clamped-clamped-clamped thin plate with different aspect ratios



(*a*) Mode 1 (*b*) Mode 2 (*c*) Mode 3

(*d*) Mode 4 (*e*) Mode 5 (*f*) Mode 6

Figure 2-3: First 6 mode shapes of a clamped-clamped-clamped-clamped thin plate with the aspect ratio of 1

2.3 SDF model of the plate vibration

2.3.1 Formulation of SDF model

The plate response is dominated by the modes which are forced near their resonance frequencies. They are decoupled from each other since we consider the linear theory as adequate description of the small amplitude vibration of thin plates. At low frequency forcing, the frequencies of the normal modes are sufficiently separated that the plate response is dominated by a single mode - the one whose frequency is closest to the forcing frequency.

Generally, the impact causes coupling of energy to the higher plate vibration modes, the energy lose due to the coupling for the main mode can be taken into account by additional energy dissipation other then mode damping when impact happens. The energy that is transfered to high-frequency modes consequently produces high noise level, because sound are more easily generated by high-frequency modes. Though the higher modes account for the noise generation, they have little effect on the impact dynamics because of their small displacement and velocity.

We make several assumptions of the impact process:

- The impact takes much less time then one excitation period.
- The main mode dominates the plate displacement and velocity all the time, or less conservatively, before the impact.
- The general coordinate of the main mode doesn't change during the impact.
- The mode velocity of the main mode changes direction and decrease in amplitude by a factor of e after the impact. The value of e is determined by the energy lose of the main mode because of either mode coupling or acoustics dispersion or energy lost in the striker.

As we can image, those are reasonable assumptions for the low-frequency modes of the plate when the impact is not strong.

We use the subscript s to denote the order of the normal coordinate i whose mode frequency ω_i is nearest to the excitation frequency ω_e. Then the plate motion, which can be represented by a single normal coordinate x_s can be expressed as:

$$\begin{cases} \ddot{x_s} + 2\zeta_s\omega_s\dot{x}_s + \omega_s^2 x_s = F_{es}cos(\omega_e t + \phi) \\ x_s(t^+) = x_s(t^-)\ when\ x_s(t) \longrightarrow h \\ \dot{x}_s(t^+) = -e\dot{x}_s(t^-)\ when\ x_s(t) \longrightarrow h \end{cases} \tag{2.26}$$

where

$$F_{es} = \frac{L_s}{m_s} P_e \tag{2.27}$$

P_e is the amplitude of the sinusoidal pressure exerted on the plate.

2.3.2 Nondimemsionalization of the SDF equation

For the convenience of analysis, we can first nondimemsionalize equation 2.26. Let

$$\omega = \frac{\omega_e}{\omega_s} \tag{2.28}$$

$$\tau = \frac{t}{\omega_s} \tag{2.29}$$

$$x = \frac{x_s}{h} \tag{2.30}$$

$$F = \frac{F_{es}}{{\omega_s}^2 h} \tag{2.31}$$

$$\zeta{=}\zeta_s \tag{2.32}$$

We have

$$\begin{cases} \frac{d^2}{d\tau^2}x + 2\zeta\frac{d}{d\tau}x + x = Fcos(\omega\tau + \phi) \\ x(\tau^+) = x(\tau^-)\ when\ x(\tau) \longrightarrow 1 \\ \frac{d}{d\tau}x(\tau^+) = -e\frac{d}{d\tau}x(\tau^-)\ when\ x(\tau) \longrightarrow 1 \end{cases} \tag{2.33}$$

Note now gap h is no longer an independent parameter, but appears in x and F.

We begin the analytical study with the new equation (2.33), in the later part of the thesis, we will come back to the dimensionalized form when we consider the impact velocity.

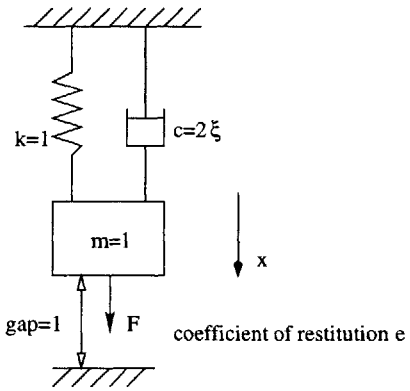


Figure 2-4: Single degree of freedom model

The single-degree of freedom model is shown in Figure 2-4. which is a linear mass-spring-damper system and plus, that's where the nonlinearity comes, a striker, or the bottom obstacle in the figure.

It should be noted that in the later part of the thesis, $\dot{()} = \frac{d}{d\tau}$.

Chapter 3

Analytical Study of the SDF model, the Bifurcation Sets for Periodic Solutions

The single mode impact model is identical to those of a single-degree-of-freedom impact oscillator studied by previous investigators (Nordmark 1991, Thompson, Bishop, and Foale 1994, Peterka 1997). The bifurcation when impact is taking place is called the grazing bifurcation. Some unique features associated with grazing bifurcation have been reported by Nordmark (1991). However, no systematical result for the impact motion with damping has been published yet, and the multiplicity of periodic solutions at the grazing bifurcation has not been analyzed. In the following, we study periodic solutions which bifurcate at the grazing bifurcation point. These solutions have periods which are n-times the period of the forcing. We show that periodic solutions n=1, 2, 3, ... exist at the grazing bifurcation. We use 1/n to denote such orbits. We further analyze the stability of these solutions and the conditions when these solutions no longer exist owing to the additional impact - secondary grazing bifurcations.

In this chapter, we show the multiplicity of periodic orbits by constructing such orbits. Such orbits have one impact during n cycles of the forcing.

3.1 Transformation of the SDF equation into algebric form

In last chapter, we get the nondimensionalized equation of motion (2.33) for the SDF vibration. To solve periodic solution for the motion, we have to first transform the equation into algebra form.

The motion between two impact is described by the linear equation of motion

$$\begin{cases} \ddot{x} + 2\zeta\dot{x} + x = F\cos(\omega\tau - \phi) \\ x(0) = h,\ \dot{x}(0) = -v_0,\ x(T) = 1,\ \dot{x}(T) = v_{im} \\ v_0 = ev_{im} \\ T = \frac{2n\pi}{\omega} \end{cases} \tag{3.1}$$

where h is the gap, e the coefficient of restitution, v_{im} the impact velocity; it is related to the rebound velocity v_0 through the coefficient of restitution e by $v_0 = ev_{im}$. F and ω are the dimensionless forcing amplitude and frequency. $T = 2n\pi/\omega$ is the period of the motion; it can be an integer multiple of the forcing period. In that case, n stands for the number of forcing cycles during which one impact occurs. We will call such orbit 1/n orbit. The displacement and velocity of the linear ordinary differential equation between two impacts are given by

$$\begin{cases} x = A_c\cos\omega\tau + A_s\sin\omega\tau + \mathrm{e}^{-\zeta\tau}(B_c\cos\omega_d\tau + B_s\sin\omega_d\tau) \\ v = \frac{d}{d\tau}x \end{cases} \tag{3.2}$$

where

$$\omega_d = \sqrt{1-\zeta^2};$$

Substituting equation (3.2) into equation (3.1), we get seven equations for A_c, A_s, B_c, B_s, v_0, v_{im} and ϕ.

Solve for the seven equations, we can get a quadratic equation for v_0. Solve for v_0, we have:

$$v_0 = v_0(F, \omega, n) \tag{3.3}$$

Although the above steps for calculating the periodic orbits are very easy to follow, the algebraic expressions are very cumbersome. We carried out all these and later calculations in this chapter using Mathematica.

3.2 General characteristics of the periodic solution

Image there is a parameter plane (F, w) for the periodic motion $1/n$, we can move our parameter (F, w) around and check the onset and end of the $1/n$ motion and also its stability. In this section, we show the general idea of the motion and define some terms.

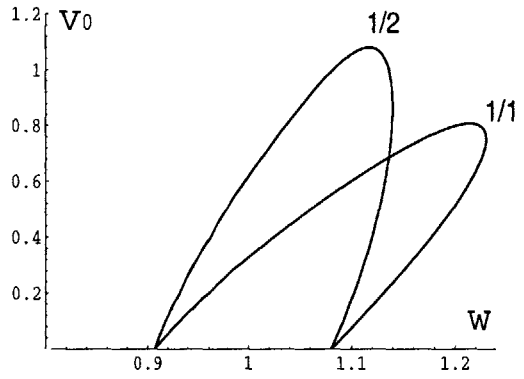
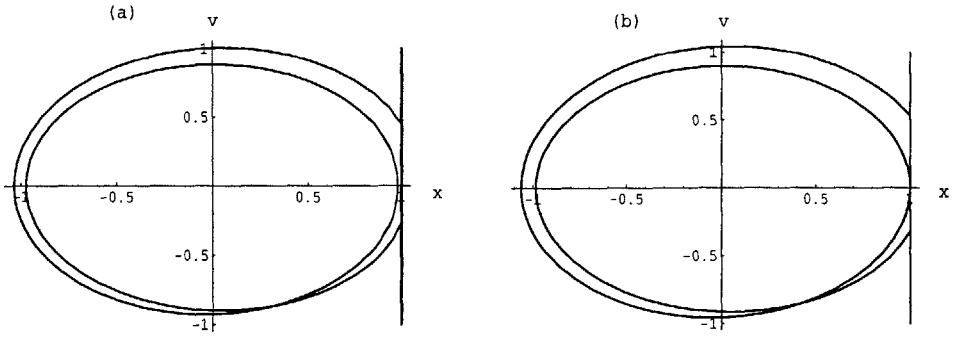


Figure 3-1: Periodic branches of 1/1 and 1/2 solutions

When impact is just about to occur, as will predicted by linear theory, $A_c^2 + A_s^2 = 1$; we have the following relationship between the forcing amplitude and the frequency frequency for the grazing bifurcation

$$F = \sqrt{(1-\omega^2)^2 + (2\zeta\omega)^2}. \tag{3.4}$$

Figure 3-1 shows the bifurcation diagram of the impact velocities as functions of the forcing frequency of 1/1 and 1/2 periodic orbits for forcing amplitude $F = 0.2$. The rest of the parameters are $\zeta = 0.05$, $e = 0.6$. We note that at the grazing bifurcation both the 1/1 and 1/2 orbits come into existence. In fact, it is possible that



(a) before grazing (b) grazing bifurcation

Figure 3-2: Secondary grazing bifurcation of the 1/2 orbit.

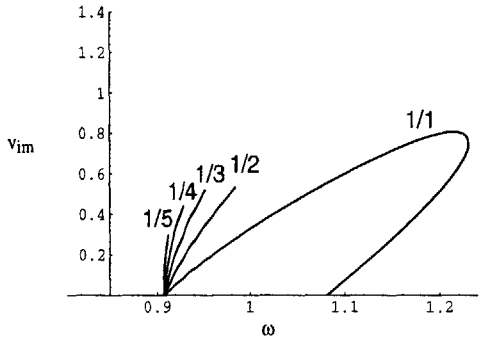


Figure 3-3: Existence of the 1/n orbit considering secondary grazing bifurcation.

periodic orbits 1/n with $n > 2$ also exist. Before presenting these results, we need to examine the possibility of secondary grazing bifurcation which will significantly modify the bifurcation diagram.

As already can be seen in Figure 3-1, the periodic orbits may exist beyond the impact limit predicted by equation (3.4).

The above construction of orbits assumes that no impact takes place during n-cycles of forcing. Careful examination of the periodic orbits reveals that for $n > 1$ secondary grazing bifurcations terminate these orbits and lead to new types of periodic orbits such as 2/n orbits. Figure 3-2 (a-b) shows the 1/2 periodic orbits in the phase plane (x, $\dot{x}$). For parameters corresponding to those in Figure 3-2(b), we find that at $\omega = 0.985$, the 1/2 orbit comes to a grazing bifurcation thus 1/2 orbit no longer exists for $\omega > 0.985$.

By examining the clearance Max$\{1 - x(t)\}$ between the plate and the stopper during one period, $0 < t < 2n\pi/\omega$, we can identify the secondary grazing bifurcations. At the secondary bifurcations, the original 1/n periodic orbit ceases to exist. Taking into consideration of these secondary grazing bifurcations, we obtain the bifurcation diagrams in Figure 3-3. Only orbits with $n <= 5$ are plotted. As n increase, the parameter intervals ω corresponding to the existence of $1/n$ orbits diminish.

To summarize, we find that at the grazing bifurcation, periodic orbits $1/n$ for arbitrary n come into existence. Except 1/1 orbit, all of them terminate at secondary bifurcations.

The periodic orbit shown in figure 3-3 may be unstable, we can use perturbation method to determine the stability of the orbit. As will be shown in the next section, only 1/1 orbit in figure 3-3 is stable.

3.3 Construction of Bifurcation sets

As pointed out before, forcing amplitude F and forcing frequency ω determine if a certain periodic $1/n$ orbit exists. We can plot those combination of F and ω at which $1/n$ motion exists in the F-ω plane. We call that bifurcation sets.

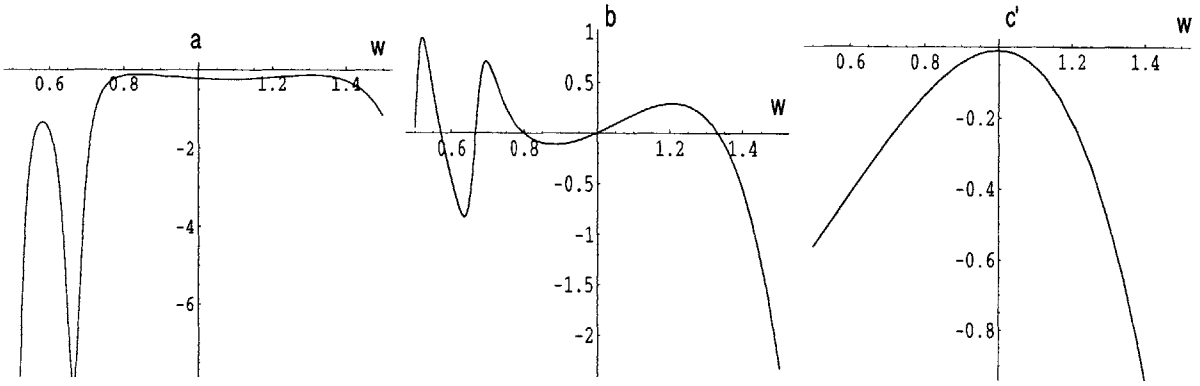
Using Mathematica, we combine the analyses in the last chapter to obtain bifurcation sets. The bifurcation sets give us the partition of the parameter space (F, ω) into regions of same qualitative dynamics. For the later work, we only consider the fixed parameters to be $\zeta = 0.05$, $e = 0.6$.

We first intend to construct the region of existence for different $1/n$ periodic motion when ω is from 0.6 to 1.5 and F is from 0 to 0.6. Secondly we further restrict the region of existence by the stability curve(s) so as to get a region of stability for $1/n$ motion. As is found out in our research, the criteria for existence or stability is more convenient to be expressed in terms of the initial velocity v_0, rather than F. So the basic approach for us to build the regions is to first set ω as a certain value and using existence or stability criteria to find out the v_0 division for which those criteria are satisfied, and then use the relation of v_0 with F, ω to determine the F division for

which those criteria are satisfied. Varying ω from 0.6 to 1.5, we can build the whole regions very well.

3.3.1 Construction of the region of existence

Positive v_0 criterion



(a) a(ω,2) (b) b(ω,2) (c) c' (ω)

Figure 3-4: a,b,c' for n=2.

The existence criterion 1 requires v_0 to be a real positive number. As pointed out before, the v_0 satisfy a quadratic equation

$$a{v_0}^2 + bv_0 + c = 0 \tag{3.5}$$

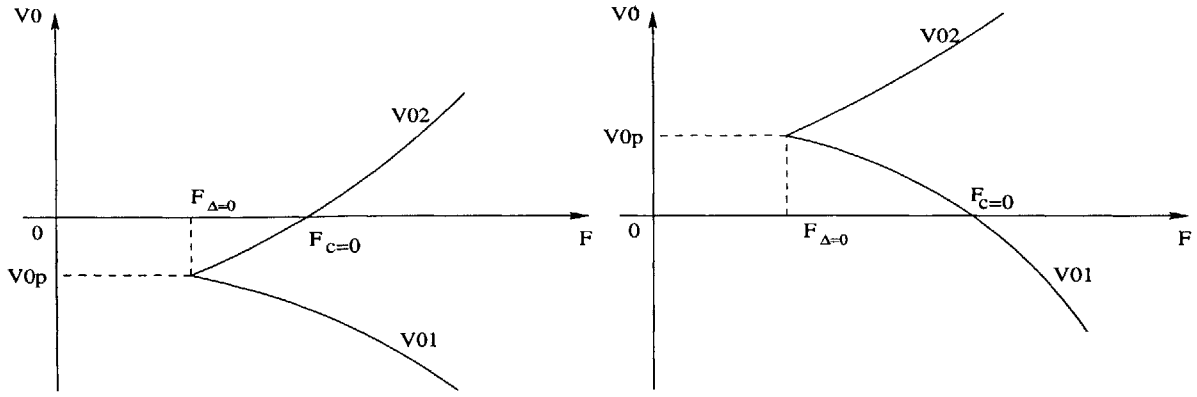
where, as it is found out by our numerical study and as shown in figure 3-4 for a special case $n = 2$.

$$a = a(\omega, n) < 0 \tag{3.6}$$

$$b = b(\omega, n) \tag{3.7}$$

$$c = F^2 + c'(\omega),\ c'(\omega) < 0 \tag{3.8}$$

The problem boils down to that what's the (F, ω) can results in a positive v_0. We plotted the functions of $a(\omega, n)$, $b(\omega, n)$, $c'(\omega)$ with n being set to be $1, 2, 3...$ and find



(a) $b < 0$ (b) $b > 0$

Figure 3-5: Dependence of v_{01} and v_{02} on F.

out that a is always negative, b can be positive or negative, c' is always negative. The solution for equation (3.5) is

$$v_{01} = (-b - \Delta)/2a \tag{3.9}$$

$$v_{02} = (-b + \Delta)/2a \tag{3.10}$$

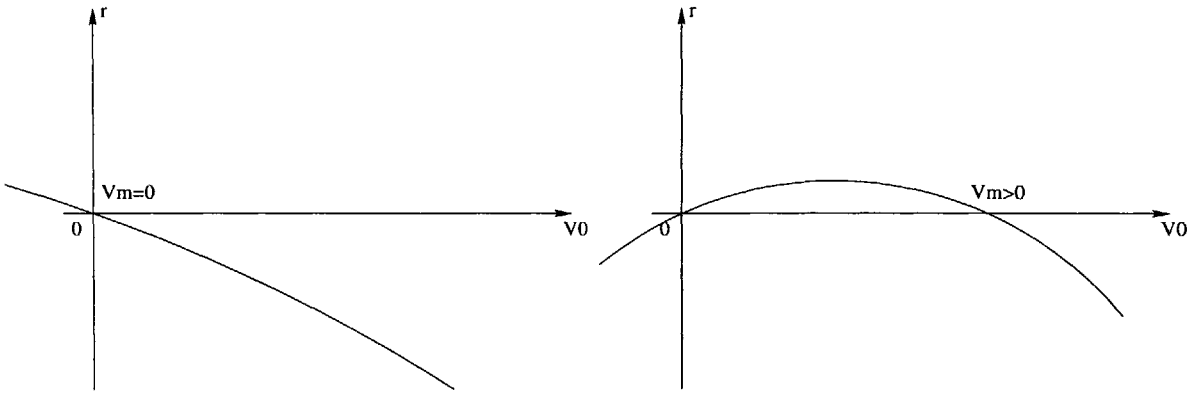
where $\Delta = b^2 - 4ac$. For Δ to be positive, that is to say $c > \frac{b^2}{4a}$ We should have

$$F > F_{\Delta=0} = F_{\Delta=0}(\omega, n) = \sqrt{\frac{b^2}{4a} - c'} \tag{3.11}$$

The dependence of v_{01} and v_{02} on F is shown in Figure 3.3.1. Note with the increasing of F, v_{01} always decrease while v_{02} always increase. We have existence of the $1/n$ motion only if we have at least one of v_{01} or v_{02} positive. Depending on if b is positive or negative, we have different lowest limit for F to allow a positive v_0. In the Figure 3.3.1.

$$F_{c=0} = F_{c=0}(\omega, n) = \sqrt{-c'} \tag{3.12}$$

$$v_{0p} = -\frac{b}{2a} \tag{3.13}$$



(*a*) first case, v_m zero (*b*) second case, v_m positive

Figure 3-6: Two cases of the dependence of r on v_0.

So all in all, in terms of F, the criterion of positive v_0 can be expressed as,

$$F > \begin{cases} F_{\Delta=0}, & \text{when } b > 0 \\ F_{c=0}, & \text{when } b < 0 \end{cases} \tag{3.14}$$

Note when $b > 0$, orbits corresponding to v_{01} and v_{02} coexist for $F_{\Delta=0} < F < F_{c=0}$, because in such region of F, both solutions are positive.

No in-between impact criterion

The existence criterion 2 requires no "unwanted" additional impact between the two impacts of 1/n periodic motion. Saying it more clearly, the existence criterion 2 prevent the displacement of 1/n motion between two impacts from going beyond the clearance. There is no analytical way to give a expression of maximum displacement between two impacts, so a numerical checking of the maximum displacement is implemented and some properties of the clearance criterion are revealed. First Define

$$r = x_{max} - 1 \tag{3.15}$$

where x_{max} is the maximum value of x between two impacts.

There are two possibility for the dependence of r over v_0, as found out by numerical way and shown in Figure 3.3.1.

The no in-between impact criterion is satisfied when

$$0 < v_0 < v_{0m} = v_{0m}(\omega, n) = \begin{cases} a\ possitive\ value & \text{in the first case} \\ 0 & \text{in the second case} \end{cases} \tag{3.16}$$

Still ahead we have to know in what region of F can the no in-between impact criterion be satisfied.

If $b < 0$, as shown in Figure 3.3.1(a), only v_{02} can be positive. For

$$v_{02} < v_{0m} \tag{3.17}$$

we have

$$F < F_m = F_m(\omega, n) = \sqrt{\frac{b^2 - (2av_{0m} + b)^2}{4a} - c'} \tag{3.18}$$

If $b > 0$ and $v_{0p} < v_{0m}$, refer to Figure 3.3.1(b), the branch of positive v_{01} is always smaller then v_{0m}. And when $F < F_m$, the branch of v_{02} is also smaller then v_{0m}. So the no in-between impact condition can be expressed as:

$$F_{\Delta=0} < F < F_h = Maximum\{F_m, F_{c=0}\} \tag{3.19}$$

Note in this case, we always have some region in which two possible solution orbits coexist.

If $b > 0$and $v_{0p} > v_{0m}$, the v_{02} solution never satisfies the clearance condition, and for the solution orbit corresponding to v_{01}, it satisfies the clearance criterion when

$$F_m < F < F_{c=0} \tag{3.20}$$

where $F > F_m$ is the solution to the equation

$$v_{01} > v_{0m} \tag{3.21}$$

We use the same notation F_m for both the solution to equation (3.17) and (3.21)

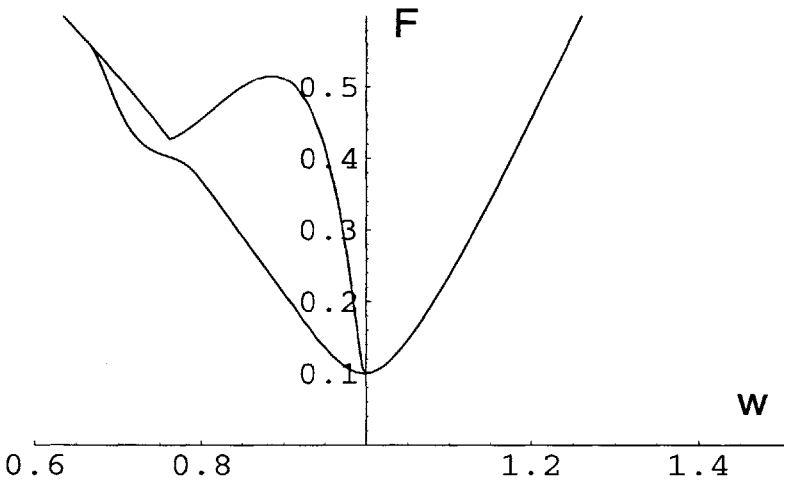


Figure 3-7: The region of existence for 1/2 motion.

because the expressions are the same.

Region of existence

Now we can know, taking both existence criteria into consideration, the begin point and end point of F for the motion to exist, i.e.

$$\begin{cases} F_{c=0} < F < F_m, & \text{when } b < 0 \\ F_{\Delta=0} < F < F_h, & \text{when } b > 0,\ v_{0m} > v_{0p} \\ F_m < F < F_{c=0}, & \text{when } b > 0,\ v_{0m} < v_{0p} \end{cases} \tag{3.22}$$

Note depending on the value of parameters, we some times only have the upper bound and the lower bound of F the same as $F_{c=0}$, corresponding to one of v_{01} and v_{01} is zero. In such case under no F value can the solution exist. Varying ω from 0.6 to 1.5, we get the region of existence. Figure 3-7 shows the region for 1/2 periodic motion. Note we don't have any region of existence for 1/2 when $\omega > 1$.

3.3.2 Construction of the region of stability

The periodic orbit found above may be unstable. To determine the stability, we subject our solution to small perturbation of the initial time delay and velocity, i.e.

$$x(\Delta t_0) = 1,\ \dot{x}(\Delta t_0) = -(v_0 + \Delta v_0) \tag{3.23}$$

These perturbations will result in variations of the constant B_c and B_s which can be found by the above two equations. These perturbations lead to the time delay at the impact and the subsequent rebound velocity, i.e.

$$x(T + \Delta t_1) = h,\ \dot{x}(T + \Delta t_1) = \frac{v_0 + \Delta v_1}{e} \tag{3.24}$$

We substitute the variation of the constants B_c and B_s into the above two equations to obtain Δt_1 and Δv_1, which are written in the following form

$$\begin{pmatrix} \Delta t_1 \\ \Delta v_1 \end{pmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{pmatrix} \Delta t_0 \\ \Delta v_0 \end{pmatrix} \tag{3.25}$$

Whether the eigenvalues of the above matrix lie within the unit circle determines the stability of the periodic orbits. Define

$$tr = m_{11} + m_{22},\ det = m_{11}m_{22} - m_{12}m_{21}$$

The stability criterion can be expressed as

$$(1 - tr + det)(1 + tr + det) > 0 \tag{3.26}$$

Anyway, Mathematica handles the calculation well. Calculation shows that $1/n$ motion comes into stability when

$$v_0 > v_{0s} \tag{3.27}$$

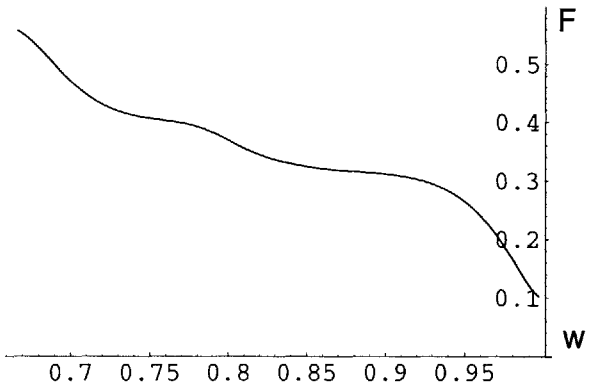


Figure 3-8: F_s n=2

where

$$v_{0s} = v_{0s}(\omega, n) > 0$$

Beyond v_{0s} the motion is stable and below it motion is unstable. As found in calculation, whenever v_{02} begins with a positive value v_p, i.e. when $b > 0$, this value v_p is exactly v_{0s}, which is still to be interpreted by theory. This means in such case solution corresponding to v_{01} is always unstable, because v_{01} is smaller than v_p. Recall positive v_{01} doesn't exist when $b < 0$, we can know v_{01} will never account for any stable $1/n$ solution, i.e., solution of v_{01} never come into existence in real numerical simulation or experiment. When $b < 0$, only v_{02} solution comes into existence, and when $v_{02} > v_{0s}$, motion becomes stable, solve for F we get

$$F > F_s \tag{3.28}$$

where

$$F_s = F_s(\omega, n) = \sqrt{\frac{b^2 - (2av_{0s} + b)^2}{4a}} - c'$$

We have a stability curve for the $1/n$ motion beyond which it, if existing, is stable, and below which it, if existing, is unstable.

To construct the region of existence and stability, we need two steps.

1. Get rid of the region of existence in which only the orbit solution corresponding to v_{01} exist, or rather to say, reconstruct the region of existence only corre-

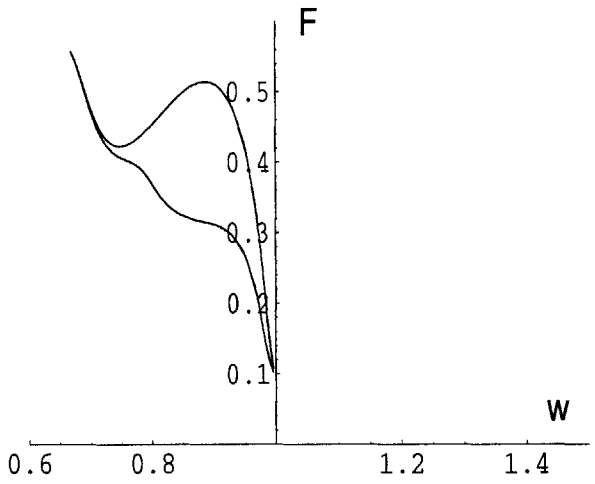


Figure 3-9: The region of stability for 1/2 motion

sponding to the solution of v_{02} by following a similar way carried out in the implementation of existence criterion 1 and 2.

2. Plot the part of region of existence got from step 1 which is above the stability curve.

   The region of stability is shown for 1/2 motion in Figure 3-9.Note the upper bound is a little different from the upper bound shown in Figure 3-7, that's because now we only consider the region corresponding to v_{02}.

3.3.3 Derived bifurcation sets and some discussion

Higher $1/n$ periodic motion will ether have smaller region of existence and stability, or a smaller domain of attraction compared with the first several region of stability and existence of $1/n$ periodic motions, in the same way they has little practical meaning. However, any $1/n$ region of existence or stability can be acquired by similar approach.

Note the region of 1/n motion built by the above criteria doesn't mean that there can't be other motion exists inside the region got, but it does imply no 1/n motion can exist or is stable outside of the region got. Actually, those region of existence or stability for different 1/n motion can overlap each other, as can be seen in the figures.

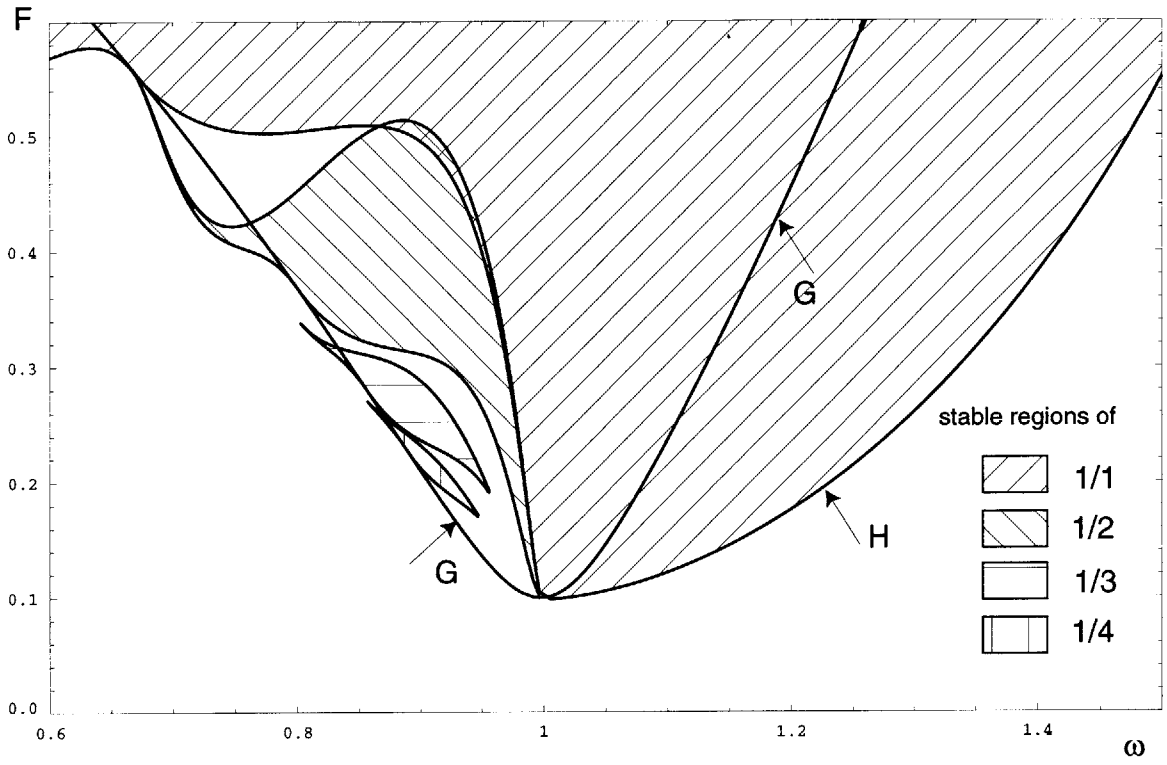


Figure 3-10: Bifurcation sets for n=1,2,3,4

The $1/n$ periodic motion solution is already complex, so we can image those 2/n, 3/n motions will be too complex to solve by such criterion approach, in practical it's impossible to get any similar region of existence or stability for those motions. A simulation program is written to deal with the problem, and much rich content of the SDF rattle vibration are revealed by that program, whose results will be discussed in next chapter. However, it should be noted that the numerical simulation cannot give us any information of unstable motion and also can not pick up those motions with a very small domain of attraction. So analytical study in this chapter is very important.

Chapter 4

Numerical Integration of the SDF equation, Bifurcation diagrams and Impact Velocity

As we see, bifurcation sets tell us basic behavior of the impact vibration, such as the stability of the $1/n$ motions. But the limitation of computation power prevents us from getting more details of the vibration pattern beyond the simple $1/n$ motions. Aperiodic and even chaotic solutions are also possible which cannot be predicted by the analytical method. To know more about the dynamics of the SDF impact vibration, a Java program is coded, to help us integrate the motion and provide convenient user interface.

4.1 Integration methods

When the mass is not touching the striker, Runga-Kutta method are used to integrate the equation; when impact happens, special algorithm are used to make sure that the impact is modeled accurate enough.

4.1.1 Runga-Kutta method

To integrate the second order equation

$$\begin{cases} \frac{d^2}{dt^2}x = F(t, x, \frac{d}{t}x) \\ x(t_0) = x_0, x^{'}(t_0) = x_0^{'} \end{cases} \tag{4.1}$$

introduce $v = \frac{d}{dt}x$, and $\frac{d}{dt}v = \frac{d^2}{dt^2}x$. it becomes:

$$\begin{cases} \frac{d}{dt}x = v \\ \frac{d}{dt}v = F(t, x, v) \\ x(t_0) = x_0, v(t_0) = v_0 \end{cases} \tag{4.2}$$

The Runga-Kutta method is the following:

$$x_{n+1} = x_n + hv_n + \frac{1}{6}(l_1 + l_2 + l_3) \tag{4.3}$$

$$v_{n+1} = v_n + \frac{1}{6}(l_1 + 2l_2 + 2l_3 + l_4)$$

$$l_1 = hF(t_n, x_n, v_n)$$

$$l_2 = hF(t_n + \frac{h}{2}, x_n + \frac{hv_n}{2}, v_n)$$

$$l_3 = hF(t_n + \frac{h}{2}, x_n + \frac{hv_n}{2} + \frac{hl_1}{4}, v_n + \frac{l_2}{2})$$

$$l_4 = hF(t_n + h, x_n + hz_n + + \frac{hl_2}{2}, v_n + l_3)$$

$$x(t_{n+1}) \approx x_{n+1},\ (n = 0,\ 1,\ 2,\ ...)$$

$$error\ R = O(h^5)$$

In the simulation code, the time step h is chosen as $\frac{1}{200}$ of the period of the excitation.

4.1.2 impact detection and disposal

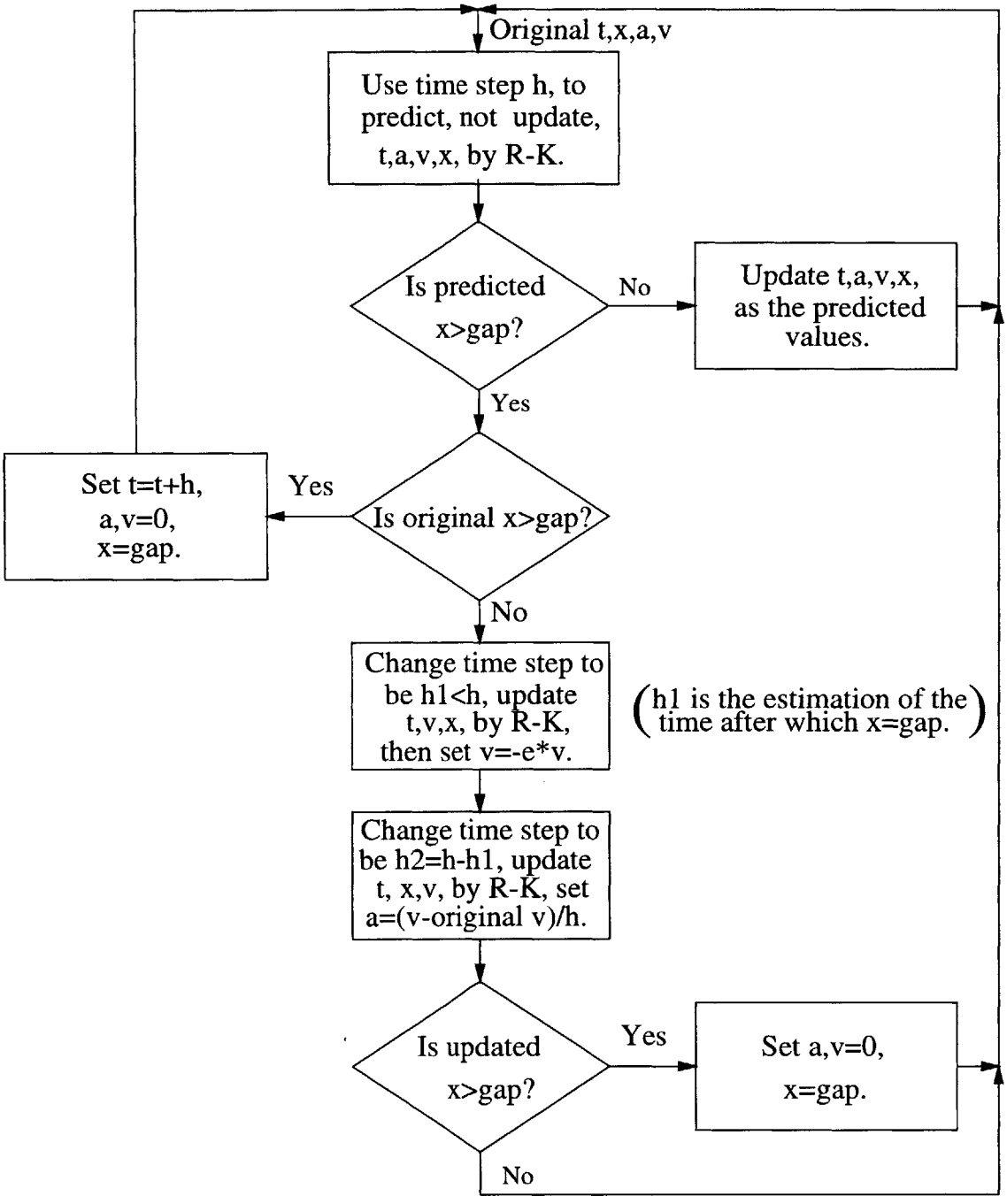


Figure 4-1: The algorithm for the impact detection and integration.

The impact is an instant event mathematically, however, in the numerical simulation, we have a finite integration time step. For that reason, we have to set up practical goal on how much detail of the motion we want to capture. The SDF impact process may be very complicated in some cases. For example, when F is too big, mathematically the mass may bump for infinite times, but in a finite time, before it finally stuck with the striker for a short time. For such practical consideration, we don't intend to follow every impacts, but just keep a time step of $\frac{forcing\ period}{200}$.

In each integration, we first use Runga-Kutta to predicted the a, v, x for the next time step. If both the predicted and current x pass the striker, that is $x > gap$(gap=1 in the nondimensionalized SDF model), we set $a, v = 0$, $x = gap$. If only the predicted $x > gap$, we change the time step h to be a smaller one $h1$, which is the linear prediction of how long it takes the mass to reach the striker estimated by the current x and v. Velocity is timed by $-e$ and then the rest of the time step $h2 = h - h1$ is used for the Runga-Kutta to update the motion parameters. The Algorithm is shown in the flow map in Figure 4-1.

4.1.3 Java program for the SDF simulation

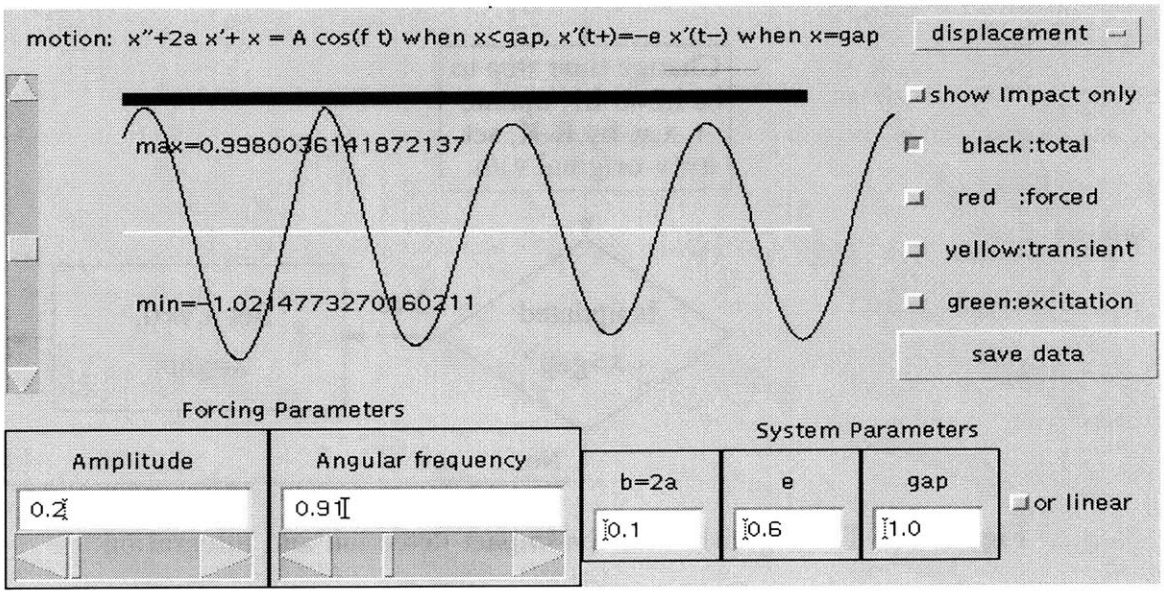


Figure 4-2: The program interface for simulation study

There are many parameters in the SDF impact system that can be changed during simulation. And due to the nonlinear property, solution depends on initial condition sensitively. To capture the full dynamics of the motion, some program with good interface to manipulate the system parameters, and also instant showing of the motion are desired. Since Java has whole bunch of interface classes, it is chosen as the programming language.

The Java program has the function of showing the displacement(velocity, acceleration) vs. time, phase diagram, and impact velocity vs. changing system parameter. Figure 4-2 shows the interface of that program.

The Java program are divided into 5 portions with different function.

Impact_sys.java The dynamical part, handling the running of the impact system.

V_impFrame.java The part handling the calculation of impact velocity depending on different changing parameters, also it provide a window for data input.

Screen.java The plotting part, handling the showing of the simulation results.

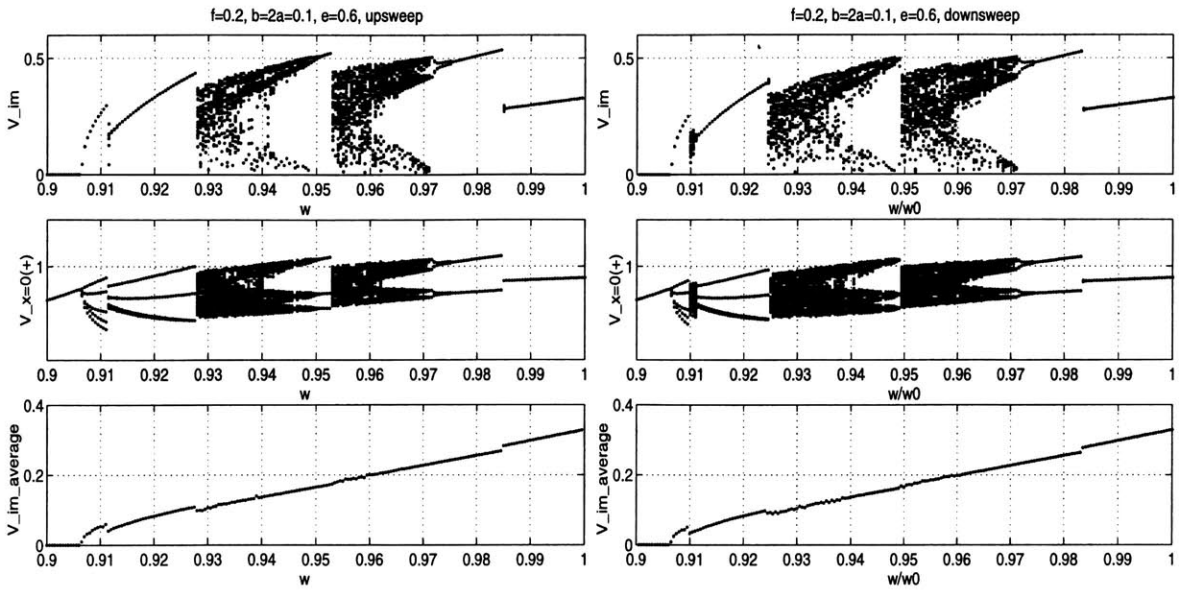
SaveWindow.java The data output part, used to save data in text file.

iap.java The interface part, linking other parts together, making them an integrated applet.

See appendix A for the program code for each portions.

4.2 Vibration pattern revealed by simulation

Figure 4-3 shows the bifurcation diagram of the impact velocity versus forcing frequency. However new solutions including chaotic solutions at the secondary grazing bifurcations are missing in that diagram. We use 4th order Runge-Kutta method to simulate the dynamics of the one-mode model. The bifurcation diagrams are given in terms of the impact velocity v_{im}, $v_{x=0}(+)$, which is the positive velocity at which $x(t)$ crosses $x = 0$, and the average impact velocity ${v_{im}}^{average}$. These bifurcation



(*a*) w up sweep (*b*) w down sweep

Figure 4-3: SDF mode simulation for the impact velocity with changing excitation frequency

diagrams are obtained by numerically integrating the equation for 100 forcing cycles for fixed parameter values. The first 70 forcing cycles are discarded. Therefore, if the motion is periodic of type m/n, we have m impact velocities and n values of $v^+_{x=0}$. When simulation is complete for that forcing frequency, the position and velocity is used as initial conditions for simulations at another forcing frequency which is slightly changed from the previous frequency. This simulation procedure is analogous to slow frequency-sweep experiment. For nonlinear systems, we expect differences in system dynamics between an up-sweep and a down-sweep experiments.

We present in Figure 4-3(a) the simulation result of a typical up-sweep simulation for $F = 0.2$. Figure 4-3(a) plot respectively the impact velocity, the positive velocity crossing the line $x = 0$, and the average impact velocity per forcing cycle. Periodic orbits of 1/n type, i.e. one impact in n-cycles, have one impact velocity and n positive velocities crossing $x = 0$. As frequency is slowly increased, the non-impact solution ceases to exist at the grazing bifurcation point $\omega = 0.907$ as predicted by linear theory. Immediately after the grazing bifurcation, 1/5 periodic orbit is born and remain stable. It terminates at a secondary grazing bifurcation at $\omega = 0.913$.

Afterwards, the 1/4 orbit becomes stable. It remains stable until a secondary grazing bifurcation at $\omega = 0.928$. Chaotic dynamics follows this bifurcation. An inverse period doubling bifurcation leads to stable 1/3 orbit at $\omega = 0.950$. The 1/3 orbits terminates at $\omega = 0.953$ through a secondary grazing bifurcation. Chaos again occurs. Another inverse period-doubling bifurcation leads to stable 1/2 orbits at $\omega = 0.973$. It terminates at $\omega = 0.985$ through the secondary grazing bifurcation. The stable 1/1 orbit is now observed. It remains stable until $\omega = 1.230$ at which it merges with the unstable 1/1 orbit through a saddle-node bifurcation (Figure 3-3) and disappears.

In a frequency down-sweep simulation, Figure 4-3(b), the major difference is the absence of impact for $\omega > 1.081$ as predicted by linear theory. The rest of the bifurcation sequences are similar to the up-sweep simulation.

The grazing bifurcation is different from ordinary bifurcations (saddle-node, pitchfork, Hopf) in the following aspects:

- The original periodic orbit no longer exists after the bifurcation; hence it is meaningless to discuss the stability of the original periodic orbit after the bifurcation.
- There can be many periodic orbits (possibly infinitely many) bifurcating simultaneously at the grazing point.
- The new periodic solutions may cease to exist at secondary grazing bifurcations.

4.3 Impact velocity

When the small gap between a panel and a striker makes the impact event unavoidable, the impact velocity in very important in determining the radiated sound. In the following, we study the dependence of the impact velocity on the gap and on the coefficient of restitution.

4.3.1 Expression of the actual parameters by non-dimensional ones

In section 2.3, we nondimensionalized the equation of motion. however, in order to analyze the actually impact velocity, we have to go through the opposite way, to find out the expression of the actually physical parameters by the nondimensionalized values. The clue lies in (2.28), deriving from (2.28), we have

$$v_{sim} = \omega_s h v_{im}$$

where v_{sim} is the actual impact velocity and v_{im} is the nondimensionalized velocity in simulation. so

$$v_{im} = \frac{v_{sim}}{\omega_s h} \tag{4.4}$$

Gap h between the plate and the striker is an important factor to the impact dynamics. It's desired to know when we keep the dimensional excitation amplitude F_{es} constant and change the h, how is the impact velocity going to change. We don't have explicit appearance of h in our SDF equation of motion, but it was taken into account in both F and x. Next we do some mathematical operation to reveal the dependence of v_{sim} on h.

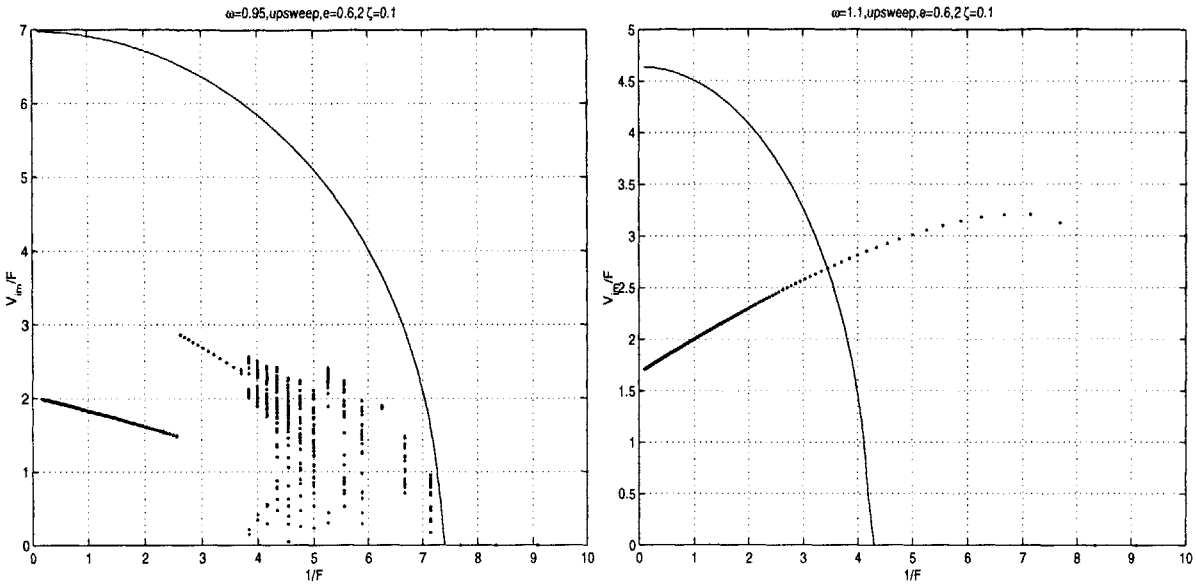
Recall we have $F = \frac{F_{es}}{{\omega_s}^2 h}$. so:

$$\begin{cases} \frac{1}{F} = \frac{{\omega_s}^2 h}{F_{es}} \\ \frac{v_{im}}{F} = \frac{\omega_s v_{sim}}{F_{es}} \end{cases} \tag{4.5}$$

Because ω_s, F_{es} are known constants and F, v_{im} are known parameters in simulation, Equation (4.5) tells us the dependence of v_{sim} upon h.

4.3.2 Dependence of the impact velocity on physical parameters

On gap



(*a*) w=0.95 (*b*) w=1.1

Figure 4-4: SDF mode simulation for the impact velocity with up-sweeping gap

To study the impact velocity with varying gap at fixed dimensional forcing amplitude F_{es}, we plot V_{im}/F as functions of $1/F$ at two different forcing frequencies. As equation (4.5) suggests, those two values are proportional to the actual impact velocity and gap. Figures 4-4(a) corresponds to the impact velocity and the average impact velocity when the forcing frequency is below the resonance frequency. Note that the average impact velocity increases as the gap decreases. Figure 4-4(b) shows the corresponding impact velocity and average impact velocity when the forcing frequency is above the resonance frequency. Note that the impact velocity is larger for larger gaps.

Without nonlinearly model, we can't get the good estimation of the impact velocity. We can compare our calculation of impact velocity with what got from linear estimation.

First the linear theory for the impact velocity: one of them estimate the impact velocity to be the velocity when the mass is passing through the imaginary striker, without actually hitting it.

The equation of motion is

$$\ddot{x} + 2\zeta\dot{x} + x = Fcos(\omega t) \tag{4.6}$$

$$x_{max} = \frac{F}{\sqrt{(1-w^2)^2 + (2\zeta\omega)^2}} \tag{4.7}$$

$$x = x_{max}cos(\omega t) \tag{4.8}$$

$$v = \frac{d}{dt}x = -\omega x_{max}sin(\omega t) \tag{4.9}$$

$$v_{imp} = v_{x=1} \tag{4.10}$$

Derive from equation array (4.7), we get

$$\frac{v_{imp}}{F} = -\omega\sqrt{\frac{1}{(1-w^2)^2 + (2\zeta\omega)^2} - \frac{1}{F^2}} \tag{4.11}$$

when

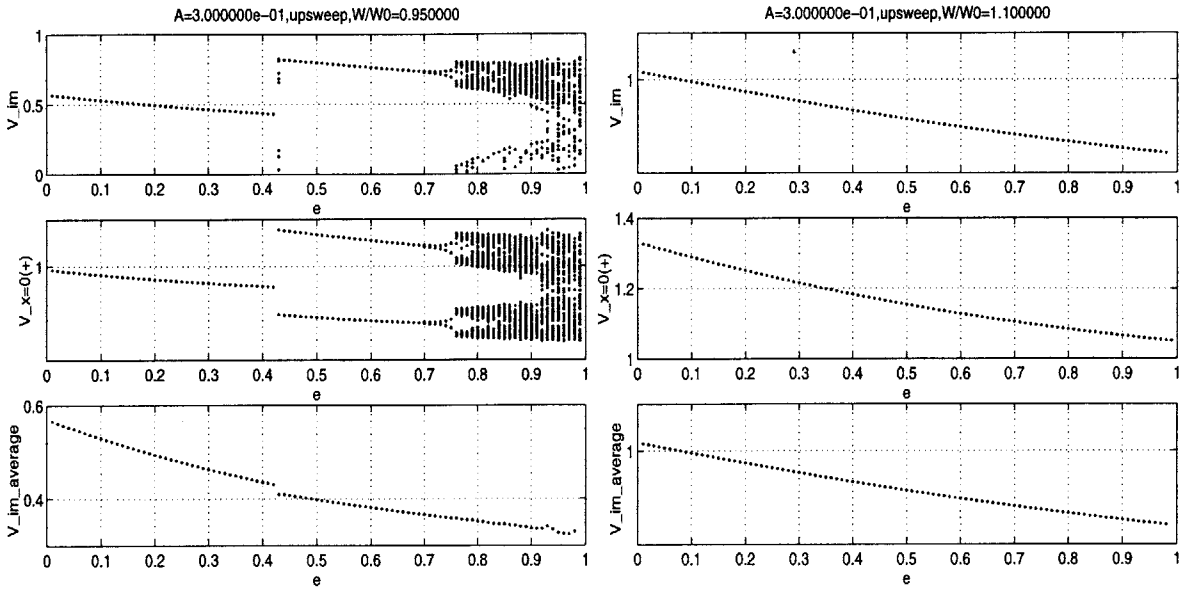
$$\frac{1}{F} < \sqrt{\frac{1+\omega^2}{(1-w^2)^2 + (2\zeta\omega)^2}}$$

and when $\frac{1}{F}$ doesn't satisfy the above situation, there is no impact anticipated by linear theory.

The so called linear prediction of the impact velocity is also shown in figure 4-4(a) and 4-4(b) by continuous curve, we can see the huge difference between the linear and nonlinear prediction, moreover, the restitutional ratio e can't enter the consideration of linear estimation.

On e

The impact velocity depends on the coefficient of restitution e in a surprising way. At fixed forcing amplitudes and frequencies, the impact velocities are plotted against e. When forced below the resonance frequency, Figure 4-5(a), the average impact velocity decreases as e increases. However, because of the period doubling bifurcation,



(a) w=0.95 (b) w=1.1

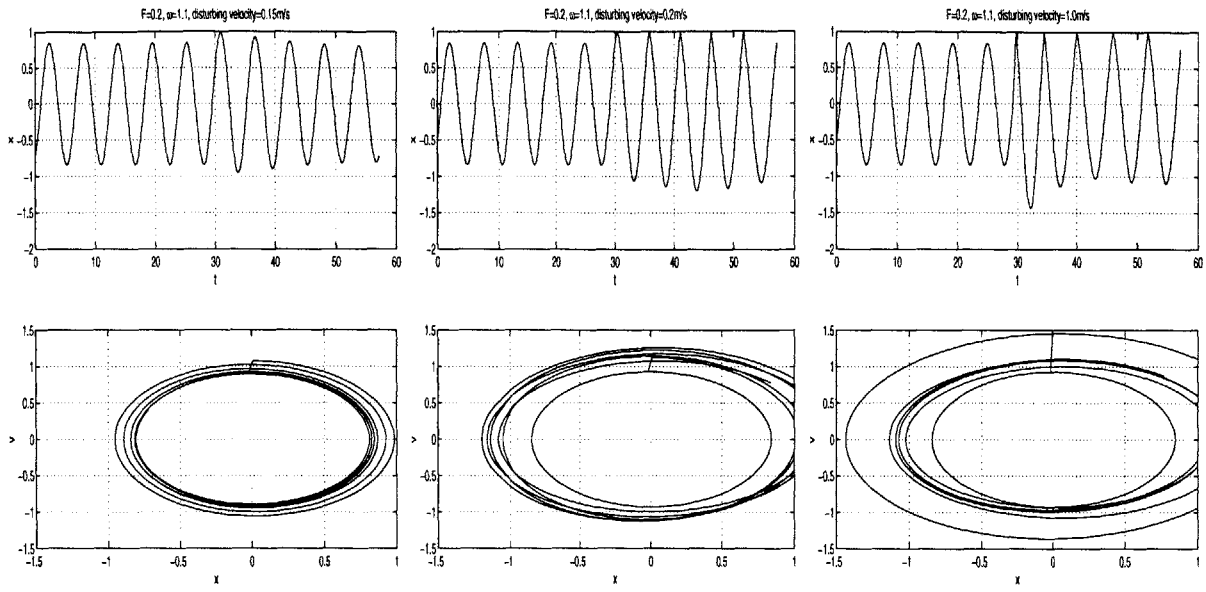
Figure 4-5: SDF mode simulation for the impact velocity with up-sweeping coefficient of restitution e

the peak impact velocity experiences a large jump at $e = 0.42$. When forced above the resonance frequency, Figure 4-5(b), the qualitative dynamics does not change as e changes. However, the impact velocity decreases monotonically as e increases.

4.3.3 Effect of disturbance on the plate vibration

We have a large hysteresis region when the forcing frequency is above the main mode frequency, in which we may have both the non-impact and the strong-impact plate responses depending on the initial condition of the plate. If we have initially a non-impact response for the plate, we can image that with some disturbance, the plate can be pushed into the strong-impact response. In figure 4-6, we model the disturbance as some velocity increase when the mass is passing through the zero displacement with the positive velocity. As we can see, if the disturbance velocity is big enough, as shown in figure (4-6)(b)(c), it can set the plate into strong-impact motion.

Suppose initially the SDF system is in a non-impact response under some particular F and ω. If we give a disturbance velocity to the system, whether the system will settle into strong-impact response depends on F, ω and $V_{disturb}$. We got three demar-



(a) (b) (c)

Figure 4-6: The plate response with disturbances of different values.

cation curves for the system, which are shown in figure 4-7. Each curve corresponds to a constant disturbance velocity, above it the system will settle into strong-impact motion due to the disturbance, below it the system will remain to have non-impact motion.

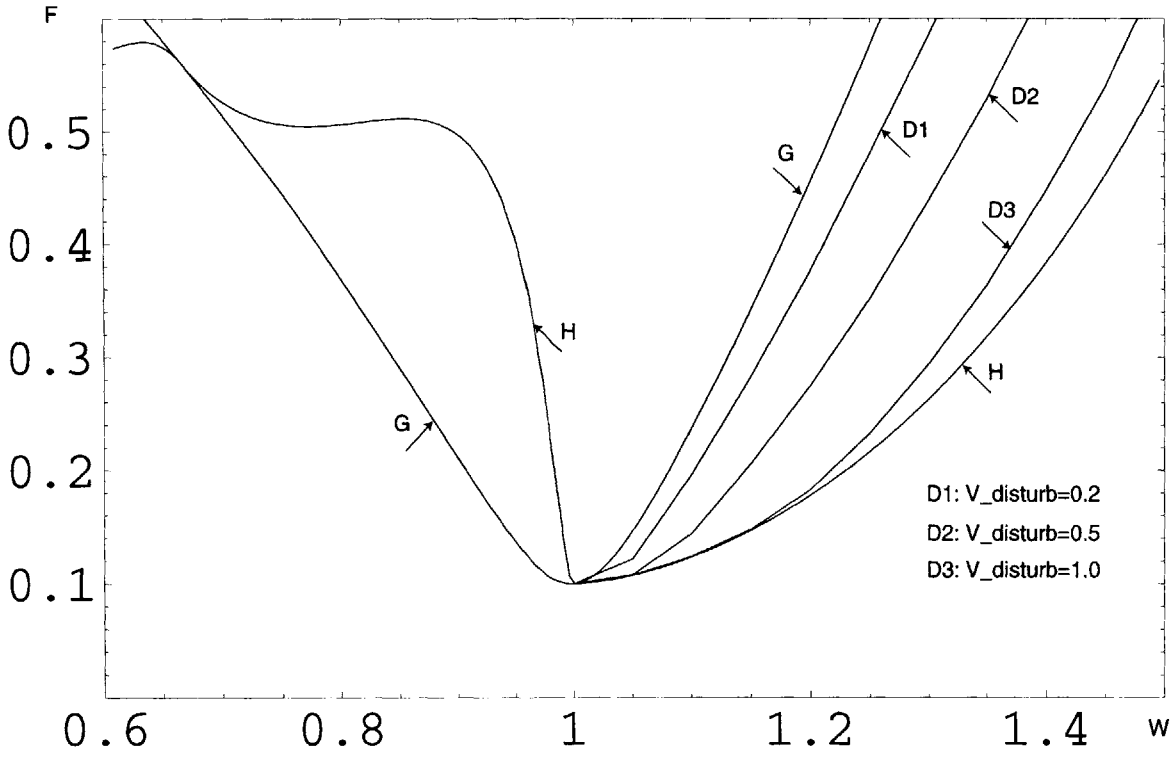


Figure 4-7: Effect of various amplitude disturbance on the plate response.

Chapter 5

Finite Element Simulation of the Plate Vibration

We carry out FEM simulation of the plate impact vibration not only as a verification to SDF model, moreover, it can reveal the higher modes of the plate excited by the impact. The FEM results show that our SDF model captures the main characteristics of the plate impact vibration. All the FEM simulations were done in ADINA.

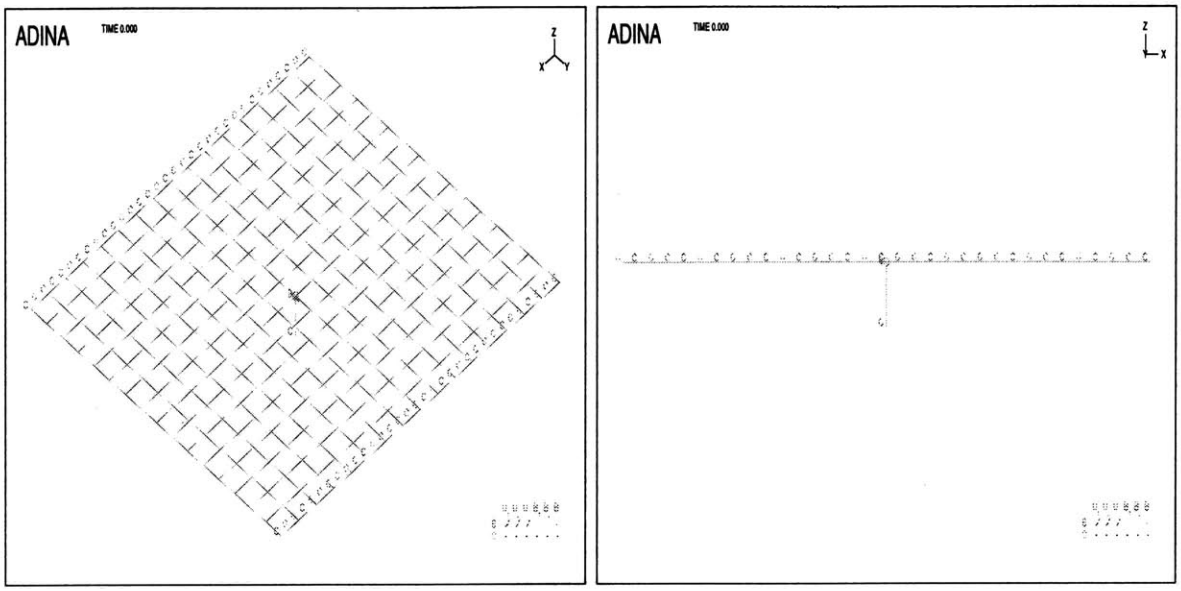
5.1 FEM model of the plate-striker system

5.1.1 System parameters

We defined the sizes and materials of the plate-striker system basically the same as in our experiment, which will be described in chapter 6. For the excitation pressure amplitude and frequency, we choose them so that after nondimensionalization, the excitation amplitude F is small and the excitation angular frequency ω is near one. Damping ratio is configured so that the high-frequency modes will decay faster.

The plate: 16×16 mesh 8-node shell element.

1. length: $a = 0.254m(clamped)$.
2. width: $b = 0.2286m(free)$.



(*a*) isometric view (*b*) front view

Figure 5-1: The configuration of the plate-striker in ADINA

3. thickness $h_p = 1mm$.
4. material: aluminum. $\rho = 2710kg/m^3$, $E = 7 \times 10^{10}Pa$, $\nu = 0.34$.

The striker: 2-node truss element.

1. length: $0.019m$.
2. cross section area: $0.4cm^2$.
3. distance from plate: $1mm$.
4. material: steel. $\rho = 7800kg/m^3$, $E = 7 \times 2^{11}Pa$, $\nu = 0.3$.

The damping ratio: In the FEM simulation, we choose the transient dynamics method to integrate the equation, so every plate mode will contribute to the overall dynamics, no matter what its mode frequency is. We configure the high-modes to have larger damping ratio because they can easily cause acoustical dispersion.

$$2\zeta_i = \frac{\alpha}{\omega_i} + \beta\omega_i \tag{5.1}$$

The α and β are chosen so that the first two damping ratio $2\zeta_1$ and $2\zeta_2$ are both 0.1. Based on such assumption, the damping ratio is actually configured

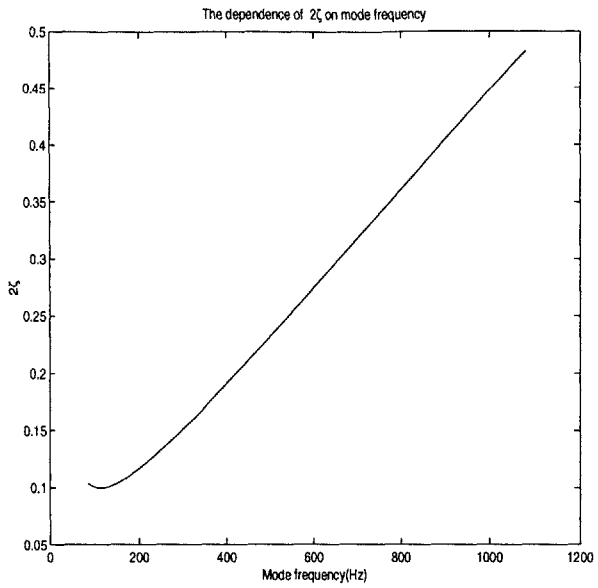


Figure 5-2: Damping ratio assumption in FEM simulation

as shown in figure 5-2.

The excitation pressure and frequency: To compare with our SDF model results, we set the excitation pressure P_e and frequency f so that the nondimensionalized value F and ω are in approximately the same region, see Figure 3-10, as we studied in our SDF model.

For the plate model we defined, checking Table 2.2,we have:

$$Q_1 = 0.5,\ Q_2 = 0.402,\ \lambda = 27.4$$

Substituting the plate parameters into equation (2.22) and (2.25) we get $f_1 = 105Hz$. Therefore

$$\omega = f/f_1 = f/105 \tag{5.2}$$

Using equation (2.16), (2.17), (2.27) and (2.31), we have:

$$F = 1.05 \times 10^{-3} P_e \ when\ h = 1mm \tag{5.3}$$

We will use equation (5.2) and (5.3) to choose our forcing frequency f and pressure P_e in FEM model.

5.1.2 Practical problem related to the time step in integration

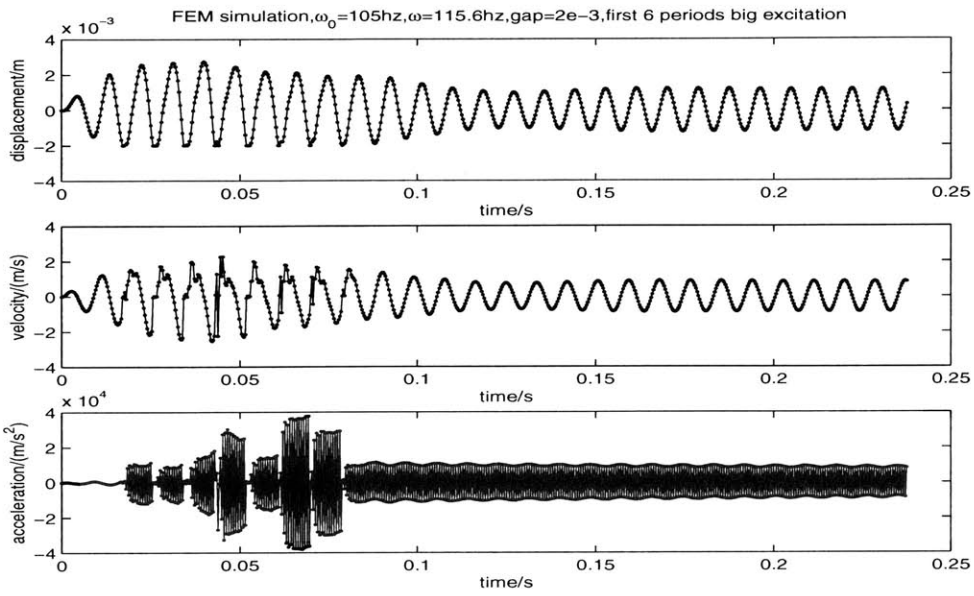


Figure 5-3: The practical issue about the integration time step.

Because of the running time and computer disk space limitation, we can't have a very small simulation time step. In our simulation, we typically use a time step that is $\frac{1}{30}$ of the excitation period. A practical problem arises: When the impact happens, it will cause a lot of high-frequency modes; how are we going to capture those modes? and will those high-modes motion cause instability during integration? The answer to the first question is that we can not capture all the modes. Fortunately, the dynamics integration method used in ADINA insures stability even when the integration time step is much larger than some of the mode period. That means, even though we can't keep the error resulted from under-sampling of high-modes motion to be zero, they won't increase exponentially with time. The stability analysis of different integration methods can be found in [13] chapter 9.

Figure 5-3 demonstrates very clearly the above point. This is a simulation result

of the displacement, velocity, acceleration time history of the plate center. The excitation are high for the first several excitation periods but decreases later, so the impact happens only in the first several periods. We can see the error does exist by checking the acceleration, whose frequency is much bigger then the sampling frequency. We can also see the error doesn't increase by checking the behavior of the acceleration after the last impact. Further more, we can say the error doesn't count in our displacement and velocity simulation results, because after the last impact, the waveform is nearly a sinusoidal one, as predicted by the linear theory. The reason is that the high frequency modes have smaller velocity or displacement, so the integration error in velocity and displacement results are small enough not to be recognized.

5.2 Simulation when there is no striker

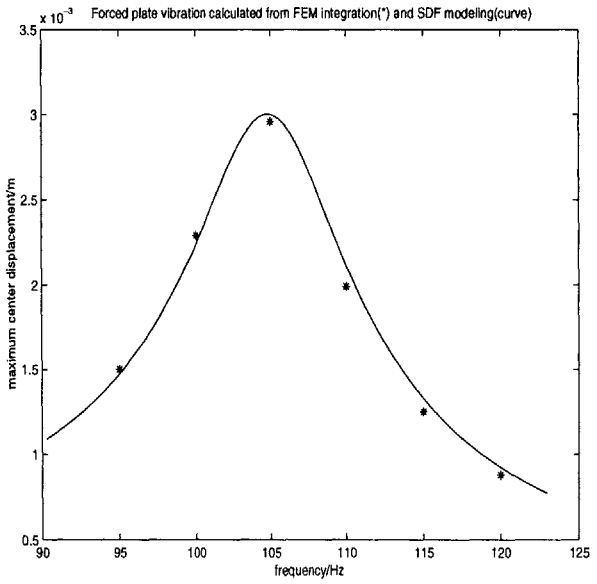


Figure 5-4: The comparison of FEM and SDF of plate vibration without stopper

In this section we do the FEM simulation without the striker; then we compare our SDF model prediction with the FEM simulation results. We choose the excitation pressure $P_e = 285.0Pa$, and excitation frequency f_e from $95Hz$ to $120Hz$. Since f is near $f_1 = 105Hz$, the first mode is the dominating one. Using (2.16), (2.17), (2.27), the first equation of equation array (2.26), we know our SDF model predicts the

maximum displacement of the plate center to be:

$$X_{max} = \frac{3 \times 10^{-4}}{\sqrt{((1 - (\frac{f}{105})^2)^2 + (2\zeta\frac{f}{105})^2)}} \tag{5.4}$$

where $2\zeta = 0.1$, to be equal to the damping assumption in our FEM simulation.

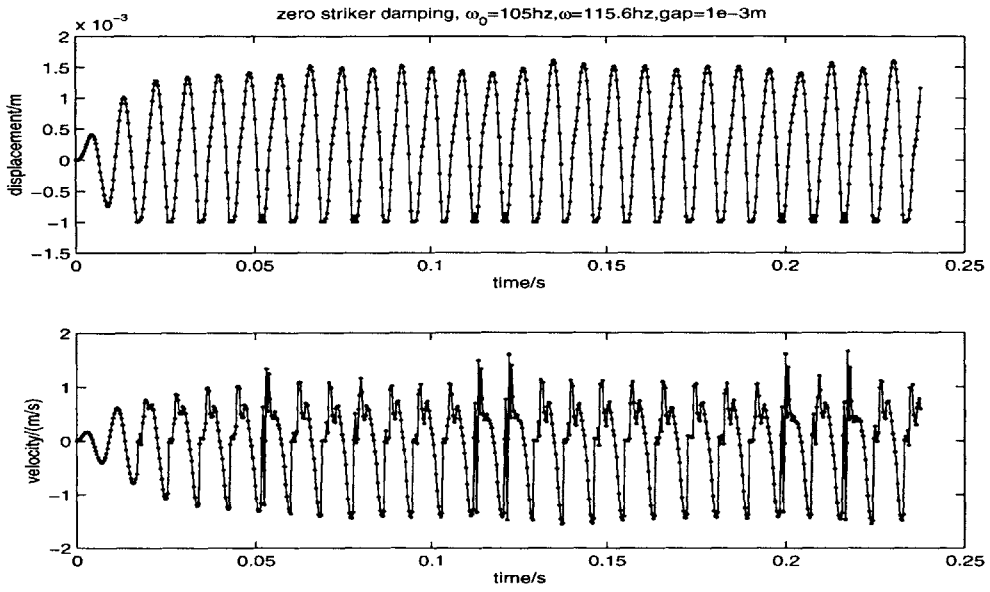
Alternatively, we can find out the maximum plate center displacement by the finite element simulation. Both results from SDF model and FEM simulation are plotted together in Figure 5-4, we see they agree with each other very well.

The 2nd mode frequency of the plate is $120Hz$, it seems the actually plate response should have a peak when the excitation frequency is around $120Hz$, which is not seen in the simulation. The reason is that for the 2nd mode, $Q_1 = 0$, so the excitation for this mode is actually zero.

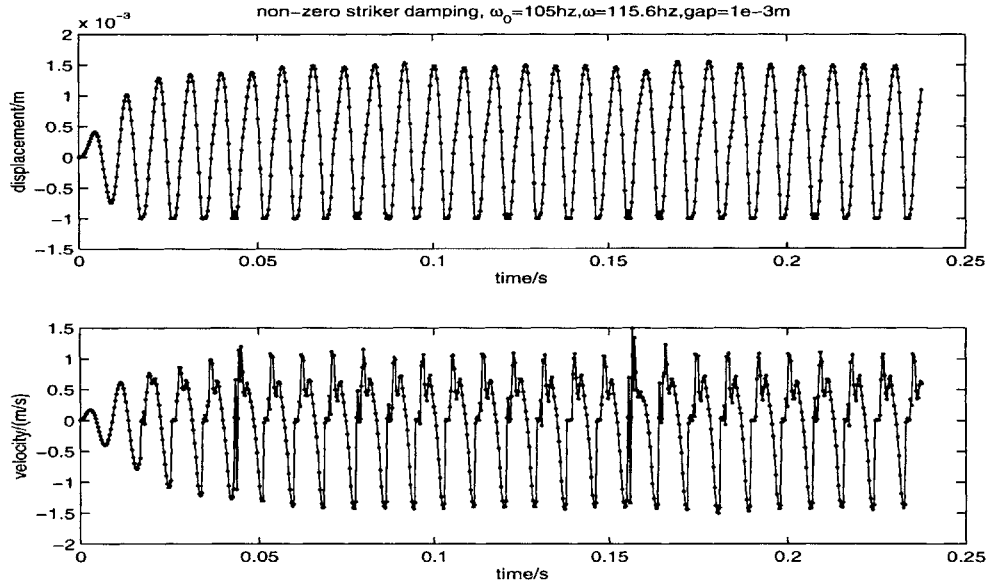
5.3 Simulation when there is striker

5.3.1 Identify the main energy dissipation during impact

There are two causes for the energy dissipation for the main mode during impact, one is the striker damping, the other is the mode coupling to the higher modes. In our configuration, the striker has a small cross section area, so we model it as a truss in the FEM simulation, we guess that the energy lose is mainly due to the mode coupling. To verify this point, we carry out two simulation under identical situation except that the striker damping is different. In the first simulation, the damping of the striker follows the same damping equation of the plate (5.1). Since the mode frequency of the striker is very high(kHz), the damping is large. In the second simulation, the damping of the striker is set to zero.



(*a*) striker has no damping



(b) striker has high damping

Figure 5-5: Comparison of the time history when the striker has no damping or high damping

Simulation results for the plate center are presented in Figure 5-5, as we can see, they are pretty similar. The conclusion is that mode coupling dominate the energy lose of the main mode during impact.

An important issue arises. The restitutional coefficient e for the main mode is actually dependent on the energy coupling mechanism with the higher-modes. To know the e, further research about the coupling effect is required.

We will use the high damping ratio setting for the striker for the later simulation.

5.3.2 Relation between the impact velocity and the gap

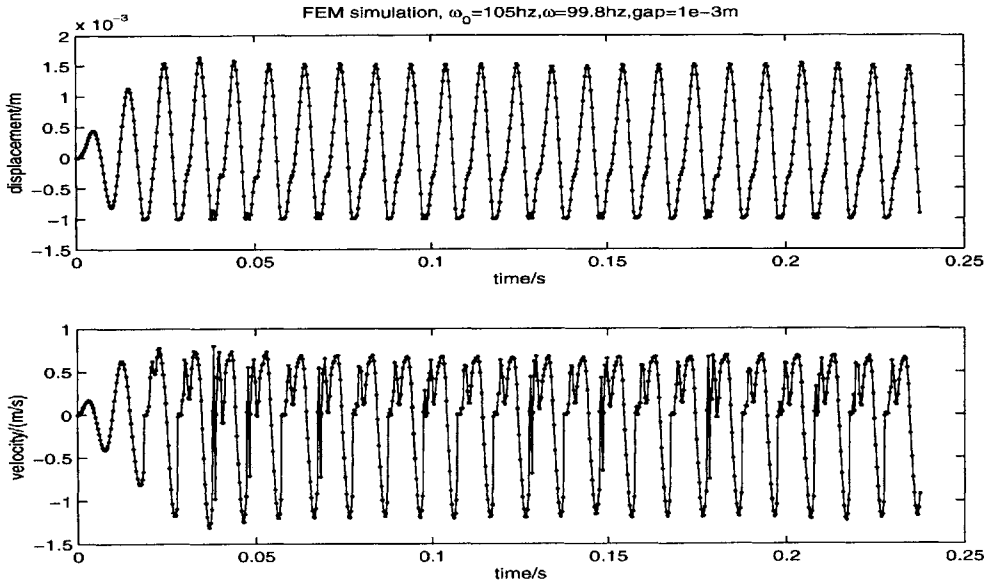


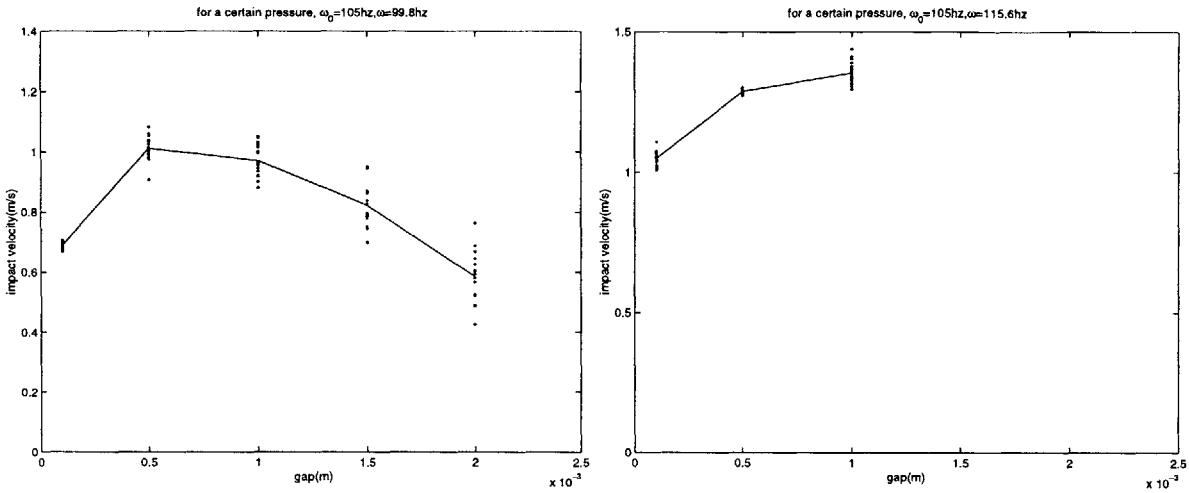
Figure 5-6: FEM simulation, $F = 0.3$, $\omega = 0.95$

Using forcing pressure $P_e = 285Pa$, from equation (5.3), we know $F = 0.3$ for $h = 1mm$. The first mode frequency of the plate is $105Hz$. Two sets of forcing frequency are used, one is $99.8Hz$, corresponding to $\omega = 0.95$; the other is $115.6Hz$, corresponding to $\omega = 1.1$. Under each forcing amplitude and frequency, the gap h are varied, such that F will have a value in the region of 0.1 to 3, and then several simulation results are gotten.

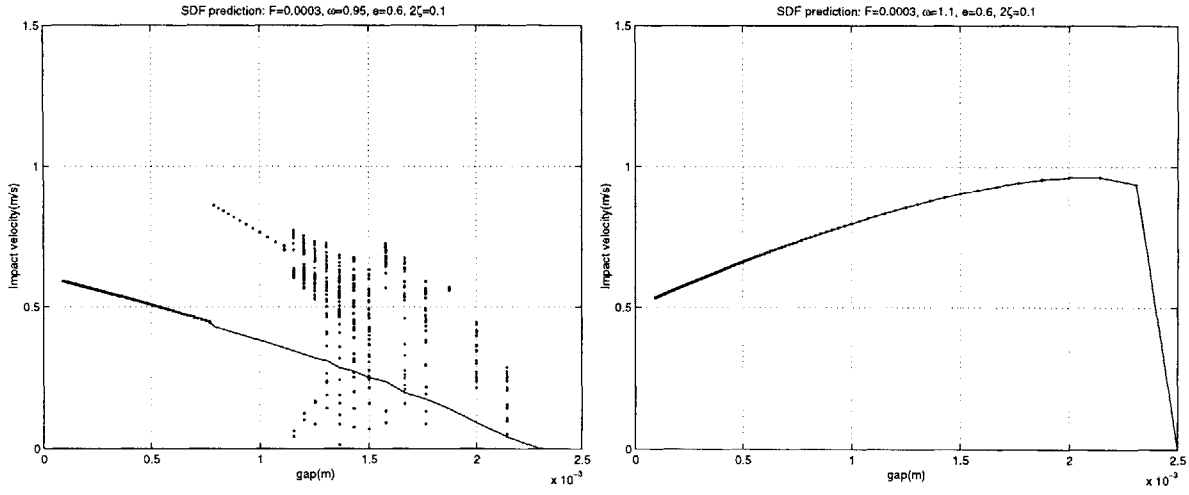
Figure 5-5(b) and Figure 5-6 show the time history of two case each with gap $h = 1mm$, Figure 5-5(b) is for $\omega = 0.95$ and Figure 5-6 is for $\omega = 1.1$. The time for

ADINA to get each of such time history is around 100 minutes.

We evaluate the impact velocity by checking the sudden velocity change point in the time history. Take the period just before 0.1 second in figure 5-6(lower one) for instance, there is a sudden jump of the velocity dots from around $-1m/s$ to $0m/s$, therefore in this case we evaluate the impact velocity to be around $1m/s$.



(a) The FEM simulation, for $e \approx 0.2$.



(b) The SDF prediction, for $e = 0.6$

Figure 5-7: Comparison of Impact velocity got from FEM simulation and SDF prediction.

Figure 5-7(a) shows the dependence of impact velocity upon gap for forcing frequency both below and above the first mode frequency of the plate. We discard the first several impact velocity, regarding them as transient motion. In that figure, dots

denote the individual impact velocity of ten impact events, not including the first several one since they are regarded as transient impacts. The solid line link the average impact velocity together. For comparison, the SDF prediction of the impact velocity are shown in Figure 5-7(b).

Several facts are shown in the figure 5-7:

1. The FEM simulations show that impact velocity will increase(decrease) with the increasing gap if the excitation frequency is above(below) the mode frequency, which is predicted by our SDF model.

2. Our SDF model suggests a large hysteresis region of the plate response if the excitation frequency is above the mode frequency, where the plate may have impact motion or non-impact motion depending on the initial value. Because of the zero initial value of the plate in the FEM model, we haven't got the impact motion simulation results.

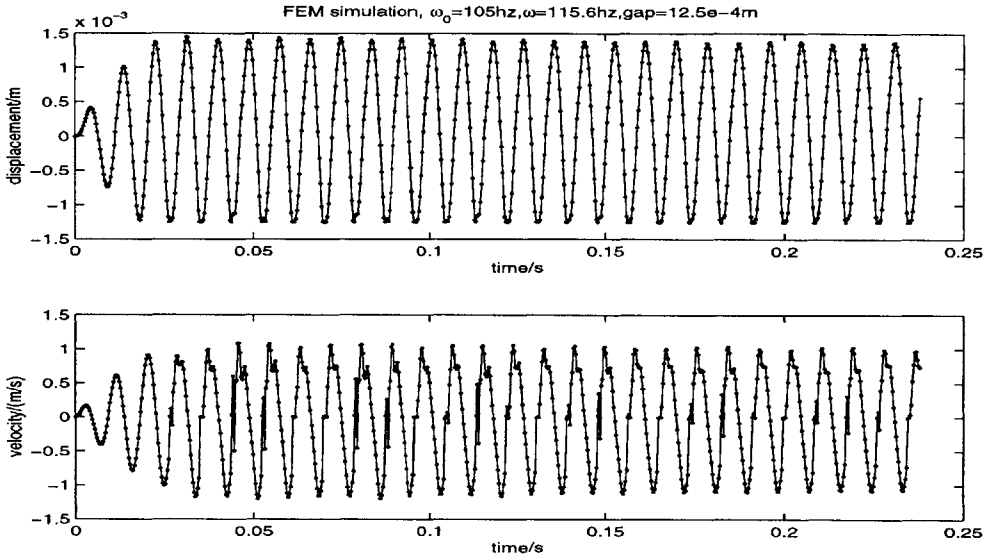
3. By checking figure 5-6, we get an estimation that e is around 0.2. Refer to Figure 4-5, we know lower e results in higher impact velocity, quantitatively the impact velocity with $e = 0.2$ is around twice the impact velocity with $e = 0.6$. That accounts for why the impact velocity in our FEM simulation is around twice that predicted by SDF model with a higher $e = 0.6$.

4. When the gap is near zero for the $\omega = 0.95$ case, the impact velocity is lower than when the gap is bigger, which doesn't agree with our SDF model. The reason is that if h is too small, the nondimensionalized excitation amplitude F will be much larger. At that case, our SDF assumption that the main mode dominates the dynamics no longer holds.

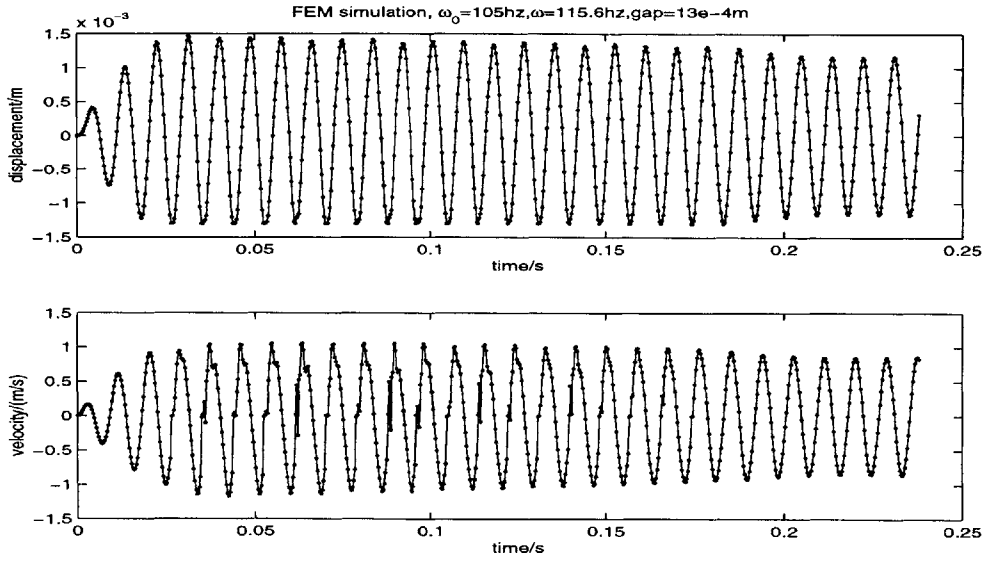
5. We don't see $1/2$ periodic motion in the FEM simulation, though it is predicted by our SDF model. Two possibility exist: first, the integration may not be long enough to reach the steady state of the $1/2$ motion; second, $1/n$, $n = 2, 3, 4...$, motions may not exist for the actual plate which has many degree of freedom. However, we can still observe that the impact velocity distribution is more

messy when the forcing frequency is below the main mode frequency, which is suggested by our SDF model.

5.3.3 The transient impact motion of the plate



(a) gap=12.5e-4m



(b) gap=13e-4m

Figure 5-8: FEM simulation for the transient impacts.

In the FEM simulation, we did some simulation with zero initial condition and with the gap a little bit larger than the value which linear theory will predict no steady impact motion. The results are shown in figure 5-8. We can see that the impact motion can continue for many periods, and is possible to continue for ever for the case (a). Note the impact motion in case(a) has a smaller impact velocity($< 1m/s$) than the one predicted by our SDF model, which is supposed to be, refer to figure 5-7(b), a little bitter larger than the one in figure 5-6($\approx 1m/s$). Such kind of weak impact velocty motion is probably the third possible response for the plate due to the high-frequency modes nonlinearity, different from the linearly-predicted no-impact motion or the SDF-predicted strong-impact motion. Small disturbance can easily set the plate into such weak impact motion and can also easily terminate it, in this sense it's transient. Actually such transient impact can be heard frequently in real world.

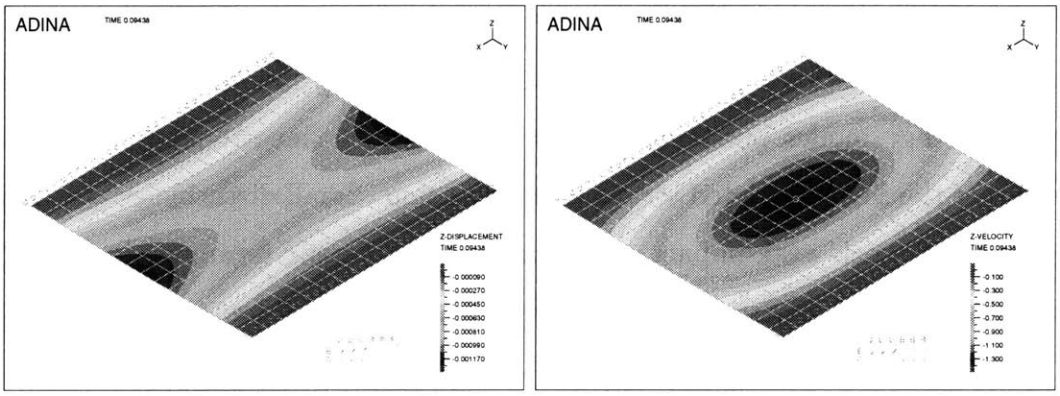
5.3.4 The plate displacement and velocity profile

It's interesting to see the actual distance and velocity distribution on the plate. Such figures are obtained for $\omega = 1.1$, $h = 1mm$ and are show in Figure 5-9.

Refer to the mode shapes shown in Figure 2-2, we can get some idea of how modes couple and interact during the impact motion.

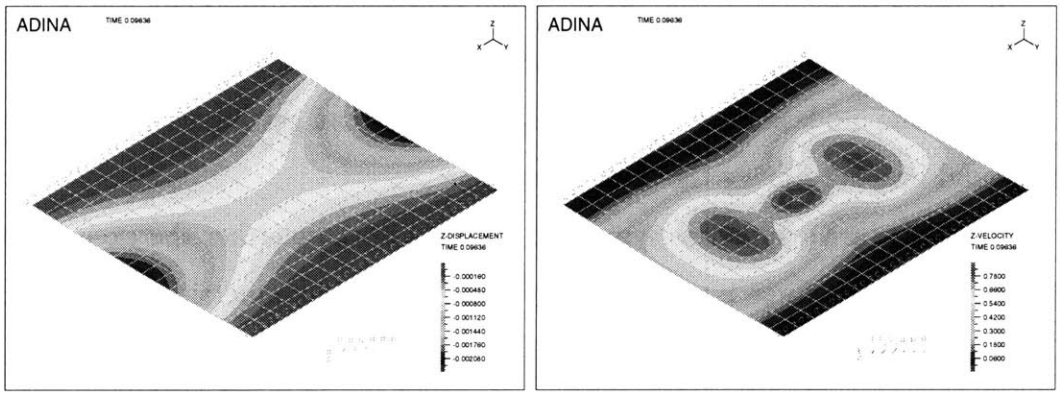
The profiles are symmetric-symmetric. That's because either the excitation pressure or the contact force is symmetric-symmetric, only symmetric-symmetric modes can be generated.

The mode shape of mode 1 dominates the profiles. But the second symmetric-symmetric mode, which is mode 3 shown in Figure 2-2(c), changes the profiles to some extend. As we can see, the contribution from mode 3 is more apparent after the impact than before the impact, that's because mode 3 attenuates faster than mode 1 due to the larger damping ratio.



(a) (b)

Just before the impact



(c) (d)

Just after the impact

Figure 5-9: The displacement and velocity of the plate during the vibration

Chapter 6

Experimental study

In order to test the adequacy of our single-degree-of-freedom model, we carry out experiments on a thin plate impacting on a striker.

6.1 Equipment setup

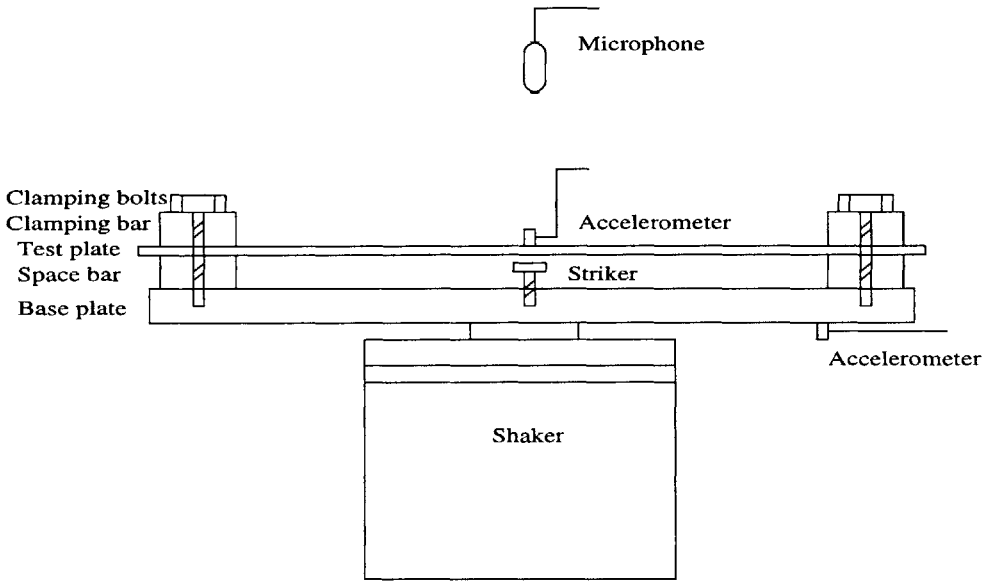


Figure 6-1: Experiment setup

A test apparatus shown in Figure 6-1 is set up on an electro-magnetic shaker. A thick aluminum square plate of the size 10 × 10 inches is mounted on a B&K 4801 shaker to be used as a base-excitation. A thin plate is then mounted on the base.

The plate are clamped on two opposite edges and free on the other two edges. The dimensions of thin plates are 22.86 $\times$ 25.40 mm (9 by 10 inches). The shorter edges are free. This test configuration is arrived at after several unsuccessful trials. The difficulty was to minimize the effect of the squeezed air between the thin plate and the base. The effect of the squeezed air would diminish if the fundamental frequency of the thin plate is very low. The geometry and the boundary conditions are chosen to achieve low resonance frequency and to allow the air in the gap to freely move.

A bolt connected to the center of the base acts as a striker. The gap between the striker and the top plate can be adjusted.

To measure the excitation and the plate responses, accelerometers are mounted to the bottom and top plates. For sound measurement, a microphone is placed about 0.3 meters from the top plate. An FFT analyzer is used as the data acquisition, storage and processing device. A laser vibrometer is used to measure the velocity of the center of the plate.

Several factors severely complicate our experiment. First of all, our shaker system is open-loop; hence the shaker response is affected by the thin plate dynamics. This coupling becomes strong especially when impact occurs between the thin plate and the striker. As a result, the base excitation contains higher harmonics and the amplitudes of the higher harmonics often exceed the amplitude at the fundamental frequency. Second, our shaker excitation contains a component caused by the cooling fan, which has a blade frequency near 88 Hz. This secondary excitation superimposes onto the shaker excitation and causes beating phenomenon.

Since our open-loop shaker system generates satisfactory sinusoidal base-excitations in the absence of the impact, we report the experimental threshold for the onset of the impact. We also avoid the case with very small gaps to minimize the relative effect of the cooling fan excitation. Our emphasis is placed on the frequency dependence of the threshold, the hysteresis, and the sound pressure levels.

6.2 Experimental Results

6.2.1 Measuring of the damping ratio of the plate

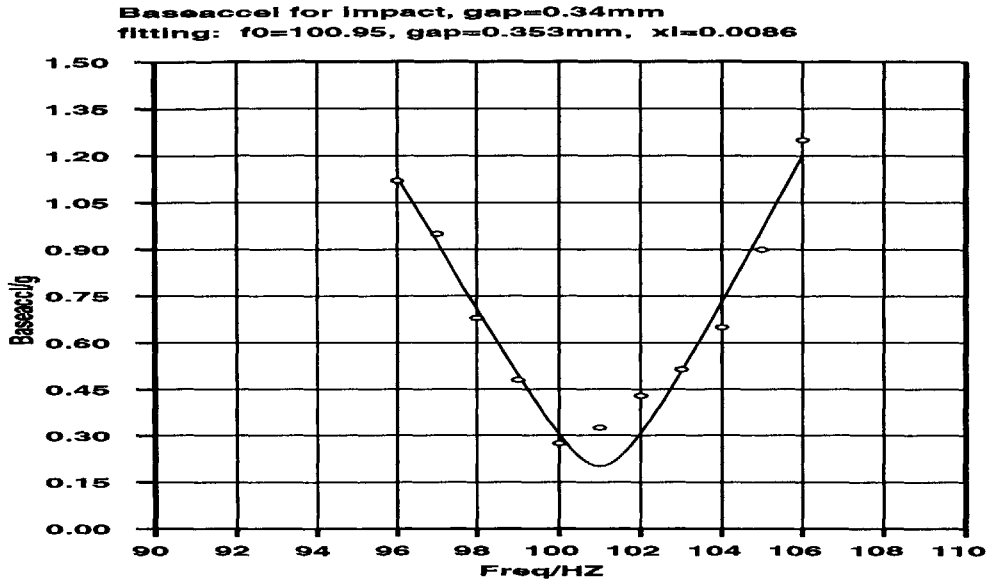
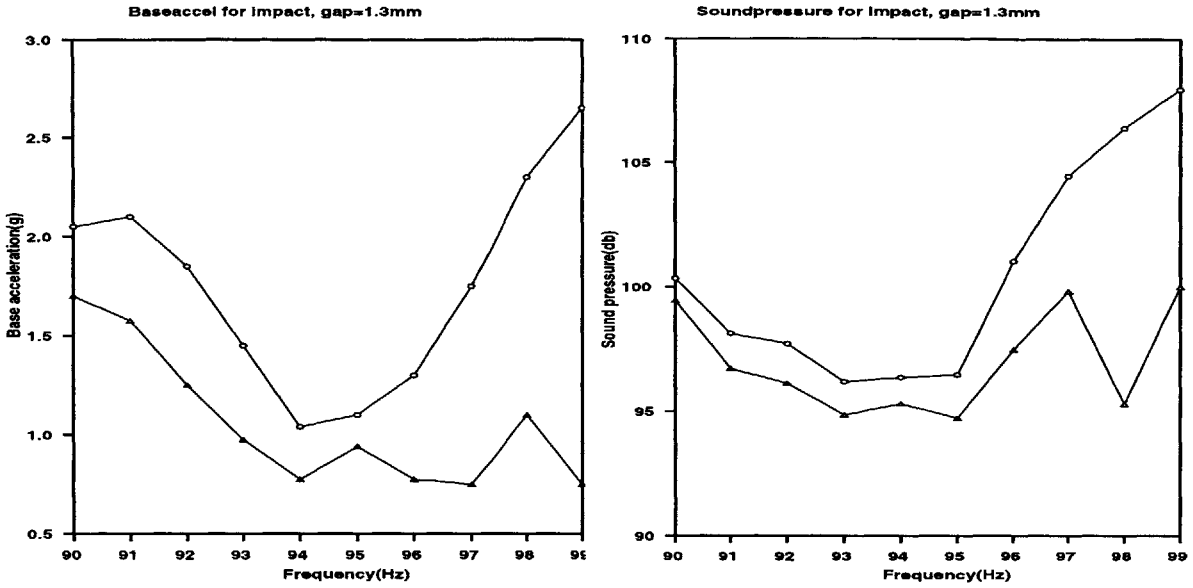


Figure 6-2: The fitting of the grazing point using SDF model

Experiment was conducted by varying the excitation frequency and changing the excitation amplitude, so as let the plate can just touch the striker. By fitting the data into the dimensional form of Equation (3.4), we can estimate the damping ratio and the gap between striker and plate center. By comparing the gap between fitting value and the actually measured value, we can know the accuracy of the SDF assumption Equation (3.4). The result is shown in Figure 6-2. We can see the measured gap is $0.34mm$, and the gap estimated by fitting is $0.35mm$, pretty close. And the estimated damping ratio for the plate is $2\zeta = 0.017$.

6.2.2 The impact vibration experiment

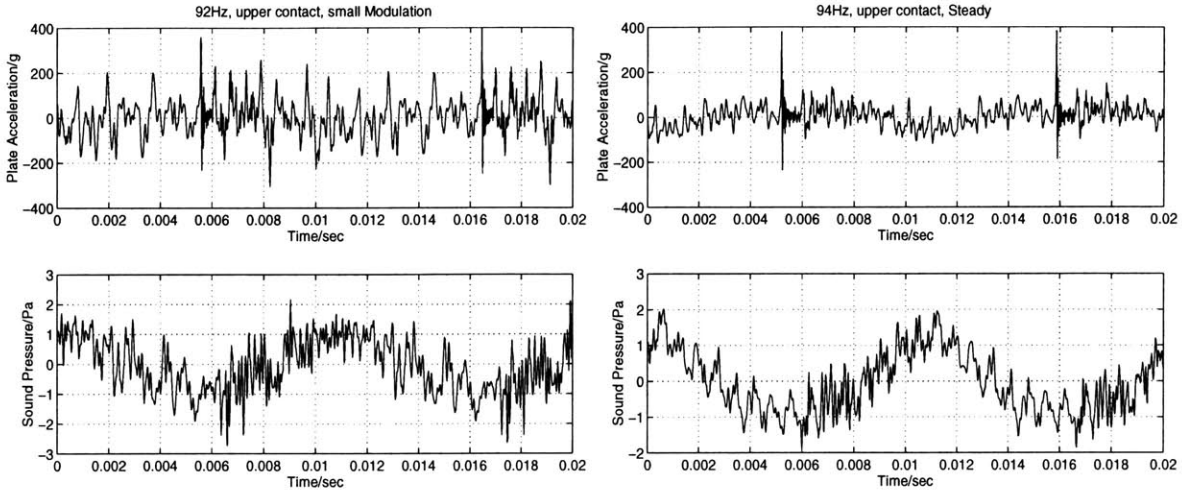
Experimental data are taken for an aluminum plate with thickness 0.96 mm.Based on the acceleration of the thin plate when impact is just about to occur, we can infer the gap between the plate and the striker to be 1.3mm. The plate is excited at a fixed frequency. The forcing amplitude is slowly increased until impact appears. The



(a) base acceleration (b) sound pressure

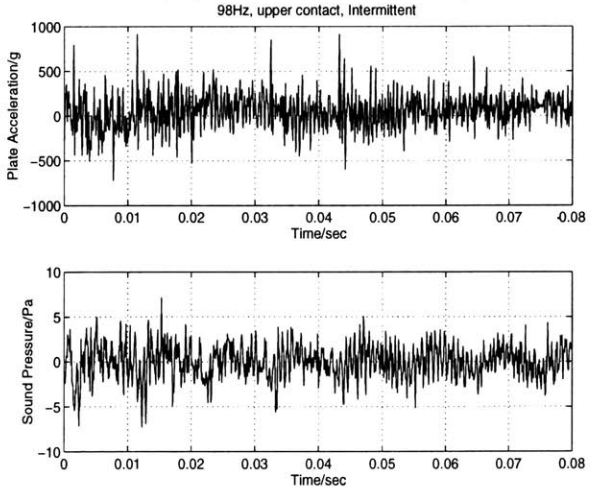
Figure 6-3: The base acceleration and sound pressure at the edge of impact and non-impact

corresponding forcing amplitude gives the upper curve in Fig 6-3(a). Similarly, by slowly decreasing the forcing amplitude until impact disappears, we obtain the lower curve in Fig 6-3(a). The difference between the two thresholds indicates the hysteresis in the system dynamics. The upper curve has the lowest value near 94 Hz, which is close to the theoretical resonance frequency of the lowest mode.

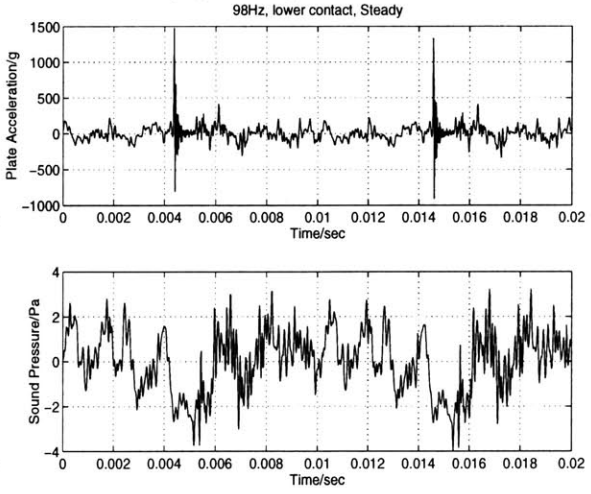


(a) small modulation pattern

(b) steady Pattern



(c) Intermittent pattern



(d) steady pattern

Figure 6-4: Experimental waveforms of plate acceleration and sound pressure corresponding to some cases in the previous figure

The impact patterns are different at different forcing frequencies. Fig 6-4(a)-(d) show the plate acceleration at the center and the sound pressure measured by the microphone corresponding to the forcing parameters in Fig. At (a) and (b), the impact pattern is periodic with one impact in each forcing cycle. At (c), there are more than one impact within one forcing cycle. The impact pattern is not steady. However at a reduced forcing amplitude, (d), the impact pattern is steady with one impact in each forcing cycle.

We notice significant differences of plate impact depending on whether the forcing is above or below the resonance frequency. First of all, as can be seen in Figure 6-4(a), the hysteresis above the resonance frequency is much larger. This is consistent with our analysis of a single-degree-of-freedom model; Figure 6-5 shows a large hysteresis region for forcing frequencies above the resonance. Second, the rattle noise level above the resonance frequency is higher than that below the resonance frequency. This is shown in Figure 6-3(b) which is a plot of the sound pressure level measurement at the threshold of impact.

The impact dynamics determines the radiated sound pressure through the impact velocity. Figure 6-6 shows the sound pressure together with the plate velocity. An impact event is noted by very fast change in the impact velocity and a large peak in the sound pressure. Impact event is noted by small circle in the impact velocity time history of Figure 6-6. Notice a time delay between the impact and the peak sound pressure. Occasionally, a larger peak of the sound pressure follows the first peak as shown in Figure 6-6(b) . We define these two peaks as impact sound pressure(first peak) and maximum sound pressure. In most cases, the maximum sound pressure and the impact sound pressure coincide, as in Figure 6-6(a).

We have also plotted the RMS sound pressure as it changes with the impact velocity. As can be seen in Figure 6-7, in the first order of approximation, the sound pressure level, no matter the maximum, impact, or average one, is proportional to the impact velocity.

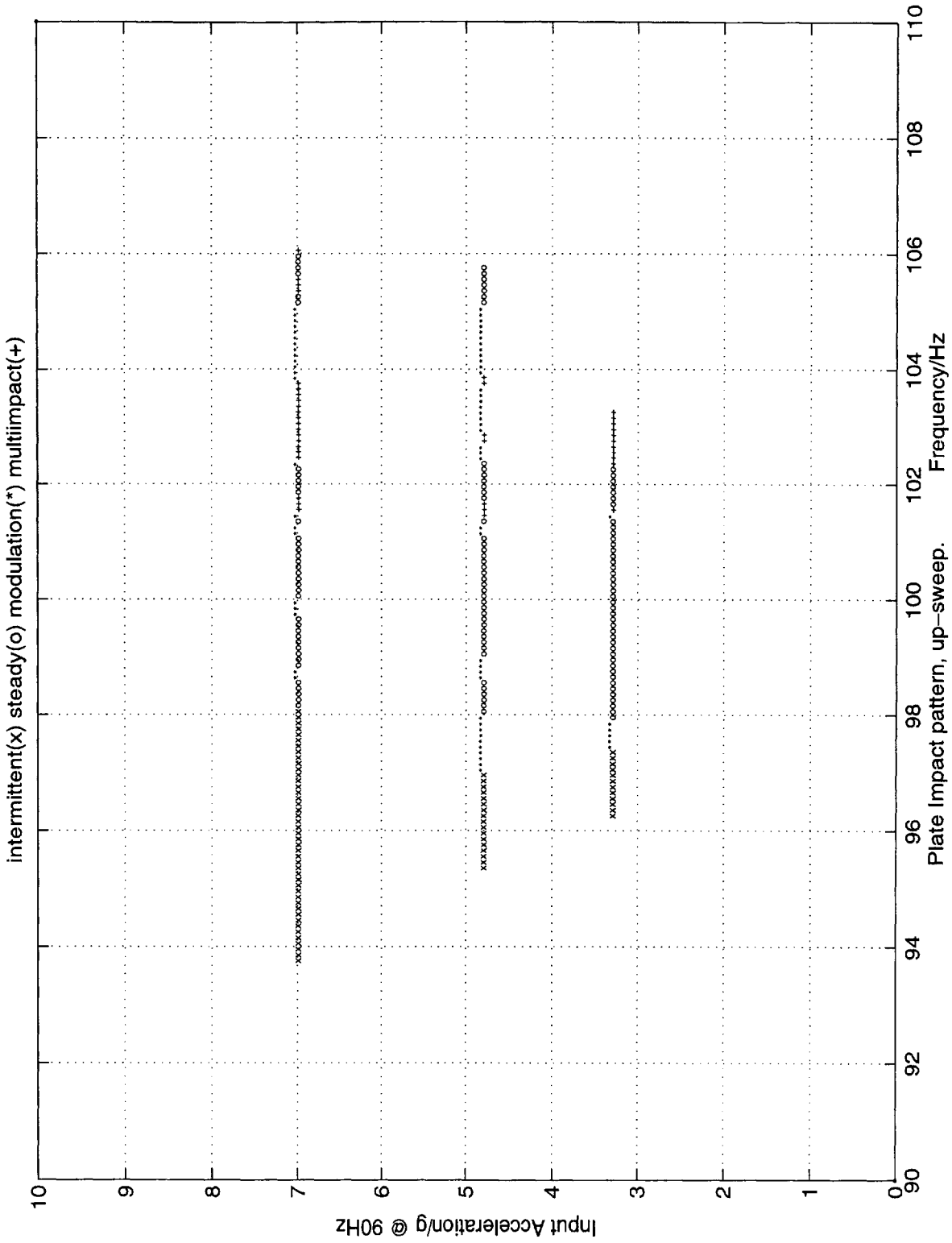
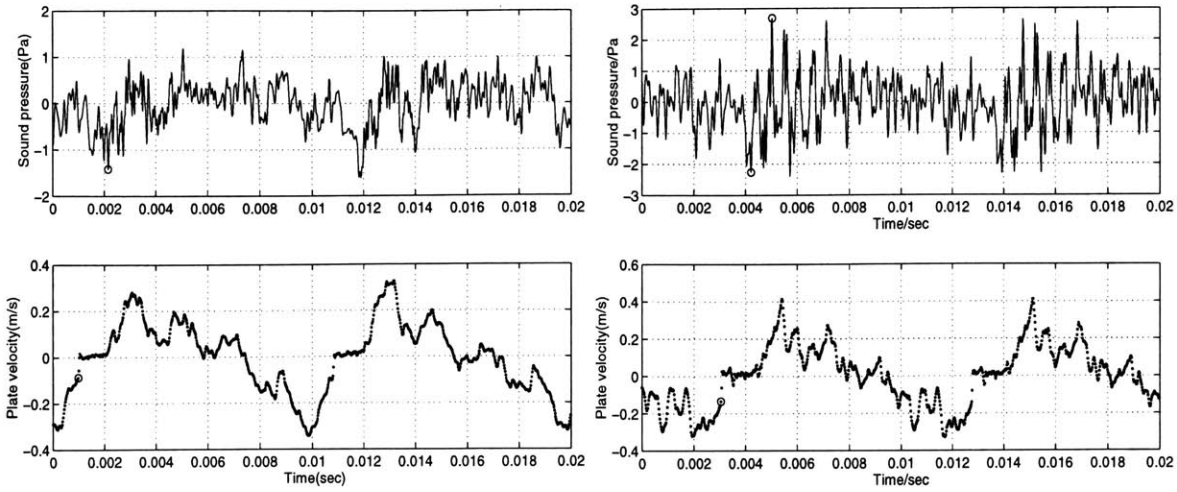


Figure 6-5: vibration pattern, plate resonant frequency $\approx$ 98Hz.



(a) (b)

Figure 6-6: Two sample experimental waveforms of plate center velocity and sound pressure.

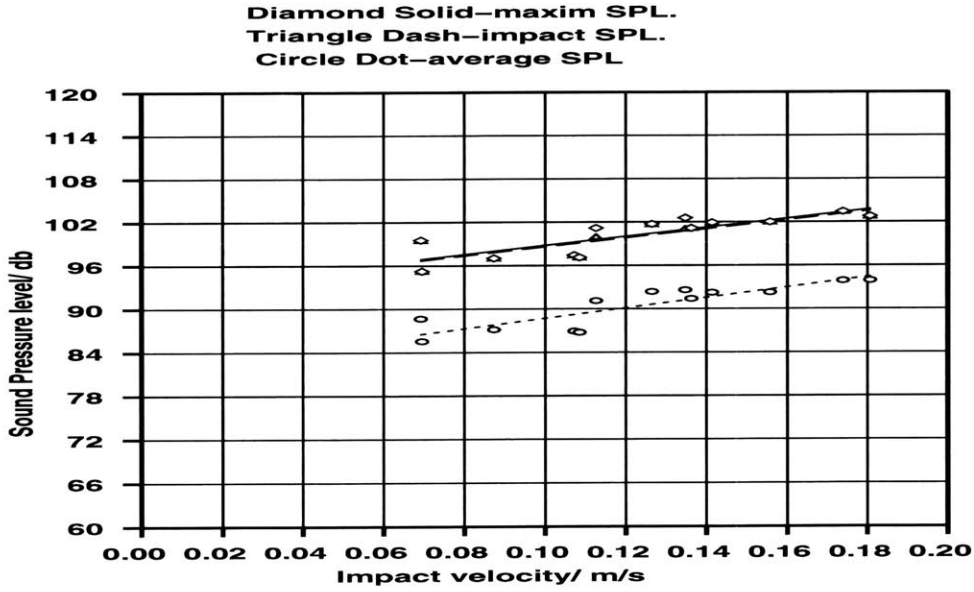


Figure 6-7: SPL vs impact velocity

Chapter 7

Conclusion

7.1 Main results

The SDF model serves as an important tool to the understanding of the plate vibration. It shows:

- plate can have complicate dynamics pattern; nevertheless, the average impact velocity will change gradually with the changing of system parameters or excitation variables.
- Impact velocity increases with the increasing of gap if the excitation frequency is above the main mode frequency of the plate, and will decrease with the increasing of gap if the excitation frequency is below the main mode frequency of the plate.
- The bigger the bouncing back ratio e for the main mode, the smaller the impact velocity.

The FEM simulation shows:

- The e depends on the energy coupling from the main mode to the higher mode.
- the SDF model captures the dependence of impact velocity upon gap, thus a reasonable model.

The experiment shows:

- The plate vibration does have complex patterns. The hysteresis when the excitation frequency above the natural frequency is really apparent, same as indicated by our SDF model.
- It's a good estimation that the rattle noise sound pressure level is linear to the impact velocity.

7.2 Further work

To know the e, we at least need to include the second symmetric-symmetric mode in our simulation. Hopefully that will give us good estimation of the impact velocity. And then we should carry on the study of coupling to higher modes, which will produce noise.

This thesis puts emphasis on the dynamics of the rattle vibration, for estimation of the noise level, we still have to know the dependence of the noise on the impact dynamics. Our experiment study shows there is, in the first order approximation, linear dependence of noise level on the impact velocity; further study is required.

Appendix A

Java program for the SDF model simulation

The java program is divided into 5 portions with different functions.

A.1 Impact_sys.java

```
import java.awt.*;
import java.util.*;
import java.applet.Applet;
import java.awt.image.*;
import java.awt.Graphics.*;
import java.io.*;
import java.awt.event.*;
import java.lang.Double.*;

class Impact_sys {                                                    10
  iap miap;
  double A,F;     // A is forcing amplitude, F is forcing angular frequency
  double m,b,k,gap,e; // System parameters
  double t,x,v,a;       // Dynamic variables
  double h;        // Numerical animation's time step
  int accurate_extend=200; // h=T_forcing/200
  double faye; // excitation phase.

  // set the initial value
  Impact_sys                                                          20
  (iap iap, double iA, double iF, double im, double ib, double ik, double igap, double ir) {
    miap=iap;
    A=iA;
    F=iF;
```

```
    m=im;
    b=ib;
    k=ik;
    gap=igap;
    e=ir;
    t=x=v=a=0;                                                    30
    setF_related(iF);
    faye=0.0;
  }

  double cos_term_at(double t,double phase) {
    return Math.cos(F*t+faye+phase);
  }

  double acceleration(double t, double x, double v) {
    return (A*cos_term_at(t,0.0)-b*v-k*x)/m;                      40
  }

  // the function to set the related value corresponding to the excitation frequency.
  void setF_related(double nF) {
    faye+=(F-nF)*t;
    faye=faye-Math.PI*2.0*(int)(faye/Math.PI/2.0);
    F=nF;
    h=1.0/F/accurate_extend*2*Math.PI;
    miap.t_span=1.0/F*miap.mShowingPanel.getValue()*2*Math.PI;
  }                                                               50

double v_lastimpact;

  // the function to move the system into the next time step.
  boolean evolve(double ih) {    // return true if impacted
    double gap;

    // if no stopper is assumed, set gap to be infinity.
    if (miap.mSystemPanel.linearChecker.getState()) gap=1e99;
    else gap=this.gap;                                            60

    boolean impacted=false;

    boolean stuck=(x>=gap);
    double mh=ih;
    double l1,l2,l3,l4;
    l1=mh*acceleration(t,x,v);
    l2=mh*acceleration(t+mh/2.0,x+mh*v/2.0,v+l1/2.0);
    l3=mh*acceleration(t+mh/2.0,x+mh*v/2.0+mh*l1/4.0,v+l2/2.0);
    l4=mh*acceleration(t+mh,x+mh*v+mh*l2/2.0,v+l3);               70
    if ( x+mh*(v+(l1+l2+l3)/6) < gap) {
      t+=mh;
      x+=mh*(v+(l1+l2+l3)/6);
      v+=(l1+2*l2+2*l3+l4)/6;
      a=acceleration(t,x,v);
    }
    else if (stuck) {
      t+=mh;
```

```
  x=gap;
  v=a=0;
 }
 else {
  impacted=true;
  double v_old=v;
  mh=(gap-x)/( (v+(l1+l2+l3)/12) );
  l1=mh*acceleration(t,x,v);
  l2=mh*acceleration(t+mh/2.0,x+mh*v/2.0,v+l1/2.0);
  l3=mh*acceleration(t+mh/2.0,x+mh*v/2.0+mh*l1/4.0,v+l2/2.0);
  l4=mh*acceleration(t+mh,x+mh*v+mh*l2/2.0,v+l3);
  t+=mh;
  x+=mh*(v+(l1+l2+l3)/6);
  v=-e* ( v+(l1+2*l2+2*l3+l4)/6 );
  v_lastimpact=v;
  mh=ih-mh;
  l1=mh*acceleration(t,x,v);
  l2=mh*acceleration(t+mh/2.0,x+mh*v/2.0,v+l1/2.0);
  l3=mh*acceleration(t+mh/2.0,x+mh*v/2.0+mh*l1/4.0,v+l2/2.0);
  l4=mh*acceleration(t+mh,x+mh*v+mh*l2/2.0,v+l3);
  t+=mh;
  x+=mh*(v+(l1+l2+l3)/6);
  v+=(l1+2*l2+2*l3+l4)/6;
  a=(v-v_old)/ih;
  if (x>gap && v>0.0) {
       a=v=0.0;
       x=gap;
  }
 }
 return impacted;
}

// function to fill the simulation data buffets.
void fillbuffer() {
 miap.t_begin=(Math.abs(miap.t_begin-t)>2*h)?t:miap.t_begin;
 double t_end= miap.t_begin+ miap.t_span;
 double x_max=x,x_min=x;
 double v_max=v,v_min=v;
 double a_max=a,a_min=a;
 int i;
 double faye;

 for (i=0; t<t_end-miap.t_span*0.0001; i++) {
  miap.tbuffer[i]= t- miap.t_begin;
  miap.xbuffer[i]=x;
  miap.vbuffer[i]=v;
  miap.abuffer[i]=a;
  faye=Math.atan( b*F/(1-F*F) );
  faye=(faye<0)?faye+Math.PI:faye;

  miap.x_forbuffer[i]=A/Math.sqrt((1-F*F)*(1-F*F)+(b*F)*(b*F))*cos_term_at(t,-faye);
  miap.excitationbuffer[i]=        cos_term_at(t,0.0);
  x_max=(x_max>x)?x_max:x;
  x_min=(x_min<x)?x_min:x;
```

```
    v_max=(v_max>v)?v_max:v;
    v_min=(v_min<v)?v_min:v;
    a_max=(a_max>a)?a_max:a;
    a_min=(a_min<a)?a_min:a;

    if (miap.mNotePanel.impactonlybox.getState())
        getimpactv(t_end);
    else evolve(h);                                          140
  }
  miap.buffersize=i;
  miap.t_begin=t_end;
  miap.x_max=x_max;
  miap.x_min=x_min;
  miap.v_max=v_max;
  miap.v_min=v_min;
  miap.a_max=a_max;
  miap.a_min=a_min;
                                                             150
}


// the fuction to discard transient motion.
void gointosteady(int periods) {
  int i_max=periods*accurate_extend;
  int i;

  for (i=0; i<i_max; i++) {
    evolve(h);                                               160
  }
}

// function to get the next impact velocity.
double getimpactv(double t_end) {
  double v_average=0;
  for (; t<t_end;) {
    if (evolve(h)) return -v_lastimpact/e;
  }
  return v_average;                                          170
}

// function to get the velocity when the mass is passing the zero displacement.
double getx0v(double t_end) {
  double v_average=0;
  double x0=0.0;

  double x_old=x;
  for (; t<t_end;) {
    evolve(h);                                               180
    if (x_old<x0 && x>x0) {
        return v-a*h*(x-x0)/(x-x_old);
    }
    x_old=x;
  }
  return v_average;
```

```
}


// function to fill in impact velocities corresponding to one set of system parameter
// into the velocity buffets.
double fillv_impbuffer(int i) {
  double vm;
  double t_begin;
  double t_end;
  double v_average=0;
  double periods=64.0;
  double v_old, v_old1, v_old2;
  // compare v and v_old to prevent 1/1 motion from takeing up too much space!
  int old_v_buffersize=miap.v_buffersize;

  if (miap.varyingitem==miap.OMEGA) {
    A=miap.mForcingPanel.forcingAccelerationPanel.getValue();
    setF_related(miap.v_imp_x);
  }
  else if (miap.varyingitem==miap.AMPLITUDE) {
    A=miap.v_imp_x;
    setF_related(miap.mForcingPanel.forcingFrequencyPanel.getValue());
  }
  else {
    e=miap.v_imp_x;
  }

  miap.mForcingPanel.forcingAccelerationPanel.setValue(A);
  miap.mForcingPanel.forcingFrequencyPanel.setValue(F);
  miap.mSystemPanel.ePanel.setValue(e);

  gointosteady(100);
  t_begin=t_end=t+periods*2*Math.PI/F;

  v_old=v_old1=v_old2=-1e99;
  vm=0;
  for (;t<t_end ;v_average+=vm/periods) {
    miap.v_imp_xbuffer[miap.v_buffersize]=miap.v_imp_x;
    v_old2=v_old1;
    v_old1=v_old;
    v_old=vm;
    if ( (vm=getimpactv(t_end))>1e-6 && Math.abs(vm-v_old)>1e-6*Math.abs(vm) &&
          Math.abs(vm-v_old1)>1e-6*Math.abs(vm) &&
          Math.abs(vm-v_old2)>1e-6*Math.abs(vm)) {
        miap.v_impbuffer[miap.v_buffersize]=vm;
        miap.v_buffersize++;
    }
  }
  if (miap.v_buffersize==old_v_buffersize) {
    miap.v_impbuffer[miap.v_buffersize]=0.0;
    miap.v_buffersize++;
  }

   int old_v_x0buffersize=miap.v_x0buffersize;
```

```
    t_end=t+periods*2*Math.PI/F;

    v_old=v_old1=v_old2=-1e99;
    //  if 1/1, not  want the same v at 0 take up too many space!!!
   for (;t<t_end  && Math.abs(vm-v_old)>1e-6*Math.abs(v)  &&
            Math.abs(vm-v_old1)>1e-6*Math.abs(vm) &&
            Math.abs(vm-v_old2)>1e-6*Math.abs(vm);) {
     miap.v_x0_xbuffer[miap.v_x0buffersize]=miap.v_imp_x;
     v_old2=v_old1;
     v_old1=v_old;                                                    250
     v_old=vm;
     if ( (vm=getx0v(t_end))>1e-6 ) {
          miap.v_x0buffer[miap.v_x0buffersize]=vm;
          miap.v_x0buffersize++;
     }
   }
   if (miap.v_x0buffersize==old_v_x0buffersize) {
     miap.v_x0buffer[miap.v_x0buffersize]=0.0;
     miap.v_x0buffersize++;
   }                                                                  260
     return v_average;
  }

}
```

A.2 V_impFrame.java

```
import java.awt.*;
import java.util.*;
import java.applet.Applet;
import java.awt.image.*;
import java.awt.Graphics.*;
import java.io.*;
import java.awt.event.*;
import java.lang.Double.*;

class V_impFrame                                                    10
extends Frame implements Runnable {
  iap miap;
  String conname, begname, endname, stpname;
  TextField
    conField, begField, endField, stpField;
  Button startButton;
  Button stopButton;
  boolean beingrunned;
  boolean inAnApplet=true;
                                                                    20
  V_impFrame(iap iap, int varyingitem) {
    miap=iap;
    beingrunned=false;
    if (varyingitem==miap.OMEGA) {
      miap.varyingitem=miap.OMEGA;
      conname="A=";
      begname="w_begin=";
      endname="w_end=";
      stpname="w_step=";
    }                                                               30
    else if (varyingitem==miap.AMPLITUDE) {
      miap.varyingitem=miap.AMPLITUDE;
      conname="w=";
      begname="A_begin=";
      endname="A_end=";
      stpname="A_step=";
    }
    else {
      miap.varyingitem=miap.RESTITUTION;
      conname="==";                                                 40
      begname="e_begin=";
      endname="e_end=";
      stpname="e_step=";
    }

    conField=new TextField("0.4",10);
    begField=new TextField("0.5",10);
    endField=new TextField("1.5",10);
    stpField=new TextField("0.1",10);
    startButton=new Button("start");                                50
    startButton.enable();
```

```
  stopButton=new Button("stop");
  stopButton.disable();

  setLayout(new GridLayout(4,0));
  add( Paneladded(begname, begField) );
  add( Paneladded(endname, endField) );
  add( Paneladded(stpname, stpField) );
  Panel nPanel=new Panel();
  nPanel.setLayout(new GridLayout(0,2));                                    60
  nPanel.add(startButton);
  nPanel.add(stopButton);
  add(nPanel);
  setTitle("Panel for calculate V_imp");
  pack();
  show();
  validate();
}

public void run() {                                                         70
  double v_imp_max=-1e99,v_imp_min=1e99;
  double v_imp_x_max=-1e99,
    v_imp_x_min=1e99;
  int i;
  double v;

  miap.v_buffersize=0;
  miap.vave_buffersize=0;
  miap.v_x0buffersize=0;
                                                                            80
  for (i=0,miap.v_imp_x=miap.v_imp_x_begin;
         beingrunned &&
           miap.v_imp_x_step*
           (miap.v_imp_x-miap.v_imp_x_end)<=0;
         i++,
           miap.v_imp_x+=miap.v_imp_x_step) {
    v=miap.mImpact_sys.fillv_impbuffer(i);
    miap.v_impavebuffer
        [miap.vave_buffersize]=v;
    miap.v_impave_xbuffer                                                   90
        [miap.vave_buffersize]=miap.v_imp_x;
    miap.vave_buffersize++;
    v_imp_max=(v_imp_max>v)?v_imp_max:v;
    v_imp_min=(v_imp_min<v)?v_imp_min:v;
    v_imp_x_max=(v_imp_x_max>miap.v_imp_x)?
        v_imp_x_max:miap.v_imp_x;
    v_imp_x_min=(v_imp_x_min<miap.v_imp_x)?
        v_imp_x_min:miap.v_imp_x;
  }
                                                                            100
  miap.v_imp_max=v_imp_max;
  miap.v_imp_min=v_imp_min;
  miap.v_imp_x_max=v_imp_x_max;
  miap.v_imp_x_min=v_imp_x_min;
```

```
 miap.mScreen.setx_value_ratio
   (miap.mScreen.V_IMP_X);
 miap.mScreen.sety_value_ratio
   (miap.mScreen.V_IMP);
 miap.mScreen.showbuffer                                          110
   (miap.mScreen.V_IMP_X,miap.mScreen.V_IMP);
 startButton.enable();
 stopButton.disable();
 beingrunned=false;
}



Panel Paneladded(String name,
                 TextField textField) {                           120
 Panel nPanel=new Panel();
 nPanel.setLayout(new GridLayout(0,2));
 nPanel.add(new Label(name) );
 nPanel.add(textField);
 return nPanel;
}

double getValue(TextField textField) {
 double f=0;
 try {                                                            130
   f=Double.valueOf
       (textField.getText()).doubleValue();
 }
 catch(java.lang.NumberFormatException e)
   { nonumbererro=false; }
 return f;
}

boolean nonumbererro=true;
                                                                  140
public boolean action(Event e, Object arg) {
 nonumbererro=true;
 if (e.target==startButton) {
  miap.v_imp_x_begin=getValue(begField);
  miap.v_imp_x_end=getValue(endField);
  miap.v_imp_x_step=getValue(stpField);
  if ( (!beingrunned) && nonumbererro) {
    startButton.disable();
    stopButton.enable();
    beingrunned=true;                                             150
    new Thread(this).start();
  }
 }
 else if (e.target==stopButton) {
   if (beingrunned) {
       beingrunned=false;
       startButton.enable();
       stopButton.disable();
   }
```

```
    }                                                                    160
    return true;
  }

  public boolean handleEvent(Event event) {
    if (event.id == Event.WINDOW_DESTROY) {
      if (inAnApplet) {
          dispose();
      } else {
          System.exit(0);
      }                                                                  170
    }
    return super.handleEvent(event);
  }
}
```

A.3 Screen.java

```
import java.awt.*;
import java.util.*;
import java.applet.Applet;
import java.awt.image.*;
import java.awt.Graphics.*;
import java.io.*;
import java.awt.event.*;
import java.lang.Double.*;

class Screen extends Canvas {                                          10
  iap miap;
  double x_value_ratio;  // set by self judge from t_span
  double y_value_ratio;  // set by user control
  boolean autoshow;
  boolean auto_on;

  static final int T=-1;
  static final int X_TOTAL=0;
  static final int X_FORCED=1;
  static final int X_TRANSIENT=2;                                      20
  static final int V_TOTAL=3;
  static final int V_FORCED=4;
  static final int V_TRANSIENT=5;
  static final int A_TOTAL=6;
  static final int A_FORCED=7;
  static final int A_TRANSIENT=8;
  static final int V_IMP=9;
  static final int V_IMP_X=10;


                                                                       30
  int x_valuetobeshow;
  int y_valuetobeshow;

  Image offImage;
  Graphics offGraphics;
  Dimension offDimension;

  Screen(iap iap,  double iy_value_ratio) {
    miap=iap;
    y_value_ratio=iy_value_ratio;                                      40
    autoshow=false;
    auto_on=true;
    x_valuetobeshow=T;
    y_valuetobeshow=X_TOTAL;
  }

  int getx(int i, int x_valuetobeshow) {
    int zero;
    Dimension d=size();
    if (x_valuetobeshow==T) {                                          50
      zero=d.width/ 10;
```

```
    return zero + (int)(x_value_ratio*miap.tbuffer[i]);
  }
  else if (x_valuetobeshow==X_TOTAL) {
    zero=d.width/2;
    return zero+(int) x_value_ratio*miap.xbuffer[i] );
  }
  else if (x_valuetobeshow==V_IMP_X) {
    zero=d.width/10;
    return zero+(int)      (x_value_ratio * (miap.v_imp_xbuffer[i]- miap.v_imp_x_min) );
  }
  else return 0;
}


int gety(int i, int y_valuetobeshow) {
  //  type=0:total response,
  //      =1:forced response,
  //      =2:transient response

  Dimension d=size();
  int zero=d.height/2;
  int y=0;
  if(y_valuetobeshow==X_TOTAL)
    y= zero-(int) (y_value_ratio*miap.xbuffer[i]);
  else if(y_valuetobeshow==X_FORCED)
    y= zero-(int) (y_value_ratio*miap.x_forbuffer[i]);
  else if(y_valuetobeshow==X_TRANSIENT)
    y= zero-(int) (y_value_ratio*(miap.xbuffer[i]-miap.x_forbuffer[i]) );

  else if(y_valuetobeshow==V_TOTAL)
    y= zero-(int) (y_value_ratio*miap.vbuffer[i]);
  else if(y_valuetobeshow==V_FORCED)
    y= zero-(int)(y_value_ratio*
                  (miap.x_forbuffer[i+1]-miap.x_forbuffer[i-1])/2.0/miap.mImpact_sys.h);
  else if(y_valuetobeshow==V_TRANSIENT)
    y=zero-(int) (y_value_ratio*
                  ( miap.vbuffer[i]-(miap.x_forbuffer[i+1]-
                                     miap.x_forbuffer[i-1])/2.0/miap.mImpact_sys.h) );

  else if(y_valuetobeshow==A_TOTAL)
    y= zero-(int) (y_value_ratio*miap.abuffer[i]);
  else if(y_valuetobeshow==A_FORCED)
    y= zero-(int)
       (y_value_ratio*(-miap.mImpact_sys.F*miap.mImpact_sys.F)*miap.x_forbuffer[i]);
  else if(y_valuetobeshow==A_TRANSIENT)
    y= zero-(int)(y_value_ratio*( miap.abuffer[i]-
                                  (-miap.mImpact_sys.F*miap.mImpact_sys.F)
                                  *miap.x_forbuffer[i]) );

  else if(y_valuetobeshow==V_IMP) {
    zero=d.height*9 / 10;
    return zero - (int) (y_value_ratio*(miap.v_impbuffer[i]-miap.v_imp_min) );
  }
```

```
    if (y<0) y=-1;
    if (y>d.height) y=d.height;
    return y;
  }

  void setx_value_ratio(int x_valuetobeshow) {
    Dimension d=size();
    if (x_valuetobeshow==T) x_value_ratio=0.8*d.width/miap.t_span;
    else if (x_valuetobeshow==X_TOTAL)
      x_value_ratio=0.6*d.width/2/( (miap.x_max>-miap.x_min)?miap.x_max:-miap.x_min );
    else if (x_valuetobeshow==V_IMP_X)
      x_value_ratio=0.8*d.width/(miap.v_imp_x_max-miap.v_imp_x_min);
  }

  void sety_value_ratio(int y_valuetobeshow) {
    Dimension d=size();
    if (y_valuetobeshow==X_TOTAL)
      y_value_ratio=0.8*d.height/2/ ( (miap.x_max>-miap.x_min)?miap.x_max:-miap.x_min );
    else if (y_valuetobeshow==V_TOTAL)
      y_value_ratio=0.8*d.height/2/( (miap.v_max>-miap.v_min)?miap.v_max:-miap.v_min );
    else if (y_valuetobeshow==A_TOTAL)
      y_value_ratio=0.8*d.height/2/( (miap.a_max>-miap.a_min)?miap.a_max:-miap.a_min );
    else if (y_valuetobeshow==V_IMP)
      y_value_ratio=0.8*d.height/( miap.v_imp_max-miap.v_imp_min );
    miap.mAdjustPanel.slider.setValue(0);
    //  change slider position
  }

  void showbuffer(int x_valuetobeshow, int y_valuetobeshow) {

    this.x_valuetobeshow=x_valuetobeshow;
    this.y_valuetobeshow=y_valuetobeshow;
    if (auto_on) {
      setx_value_ratio(x_valuetobeshow);
      sety_value_ratio(y_valuetobeshow);
    }
    repaint();
  }

  public void update ( Graphics g) {
    boolean total=miap.mNotePanel.totalbox.getState();
    boolean forced=miap.mNotePanel.forcedbox.getState();
    boolean transt=miap.mNotePanel.transientbox.getState();
    boolean excitation=miap.mNotePanel.excitationbox.getState();

    Dimension d=size();
    int zero=d.height/2;

    if (offGraphics==null || d.width!=offDimension.width || d.height!=offDimension.height) {
      offDimension=d;
      offImage = createImage(d.width, d.height);
      offGraphics = offImage.getGraphics();
    }
```

```
offGraphics.clearRect(0,0,d.width-1,d.height-1);
offGraphics.setColor(Color.white);
offGraphics.drawLine(d.width/10,d.height/2,d.width*9/10,d.height/2);

int depth=8;
int gy=d.height/2-(int)(miap.mImpact_sys.gap*y_value_ratio+depth);
if (y_valuetobeshow==X_TOTAL && gy<d.height&&gy>0) {
  offGraphics.setColor(Color.blue);
  offGraphics.fillRect(d.width/10,gy, d.width*8/10,depth);
}

if (x_valuetobeshow==T) {
  for (int i=1; i<miap.buffersize-2; i++) {
      if (total) {
        offGraphics.setColor(Color.black);
        offGraphics.drawLine(getx(i,T),gety(i,y_valuetobeshow),
          getx(i+1,T),gety(i+1,y_valuetobeshow));

      }
      if (forced) {
        offGraphics.setColor(Color.red);
        offGraphics.drawLine(getx(i,T),gety(i,y_valuetobeshow+1),
                             getx(i+1,T),gety(i+1,y_valuetobeshow+1));
      }
      if (transt) {
        offGraphics.setColor(Color.yellow);
        offGraphics.drawLine(getx(i,T),gety(i,y_valuetobeshow+2),
                             getx(i+1,T),gety(i+1,y_valuetobeshow+2) );
      }
      if (excitation) {
        offGraphics.setColor(Color.green);
        offGraphics.drawLine(getx(i,T),zero-(int)(miap.excitationbuffer[i]*zero*0.618 ),
                             getx(i+1,T),
                             zero-(int)( miap.excitationbuffer[i+1]*zero*0.618) );
      }
  }
}

else if (x_valuetobeshow==X_TOTAL) {
  for (int i=1; i<miap.buffersize-2; i++) {
  offGraphics.setColor(Color.black);
  offGraphics.drawLine(getx(i,X_TOTAL),gety(i,y_valuetobeshow),
                       getx(i+1,X_TOTAL),gety(i+1,y_valuetobeshow));
  }
}

else if (x_valuetobeshow==V_IMP_X) {
  for (int i=0; i<miap.v_buffersize-1; i++) {
  offGraphics.setColor(Color.black);
  offGraphics.drawLine(getx(i,V_IMP_X),gety(i,V_IMP),
                       getx(i+1,V_IMP_X),gety(i+1,V_IMP));
  offGraphics.drawRect(getx(i,V_IMP_X)-2,gety(i,V_IMP)-2,4,4);
  offGraphics.drawRect(getx(i+1,V_IMP_X)-2, gety(i+1,V_IMP)-2,4,4);
  }
```

```
    }

    double max=0,min=0;
    if (y_valuetobeshow==X_TOTAL) {
      max=miap.x_max;
      min=miap.x_min;
    }                                                                    220
    else  if (y_valuetobeshow==V_TOTAL) {
      max=miap.v_max;
      min=miap.v_min;
    }
    else if (y_valuetobeshow==A_TOTAL) {
      max=miap.a_max;
      min=miap.a_min;
    }
    else if (y_valuetobeshow==V_IMP) {
      max=miap.v_imp_max;                                                230
      min=miap.v_imp_min;
    }

    offGraphics.setColor(Color.black);
    offGraphics.drawString("max="+String.valueOf(max),50,50);
    offGraphics.drawString("min="+String.valueOf(min),50,d.height-50);
    g.drawImage( offImage,0,0,null);

  }
                                                                         240
  public void paint( Graphics g) {
    update(g);
  }
}
```

A.4 SaveWindow.java

```
import java.awt.*;
import java.util.*;
import java.applet.Applet;
import java.awt.image.*;
import java.awt.Graphics.*;
import java.io.*;
import java.awt.event.*;
import java.lang.Double.*;

class SaveWindow extends Frame {                                    10
  iap miap;
  boolean inAnApplet=true;
  TextField titleField;
  TextField desField;
  Button saveButton;

  public SaveWindow(iap iap) {
    miap=iap;
    titleField=new TextField(" ", 20);
    desField=new TextField(" ",20);                                 20
    saveButton=new Button("save");
    setLayout(new GridLayout(3,0));
    add(titleField);
    add(desField);
    add(saveButton);
    setTitle("save window");
    pack();
    show();
  }
                                                                    30
   public boolean handleEvent(Event event) {
    if (event.id == Event.WINDOW_DESTROY) {
      if (inAnApplet) {
          dispose();
      } else {
          System.exit(0);
      }
    }
    return super.handleEvent(event);
  }                                                                 40

  String varyingitem;
  double constant1;
  double constant2;

  public boolean action(Event e, Object arg) {
    if (e.target instanceof Button) {
      File outputFile;
      try {
      outputFile=new File("data/"+titleField.getText().trim());     50
      if  ( outputFile.exists() ) return true;
```

```
saveButton.disable();
FileOutputStream fos=new FileOutputStream(outputFile);
fos.write(String.valueOf(miap.mScreen.x_valuetobeshow).getBytes());
fos.write(' ');
fos.write(String.valueOf(miap.mScreen.y_valuetobeshow).getBytes());
fos.write('\n');
fos.write(String.valueOf(miap.mImpact_sys.h).getBytes());
fos.write(' ');
fos.write(String.valueOf(miap.mShowingPanel.getValue()).getBytes());                    60
fos.write('\n');


// save the data file head.
if (miap.mScreen.x_valuetobeshow==miap.mScreen.T ||
      miap.mScreen.x_valuetobeshow==miap.mScreen.X_TOTAL) {

    fos.write(String.valueOf(miap.t_begin-miap.t_span+miap.tbuffer[0]).getBytes());
    fos.write(' ');
    fos.write(String.valueOf(miap.xbuffer[0]).getBytes());                              70
    fos.write(' ');
    fos.write(String.valueOf(miap.vbuffer[0]).getBytes());
    fos.write('\n');
    fos.write(String.valueOf(miap.mImpact_sys.A).getBytes());
    fos.write(' ');
    fos.write(String.valueOf(miap.mImpact_sys.F).getBytes());
    fos.write(' ');
    fos.write(String.valueOf(miap.mImpact_sys.faye).getBytes());
    fos.write('\n');
    fos.write(String.valueOf(miap.mImpact_sys.gap).getBytes());                         80
    fos.write(' ');
    fos.write(String.valueOf(miap.mImpact_sys.e).getBytes());
    fos.write('\n');
    fos.write(String.valueOf(miap.buffersize).getBytes());
    fos.write('\n');
}

else if (miap.mScreen.x_valuetobeshow==miap.mScreen.V_IMP_X) {
    if (miap.varyingitem==miap.OMEGA) {
      varyingitem="true";                                                               90
      // true=A constanted
      constant1=miap.mForcingPanel.forcingAccelerationPanel.getValue();
      constant2=miap.mImpact_sys.e;
    }
    else if (miap.varyingitem==miap.AMPLITUDE) {
      varyingitem="false"; // false=W constanted
      constant1=miap.mForcingPanel.forcingFrequencyPanel.getValue();
      constant2=miap.mImpact_sys.e;
    }
    else {                                                                              100
      varyingitem="faltue"; // faltue=e constanted
      constant1=miap.mForcingPanel.forcingAccelerationPanel.getValue();
      constant2=miap.mForcingPanel.forcingFrequencyPanel.getValue();
    }
    fos.write(varyingitem.getBytes());
```

```
        fos.write('\n');
        fos.write(String.valueOf(constant1).getBytes());
        fos.write('\n');
        fos.write(String.valueOf(miap.v_imp_x_begin).getBytes());
        fos.write(' ');                                                    110
        fos.write(String.valueOf(miap.v_imp_x_end).getBytes());
        fos.write(' ');
        fos.write(String.valueOf(miap.v_imp_x_step).getBytes());
        fos.write('\n');
        fos.write(String.valueOf(miap.mImpact_sys.gap).getBytes());
        fos.write(' ');
        fos.write(String.valueOf(constant2).getBytes());
        fos.write('\n');
        fos.write(String.valueOf(miap.v_buffersize).getBytes());
        fos.write(' ');                                                    120
        fos.write(String.valueOf(miap.v_x0buffersize).getBytes());
        fos.write(' ');
        fos.write(String.valueOf(miap.vave_buffersize).getBytes());
        fos.write('\n');
    }

    fos.write(desField.getText().getBytes());
    fos.write('\n');
    fos.write("HEAD_END".getBytes());
    fos.write('\n');                                                       130


    // save the data file body.
    if (miap.mScreen.x_valuetobeshow==miap.mScreen.T) {
        for (int i=0; i<miap.buffersize; i++) {
          fos.write(String.valueOf(new Double(miap.tbuffer[i]).floatValue()).getBytes());
          fos.write(' ');
          fos.write(String.valueOf(new Double(miap.xbuffer[i]).floatValue()).getBytes());
          fos.write(' ');
          fos.write(String.valueOf(new Double(miap.vbuffer[i]).floatValue()).getBytes());  140
          fos.write(' ');
          fos.write(String.valueOf(new Double(miap.abuffer[i]).floatValue()).getBytes());
          fos.write('\n');
        }
    }

    if (miap.mScreen.x_valuetobeshow==miap.mScreen.V_IMP_X ||
          miap.mScreen.x_valuetobeshow==miap.mScreen.V_IMP) {

        for (int i=0; i<miap.v_buffersize; i++) {                          150
          fos.write(String.valueOf
                    (new Double(miap.v_imp_xbuffer[i]).floatValue()).getBytes());
          fos.write(' ');
          fos.write(String.valueOf
                    (new Double(miap.v_impbuffer[i]).floatValue()).getBytes());
          fos.write('\n');
        }

        for (int i=0; i<miap.v_x0buffersize; i++) {
```

```
            fos.write(String.valueOf                                                    160
                        (new Double(miap.v_x0_xbuffer[i]).floatValue()).getBytes());
            fos.write(' ');
            fos.write(String.valueOf
                        (new Double(miap.v_x0buffer[i]).floatValue()).getBytes());
            fos.write('\n');
          }

          for (int i=0; i<miap.vave_buffersize; i++) {
            fos.write(String.valueOf
                        (new Double (miap.v_impave_xbuffer[i]).floatValue()).getBytes());   170
            fos.write(' ');
            fos.write(String.valueOf
                        (new Double (miap.v_impavebuffer[i]).floatValue()).getBytes());
            fos.write('\n');
          }
      }

      fos.close();
      } catch (FileNotFoundException ee) {
          System.err.println("FileStreamsTest:" +ee);                                   180
      } catch (IOException ee) {
          System.err.println("FileStreamsTest:  "+ee);
      }
      saveButton.enable();
      return true;
    }
    return false;
  }

}                                                                                       190
```

A.5 iap.java

```
import java.awt.*;
import java.util.*;
import java.applet.Applet;
import java.awt.image.*;
import java.awt.Graphics.*;
import java.io.*;
import java.awt.event.*;
import java.lang.Double.*;

public class iap extends Applet {                                    10

  NotePanel mNotePanel;
  ForcingPanel mForcingPanel;
  SystemPanel mSystemPanel;
  ShowingPanel mShowingPanel;
  AdjustPanel mAdjustPanel;
  Screen mScreen;
  Impact_sys mImpact_sys;
  AutoShowThread mAutoShowThread;
  V_impFrame mV_impFrame;                                            20

  //*************************************
  //   <<<Data buffer area>>>
  double t_begin,t_span;
  double[] tbuffer;  //tthe first one is 0
  double[] xbuffer;
  double[] vbuffer;
  double[] abuffer;
  double[] x_forbuffer;
  double[] excitationbuffer;                                         30
  double[] v_impbuffer;
  double[] v_imp_xbuffer;
  double[] v_impavebuffer;
  double[] v_impave_xbuffer;
  double[] v_x0buffer;
  double[] v_x0_xbuffer;
  int varyingitem=1;
  static final int  AMPLITUDE=1;
  static final int OMEGA=2;
  static final int RESTITUTION=3;                                    40
  double v_imp_x_begin=0.5;
  double v_imp_x_step=0.1;
  double v_imp_x_end=2;
  double v_imp_x;

  int buffersize;
  int v_buffersize;
  int vave_buffersize;
  int v_x0buffersize;
                                                                     50
  double x_max=0,x_min=0;
```

```
double v_max=0,v_min=0;
double a_max=0,a_min=0;
double v_imp_max=-1e99,v_imp_min=1e99;
double v_imp_x_max=0,v_imp_x_min=0;
//*****************************************

public void init() {

  Panel mbottomPanel=new Panel();                                          60
  mForcingPanel=new ForcingPanel(this);
  mSystemPanel= new SystemPanel(this);

  mbottomPanel.setLayout(new GridLayout(1,0) );
  mbottomPanel.add(mForcingPanel);
  mbottomPanel.add(mSystemPanel);

  mNotePanel=new NotePanel(this);

  mShowingPanel=new ShowingPanel(this);                                    70
  mAdjustPanel=new AdjustPanel(this);
  mScreen=new Screen(this,1.0);
  setLayout(new BorderLayout());
  add("East", mNotePanel);
  add("South", mbottomPanel);
  add("North", mShowingPanel);
  add("Center",mScreen);
  add("West",mAdjustPanel);
  validate();
                                                                           80
  t_begin=0.0;
  tbuffer=new double[22000];
  xbuffer=new double[22000];
  vbuffer=new double[22000];
  abuffer=new double[22000];
  x_forbuffer=new double[22000];
  excitationbuffer=new double[22000];
  v_impbuffer=new double[20000];
  v_impavebuffer=new double[10000];
  v_impave_xbuffer=new double[10000];                                      90
  v_imp_xbuffer=new double[20000];
  v_x0buffer=new double[20000];
  v_x0_xbuffer=new double[20000];

  mImpact_sys=new Impact_sys(this,
                             mForcingPanel.forcingAccelerationPanel.getValue(),
                             mForcingPanel.forcingFrequencyPanel.getValue(),
                             1,
                             mSystemPanel.bPanel.getValue(),
                             1,                                            100
                             mSystemPanel.gapPanel.getValue(),
                             mSystemPanel.ePanel.getValue() );
  t_span=1.0/mForcingPanel.forcingFrequencyPanel.getValue()
    *mShowingPanel.getValue()*2*Math.PI;
}
```

```
 public static void main(String args[]) {
  // Create the applet object.
  iap miap = new iap();
                                                                    110
  // Create the main window with the applet embedded in it.
  new MainWindow(miap, 1024, 512);
 }

}

class MainWindow extends Frame {
  // MainWindow constructor.MainWindow(iap iap, int iWidth, int iHeight) {
       // Set the size of the frame, and its title.
       setSize(iWidth, iHeight);                                    120
       setTitle(iap.getClass().getName());

       // Enable window events.
       enableEvents(AWTEvent.WINDOW_EVENT_MASK);

       // Add the applet to the frame center.
       add("Center", iap);

       // Initialize the applet.
       iap.init();                                                  130

       // Make the frame visible.
       show();

       // Start the applet.
       iap.start();
  }

 // Handle window events received by the frame.
 public void processWindowEvent(WindowEvent event) {               140
    if (event.getID() == WindowEvent.WINDOW_CLOSING) {
       System.exit(0);
  }
 }
}

class NotePanel extends Panel{
 iap miap;
 Checkbox impactonlybox;
 Checkbox totalbox;                                                 150
 Checkbox forcedbox;
 Checkbox transientbox;
 Checkbox excitationbox;
 Label zeroLabel;
 Label strikerLabel;
 Button saveButton;

 NotePanel(iap iap) {
  miap=iap;
```

```
  setLayout( new GridLayout(6,0) );

  impactonlybox=new Checkbox("show Impact only");
  impactonlybox.setState(false);
  add(impactonlybox);

  totalbox=new Checkbox("black :total     ");
  totalbox.setState(true);
  add(totalbox);

  forcedbox=new Checkbox("red    :forced    ");
  forcedbox.setState(false);
  add(forcedbox);

  transientbox=new Checkbox("yellow:transient");
  transientbox.setState(false);
  add(transientbox);

  excitationbox=new Checkbox("green:excitation");
  excitationbox.setState(false);
  add(excitationbox);


  saveButton=new Button("save data");
  saveButton.enable();
  add(saveButton);

 }

 public boolean action(Event e, Object arg) {
  if (e.target instanceof Checkbox) {
    if (e.target==impactonlybox) {
         if (!impactonlybox.getState() ) {
           miap.mShowingPanel.setValue(6);
           miap.t_span=1.0/miap.mForcingPanel.forcingFrequencyPanel.getValue()
   *miap.mShowingPanel.getValue()*2*Math.PI;

         }
    }
    else {
         miap.mScreen.showbuffer
         (miap.mScreen.x_valuetobeshow,miap.mScreen.y_valuetobeshow);
    }
    return true;
  }

  else if (e.target instanceof Button) {
    new SaveWindow(miap);
    return true;
  }
  return false;
 }

}
```

```
class AutoShowThread extends Thread {
 iap miap;
 AutoShowThread(iap iap) {
  miap=iap;                                                          220
  setPriority(Thread.MIN_PRIORITY);
 }

 public void run() {
  for (;;) {
   miap.mImpact_sys.fillbuffer();
   miap.mScreen.showbuffer
   (miap.mScreen.x_valuetobeshow,miap.mScreen.y_valuetobeshow);
   try {
        sleep(50);                                                   230
   } catch (InterruptedException e) {}
  }
 }
}

class AdjustPanel extends Panel {

 iap miap;
 Scrollbar slider;
 int min=-50;                                                        240
 int max=50;
 int oldvalue;

 AdjustPanel(iap iap) {
  miap=iap;
  setLayout(new GridLayout(1,1));
  slider=new Scrollbar(Scrollbar.VERTICAL,0,(max-min)/10,min,max);
  add(slider);
  oldvalue=0;
 }                                                                   250

  public boolean handleEvent(Event e) {
   if (e.target instanceof Scrollbar) {
    miap.mShowingPanel.autoScaleChecker.setState(false);
    miap.mScreen.auto_on=false;
    int value=slider.getValue();
    if (value<oldvalue) {
        oldvalue=value;
        miap.mScreen.y_value_ratio*=1.2;
   }                                                                 260
    else {
        miap.mScreen.y_value_ratio/=1.2;
        oldvalue=value;
    }
    miap.mScreen.showbuffer
     (miap.mScreen.x_valuetobeshow,miap.mScreen.y_valuetobeshow);
  }
```

```
    return super.handleEvent(e);
  }
}

class ForcingParameterPanel extends Panel {
  TextField textField;
  Scrollbar slider;
  Label label;
  int min = 500;
  int max=2000;
  iap miap;
  ForcingPanel mForcingPanel;
  static final String AMP="Amplitude";
  static final String FRE="Angular frequency";
  ForcingParameterPanel(iap iap, ForcingPanel forcingPanel,String myTitle) {
    miap=iap;
    mForcingPanel=forcingPanel;
    setLayout(new  GridLayout(3,0));
    label=new Label(myTitle, Label.CENTER);
    add(label);
    textField=new TextField("1",10);
    add(textField);
    slider=new Scrollbar(Scrollbar.HORIZONTAL, 1000, (max-min)/15,min,max);
    add(slider);
  }

  public void paint(Graphics g) {
    Dimension d=size();
    g.drawRect(0,0,d.width-1,d.height-1);
  }

  public Insets insets() {
    return new Insets(5,5,5,8);
  }

  double getValue() {
    double f;
    try {
      f=Double.valueOf(textField.getText()).doubleValue();
    } catch (java.lang.NumberFormatException e) {
      f=0.0;
    }
    return f;
  }

  public boolean action (Event e, Object arg) {
    if (e.target instanceof TextField) {
      setSliderValue(getValue());
      if (label.getText()==AMP) miap.mImpact_sys.A=getValue();
      else if  (label.getText()==FRE) miap.mImpact_sys.setF_related(getValue() );
      return true;
    }
```

```
    return false;
  }

  public boolean handleEvent(Event e) {
    if (e.target instanceof Scrollbar) {
      textField.setText(String.valueOf(slider.getValue()/1000.0));
      if (label.getText()=="Amplitude")       miap.mImpact_sys.A=getValue();
      else if  (label.getText()=="Angular frequency") {
           miap.mImpact_sys.setF_related(getValue() );                    330
      }
    }
    return super.handleEvent(e);
  }

  void setValue(double f) {
    setSliderValue(f);
    textField.setText(String.valueOf(f));
  }
                                                                          340
  void setSliderValue(double f) {
    int sliderValue=(int)(f*1000);
    if (sliderValue>max) sliderValue=max;
    if (sliderValue <min) sliderValue=min;
    slider.setValue(sliderValue);
  }
}

class ForcingPanel extends Panel {
  iap miap;                                                               350
  ForcingParameterPanel forcingAccelerationPanel;
  ForcingParameterPanel forcingFrequencyPanel;

  ForcingPanel(iap iap) {
    miap=iap;
    GridBagConstraints c=new GridBagConstraints();
    GridBagLayout gridbag=new GridBagLayout();
    setLayout(gridbag);

    c.fill=GridBagConstraints.BOTH;                                       360
    // HORIZONTAL;

    Label label=new Label("Forcing Parameters",Label.CENTER);
    c.gridwidth= GridBagConstraints.REMAINDER;
    gridbag.setConstraints(label,c);
    add(label);

    forcingAccelerationPanel=new ForcingParameterPanel(miap,this,"Amplitude");
    forcingFrequencyPanel=new ForcingParameterPanel(miap,this,"Angular frequency");
                                                                          370
    c.weightx=1.0;
    c.gridwidth=1;
    gridbag.setConstraints(forcingAccelerationPanel,c);
    add(forcingAccelerationPanel);
```

```
    c.gridwidth=GridBagConstraints.REMAINDER;
    gridbag.setConstraints(forcingFrequencyPanel,c);
    add(forcingFrequencyPanel);
  }
}

class SystemParameterPanel extends Panel {
  iap miap;
  SystemPanel mSystemPanel;
  TextField textField;
  String mytitle;
  Label label;

  SystemParameterPanel
  (iap iap,SystemPanel systemPanel, String myTitle, double init_value) {
    setLayout(new  GridLayout(2,0));
    miap=iap;
    mSystemPanel=systemPanel;
    mytitle=myTitle;
    label=new Label(myTitle, Label.CENTER);
    add(label);
    textField=new TextField("1",4);
    add(textField);
    textField.setText(String.valueOf(init_value));
  }

  public void paint(Graphics g) {
    Dimension d=size();
    g.drawRect(0,0,d.width-1,d.height-1);
  }

  public Insets insets() {
    return new Insets(5,5,5,8);
  }

  double getValue() {
    double f;
    try {
      f=Double.valueOf(textField.getText()).doubleValue();
    } catch (java.lang.NumberFormatException e)
      {     f=1.0;    }
  return f;
  }

  void setValue(double f) {
    textField.setText(String.valueOf(f));
  }

  public boolean action(Event e, Object arg) {
    if (e.target instanceof TextField) {
      if (label.getText().equals("b=2a")) miap.mImpact_sys.b=getValue();
      if (label.getText().equals("gap"))  miap.mImpact_sys.gap=getValue();
      if (label.getText().equals("e"))    miap.mImpact_sys.e=getValue();
```

```
      return true;
    }
    return false;
  }
}

class SystemPanel extends Panel {
  iap miap;
  SystemParameterPanel bPanel;
  SystemParameterPanel gapPanel;
  SystemParameterPanel ePanel;
  Checkbox linearChecker;

  SystemPanel(iap iap) {
    miap=iap;
    GridBagConstraints c=new GridBagConstraints();
    GridBagLayout gridbag=new GridBagLayout();
    setLayout(gridbag);

    c.fill=GridBagConstraints.HORIZONTAL;

    Label label=new Label
      ("System Parameters",Label.CENTER);
    c.gridwidth= GridBagConstraints.REMAINDER;
    gridbag.setConstraints(label,c);
    add(label);

    bPanel=new SystemParameterPanel(miap,this,"b=2a",0.1);
    gapPanel=new SystemParameterPanel(miap,this,"gap",1.0);
    ePanel=new SystemParameterPanel(miap,this,"e",0.6);

    c.weightx=1.0;
    c.gridwidth=1;
    gridbag.setConstraints(bPanel,c);
    add(bPanel);

    c.weightx=1.0;
    c.gridwidth=1;
    gridbag.setConstraints(ePanel,c);
    add(ePanel);c.weightx=1.0;

    c.gridwidth=1;
    gridbag.setConstraints(gapPanel,c);
    add(gapPanel);

    linearChecker=new Checkbox("or linear");
    linearChecker.setState(false);
    c.weightx=GridBagConstraints.REMAINDER;
    c.gridwidth=1;
    gridbag.setConstraints(linearChecker,c);
    add(linearChecker);


  }
```

```
}

class ShowingPanel extends Panel {

  iap miap;
  Label equationLabel;
  Choice typeChooser;                                                    490
  Checkbox autoshowChecker;
  Checkbox autoScaleChecker;
  Button stepGenerator;
  TextField textField;

  ShowingPanel(iap iap) {

    miap=iap;

    setLayout(new FlowLayout());                                         500

    equationLabel=new Label
    ("motion:  x''+2a x'+x=A cos(f t) when x<gap, x'(t+)=-e x'(t-) when x=gap");
    add(equationLabel);

    typeChooser=new Choice();
    typeChooser.addItem("displacement");
    typeChooser.addItem("velocity");
    typeChooser.addItem("acceleration");
    typeChooser.addItem("phase diagram");                                510
    typeChooser.addItem("v_imp-w");
    typeChooser.addItem("v_imp-A");
    typeChooser.addItem("v_imp-e");
    add(typeChooser);

    autoshowChecker=new Checkbox("Auto show");
    autoshowChecker.setState(false);
    add(autoshowChecker);

    autoScaleChecker= new Checkbox("Auto scale");                        520
    autoScaleChecker.setState(true);
    add(autoScaleChecker);

    stepGenerator=new Button("step");
    stepGenerator.enable();
    add(stepGenerator);

    textField=new TextField("6",4);
    add(textField);
    add(new Label("periods shown") );                                    530

  }

  double getValue() {
    double f;
    try {
      f=Double.valueOf(textField.getText()).doubleValue();
```

```
  }
  catch (java.lang.NumberFormatException e) {
    f=1.0;
  }
  return f;
}

void setValue(double f) {
   textField.setText(String.valueOf(f));
}

public boolean action(Event e, Object arg) {
  if (e.target instanceof Choice) {
    if (typeChooser.getSelectedItem().equals("displacement")) {
         miap.mScreen.setx_value_ratio(miap.mScreen.T);
         miap.mScreen.sety_value_ratio(miap.mScreen.X_TOTAL);
         miap.mScreen.showbuffer(miap.mScreen.T,miap.mScreen.X_TOTAL);
    }
    else if
         (typeChooser.getSelectedItem().equals("velocity")) {
         miap.mScreen.setx_value_ratio(miap.mScreen.T);
         miap.mScreen.sety_value_ratio(miap.mScreen.V_TOTAL);
         miap.mScreen.showbuffer(miap.mScreen.T,miap.mScreen.V_TOTAL);
    }
    else if
         (typeChooser.getSelectedItem().equals("acceleration")) {
         miap.mScreen.setx_value_ratio(miap.mScreen.T);
         miap.mScreen.sety_value_ratio(miap.mScreen.A_TOTAL);
         miap.mScreen.showbuffer(miap.mScreen.T,miap.mScreen.A_TOTAL);
    }
    else if
         (typeChooser.getSelectedItem().equals("phase diagram")) {
         miap.mScreen.setx_value_ratio(miap.mScreen.X_TOTAL);
         miap.mScreen.sety_value_ratio(miap.mScreen.V_TOTAL);
         miap.mScreen.showbuffer(miap.mScreen.X_TOTAL,miap.mScreen.V_TOTAL);
    }
    else if (typeChooser.getSelectedItem().equals("v_imp-w")) {
         if (miap.mAutoShowThread!=null) {
           autoshowChecker.setState(false);
           miap.mAutoShowThread.stop();
           miap.mAutoShowThread=null;
           stepGenerator.disable();
           miap.mScreen.autoshow=false;
          }
          miap.varyingitem=miap.OMEGA;
          if (true) {
            miap.mV_impFrame=new V_impFrame(miap, miap.OMEGA);
          }

    }
    else if (typeChooser.getSelectedItem().equals("v_imp-A")) {
         if (miap.mAutoShowThread!=null) {
           autoshowChecker.setState(false);
           miap.mAutoShowThread.stop();
```

```
        miap.mAutoShowThread=null;
        stepGenerator.disable();
        miap.mScreen.autoshow=false;
      }
      miap.varyingitem=miap.AMPLITUDE;
      if (true) {
        miap.mV_impFrame=new V_impFrame(miap, miap.AMPLITUDE);
      }
  }
  else if (typeChooser.getSelectedItem().equals("v_imp-e")) {
      if (miap.mAutoShowThread!=null) {
        autoshowChecker.setState(false);
        miap.mAutoShowThread.stop();
        miap.mAutoShowThread=null;
        stepGenerator.disable();
        miap.mScreen.autoshow=false;
      }
      miap.varyingitem=miap.RESTITUTION;
      if (true) {
        miap.mV_impFrame=new V_impFrame(miap, miap.RESTITUTION);
      }
  }
  return true;
}

else if (e.target == autoshowChecker) {
  if (!typeChooser.getSelectedItem().equals("v_imp-w") &&
  !typeChooser.getSelectedItem().equals("v_imp-A")    ) {
      if (autoshowChecker.getState()) {
        stepGenerator.disable();
        miap.mAutoShowThread=new AutoShowThread(miap);
        miap.mAutoShowThread.start();
        miap.mScreen.autoshow=true;
      }
      else if (miap.mAutoShowThread!=null) {
        miap.mAutoShowThread.stop();
        miap.mAutoShowThread=null;
        stepGenerator.enable();
        miap.mScreen.autoshow=false;
      }
  }
  return true;
}

else if (e.target == autoScaleChecker) {
  if (autoScaleChecker.getState()) {
      miap.mScreen.auto_on=true;
      miap.mScreen.showbuffer
      (miap.mScreen.x_valuetobeshow,miap.mScreen.y_valuetobeshow);
  }
  else {
      miap.mScreen.auto_on=false;
  }
}
```

```
    else if (e.target instanceof TextField) {
     if ((!miap.mNotePanel.impactonlybox.getState()) && getValue()>180) setValue(180);
     miap.t_span=1.0/miap.mForcingPanel.forcingFrequencyPanel.getValue()*
     getValue()*2*Math.PI;                                                          650
     return true;
    }

    else if (e.target==stepGenerator) {
       if (!typeChooser.getSelectedItem().equals("v_imp-w") &&
          !typeChooser.getSelectedItem().equals("v_imp-A")    ) {
            miap.mImpact_sys.fillbuffer();
            miap.mScreen.showbuffer
            (miap.mScreen.x_valuetobeshow,miap.mScreen.y_valuetobeshow);
          }                                                                         660
            return true;
          }
    return false;
  }
}
```

Bibliography

References

[1] Akay, A. A review of impact noise. J. Acoust. Soc. Am. 64, 977-987, 1978.

[2] Kojima, N., Ikoma, K. & Fukuda, M. Estimation of noise emitted by vibration of a plate. Bulletin of JSME. 24, 1233-1238, 1981.

[3] Richards, E.J. On the prediction of impact noise, III: energy accountancy in inductrial machines. J. Sound and Vibration 76, 187-232, 1981.

[4] Igarashi, T. & Aimoto, T. Studies on impact sound. Bulletin of JSME. 28, 1247-1254, 1985.

[5] Thompson, M.G., Bishop, S.R., & Foale, S. An experimental study of low velocity impacts. Machine Vibration, Springer-Verlag, London, 10-17, 1994.

[6] Peterka, F. Dynamics of the impact oscillator. IUTAM Symposium CHAOS'97: Applications of Nonlinear and Chaotic Dynamics in Mechanics. Ithaca, New York, 1997.

[7] Nordmark, A.B. Non-periodic motion caused by grazing incidence in an impact oscillator. J. Sound and Vibration. 145, 279-297, 1991.

[9] Shaw, S.W., & Holmes, P.J. A periodically forced piecewise linear oscillator. J. Sound and Vibration. 90, 129-155, 1983.

[10] Hu, H.Y. Detection of grazing orbits and incident bifurcations of a forced continuous, piecewise-linear oscillator. J. Sound and Vibration. 187, 485-493, 1995.

[11] Clough, R. and Penzien, J. Dynamics of Structures. McGraw-Hill, New York, 1975

[12] Leissa, A.W. Vibration of Plates. NASA-Sp-160. 1969.

[13] Bathe, K.J. Finite Element Procedures. 1996.

[14] ADINA R & D, Inc., ADINA Users Guide. 1996.

[15] Crandall S.H., Karnopp D.C., Kurtz E.F., Prodmore-Brown D.C. Dynamics of Mechanical and Electromechanical Systems. 1968.

[16]Strogatz S.H. Nonlinear Dynamics and Chaos. 1994.

[17]Mary Campione, Kathy Walrath, The Online Java Tutorial http://java.sun.com/docs/books/tutorial/. 1997.

6716-45